5.4. При использовании метода вычисления моментов было найдено управление в непрерывном времени, в то время как из-за использования цифровой машины при реализации управления манипулятором робота оно должно осуществляться в дискретном времени, т. е. с помощью системы с дискретными параметрами. Приведите условие, при котором возможна указанная реализация управления.

5.5. Уравнения движения двухзвенного манипулятора робота (разд. 3.2.6) записываются в компактной матрично-векторной форме

$$\begin{bmatrix} d_{11}(\theta_2) & d_{12}(\theta_2) \\ d_{12}(\theta_2) & d_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1(t) \\ \ddot{\theta}_2(t) \end{bmatrix} +$$

$$+ \left[\beta_{12}(\theta_2)\dot{\theta}_2^2 + 2\beta_{12}(\theta_2)\dot{\theta}_1\dot{\theta}_2 - \beta_{12}(\theta_2)\dot{\theta}_1^2\right] + \begin{bmatrix} c_1(\theta_1, \theta_2)\, g \\ c_2(\theta_1 \theta_2)\, g \end{bmatrix} = \begin{bmatrix} \tau_1(t) \\ \tau_2(t) \end{bmatrix},$$

где g — ускорение свободного падения. Определите соответствующие векторы переменного состояния $\mathbf{x}(t)$ и вектор управления $\mathbf{u}(t)$ для этой динамической системы. Предполагая, что существует $\mathbf{D}^{-1}(\Theta)$, запишите уравнения движения такого робота для параметров d_{ij}, β_{ij} и c_i в пространстве состояний с использованием найденных вектора переменных состояния и вектора управления.

5.6. Постройте устройство управления с переменной структурой для робота, описанного в упр. 5.5 (разд. 5.5).

5.7. Постройте устройство нелинейного независимого управления с обратной связью для робота из задачи 5.5 (см. п. 5.6).

5.8. Определите якобиан в базовой системе координат для робота из упр. 5.5 (см. приложение Б).

5.9. Назовите два основных недостатка использования независимого управления движением по скорости.

5.10. Назовите два основных недостатка использования независимого управления движением по ускорению.

5.11. Назовите два основных недостатка использования адаптивного управления по заданной модели.

5.12. Назовите два основных недостатка использования адаптивного управления по возмущению.

Глава 6.
ОЧУВСТВЛЕНИЕ

Не так же ты доверяешь чувствам,
как и тому, что видишь ты?

В. Шекспир

6.1. ВВЕДЕНИЕ

Внешние устройства очувствления позволяют взаимодействовать роботу с внешней средой в интерактивном режиме, который отличается от режима работы по жестко заданной программе, предусматривающей выполнение повторяющихся операций без обратной связи с внешней средой. Хотя последний режим преобладает в современных промышленных роботах, очувствление и высокий уровень технического интеллекта обеспечивает более активное взаимодействие машин с внешней средой, что, несомненно, является перспективной областью развития робототехники. Роботу, который может «видеть» и «чувствовать», легче выполнять сложные задачи, кроме того, в этом случае снижаются требования к точности устройств управления. Очувствленные обучаемые системы обладают возможностью адаптации при выполнении широкого круга задач. Благодаря этому повышается степень универсальности, что в конечном счете приводит к снижению стоимости продукции и технического обслуживания.

Функционально датчики роботов можно подразделить на два основных типа: датчики внутреннего состояния и датчики внешнего состояния. Датчики внутреннего состояния служат для формирования сигналов в цепях обратных связей по положению и скорости звеньев манипулятора [1]. Эти измерения используются при управлении роботом, которое рассматривалось в гл. 5. Датчики внешнего состояния предназначены для измерения параметров в дальней и ближней зонах и для тактильных измерений. Внешнее очувствление, которое рассматривается в гл. 6—8, используется для управления движением робота, а также при идентификации объектов и манипулирования с ними.

Датчики внешнего состояния в свою очередь подразделяются на контактные и бесконтактные. Контактные датчики, как сле-

[1] В последнее время получают развитие комбинированные методы управления, включая очувствление по силе и моменту. — *Прим. ред.*

дует из их названия, производят измерения при контакте с объектом в процессе касания, проскальзывания или кручения. Принцип действия бесконтактных датчиков основан на определении изменений акустического или электромагнитного полей при взаимодействии с объектом. Наиболее важными примерами использования бесконтактных датчиков является измерение положения объекта в дальних и ближних зонах, а также определение характеристик объекта оптическим методом.

Основное внимание в данной главе уделено датчикам измерения в дальней и ближней зонах, а также тактильным и силомоментным датчикам. Вопросы технического зрения подробно рассматриваются в гл. 7 и 8. Отметим, что техническое зрение и датчики измерения в дальней зоне обычно дают основной объем информации при управлении движением манипулятора, а датчики измерения в ближней зоне и тактильные датчики используются на конечных этапах движения манипулятора при захвате объекта. Силомоментные датчики применяются в качестве устройств обратной связи для управления манипулятором после захвата объекта, например для исключения возможности его повреждения или проскальзывания.

6.2. ДАТЧИКИ ИЗМЕРЕНИЯ В ДАЛЬНЕЙ ЗОНЕ

С помощью датчиков измерения в дальней зоне определяется расстояние от точки отсчета, обычно связанной с самим датчиком, до объекта в рабочем диапазоне измерений. Человек оценивает расстояние с помощью зрения (гл. 7), в то время как некоторые животные, например летучие мыши, определяют расстояние по времени посылки и отражения акустического сигнала. Датчики измерения в дальней зоне используются для навигации робота и обхода препятствий, когда требуется оценить расстояния до ближайших объектов или определить местоположение и форму объектов в рабочем пространстве робота. В данном разделе рассматриваются некоторые методы измерения в дальней зоне.

6.2.1. Триангуляция

Одним из простейших методов измерения в дальней зоне является метод триангуляции (рис. 6.1). Объект освещают узким лучом света, направленным на его поверхность. Движение луча в плоскости определяется линией от объекта до приемника света и линией от приемника до источника света. Если приемник расположен на малом участке поверхности, при обнаружении приемником светового пятна расстояние D до освещенного участка поверхности может быть вычислено из геометрических соотношений (рис. 6.1), так как угол, образованный источником

и базовой линией, и расстояние B между источником и приемником известны.

Этот метод реализует точечное измерение. Если система источник — приемник движется в фиксированной плоскости

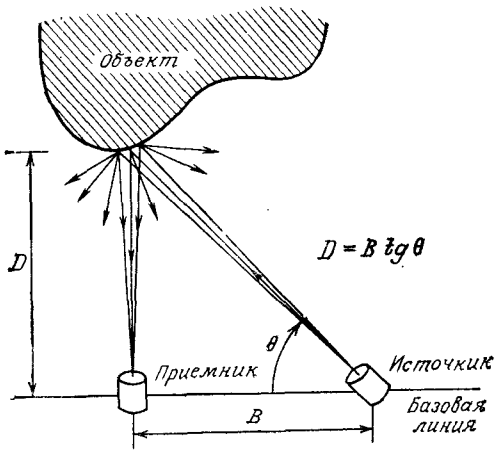


Рис. 6.1. Измерение расстояния методом триангуляции [137].

(вверх и вниз и в стороны в плоскости, перпендикулярной плоскости рис. 6.1 и включающей базовую линию), то в этом случае можно получить группу точек, расстояния от которых до приемника известны. Эти расстояния легко перенести в трехмерную систему координат путем сканирования (рис. 6.2). На рис. 6.2, *а* показано расположение сканируемых объектов, а на рис. 6.2, *б* — результаты сканирования в виде изображения, интенсивность

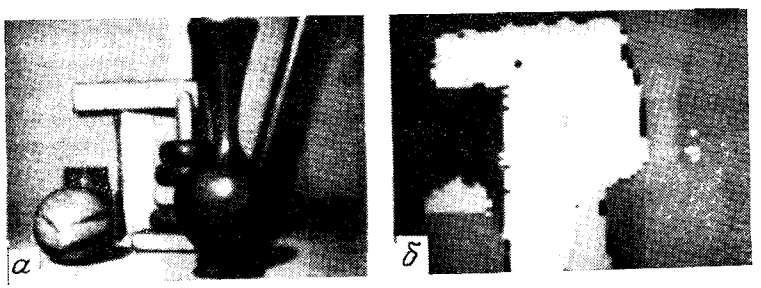
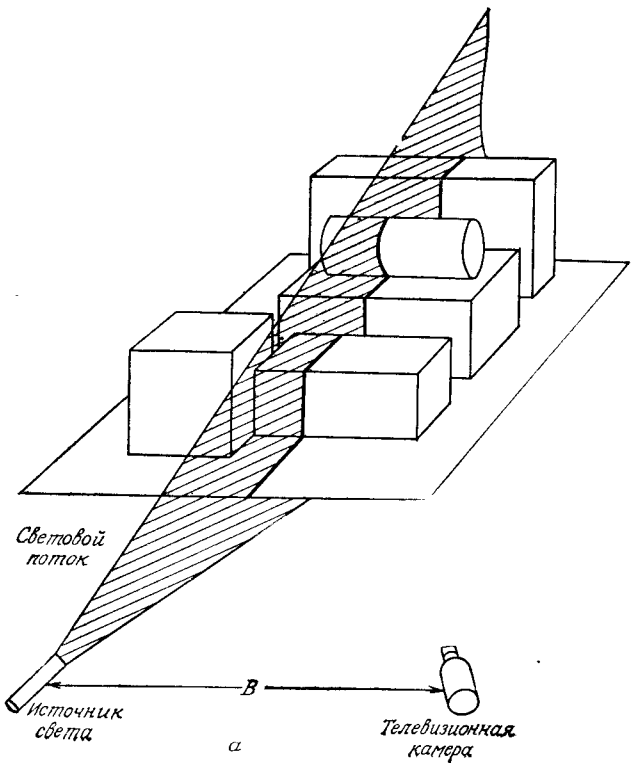


Рис. 6.2. Сканируемые объекты (*а*) и изображение с интенсивностью, пропорциональной расстоянию (*б*) [137].

которого (чем ближе, тем темнее) пропорциональна зоне дальности, измеренной от плоскости движения до пары источник — приемник.

6.2.2. Метод подсветки

Данный метод состоит в проецировании светового потока на группу объектов и использовании изменения формы потока для вычисления расстояния. Одним из наиболее распространенных световых потоков, применяемых в настоящее время, является световая полоса, генерируемая через цилиндрические линзы или узкую щель. Пересечение светового потока с объектами в рабочем пространстве (рис. 6.3) образует на них полосы света, фиксируемые телевизионной камерой, помещенной на расстоянии B от источника света. Такая ситуация легко анализируется компьютером при определении расстояния. Например, отклонение лучей указывает на изменение поверхности, а разрыв соответствует промежутку между поверхностями.



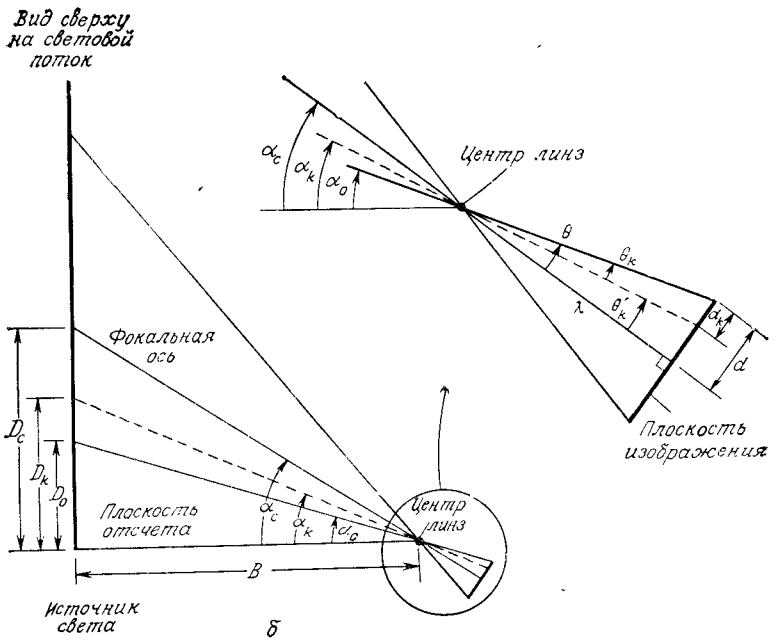


Рис. 6.3. Измерение расстояний методом подсветки. Вид сверху (*б*) на расположение элементов системы, изображенной на рис. *а*, позволяет упростить процесс калибровки.

Для получения базовых значений расстояний вначале проводят калибровку системы. Один из простейших вариантов идентификации показан на рис. 6.3, *б*, который представляет собой вид сверху системы, представленной на рис. 6.3, *а*. При таком расположении источник света и камеру помещают на одинаковой высоте, а световой поток перпендикулярен линии, соединяющей источник света и центр объектива камеры. Назовем вертикальную плоскость, проходящую через эту линию, плоскостью отсчета. Очевидно, что плоскость отсчета перпендикулярна световому потоку и любая вертикальная плоская поверхность, пересекающая луч, дает вертикальную полосу света (рис. 6.3, *а*), каждая точка которой будет равноудалена по нормали от плоскости отсчета.

Целью расположения элементов системы, показанного на рис. 6.3, *б*, является размещение камеры таким образом, чтобы каждая вертикальная полоса была также вертикальной в плоскости изображения Следовательно, каждая точка в одном столбце изображения будет найдена, поскольку находится на известном расстоянии от плоскости отсчета.

В большинстве систем, основанных на методе подсветки, используют цифровые изображения. Предположим, что изображе-

ние, полученное камерой, преобразовано в цифровой массив размерностью $N \times M$ (разд. 7.2) и пусть $y = 0, 1, 2, \ldots, M-1$ является номером столбца этого массива. Ниже показано, что процедура калибровки состоит в измерении расстояния B между источником света и центром линз и последующим измерением углов α_c и α_0. После того как эти величины становятся известны, из правил элементарной геометрии следует, что d (рис. 6.3, *б*) вычисляется по формуле

$$d = \lambda \operatorname{tg} \theta. \tag{6.2-1}$$

где λ — фокальная длина линз и

$$\theta = \alpha_c - \alpha_0. \tag{6.2-2}$$

Для цифрового изображения, содержащего M столбцов, приращение расстояния d_k между столбцами определяется по формуле

$$d_k = k \frac{d}{M/2} = \frac{2kd}{M} \tag{6.2-3}$$

для $0 \leqslant k \leqslant M/2$. (В изображении на мониторе $k = 0$ соответствовало бы крайнему слева столбцу, а $k = M/2$ — центральному столбцу.)

Угол α_k, образованный проекцией произвольной полосы, легко получить, отметив, что

$$\alpha_k = \alpha_c - \theta'_k, \tag{6.2-4}$$

где

$$\operatorname{tg} \theta'_k = \frac{d - d_k}{\lambda}, \tag{6.2-5}$$

или, используя равенство (6.2-3),

$$\theta'_k = \operatorname{arctg}\left[\frac{d(M-2k)}{M\lambda}\right], \tag{6.2-6}$$

где $0 \leqslant k \leqslant M/2$. Для оставшихся значений k (т. е. по другую сторону оптической оси) имеем

$$\alpha_k = \alpha_c + \theta''_k, \tag{6.2-7}$$

где

$$\theta''_k = \operatorname{arctg}\left[\frac{d(2k-M)}{M\lambda}\right] \tag{6.2-8}$$

для $M/2 < k \leqslant (M-1)$.

Сравнивая уравнения (6.2-6) и (6.2-8), отметим, что $\theta''_k = -\theta'_k$. Таким образом, равенства (6.2-4) и (6.2-7) идентичны для всего диапазона $0 \leqslant k \leqslant M-1$. Тогда из рис. 6.3, *б* следует, что расстояние по нормали D_k между произвольной полосой света и плоскостью отсчета будет равно

$$D_k = B \operatorname{tg} \theta_k \tag{6.2-9}$$

для $0 \leqslant k \leqslant M-1$, где α_k вычисляется либо из уравнения (6.2-4), либо из уравнения (6.2-7).

Важно отметить, что, если величины B, α_0, α_c, M и λ известны, номер столбца в цифровом изображении полностью определяет расстояние между плоскостью отсчета и всеми точками на полосе, отображенной на этом столбце. Так как M и λ являются фиксированными параметрами, процедура калибровки состоит в простом измерении B и определении α_c и α_0, как указано выше. Для определения α_c плоскую вертикальную поверхность размещают так, чтобы ее пересечение со световой полосой находилось в центре плоскости изображения (т. е. $y = M/2$). Затем измеряют величину перпендикуляра D_c между поверхностью и плоскостью отсчета. Из рис. 6.3, *б* следует, что

$$\alpha_c = \operatorname{arctg}\left[\frac{D_c}{B}\right]. \tag{6.2-10}$$

Чтобы определить α_0, перемещают поверхность ближе к плоскости отсчета, пока ее световая полоса не совместится с $y = 0$ на плоскости изображения. Затем измеряют D_0 и из рис. 6.3, *б* находят

$$\alpha_0 = \operatorname{arctg}\left[\frac{D_0}{B}\right]. \tag{6.2-11}$$

Это завершает процесс калибровки.

Основное преимущество такой системы состоит в относительной простоте измерения расстояний. После завершения калибровки расстояние, соответствующее каждому столбцу в изображении, вычисляется с помощью уравнения (6.2-9), где $k = 0, 1, 2, \ldots, M-1$, а результаты хранятся в памяти. Затем в процессе измерений расстояние до любой точки изображения получают путем простого определения номера ее столбца в изображении и обращения к соответствующей области памяти.

В заключение отметим, что для решения более общей задачи, когда источник света и камера размещаются произвольно по отношению друг к другу, можно использовать метод, описанный в разд. 7.4. Однако полученные выражения были бы значительно более сложными с точки зрения вычислений.

6.2.3. Измерители расстояния по времени прохождения сигнала

Ниже рассматриваются три метода измерения расстояния, основанные на определении времени прохождения сигнала между объектом и приемником. Два из них используют лазер, а третий — ультразвуковые сигналы.

В первом методе измеряется метод, в течение которого посланный вдоль оси световой импульс возвращается вдоль той же оси от отражающей поверхности. Расстояние до поверхности

определяется по формуле $D = cT/2$, где T — время прохождения сигнала и c — скорость света. Отметим, что поскольку скорость света примерно составляет 0,3 м/нс, электронное оборудование должно обладать частотой отсчета 50 Гц для достижения точности измерения порядка ±6,3 мм.

Система, использующая лазерные импульсы, дает двумерный массив со значениями, пропорциональными расстоянию [137]. Двумерное сканирование выполняется путем отклонения лазерного луча вращающимся зеркалом. Рабочая зона этого устройства находится в пределах 1—4 м с обеспечением точности до ±0,25 см. Пример картины выходного сигнала такой системы дан на рис. 6.4. На рис. 6.4, *а* показан набор трехмерных объ-

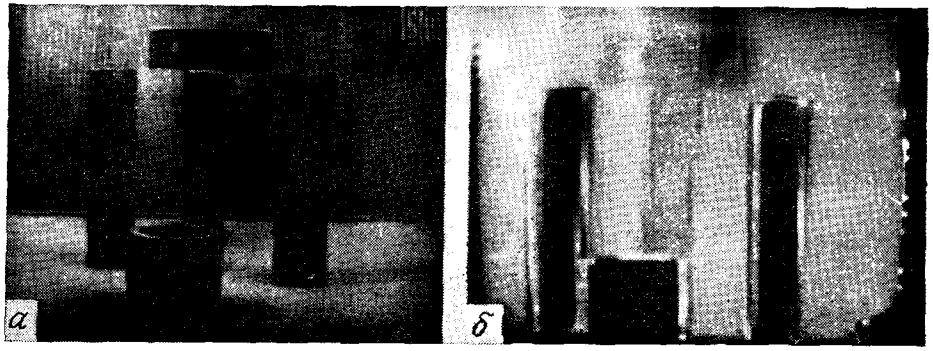


Рис. 6.4. Система объектов (*а*) и изображение (*б*) с интенсивностью, пропорциональной расстоянию [138].

ектов, а на рис. 6.4, *б* — соответствующий информационный массив, представленный в виде изображения, интенсивность каждой точки которого пропорциональна расстоянию между датчиком и отражающей поверхностью в этой точке (чем темнее, тем ближе). Яркие участки вокруг границ объектов представляют прерывистое изменение расстояния, определяемое ЭВМ с помощью специальной процедуры.

Во втором методе вместо импульсного светового сигнала используется непрерывный луч лазера и измеряется задержка (т. е. фазовый сдвиг) между посылаемыми и возвращенными лучами (рис. 6.5). Допустим, что луч лазера с длиной волны λ расщеплен на два луча. Один из них (называемый «лучом отсчета») проходит расстояние L к фазометру, а другой проходит расстояние D до отражающей поверхности. Общее расстояние, пройденное отраженным лучом, составляет $D' = L + 2D$. Допустим, что $D = 0$. При этом условии $D' = L$, и как луч отсчета, так и отраженный луч достигают фазометра одновременно. Если

увеличивать D, отраженный луч проходит большой путь и, следовательно, возникает фазовый сдвиг между двумя лучами в точке измерения (рис. 6.5, *б*). В этом случае имеем

$$D' = L + \frac{\theta}{360}\lambda. \qquad (6.2\text{-}12)$$

Отметим, что, если $\theta = 360°$, обе волны вновь совпадают и нельзя отличить $D' = L$ и $D' = L + n\lambda$, $n = 1, 2, \ldots$, основы-

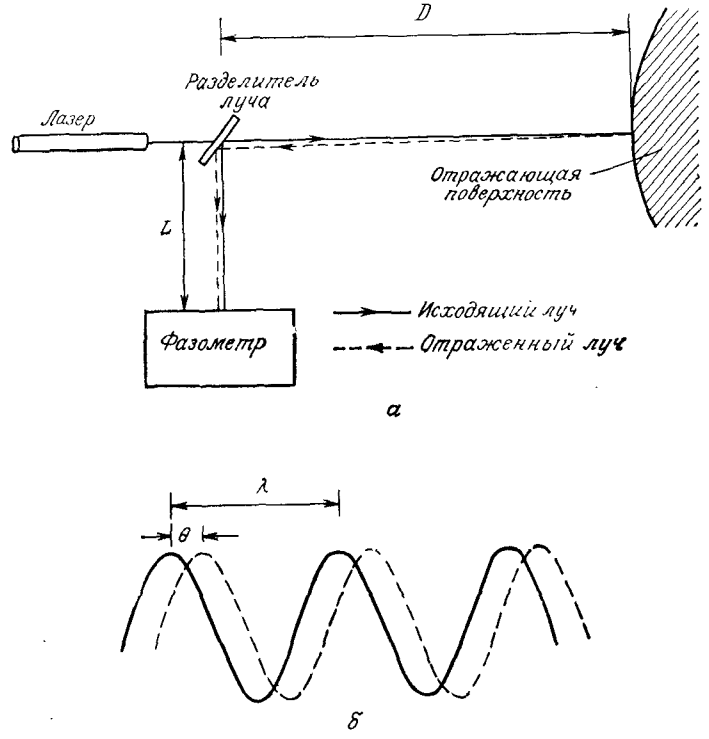


Рис. 6.5. Принцип измерения расстояния по фазовому сдвигу (*а*) и сдвиг между исходящей и отраженной световыми волнами (*б*).

ваясь на одном измерении фазового сдвига. Таким образом, решение может быть получено только в том случае, если $\theta < 360°$, либо, что то же самое, $2D < \lambda$. Так как $D' = L + 2D$, подставив это значение в уравнение (6.2-12), имеем

$$D = \frac{\theta}{360}\left(\frac{\lambda}{2}\right), \qquad (6.2\text{-}13)$$

что определяет расстояние через фазовый сдвиг, если известна длина волны.

Если длина волны лазерного луча мала (например, 632,8 нм для гелий-неонового лазера), то метод, схема которого показана на рис. 6.5, нецелесообразно применять в робототехнике.

Простейшим решением в этом случае является амплитудное моделирование лазерного луча путем использования волны с гораздо большей длиной. (Например, помня, что $c = f\lambda$, модулирующая волна с частотой $f = 10$ МГц имеет длину 30 м.) Этот метод проиллюстрирован на рис. 6.6. Основная процедура остается прежней, но сигнал отсчета является теперь функцией модулирования. Модулированный лазерный сигнал посылается на объект, а возвращенный сигнал демодулируется и сравнивается с отсчетным сигналом для определения фазового сдвига. Равенство (6.2-13) все еще имеет силу, но теперь работа происходит в более удобном диапазоне длин волн.

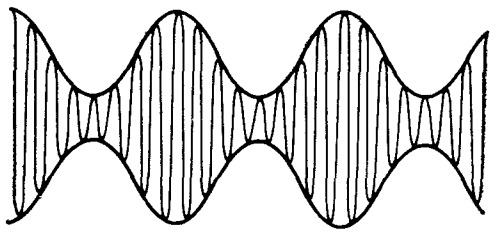


Рис. 6.6. Волновой сигнал, модулированный по амплитуде модулирующей функцией с гораздо большей длиной волны.

Важным преимуществом метода с непрерывным лучом по сравнению с импульсным методом является то, что первый дает сведения об интенсивности (чем ярче, тем ближе) [137]. Однако для работы по методу с непрерывным лучом затрачивается значительно большая мощность. Неточности при измерении расстояний, получаемые при использовании любого из этих методов, требуют усреднения отраженного сигнала для уменьшения ошибки. Если при рассмотрении задачи мы допускаем, что к действительному расстоянию добавляется шум измерения, и принимаем, что измерения являются статистически независимыми, можно показать, что стандартное отклонение от среднего значения равно $1/\sqrt{N}$ стандартных отклонений шума, где N — число усредненных проб. Другими словами, чем дольше мы усредняем, тем меньше неточности при оценке расстояния.

Пример результатов, получаемых с помощью непрерывного модулированного лазерного луча, сканируемого вращающимся зеркалом, показан на рис. 6.7, *б*. На рис. 6.7, *а* дан массив расстояний, представленный как изображение интенсивности (чем ярче, тем ближе). Реальная информация об интенсивности, полученная на соответствующем приборе, показана на рис. 6.7, *б*. Обратите внимание на то, что эти два изображения дополняют друг друга. Например, если возникает трудность при подсчете числа объектов на столе на рис. 6.7, *а*, то это просто сделать по изображению интенсивности. Напротив, невозможно определить расстояние между ближним и дальним краями крышки стола по изображению интенсивности, тогда как это легко сделать с помощью массива расстояний. Способы обработки информации подобного типа даны в разд. 6.7 и 6.8.

Ультразвуковой измеритель расстояний является другим типичным примером реализации метода измерения расстояния

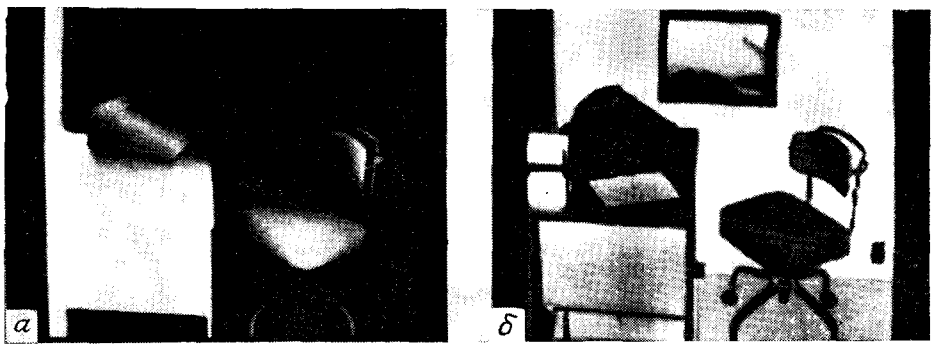


Рис. 6.7. Изображения массива расстояний (*а*) и интенсивности (*б*).

по времени прохождения сигнала. Основная идея аналогична той, что использована в методе с применением лазерных импульсов.

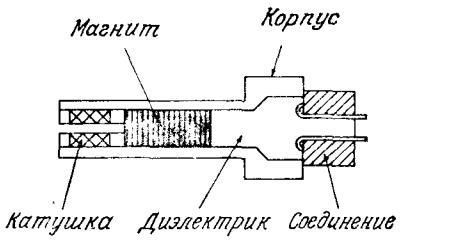
Ультразвуковой сигнал передается за короткий промежуток времени и, так как скорость звука известна для определенной среды, простое вычисление, включающее интервал времени между посылаемым и отраженным сигналами, дает оценку расстояния до отражающей поверхности.

В ультразвуковой измерительной системе, выпускаемой фирмой Polaroid, сигнал длительностью 1 мс, состоящий из 56 импульсов четырех частот (50, 53, 57 и 60 кГц), передается датчиком диаметром ≈ 38 мм. Сигнал, отраженный объектом, улавливается тем же датчиком и, проходя через усилитель и схему индикации, способен измерять расстояние в диапазоне $\sim 0{,}3$—10 м с точностью до 2,5 см. Смешанные частоты в сигнале используются для улучшения устойчивости сигнала. Отклонение в направленности этого прибора составляет $\sim 30°$, что приводит к ограничениям его использования для получения изображения систем, рассмотренных выше в данном разделе.

Это является обычной проблемой при использовании ультразвуковых датчиков, и поэтому они применяются преимущественно в навигации и для обхода препятствий. Конструкции и рабочие характеристики ультразвуковых датчиков обсуждены более подробно в разд. 6.3.

6.3. ОЧУВСТВЛЕНИЕ В БЛИЖНЕЙ ЗОНЕ

Датчики измерения в дальней зоне, описанные нами выше, дают оценку расстояния между датчиком и отражающим объектом. Датчики измерения в ближней зоне обычно имеют дискретный пороговый выходной сигнал, который определяет наличие объекта в пределах установленного пространства. Типичным является использование таких датчиков в робототехнике для получения информации в ближней зоне при захвате объекта или при его обходе. В настоящем разделе рассматриваются несколько основных методов очувствления в ближней зоне и основные рабочие характеристики датчиков.



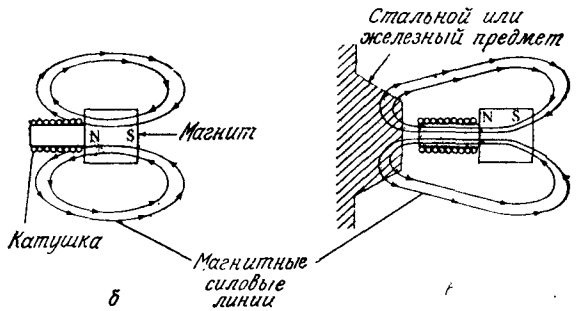
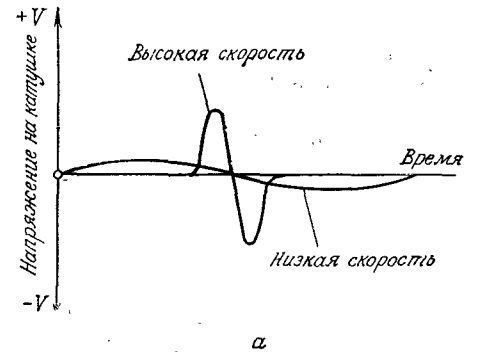


Рис. 6.8. Индуктивный датчик (*а*), форма магнитных линий при отсутствии ферромагнетика (*б*) и форма магнитных линий при наличии ферромагнетика в зоне измерения датчика (*в*) [38].

6.3.1. Индуктивные датчики

Датчики, основанные на изменении индуктивности при взаимодействии с металлическим объектом, находятся в ряду наиболее широко используемых промышленных датчиков измерения в ближней зоне. Принцип работы этих датчиков можно объяснить, используя рис. 6.8 и 6.9. На рис. 6.8, *а* представлена схема индуктивного датчика, который состоит из катушки, размещенной вслед за постоянным магнитом в корпусе.

Когда датчик приближается к ферромагнитному материалу, изменяется расположение силовых линий постоянного магнита, (рис. 6.8, *б* и *в*). При отсутствии движения силовые линии не изменяются и, следовательно, в катушке ток не индуцируется. Однако, если в поле постоянного магнита перемещается ферромагнитный материал, возникающее в результате этого изменение силовых линий индуцирует импульс тока, амплитуда и форма которого пропорциональны уровню изменения магнитного поля.



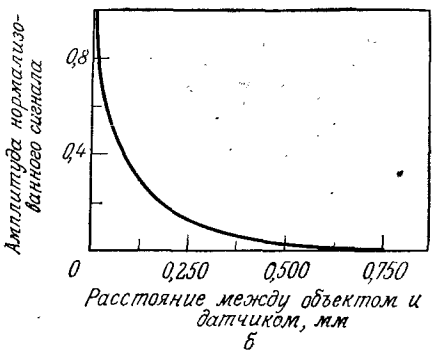


Рис. 6.9. Зависимости выходного сигнала индуктивного датчика от скорости (*а*) и (*б*).

Изменение напряжения на выходе катушки обеспечивает эффективное очувствление в ближней зоне. На рис. 6.9, *а* показано изменение напряжения в катушке в зависимости от скорости, с которой ферромагнетик вводится в поле магнита. Полярность выходного напряжения датчика зависит от того, входит объект в поле или выходит из него. На рис. 6.9, *б* показана зависимость амплитуды напряжения от расстояния между датчиком и объектом. Из этого рисунка видно, что чувствительность резко падает с увеличением расстояния и датчик эффективен только на расстояниях ~ 1 мм.

Так как для получения выходного сигнала на датчике требуется наличие относительного движения датчика и объекта, одним из методов получения дискретного порогового сигнала является интегрирование выходного сигнала. Пороговый сигнал остается на нижнем уровне, пока значение интеграла

остается ниже установленного порога. После превышения порога выходной сигнал переходит на верхний уровень, что соответствует наличию объекта в зоне измерений.

6.3.2. Датчики Холла

Из физики известно, что эффект Холла связывает напряжение между двумя точками в проводнике или полупроводниковом материале с магнитным полем, воздействующим на этот материал. Используемые сами по себе датчики Холла могут уловить только намагниченные объекты. Однако, если их использовать вместе с постоянным магнитом, как показано на рис. 6.10, они способны установить наличие всех ферромагнитных материалов. Используемый таким образом датчик Холла

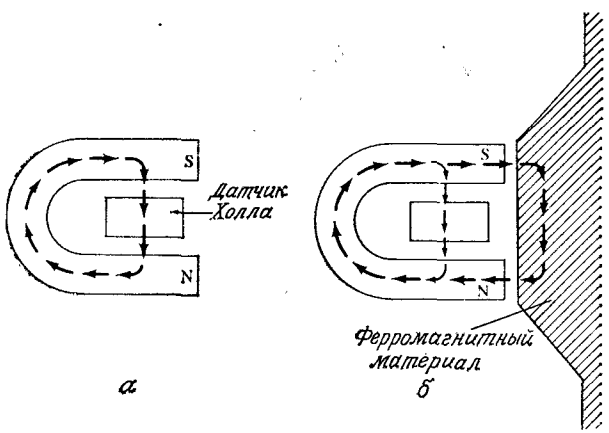


Рис. 6.10. Работа датчика Холла (*а*), снабженного постоянным магнитом (*б*).

«чувствует» сильное магнитное поле при отсутствии ферромагнетика в ближней зоне (рис. 6.10, *а*). Если такой материал вносят в ближнюю зону действия прибора, магнитное поле на датчике ослабевает вследствие замыкания линий поля через материал (рис. 6.10, *б*).

Датчики Холла основаны на возникновении силы Лоренца, действующей на заряженную частицу, движущуюся в магнитном поле. Эта сила направлена по оси, перпендикулярной плоскости, образованной направлением движения заряженной частицы и направлением поля. Таким образом, сила Лоренца определяется как $\mathbf{F} = q(\mathbf{v} \times \mathbf{B})$, где q — заряд, $\mathbf{v}$ — вектор скорости, $\mathbf{B}$ — вектор магнитного поля, а $\times$ — знак пересечения векторов. Предположим, например, что ток проходит через полупроводник *n*-типа, который находится в магнитном поле (рис. 6.11). Помня, что электроны являются основными носителями в материалах *n*-типа и что движение дырочного тока противоположно потоку электронов, ясно, что сила, действующая на движущиеся отрицательно заряженные частицы имела бы направление, показанное на рис. 6.11. Эта сила действует на электроны, которые скапливаются в нижней части материала. Таким образом возникает напряжение, которое в данном случае имеет положительный знак в верхней части материала. При внесении ферромагнетика в зону действия полупроводникового магнитного прибора напряженность магнитного поля увеличивается. Следовательно, уменьшается сила Лоренца, и в итоге — напряжение на полупроводнике. На этом падении напряжения

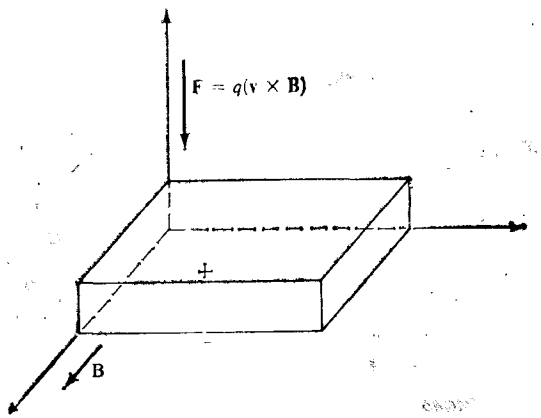


Рис. 6.11. Возникновение эффекта Холла.

и основан принцип измерения в ближней зоне датчиками Холла. Дискретный выходной сигнал, определяющий наличие объекта, реализуется пороговым ограничением выходного напряжения датчика.

Отметим, что использование таких полупроводников, как кремний, имеет ряд преимуществ с точки зрения размеров, точности и устойчивости к влиянию электрических помех. Кроме того, использование полупроводниковых материалов позволяет установить электронную схему усиления и обработки сигналов непосредственно на датчике, уменьшая таким образом его размер и стоимость.

6.3.3. Емкостные датчики

В отличие от индуктивных датчиков и датчиков Холла, которые идентифицируют ферромагнитные материалы, емкостные датчики обладают способностью (с различной степенью чувствительности) обнаруживать все твердые и жидкие материалы.

Как видно из их названия, эти датчики основаны на изменении емкости, которая зависит от расстояния до поверхности объекта в зоне действия чувствительного элемента.

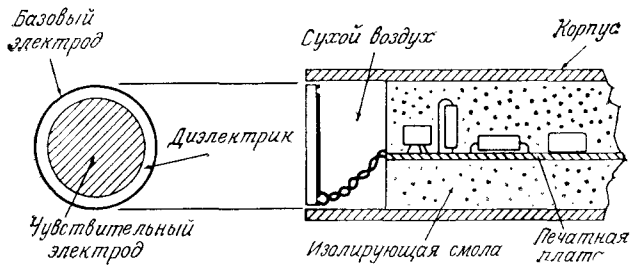


Рис. 6.12. Емкостной датчик измерения в ближней зоне [38].

Основные компоненты емкостного датчика показаны на рис. 6.12. Чувствительным элементом является конденсатор, состоящий из чувствительного электрода и базового электрода (ими могут быть, например, металлический диск и кольцо, разделенные диэлектриком). За емкостным элементом обычно помещают воздушную полость для обеспечения изоляции. В состав датчика может входить также электронная схема. В этом случае схему заплавляют в смолу, чтобы обеспечить ее герметичность и механическую защиту.

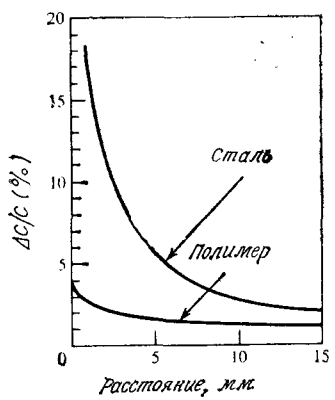


Рис. 6.13. Зависимость процентного изменения емкости датчика в ближней зоне от расстояния [38].

Существует ряд методов обнаружения в ближней зоне, основанных на изменении емкости. В одном простейшем методе конденсатор представляет собой элемент колебательного контура, колебания в котором возникают только в том случае, если емкость датчика превышает заданное пороговое значение. Колебания преобразуются затем в выходное напряжение, которое указывает на присутствие объекта в зоне измерения. Этот метод обеспечивает дискретный выходной сигнал, переключение которого зависит от значения заданного порога.

В более сложном методе используется емкостной элемент в контуре, по которому постоянно проходит синусоидальный сигнал частоты. Изменение емкости вызывает фазовый сдвиг

между сигналом эталонной частоты и сигналом от емкостного элемента. Фазовый сдвиг пропорционален изменению емкости и, следовательно, может быть использован для обнаружения объекта в ближней зоне.

На рис. 6.13 показано изменение емкости в зависимости от расстояния для датчика измерения в ближней зоне, построенного по рассмотренному принципу. Следует отметить, что на расстоянии, превышающем несколько миллиметров, чувствительность резко падает. Форма характеристики при этом зависит от материала объекта измерения. Обычно такие датчики работают в дискретном пороговом режиме. Изменение емкости выше заданного порога T соответствует наличию объекта, а ниже — его отсутствию в зоне, установленной величиной T.

6.3.4. Ультразвуковые датчики

Характеристики всех рассмотренных датчиков измерения в ближней зоне сильно зависят от материала объекта измерения. Эта зависимость может быть в значительной степени уменьшена путем использования ультразвуковых датчиков, работа которых при измерении расстояния была кратко рассмотрена

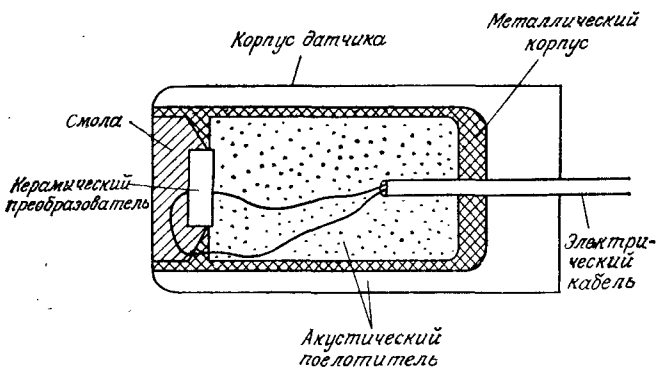


Рис. 6.14. Ультразвуковой датчик измерения в ближней зоне [39].

в разд. 6.2.3. В данном разделе более подробно рассмотрены конструкции и работа этих датчиков, а также проиллюстрировано их использование для измерений в ближней зоне.

На рис. 6.14 показана структура типичного ультразвукового преобразователя, используемого для измерения в ближней зоне. Основным элементом является электроакустический преобразователь, в качестве которого часто используется пьезоэлектрический керамический элемент. Подложка из смолы защищает преобразователь от влажности, пыли и других внешних воздействий. Она служит также как переходное акустиче-

ское сопротивление. Поскольку один и тот же преобразователь используется обычно как для передачи, так и для приема сигналов, для обнаружения объектов в ближней зоне необходимо быстрое демпфирование акустической энергии. Это достигается путем применения акустических поглотителей и развязкой преобразователя от корпуса. Конструкция корпуса позволяет получать узкий акустический поток, дающий мощный и направленный сигнал.

Для лучшего понимания работы ультразвукового датчика измерителя в ближней зоне надо провести анализ сигналов, используемых как для передачи, так и для приема акустической энергии. Типичный вид таких сигналов представлен на

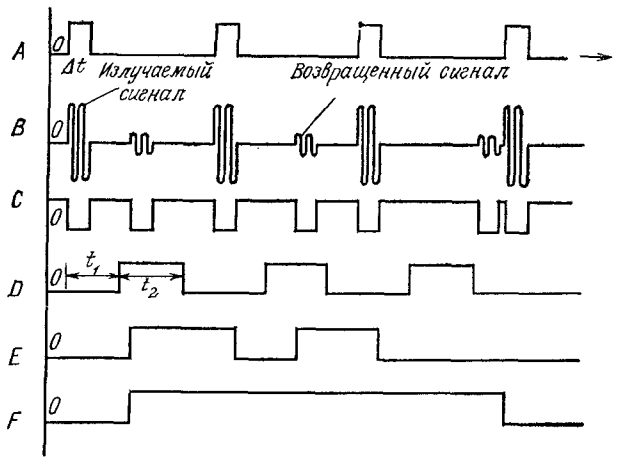


Рис. 6.15. Сигналы, используемые в ультразвуковом датчике измерения в ближней зоне [39].

рис. 6.15. Сигнал A является запорным сигналом, используемым для управления посылаемыми сигналами. Сигнал B содержит выходной и отраженный сигналы. Импульсы C выделяют сигналы передачи или приема. Для того чтобы установить различие между импульсами, соответствующими посылаемой и отраженной энергии, вводятся временные окна (сигнал D), на которых основывается принцип измерения датчика. Временной интервал Δt является минимальным временем измерения, а $\Delta t_1 + \Delta t_2$ — максимальным. Можно отметить, что эти временные интервалы соответствуют прохождению определенных расстояний со скоростью распространения звука в используемой рабочей среде. После получения отраженного сигнала (в то время как сигнал D имеет максимальное значение) вырабатывается сигнал E, величина которого принимает нулевое значение после окончания действия передающего импульса A. Наконец, сигнал F вырабатывается при появлении положительного импульса E и сбрасывается в случае отсутствия сигнала E и появления импульса A. Таким образом, сигнал F будет иметь максимальное значение при наличии объекта на расстоянии, определяемом параметрами сигнала D, т. е. сигнал F является выходным сигналом ультразвукового датчика, работающего в бинарном режиме.

6.3.5. Оптические датчики измерения в ближней зоне

Оптические датчики измерения в ближней зоне подобны ультразвуковым датчикам в том смысле, что они определяют близость объекта по его влиянию на волновой сигнал, проходящий от источника к приемнику. Один из наиболее распространенных методов измерения расстояния в ближней зоне с помощью оптических средств показан на рис. 6.16. Датчик состоит из светодиода, который выполняет роль источника ин-

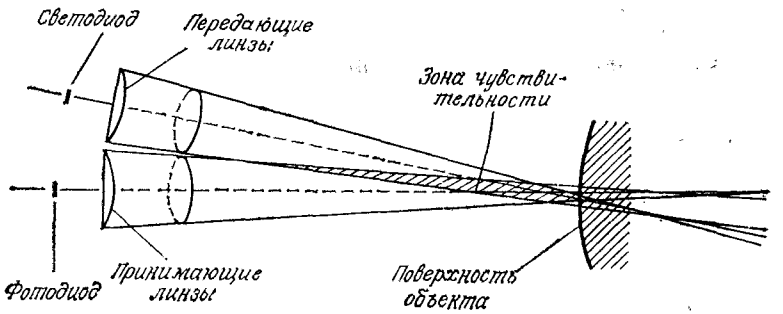


Рис. 6.16. Оптический датчик измерения в ближней зоне [251].

фракрасного излучения, и фотодиода, используемого в качестве приемника. Пучки света, сформированные фокусными системами источника и приемника в одной плоскости, пересекаются в вытянутой конусовидной зоне. Эта зона определяет рабочий диапазон датчика, так как отражающая поверхность, которая находится в зоне, освещается источником и одновременно «просматривается» приемником.

Хотя данный метод в принципе похож на метод триангуляции, рассмотренный в разд. 6.2.1, необходимо отметить, что зона измерений, показанная на рис. 6.16, обеспечивает не только точечное измерение. Другими словами, поверхность, находящаяся в любом месте указанной зоны, будет идентифицирована. Для объекта с известной ориентацией и характеристиками отражения можно осуществить калибровку интенсивности изображения в функции расстояния, однако обычно систему, приведенную на рис. 6.16, используют в режиме, при котором фор-

мируется дискретный выходной сигнал при достижении интенсивности отраженного потока определенного порогового значения.

6.4. ТАКТИЛЬНЫЕ ДАТЧИКИ

Тактильные датчики используются в робототехнике для получения информации о контакте манипулятора с объектами в рабочем пространстве. Тактильная информация может использоваться, например, для определения местоположения объекта или его распознавания, а также для управления усилием захватного устройства, воздействующим на объект манипулирования. Тактильные датчики подразделяются на два основных типа: дискретные и аналоговые. Дискретные датчики, как правило, срабатывают при наличии или отсутствии объекта, в то время как выходной сигнал аналоговых датчиков пропорционален прикладываемому усилию. Более подробно эти датчики рассматриваются в следующих разделах.

6.4.1. Дискретные пороговые датчики

Как было отмечено, дискретные тактильные датчики являются контактными приборами типа микропереключателей. В простейшем случае переключатель размещается на внутренней поверхности каждого пальца манипулятора (рис. 6.17). Этот вариант очувствления используется для определения наличия детали между пальцами схвата. Перемещая манипулятор над объектом и последовательно производя коитактирование

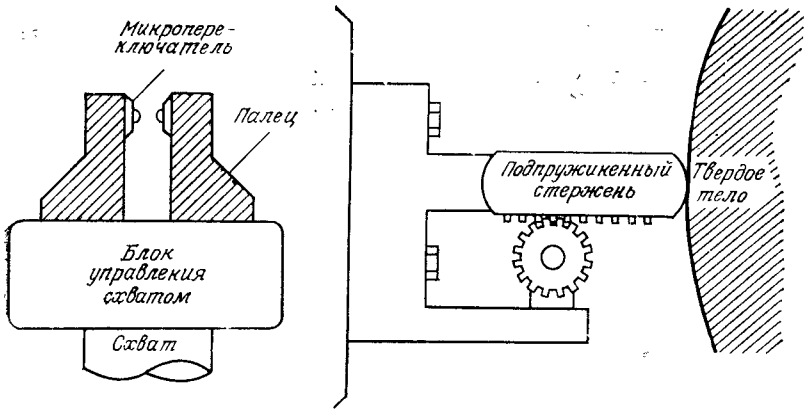


Рис. 6.17. Простой схват робота с бинарными тактильными датчиками.

Рис. 6.18. Типичный аналоговый тактильный датчик.

с его поверхностью, можно также осуществить центрирование манипулятора относительно объекта для его захвата и переноса.

Путем размещения нескольких дискретных тактильных датчиков на внутренней поверхности каждого пальца схвата достигается расширение получаемого объема информации. Кроме того, они часто ставятся на внешней поверхности конечного звена манипулятора для получения управляющих сигналов, используемых при формировании траектории движения манипулятора в рабочем пространстве. Последний вариант аналогичен использованию тактильной информации человеком, проходящим через абсолютно темную комнату.

6.4.2. Аналоговые датчики

Аналоговый тактильный датчик является регистрирующим прибором, выходной сигнал которого пропорционален прикладываемой силе. Простейший из таких приборов состоит из подпружиненного стержня (рис. 6.18), который механически связан с вращающейся осью. Горизонтальная сила, действующая на стержень, преобразуется в пропорциональный поворот оси. Этот поворот непрерывно измеряется с помощью потенциометра или кодовым устройством с дискретным выходом. При известной жесткости пружины сила соответствует указанному перемещению.

В последние годы значительное внимание уделяется развитию систем тактильного очувствления, способных получать бóльший объем информации, чем один датчик. Использование таких систем проиллюстрировано на рис. 6.19, где показан схват робота, на внутренней поверхности каждого пальца которого размещается матрица тактильных датчиков. Внешние чувствительные поверхности схвата обычно выполняются в виде дискретных устройств.

Хотя чувствительные матрицы могут выполняться из некоторого множества отдельных датчиков, одним из перспективных направлений здесь является использование электродов из проводящих материалов (например, на графитовой основе), сопротивление которых меняется в зависимости от величины давления. В таких устройствах, обычно называемых «искусственной кожей», давление от объекта вызывает соответствующие деформации, которые измеряются как непрерывно меняющееся сопротивление. Изменение сопротивления легко преобразуется в электрический сигнал, амплитуда которого пропорциональна силе, действующей на соответствующую точку поверхности материала.

Несколько основных методов, используемых при создании искусственной кожи, показаны на рис. 6.20. Схема, приведенная на рис. 6.20, *а*, основана на эффекте «окна», проявляемом про-

водящим материалом, помещенным между общим корпусом и системой электродов, выполненных на стеклокерамической печатной плате. Каждый электрод представляет собой прямоугольное поле — «окно», которое идентифицирует одну точку касания. Сила тока от общего корпуса к каждому электроду зависит от давления на проводящий материал.

В методе, показанном на рис. 6.20, *б*, пара длинных и узких электродов размещена в плоскости активных электрических контуров с использованием СБИС-технологии. Проводящий материал находится над этой плоскостью и изолирован от нее по

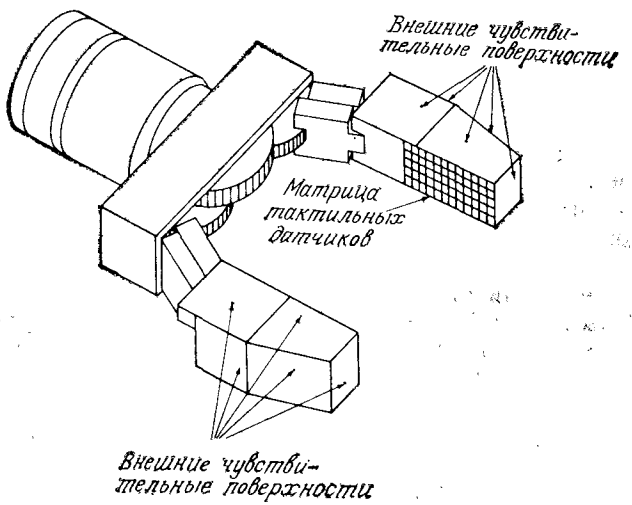


Рис. 6.19. Схват робота, оснащенный матрицами тактильных датчиков.

всей поверхности, кроме поверхности электродов. Давление от объекта изменяет сопротивление электродов, которое затем преобразуется с помощью электронных контуров, размещенных между электродными парами в сигналы тока или напряжения.

Другая возможная реализация приведена на рис. 6.20, *в*. Проводящий материал располагается между двумя взаимно перпендикулярными наборами тонких, плоских и гибких электродов. Каждое пересечение между ними, разделенное проводящим материалом, представляет собой одну чувствительную точку. Изменение сопротивления в функции давления измеряется путем смещения электродов одного набора и изменения тока в элементах второго набора. Величина тока в каждом из этих элементов пропорциональна давлению между неподвижным элементом и элементом, смещаемым внешней силой.

Наконец, система, показанная на рис. 6.20, *г*, построена с использованием анизотропного проводящего материала. Эти

материалы обладают односторонней проводимостью. В основании датчика расположена система плоских электродов, а сверху находится проводящий материал, направление проводимости которого перпендикулярно осям электродов. Проводящий материал отделен от электродов сеткой для его изоляции при отсутствии внешней силы. Когда приложена внешняя сила, происходит контакт между материалом и электродами. Возрастание

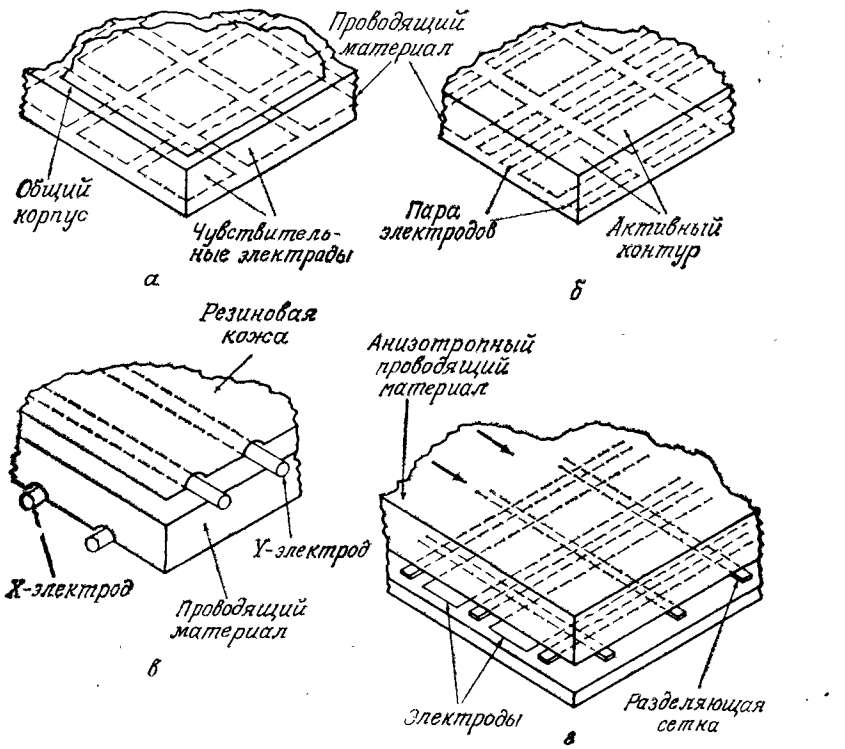


Рис. 6.20. Четыре метода реализации искусственной кожи.

силы приводит к увеличению площади контакта и уменьшению сопротивления. Как и в предыдущем методе, один из элементов чувствительной системы перемещается под действием внешней силы, а на втором производится измерение тока. Необходимо отметить, что чувствительность такого датчика зависит от толщины сепаратора.

Рассмотренные методы (рис. 6.20, *в* и *г*) основаны на соответствующем смещении элементов чувствительной системы. Это часто приводит к трудностям при получении тактильной инфор-

мации со сложных деталей, поскольку пересечение электрических контуров индуцирует помехи в виде обратных токов. Одним из решений этой проблемы является установка диодов в каждом контуре, исключающих прохождение обратных токов. Другое решение заключается в заземлении всех контуров, кроме перемещаемого контура. Вклад каждого элемента пересечения можно выявить путем индивидуального контроля всех контуров принимающей системы.

Все рассмотренные тактильные датчики измеряют силы, перпендикулярные к чувствительной поверхности датчика.

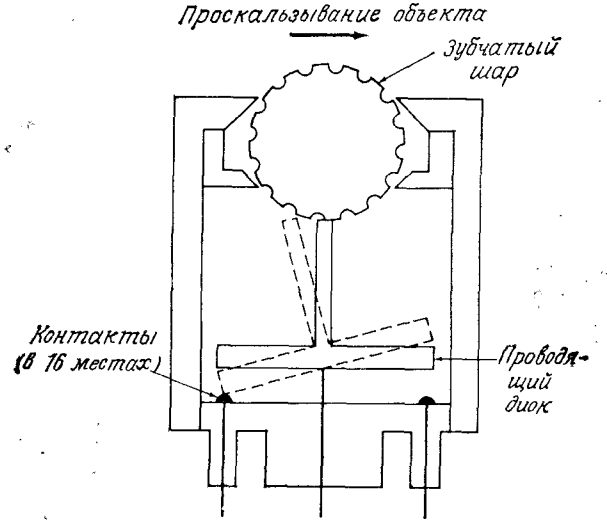


Рис. 6.21. Устройство для определения величины и направления проскальзывания [18].

Определение проскальзывания путем измерения тангенциального движения является другой важной задачей тактильного очувствления. В заключение кратко рассмотрим один из методов такого очувствления [18]. Реализация метода (рис. 6.21) включает свободно вращающийся зубчатый шар, который отклоняет тонкий стержень, установленный на оси проводящего диска. Под диском равномерно расположены электрические контакты. Вращение шара, вызванное проскальзыванием по нему объекта, приводит к вибрации стержня и диска с частотой, пропорциональной скорости вращения шара. От направления вращения зависит, какой контакт будет задействован вибрирующим диском. Усредненное направление проскальзывания определяется по импульсам в соответствующих выходных электрических контурах.

6.5. СИЛОМОМЕНТНОЕ ОЧУВСТВЛЕНИЕ

Силомоментные датчики используются в основном для определения сил реакции, возникающих при механической сборке. Основные методы в этой области направлены на очувствление сочленений и схвата робота. Возможно также очувствление, при котором силовой преобразователь устанавливается между основанием робота и установочной поверхностью для измерения компонентов сил и моментов, действующих на основание. Однако в большинстве случаев основание жестко устанавливается на твердой поверхности, и в подобного рода очувствлении не возникает необходимости. Указанное очувствление аналогично очувствлению схвата, которое подробно рассматривается в данном разделе.

Датчик сочленения измеряет в декартовых координатах силы и моменты, которые действуют на робот, и производит их векторное сложение. Для сочленения, перемещаемого с помощью двигателя постоянного тока, очувствление производится простым измерением тока якоря. Датчики схвата, которые рассмотрены в данном разделе, размещаются между конечным звеном манипулятора и схватом. Они состоят из измерителей напряжений, которые определяют отклонение механической системы под действием внешних сил. Характеристики и методологический анализ этого типа датчиков рассматриваются в данном разделе.

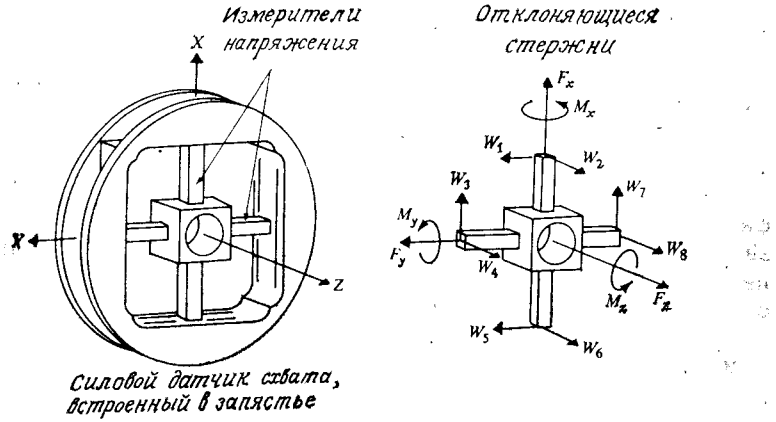
6.5.1. Элементы датчика схвата, встроенного в запястье

Датчики схватов представляют собой небольшие, чувствительные, легкие (~370 г) и относительно компактные конструкции диаметром 10 см и толщиной 3 см с динамическим диапазоном до 90 кг. Для уменьшения гистерезиса и увеличения точности измерения датчик обычно выполняется из одной твердой металлической заготовки (как правило, алюминиевой). Например, датчик, показанный на рис. 6.22, содержит восемь пар полупроводниковых измерителей напряжения, установленных на четырех отклоняющихся стержнях — по одному измерителю на каждой стороне стержня. Измерители с противоположных концов отклоняющихся стержней подсоединены дифференциально к потенциометрическому контуру, выходное напряжение на котором пропорционально компонентам сил, нормальным к плоскости измерителей. Дифференциальное включение измерителей напряжения обеспечивает автоматическую компенсацию изменений температуры.

Однако это является лишь первичной грубой компенсацией. Так как восемь пар измерителей напряжения расположены нормально к осям **x**, **y** и **z** системы координат сил, три компо-

ненты силы **F** и три компоненты момента **M** могут быть определены соответствующим сложением или вычитанием выходных напряжений. Это может быть выполнено с использованием матрицы калибровки датчика (см. разд. 6.5.2).

Большинство силовых датчиков схватов функционирует как преобразователи сил и моментов, действующих на манипулятор, в измеряемые отклонения или перемещения в схвате. Важно, чтобы движения в схвате, производимые силовым датчиком, не влияли на точность позиционирования манипулятора. Таким образом, требования к датчикам можно сформулировать следующим образом.



Силовой датчик схвата, встроенный в запястье

Рис. 6.22. Силовой датчик схвата, встроенный в запястье.

1. *Высокая жесткость.* Частота собственных колебаний механического устройства связана с его жесткостью, следовательно, высокая жесткость обеспечивает быстрое демпфирование возникающих колебаний при измерении сил и точность показаний на коротких временных интервалах. Кроме того, это снижает величину отклонений от действия сил и моментов, которая может привести к ошибке позиционирования манипулятора.

2. *Компактность конструкции.* Это позволяет облегчить движение манипулятора в условиях навала деталей, а также уменьшить вероятность столкновения датчика с объектами, находящимися в рабочем пространстве. Компактный датчик можно размещать ближе к расположенному в схвате технологическому оборудованию, благодаря чему уменьшается ошибка позиционирования оборудования из-за неадекватности рабочих условий оборудования и датчика. Кроме того, желательно расширить диапазон измерения сил и моментов. Этому способствует минимизация расстояния между манипулятором и датчиком, приводящая к уменьшению величины рычага прикладываемых к манипулятору сил.

3. *Линейность.* Хорошая линейность выхода чувствительных элементов от прикладываемых сил и моментов позволяет выделить силы и моменты с помощью простых матричных операций. Кроме того, упрощается процесс калибровки датчика силы (разд. 6.5.2).

4. *Малые величины гистерезиса и внутреннего трения.* Внутреннее трение уменьшает чувствительность измерительных элементов. Это также уменьшает гистерезисные эффекты при возвращении измерительного прибора в исходное положение.

Силовой датчик схвата, показанный на рис. 6.22, спроектирован с учетом указанных требований.

6.5.2. Выделение сил и моментов

Предположим, что взаимовлияние различных измерителей пренебрежимо мало, силовой датчик схвата работает в диапазоне упругих деформаций и измерители напряжения дают показания, которые линейно зависят от их отклонений. Тогда датчик (рис. 6.22) выдает восемь рядов измерений, которые должны быть обработаны программным путем на ЭВМ с использованием простого метода выделения трех ортогональных компонент сил и моментов относительно системы координат датчика силы. Такая обработка может быть реализована путем определения матрицы размерностью 6×8, называемой матрицей разделения силы (или матрицей калибровки датчика) $\mathbf{R}_F$, которая составляется на основе измерений силы для выделения трех ортогональных компонент силы и момента. В соответствии с рис. 6.22 разделяемый вектор силы, направленный вдоль координатных осей датчика силы, математически представляется в виде

$$\mathbf{F} = \mathbf{R}_F \mathbf{W}, \tag{6.5-1}$$

где

$$\mathbf{F} \equiv (\text{силы, моменты})^T = (F_x, F_y, F_z, M_x, M_y, M_z)^T,$$

$$\mathbf{W} \equiv (\text{ряд измерений}) = (w_1, w_2, w_3, \ldots, w_8)^T,$$

и

$$\mathbf{R}_F = \begin{bmatrix} r_{11} & \ldots & r_{18} \\ \cdot & \cdot & \cdot \\ r_{61} & \ldots & r_{68} \end{bmatrix}. \tag{6.5-2}$$

В выражении (6.5-2) $r_{ij} \neq 0$ — являются членами, требующими преобразования ряда измерений **W** (в вольтах) в силы и моменты (в ньютонах и ньютонах на метр соответственно). Если пренебречь взаимовлиянием измерителей, то, суммируя (согласно рис. 6.22) силы и моменты относительно начала координат датчика, размещенного в центре датчика силы, полу-

чим выражение (6.5-2) с некоторыми r_{ij}, равными нулю. Для датчика, представленного на рис. 6.22, матрица разделения силы по уравнению (6.5-2) примет вид

$$\mathbf{R}_F = \begin{bmatrix} 0 & 0 & r_{13} & 0 & 0 & 0 & r_{17} & 0 \\ r_{21} & 0 & 0 & 0 & r_{25} & 0 & 0 & 0 \\ 0 & r_{32} & 0 & r_{34} & 0 & r_{36} & 0 & r_{38} \\ 0 & 0 & 0 & r_{44} & 0 & 0 & 0 & r_{48} \\ 0 & r_{52} & 0 & 0 & 0 & r_{56} & 0 & 0 \\ r_{61} & 0 & r_{63} & 0 & r_{65} & 0 & r_{67} & 0 \end{bmatrix}. \qquad (6.5\text{-}3)$$

Довольно частое предположение об отсутствии взаимосвязи измерителей не соответствует действительности. Для некоторых датчиков силы соответствующая погрешность измерений достигает 5 %. Таким образом, на практике обычно необходимо заменять матрицу разделения силы $\mathbf{R}_F$ на матрицу, содержащую 48 ненулевых элементов. Такая «полная» матрица используется для калибровки датчика силы (разд. 6.5.3). Разделенный вектор силы $\mathbf{F}$ используется для получения ошибки, необходимой для выработки сигнала управления манипулятором. Недостатком использования силового датчика схвата, встроенного в запястье, является то, что он обеспечивает измерение векторов силы, разделяемых в процессе контакта элементов при сборке только в одной точке.

6.5.3. Калибровка датчика

Целью калибровки силового датчика схвата, встроенного в запястье, является определение всех 48 неизвестных элементов матрицы разделения силы (уравнение (6.5-2)) на основе экспериментальных данных. В связи с наличием взаимовлияния контуров датчика необходимо найти 48 ненулевых элементов в матрице $\mathbf{R}_F$. Калибровка силового датчика схвата производится путем определения псевдообратной матрицы калибровки $\mathbf{R}_F^*$, которая удовлетворяет соотношениям

$$\mathbf{W} = \mathbf{R}_F^* \mathbf{F} \qquad (6.5\text{-}4)$$

и

$$\mathbf{R}_F^* \mathbf{R}_F = \mathbf{I}_{8\times 8}, \qquad (6.5\text{-}5)$$

где $\mathbf{R}_F^*$ — матрица размерностью 8×6, а $\mathbf{I}_{8\times 8}$ — единичная матрица размерностью 8×8. Тогда матрица калибровки $\mathbf{R}_F^*$ из уравнения (6.5-1) может быть найдена из псевдообратной матрицы в уравнении (6.5-4) с использованием метода наименьших

квадратов. Перемножая уравнение (6.5-4) на $(\mathbf{R}_F^*)^T$, получаем

$$(\mathbf{R}_F^*)^T \mathbf{W} = [(\mathbf{R}_F^*)^T \mathbf{R}_F^*] \mathbf{F}. \qquad (6.5\text{-}6)$$

После обращения матрицы $[(\mathbf{R}_F^*)^T \mathbf{R}_F^*]$ имеем

$$\mathbf{F} = [(\mathbf{R}_F^*)^T \mathbf{R}_F^*]^{-1} (\mathbf{R}_F^*)^T \mathbf{W}. \qquad (6.5\text{-}7)$$

Сравнивая уравнения (6.5.1) и (6.5.7), получим

$$\mathbf{R}_F \cong [(\mathbf{R}_F^*)^T \mathbf{R}_F^*]^{-1} (\mathbf{R}_F^*)^T. \qquad (6.5\text{-}8)$$

Матрица $\mathbf{R}_F^*$ может быть определена путем расстановки известных весовых коэффициентов при векторах, расположенных вдоль осей системы координат датчика. Подробно экспериментальный процесс калибровки матрицы разделения силы изложен в работе [264].

6.6. ЗАКЛЮЧЕНИЕ

Материал, представленный в данной главе, характеризует уровень развития области создания датчиков роботов, дающих информацию о внешней среде. Однако надо отметить, что возможности таких датчиков гораздо более ограничены, чем человеческие возможности в этом плане.

Как отмечалось в начале данной главы, большинство современных промышленных роботов функционирует по заранее заданной жесткой программе без использования обратной связи от датчиков. Повышение интереса к гибким автоматизированным производствам привело к развитию робототехнических систем, управляемых с помощью датчиков, как к средству расширения области применения этих машин. Таким образом, совершенствование датчиков представляет собой динамично развивающуюся область с использованием известных из литературы новых методов и технологий.

Литература

Обзорными статьями по робототехнике являются [18, 91, 195, 197, 251]. Материалы по лазерным измерителям расстояний имеются в работах [67, 137, 138]. Продолжение материала, изложенного в разд. 6.3, находится в работах [38—40, 271], а в разд. 6.4 — в работах [113, 120, 187, 241], а также в работах [22, 110, 195]. Датчики для основания робота (разд. 6.5) рассматриваются в работе [63]. Дополнительные сведения по силомоментному очувствлению можно найти в работах [196, 210, 264, 313].

Упражнения

6.1. Докажите справедливость выражения (6.2-8).

6.2. Датчик измерения расстояния по времени прохождения сигнала освещает рабочее пространство с двумя объектами, имеющими изображение, показанное на телеэкране (рис. 6.23). Предполагая, что измерительная система расположена согласно рис. 6.3, *б* и что $M = 256$, $D_0 = 1$ м, $D_c = 2$ м, $B = 3$ м, а $\lambda = 35$ мм, определите расстояние между объектами в направлении светового луча.

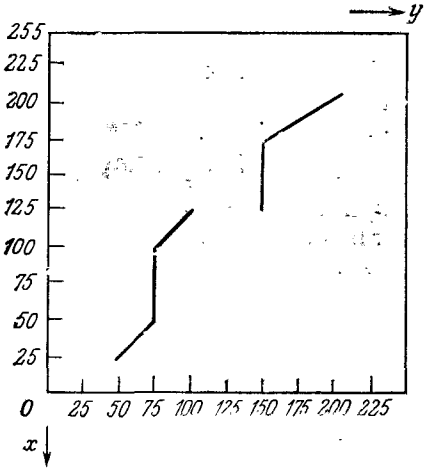


Рис. 6.23.

6.3. а) Гелий-неоновый лазер с непрерывным излучением (длина волны 632,8 мм), используемый для измерения расстояния, имеет модулированый сигнал с частотой 30 мГц. Какое расстояние до объекта соответствует фазовому сдвигу 180°?

б) Какова верхняя граница диапазона измерения расстояния, на котором прибор дает однозначные показания?

6.4. Определите верхнюю границу частоты модулирующего сигнала для обеспечения диапазоиа измерения до 5 м при использовании измерителя расстояний на базе лазера с непрерывным излучением.

6.5. а) Предположим, что точность лазерного измерителя расстояний снижается шумом с гауссовским распределением, имеющим нулевое среднее значение и стандартное отклонение 100 см. Сколько измерений необходимо усреднить, чтобы получить точность $\pm 0{,}5$ см с вероятностью 0,95? б) Как изменится количество измерений, если среднее значение шума будет 5 см?

6.6. Используя рис. 6.15, покажите вид сигналов ультразвукового датчика, способного измерять расстояние, а не только выдавать бинариый выходной сигнал в ближней зоне действия.

6.7. Предположим, что ультразвуковой датчик измерения используется для определения нахождения объекта в зоне 0,5 м. В момент $t = 0$ преобразователь выдал импульс длительностью 0,1 мс. Предположим также, что для гашения резонансных колебаний в преобразователе требуется 0,4 мс, а в окружающей среде — 20 мс. Скорость звука равна 344 м/с.

а) Какой интервал времени можно использовать в качестве окна?

б) Через какое время прибор может выдать повторный импульс?

в) Каково минимально измеряемое расстояние?

6.8. Оптический датчик расстояний (рис. 6.16) имеет зону чувствительности, сформированную пересечением двух одинаковых лучей. Конус, образуемый каждым лучом, начинается на линзах и имеет вершину, расположенную в 4 см напротив центра противоположной линзы. В какой зоне датчик обнаруживает объект, если каждая из линз имеет диаметр 4 мм, а центры линз разнесены на 6 мм? Предполагается, что объект может быть обнаружен в любом месте зоны чувствительности.

6.9. Измерительная система размерностью 3×3 тактильного датчика сканируется путем смещения по рядам, значение элементов которых равно 5 В. Столбцы считываются путем закорачивания их элементов на корпус и измерения тока. Предположим, что несмещенные ряды и несчитанные столбцы обладают высоким сопротивлением. Действие силы иа систему приводит к появлению следующих величии сопротивлений иа каждом пересечении электродов (ряд, столбец): 100 Ω на (1,1), (1,3), (3,1) я (3,3), 50 Ω на (2,2) и (3,2). Все другие пересечения имеют бесконечно большое сопротивление. Определите ток, измеренный в каждом пересеченяи ряда и столбца в системе, учитывая их взаимовлияние.

6.10. Рассмотрите упр. 6.9 в предположении:

а) все несмещенные ряды и столбцы закорочены на корпус;

б) в каждом соединении установлен диод (0,6 В) с сопротивлением.

6.11. На роботе Пума с плоскопараллельным движением схвата установлен датчик силы. Произведена процедура калибровки для получения матрицы $\mathbf{R}_F$. После завершения всех измерений на роботе был заменен схват. Есть ли необходимость повторной калибровки силового датчика схвата? Обоснуйте ответ.

Глава 7.
СИСТЕМЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ НИЗКОГО УРОВНЯ

Лучше один раз увидеть, чем сто раз услышать.

Пословица

7.1. ВВЕДЕНИЕ

«Зрительные» возможности робота, как и людей, обеспечиваются сложным чувствительным механизмом, который позволяет гибко и «осмысленно» реагировать на изменения внешней среды. Использование технического зрения и других методов очувствления, рассмотренных в гл. 6, диктуется постоянной необходимостью повышать гибкость и расширять области применения робототехнических систем. Хотя датчики расстояния, тактильные датчики и датчики силы играют большую роль в совершенствовании функционирования робота, техническое зрение является наиболее мощным источником информации для робота. Измерительные системы, методы измерения и оборудование при использовании технического зрения имеют гораздо больше возможностей, чем при использовании датчиков, работа которых описана в гл. 6.

Зрение робота можно определить как процесс выделения, идентификации и преобразования информации, полученной из трехмерных изображений. Этот процесс, называемый также *техническим* или *машинным зрением*, разделяется на шесть основных этапов: 1) снятие информации, 2) предварительная обработка информации, 3) сегментация, 4) описание, 5) распознавание, 6) интерпретация. Снятие информации представляет собой процесс получения визуального изображения. Предварительная обработка информации заключается в использовании таких методов как понижение шума или улучшение изображения отдельных деталей. Сегментация — процесс выделения на изображении интересующих объектов. При описании определяются характерные параметры (например, размеры или форма), необходимые для выделения требуемого объекта на фоне других. Распознавание представляет собой процесс идентификации объектов (например, гаечного ключа, болта, блока двигателя). Наконец, интерпретация выявляет принадлежность к группе распознаваемых объектов.

Названные этапы удобно сгруппировать в соответствии со сложностью их реализации. Выделим три уровня технического зрения: низкий, средний и высокий. Хотя четких границ между этими уровнями не существует, их выделение целесообразно для классификации различных процессов, характерных для систем технического зрения. Например, под низким уровнем технического зрения понимаются такие процессы, которые являются простыми с точки зрения осуществления автоматических действий, не требующих наличия искусственного интеллекта. К низкому уровню технического зрения мы будем также относить снятие и предварительную обработку информации. Таким образом, этот уровень охватывает процессы, начиная непосредственно от формирования изображения и кончая процессами компенсации, такими, как уменьшение шума, а также процессами выделения простейших параметров изображения, такими, как разрывы интенсивности. Подобные действия можно сравнить с действиями человека, пытающегося найти свое место в темном зале кинотеатра, в который он зашел прямо с яркого дневного света. Осмысленный процесс нахождения незанятого места не может начаться до того, как будет получено соответствующее изображение пространства.

Под средним уровнем технического зрения понимаются процессы выделения, идентификации и разметки элементов изображения, полученного на нижнем уровне. С учетом принятого подразделения технического зрения к среднему уровню относятся сегментация, описание и распознавание отдельных объектов.

Под высоким уровнем технического зрения понимаются процессы, относящиеся к искусственному интеллекту. В то время как алгоритмы, используемые на нижнем и среднем уровнях технического зрения, разработаны достаточно хорошо, знание и понимание процессов высокого уровня пока еще недостаточны. В соответствии с изложенным в гл. 8 это приводит к введению ограничений и предположений для уменьшения сложности задачи.

Рассмотренную классификацию можно использовать в основном для всех видов применяемых систем технического зрения. Это не означает, что она может служить моделью человеческого зрения и в то же время не связана с ним. Известно, например, что распознавание и интерпретация для человека представляют собой взаимосвязанные функции. Однако наличие этих связей не определяет возможностей их аналитического моделирования. Таким образом, приведенная классификация функций может рассматриваться как практический метод реализации систем технического зрения, отражающий наш уровень понимания и владения аналитическими подходами в этой области.

В главе описаны получение изображения, его предварительная обработка, а также идеи и методы реализации функций, осуществляемых на низком уровне технического зрения. Верхний (высокий) уровень технического зрения рассмотрен в гл. 8. Хотя зрение отражает объемное трехмерное пространство, в системах технического зрения используется только его плоское изображение, объемную информацию из которого получают с помощью специальных методов, таких, как метод структурного освещения (разд. 7.3) или стереоизображения (разд. 7.4).

7.2. ПОЛУЧЕНИЕ ИЗОБРАЖЕНИЯ

Визуальная информация преобразуется в электрические сигналы с помощью видеодатчиков. После пространственной дискретизации и квантования по амплитуде эти сигналы дают цифровое изображение. Ниже рассмотрены основные методы получения изображения при использовании технического зрения в роботах, влияние дискретизации на пространственное разделение и влияние квантования по амплитуде на разделение по интенсивности. Математический аппарат формирования изображения описан в разд. 7.4.

Основными устройствами, используемыми в техническом зрении роботов, являются телевизионные камеры, состоящие из электронной трубки или твердотельного чувствительного элемента с соответствующими электронными блоками. Хотя подробное изучение этих устройств не входит в рамки данной книги, мы рассмотрим принцип работы трубки видикон, которая представляет семейство телевизионных камер. Твердотельные чувствительные элементы будут представлены приборами с зарядовой связью (ПЗС). Они обладают рядом преимуществ перед электронными трубками (например, имеют меньшие вес и размеры, больший ресурс и малую энергоемкость). Однако разрешающая способность некоторых трубок пока выше возможностей твердотельных камер.

Трубка видикон представляет собоой цилиндрическую стеклянную оболочку, содержащую с одного конца электронную пушку, а с другого — экран и мишень (рис. 7.1, *а*). Электронный луч фокусируется и отклоняется с помощью напряжения, прикладываемого к катушкам. Отклоняющий контур обеспечивает сканирование луча по внутренней поверхности мишени для «считывания» изображения. Внутренняя поверхность стеклянного экрана покрыта прозрачной металлической пленкой, которая образует электрод, формирующий электрический видеосигнал. На металлическую пленку нанесен тонкий фоточувствительный слой мишени. Этот слой состоит из мелких шаровидных частиц, сопротивление которых обратно пропорционально интенсивности светового потока. За фоточувствительной мишенью расположена

положительно заряженная тонкая проволочная решетка, которая тормозит электроны, испускаемые пушкой, так что они попадают на мишень со скоростью, близкой к нулю.

В нормальном режиме на металлическое покрытие экрана подается положительный потенциал. При отсутствии света фоточувствительный материал ведет себя как диэлектрик, так как

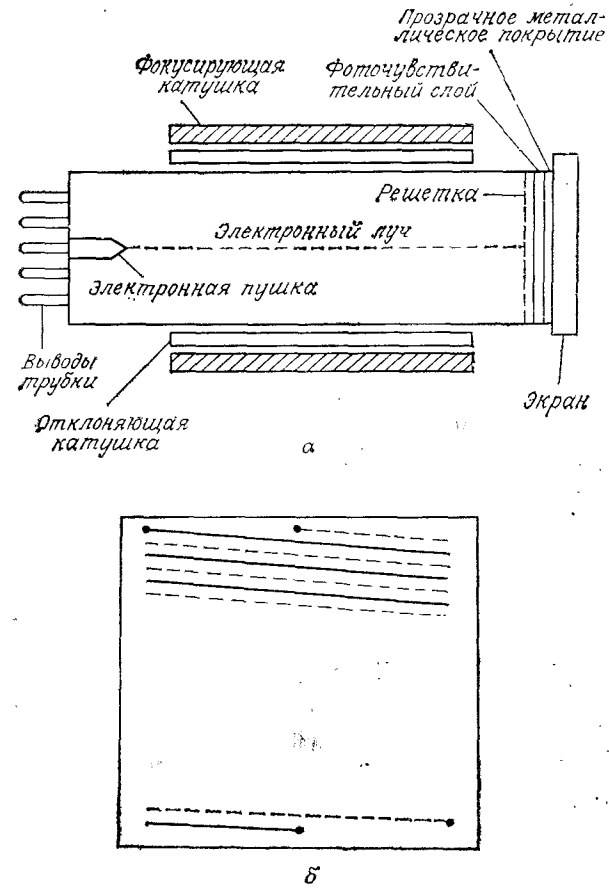


Рис. 7.1. Схема трубки видикон (*а*) и сканирование электронным лучом (*б*).

потенциал на внутренней поверхности мишени, вызываемый электронным лучом, компенсируется положительным зарядом на металлическом покрытии. Таким образом, при сканировании электронного луча по поверхности мишени фоточувствительный слой становится накопителем отрицательного заряда на внутренней поверхности и положительного заряда на внешней поверхности. Когда на поверхностный слой мишени попадает свет, его

сопротивление падает и появляется электрический ток, нейтрализующий положительный заряд. Так как поток электронов пропорционален величине светового потока в каждой точке мишени, на поверхностном слое мишени образуется изображение, соответствующее световому изображению на экране трубки, т. е. остаточная концентрация электронных зарядов выше в темных местах и ниже в светлых местах мишени. Таким образом, в процессе сканирования мишени образуется ток в металлическом слое, который подается на выводы трубки. Величина тока пропорциональна числу перемещающихся электронов и, следовательно, интенсивности светового потока в соответствующем месте, по которому проходит сканирующий луч. Это изменение тока во время сканирования электронного луча после его обработки в электронном блоке камеры формирует видеосигнал, пропорциональный интенсивности входного изображения.

Основной стандартный вариант сканирования, используемый в США, показан на рис. 7.1, *б*. Электронный луч сканирует по всей поверхности мишени 30 раз в 1 с. Полный объем сканирования (называемый кадром) состоит из 525 линий, 480 из которых содержат информацию об изображении. Если линии сканировать последовательно, а результат подавать на телевизионный монитор, то изображение будет заметно дрожать. Это можно устранить, используя механизм сканирования, при котором кадр делится на две соединенные области по 262,5 линии. Они сканируются 60 раз в 1 с, т. е. с удвоенной скоростью. Первая область в каждом кадре сканируется по нечетным линиям (показаны штрихами на рис. 7.1, *б*), в то время как вторая область сканируется по четным линиям. Данная схема сканирования, соответствующая соглашению по сканированию RETMA, является стандартной для телепередач в США. Другие стандарты применяются при использовании более высоких скоростей сканирования кадра, но принцип их действия по существу совпадает с описанным. Например, распространенный метод сканирования в машинном зрении и при получении цифрового изображения основан на применении 559 линий, 512 из которых содержат данные об изображении. Работа с числами в виде целых степеней числа 2 имеет ряд преимуществ как в аппаратной, так и в программной реализации.

При рассмотрении устройств на ПЗС удобно подразделить датчики на два типа: датчики линейного сканирования и датчики с плоскостной структурой. Основной частью ПЗС-датчиков является ряд кремниевых чувствительных элементов, называемых фотоячейками. Фотоны от отображаемого объекта проходят через входиую прозрачную поликристаллическую кремниевую структуру и абсорбируются в кристалле кремния, образуя пары электрон — дырка. Полученные фотоэлектроны собираются на фотоячейках, при этом величина заряда на каждой фотоячейке

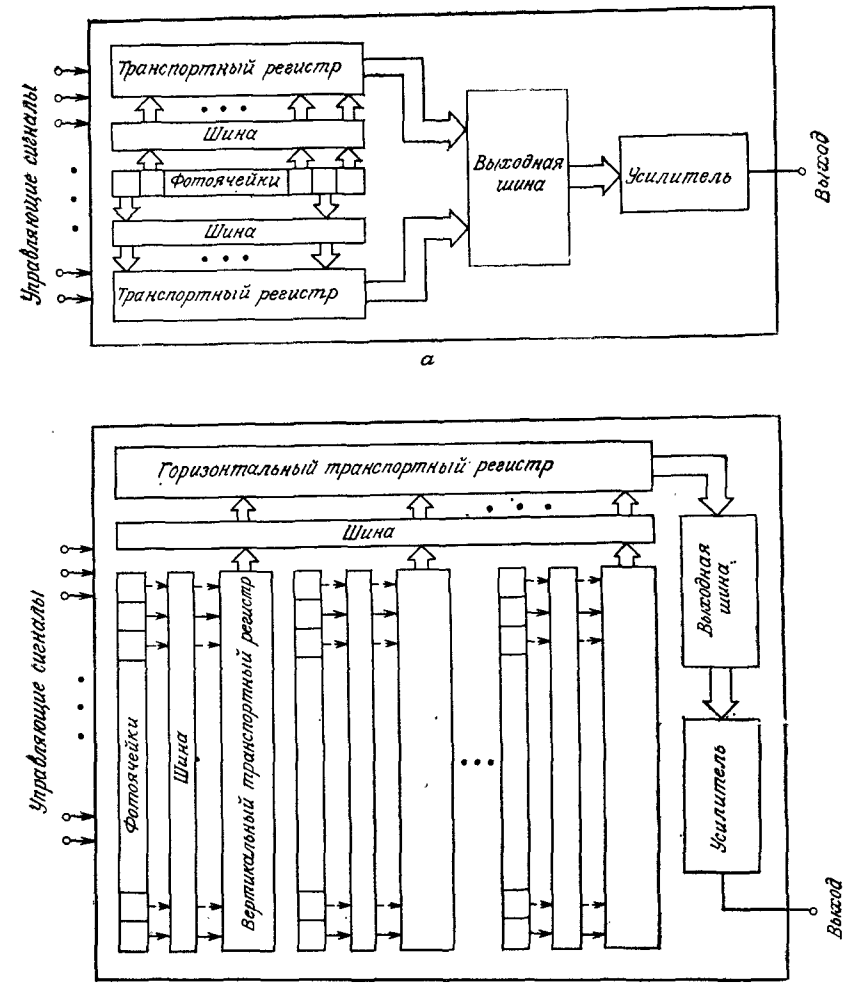


Рис. 7.2. ПЗС-датчик линейного сканирования (*а*) и ПЗС-датчик с плоской структурой (*б*).

пропорциональна соответствующей интенсивности светового потока. Типичный датчик линейного сканирования (рис. 7.2, *а*) состоит из ряда фоточувствительных элементов, из двух шин, используемых для передачи содержимого с фоточувствительных элементов в транспортные регистры, а также из выходной шины, служащей для передачи содержимого из транспортных регистров на усилитель. На выходе усилителя формируется сигнал напряжения, величина которого пропорциональна содержимому фотоячеек.

ПЗС-датчики с плоскостной структурой аналогичны датчикам линейного сканирования с тем отличием, что в них фотоячейки расположены в форме матрицы, а между рядами фотоячеек имеется комбинация переходных транспортных регистров (рис. 7.2, *б*). Информация с нечетных номеров фотоячеек поступает последовательно в вертикальные транспортные регистры, а затем — в горизонтальный транспортный регистр. Информация с этого регистра поступает на усилитель, выход которого сопряжен с видеоканалом. Повторение данного процесса для фотоячеек с четными номерами относится к обработке второй области телевизионного кадра. Такое «сканирование» производится 30 раз в 1 с.

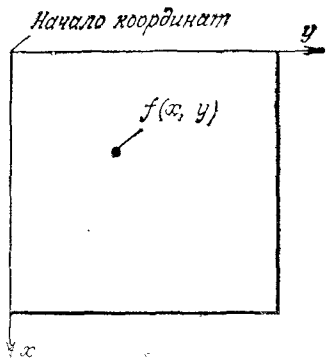


Рис. 7.3. Обозначения координат при описании изображения. Значение произвольной точки (x, y) задается величиной (интенсивностью) функции f в этой точке.

Камеры с линейным сканированием воспроизводят, естественно, только одну линию входного изображения. Эти камеры удобно использовать в случае движения объекта относительно датчика (например, болтов на конвейере). Движение объекта в направлении, перпендикулярном датчику, дает двумерное изображение. Нередко используются датчики линейного сканирования с разрешающим диапазоном, включающим 256—2048 элементов. Разрешающая способность датчиков с плоскостной структурой составляет 32×32 элементов на нижней границе и 256×256 элементов для средних диапазонов. Современные промышленные устройства с высокой разрешающей способностью содержат 480×380 элементов, а экспериментальные ПЗС-датчики имеют 1024×1024 элементов и больше.

Обозначим через $f(x, y)$ двумерное изображение (рис. 7.3), получаемое телевизионной камерой или другим устройством, дающим изображение. Здесь x и y пространственные координаты (т. е. координаты плоскости изображения), а величина f в произвольной точке (x, y) пропорциональна яркости (интенсивности) изображения в этой точке. Переменной z обозначим изменение интенсивности изображения для случая, когда пространственное расположение этих изменений не существенно.

Для получения удобной с точки зрения вычисления формы функции изображения $f(x, y)$ ее необходимо дискретизировать как пространственно, так и по амплитуде (интенсивности). Дискретизацию по пространственным координатам (x, y) будем называть *дискретизацией изображения*, а амплитудную дискретизацию — *квантованием по интенсивности или квантованием по*

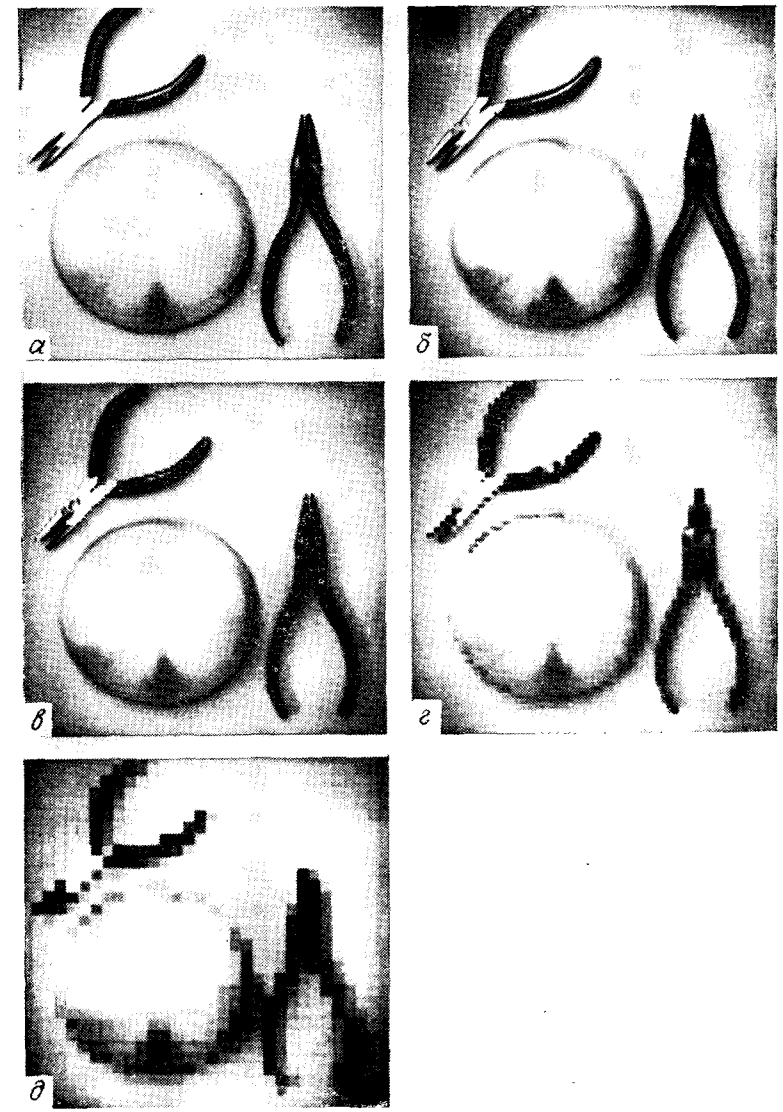


Рис. 7.4. Эффекты уменьшения величины дискретизации. (*а*) 512×512; (*б*) 256×256; (*в*) 128×128; (*г*) 64×64; (*д*) 32×32.

уровню серого. Последний термин применяется для одноцветных изображений и отражает тот факт, что изображение меняется по тону от черного до белого в серых оттенках. Термины интенсивности и уровня серого будем использовать попеременно.

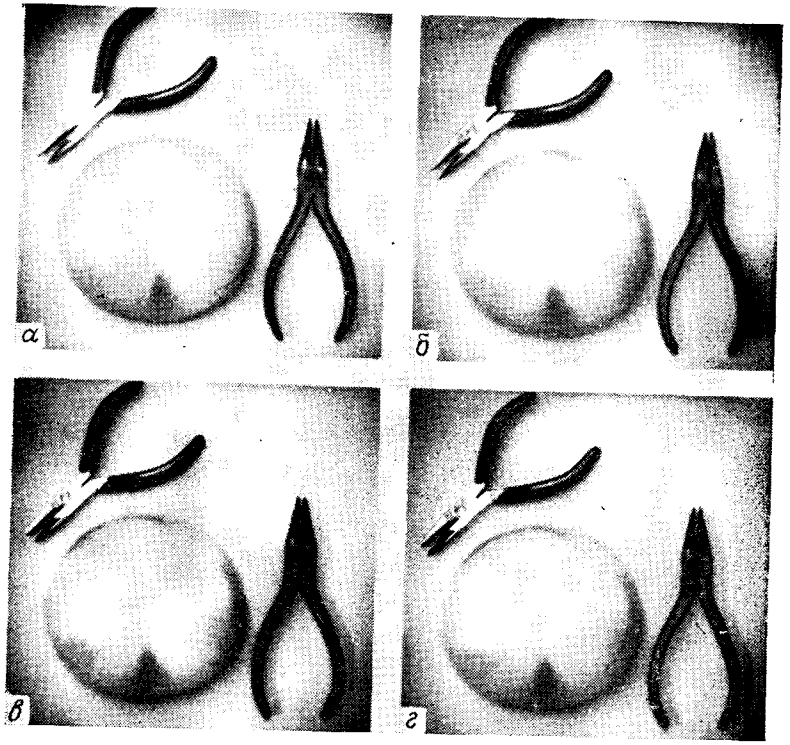


Рис. 7.5. Изображение размером 512×512 соответственно с 256, 128, 64, 32, 16, 8, 4 и 2 уровнями интенсивности.

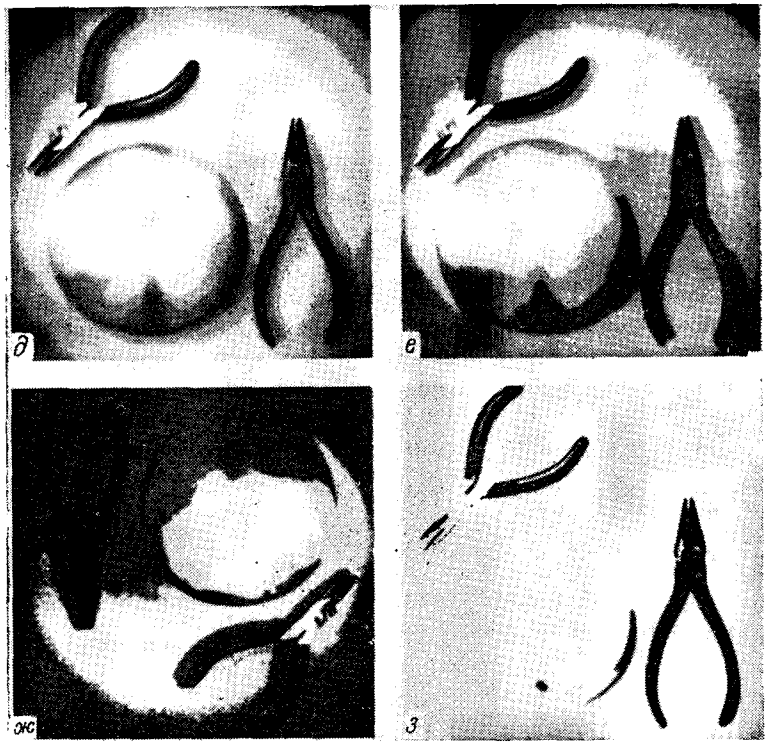
Предположим, что непрерывное изображение дискретизировано равномерно на N рядов и M столбцов, причем каждая дискретная величина проквантована по интенсивности. Такая система, называемая *цифровым изображением*, может быть представлена в виде

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \dots & f(0, M-1) \\ f(1, 0) & f(1, 1) & \dots & f(1, M-1) \\ \dots & \dots & & \dots \\ f(N-1, 0) & f(N-1, 1) & \dots & f(N-1, M-1) \end{bmatrix}, \tag{7.2-1}$$

где и x и y теперь дискретные переменные:

$$x = 0, 1, 2, \dots, N-1; \quad y = 0, 1, 2, \dots, M-1.$$

Каждый элемент системы называется *элементом изображения, элементом картинки* или *пикселом*. В соответствии с рис. 7.3 можно отметить, что $f(0, 0)$ является пикселом начала коорди-



Продолжение рис. 7.5.

нат изображения, $f(0, 1)$ — правый от него пиксел и так далее. Обычно на практике величины N, M и число уровней дискретной интенсивности каждого квантованного пиксела задают в виде целых степеней числа 2.

Для уяснения процессов дискретизации и квантования рассмотрим рис. 7.4. На рис. 7.4, *а* приведено изображение, дискретизированное в систему пикселов размером $N \times N$ с $N = 512$, при этом интенсивность каждого пиксела квантована по одному из 256 дискретных уровней. На рис. 7.4, *б—д* показано то же изображение, но соответственно с $N = 256$, 128, 64 и 32. Везде число уровней интенсивности равнялось 256. Поскольку площадь дисплея для каждого изображения была также одинаковой (512×512 воспроизводящих точек), величина пикселов в изображении с меньшей разрешающей способностью удваивалась по сравнению с предыдущим изображением для того, чтобы заполнить все рабочее поле дисплея. В результате изображение принимает шахматную структуру, которая особенно заметна на картинках с низшей разрешающей способностью. Можно

отметить, что качество изображения размером 256 × 256 довольно близко к приведенному на рис. 7.4, *а*, в то время как для других значений оно резко снижается.

Рис. 7.5 иллюстрирует эффект, появляющийся при уменьшении числа уровней интенсивности в случае постоянного пространственного разрешения 512 × 512. Изображения с числом уровней 256, 128 и 64 достаточно качественны. Изображения с 32 уровнями уже несколько хуже особенно на участках с близкими уровнями интенсивности, что происходит из-за использования слишком малого числа уровней интенсивности, представляющего каждый пиксел. Это еще более заметно при появлении пересекающихся структур, называемых ложными контурами, на изображении с 16 уровнями. При меньшем числе уровней качество резко ухудшается.

Степень дискретизации и число уровней интенсивности, необходимые для получения требуемого в системе воспроизведения изображения объекта, зависят от самого изображения и от вида его использования. Для сравнения можно указать, что для получения качественной черно-белой телевизионной картинки требуется 512 × 512 пикселов со 128 уровнями интенсивности. Как правило, универсальная система технического зрения должна иметь как минимум разрешающую способность порядка 256 × × 256 пикселов с 64 уровнями интенсивности.

7.3. МЕТОДЫ ОСВЕЩЕНИЯ

Освещение объектов служит важным фактором, который влияет на алгоритм получения изображения в целом. Произвольное освещение внешней среды часто не является удовлетворительным, так как оно может привести к пониженной контрастности изображения, к появлению бликов, теней и неинформативных деталей. Правильно организованная система освещения уменьшает сложность получаемого изображения при увеличении информации, требуемой для обнаружения и выделения объекта.

На рис. 7.6 показаны четыре основные схемы, используемые для освещения рабочего пространства робота. Для объектов с гладкими поверхностями правильной формы можно применить метод рассеянного освещения (рис. 7.6, *а*). Такая схема освещения обычно используется в тех случаях, когда важными являются характеристики поверхности объекта (рис. 7.7). Теневое освещение (рис. 7.6, *б*) дает черно-белое (дискретное) изображение. Этот метод целесообразно применять при необходимости распознавания или измерения силуэтов объектов. Пример такого случая показан на рис. 7.8.

Метод структурного освещения (рис. 7.6, *в*) заключается в проецировании на рабочую поверхность световых точек, полос или решеток. Этот метод освещения имеет два важных преимущества. Первое преимущество заключается в упрощении задачи нахождения объекта за счет подачи в рабочее пространство известного светового рисунка, по искажению которого определяется наличие объекта. Второе преимущество — возможность получения пространственных характеристик объекта по анализу формы искажений светового рисунка. Два примера реализации метода структурного освещения приведены на рис. 7.9. В первом случае показан объект, освещенный параллельными плоскостями света, которые образуют световые полосы на пересечении с

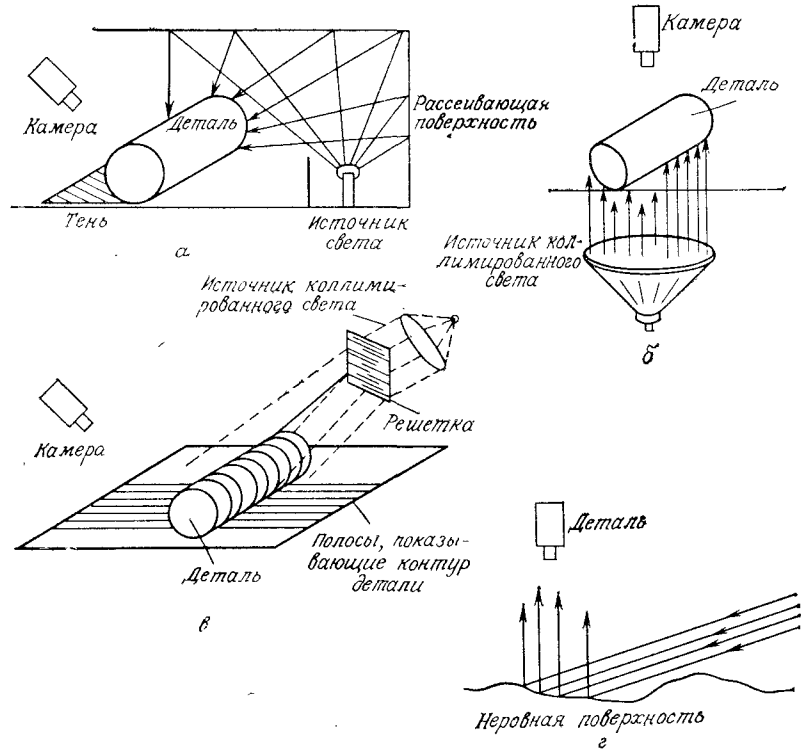


Рис. 7.6. Четыре основные схемы освещения [201].

плоской поверхностью. Во втором случае производится проецирование двух световых плоскостей с разных направлений, но сходящихся на поверхности в одну полосу (рис. 7.10, *а*). Камера линейного сканирования, размещенная над поверхностью и сфокусированная на полосе, при отсутствии объекта будет отображать непрерывную линию света. Эта линия прерывается любым объектом, пересекающим одновременно обе световые плоскости. Данный метод удобно использовать при движении объектов по ленте конвейера перед камерой. Как показано на рис. 7.10, *б*,

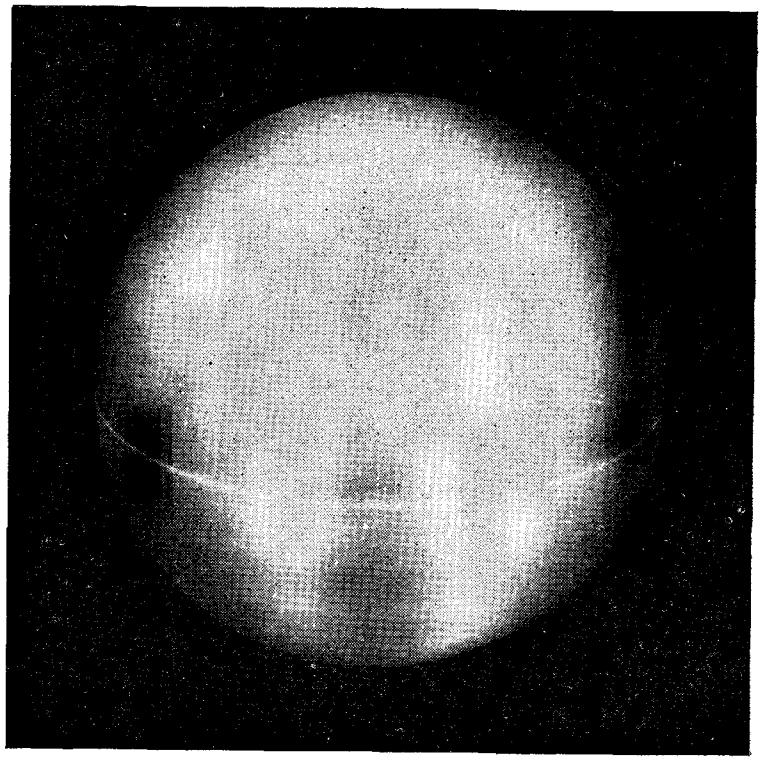


Рис. 7.7. Пример рассеянного освещения.

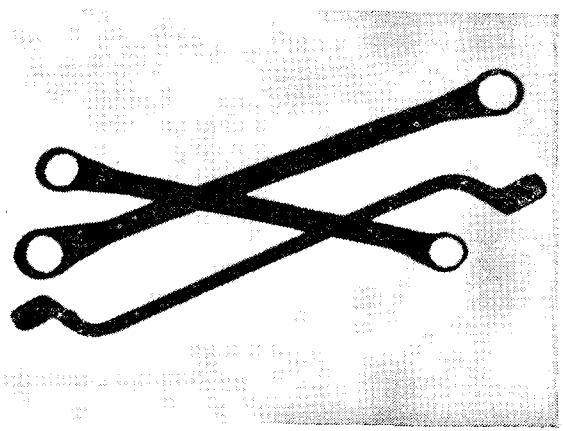


Рис. 7.8. Пример теневого освещения.

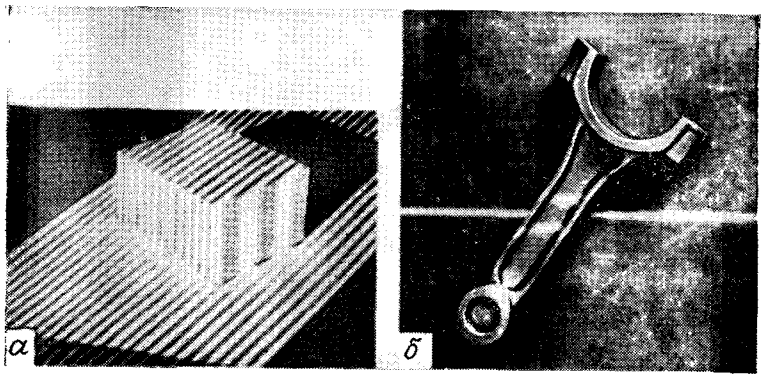


Рис. 7.9. Два примера структурного освещения [250, 203].

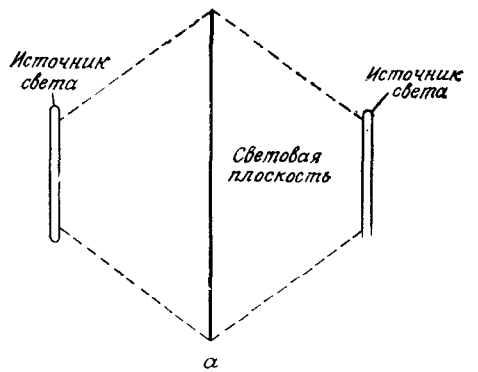


Рис. 7.10. Вид сверху (*а*) на две световые плоскости, пересекающиеся на одной линии. Объект (*б*) будет обнаружен камерой, только если он пересекает обе световые плоскости [121].

наличие двух источников света гарантирует попадание объекта в световую полосу, если она расположена прямо напротив камеры. Отметим, что камера линейного сканирования регистрирует только ту линию, на которой сходятся две световые плоскости, но при движении объекта относительно камеры может быть получена двумерная информация.

Метод направленного освещения (рис. 7.6, *г*) используется в основном для обследования объекта. Такие дефекты поверхности, как впадины и трещины, определяются с помощью точно

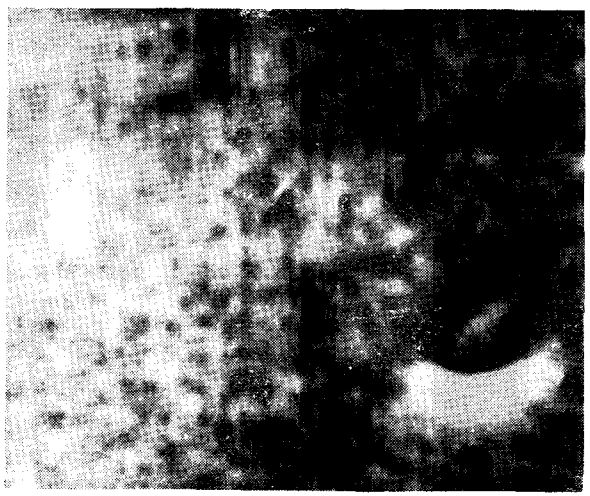


Рис. 7.11. Пример направленного освещения [201].

направленного светового луча (например, лазерного луча) путем измерения величины его рассеивания. Если поверхность без дефектов, то на камеру попадает небольшое количество света. Если же на поверхности имеются дефекты, поток света в сторону камеры увеличивается и позволяет их выявить (рис. 7.11).

7.4. ГЕОМЕТРИЯ ИЗОБРАЖЕНИЯ

Ниже рассматриваются важные преобразования, используемые при получении изображения, математическая модель камеры и ряд аспектов задачи формирования стереоизображения. Некоторые из этих преобразований уже были рассмотрены в гл. 2 в связи с кинематикой манипулятора робота. Здесь обсуждаются сходные проблемы, но с точки зрения получения изображения.

7.4.1. Некоторые основные преобразования

Дадим представления о таких задачах, как вращение, изменение масштаба и смещение (сдвиг) изображений. Все преобразования записываются в трехмерной декартовой системе координат, в которой координаты точки записываются как (X, Y, Z). В случае двумерных изображений для обозначения координат пиксела используется сокращенная запись (x, y). В обычной терминологии выражение (X, Y, Z) представляет собой пространственные координаты точки.

Смещение. Предположим, что требуется сместить точку с координатами (X, Y, Z) в новое место, используя перемещения (X_0, Y_0, Z_0). Смещение легко выполняется по следующим формулам:

$$X^* = X + X_0,\ Y^* = Y + Y_0,\ Z^* = Z + Z_0,$$

где (X^*, Y^*, Z^*) — координаты новой точки. Эти выражения могут быть записаны в матричной форме

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \qquad (7.4\text{-}2)$$

Как показано ниже, для получения сложного изображения часто используется сочетание нескольких преобразований, таких, как смещение после изменения масштаба и вращения. Запись такого процесса существенно упрощается при применении квадратичных матриц. Соответственно уравнение (4.7.2) запишется в следующем виде:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \qquad (7.4\text{-}3)$$

Для параметров X^*, Y^* и Z^* уравнения (7.4-2) и (7.4-3) полностью эквивалентны.

В этом разделе мы будем использовать выражение для объединенной матрицы

$$\mathbf{v}^* = \mathbf{A}\mathbf{v}, \qquad (7.4\text{-}4)$$

где **A** — матрица преобразования размером 4×4, **v** — вектор-столбец, содержащий начальные координаты:

$$\mathbf{v} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{7.4-5}$$

а $\mathbf{v}^*$ — вектор-столбец, элементами которого являются преобразованные координаты:

$$\mathbf{v}^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix}. \tag{7.4-6}$$

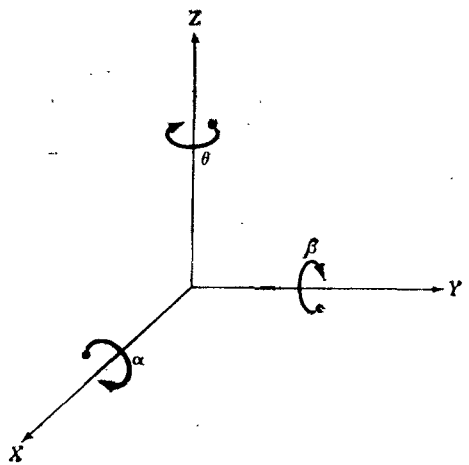


Рис. 7.12. Вращение точки относительно осей координат. Углы измеряются по часовой стрелке, если смотреть вдоль осей вращения на начало координат.

С использованием этих выражений матрица, с помощью которой выполняется смещение, запишется в виде

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{7.4-7}$$

а процесс смещения определяется уравнением (7.4.4), т. е. $\mathbf{v}^* = \mathbf{Tv}$.

Изменение масштаба. Масштабирование на коэффициенты S_x, S_y и S_z по осям X, Y и Z производится с помощью матрицы преобразования

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-8}$$

Вращение. Преобразования, используемые для осуществления трехмерного вращения, значительно сложнее других преобразований. Простейшим видом этих преобразований является вращение вокруг координатных осей. Для вращения заданной точки вокруг произвольной точки в пространстве требуется выполнить три преобразования. Первое преобразование смещает произвольную точку в начало координат, второе — осуществляет вращение, а третье — возвращает точку в исходное положение.

В соответствии с рис. 7.12 вращение точки относительно координатной оси Z на угол θ реализуется с помощью преобразования

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-9}$$

Угол вращения θ измеряется по часовой стрелке, если смотреть на начало координат из точки на оси $+Z$. Отметим, что это преобразование осуществляется только для координат X и Y.

Вращение точки вокруг оси X на угол α выполняется с помощью преобразования

$$\mathbf{R}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-10}$$

Наконец, вращение точки относительно оси Y **на угол** β реализуется **с помощью** преобразования

$$\mathbf{R}_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-11}$$

Взаимные и обратные преобразования. Реализация нескольких преобразований может быть представлена одной матрицей преобразования размерностью 4×4. Например, смещение, масштабирование и вращение точки **v** относительно оси Z записываются в виде

$$\mathbf{v}^* = \mathbf{R}_\theta [\mathbf{S}(\mathbf{Tv})] = \mathbf{Av}, \tag{7.4-12}$$

где **A** — матрица размерностью 4×4: $\mathbf{A} = \mathbf{R}_\theta\mathbf{ST}$.

Необходимо отметить, что эти матрицы обычно не переставляются, поэтому важен порядок их применения.

Хотя приведенное рассмотрение ограничивается преобразованиями одной точки, аналогичные приемы распространяются для одновременных преобразований группы из m точек с помощью одного преобразования. Пусть с учетом уравнения (7.4-5) $\mathbf{v}_1$, $\mathbf{v}_2$, ..., $\mathbf{v}_m$ представляют координаты m точек. Если сформировать матрицу **V** размерностью $4 \times m$, столбцы которой являются вектор-столбцами, то одновременное преобразование всех этих

точек с помощью матрицы преобразования **A** размерностью 4×4 записывается в виде

$$\mathbf{V}^* = \mathbf{AV}. \tag{7.4-13}$$

Полученная матрица $\mathbf{V}^*$ имеет размерность $4 \times m$. Ее i-й столбец $\mathbf{v}_i^*$ содержит координаты преобразованной точки, соответствующей $\mathbf{v}_i$.

В заключение отметим, что из матриц многих рассмотренных преобразований легко получить обратные матрицы, используемые для выполнения обратных преобразований. Например, матрица обратного смещения имеет вид

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-14}$$

Аналогично обратная матрица вращения $\mathbf{R}_\theta^{-1}$ определяется как

$$\mathbf{R}_\theta^{-1} = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) & 0 & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7.4-15}$$

Обращение более сложных матриц преобразования обычно производится численными методами.

7.4.2. Оптические преобразования

Оптические преобразования, которые называются также преобразованиями изображений, проецируют точки трехмерного пространства на плоскость. Оптические преобразования играют основную роль при получении изображения, поскольку они определяют способ отображения пространственных объектов. Хотя ниже в этом разделе оптические преобразования записываются в матричной форме размерностью 4×4, они существенно отличаются от рассмотренных в предыдущем разделе, поскольку являются нелинейными, так как включают операции деления координат.

На рис. 7.13 приведена схема процесса формирования изображения. Предположим, что система координат камеры (x, y, z) имеет плоскость изображения, совпадающую с плоскостью xy, а оптическая ось, установленная по центру линзы, направлена вдоль оси z. Таким образом, центр плоскости изображения является началом координат, а центр линзы имеет координаты $(0, 0, \lambda)$. Если камера сфокусирована на дистанционные объекты, то λ представляет собой фокусное расстояние линзы. Здесь предполагается, что система координат камеры совпадает с декартовой системой координат (X, Y, Z). Это ограничение будет снято в следующем разделе.

Пусть (X, Y, Z) — координаты произвольной точки трехмерного пространства (рис. 7.13). Предположим для последующих рассуждений, что $Z > \lambda$, т. е. что все интересующие нас точки

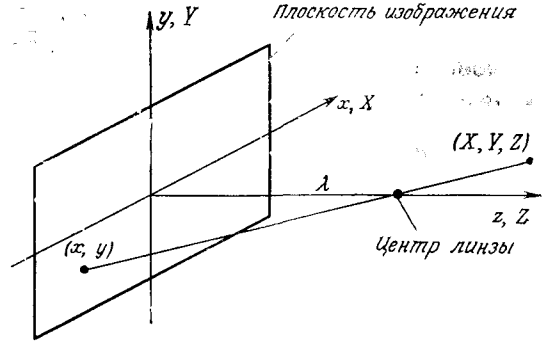


Рис. 7.13. Основная схема процесса получения изображения. Система координат камеры (x, y, z) совпадает с декартовой системой координат (X, Y, Z).

пространства находятся перед линзой. Сначала требуется установить взаимосвязь между точкой (X, Y, Z) и координатами (x, y) ее проекции на плоскость изображения. Это легко сделать, рассмотрев соответствующие треугольники. Из рис. 7.13 следует

$$\frac{x}{\lambda} = -\frac{X}{Z-\lambda} = \frac{X}{\lambda - Z} \tag{7.4-16}$$

и

$$\frac{y}{\lambda} = -\frac{Y}{Z-\lambda} = \frac{Y}{\lambda - Z}, \tag{7.4-17}$$

где минусы перед X и Y означают, что точки на изображении перевернуты.

Спроецированные на плоскость изображения точки трехмерного пространства определяются непосредственно из уравнений (7.4-16) и (7.4-17):

$$x = \frac{\lambda X}{\lambda - Z} \tag{7.4-18}$$

и

$$y = \frac{\lambda Y}{\lambda - Z}. \tag{7.4-19}$$

Важно отметить, что данные уравнения являются нелинейными, потому что они включают операцию деления на перемен-

ную Z. Хотя ими можно пользоваться в приведенном виде, часто удобно применять матричную форму записи, как это делалось в предыдущем разделе для описания процессов вращения, смещения и масштабирования. Такую запись легко получить с помощью однородных координат.

Одиородные координаты точки с декартовыми координатами (X, Y, Z) определяются как (kX, kY, kZ, k), где k — произвольная ненулевая постоянная. Ясно, что обратное преобразование однородных координат в декартовые осуществляется путем деления трех однородных координат на четвертую. Точка в декартовой системе координат определяется в векторной форме следующим образом:

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{7.4-20}$$

а соответствующая запись в однородных координатах будет иметь вид

$$\mathbf{w}_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}. \tag{7.4-21}$$

Определим матрицу оптического преобразования в виде

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix}. \tag{7.4-22}$$

Тогда произведение $\mathbf{P}\mathbf{w}_h$ дает вектор, который обозначим через $\mathbf{c}_h$:

$$\mathbf{c}_h = \mathbf{P}\mathbf{w}_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ \frac{-kZ}{\lambda} + k \end{bmatrix}. \tag{7.4-23}$$

Элементы вектора $\mathbf{c}_h$ являются координатами камеры в однородной форме. Как было отмечено выше, эти координаты могут быть переведены в декартовые координаты путем деления каждого из трех элементов вектора $\mathbf{c}_h$ на четвертый. Таким образом, декартовы координаты любой точки связаны с системой координат камеры следующей векторной формой:

$$\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{\lambda X}{\lambda - Z} \\ \frac{\lambda Y}{\lambda - Z} \\ \frac{\lambda Z}{\lambda - Z} \end{bmatrix}. \tag{7.4-24}$$

Два первых элемента вектора $\mathbf{c}$ являются координатами (x, y) спроецированной на плоскость изображения пространственной точки (X, Y, Z) в соответствии с уравнениями (7.4-18) и (7.4-19). Третий элемент для рассматриваемой схемы (рис. 7.13) не существен. Как показано ниже, этот элемент выступает в качестве свободной переменной в обратном оптическом преобразовании.

Обратное оптическое преобразование переводит точку с плоскости изображения обратно в трехмерное пространство. Таким образом, из уравнения (7.4-23) получим

$$\mathbf{w}_h = \mathbf{P}^{-1}\mathbf{c}_h, \tag{7.4-25}$$

где $\mathbf{P}^{-1}$ определяется в виде

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix}. \tag{7.4-26}$$

Предположим, что заданная точка на изображении имеет координаты $(x_0, y_0, 0)$, где 0 по координате z отражает тот факт, что плоскость изображения находится на $z = 0$. Эта точка может быть записана в однородной форме вектором

$$\mathbf{c}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix}. \tag{7.4-27}$$

Используя уравнение (7.4-25), получим соответствующий вектор для однородных пространственных координат

$$\mathbf{w}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \tag{7.4-28}$$

или для декартовых координат

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix}. \tag{7.4-29}$$

Поскольку $Z = 0$, данное выражение верно не только для одной точки пространства. Это объясняется тем, что отображение трехмерного пространства на плоскость изображения является преобразованием многих точек в одну. Точка изображения (x_0, y_0) соответствует ряду пространственных точек, лежащих на одной линии, которая проходит через $(x_0, y_0, 0)$ и $(0, 0, \lambda)$. Уравнения этой линии в декартовой системе координат получаются из уравнений (7.4-18) и (7.4-19) в виде

$$X = \frac{x_0}{\lambda}(\lambda - Z) \tag{7.4-30}$$

и

$$Y = \frac{y_0}{\lambda}(\lambda - Z). \tag{7.4-31}$$

Из этих уравнений видно, что если нет достаточной информации о пространственной точке (например, ее координаты Z), полученной из точки на изображении, то нельзя точно восстановить точку в трехмерном пространстве по ее изображению. Это вполне естественное соображение может быть использовано для определения обратного оптического преобразования путем простой подстановки элемента z в вектор $\mathbf{c}_h$ вместо 0 в качестве свободной переменной. Таким образом, получим

$$\mathbf{c}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ k \end{bmatrix}. \tag{7.4-32}$$

Теперь из уравнения (7.4-25) имеем

$$\mathbf{w}_h = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ \frac{kz}{\lambda} + k \end{bmatrix}. \tag{7.4-33}$$

После преобразования в декартовые координаты получим

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{\lambda x_0}{\lambda + z} \\ \frac{\lambda y_0}{\lambda + z} \\ \frac{\lambda z}{\lambda + z} \end{bmatrix}. \tag{7.4-34}$$

Другими словами, используя z как свободную переменную, получим уравнения

$$X = \frac{\lambda x_0}{\lambda + z},$$
$$Y = \frac{\lambda y_0}{\lambda + z}, \tag{7.4-35}$$
$$Z = \frac{\lambda z}{\lambda + z}.$$

Выразив z в последнем уравнении через Z и подставляя его в два первых уравнения, получим

$$X = \frac{x_0}{\lambda}(\lambda - Z), \tag{7.4-36}$$

$$Y = \frac{y_0}{\lambda}(\lambda - Z), \tag{7.4-37}$$

что согласуется с тем, что для восстановления точки в трехмерном пространстве путем обратного оптического преобразования требуется знание по крайней мере одной декартовой координаты точки. Эта проблема обсуждается также в разд. 7.4.5.

7.4.3. Модель камеры

Уравнения (7.4-23) и (7.4-24) характеризуют формирование изображения путем проекции точки пространства на плоскость изображения. Таким образом, эти два уравнения представляют основную математическую модель воспроизводящей камеры. Данная модель базируется на предположении, что система координат камеры и декартова система координат совмещены. В этом разделе рассмотрим более общий случай, при котором две системы координат могут быть разделены. Однако главная цель получения координат произвольной точки пространства на плоскости остается той же.

В случае, изображенном на рис. 7.14, декартова система координат (X, Y, Z) используется для определения положения как камеры, так и точек пространства, обозначенных через $\mathbf{w}$. На этом рисунке также показаны система координат камеры (x, y, z) и точки изображения, обозначенные через $\mathbf{c}$. Предпо-

лагается, что камера установлена на шарнире, который обеспечивает поворот по углу θ и наклон по углу α. Здесь поворот определяет угол между осями x и X, а наклон — угол между осями z и Z. Смещение центра шарнира от начала декартовой

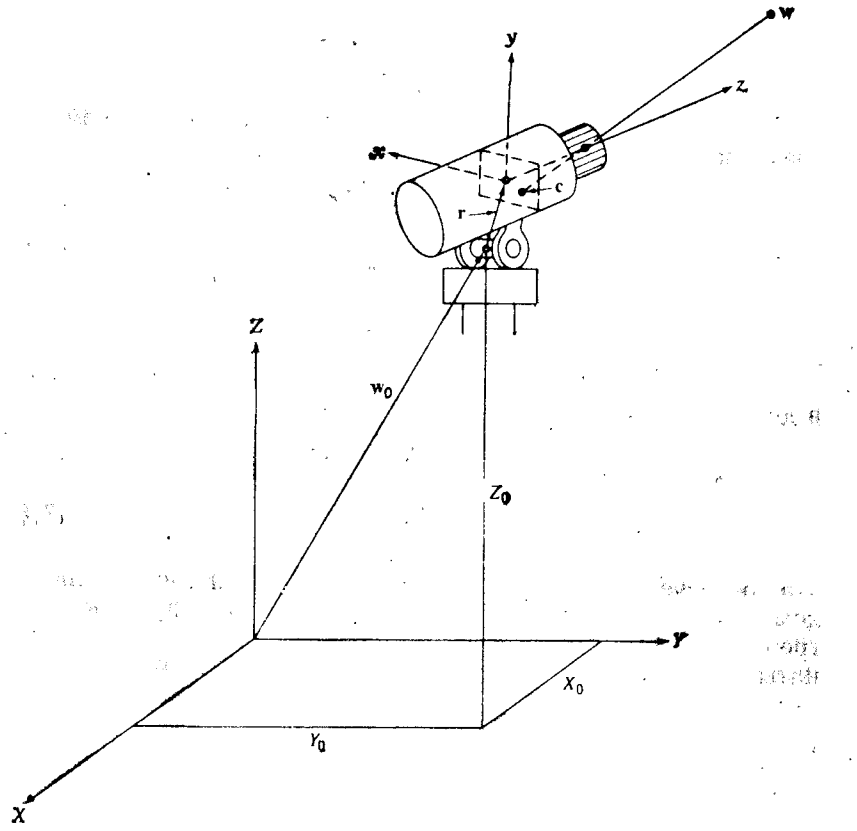


Рис. 7.14. Геометрическое расположение элементов при получении изображения в двух системах координат.

системы координат обозначено вектором $\mathbf{w}_0$, а смещение центра плоскости изображения относительно центра шарнира — вектором $\mathbf{r}$ с элементами (r_1, r_2, r_3).

Весь необходимый материал для получения модели камеры, основанной на геометрическом расположении элементов, которые показаны на рис. 7.14, изложен в двух предыдущих разделах. Метод заключается в приведении в соответствие системы координат камеры и декартовой системы координат путем ряда преобразований. После этого можно легко использовать оптическое преобразование, заданное уравнением (7.4-22) для по-

лучения координат любой точки пространства на плоскости изображения. Другими словами, перед использованием оптического преобразования задача предварительно сводится к геометрической системе, приведенной на рис. 7.13.

Предположим, что камера сначала находится в нормальном положении, т. е. центр шарнира и начало координат плоскости изображения совмещены с началом декартовой системы координат и все их оси совпадают. Перевод из нормального положения к геометрическому расположению, показанному на рис. 7.14, может быть осуществлен несколькими путями. Зададим следующую последовательность операций: 1) сдвиг центра шарнира относительно начала координат, 2) поворот оси x, 3) наклон оси z, 4) сдвиг плоскости изображения относительно центра шарнира.

Ясно, что указанная последовательность механических операций не обеспечивает необходимое перемещение всех точек пространства, поскольку вид группы точек, воспроизводимый камерой после их перемещения из нормального положения, будет совершенно другой. Однако можно вернуть все точки пространства обратно в нормальное положение, применяя к ним ту же последовательность операций. Поскольку расположение камеры в нормальном положении соответствует рис. 7.13, при реализации оптического преобразования задача сводится к применению ряда указанных операций соответственно к каждой точке пространства.

Сдвиг начала декартовой системы координат в центр шарнира выполняется с помощью следующей матрицы преобразований:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (7.4\text{-}38)$$

Другими словами, однородная точка пространства $\mathbf{w}_h$ с координатами (X_0, Y_0, Z_0) после преобразования $\mathbf{G}\mathbf{w}_h$ является началом новой системы координат.

Как указывалось выше, угол поворота измеряется между осями x и X. В нормальном положении эти оси совпадают. В результате поворота оси x на заданный угол производится вращение по углу θ. Вращение производится относительно оси z и осуществляется с использованием матрицы преобразования $\mathbf{R}_\theta$, заданной уравнением (7.4-9), т. е. применение этой матрицы ко всем точкам, включая точку $\mathbf{G}\mathbf{w}_h$, позволяет эффективно поворачивать ось x в требуемое положение. При использовании уравнения (7.4-9) важно иметь в виду условие, соответствующее рис. 7.12, а именно положительные углы получаются при

вращении точек по часовой стрелке, что предполагает вращение камеры против часовой стрелки вокруг оси z. Неподвижное (0°) положение соответствует случаю совпадения осей x и X.

До этого момента оси z и Z еще совпадают. Поскольку наклон определяется углом между этими двумя осями, поворачиваем камеру на угол α путем вращения оси z на данный угол. Вращение производится относительно оси x с использованием матрицы преобразования $\mathbf{R}_\alpha$ (уравнение (7.4-10)) для всех точек, включая точку $\mathbf{R}_\theta\mathbf{G}\mathbf{w}_h$. Как и раньше, вращение камеры против часовой стрелки дает положительные углы, а поворот в 0° получается при совпадении осей z и Z [1].

В соответствии с разделом 7.4.4 две матрицы вращения могут быть совмещены в одной матрице $\mathbf{R} = \mathbf{R}_\alpha\mathbf{R}_\theta$.

Далее из уравнений (7.4-9) и (7.4-10) имеем

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta\cos\alpha & \cos\theta\cos\alpha & \sin\alpha & 0 \\ \sin\theta\sin\alpha & -\cos\theta\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (7.4\text{-}39)$$

Наконец, смещение начала координат плоскости изображения на вектор $\mathbf{r}$ производится с помощью матрицы преобразования

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (7.4\text{-}40)$$

Таким образом, применяя к точке $\mathbf{w}_h$ ряд преобразований $\mathbf{CRG}\mathbf{w}_h$, совмещаем декартову систему координат и систему координат камеры. Координаты точки $\mathbf{w}_h$ на плоскости изображения окончательно получают с использованием уравнения (7.4-22). Другими словами, однородная точка пространства, которая фиксируется камерой, расположенной в соответствии с рис. 7.14, имеет следующую запись в системе однородных координат камеры:

$$\mathbf{c}_h = \mathbf{PCRG}\mathbf{w}_h. \qquad (7.4\text{-}41)$$

Это уравнение представляет оптическое преобразование, включающее две системы координат.

Как показано в разд. 7.4.2, декартовы координаты (x, y) точки изображения получаются путем деления первого и второго элементов вектора $\mathbf{c}_h$ на четвертый. Расписывая уравнение (7.4-41) и возвращаясь к декартовой системе координат, получим

$$x = \lambda \frac{(X - X_0)\cos\theta + (Y - Y_0)\sin\theta - r_1}{-(X - X_0)\sin\theta\sin\alpha + (Y - Y_0)\cos\theta\sin\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda} \qquad (7.4\text{-}42)$$

и

$$y = \lambda \frac{-(X - X_0)\sin\theta\cos\alpha + (Y - Y_0)\cos\theta\cos\alpha + (Z - Z_0)\sin\alpha - r_2}{-(X - X_0)\sin\theta\sin\alpha + (Y - Y_0)\cos\theta\sin\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda}. \qquad (7.4\text{-}43)$$

Эти уравнения определяют координаты изображения точки $\mathbf{w}$, пространственные координаты которой (X, Y, Z). Отметим, что данные уравнения приводятся к уравнениям (7.4-18) и (7.4-19) при $X_0 = Y_0 = Z_0 = 0$, $r_1 = r_2 = r_3 = 0$ и $\alpha = \theta = 0°$.

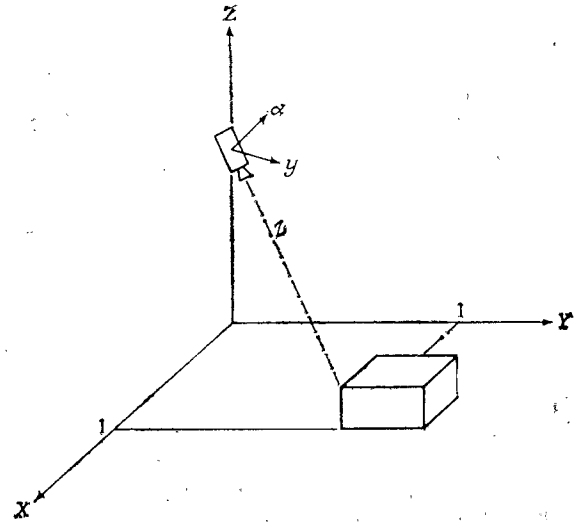


Рис. 7.15. Камера, направленная на пространственный объект.

Пример. Для иллюстрации рассмотренного метода предположим, что требуется найти изображение угла блока, показанного на рис. 7.15. Камера смещена от начала координат и направлена на объект с поворотом на 135° и наклоном в 135°. Условимся, что изменение углов происходит в положительном направлении при вращении камеры против часовой стрелки, если смотреть на начало координат вдоль оси вращения.

Воспроизведем подробно операции, необходимые для перемещения камеры из нормального положения к положению, по-

[1] Наглядность этих преобразований можно обеспечить, сконструировав систему осей (например, с помощью устройств для чистки трубок), пометив оси x, y, z и произведя поочередное вращение осей вручную.

казанному на рис. 7.15. Камера изображена в нормальном положении на рис. 7.16, *а* и в смещенном относительно начала координат положении на рис. 7.16, *б*. Важно отметить, что после этой операции оси декартовой системы координат используются только для угловых отсчетов, т. е. после перемещения декартовой системы координат все вращения производятся относитель-

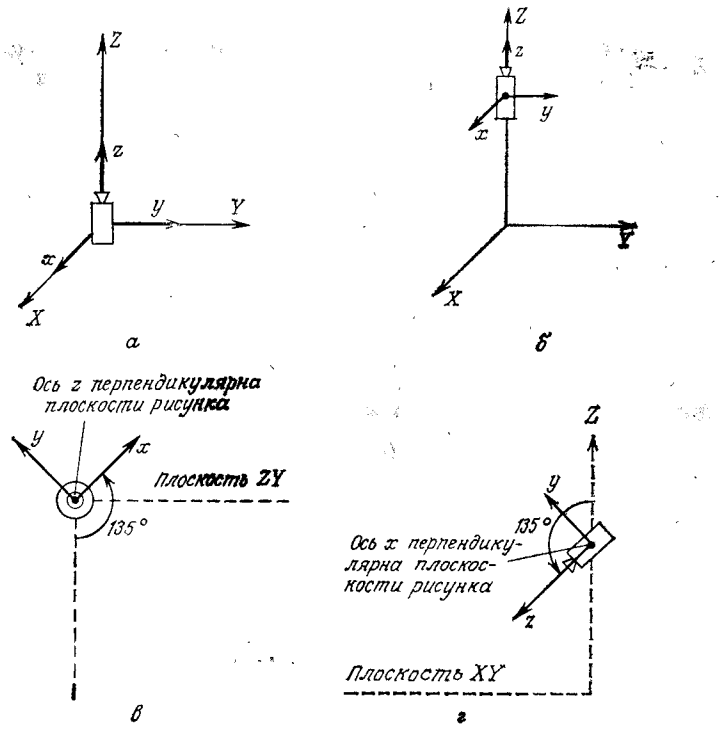


Рис. 7.16. (*а*) Камера в нормальном положении. (*б*) Смещение центра шарнира из начала координат. (*в*) Вращение вокруг оси *z* при повороте. (*г*) Вращение вокруг оси *x* при наклоне.

но новых осей (осей камеры). На рис. 7.16, *в* показан вид вдоль оси *z* камеры при осуществлении поворота. В этом случае вращение камеры вокруг оси *z* происходит против часовой стрелки, поэтому точки пространства вращаются относительно этой оси в противоположном направлении по положительному направлению угла θ. На рис. 7.16, *г* показано положение системы вдоль оси *x* камеры перед осуществлением наклона после завершения поворота. Вращение вокруг этой оси происходит против часовой стрелки по положительному направлению угла α. Оси декартовой системы координат показаны на рис. 7.16, *в* и *г* штриховыми

линиями, чтобы подчеркнуть их значение только как базы отсчета при реализации поворота и наклона. На рис. 7.16, *г* не отражена окончательная операция смещения плоскости изображения относительно центра шарнира.

В задаче используются следующие параметры: $X_0 = 0$ м, $Y_0 = 0$ м, $Z_0 = 1$ м, $\alpha = 135°$, $\theta = 135°$, $r_1 = 0{,}03$ м, $r_2 = r_3 = 0{,}02$ м, $\lambda = 35$ мм $= 0{,}035$ м.

Искомый угол имеет координаты $(X, Y, Z) = (1; 1; 0{,}2)$.

Для того чтобы подсчитать координаты изображения угла блока, достаточно просто подставить заданные значения параметров в уравнения (7.4-42) и (7.4-43), т. е.

$$x = \lambda \frac{-0{,}03}{-1{,}53 + \lambda} \quad \text{и} \quad y = \lambda \frac{-0{,}42}{-1{,}53 + \lambda}.$$

Подставляя $\lambda = 0{,}035$, получим координаты изображения $x = 0{,}0007$ м и $y = 0{,}009$ м.

Отметим, что эти координаты легко укладываются на плоскость изображения размером $0{,}025 \times 0{,}025$ м. Если же, например, применять линзу с фокусным расстоянием 200 мм, то просто показать, что полученное изображение угла блока будет находиться вне эффективного поля зрения камеры.

Наконец, отметим, что все соответствующие уравнениям (7.4-42) и (7.4-43) координаты получены относительно центра плоскости изображения. При действии ранее установленного условия о расположении начала координат плоскости изображения в верхнем левом углу необходимо изменить значения координат.

7.4.4. Калибровка камеры

В разделе 7.4.3 были получены точные уравнения для координат (x, y) изображения пространственной точки **w**. В соответствии с уравнениями (7.4-42) и (7.4-43) использование этих уравнений требует знания фокусного расстояния, смещений камеры и углов поворота и наклона. Хотя эти параметры могут быть измерены непосредственно, иногда бывает удобно (например, при частом перемещении камеры) определять один или несколько из указанных параметров, используя саму камеру в качестве измерительного прибора. Это требует наличия ряда точек изображения с известными декартовыми координатами. Процесс вычисления параметров камеры, использующий эти известные точки, часто называют *калибровкой камеры*.

Пусть в соответствии с уравнением (7.4-41) $\mathbf{A} = \mathbf{PCRG}$. Элементы **A** содержат все параметры камеры, а из уравнения (7.4-41) известно, что $\mathbf{c}_h = \mathbf{A}\mathbf{w}_h$. Полагая $k = 1$, можно записать

в однородной форме

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad \textbf{(7.4-44)}$$

Из двух предыдущих разделов следует, что координаты камеры в декартовой системе задаются в виде

$$x = \frac{c_{h1}}{c_{h4}} \quad (7.4\text{-}45)$$

и

$$y = \frac{c_{h2}}{c_{h4}}. \quad (7.4\text{-}46)$$

Подставляя $c_{h1} = xc_{h4}$ и $c_{h2} = yc_{h4}$ в уравнение (7.6-44) и раскрывая матричное произведение, получим

$$\begin{aligned} xc_{h4} &= a_{11}X + a_{12}Y + a_{13}Z + a_{14}, \\ yc_{h4} &= a_{21}X + a_{22}Y + a_{23}Z + a_{24}, \\ c_{h4} &= a_{41}X + a_{42}Y + a_{43}Z + a_{44}, \end{aligned} \quad (7.4\text{-}47)$$

где выражение для c_{h3} может быть опущено, так как оно связано с z.

Подставляя c_{h4} в первые два уравнения (7.4-47), получим два уравнения с двенадцатью неизвестными коэффициентами

$$a_{11}X + a_{12}Y + a_{13}Z - a_{41}xX - a_{42}xY - a_{43}xZ - a_{44}x + a_{14} = 0, \quad (7.4\text{-}48)$$

$$a_{21}X + a_{22}Y + a_{23}Z - a_{41}yX - a_{42}yY - a_{43}yZ - a_{44}y + a_{24} = 0. \quad (7.4\text{-}49)$$

Далее процесс калибровки состоит из трех этапов. Первый этап заключается в определении $m \geqslant 6$ точек пространства с известными координатами (X_i, Y_i, Z_i), $i = 1, 2, \ldots, m$. Поскольку имеются два уравнения, включающие координаты этих точек, то их необходимое количество должно по крайней мере равняться шести. На втором этапе получают соответствующие изображения этих точек (x_i, y_i), $i = 1, 2, \ldots, m$, с помощью камеры. Третий этап состоит в использовании полученных результатов в уравнениях (7.4-48) и (7.4-49) для определения неизвестных коэффициентов. Для нахождения оптимального решения линейной системы уравнений типа (7.4-48) и (7.4-49) существует много численных методов [218].

7.4.5. Стереоизображение

Как указано в разд. 7.4.2, проецирование трехмерного пространства на плоскость изображения является преобразованием, при котором несколько точек сливаются в одну. Это означает, что точка изображения неоднозначно определяет расположение соответствующей точки пространства. В данном разделе показывается, что информация о глубине изображения может быть получена при использовании методов стереоскопического изображения (или стереоизображения).

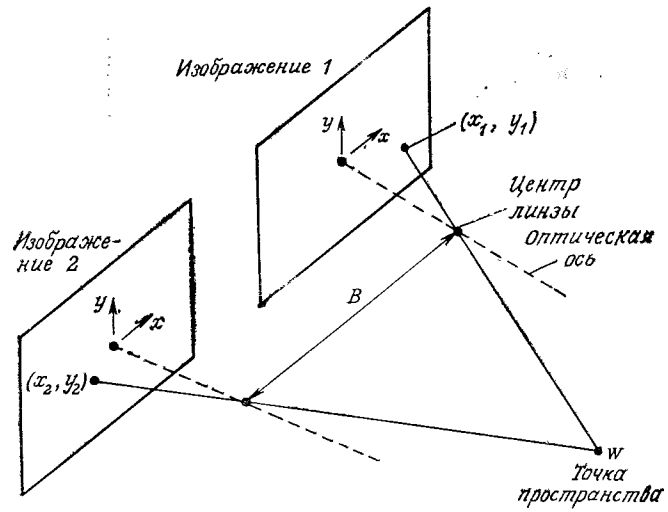


Рис. 7.17. Схема получения стереоизображения.

Стереоизображение включает два отдельных вида изображаемого объекта (рис. 7.17), например пространственной точки **w**. Расстояние между центрами двух линз называется *базовой линией*. Требуется определить координаты (X, Y, Z) точки **w**, заданной точками ее изображения (x_1, y_1) и (x_2, y_2).

Предполагается, что камеры идентичны и системы координат обеих камер полностью совпадают, отличаясь только расположением их начал. Эти условия обычно встречаются на практике. Как и прежде, считается, что при совмещении системы координат камеры с декартовой системой координат плоскость изображения xy совпадает с плоскостью XY декартовой системы координат. При сделанных предположениях координата Z точки **w** одинакова для обеих систем координат камер.

Допустим, что первая камера совмещена с декартовой системой координат, как показано на рис. 7.18. Тогда по уравне-

нию (7.4-31) точка w лежит на линии с координатами

$$X_1 = \frac{x_1}{\lambda}(\lambda - Z_1), \tag{7.4-50}$$

где индексы у X и Z обозначают, что к началу декартовой системы координат передвинута первая камера, а вторая камера и точка **w** также переместятся в этой системе. При этом сохранено относительное расположение элементов системы, показанное на рис. 7.17. Если вместо этого к началу декартовой системы

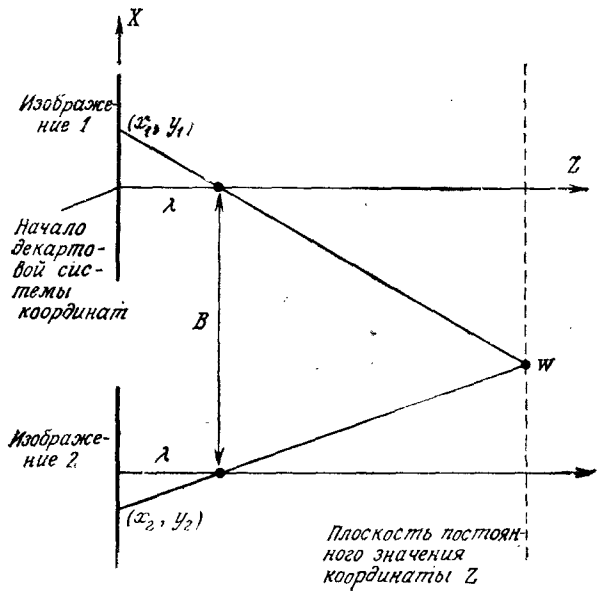


Рис. 7.18. Вид сверху на рис. 7.17 при совмещении первой камеры с декартовой системой координат.

координат передвинута вторая камера, то точка **w** лежит на линии с координатами

$$X_2 = \frac{x_2}{\lambda}(\lambda - Z_2). \tag{7.4-51}$$

Однако благодаря наличию расстояния между камерами и тому, что координаты Z точки **w** одинаковы в обеих системах координат камер, имеем

$$X_2 = X_1 + B \tag{7.4-52}$$

и

$$Z_2 = Z_1 = Z, \tag{7.4-53}$$

где, как отмечено выше, B — базовая линия.

Подставляя уравнения (7.4-52) и (7.4-53) в уравнения (7.4-50) и (7.4-51), получим уравнения

$$X_1 + B = \frac{x_2}{\lambda}(\lambda - Z) \tag{7.4-54}$$

и

$$X_1 = \frac{x_1}{\lambda}(\lambda - Z). \tag{7.4-55}$$

Вычитая уравнение (7.4-55) из уравнения (7.4-54) и решая его относительно Z, получим

$$Z = \lambda - \frac{\lambda B}{x_2 - x_1}. \tag{7.4-56}$$

Отсюда видно, что координата Z точки **w** легко вычисляется при известной разности между соответствующими координатами x_2 и x_1 изображения, а также значений базовой линии и фокусного расстояния. Декартовы координаты X и Y получаются затем прямо из уравнений (7.4-30) и (7.4-31) с использованием координат (x_1, y_1) или (x_2, y_2).

Наибольшую трудность при применении уравнения (7.4-56) для получения координаты Z представляет нахождение двух соответствующих точек на различных изображениях одного объекта. Поскольку эти точки обычно находятся в непосредственной близости, часто используют метод, при котором на одном изображении выделяют точку внутри малой окрестности, а затем пытаются определить соответствующую окрестность на другом изображении корреляционными методами, изложенными в гл. 8. Процесс сопоставления обычно ускоряется, если объект имеет характерные признаки, такие, как выступающие углы.

В заключение отметим, что процесс калибровки, рассмотренный в предыдущем разделе, может непосредственно использоваться для стереоизображения простой его реализацией независимо для каждой камеры.

7.5. НЕКОТОРЫЕ ОСНОВНЫЕ ВЗАИМОСВЯЗИ МЕЖДУ ПИКСЕЛАМИ

Ниже рассматривается несколько простых, но важных взаимосвязей между пикселами в цифровом изображении. Как и в предыдущем разделе, изображение будем обозначать $f(x, y)$. При обсуждении отдельного пиксела будем использовать сокращенные обозначения p и q. Для подгруппы пикселов используется обозначение S.

7.5.1. Соседние пикселы

Пиксел p с координатами (x, y) имеет четыре горизонтальных и вертикальных соседних пиксела с координатами

$$(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1).$$

Эта группа пикселов, называемая «четыре соседа p», обозначается через $N_4(p)$. Отметим, что данные четыре пиксела находятся на одном расстоянии от (x, y), а также некоторые из соседних пикселов p могут быть за пределами цифрового изображения, если (x, y) находится на границе изображения.

Четыре диагональных соседних пиксела p имеют координаты

$$(x+1, y+1), \quad (x+1, y-1), \quad (x-1, y+1), \quad (x-1, y-1)$$

и обозначаются через $N_D(p)$. Эти точки вместе с четырьмя указанными выше называются «восемь соседей p» и обозначаются через $N_8(p)$. Некоторые из точек $N_D(p)$ и $N_8(p)$ также могут выходить за пределы изображения, если (x, y) находится на границе изображения.

7.5.2. Связи

Пусть V — ряд значений интенсивности пикселов, которые могут быть соединены. Например, если требуется связать пикселы с интенсивностью 59, 60 и 61, то $V = \{59, 60, 61\}$. Рассмотрим три типа связей:

1. *Четырехсвязный.* Два пиксела p и q со значениями интенсивностей из V являются четырехсвязными, если q относится к к группе $N_4(p)$.

2. *Восьмисвязный.* Два пиксела p и q со значениями интенсивностей из V являются восьмисвязными, если q относится к группе $N_8(p)$.

3. *m-связный* (смешанная связь). Два пиксела p и q со значениями интенсивности из V являются m-связными, если

а) q относится к группе $N_4(p)$;

б) q относится к группе $N_D(p)$ и множество $N_4(p) \cap N_4(q)$ — пустое. (Это множество пикселов, являющихся четырьмя соседними как по отношению к p, так и по отношению к q со значениями интенсивностей из V.)

Смешанная связь является модификацией восьмисвязного типа систем и вводится для исключения множественности соединений, которая часто вызывает трудности при использовании восьмисвязных систем. Рассмотрим для примера систему пикселов, показанную на рис. 7.19, *а*. Предположим, что $V = \{1, 2\}$. Связи восьми соседних пикселов с пикселом, имеющим значение 2, обозначим штриховыми линиями (рис. 7.19, *б*). Отметим неопределенность полученных связей ввиду их множественности. Эта неопределенность снимается при использовании m-связной системы (рис. 7.19, *в*).

Пиксел p примыкает к пикселу q, если они связаны. Можно подразделить примыкание на 4-, 8- и m-размерное в зависимости от определенного типа связей. Два изображения подгрупп S_1 и S_2 примыкают друг к другу, если несколько пикселов из S_1 примыкают к нескольким пикселам из S_2.

Путь от пиксела p с координатами (x, y) к пикселу q с координатами (s, t) представляет собой последовательность определенных пикселов с координатами $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, где $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$, (x_i, y_i) — пиксел, прилежащий к (x_{i-1}, y_{i-1}), $1 \leqslant i \leqslant n$, а n — длина пути. Можно подразделить пути на 4-, 8- и m-размерные в зависимости от типа используемого примыкания.

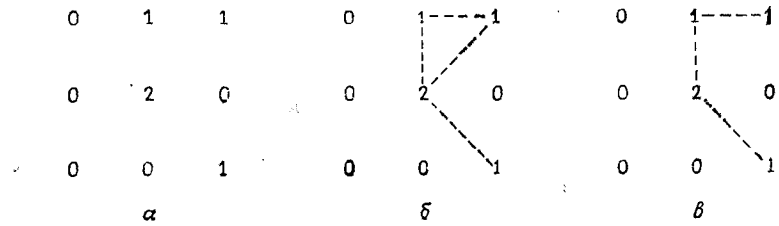


Рис. 7.19. Система пикселов (*а*). Связь восьми соседних пикселов с пикселом со значением 2 (*б*). Связь m-соседних пикселов с пикселом со значением 2 (*в*).

Если p и q являются пикселами изображения одной подгруппы S, то p считается связанной с q в S, если путь от p до q состоит только из пикселов, лежащих в S.

Для любого пиксела p, принадлежащего S, группа пикселов в S, связанных с p, называется связанным компонентом S. Отсюда следует, что два любых пиксела связанного компонента связаны друг с другом, а несвязанного — разделены.

7.5.3. Измерение расстояний

Определим для пикселов p, q, z соответственно с координатами (x, y), (s, t) и (u, v) функцию расстояния или метрику D следующим образом:

1) $D(p, q) \geqslant 0$ [$D(p, q) = 0$, если $p = q$],
2) $D(p, q) = D(q, p)$,
3) $D(p, z) \leqslant D(p, q) + D(q, z)$.

Евклидово расстояние между p и q определяется по формуле

$$D_e(p, q) = [(x-s)^2 + (y-t)^2]^{1/2}. \qquad (7.5\text{-}1)$$

При измерении этого расстояния пикселы, имеющие расстояние, меньшее или равное некоторой величине r от (x, y), располагаются в окружности радиусом r с центром в (x, y). Расстояние D_4, называемое *модульным*, между p и q определяется выражением

$$D_4(p, q) = |x-s| + |y-t|. \qquad (7.5\text{-}2)$$

В этом случае пикселы, имеющие расстояние D_4, меньшее или равное некоторой величине r от (x, y), образуют ромбовидную структуру с центром в (x, y). Например, пикселы с расстоянием $D_4 \leqslant 2$ от (x, y) (центральной точки) образуют следующие контуры с равными от центра расстояниями:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

Отметим, что пикселы с $D_4 = 1$ являются четырьмя соседними к (x, y).

Расстояние D_8, называемое также *шахматным*, между p и q определяется по формуле

$$D_8(p, q) = \max(|x - s|, |y - t|). \qquad (7.5\text{-}3)$$

При этом пикселы с расстоянием D_8, меньшим или равным некоторой величине r, образуют квадрат с центром в (x, y). Например, пикселы с расстоянием $D_8 \leqslant 2$ от центральной точки (x, y) образуют следующие равноудаленные от центра контуры:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

Пикселы с $D_8 = 1$ составляют восемь соседних к точке (x, y) пикселов.

Можно отметить, что расстояние D_4 между двумя точками p и q равно кратчайшему из четырех путей между ними. То же самое относится к расстоянию D_8. В действительности можно определять расстояния D_4 и D_8 между p и q независимо от наличия связующих их путей, поскольку для нахождения этих расстояний требуются только значения координат данных точек. Однако, когда имеют дело с m-связностью, величина расстояния (длина пути) между двумя пикселами зависит от значения пикселов вдоль пути, а также от их соседей. Рассмотрим для примера систему пикселов, в которой p_1, p_2 и p_4 имеют значения 0 или 1

$$\begin{array}{cc} p_3 & p_4 \\ p_1 & p_2 \\ p & \end{array}$$

Если предположить связность пикселов со значением 1 при p_1 и p_3 равными 0, то m — расстояние между p и p_4 равно 2. Если p_1 или p_3 равно 1, расстояние равно 3. Если же p_1 и p_3 равно 1, то расстояние будет равно 4.

7.6. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ИНФОРМАЦИИ

Ниже рассматриваются некоторые методы предварительной обработки информации, используемые в системах технического зрения роботов. Хотя число методов, пригодных для предварительной обработки основных параметров изображения, довольно велико, требованиям по скорости вычислений и по эксплуатационной стоимости, лежащим в основе систем технического зрения, удовлетворяет только определенная подгруппа этих методов. Рассматриваемые ниже методы предварительной обработки информации являются типичными с точки зрения удовлетворения указанным требованиям.

7.6.1. Основные понятия

Рассмотрим два основных подхода к предварительной обработке информации. Первый подход основан на методах пространственной области, а второй — на методах частотной области с использованием преобразования Фурье. Вместе эти подходы охватывают большинство из существующих алгоритмов предварительной обработки информации, применяемых в системах технического зрения роботов.

Методы пространственной области. К пространственной области относится совокупность пикселов, составляющих изображение. Методы пространственной области являются процедурами, оперирующими непосредственно с этими пикселами. Функции предварительной обработки в пространственной области записываются в виде

$$g(x, y) = h[f(x, y)], \qquad (7.6\text{-}1)$$

где $f(x, y)$ — входное изображение, $g(x, y)$ — выходное (обработанное) изображение, а h — оператор функции f, определенный в некоторой области (x, y). Оператор h можно применять также к ряду входных изображений для формирования, например, суммы пикселов K изображений при уменьшении шума (разд. 7.6.2).

Основным подходом при определении окрестности точки (x, y) является использование квадратной или прямоугольной области части изображения с центром в точке (x, y) (рис. 7.20). Центр этой части изображения перемещается от пиксела к пикселу, начиная, например, от левого верхнего угла; при этом для получения $g(x, y)$ оператор применяется для каждого положе-

ния (x, y). Хотя иногда используются и другие формы окрестности (например, круг), квадратные формы более предпочтительны из-за простоты их реализации.

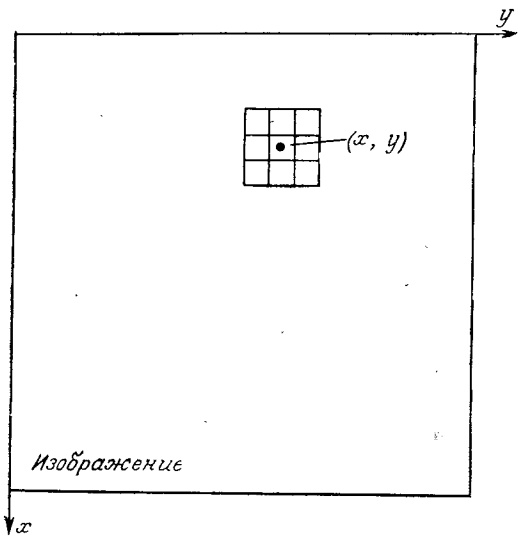


Рис. 7.20. Окрестность размерностью 3×3 точки (x, y) изображения.

Простейшая форма h получается, когда окрестность имеет размерность 1×1 и, следовательно, g зависит только от значения f в точке (x, y). В этом случае h становится картой интенсивности или преобразованием T вида

$$s = T(r), \qquad (7.6\text{-}2)$$

где для простоты введены переменные s и r, обозначающие соответственно интенсивность $f(x, y)$ и $g(x, y)$ в любой точке (x, y). Этот тип преобразования более подробно обсуждается в разд. 7.6.3.

−1	−1	−1
−1	8	−1
−1	−1	−1

Рис. 7.21. Маска для обнаружения отдельных точек, отличающихся от постоянного фона.

Один из наиболее часто встречающихся методов пространственной области основан на использовании так называемых масок свертки (или шаблонов, окон или фильтров). Обычно маска представляет собой небольшую (например, размерность 3×3) двумерную систему (рис. 7.20), коэффициенты которой выбираются таким образом, чтобы обнаружить заданное свойство изображения. Предположим для начала, что дано изображение с постоянной интенсивностью, которое содержит отдельные удаленные друг от друга пикселы с отличной от фона интенсивностью. Эти точки могут быть обнаружены маской, показанной на рис. 7.21. Процесс заключается в следующем. Центр маски (помеченный цифрой 8) перемещается по изображению определенным образом. При совпадении центра маски с положением каждого пиксела производится умножение значений всех пикселов, находящихся под маской, на соответствующий коэффициент на маске, т. е. значение пиксела под центром маски умножается на 8, а значения восьми соседних пикселов умножаются на −1. Затем результаты этих девяти умножений суммируются. Если все пикселы под маской имеют одинаковые значения (постоянный фон), то сумма будет равна нулю. Если же центр маски разместится над точкой с другой интенсивностью, сумма будет отлична от нуля. В случае размещения указанной точки вне центра сумма также будет отлична от нуля, но на меньшую величину. Это различие может быть устранено путем сравнения значения суммы с пороговым значением.

w_1 $(x-1, y-1)$	w_2 $(x-1, y)$	w_3 $(x-1, y+1)$
w_4 $(x, y-1)$	w_5 (x, y)	w_6 $(x, y+1)$
w_7 $(x+1, y-1)$	w_8 $(x+1, y)$	w_9 $(x+1, y+1)$

Рис. 7.22. Общая маска размерностью 3×3 с коэффициентами и соответствующими расположениями пикселов изображения.

Если величины $w_1, w_2, \ldots, w_9$ представляют собой коэффициенты маски пиксела (x, y) и его восьми соседей (рис. 7.22), предыдущее рассмотрение можно представить как выполнение следующей операции:

$$\begin{aligned} h[f(x, y)] = {} & w_1 f(x-1, y-1) + w_2 f(x-1, y) + \\ & + w_3 f(x-1, y+1) + w_4 f(x, y-1) + \\ & + w_5 f(x, y) + w_6 f(x, y+1) + \\ & + w_7 f(x+1, y-1) + w_8 f(x+1, y) + \\ & + w_9 f(x+1, y+1) \end{aligned} \qquad (7.6\text{-}3)$$

на окрестности размерностью 3×3 точки (x, y).

Отметим, что использование окрестности не ограничивается областями размерностью 3×3 и случаями, которые будут приведены в дальнейшем, например снижение шума, получение

переменных порогов изображения, подсчет измерений параметров изображения и формирование структуры объекта.

Методы частотной области. К частотной области относится совокупность комплексных пикселов в виде преобразования Фурье от изображения. Понятие «частота» используется при интерпретации преобразования Фурье и вытекает из того факта, что результат этого преобразования представляет собой сумму синусоид. Из-за повышения требований к обработке результатов методы частотной области не так широко используются в техническом зрении роботов, как методы пространственной области. Однако преобразование Фурье играет важную роль при анализе движения объекта и при описании объекта. Кроме того, многие пространственные методы для улучшения качества и восстановления изображения базируются на концепциях преобразования Фурье. Более подробно преобразования Фурье и его свойства изложены в работе [104].

Сначала рассмотрим дискретную функцию одной переменной $f(x)$, $x = 0, 1, 2, \ldots, N-1$. Прямое преобразование Фурье от $f(x)$ определяется следующим образом:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)\, e^{-j2\pi ux/N} \qquad (7.6\text{-}4)$$

для $u = 0, 1, 2, \ldots, N-1$. В этом уравнении $j = \sqrt{-1}$, а u — частотная переменная. Обратное преобразование Фурье от $F(u)$ восстанавливает функцию $f(x)$ и определяется в виде

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{j2\pi ux/N} \qquad (7.6\text{-}5)$$

для $x = 0, 1, 2, \ldots, N-1$. Справедливость этих уравнений, называемых парой преобразования Фурье, легко проверяется подстановкой уравнения (7.6-4) для $F(u)$ в уравнение (7.6-5), или наоборот. В обоих случаях получается тождество.

При прямом использовании уравнения (7.6-4) для $u = 0, 1, 2, \ldots, N-1$ требуется $\sim N^2$ операций сложения и умножения. При применении быстрого преобразования Фурье (БПФ) число операций сокращается до $N \log_2 N$, где предполагается, что N — целая степень числа 2. То же самое относится к уравнению (7.6-5) для $x = 0, 1, 2, \ldots, N-1$. Ряд алгоритмов БПФ реализуется на различных языках программирования.

Прямое и обратное двумерные преобразования Фурье для изображения размерностью $N \times N$ определяются формулами

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux+vy)/N} \qquad (7.6\text{-}6)$$

для $u, v = 0, 1, 2, \ldots, N-1$ и

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v)\, e^{j2\pi(ux+vy)/N} \qquad (7.6\text{-}7)$$

для $x, y = 0, 1, 2, \ldots, N-1$. Можно показать, что каждое из данных уравнений выражается в виде отдельных одноразмерных сумм в форме уравнения (7.6-4). Это позволяет проводить прямую процедуру вычисления двумерного преобразования Фурье, используя только одноразмерный БПФ-алгоритм. Сначала вычисляются и запоминаются преобразования каждой строки функции $f(x, y)$ (образуется таким образом двумерный массив промежуточных результатов). Полученные результаты умножаются на N, и вычисляются одномерные преобразования каждого столбца. Окончательным результатом является $F(u, v)$. Порядок вычисления строк и столбцов не влияет на конечный результат.

Преобразование Фурье можно использовать, как это показано в гл. 8, при решении многих задач систем технического зрения. Например, с помощью представления границы объекта в виде одномерного массива точек и вычисления их преобразования Фурье полученные значения $F(u)$ могут быть использованы в качестве описания формы границы. Одномерное преобразование Фурье является также эффективным при обнаружении движения объекта. Ипользование дискретного двумерного преобразования Фурье возможно при изменении, увеличении и восстановлении изображений, хотя, как уже отмечалось, применение этого метода в промышленных системах технического зрения до сих пор ограничено сложностью требуемых вычислений. В заключение отметим, что двумерное аналоговое преобразование Фурье может быть осуществлено (со скоростью света) оптическими средствами. Этот подход, требующий использования прецизионного оптического оборудования, применяется в промышленных условиях для решения таких задач, как проверка качества поверхности металла после финишной обработки.

7.6.2. Сглаживание

Операции сглаживания используются для снижения шума и других помех, которые могут появляться на изображении в результате дискретизации, квантования, передачи или возмущений внешней среды при получении изображения. Ниже рассматриваются некоторые методы быстрого сглаживания, которые могут быть использованы в системах технического зрения роботов.

Усреднение окрестности. Усреднение окрестности является прямым методом пространственной области для сглаживания

изображения. Для имеющегося изображения $f(x, y)$ процесс заключается в получении сглаженного изображения $g(x, y)$, интенсивность которого в каждой точке (x, y) равна усредненному значению интенсивности пикселов функции f, содержащихся в заданной окрестности точки (x, y). Другими словами, сглаженное изображение получается при использовании соотношения

$$g(x, y) = \frac{1}{P} \sum_{(n, m) \in S} f(n, m) \qquad (7.6\text{-}8)$$

для всех x и y функции $f(x, y)$. S — множество координат точек в окрестности точки (x, y), включая саму точку (x, y), а P — общее число точек в окрестности. Отметим путем сравнения уравнений (7.6-8) и (7.6-3), что для окрестности размерностью 3×3 первое уравнение является частным случаем второго при $w_i = 1/9$. Конечно, уравнение (7.6-8) не ограничивается применением квадратной формы окрестности, но, как указывалось в разд. 7.6-1, эта форма наиболее предпочтительна для систем технического зрения роботов.

Пример. На рис. 7.23 показан эффект сглаживания при усреднении окрестности. Отметим, что степень сглаживания строго пропорциональна размерности используемой окрестности. В большинстве процессоров масок сглаженное значение каждого пиксела определяется до того, как изменяются значения других пикселов.

Усредненная фильтрация. Одной из принципиальных трудностей усреднения окрестности является наличие расплывчатых изображений кромок и других характерных деталей. Эта расплывчатость часто может быть значительно уменьшена при использовании так называемых усредняющих фильтров, в которых вместо рассмотренного усреднения интенсивность каждого пиксела заменяется средним значением интенсивности в заданной окрестности пиксела.

Напомним, что среднее значение M ряда величин предполагает, что часть этих величин имеет значение, меньшее M, а часть — превышающее M. Для осуществления усредненной фильтрации в окрестности пиксела сначала выделяют значения пиксела и его соседей, определяют среднее значение и присваивают это значение пиксела. Например, в окрестности размерностью 3×3 средним значением является пятая наибольшая величина, в окрестности размерностью 5×5 — тринадцатая наибольшая величина и т. д. Когда несколько значений в окрестности равны, они группируются следующим образом. Предположим, что окрестность размерностью 3×3 содержит величины (10, 20, 20, 20, 15, 20, 20, 25, 100). Эти величины группируются в виде (10, 15, 20, 20, 20, 20, 20, 25, 100), что приводит к среднему значению 20. Таким образом, основным назначением

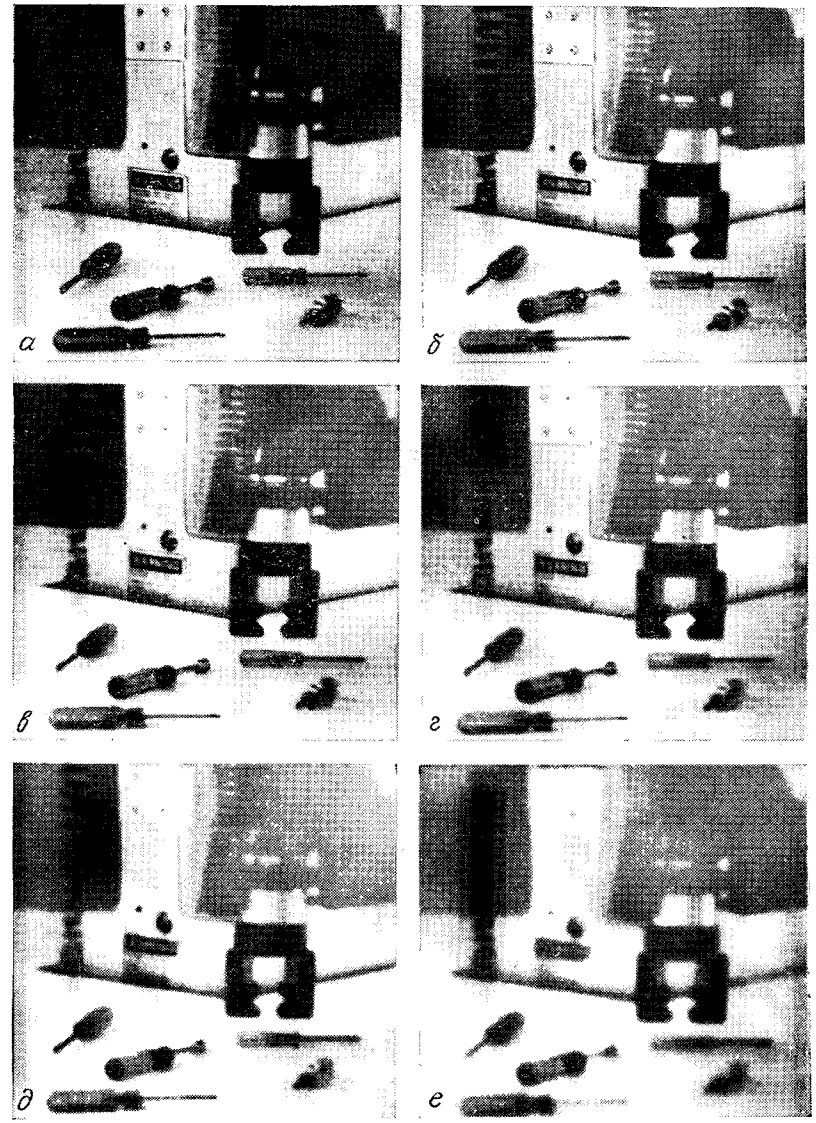


Рис. 7.23. Изображение с помехами (*а*). Результат усреднения каждого пиксела по его четырем соседям (*б*) и результаты использования окрестностей соответственно с размерностями 3×3, 5×5, 7×7 и 11×11 (*в* — *е*).

усредненной фильтрации является выравнивание интенсивности выделяющихся точек с интенсивностью соседних точек. Это позволяет устранить появляющиеся в поле маски фильтра изолированные пики интенсивности.

Рис. 7.24. Исходное изображение (*а*). Изображение с импульсным шумом (*б*), результат использования окрестности размерностью 5×5 (*в*) и результат использования усредняющего фильтра размерностью 5×5 (*г*).

Пример. На рис. 7.24, *а* показано исходное изображение, а на рис. 7.24, *б* — то же самое изображение, но приблизительно с 20 %-ным искажением пикселов «импульсным шумом». На рис. 7.24, *в* приведен результат усреднения размерностью 5×5, а на рис. 7.24, *г* — результат использования усредненного фильтра размерностью 5×5. Преимущество усредненной фильтрации перед усреднением окрестности очевидно. Три светлые точки, оставшиеся на рис. 7.24, *г*, вызваны большой концентрацией

шума в этих точках, повлиявшей на вычисление среднего значения. Повторное или многократное использование усредненного фильтра позволяет устранить и эти точки.

Усреднение изображения. Рассмотрим изображение с помехами $g(x, y)$, которое состоит из суммы шумового сигнала $n(x, y)$ и неискаженного изображения $f(x, y)$, т. е.

$$g(x, y) = f(x, y) + n(x, y), \tag{7.6-9}$$

где предполагается, что шум не коррелирован и имеет нулевое среднее значение. Целью следующей процедуры является получение сглаженного изображения путем задания ряда изображений с помехами $g_i(x, y)$, $i = 1, 2, \ldots, K$.

Если характеристики шума удовлетворяют приведенным ограничениям, несложно показать [224], что при формировании изображения $\bar{g}(x, y)$ усреднением K различными изображениями с помехами

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y) \tag{7.6-10}$$

получим

$$E\{\bar{g}(x, y)\} = f(x, y) \tag{7.6-11}$$

и

$$\sigma^2_{\bar{g}}(x, y) = \frac{1}{K} \sigma^2_n(x, y), \tag{7.6-12}$$

где $E\{\bar{g}(x, y)\}$ — ожидаемое значение $\bar{g}$, а $\sigma^2_{\bar{g}}(x, y)$ и $\sigma^2_n(x, y)$ — отклонения $\bar{g}$ и n. Стандартное отклонение любой точки в усредненном изображении равно

$$\sigma_{\bar{g}}(x, y) = \frac{1}{\sqrt{K}} \sigma_n(x, y). \tag{7.6-13}$$

Из уравнений (7.6-12) и (7.6-13) видно, что при возрастании K отклонения значений пикселов уменьшаются. Поскольку $E\{\bar{g}(x, y)\} = f(x, y)$, то $\bar{g}(x, y)$ приближается к неискаженному изображению $f(x, y)$ при увеличении числа изображений с помехами, используемых в процессе усреднения.

Необходимо отметить, что в рассмотренном методе предполагалось фиксированное пространственное расположение изображений с помехами при изменении только интенсивностей пикселов. Для систем технического зрения роботов это означает, что все объекты в рабочем пространстве должны быть неподвижны относительно камеры в процессе усреднения. Многие системы технического зрения имеют возможность выполнения полного цикла сложения изображений за один дискретный интервал времени (т. е. за одну тридцатую секунды). Таким образом, сложение, например, 16 изображений займет время

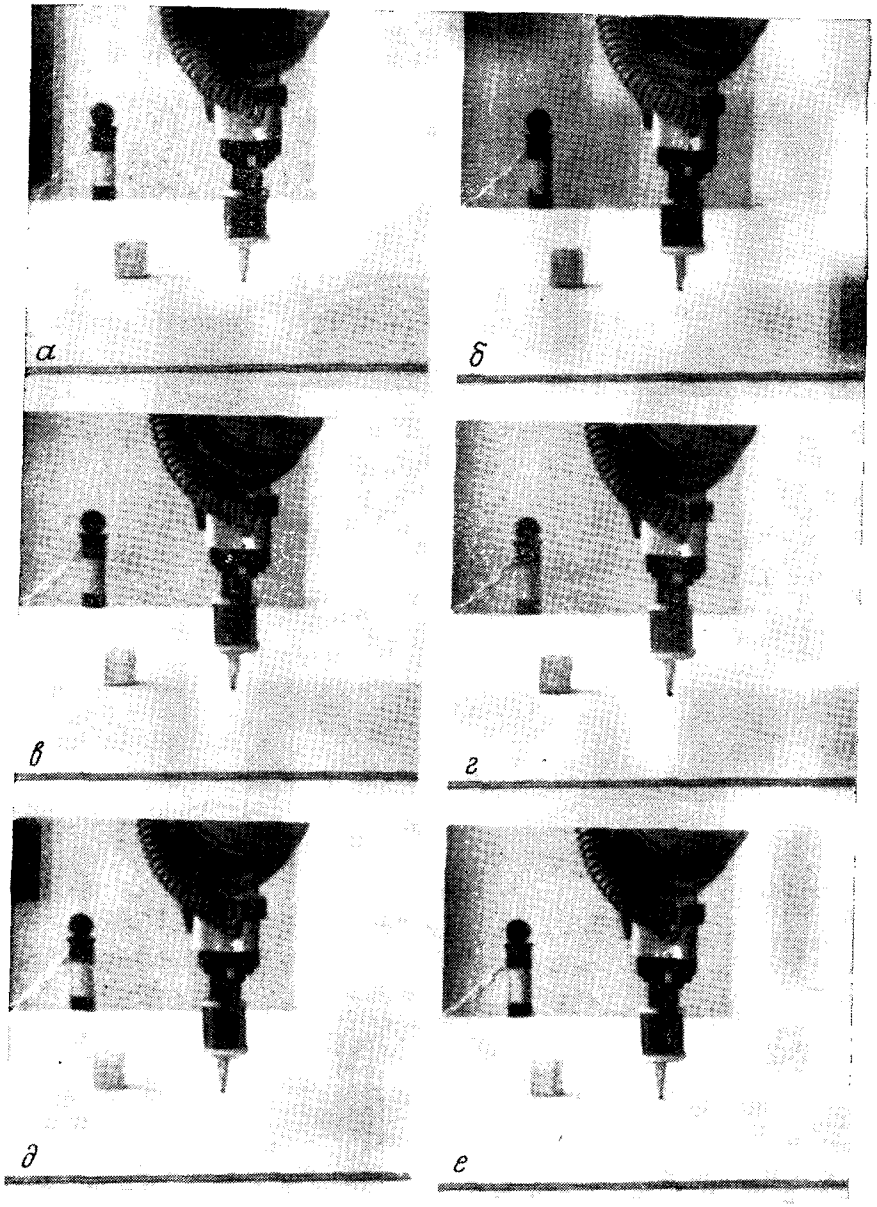


Рис. 7.25. Исходное изображение с помехами (*а*) и результаты усреднения с помощью 4, 3, 16, 32 и 64 подобных изображений (*б* — *е*).

порядка 0,5 с, в течение которого не должно быть движения объектов.

Пример. Метод усреднения проиллюстрирован на рис. 7.25. На рис. 7.25, *а* показано исходное изображение с помехами, а на рис. 7.25, *б*—*е* приведены соответственно результаты усреднения этого изображения с помощью 4, 8, 16, 32 и 64 подобных изображений. Отметим, что качество достаточно приемлемо при $K = 32$.

Сглаживающие дискретные изображения. Дискретные изображения получаются при использовании контурного или структурированного освещения (разд. 7.3) или при определении кромок или пороговых значений (разд. 7.6.4 и 7.6.5). Условимся обозначать темные точки через 1, а светлые точки через 0. Таким образом, поскольку изображения состоят из двух значений интенсивности, помехи в этом случае проявляются в виде таких эффектов, как наличие размытых границ, небольших окружностей, стертых углов и отдельных точек.

a	*b*	*c*
d	*p*	*e*
f	*g*	*h*

Рис. 7.26. Соседи пиксела *p*, используемые при сглаживании дискретных изображений. Темные пикселы обозначаются через 1, а светлые — через 0.

Основная идея, лежащая в основе рассматриваемых в данном разделе методов, состоит в определении булевой функции, вычисляемой в окрестности с центром в пикселе *p*, и в присвоении пикселу *p* значения 1 или 0 в зависимости от пространственного расположения и дискретных значений соседних пикселов. Из-за ограниченности допустимого времени вычисления в производственных задачах, для решения которых применяется техническое зрение, анализ обычно производится в диапазоне восьми соседних с центральным пикселов. Это определяет использование маски размерностью 3×3 (рис. 7.26). В процессе сглаживания, во-первых, заполняются небольшие (размером в один пиксел) пробелы на темных местах изображения, во-вторых, ликвидируются небольшие дефекты в виде трещин на прямоугольных сегментах, в-третьих, удаляются изолированные единичные значения, в-четвертых, спрямляются небольшие выпуклости вдоль прямоугольных сегментов и, в-пятых, восстанавливаются утраченные угловые точки.

На рис. 7.26 два первых из указанных результатов процесса сглаживания осуществляются с помощью булевого выражения

$$B_1 = p + b \cdot g \cdot (d + e) + d \cdot e \cdot (b + g), \qquad (7.6\text{-}14)$$

где точка и плюс обозначают соответственно логические операции И или ИЛИ. По условию темные пикселы, содержащиеся в поле маски, обозначаются логической единицей, а светлые пикселы — логическим нулем. Тогда, если $B_1 = 1$, присваиваем пикселу p значение 1, в противном случае — значение 0. Уравнение (7.6-14) применяется одновременно ко всем пикселам, так как следующее значение положения каждого пиксела определяется до того, как изменяются значения других пикселов.

Третий и четвертый результаты процесса сглаживания реализуются с помощью определения булевого выражения

$$B_2 = p \cdot [(a + b + d) \cdot (e + g + h) + (b + c + e) \times \times (d + f + g)] \qquad (7.6\text{-}15)$$

одновременно для всех пикселов. Как и прежде, предполагается, что $p = 1$ при $B_2 = 1$ и $p = 0$ при $B_2 = 0$.

Восстановление точек верхних правых углов производится с помощью уравнения

$$B_3 = \bar{p} \cdot (d \cdot f \cdot g) \cdot (\overline{a + b + c + e + h}) + p, \qquad (7.6\text{-}16)$$

где чертой обозначено логическое дополнение. Аналогично восстанавливаются точки правых нижних, левых верхних и левых нижних углов соответственно по уравнениям

$$B_4 = \bar{p} \cdot (a \cdot b \cdot d) \cdot (\overline{c + e + f + g + h}) + p, \qquad (7.6\text{-}17)$$

$$B_5 = \bar{p} \cdot (e \cdot g \cdot h) \cdot (\overline{a + b + c + d + f}) + p \qquad (7.6\text{-}18)$$

и

$$B_6 = \bar{p} \cdot (b \cdot c \cdot e) \cdot (\overline{a + d + f + g + h}) + p. \qquad (7.6\text{-}19)$$

Четыре последних уравнения дают пятый результат процесса сглаживания.

Пример. Рассмотренный метод проиллюстрирован на рис. 7.27. На рис. 7.27, *а* приведено дискретное изображение с помехами, а на рис. 7.27, *б* показан результат использования выражения B_1. Отметим, что при этом заполняются пробелы вдоль границы и внутри темной области изображения. На рис. 7.27, *в* дана картина применения выражения B_2 к изображению на рис. 7.27, *б*. Как и ожидалось, устранились выпуклости на границе темной области, а также отдельные точки изображения (изображение полностью заполнено нулями от своей границы до границы темной области). Наконец, на рис. 7.27, *г* приведен результат использования выражений $B_3 - B_6$ к изображению на рис. 7.27, *в*. В данном случае работает только выражение B_4.

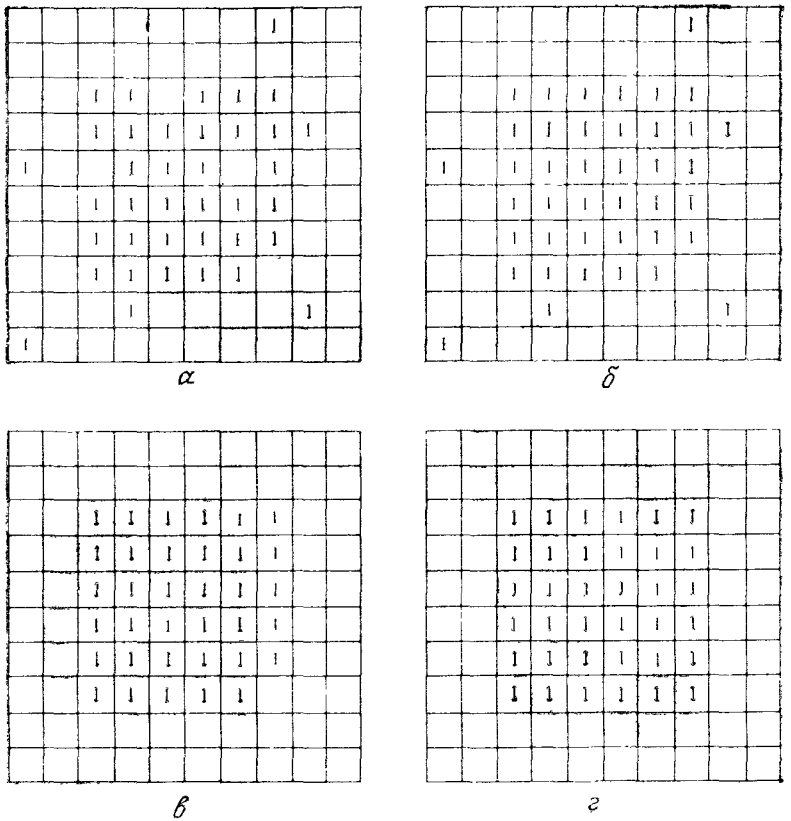


Рис. 7.27. Исходное изображение (*а*). Результаты использования выражений B_1 и B_2 (*б* и *в*) и окончательный результат после использования выражений $B_3 - B_6$ (*г*).

7.6.3. Улучшение качества изображения

Одной из принципиальных трудностей во многих задачах систем технического зрения нижнего уровня является необходимость автоматической адаптации при изменении освещения. Возможность компенсации таких эффектов, как тени и блики на изображении, часто является решающей для успешного выполнения дальнейшего алгоритма обработки информации. Ниже рассматривается несколько методов решения этих и подобных задач. Напомним, что улучшение качества является

основной целью при обработке цифрового изображения и анализе объектов. Мы ограничимся дискретными методами, пригодными для систем технического зрения роботов. Под пригодностью здесь понимается наличие высокого быстродействия и средней сложности используемого оборудования.

Гистограммное выравнивание. Пусть переменная r обозначает интенсивность пикселов корректируемого изображения. Сначала предположим, что r — нормализованная непрерывная переменная, находящаяся в диапазоне $0 \leqslant r \leqslant 1$. Дискретный случай мы рассмотрим немного ниже.

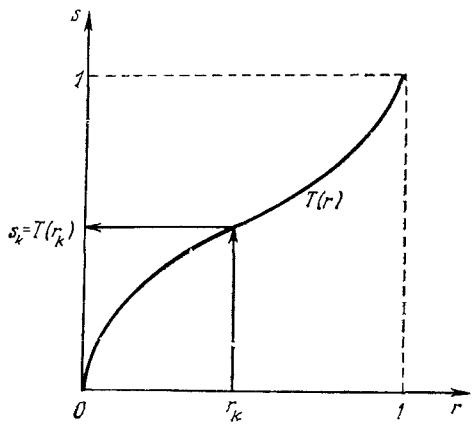


Рис. 7.28. Функция преобразования интенсивности.

Для любого r на интервале [0, 1] выполняется преобразование вида

$$s = T(r),$$

которое определяет значение интенсивности s для значения каждого пиксела r в исходном изображении. Это предполагает, что функция преобразования T удовлетворяет условиям:

1. $T(r)$ — однозначная и монотонно возрастающая на интервале $0 \leqslant T(r) \leqslant 1$.

2. $0 \leqslant T(r) \leqslant 1$ для $0 \leqslant r \leqslant 1$.

Первое условие исключает переход с черной шкалы интенсивности на белую, а второе условие гарантирует нахождение интенсивности изображения в диапазоне от 0 до 1 для всех значений пикселов. Функция преобразования, удовлетворяющая этим условиям, показана на рис. 7.28.

Функция обратного преобразования от s к r имеет вид

$$r = T^{-1}(s), \tag{7.6-21}$$

где предполагается, что функция $T^{-1}(s)$ удовлетворяет двум указанным выше условиям.

Переменные интенсивности r и s квантуются на интервале [0, 1] случайным образом и, следовательно, могут характеризоваться с помощью соответствующих функций плотности вероятности (ФПВ) $p_r(r)$ и $p_s(s)$. Основная тональность изображения зависит от ФПВ интенсивности. Например, изображение с пикселами, имеющими ФПВ, которая приведена на рис. 7.29, *а*, будет обладать достаточно темной тональностью, поскольку большинство значений пикселов сконцентрировано на темном

участке шкалы интенсивности. С другой стороны, изображение, в котором пикселы имеют распределение интенсивности, соответствующее рис. 7.29, *б*, будет состоять в основном из светлых тонов.

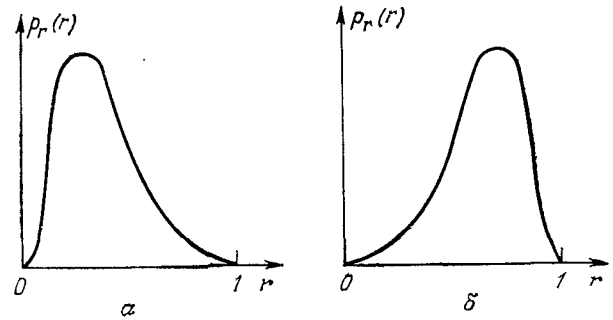


Рис. 7.29. ФПВ интенсивности темного изображения (*а*) и светлого изображения (*б*).

Из теории вероятности следует, что, если $p_r(r)$ и $T(r)$ известны, а $T^{-1}(s)$ — удовлетворяет первому условию, то ФПВ преобразованной интенсивности имеет вид

$$p_s(s) = \left[p_r(r)\frac{dr}{ds}\right]_{r=T^{-1}(s)}. \tag{7.6-22}$$

Предположим, что задана функция преобразования вида

$$s = T(r) = \int_0^r p_r(w)\,dw, \quad 0 \leqslant r \leqslant 1, \tag{7.6-23}$$

где w — вспомогательная переменная интегрирования. Правая часть этого уравнения представляет собой интегральную функцию распределения $p_r(r)$, которая удовлетворяет двум приведенным выше условиям. Производная по s от r для данной функции преобразования легко определяется выражением

$$\frac{ds}{dr} = p_r(r). \tag{7.6-24}$$

Подставляя dr/ds в уравнение (7.6-22), получим

$$p_s(s) = \left[p_r(r)\frac{1}{p_r(r)}\right]_{r=T^{-1}(s)} = [1]_{r=T^{-1}(s)} = 1, \quad 0 \leqslant s \leqslant 1. \tag{7.6-25}$$

Данное выражение представляет собой равномерную плотность на интервале существования преобразуемой переменной s. Отметим, что полученный результат не зависит от обратной функции преобразования. Это важно, так как часто аналитиче-

ское определение $T^{-1}(s)$ представляет большие трудности. Необходимо также отметить, что использование функции преобразования, заданной уравнением (7.6-23), дает новую интенсивность, которая всегда имеет новую ФПВ, не зависящую от формы $p_r(r)$. Данное свойство идеально соответствует автоматической коррекции. Основной эффект рассмотренного преобразования заключается в выравнивании распределения интенсивностей. Как показано ниже, это существенно влияет на качество изображения.

Для использования в цифровой обработке изложенный способ должен быть сформулирован в дискретной форме. Для интенсивностей рассматриваемых дискретных величин вероятности задаются соотношением

$$p_r(r_k) = \frac{n_k}{n}, \quad 0 \leqslant r_k \leqslant 1, \quad k = 0, 1, 2, \ldots, L-1, \quad (7.6\text{-}26)$$

где L — число дискретных уровней интенсивности, $p_r(r_k)$ — оценка вероятности интенсивности r_k, n_k — число появлений данной интенсивности на изображении, а n — общее число пикселов на изображении. График зависимости $p_r(r_k)$ от r_k обычно называют гистограммой, а метод, используемый для получения равномерной гистограммы, известен и как *гистограммная коррекция* или *гистограммная линеаризация.*

Дискретная форма уравнения (7.6-23) имеет вид

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j) \quad (7.6\text{-}27)$$

для $0 \leqslant r_k \leqslant 1$ и $k = 0, 1, 2, \ldots, L-1$. Из этого уравнения следует, что для получения значения s_k, соответствующего r_k, производится простое суммирование элементов гистограммы от 0 до r_k.

Обратное дискретное преобразование имеет вид

$$r_k = T^{-1}(s_k), \quad 0 \leqslant s_k \leqslant 1, \quad (7.6\text{-}28)$$

где предполагается, что как $T(r_k)$, так и $T^{-1}(s_k)$ удовлетворяют двум указанным выше условиям. Хотя функция $T^{-1}(s_k)$ не используется в гистограммной коррекции, она играет основную роль в гистограммной детализации, рассматриваемой ниже.

Пример. Для иллюстрации гистограммного выравнивания на рис. 7.30, *а* показано исходное изображение, а на рис. 7.30, *б* — его гистограмма. Результат использования уравнения (7.6-27) для этого изображения приведен на рис. 7.30, *в*, а соответствующая выровненная гистограмма — на рис. 7.30, *г*. Лучшее изображение деталей на скорректированной картине очевидно. Отметим, что при этом гистограмма не является совершенно ровной, что обычно происходит, когда для дискретных величин используется метод, полученный для непрерывных величин.

Метод задания гистограммы. Гистограммное выравнивание удобно для автоматической коррекции, поскольку оно основано на функции преобразования, т. е. однозначно определяется с помощью гистограммы исходного изображения. Однако этот метод ограничен в том смысле, что в нем используется только функция гистограммной линеаризации. Этого недостаточно,

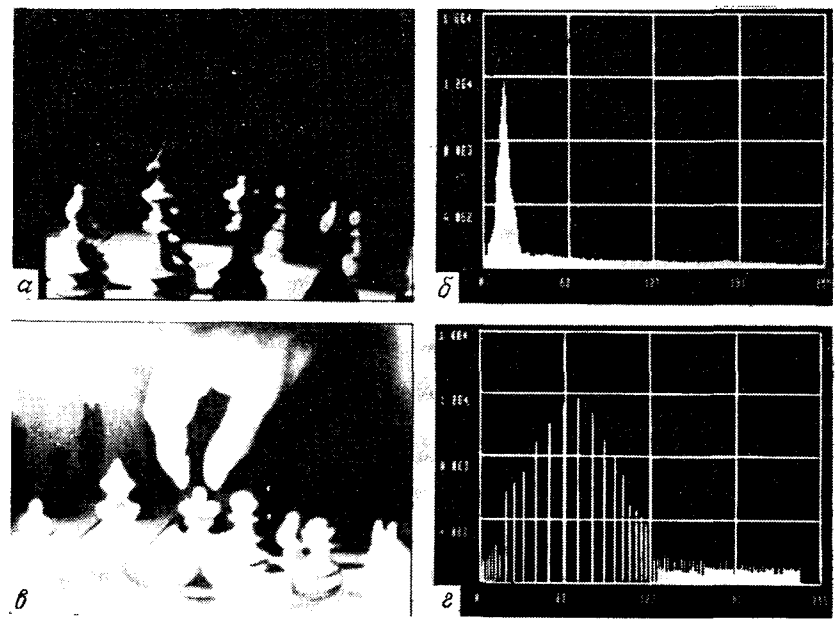


Рис. 7.30. Исходное изображение (*а*) и его гистограмма (*б*). Изображение с выровненной гистограммой (*в*) и его гистограмма (*г*) [315].

когда исходной информацией является желаемая форма выходной гистограммы. Обобщим теперь гистограммный способ на случай получения изображения с гистограммой определенной интенсивности. Как показано ниже, гистограммное выравнивание является частным случаем этого метода.

Сначала оперируем с непрерывными величинами. Пусть $p_r(r)$ и $p_z(z)$ — соответственно исходная и желаемая ФПВ интенсивности. Предположим, что заданное изображение с первой гистограммой выровнено с помощью уравнения (7.6-23), т. е.

$$s = T(r) = \int_0^r p_r(w)\, dw. \quad (7.6\text{-}29)$$

Если желаемое изображение было бы доступно, его уровни интенсивности также могли бы быть выровнены с использованием функции преобразования

$$v = G(z) = \int_0^z p_z(w)\,dw. \qquad (7.6\text{-}30)$$

Тогда бы обратное преобразование $z = G^{-1}(v)$ восстанавливало желаемые первоначальные уровни. Это, конечно, гипотетическое утверждение, поскольку уровни z — именно то, что необходимо получить. Отметим, однако, что $p_s(s)$ и $p_v(v)$ будут идентичными равномерными плотностями, так как использование уравнений (7.6-29) и (7.6-30) обеспечивает равномерную плотность независимо от формы ФПВ под интегралом. Таким образом, если вместо применения v в обратном преобразовании использовать обратные уровни s, полученные из исходного изображения, то результирующие уровни $z = G^{-1}(s)$ будут иметь желаемую ФПВ $p_z(z)$. В предположении, что $G^{-1}(s)$ является однозначной функцией, алгоритм можно обобщить следующим образом:

1. Выровнять уровни интенсивности исходного изображения с помощью уравнения (7.6-29).
2. Определить желаемую ФПВ интенсивности и получить функцию преобразования $G(z)$, используя уравнение (7.6-30).
3. Осуществить обратное преобразование $z = G^{-1}(s)$ к уровням интенсивности изображения с выровненной по п. 1 гистограммой.

Этот алгоритм дает выходное изображение с заданной ФПВ интенсивности.

Два требуемых для определения гистограммы преобразования $T(r)$ и $G^{-1}(s)$ можно объединить в одно преобразование

$$z = G^{-1}(s) = G^{-1}[T(r)], \qquad (7.6\text{-}31)$$

которое связывает r с z. Отметим, что при $G^{-1}[T(r)] = T(r)$ этот метод сводится к гистограммному выравниванию.

Из уравнения (7.6-31) видно, что исходное изображение не нуждается в точном гистограммном выравнивании для определения гистограммы. При этом требуется только, чтобы функция $T(r)$ зависела и связывалась с $G^{-1}(s)$ одним преобразованием, т. е. могла быть применена непосредственно к исходному изображению. Основная трудность использования двух преобразований или их совместного выражения (7.6-31) для непрерывных переменных заключается в аналитическом определении обратной функции. В дискретном случае эта проблема упрощается из-за обычно сравнительно небольшого (например, 256) числа отдельных уровней, что делает осуществимым подсчет и хранение информации о каждом возможном целом значении пиксела.

Формулировка данного алгоритма для дискретного случая сходна с дискретным выражением, рассмотренным в предыдущем разделе:

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j), \qquad (7.6\text{-}32)$$

$$G(z_i) = \sum_{j=0}^{i} d_z(z_j) \qquad (7.6\text{-}33)$$

и

$$z_i = G^{-1}(s_i), \qquad (7.6\text{-}34)$$

где $p_r(r_j)$ вычисляется из исходного изображения, а $p_z(z_j)$ задается.

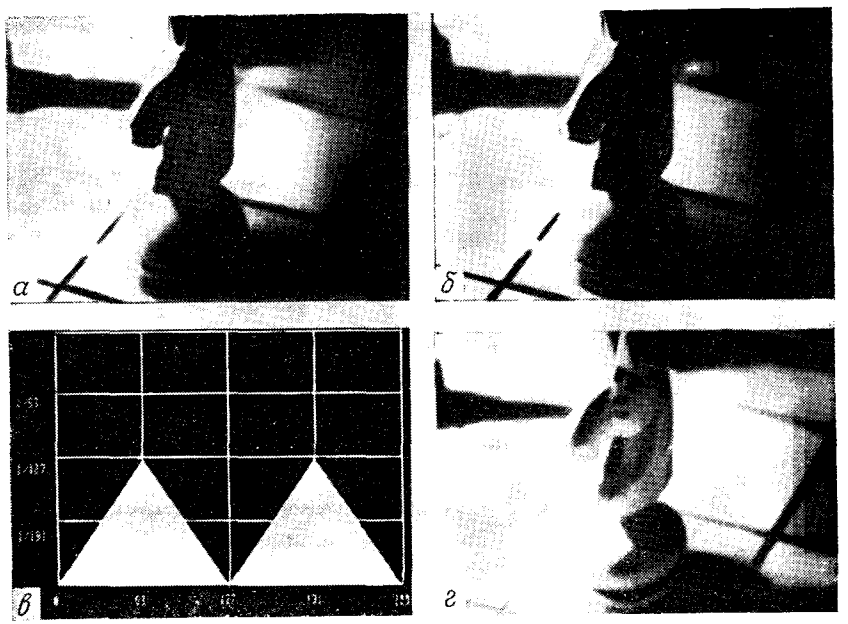


Рис. 7.31. Исходное изображение (*а*) и результат гистограммного выравнивания (*б*). Заданная гистограмма (*в*) и результат использования метода задания гистограммы (*г*) [315].

Пример. Иллюстрация метода задания гистограммы приведена на рис. 7.31. На рис. 7.31, *а* показано исходное изображение, а на рис. 7.31, *б* — результат выравнивания. На рис. 7.31, *в* изображена заданная гистограмма, а на рис. 7.31, *г* — результат использования этой гистограммы с помощью рассмотренного алгоритма. Отметим, что в данном случае применение

гистограммного выравнивания мало влияет на качество изображения.

Локальное улучшение качества. Приведенные методы гистограммного выравнивания и задания гистограммы являются общими в смысле изменения интенсивности пикселов с помощью функции преобразования на всем поле изображения. Хотя эти методы дают общее улучшение качества, часто необходимо выделить детали на достаточно малых участках изображения. Поскольку качество пикселов на этих участках может мало влиять на вычисления при преобразовании всего изображения, использование общих методов редко приводит к улучшению качества на малых участках. Решением данной проблемы является усовершенствование функций преобразования, основанных на распределении интенсивности или на других свойствах, в окрестности каждого пиксела заданного изображения.

Рассмотренные выше гистограммные методы легко трансформируются для локального улучшения качества. Трансформация заключается в определении окрестности размерностью $n \times m$ и в перемещении ее центра от пиксела к пикселу. В каждом положении вычисляется гистограмма точек в данной окрестности и определяется функция преобразования гистограммного выравнивания или функция преобразования метода задания гистограммы. Эта функция служит для отображения интенсивности пиксела, расположенного в центре окрестности. Затем центр участка размерностью $n \times m$ перемещается на соседний пиксел и процесс повторяется. Поскольку при перемещении участка от пиксела к пикселу меняется только один новый ряд или столбец окрестности, то можно осуществить пересчет гистограммы, полученной для предыдущего положения с новыми параметрами, появляющимися на каждом шаге движения. Этот метод имеет очевидные преимущества перед методом с повторяющимися вычислениями гистограммы всех точек области $n \times m$ при ее перемещении каждый раз на один пиксел. Другим способом, часто используемым для упрощения вычислений, является применение непересекающихся областей. Это, однако, приводит к появлению нежелательного шахматного распределения интенсивности на изображении.

Пример. Локальное гистограммное выравнивание с перемещением окрестности от пиксела к пикселу проиллюстрировано на рис. 7.32. На рис. 7.32, *а* показано изображение с пятью темными квадратными участками на фоне с постоянной интенсивностью. Изображение слегка расплывчато в результате использования маски размерностью 7×7 для уменьшения шума (разд. 7.6.2). На рис. 7.32, *б* приведен результат гистограммного выравнивания. Наиболее характерной особенностью этого изображения является увеличение помех, что обычно происходит при использовании в этом методе изображений с шумом,

даже если они были предварительно сглажены перед выравниванием. На рис. 7.32, *в* показан результат локального гистограммного выравнивания с применением окрестности размерностью 7×7. Отметим, что на темных участках изображения обнаружилась внутренняя структура, которая не была видна ни на одном из двух предыдущих изображений. Помехи по

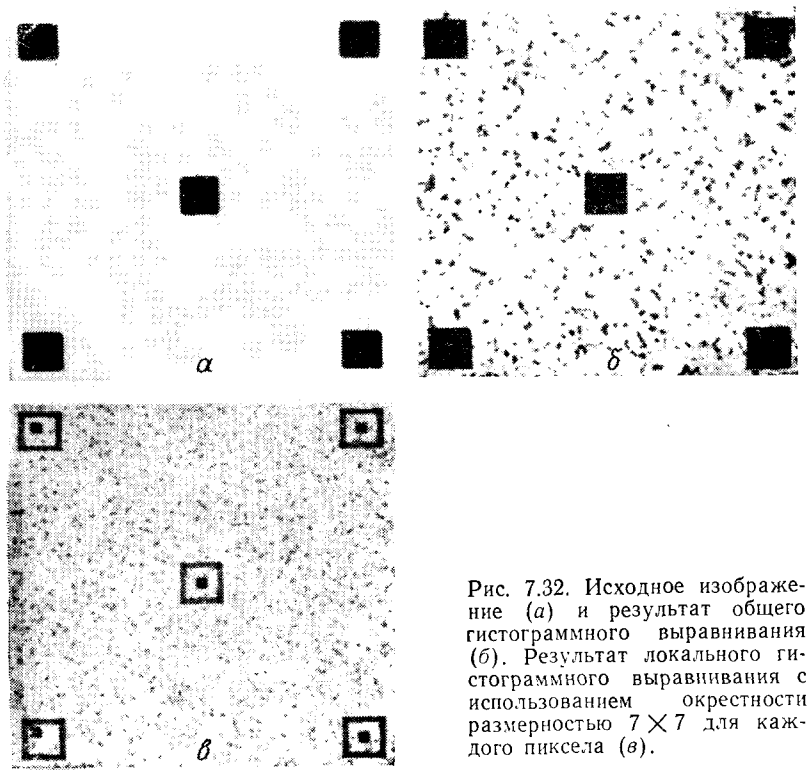


Рис. 7.32. Исходное изображение (*а*) и результат общего гистограммного выравнивания (*б*). Результат локального гистограммного выравнивания с использованием окрестности размерностью 7×7 для каждого пиксела (*в*).

сравнению с исходным изображением выросли, но их величина намного меньше, чем на втором изображении, благодаря локальности метода. Этот пример наглядно показывает необходимость локального улучшения качества, когда интересующие детали слишком малы, чтобы существенно влиять на общие характеристики методов обработки целого изображения.

Качество изображения на локальных участках можно улучшать, используя другие параметры интенсивностей пикселов в окружности. Двумя такими часто используемыми параметрами являются среднее значение интенсивности и изменение интенсивности (или стандартное отклонение). Среднее значение — это мера средней яркости, а изменение интенсивности —

мера контрастности изображения. Типичное локальное преобразование, основанное на этих параметрах, переводит интенсивность исходного изображения $f(x, y)$ в интенсивность нового изображения $g(x, y)$ путем осуществления следующей операции над расположением (x, y) каждого пиксела:

$$g(x, y) = A(x, y)[f(x, y) - m(x, y)] + m(x, y), \quad (7.6\text{-}35)$$

где

$$A(x, y) = k\frac{M}{\sigma(x, y)}, \quad 0 < k < 1. \quad (7.6\text{-}36)$$

В этой записи $m(x, y)$ и $\sigma(x, y)$ — соответственно среднее значение интенсивности и стандартное отклонение, подсчитан-

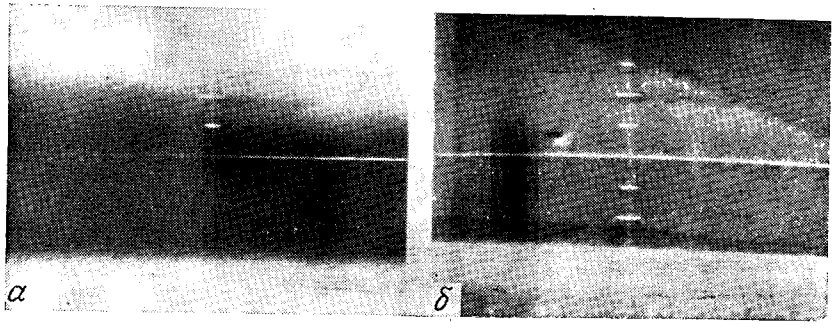


Рис. 7.33. Изображения до (*а*) и после (*б*) локального улучшения качества [207].

ные для окрестности с центром в точке (x, y), M — среднее значение всей функции $f(x, y)$, а k — постоянная из указанного диапазона.

Важно отметить, что A, m и σ — переменные величины, которые зависят от заданной окрестности точки (x, y). Локальные изменения увеличиваются за счет умножения разности между $f(x, y)$ и локальным средним значением на локальный коэффициент $A(x, y)$. Поскольку величина $A(x, y)$ обратно пропорциональна стандартному отклонению интенсивности, участки с низкой контрастностью имеют большее усиление. Среднее значение подставляется снова в уравнение (7.6-35) для восстановления среднего уровня интенсивности изображения на локальном участке. На практике желательно подставлять часть среднего значения и ограничивать изменение функции $A(x, y)$ в пределах $[A_{\text{мин}}, A_{\text{макс}}]$ для уравновешивания больших отклонений интенсивности на отдельных участках.

Пример. Оборудование для реализации метода улучшения качества изображения и способ обработки изображения в реальном масштабе времени (т. е. со скоростью 30 кадров в 1 с) описаны в работе [207]. Пример возможностей метода, использующего локальный участок размером 15×15 пикселов, приведен на рис. 7.33. Заметно улучшение качества изображения деталей на границе между двумя областями различных интенсивностей, а также улучшение воспроизведения интенсивности внутри каждой из этих областей.

7.6.4. Определение кромок

Определение кромок занимает центральное место при обработке информации в системах технического зрения, представляя собой начальный этап реализации многочисленных алгоритмов идентификации объектов. В этой главе рассматриваются основные методы определения точек кромок. Способы последующей обработки этих точек излагаются в гл. 8.

Основные положения. Основным принципом большинства методов определения кромок является вычисление частных производных. Поясним это положение с помощью рис. 7.34. На рис. 7.34, *а* приведены изображение светлого объекта простой формы на темном фоне, кривая интенсивности вдоль горизонтальной линии сканирования изображения, а также первая и вторая производные этой кривой. Отметим, что участки кривой, соответствующие кромке (переход от темной области к светлой), представляют собой наклонные, а не вертикальные линии, какие должны быть в случае резкого изменения интенсивности. Это отражает тот факт, что кромки на цифровом изображении в результате дискретизации обычно слегка размыты.

В указанной модели кромки первая производная всех участков кривой с постоянной интенсивностью равна нулю и является постоянной величиной на участках изменения интенсивности. С другой стороны, вторая производная равна нулю на всех участках, кроме начальных и конечных точек изменения интенсивности. С учетом этого очевидно, что величина первой производной может быть использована для обнаружения наличия кромки, а знак второй производной — для определения того, на темной (фон) или на светлой (объект) стороне кромки располагается пиксел кромки. Например, знак второй производной (рис. 7.34, *а*) положителен для пикселов, расположенных на внешних сторонах кромок объекта, и отрицателен для пикселов внутренних светлых сторон этих кромок. Те же рассуждения справедливы для случая нахождения темного объекта на светлом фоне (рис. 7.34, *б*). Отметим, что знак второй

производной при этом также позволяет определить переход интенсивности на кромке.

Хотя данное рассмотрение ограничивается одномерным горизонтальным профилем, аналогично можно определять кромки при любой их ориентации на изображении. Для этого изучается

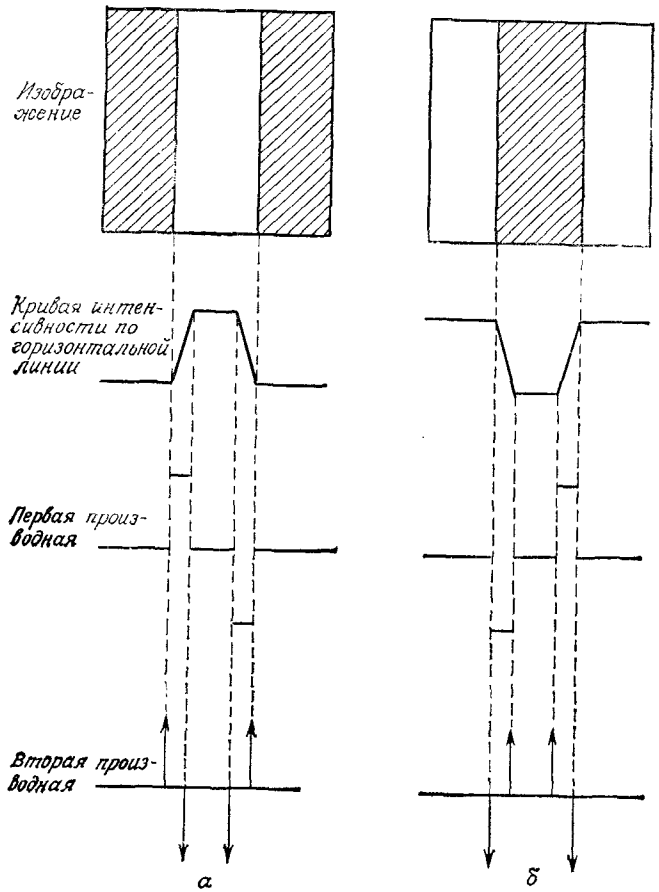


Рис. 7.34. Определение кромок с помощью производных. *а*) — светлый объект на темном фоне, *б*) — темный объект на светлом фоне.

профиль, перпендикулярный к направлению кромки в любой заданной точке, а результаты находятся указанным выше образом. Как показано ниже, первая производная в любой точке изображения получается из величины градиента в этой точке, а вторая производная определяется с помощью преобразования Лапласа.

Вычисление градиента. Градиент изображения $f(x, y)$ в точке (x, y) определяется как двумерный вектор

$$\mathbf{G}[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \qquad (7.6\text{-}37)$$

Из векторного анализа известно, что вектор $\mathbf{G}$ указывает направление максимального изменения функции f в точке (x, y). Однако при определении кромок представляет интерес величина этого вектора, называемого обычно *градиентом* и обозначаемого как $G[f(x, y)]$, где

$$G[f(x, y)] = [G_x^2 + G_y^2]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}. \qquad (7.6\text{-}38)$$

На практике, как правило, градиент аппроксимируется абсолютными значениями

$$G[f(x, y)] \cong |G_x| + |G_y|. \qquad (7.6\text{-}39)$$

Эта аппроксимация значительно упрощает реализацию метода, в особенности при использовании распространенного оборудования.

Из уравнения (7.6-38) следует, что вычисление градиента основано на нахождении первых производных $\partial f/\partial x$ и $\partial f/\partial y$. Для цифрового изображения это можно сделать несколькими путями. Один из подходов состоит в использовании разности между соседними пикселами

$$G_x = \frac{\partial f}{\partial x} = f(x, y) - f(x - 1, y), \qquad (7.6\text{-}40)$$

$$G_y = \frac{\partial f}{\partial y} = f(x, y) - f(x, y - 1). \qquad (7.6\text{-}41)$$

Несколько более сложный способ включает пикселы в окрестности размерностью 3×3 с центром в точке (x, y):

$$G_x = \frac{\partial f}{\partial x} = [f(x + 1, y - 1) + 2f(x + 1, y) + f(x + 1, y + 1)] - [f(x - 1, y - 1) + 2f(x - 1, y) + f(x - 1, y + 1)] = (g + 2h + i) - (a + 2b + c), \qquad (7.6\text{-}42)$$

$$G_y = \frac{\partial f}{\partial y} = [(x - 1, y + 1) + 2f(x, y + 1) + f(x + 1, y + 1)] - [f(x - 1, y - 1) + 2f(x, y - 1) + f(x + 1, y - 1)] = (c + 2e + i) - (a + 2d + g), \qquad (7.6\text{-}43)$$

где буквы от a до i обозначают соседние точки центра (x, y). Окрестность размерностью 3×3 точки (x, y) в упрощенной

записи показана на рис. 7.35, *а*. Отметим, что ближайшие к точке (x, y) пикселы в данном варианте определения производных цифрового изображения имеют значение, равное 2. Вычисление градиента в области размерностью 3×3 имеет преимущество по сравнению с использованием уравнений (7.6-40) и (7.6-41) в смысле большего усреднения, что делает градиент менее чувствительным к помехам. В принципе возможно применение и более широких окрестностей для нахождения градиента, но

a	b	c
d	(x, y)	e
g	h	i

а

−1	−2	−1
0	0	0
1	2	1

б

−1	0	1
−2	0	2
−1	0	1

в

Рис. 7.35. Окрестность размерностью 3×3 точки (x, y) (*а*) и маски для вычисления G_x и G_y (*б* и *в*).

в промышленных системах технического зрения распространено использование размерности 3×3 благодаря высокой скорости вычислений и умеренным требованиям к оборудованию для этого случая.

В соответствии с изложенным в разд. 7.6.1 значение G_x по уравнению (7.6-42) можно определить с помощью маски (рис. 7.35, *б*). Аналогично находится и значение G_y (рис. 7.35, *в*). Эти две маски обычно называются *операторами Собеля*. Использование данных масок в произвольной точке (x, y) сочетает в себе результат применения уравнений (7.6-38) или (7.6-39) в виде аппроксимации градиента в этой точке. Перемещая маски по изображению $f(x, y)$, получают градиенты во всех его точках.

Существует множество способов формирования выходного изображения $g(x, y)$, основанных на вычислении градиента. Простейшим способом является задание функции g в точке

(x, y) значения, равного величине градиента входного изображения f в этой точке, т. е.

$$g(x, y) = G[f(x, y)]. \tag{7.6-44}$$

Пример использования такого подхода при получении градиентного изображения приведен на рис. 7.36.

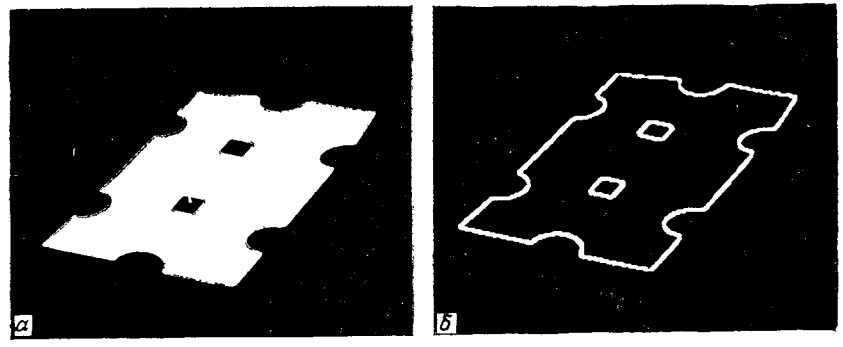


Рис. 7.36. Исходное изображение (*а*) и результат использования уравнения (7.6-44) (*б*).

Другим способом получения дискретного изображения является применение следующих соотношений:

$$g(x, y) = \begin{cases} 1 & \text{при } G[f(x, y)] > T, \\ 0 & \text{при } G[f(x, y)] \leqslant T, \end{cases} \tag{7.6-45}$$

где T — неотрицательная пороговая величина. В этом случае имеют значения только пикселы кромки, градиенты которых превышают величину T. Таким образом, использование уравнения (7.6-45) может рассматриваться как процесс выделения только тех пикселов, которые характеризуются значительным (определенным величиной T) перепадом интенсивности. Дальнейший анализ полученных пикселов обычно требует стирания отдельных точек и связывания пикселов вдоль соответствующих контуров, которые однозначно определяют объекты на изображениях. Применение для этих целей уравнения (7.6-45) рассматривается и иллюстрируется в разд. 8.2.1.

Оператор Лапласа. Оператор Лапласа является оператором производных второго порядка вида

$$L[f(x, y)] = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{7.6-46}$$

Для дискретных изображений оператор Лапласа определяется следующим образом:

$$L[f(x, y)] = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y). \quad (7.6\text{-}47)$$

Эта цифровая запись оператора Лапласа дает нуль на участках постоянной интенсивности и на участках изменяющейся интенсивности на кромках, что характерно для производной второго порядка. Решение уравнения (7.6-47) может быть основано на применении маски, данной на рис. 7.37.

0	1	0
1	-4	1
0	1	0

Рис. 7.37. Маска для вычисления оператора Лапласа.

Хотя, как было отмечено в начале этого раздела, оператор Лапласа определяет переход интенсивности, он редко сам по себе используется для нахождения кромки. Дело в том, что, являясь оператором производных второго порядка, оператор Лапласа дает результат обычно очень чувствительный к помехам. Таким образом, этот оператор, как правило, выполняет второстепенную роль при определении, на какой из сторон (темной или светлой) кромки находится данный пиксел.

7.6.5. Пороговое разделение

Пороговое разделение является одним из основных методов, используемых в промышленных системах технического зрения для обнаружения объектов, особенно в случаях, когда требуется наличие высокой пропускной способности системы. В данном разделе рассматриваются особенности порогового разделения, которые относятся к нижнему уровню обработки информации. Применение порогового разделения на более высоком уровне обработки излагается в гл. 8.

Предположим, что гистограмма интенсивности (рис. 7.38, *а*) соответствует изображению $f(x, y)$, состоящему из светлых объектов на темном фоне. При этом интенсивности пикселов объектов и фона разделяются на две выделяющиеся части. Одним из очевидных путей отделения объектов от фона является выбор порогового значения T, которое разделяет эти группы интенсивностей. Тогда любая точка (x, y), для которой $f(x, y) > T$, принадлежит объекту, а в противном случае — фону. Несколько более общий вариант этого подхода показан на рис. 7.38, *б*, где гистограмма изображения характеризуется тремя отдельными частями, соответствующими, например, двум светлым объектам

на темном фоне. В этом случае используется тот же подход, и точка (x, y) классифицируется как принадлежащая к первому объекту при $T_1 < f(x, y) \leqslant T_2$, ко второму объекту при $f(x,y) > T_2$ и к фону при $f(x, y) \leqslant T_1$.

Данный вариант *многоуровнего порогового разделения* обычно менее надежен, чем одноуровневый вариант, из-за трудности установки нескольких пороговых значений, которые эффективно разделяли бы интересующие области гистограммы, особенно когда их число достаточно велико. Обычно такие задачи при использовании порогового разделения лучше решаются с помощью одной меняющейся пороговой величины (гл. 8).

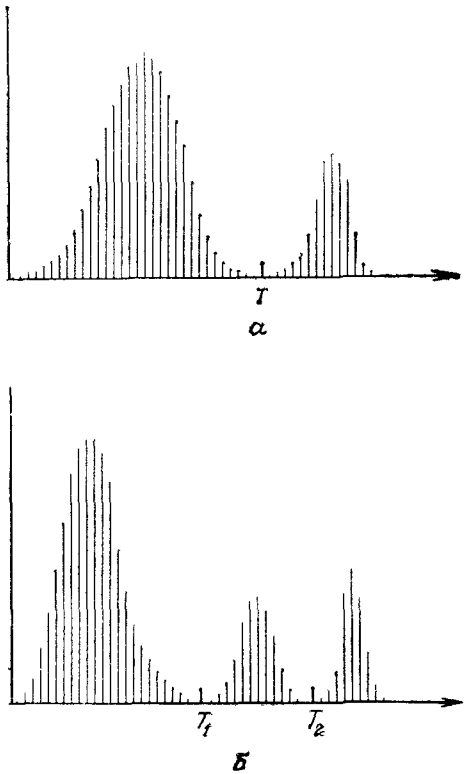


Рис. 7.38. Гистограммы интенсивности, которые могут быть разделены с помощью (*а*) одного порогового значения и (*б*) нескольких пороговых значений.

В соответствии с изложенным подходом можно рассматривать пороговое разделение как операцию, включающую действия с функцией T в виде

$$T = T[x, y, p(x, y), f(x, y)], \quad (7.6\text{-}47)$$

где $f(x, y)$ — интенсивность точки (x, y), а $p(x, y)$ обозначает некоторое частное свойство этой точки, например среднюю интенсивность окрестности с центром в точке (x, y). Пороговое изображение получается путем определения

$$g(x, y) = \begin{cases} 1 & \text{при } f(x, y) > T, \\ 0 & \text{при } f(x, y) \leqslant T. \end{cases} \quad (7.6\text{-}48)$$

Таким образом, на изображении $g(x, y)$ пикселы со значением 1 (или с любым другим соответствующим уровнем интенсивности) относятся к объектам, в то время как пикселы со значением 0 относятся к фону.

Когда величина T зависит только от $f(x, y)$, порог называется *глобальным*. Пример такого порога приведен на рис. 7.38, *а*. Если величина T зависит как от $f(x, y)$, так и от $p(x, y)$, то порог называется *локальным*. Если к тому же величина T зависит от пространственных координат x и y, порог называется *динамическим*. К нижнему уровню технического зрения относят методы порогового разделения, которые основаны на одной глобальной величине T. Поскольку пороговое разделение играет

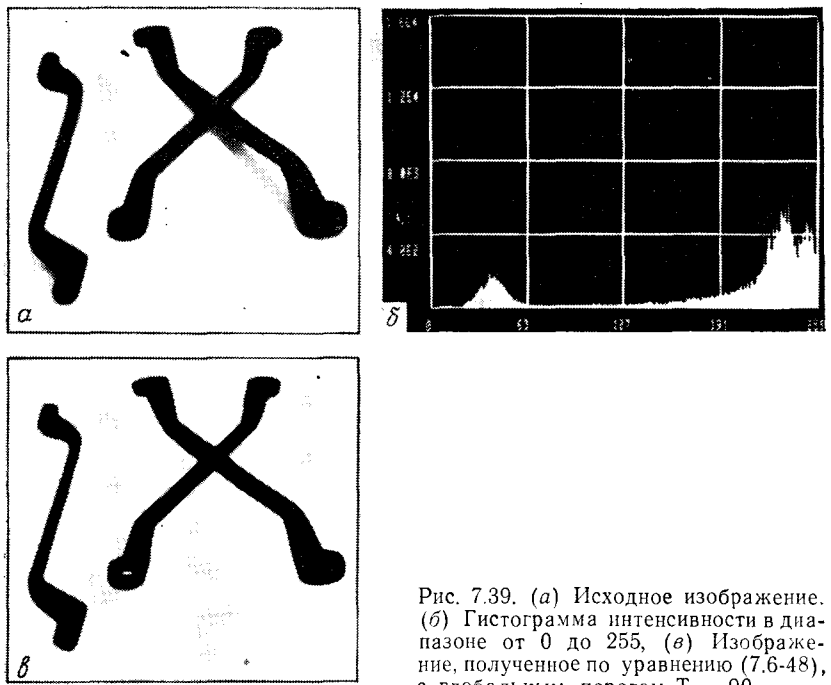


Рис. 7.39. (*а*) Исходное изображение. (*б*) Гистограмма интенсивности в диапазоне от 0 до 255, (*в*) Изображение, полученное по уравнению (7.6-48), с глобальным порогом T = 90.

центральную роль при выделении объектов, его более сложные варианты относятся к среднему уровню технического зрения. Простой пример глобального порогового разделения показан на рис. 7.39.

7.7. ЗАКЛЮЧЕНИЕ

Представленный в этой главе материал охватывает широкий круг методов обработки информации, обычно относящихся к нижнему уровню технического зрения. Как отмечено в разд. 7.1, системы технического зрения решают пространственные задачи, и большинство алгоритмов (особенно на нижнем уровне) основаны на изображениях трехмерных объектов. Важными методами для получения информации о глубине изображения являются методы измерения, изложенные в разд. 7.2, и методы структурированного освещения, описанные в разд. 7.3. Отметим, что многие изложенные способы имеют гораздо бо́льшую область применения. Показательным примером является улучшение качества изображения, которое долго оставалось трудной задачей при обработке цифровых изображений. Одним из основных требований при использовании систем технического зрения — предъявляемые и часто противоречащие друг другу требования низкой стоимости и высокого быстродействия.

Литература

Более подробный материал по оборудованию для получения изображений дан в [79, 119, 74]. Тема, изложенная в разделе 7.3, основана на работах [201, 121, 203]. Преобразования, рассмотренные в разделах 7.4.1 и 7.4.2, могут быть найдены в большинстве книг по вычислительной графике, например в [212]. Дополнительные сведения по моделированию и калибровке камер находятся в [66, 317]. Обзорная статья [12] содержит обширное количество ссылок на литературу по обработке стереоизображений.

Материал раздела 7.5 базируется на работах [288, 252], а раздела 7.6.1 — на [104]. Подробности использования усредняющих факторов даны в [129, 314, 44], а метод сглаживания путем усреднения изображения — в [148]. Методы сглаживания для дискретных изображений, рассмотренные в разделе 7.6.2, основаны на статье [295].

Материал раздела 7.6.3 соответствует работам [101, 315]. Детальное изложение локального улучшения качества дано в [143, 114, 207], а методов определения кромок — в [249]. Более поздний обзор этих методов находится в [54]. Результаты в данной области, учитывающие быстродействие методов, приведены в еще более поздних работах [153, 44]. С вводным материалом по определению кромок можно ознакомиться в [104]. Книга [252] содержит подробное описание методов порогового разделения. Представляет интерес также обзорная статья [305].

Упражнения

7.1. Какой объем памяти требуется для запоминания информации изображения размерностью 512×512, в котором каждый пиксел может иметь 256 значений интенсивности.

7.2. Предложите метод, в котором используется одна световая плоскость для определения диаметра цилиндрических объектов. Предполагается, что используется линейная камера с разрешающей способностью в N пикселов, а также что расстояние между камерой и центром цилиндров постоянно.

7.3. (а) Рассмотрите точность вашего решения задачи 7.2 с точки зрения разрешающей способности камеры (N точек на одной линии) и максимального диаметра цилиндра $D_{макс}$. (б) Какова максимальная ошибка, если N равно 2048, а $D_{макс} = 1$ м?

7.4. Определите изображение пространственной точки с координатами $(1/2, 1/2, \sqrt{2}/2)$ лежащей на оптической оси камеры, размещенной в точке (0, 0, 2) под углом 135° с наклоном 135°. Предполагается, что используется 50-мм линза, а $r_1 = r_2 = r_3 = 0$.

7.5. Из уравнения (7.4-41) получите уравнения (7.4-42) и (7.4-43).

7.6. Покажите, что расстояние D_4 между двумя точками p и q равно кратчайшему из четырех путей между этими точками.

7.7. Покажите, что метод преобразования Фурье для подсчета $F(u)$ может быть использован без вычисления обратных преобразований.

7.8. Проверьте, что подстановка уравнения (7.6-4) в уравнение (7.6-5) дает тождество.

7.9. Приведите булевское выражение, эквивалентное уравнению (7.6-16), для окна размерностью 5×5.

7.10. Разработайте процедуру для вычисления среднего значения расположения соседних элементов в области размерностью $n + n$.

7.11. Объясните, почему метод дискретного выравнивания гистограммы не дает в общем случае ровной гистограммы.

7.12. Предложите метод вычисления локальной гистрограммы для метода улучшения качества, рассмотренного в пункте 7.6.3.

7.13. Результаты, полученные за один проход через изображение некоторых двумерных масок, могут быть получены также с помощью двух проходов одномерной маски. Например, результат использования сглаживающей маски размерностью 3×3 с коэффициентами 1/9 (см. п. 7.6.2) может быть получен также с помощью одного прохода через изображение маски [1 1 1] и второго прохода через маску $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Полученный результат умножается на 1/9. Покажите, что маски Собеля (рис. 7.35) могут быть получены с помощью одного прохода дифференциальной маски вида [—1 0 1] (или ее вертикального эквивалента), а также сглаживающей маски вида [1 2 1] (или ее вертикального эквивалента).

7.14. Покажите, что дискретное преобразование Лапласа, данное в уравнении (7.6-47), пропорционально с коэффициентом —1/4 вычитанию из $f(x, y)$ среднего четырех соседей (x, y).

Глава 8.
СИСТЕМЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ ВЫСОКОГО УРОВНЯ

Только художник может создать произведение, которое затрудняет восприятие.

Теодор Гилл

8.1. ВВЕДЕНИЕ

С целью классификации методов и подходов, используемых в системах технического зрения, мы рассмотрели в разд. 7.1 три основных подкласса: зрение низкого, среднего и высокого уровней. Системы технического зрения низкого уровня, подробно рассмотренные в гл. 7, предназначены для обработки информации с датчиков очувствления.

Эти системы можно отнести к классу «интеллектуальных» машин, если они обладают следующими признаками (признаками интеллектуального поведения):

1) возможностью выделения существенной информации из множества независимых признаков;

2) способностью к обучению на примерах и обобщению этих знаний с целью их применения в новых ситуациях;

3) возможностью восстановления событий по неполной информации;

4) способностью определять цели и формулировать планы для достижения этих целей.

Создание систем технического зрения с такими свойствами для ограниченных видов рабочего пространства в принципе возможно, но характеристики таких систем далеки от возможностей человеческого зрения. В основе технического зрения лежит аналитическая формализация, направленная на решение конкретных задач. Машины с сенсорными характеристиками, близкими к возможностям человека, по-видимому, появятся еще не скоро. Однако отметим, что копирование природы не является единственным решением этой проблемы. Читателю наверняка известны ранние экспериментальные образцы аэропланов с машущими крыльями и другими особенностями полета птиц. Современное решение задачи о полете в пространстве в корне отличается от решений, подсказанных природой. По скорости и достижимой высоте самолеты намного превосходят возможности птиц.

Как уже говорилось в разд. 7.1, системы технического зрения среднего уровня связаны с задачами сегментации, описания и

распознавания отдельных объектов. Как будет видно из следующих разделов, эти задачи охватывают множество подходов, основанных на аналитических представлениях. Системы технического зрения высокого уровня решают проблемы, рассмотренные выше. Для более ясного понимания проблем технического зрения высокого уровня и его связи с техническим зрением низкого и среднего уровней введем ряд ограничений и упростим решаемую задачу.

8.2. СЕГМЕНТАЦИЯ

Сегментацией называется процесс подразделения сцены на составляющие части или объекты. Сегментация является одним из основных элементов работы автоматизированной системы технического зрения, так как именно на этой стадии обработки объекты выделяются из сцены для дальнейшего распознавания и анализа. Алгоритмы сегментации, как правило, основываются на двух фундаментальных принципах: разрывности и подобии. В первом случае основной подход основывается на определении контуров, а во втором — на определении порогового уровня и расширении области. Эти понятия применимы как к статическим, так и к динамическим (зависящим от времени) сценам. В последнем случае движение может служить мощным средством для улучшения работы алгоритмов сегментации.

8.2.1. Проведение контуров и определение границы

Методы, приведенные в разд. 7.6.4, определяют разрывы в интенсивности представления образа объекта. В идеальном случае эти методы определяют пикселы, лежащие на границе между объектом и фоном. На практике данный ряд пикселов редко полностью характеризует границу из-за шума, разрывов на границе вследствие неравномерной освещенности и других эффектов, приводящих к размытию изображения. Таким образом, алгоритмы обнаружения контуров сопровождаются процедурами построения границ объектов из соответствующих последовательностей пикселов. Ниже рассмотрено несколько методик, пригодных для этой цели.

Локальный анализ. Одним из наиболее простых подходов соединения точек контура является анализ характеристик пикселов в небольшой окрестности (например, в окрестности размером 3×3 или 5×5) каждой точки (x, y) образа, который уже подвергся процедуре обнаружения контура. Все точки, являющиеся подобными (определение критерия подобия дано ниже), соединяются, образуя границу из пикселов, обладающих некоторыми общими свойствами.

При таком анализе для установления подобия пикселов контура необходимо определить: 1) величину градиента, требуемого для построения контурного пиксела, и 2) направление градиента. Первая характеристика обозначается величиной $G[f(x, y)]$, определенной в уравнении (7.6-38) или (7.6-39). Таким образом, пиксел контура с координатами (x', y') подобен по величине в определенной ранее окрестности (x, y) пикселу с координатами (x, y), если справедливо неравенство

$$|G[f(x, y)] - G[f(x', y')]| \leqslant T, \tag{8.2-1}$$

где T — пороговое значение.

Направление градиента устанавливается по углу вектора градиента, определенного в уравнении (7.6-37), т. е.

$$\theta = \operatorname{arctg}\left[\frac{G_y}{G_x}\right], \tag{8.2-2}$$

где θ — угол (относительно оси x), вдоль которого скорость изменения имеет наибольшее значение, как показано в разд. 7.6.4. Тогда можно сказать, что угол пиксела контура с координатами (x', y') в некоторой окрестности (x, y) подобен углу пиксела с координатами (x, y) при выполнении следующего неравенства:

$$|\theta - \theta'| < A, \tag{8.2-3}$$

где A — пороговое значение угла. Необходимо отметить, что направление контура в точке (x, y) в действительности перпендикулярно направлению вектора градиента в этой точке. Однако для сравнения направлений неравенство (8.2-3) дает эквивалентные результаты.

Основываясь на этих предположениях, мы соединяем точку в некоторой окрестности (x, y) с пикселом, имеющим координаты (x, y), если удовлетворяются критерии по величине и направлению. Двигаясь от пиксела к пикселу и представляя каждую присоединяемую точку как центр окрестности, процесс повторяется для каждой точки образа. Для установления соответствия между уровнями интенсивности освещения и последовательностями пикселов контура применяется стандартная библиотечная процедура.

Пример. В качестве иллюстрации описанной выше процедуры рассмотрим образ задней дверцы автомобиля, приведенный на рис. 8.1, *а*. Цель состоит в определении размеров прямоугольников, с помощью которых можно построить качественное изображение. Построение таких прямоугольников осуществляется в результате определения строго горизонтальных и вертикальных контуров. На рис. 8.1, *б* и *в* показаны горизонтальные и вертикальные компоненты операторов Соболя, рассмотренные в разд. 7.6.4. На рис. 8.1, *г* показаны результаты соединения точек, которые одновременно имеют градиенты больше 25 и направле-

ния этих градиентов отличаются не более чем на 15 градусов. Горизонтальные линии построены путем последовательного применения этих критериев к каждой строке изображения (рис 8.1, *в*), а вертикальные линии — последовательным сканированием изображения (рис. 8.1, *б*) по столбцам. Дальнейший

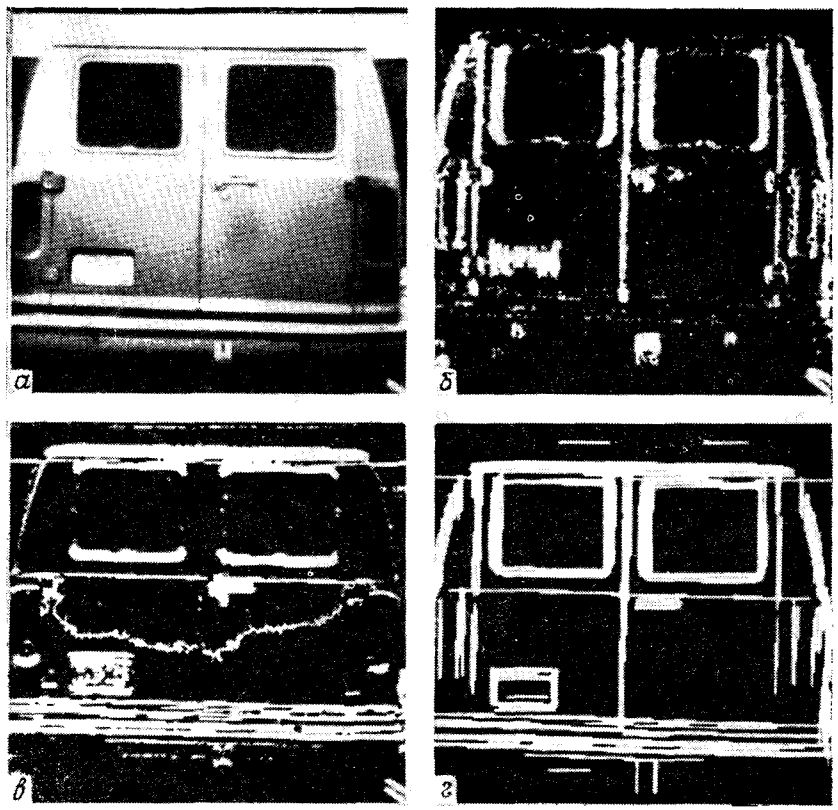


Рис. 8.1. Исходный образ (*а*), горизонтальная составляющая градиента (*б*), вертикальная составляющая градиента (*в*) и результат соединения точек контура (*г*). Предоставлено фирмой Perceptics.

процесс состоял в соединении сегментов контура, разделенных небольшими промежутками, и в объединении отдельных коротких сегментов.

Глобальный анализ с помощью преобразования Хоуга. Рассмотрим метод соединения граничных точек путем определения их расположения на кривой специального вида. Первоначально предполагая, что на плоскости xy образа дано n точек, требуется найти подпоследовательности точек, лежащих на прямых линиях. Одно из возможных решений состоит в построении всех линий, проходящих через каждую пару точек, а затем в нахождении всех подпоследовательностей точек, близких к определенным линиям. Задача, связанная с этой процедурой, заключается в нахождении $n(n-1)/2 \sim n^2$ линий и затем в осуществлении $n[n(n-1)]/2 \sim n^3$ сравнений каждой точки со всеми линиями. Этот процесс трудоемок с вычислительной точки зрения за исключением самых простых приложений.

Данную задачу можно решить по-другому, применяя подход, предложенный Хоугом и называемый *преобразованием Хоуга* [127]. Рассмотрим точку (x_i, y_i) и общее уравнение прямой линии $y_i = ax_i + b$. Имеется бесконечное число линий, проходящих

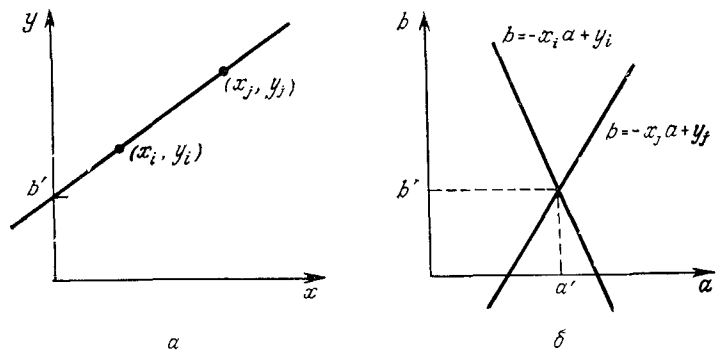


Рис. 8.2. Плоскость xy (*а*) и пространство параметров (*б*).

через точку (x_i, y_i), но все они удовлетворяют уравнению $y_i = ax_i + b$ при различных значениях a и b. Однако, если мы запишем это уравнение в виде $b = -x_i a + y_i$ и рассмотрим плоскость ab (*пространство параметров*), тогда мы имеем уравнение *одной* линии для фиксированной пары чисел (x_i, y_i). Более того, вторая точка (x_j, y_j) также имеет в пространстве параметров связанную с ней линию, которая пересекает другую линию, связанную с точкой (x_i, y_i) в точке (a', b'), где значения a' и b' — параметры линии, на которой расположены точки (x_i, y_i) и (x_j, y_j) в плоскости xy. Фактически все точки, расположенные на этой линии, в пространстве параметров будут иметь линии пересечения в точке (a', b') (рис. 8.2).

Вычислительная привлекательность преобразования Хоуга заключается в разделении пространства параметров на так называемые *собирающие элементы* (рис. 8.3), где $(a_{\text{макс}}, a_{\text{мин}})$ и $(b_{\text{макс}}, b_{\text{мин}})$ — допустимые величины параметров линий. Собирающий элемент $A(i, j)$ соответствует площади, связанной с координатами пространства параметров (a_i, b_j). Вначале эти эле-

менты считаются равными нулю. Тогда для каждой точки (x_k, y_k) в плоскости образа мы полагаем параметр a равным каждому из допустимых значений на оси a и вычисляем соответствующее b, используя уравнение $b = -x_k a + y_k$. Полученное значение b затем округляется до ближайшего допустимого значения на оси b. Если выбор a_p приводит к вычислению b_q, мы полагаем $A(p, q) = A(p, q) + 1$. После завершения этой процедуры значение M в элементе $A(i, j)$ соответствует M точкам в плоскости xy, лежащим на линии $y = a_i x + b_j$. Точность расположения этих точек на одной прямой зависит от числа разбиений плоскости ab. Отметим, что, если мы разбиваем ось a на K частей, тогда для каждой точки (x_k, y_k) мы получаем K значений b, соответствующих K возможным значениям a. Поскольку имеется n точек образа, процесс состоит из nK вычислительных операций. Поэтому приведенная выше процедура линейна относительно n и имеет меньшее число вычислительных операций, чем процедура, описанная выше, если $K \leqslant n$.

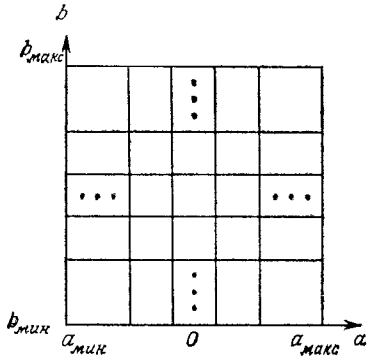


Рис. 8.3. Разбиение плоскости параметров на элементы с целью использования преобразования Хоуга.

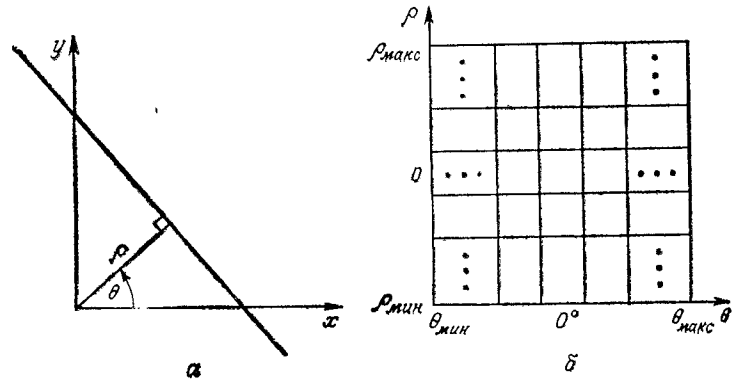
Проблема, связанная с представлением прямой линии уравнением $y = ax + b$, состоит в том, что оба параметра a и b стремятся к бесконечности, если линия принимает вертикальное положение. Для устранения этой трудности используется нормальное представление прямой линии в виде

$$x \cos\theta + y \sin\theta = \beta. \qquad (8.2\text{-}4)$$

Смысл параметров уравнения (8.2-4) ясен из рис. 8.4, *а*. Это представление для построения таблицы собирающих элементов используется так же, как метод, изложенный выше, но вместо прямых линий мы имеем синусоидальные кривые в плоскости $\theta\rho$. Как и прежде, M точек, лежащих на прямой $x \cos\theta_i + y \sin\theta_i = \rho_j$, соответствуют M синусоидальным кривым, которые пересекаются в точке (θ_i, ρ_j) пространства параметров. Если используется метод возрастания θ и нахождения для него соответствующего β, процедура дает M точек в собирающий элемент $A(i, j)$, связанный с точкой (θ_i, ρ_j). Разбиение пространства параметров приведено на рис. 8.4, *б*.

Пример. Использование преобразования Хоуга, основанного на уравнении (8.2-4), проиллюстрировано на рис. 8.5. На рис. 8.5, *в* уровень яркости в плоскости $\theta\rho$ пропорционален числу

точек в собирающих элементах. На этом рисунке абсцисса соответствует θ и ордината ρ в диапазонах $\pm 90°$ и $\pm\rho_{макс}$ соответственно. В данном случае величина $\rho_{макс}$ равна расстоянию от одного угла до другого в исходном образе. Центру рис. 8.5, *в* соответствуют $\theta = 0°$ и $\rho = 0$. Интересно отметить, что яркие пятна (большое число точек в собирающем элементе) около 0° соответствуют вертикальным линиям и около $\pm 90°$ соответствуют горизонтальным линиям рис. 8.5, *б*. На рис. 8.5, *г* показаны линии, которые определены этим методом и накладываются на исходный образ. Различие обусловлено ошибкой дискретизации θ и ρ в пространстве параметров.



Рнт. 8.4. Нормальное представление линии (*а*) и разбиение плоскости $\theta\rho$ на элементы (*б*).

Хотя наше внимание было сосредоточено только на прямых линиях, преобразование Хоуга применимо к любой функции вида $g(\mathbf{x}, \mathbf{c}) = 0$, где $\mathbf{x}$ — вектор координат, а $\mathbf{c}$ — вектор коэффициентов. Например, геометрическое место точек, лежащих на окружности

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2, \qquad (8.2\text{-}5)$$

легко может быть определено с помощью подхода, изложенного выше. Основное отличие заключается в том, что теперь мы имеем три параметра c_1, c_2 и c_3, которые образуют в трехмерном параметрическом пространстве ячейки в форме кубов и собирающие элементы вида $A(i, j, k)$. Процедура состоит в задании приращений c_1 и c_2, решении уравнения (8.2-5) относительно c_3 и в изменении собирающего элемента, соответствующего ячейке, связанной с тройкой (c_1, c_2, c_3). Очевидно, что сложность преобразования Хоуга существенно зависит от числа координат и коэффициентов в данном функциональном представлении.

В заключение отметим, что определение кривых, не имеющих простого аналитического представления, возможно на основе

дальнейшего обобщения преобразования Хоуга. Более подробно этот вопрос рассмотрен в работе [10].

Глобальный анализ с помощью методов теории графов. Изложенные выше методы основаны на задании последователь-

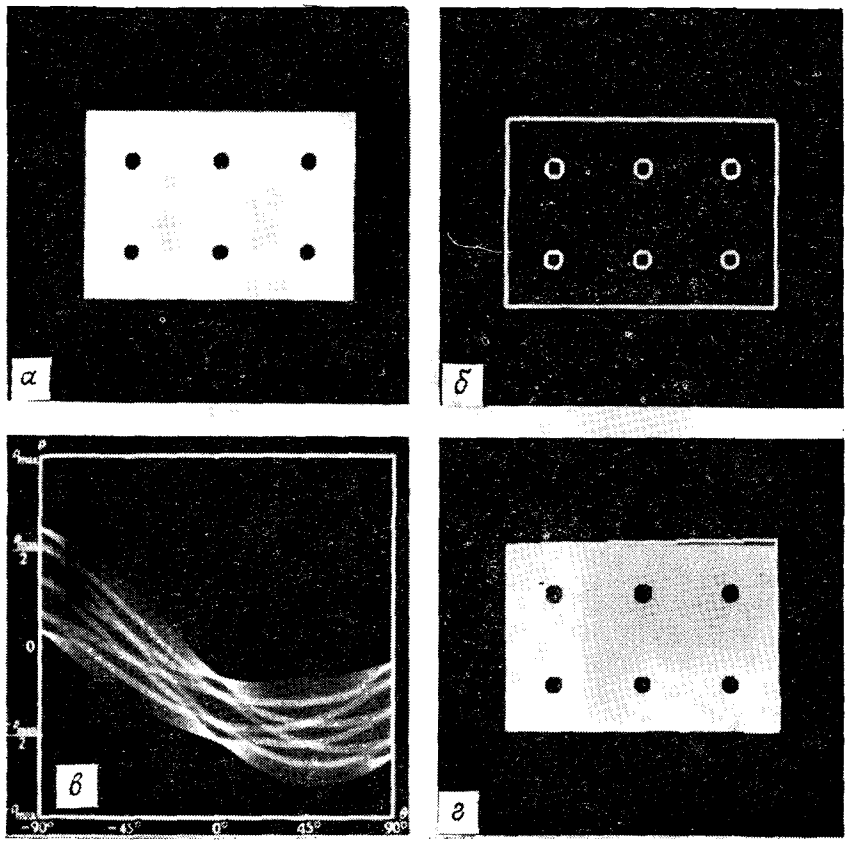


Рис. 8.5. Образ детали (*а*), градиентный образ (*б*), график преобразования Хоуга (*в*) и обнаруженные линии, наложенные на исходный образ (*г*). (Предоставлено D. Cate из фирмы Texas Instruments.)

ности точек контура, полученных в результате градиентного преобразования. Этот метод редко применяется для предварительной обработки данных в ситуациях, характеризуемых высоким уровнем шума, вследствие того, что градиент является производной и усиливает колебания интенсивности. Рассмотрим глобальный подход, основанный на представлении сегментов контура в виде графа и поиске на графе пути наименьшей стоимости, который соответствует значимым контурам. Этот подход представляет приближенный метод, эффективный при наличии шума. Как и следует ожидать, эта процедура значительно сложнее и требует больше времени обработки, чем методы, изложенные выше.

Сначала дадим несколько простых определений. *Граф* $G = (N, A)$ представляет собой конечное, непустое множество вершин N вместе с множеством A неупорядоченных пар различных элементов из N. Каждая пара из A называется *дугой*.

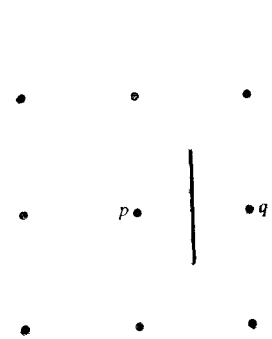


Рис. 8.6. Элемент контура, расположенный между пикселами p и q.

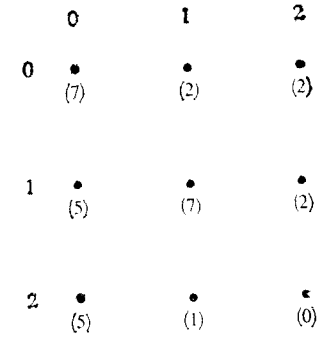


Рис. 8.7. Образ размерностью 3×3.

Граф, в котором дуги являются направленными, называется *направленным графом*. Если дуга выходит из вершины n_i к вершине n_j, тогда n_j называется *преемником* вершины n_i. В этом случае вершина n_i называется *предшественником вершины* n_j. Процесс идентификации преемников каждой вершины называется *расширением* этой вершины. В каждом графе определяются уровни таким образом, чтобы нулевой уровень состоял из единственной вершины, называемой *начальной*, а последний уровень — из вершин, называемых *целевыми*. Каждой дуге (n_i, n_j) приписывается *стоимость* $c(n_i, n_j)$. Последовательность вершин $n_1, n_2, \ldots, n_k$, где каждая вершина n_i является преемником вершины n_{i-1}, называется *путем* от n_1 к n_k, а стоимость пути определяется формулой

$$c = \sum_{i=2}^{k} c(n_{i-1}, n_i). \qquad (8.2\text{-}6)$$

Элемент контура мы определим как границу между двумя пикселами p и q (рис. 8.6). В данном контексте под контуром понимается последовательность элементов контура.

Для иллюстрации применения введенных понятий к определению контура рассмотрим образ размерностью 3×3, приведенный на рис. 8.7, где цифры в скобках обозначают интенсивность,

а без скобок — координаты пикселов. Каждому элементу контура, определенному пикселами p и q, приписываем стоимость

$$c(p, q) = H - [f(p) - f(q)], \qquad (8.2\text{-}7)$$

где H — наибольшее значение интенсивности образа (в данном случае оно равно 7), $f(p)$ — значение интенсивности в точке p, а; $f(q)$ — значение интенсивности в точке q.

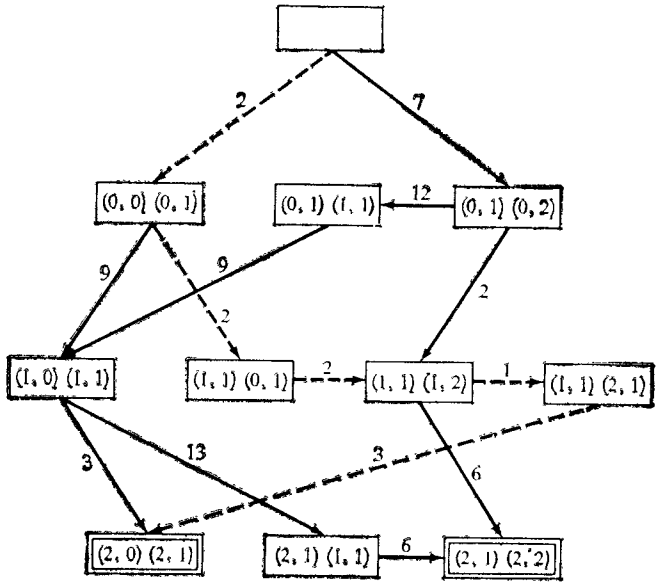


Рис. 8.8. Граф для определения контура образа, приведенного на рис. 8.7. Для каждой вершины пары $(a, b)(c, d)$ соответствуют точкам p и q. Отметим, что если идти по образу сверху вниз, то p будет находиться справа. Путь минимальной стоимости помечен штриховыми линиями [191].

Граф для этой задачи показан на рис. 8.8. Каждая вершина графа соответствует элементу контура, и две вершины связываются дугой в том случае, если два соответствующих элемента контура, взятые последовательно, могут быть частью контура. Стоимость каждого элемента контура, рассчитанная по уравнению (8.2-7), показана дугой, ведущей в этот элемент, а целевые вершины показаны двойными прямоугольниками. Каждый путь между начальной и целевой вершиной является возможным контуром. Для простоты предполагалось, что контур начинается в верхнем ряду и заканчивается в последнем, так что первый элемент контура может быть только [(0, 0), (0, 1)] или [(0, 1), (0, 2)] и последний элемент [(2, 0), (2, 1)] или [(2, 1), (2, 2)]. Путь минимальной стоимости, рассчитанный по уравнению

(8.2-6), показан штриховой линией, а соответствующий контур представлен на рис. 8.9.

Как правило, задача определения пути минимальной стоимости непроста с вычислительной точки зрения. Обычно критерием оптимальности является быстродействие. Алгоритм, приведенный ниже, типичен для класса процедур, использующих эвристики с целью сокращения объема поиска. Пусть $r(n)$ — оценка стоимости пути минимальной стоимости из начальной вершины s в целевую вершину, причем путь проходит через вершину n. Эту оценку можно представить в виде суммы пути минимальной стоимости s в n и оценки стоимости пути из n в целевую вершину, т. е.

$$r(n) = g(n) + h(n). \qquad (8.2\text{-}8)$$

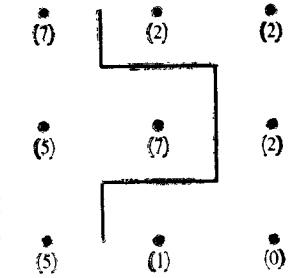


Рис. 8.9. Контур, соответствующий пути минимальной стоимости, показанному на рис. 8.8.

Здесь $g(n)$ является оценкой стоимости пути наименьшей стоимости из s в n, определенного ранее, а $h(n)$ определяется с помощью использования эвристической информации (например, проводя расширение только некоторых вершин, основанное на предварительном вычислении стоимостей при определении путей, ведущих к вершине $h(n)$). Алгоритм поиска на графе, основанный на применении оценки $r(n)$, приведен ниже:

Шаг 1. Поместить стартовую вершину *OPEN* и положить $g(s) = 0$.

Шаг 2. Если нет вершин, помеченных как *OPEN*, то выход на аварийный останов; в противном случае продолжение работы.

Шаг 3. Пометить *CLOSED* помеченную *OPEN* вершину n, оценка которой $r(n)$, определенная по уравнению (8.2-8), имеет наименьшее значение. (Связи для минимальных значений r устанавливаются произвольно, но всегда в пользу целевой вершины.)

Шаг 4. Если n является целевой вершиной, то выход на останов с выдачей пути, полученного в результате обратного прохождения по указателям направления; в противном случае продолжение работы.

Шаг 5. Расширение вершины n, т. е. генерирование всех ее преемников (если преемников нет, переход к шагу 2).

Шаг 6. Если преемник n_i не помечен, положить

$$r(n_i) = g(n) + c(n, n_i),$$

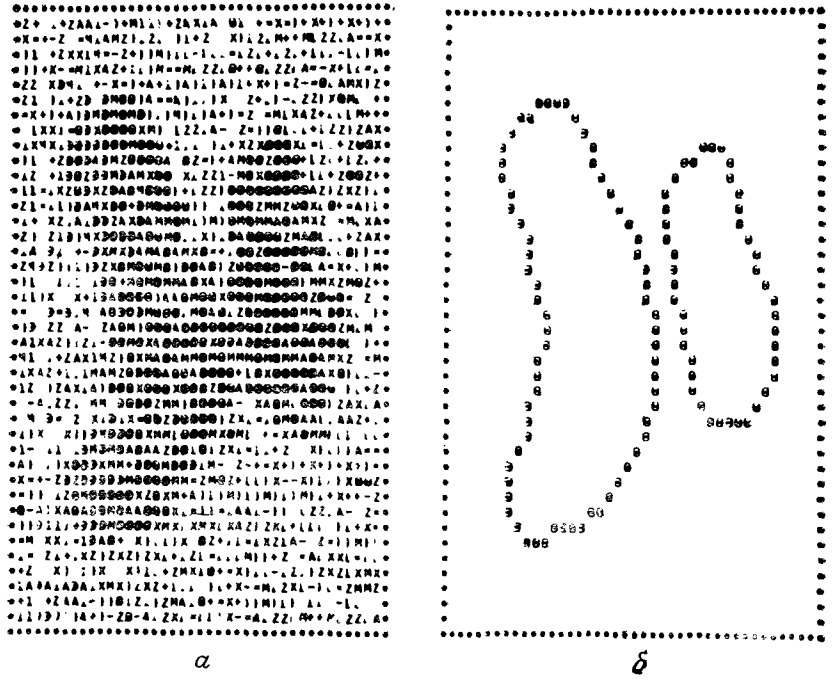
пометить его *OPEN* и направить указатели от него к вершине n.

Шаг 7. Если преемник n_i помечен *CLOSED* или *OPEN*, изменить его значение по формуле

$$g'(n_i) = \min[g(n_i),\ g(n) + c(n,\ n_i)].$$

Пометить *OPEN* те преемники, помеченные *CLOSED*, значения g' которых были уменьшены таким образом. От этих вершин направить указатели к вершине п. Перейти к шагу 2.

Как правило, этот алгоритм не гарантирует нахождение пути минимальной стоимости; его преимущество заключается в быст-



а *б*

Рис. 8.10. Зашумленный образ (*а*) и результат определения контура, полученного с помощью метода эвристического поиска на графе [292] (*б*).

родействии за счет использования эвристик. Однако можно показать, что, если $h(n)$ является нижней границей стоимости пути минимальной стоимости от вершины n к целевой вершине, тогда процедура действительно определит оптимальный путь к целевой вершине [115]. Если эвристическая информация отсутствует, т. е. $h \equiv 0$, процедура сводится к *алгоритму постоянной стоимости* [59].

Пример. Типичный результат, полученный в результате работы этой процедуры, приведен на рис. 8.10. На рис. 8.10, *а* показан зашумленный образ, и на рис. 8.10, *б* результат сегментации контура, полученный с помощью поиска на соответствующем графе путей наименьшей стоимости. В данном примере были использованы эвристики, предотвращающие расширение тех вершин, стоимости которых превышали заданное пороговое значение.

8.2.2. Определение порогового уровня

Понятие порогового уровня (порога) введено в разд. 7.6.5 как тест вида

$$T = T[x,\ y,\ p(x,\ y),\ f(x,\ y)], \tag{8.2-9}$$

где $f(x, y)$ — интенсивность в точке (x, y), $p(x, y)$ — некоторое локальное свойство, определяемое в окрестности этой точки. Пороговое изображение дается следующим выражением:

$$g(x,\ y) = \begin{cases} 1, & \text{если } f(x,\ y) > T, \\ 0, & \text{если } f(x,\ y) \leqslant T, \end{cases} \tag{8.2-10}$$

так что пикселы в $g(x, y)$, имеющие значение 1, соответствуют объектам, а пикселы, имеющие значение 0, соответствуют фону. В уравнении (8.2-10) предполагается, что интенсивность объектов больше интенсивности фона. Противоположное условие получается путем изменения знаков в неравенствах.

Глобальные и локальные пороги. Если значение T в уравнении (8.2-9) зависит только от $f(x, y)$, то, как показано в разд. 7.6.5, порог называется *глобальным*. Если значение T зависит как от $f(x, y)$, так и от $p(x, y)$, порог называется *локальным*. Если, кроме того, T зависит от пространственных координат x и y, в этом случае он называется *динамическим порогом*.

Глобальные пороги применяются в ситуациях, когда имеется явное различие между объектами и фоном и где освещенность достаточно однородна. Методы обратной и структурированной освещенности, изложенные в разд. 7.3, обычно дают изображения, которые могут быть сегментированы путем применения глобальных порогов. Но, как правило, произвольное освещение рабочего пространства приводит к изображениям, которые, если исходить из определения порогового уровня, требуют локального анализа для компенсации таких эффектов, как неоднородность освещения, тени и отражение.

Ниже мы рассмотрим ряд методов для выбора порогов, используемых при сегментации. Хотя некоторые из них могут применяться для выбора глобального порога, они обычно используются в ситуациях, требующих анализа локального порога.

Выбор оптимального порога. Часто рассматривают гистограмму, состоящую из суммы значений функции плотности вероятности. В случае бимодальной гистограммы аппроксимирующая ее функция дается уравнением

$$p(z) = P_1 p_1(z) + P_2 p_2(z), \tag{8.2-11}$$

где интенсивность z — случайная переменная величина, $p_1(z)$ и $p_2(z)$ — функции плотности вероятности, а P_1 и P_2 — *априорные*

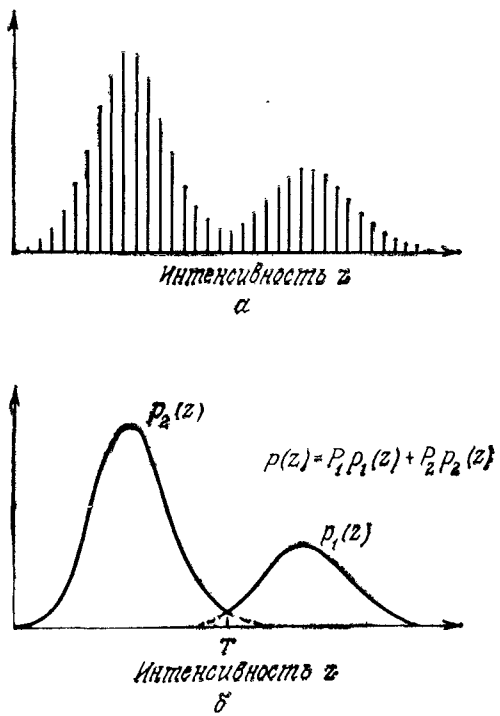


Рис. 8.11. Гистограмма интенсивности (*а*) и ее **аппроксимация в виде** суммы двух функций плотности вероятности (*б*).

вероятности. В данном случае априорные вероятности означают появление двух видов уровней интенсивности на образе. Например, рассмотрим образ, гистограмма которого приведена на рис. 8.11, *а*. Полная гистограмма может быть аппроксимирована суммой двух функций плотности вероятности, как показано на рис. 8.11, *б*. Если известно, что объект состоит из светлых пикселов и они занимают 20 % площади образа, то $P_1 = 0{,}2$. Необходимо, чтобы

$$P_1 + P_2 = 1. \tag{8.2-12}$$

В данном случае это означает, что на остальную часть образа приходится 80 % пикселов фона.

Введем две следующие функции от z:

$$d_1(z) = P_1 p_1(z), \tag{8.2-13}$$

$$d_2(z) = P_2 p_1(z). \tag{8.2-14}$$

Из теории принятия решений [290] известно, что средняя ошибка определения пиксела объекта в качестве фона (и наоборот) минимизируется с помощью следующего правила: рассматривая пиксел со значением интенсивности z, мы подставляем это значение z в уравнения (8.2-13) и (8.2-14). Затем мы определяем пиксел как пиксел объекта, если $d_1(z) > d_2(z)$, или как пиксел фона, если $d_2(z) > d_1(z)$. Тогда оптимальный порог определяется величиной z, для которой $d_1(z) = d_2(z)$. Таким образом, полагая в уравнениях (8.2-13) и (8.2-14) $z = T$, получаем, что оптимальный порог удовлетворяет уравнению

$$P_1 p_1(T) = P_2 p_2(T). \tag{8.2-15}$$

Итак, если известны функциональные зависимости $p_1(z)$ и $p_2(z)$, это уравнение можно использовать для нахождения оптимального порога, который отделяет объекты от фона. Если этот порог известен, уравнение (8.2-10) может быть использовано для сегментации данного образа.

В качестве важной иллюстрации применения уравнения (8.2-15) предположим, что $p_1(z)$ и $p_2(z)$ являются плотностями вероятности распределения Гаусса, т. е.

$$p_1(z) = \frac{1}{\sqrt{2\pi}\,\sigma_1} \exp -\left[\frac{(z-m_1)^2}{2\sigma_1^2}\right], \tag{8.2-16}$$

$$p_2(z) = \frac{1}{\sqrt{2\pi}\,\sigma_2} \exp -\left[\frac{(z-m_2)^2}{2\sigma_2^2}\right]. \tag{8.2-17}$$

В этих выражениях положим $z = T$, подставим их в уравнение (8.2-15) и после упрощения получим квадратное уравнение относительно T:

$$AT^2 + BT + C = 0, \tag{8.2-18}$$

где

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2, \\ B &= 2(m_1\sigma_2^2 - m_2\sigma_1^2), \\ C &= \sigma_1^2 m_2^2 - \sigma_2^2 m_1^2 + 2\sigma_1^2\sigma_2^2 \ln\frac{\sigma_2 P_1}{\sigma_1 P_2}. \end{aligned} \tag{8.2-19}$$

Возможность двух решений указывает на то, что для получения оптимального решения могут потребоваться два значения порогового уровня.

Если стандартные отклонения равны ($\sigma_1 = \sigma_2 = \sigma$), в этом случае имеется единственное значение порогового уровня:

$$T = \frac{m_1 + m_2}{2} + \frac{\sigma^2}{m_1 - m_2} \ln \frac{P_2}{P_1}. \qquad (8.2\text{-}20)$$

Если $\sigma = 0$ или $P_1 = P_2$, оптимальное значение порогового уровня является арифметическим средним математических ожиданий. Сформулированное условие означает, что интенсивности объекта и фона постоянны для всего образа. Последнее условие означает, что пикселы объекта и фона имеют одинаковую вероятность появления, причем каждый раз число пикселов объекта равно числу пикселов фона.

Пример. Рассмотрим сегментацию механических деталей, показанных на рис. 8.12, *а*, где мы пренебрежем решеткой, наложенной на образ. На рис. 8.12, *б* приведен результат вычисления глобальной гистограммы, соответствующей бимодальной гауссовой плотности распределения вероятности, устанавливающей оптимальный глобальный порог, который окончательно используется в уравнении (8.2-10) для сегментации образа. Видно, что отклонения в интенсивности делают этот подход практически непригодным. Однако он может быть реализован на локальных участках образа, определенных решеткой на рис. 8.12, *а*.

После разбиения образа на подобразы для каждого из них вычисляется гистограмма и проводится тест на бимодальность. Бимодальные гистограммы соответствуют смешанной гауссовой плотности распределения вероятности, и оптимальное значение порогового уровня вычисляется по уравнениям (8.2-18) и (8.2-19). Для подобразов, не имеющих бимодальных гистограмм, значения пороговых уровней не вычисляются. Для них эти значения вычисляются путем интерполяции порогов соседних подобразов, являющихся бимодальными. Гистограммы для каждого подобраза приведены на рис. 8.12, *в*, где горизонтальные линии обозначают относительный диапазон этих гистограмм. В конце процедуры для определения порогового значения $T(x, y)$ проводится от точки к точке повторная интерполяция соседних пороговых значений. Отметим, что в данном случае мы имеем дело с динамическим значением порогового уровня, поскольку оно зависит от пространственных координат (x, y). Изменение $T(x, y)$ в зависимости от положения показано на рис. 8.12, *г*.

Окончательно пороговый образ создается в результате сравнения каждого пиксела в первоначальном образе с соответствующим пороговым уровнем. На рис. 8.12, *д* показан результат использования этого метода в данном частном случае. Налицо улучшение по сравнению с применением глобального порогового уровня. Отметим, что данный метод предполагает установление порогового уровня для каждой ячейки, а локальные пороги ин-

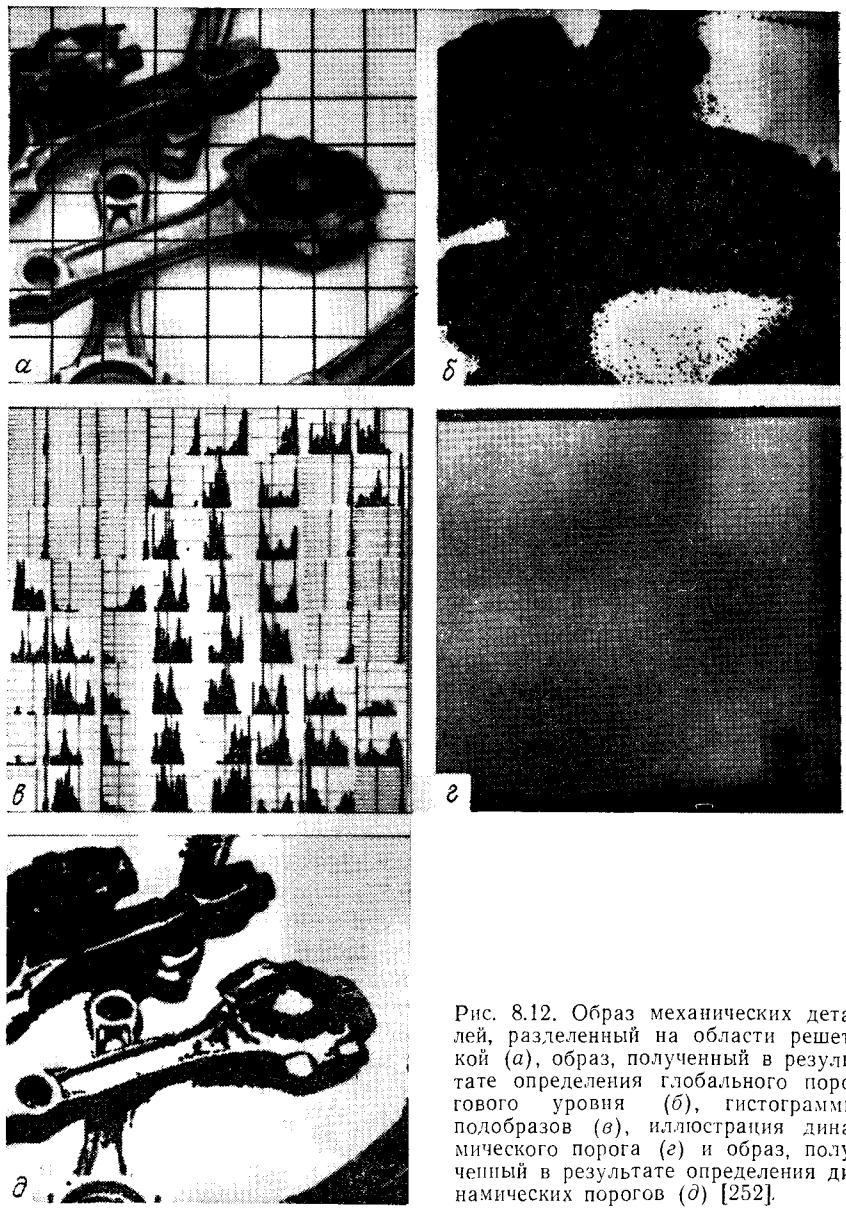


Рис. 8.12. Образ механических деталей, разделенный на области решеткой (*а*), образ, полученный в результате определения глобального порогового уровня (*б*), гистограммы подобразов (*в*), иллюстрация динамического порога (*г*) и образ, полученный в результате определения динамических порогов (*д*) [252].

терполируются с целью создания динамического порога, который окончательно используется при сегментации.

Изложенный выше подход применяется для выбора пороговых уровней изображения. Предположим, что мы можем моделировать многомодальную гистограмму в виде суммы n функций плотности вероятности:

$$p(z) = P_1 p_1(z) + \ldots + P_n p_n(z). \qquad (8.2\text{-}21)$$

Тогда задача оптимизации порогового уровня сводится к определению принадлежности данного пиксела к одной из n возможных категорий. Правило минимизации ошибки теперь основывается на n функциях вида

$$d_i(z) = P_i p_i(z), \quad i = 1, 2, \ldots, n. \qquad (8.2\text{-}22)$$

Данный пиксел интенсивности z принадлежит k-й категории, если $d_k(z) > d_j(z)$, $j = 1, 2, \ldots, n$; $j \neq k$. Как прежде, оптимальный пороговый уровень между категорией k и категорией j, обозначаемый T_{kj}, определяется из уравнения

$$P_k p_k(T_{kj}) = P_j p_j(T_{kj}). \qquad (8.2\text{-}23)$$

Как указывалось в разд. 7.6.5, при получении пороговых уровней на основе многомодальных гистограмм актуальной является задача идентификации гистограммных мод.

Определение порогового уровня на основе характеристик границы. Одним из наиболее важных аспектов при выборе порогового уровня является возможность надежно идентифицировать модовые пики для данной гистограммы. Это важно при автоматическом выборе порогового уровня в ситуациях, когда характеристики образа меняются вследствие большого разброса интенсивности. Из изложенного выше очевидно, что возможность выбора «хорошего» порогового уровня может быть существенно увеличена в случае, если пики гистограмм являются высокими, узкими, симметричными и разделены глубокими провалами.

Одним из подходов для улучшения вида гистограмм является рассмотрение только тех пикселов, которые лежат на границе (или около нее) между объектами и фоном. Одно из очевидных улучшений состоит в том, что этот подход позволяет получать гистограммы менее зависимыми от отношения между объектом и фоном. Например, гистограмма интенсивности образа, составленного из маленького объекта на большой площади постоянного фона, определялась бы большим пиком вследствие концентрации пикселов фона. С другой стороны, результирующие гистограммы имели бы пики с более сбалансированными высотами, если бы рассматривались пикселы, лежащие только на (или около) границе между объектом и фоном. Кроме того, вероятность расположения пиксела на границе объекта практически равна вероятности того, что он лежит на границе фона, что

улучшает симметрию гистограммных пиков. Окончательно, как показано ниже, использование пикселов, которые удовлетворяют некоторым простым критериям, основанным на операторах градиента и Лапласа, приводит к увеличению провалов между пиками гистограммы.

Выше мы неявно подразумевали, что граница между объектами и фоном известна. Очевидно, что во время проведения сегментации эта информация отсутствует, поскольку нахождение раздела между объектами и фоном является окончательной целью приведенной здесь процедуры. Однако из разд. 7.6.4 известно, что, вычислив градиент пиксела, можно определить, лежит ли он или не лежит на контуре. Кроме того, лапласиан может дать информацию о том, лежит ли данный пиксел на темной (т. е. фон) или светлой (объект) стороне контура. Как установлено в разд. 7.6.4, с внутренней стороны идеального контура лапласиан равен нулю, поэтому на практике можно ожидать, что провалы гистограмм, образованных пикселами, выбранными по критерию градиент/лапласиан, будут располагаться достаточно редко и иметь желаемую высоту.

Градиент $G[f(x, y)]$ любой точки образа определяется уравнением (7.6-38) или (7.6-39), лапласиан $L[f(x, y)]$ — уравнением (7.6-47). Эти два свойства можно использовать для формирования трехуровнего образа:

$$s(x, y) = \begin{cases} 0, & \text{если } G[f(x, y)] < T, \\ +, & \text{если } G[f(x, y)] \geqslant T \text{ и } L[f(x, y)] \geqslant 0, \\ -, & \text{если } G[f(x, y)] \geqslant T \text{ и } L[f(x, y)] < 0, \end{cases} \qquad (8.2\text{-}24$$

где символы 0, +, — представляют три различных уровня освещенности, а T — пороговый уровень. Предположим, что темный объект располагается на светлом фоне, тогда в соответствии с рис. 7.34, *б* применение уравнения (8.2-24) дает образ $s(x, y)$, в котором все пикселы, не лежащие на контуре (для них значение $G[f(x, y)]$ меньше T, помечены 0, все пикселы на темной стороне контура помечены + и все пикселы на светлой стороне контура помечены —. Для светлого объекта на темном фоне символы + и — в уравнении (8.2-24) меняются местами. На рис. 8.13 показан объект, построенный на светлом фоне по уравнению (8.2-24).

Только что изложенная процедура может применяться для создания сегментированного, бинарного образа, в котором 1 соответствует объектам, представляющим интерес, и 0 — фону. Отметим, что перемещение (вдоль горизонтальных или вертикальных линий сканирования) от светлого фона к темному объекту должно характеризоваться заменой знака — фона на + объекта $s(x, y)$. Внутренняя область объекта состоит из пикселов, помеченных либо 0 либо +. Окончательно перемеще-

ние от объекта к фону характеризуется заменой знака + на —. Таким образом, горизонтальные или вертикальные линии сканирования, содержащие части объекта, имеют следующую структуру:

$$(\ldots)(-, +)(0 \text{ или } +)(+, -)(\ldots),$$

где (...) является произвольной комбинацией +, — или 0. Остальные скобки содержат точки объекта и помечены 1. Все

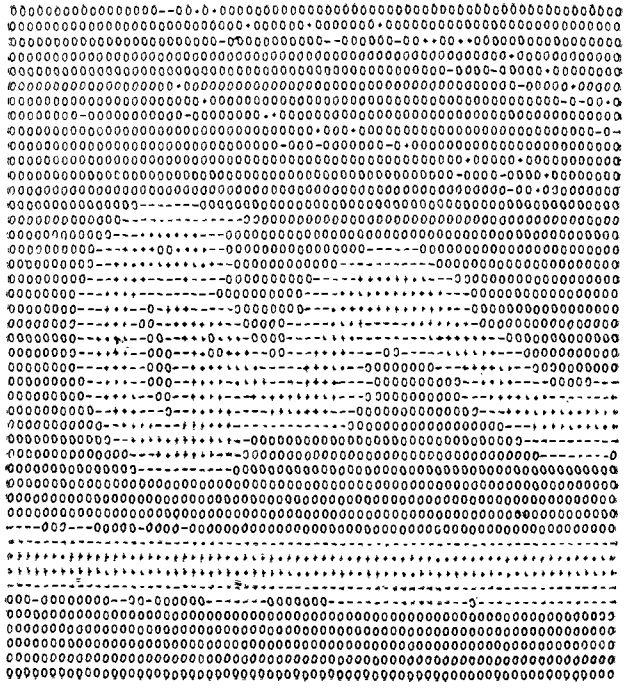
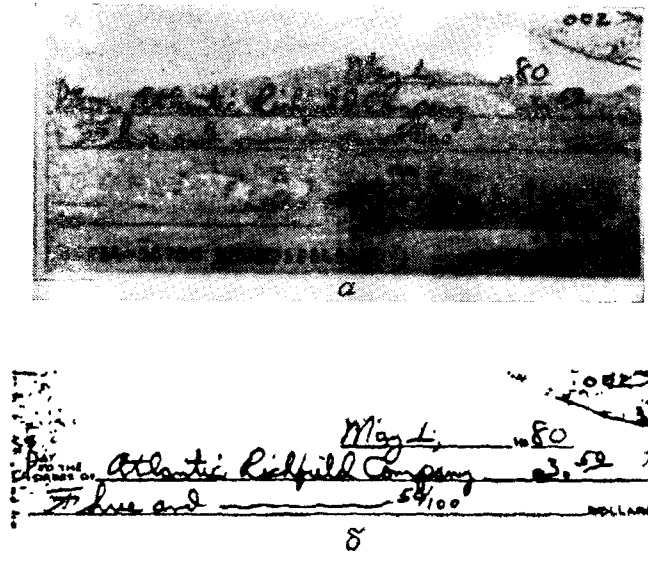


Рис. 8.13. Образ объекта, построенный с помощью уравнения (8.2-24) [306].

другие пикселы вдоль той же линии сканирования помечаются 0, за исключением всех последовательностей из (0 или +), ограниченных (—, +) и (+, —).

Пример. Для иллюстрации только что изложенного материала рассмотрим рис. 8.14, *а*, на котором показан образ обычного банковского чека. На рис. 8.15 приведена гистограмма, определяющая число пикселов, градиенты которых превышают 5. Отметим, что эта гистограмма обладает свойствами, описанными выше, т. е. она имеет две основные симметричные моды одинаковой высоты, разделенные хорошо различимым провалом. На рис. 8.14, *б* показан сегментированный образ, полученный по

уравнению (8.2-24) со значением T, примерно равным средней точке провала. Приведенный выше анализ последовательности дает бинарный результат.



а

б

Рис. 8.14. Исходный образ (*а*) и сегментированный образ (*б*) [306].

Определение порогового уровня, основанное на нескольких переменных. Изложенные выше методы связаны с определением порогового уровня для единственного переменного значения интенсивности. В некоторых приложениях можно использовать более одной переменной для характеристики каждого пиксела образа, увеличивая таким образом не только степень различия между объектом и фоном, но и между самими объектами. Одним из наиболее значимых примеров является цветное зрение, где используются красные, зеленые и голубые компоненты (КЗГ) для формирования составного цветного образа. В этом случае каждый пиксел характеризуется тремя переменными и это позволяет строить трехмерную гистограмму. Основная

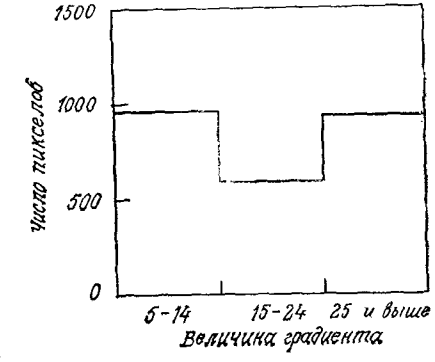


Рис. 8.15. Гистограмма пикселов, градиенты которых превышают 5 [306].

процедура та же, что и для одной переменной. Пусть, например, даны три 16-уровневых изображения, соответствующие КЗГ компонентам датчика цвета. Сформируем кубическую решетку $16 \times 16 \times 16$ и поместим в каждый элемент пикселы, КЗГ компоненты которых имеют интенсивности, соответствующие координатам, определяющим положение этого элемента. Число точек в каждом элементе решетки может быть затем разделено на общее число пикселов образа для формирования нормированной гистограммы.

Теперь выбор порога заключается в нахождении групп точек в трехмерном пространстве, где каждая «компактная» группа аналогична основной моде гистограммы одной переменной. Например, предположим, что мы ищем две значимые группы точек данной гистограммы, где одна группа соответствует объекту, а другая — фону. Принимая во внимание, что теперь каждый пиксел имеет три компоненты и может быть рассмотрен как точка трехмерного пространства, можно сегментировать образ с помощью следующей процедуры. Для каждого пиксела образа вычисляется расстояние между этим пикселом и центром каждой группы. Тогда, если пиксел располагается рядом с центром группы точек объекта, мы помечаем его 1; в противном случае мы помечаем его 0. Это понятие легко распространить на большую часть компонентов пиксела и соответственно на большую часть групп. Основная сложность состоит в том, что определение значимых групп, как правило, приводит к довольно сложной задаче, поскольку число переменных возрастает. Читатель, желающий более подробно ознакомиться с методами определения групп, может, например, обратиться к работе [290]. Другие методы сегментации изложены в работе [90].

Пример. В качестве иллюстрации подхода, основанного на применении гистограмм от многих переменных, рассмотрим рис. 8.16. На рис. 8.16, *а* представлен одноцветный образ цветной фотографии. Исходный цветной образ был составлен из трех 16-уровневых КЗГ образов. Для наших целей существенно отметить, что шарф и один из цветков на платье были ярко-красные, а волосы и лицо светлые и отличались по спектральным характеристикам от окна и других объектов деталей фона.

Рис. 8.16, *б* получен в результате определения порогового уровня для группы точек гистограммы, о которой было известно, что она содержит КЗГ компоненты, представляющие собой телесные тона. Важно отметить, что окно, которое в одноцветном образе имеет такой же диапазон интенсивности, как и волосы, не проявилось на сегментированном образе вследствие различия их мультиспектральных характеристик. Тот факт, что некоторые малые области волос в верхней части головы объекта проявились на сегментированном образе, говорит о том, что они телесного цвета. Рис. 8.16, *в* получен путем определения порогового

уровня группы точек, близкой к красной оси. В этом случае только шарф, красный цветок и несколько изолированных точек проявились на сегментированном образе. Пороговый уровень, использованный при получении обоих результатов, не превышал размера одного элемента кубической решетки. Таким образом,

Рис. 8.16. Сегментация с помощью подхода, основанного на применении гистограмм от многих переменных [104].

все пикселы с компонентами, не превышающими единичного расстояния от центра группы точек, были закодированы как белые. Любые другие пикселы рассматривались как черные.

8.2.3. Областно-ориентированная сегментация

Основные определения. Целью сегментации является разделение образа на области. В разд. 8.2.1 мы рассматривали эту проблему как нахождение границ между областями на основе принципа разрыва интенсивности, в то время как в разд. 8.2.2 сегментация осуществлялась после определения пороговых

уровней, зависящих от таких свойств пикселов, как интенсивность или цвет. В этом разделе мы рассмотрим методы сегментации, основанные на прямом нахождении областей.

Пусть R — область образа. Рассмотрим сегментацию как процесс разбиения R на n подобластей $R_1, R_2, \ldots, R_n$, так что

1. $\bigcup_{i=1}^{n} R_i = R$;
2. R_i — связная область, $i = 1, 2, \ldots, n$;
3. $R_i \cap R_j = \varnothing$ для всех i и j, $i \neq j$;
4. $P(R_i)$ есть ИСТИНА для $i = 1, 2, \ldots, n$;
5. $P(R_i \cup R_j)$ есть ЛОЖЬ для $i \neq j$, где $P(R_i)$ — логический предикат, определенный на точках из множества R_i, и $\varnothing$ — пустое множество.

Условие 1 означает, что сегментация должна быть полной, т. е. каждый пиксел должен находиться в образе. Второе условие требует, чтобы точки в области были связными (разд. 7.5.2). Условие 3 указывает на то, что области не должны пересекаться. Условие 4 определяет свойства, которым должны удовлетворять пикселы в сегментированной области. Простой пример: $P(R_i) =$ ИСТИНА, если все пикселы в R_i имеют одинаковую интенсивность. Условие 5 означает, что области R_i и R_j различаются по предикату P. Применение этих условий в алгоритмах сегментации рассматривается в следующих разделах.

Расширение области за счет объединения пикселов. Расширение области сводится к процедуре группирования пикселов или подобластей в бо́льшие объединения. Простейшей из них является *агрегирование пикселов*. Процесс начинается с выбора множества узловых точек, с которых происходит расширение области в результате присоединения к узловым точкам соседних пикселов с похожими характеристиками (интенсивность, текстура или цвет). В качестве простой иллюстрации этой процедуры рассмотрим рис. 8.17, *а*, на котором цифры внутри ячеек указывают интенсивность. Пусть точки с координатами (3, 2) и (3, 4) используются как узловые. Выбор двух начальных точек приведет к сегментации образа на две области: области R_1, связанной с узлом (3, 2), и области R_2, связанной с узлом (3, 4). Свойство P, которое мы будем использовать для того, чтобы отнести пиксел к той или иной области, состоит в том, что модуль разности между интенсивностями пиксела и узловой точки не превышает пороговый уровень T. Любой пиксел, удовлетворяющий этому свойству одновременно для обоих узлов, произвольно попадает в область R_1. На рис. 8.17, *б* показан результат, полученный для $T = 3$. В этом случае сегментация проводится для двух областей, причем точки в R_1 обозначаются буквой *a*, точки в R_2 буквой *b*. Необходимо отметить, что независимо от того, в какой из этих двух областей будет взята начальная точка, окончательный результат будет один и тот же. Если, с другой стороны, выбрать $T = 8$, была бы получена единственная область (рис. 8.17, *в*).

Предыдущий пример, несмотря на его простоту, иллюстрирует некоторые важные проблемы расширения области. Двумя очевидными проблемами являются: выбор начальных узлов для правильного представления областей, представляющих интерес, и определение подходящих свойств для включения точек в различные области в процессе расширения. Выбор множества, состоящего из одной или нескольких начальных точек, следует из постановки задачи. Например, в военных приложениях объекты, представляющие интерес, имеют более высокую температуру, чем фон, и поэтому проявляются более ярко. Выбор наиболее ярких пикселов является естественным начальным шагом в алгоритме процесса расширения области. При отсутствии априорной информации можно начать с вычисления для каждого пиксела набора свойств, который наверняка будет использован при установлении соответствия пиксела той или иной области в процессе расширения. Если результатом вычислений являются группы точек (кластеры), тогда в качестве узловых берутся те пикселы, свойства которых близки к свойствам центроидов этих групп. Так, в примере, приведенном выше, гистограмма интенсивностей показала бы, что точки с интенсивностью от одного до семи являются доминирующими.

	1	2	3	4	5
1	0	0	5	6	7
2	1	1	5	8	7
3	0	1	6	7	7
4	2	0	7	6	6
5	0	1	5	6	5

а

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

б

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

в

Рис. 8.17. Пример расширения области на основе использования известных начальных точек [104].

Выбор критерия подобия зависит не только от задачи, но также от вида имеющихся данных об образе. Например, анализ

информации, полученной со спутников, существенно зависит от использования цвета. Задача анализа значительно усложнится при использовании только монохроматических образов. К сожалению, в промышленном техническом зрении возможность получения мультиспектральных и других дополнительных данных об образе является скорее исключением, чем правилом. Обычно анализ области должен осуществляться с помощью набора дескрипторов, включающих интенсивность и пространственные характеристики (моменты, текстуру) одного источника изображения. В разд. 8.3 приведены дескрипторы, используемые для описания области.

Отметим, что применение только одних дескрипторов может приводить к неправильным результатам, если не используется информация об условиях связи в процессе расширения области. Это легко продемонстрировать при рассмотрении случайного расположения пикселов с тремя различными значениями интенсивности. Объединение пикселов в «область» на основе признака одинаковой интенсивности без учета условий связи приведет к бессмысленному результату при сегментаци.

Другой важной проблемой при расширении области является формулировка условия окончания процесса. Обычно процесс расширения области заканчивается, если больше не существует пикселов, удовлетворяющих критерию принадлежности к той или иной области. Выше упоминались такие критерии, как интенсивность, текстура и цвет, которые являются локальными по своей природе и не учитывают «историю» процесса расширения области. Дополнительный критерий, повышающий мощность алгоритма расширения области, включает понятие размера, схожести между пикселом-кандидатом и только что созданными пикселами (сравнение интенсивности кандидата и средней интенсивности области), а также формы области, подлежащей расширению. Использование этих типов дескрипторов основано на предположении, что имеется неполная информация об ожидаемых результатах.

Разбиение и объединение области. Изложенная выше процедура расширения области начинает работу с заданного множества узловых точек. Однако можно сначала разбить образ на ряд произвольных непересекающихся областей и затем объединять и/или разбивать эти области с целью удовлетворения условий, сформулированных в начале раздела. Итеративные алгоритмы разбиения и объединения, работа которых направлена на выполнение этих ограничений, могут быть изложены следующим образом.

Пусть R является полной областью образа, на которой определен предикат P. Один из способов сегментации R состоит в успешном разбиении площади образа на все меньшие квадратные области, так что для каждой области R_i, $P(R_i) =$ ИСТИНА. Процедура начинает работу с рассмотрения всей области R. Если $P(R) =$ ЛОЖЬ, область разбивается на квадранты. Если для какого-либо квадранта P принимает значение ЛОЖЬ, этот квадрант разбивается на подквадранты и т. д. Этот метод разбиения обычно представляется в виде так называемого квадродерева (дерева, у которого каждая вершина имеет только четыре потомка). На рис. 8.18 приведен простой пример квадродерева. Отметим, что корень дерева соответствует всему образу,

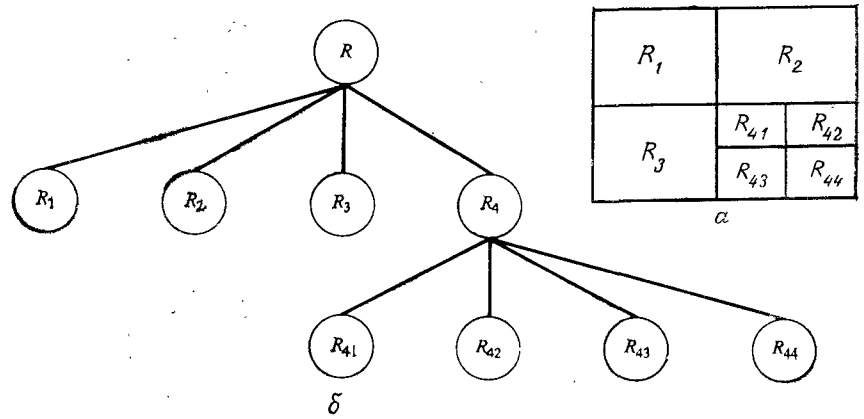


Рис. 8.18. Образ, разделенный на области ($а$), и соответствующее квадродерево ($б$).

а каждая вершина — разбиению. В данном случае только R_4 подлежит дальнейшему разбиению. Если применять только операцию разбиения, можно ожидать, что в результате окончательного разбиения всей площади образа на подобласти последние будут иметь одинаковые свойства. Это можно устранить допустимым объединением так же, как и разбиением. Для того чтобы удовлетворить условиям сегментации, введенным выше, необходимо объединять только те соседние области, пикселы которых удовлетворяют предикату P; таким образом, две соседние области R_i и R_k объединяются только в том случае, если $P(R_i \cup R_k) =$ ИСТИНА.

Изложенное выше можно представить в виде процедуры, где на каждом шаге выполняются следующие операции:

1. Разбиение области R_i, для которой $P(R_i) =$ ЛОЖЬ, на четыре непересекающихся квадранта.
2. Объединение соседних областей R_j и R_k, для которых $P(R_j \cup R_k) =$ ИСТИНА.
3. Выход на останов, когда дальнейшее объединение или разбиение невозможно.

Возможны варианты этого алгоритма [125]. Например, можно сначала разбить образ на квадратные блоки. Дальней-

шее разбиение выполняется по изложенному выше способу, но вначале объединение ограничивается группами из четырех блоков, являющихся в квадродереве потомками и удовлетворяющих предикату P. Когда дальнейшее объединение этого типа становится невозможным, процедура завершается окончательным объединением областей согласно шагу 2. В этом случае объединяемые области могут иметь различный размер. Основным преимуществом этого подхода является использование одного квадродерева для разбиения и объединения до шага, на котором происходит окончательное объединение.

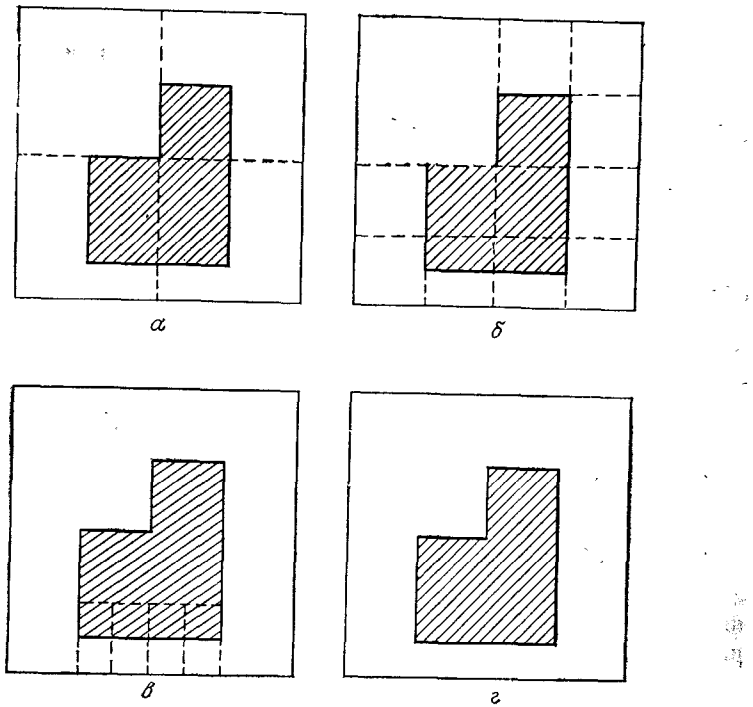


Рис. 8.19. Пример алгоритма разбиения и объединения.

Пример. Проиллюстрируем изложенный выше алгоритм разбиения и объединения рис. 8.19. В данном случае образ состоит из одного объекта и фона. Для простоты мы предполагаем, что интенсивности объекта и фона постоянные и $P(R_i) =$ ИСТИНА, если все пикселы в R_i имеют одинаковую интенсивность. Из этого следует, что для всей области образа $P(R) =$ ЛОЖЬ, поэтому образ разбивается так, как показано на рис. 8.19, *а*. На следующем шаге только область в левом верхнем углу удовлетворяет предикату, и поэтому она остается неизменной, в то время как другие разбиваются на подквадранты (рис. 8.19, *б*). В этом случае можно объединить несколько областей, за исключением двух подквадрантов, включающих нижнюю часть объекта; эти подквадранты не удовлетворяют предикату и поэтому должны подвергнуться дальнейшей операции разбиения. Результаты операции разбиения и объединения приведены на рис. 8.19, *в*. На этом шаге все области удовлетворяют предикату P, и объединение соответствующих областей после последней операции разбиения дает окончательный, сегментированный результат (рис. 8.19, *г*)

8.2.4. Применение движения

Движение представляет собой мощное средство, которое используется человеком и животными для выделения интересующих их объектов из фона. В системах технического зрения роботов движение используется при выполнении различных операций на конвейере, при перемещении руки, оснащенной датчиком, более редко при перемещении всей робототехнической системы. В этом подразделе мы рассмотрим применение движения для сегментации с точки зрения распознавания образов.

Основной подход. Один из наиболее простых подходов для определения изменений между двумя кадрами изображения (образами) $f(x, y, t_i)$ и $f(x, y, t_j)$, взятыми соответственно в моменты времени t_i и t_j, основывается на сравнении соответствующих пикселов этих двух образов. Для этого применяется процедура, заключающаяся в формировании так называемой *разности* образов.

Предположим, что мы имеем эталонный образ, имеющий только стационарные компоненты. Если сравним этот образ с таким же образом, имеющим движущиеся объекты, то разность двух образов получается в результате вычеркивания стационарных компонент (т. е. оставляются только ненулевые записи, которые соответствуют нестационарным компонентам изображения).

Разность между двумя кадрами изображения, взятыми в моменты времени t_i и t_j, можно определить следующим образом:

$$d_{ij}(x, y) = \begin{cases} 1, & \text{если } |f(x, y, t_i) - f(x, y, t_j)| > \theta, \\ 0 & \text{в противном случае,} \end{cases} \qquad (8.2\text{-}25)$$

где θ — значение порогового уровня. Отметим, что $d_{ij}(x, y)$ принимает значение 1 для пространственных координат (x, y) только в том случае, если два образа в точке с этими координатами существенно различаются по интенсивности, что определяется значением порогового уровня θ.

При анализе движущегося образа все пикселы изображений разности $d_{ij}(x, y)$, имеющие значение 1, рассматриваются как результат движения объекта. Этот подход приметим только в том случае, если два образа зарегистрированы и освещенность имеет относительно постоянную величину в пределах границ, устанавливаемых пороговым уровнем θ. На практике

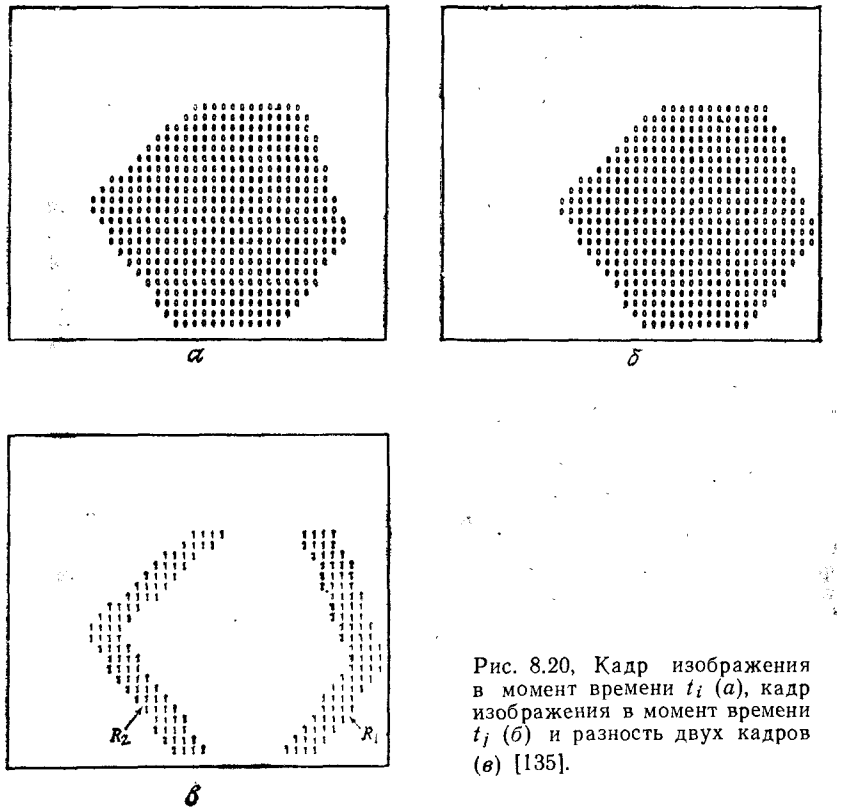


Рис. 8.20, Кадр изображения в момент времени t_i ($а$), кадр изображения в момент времени t_j ($б$) и разность двух кадров ($в$) [135].

записи в $d_{ij}(x, y)$, имеющие значение 1, часто появляются в результате действия шума. Обычно на разности двух кадров изображения такие значения выглядят как изолированные точки. Для их устранения применяется простой подход, заключающийся в формировании 4- или 8-связных областей из единиц в $d_{ij}(x, y)$, и затем пренебрегают любой областью с числом записей, меньшим заранее заданного. При этом можно не распознать малые и/или медленно движущиеся объекты, но это увеличивает вероятность того, что остающиеся записи в разности двух кадров изображения действительно соответствуют движению. Проиллюстрируем эти рассуждения рис. 8.20. На рис. 8.20, $а$ приведен эталонный кадр изображения, который соответствует моменту времени t_i и состоит из одного объекта постоянной интенсивности, движущегося с равномерной скоростью по поверхности фона, также имеющего постоянную интенсивность. На рис. 8.20, $б$ приведен кадр в момент времени t_j, а на рис. 8.20, $в$ разность кадров рассчитанная по уравнению (8.2-25) с пороговым уровнем, большим постоянной интенсивности фона. Необходимо отметить, что в процессе определения разности кадров возникли две несвязные области: одна область является результатом выделения контура передней части, а другая — задней части движущегося объекта.

Аккумулятивная разность. Как говорилось выше, разность кадров благодаря шуму часто содержит изолированные записи. Несмотря на то что число таких записей может быть сокращено или полностью ликвидировано в результате анализа связности пороговых уровней, этот процесс может также привести к потере изображений малых или медленно движущихся объектов. Ниже излагается подход для решения этой проблемы путем рассмотрения изменения в расположении пикселов на нескольких кадрах, т. е. в процесс вводится «память». Основная идея заключается в пренебрежении теми изменениями, которые возникают случайно в последовательности кадров и, таким образом, могут быть отнесены к случайному шуму.

Рассмотрим последовательность кадров изображения $f(x, y, t_1)$, $f(x, y, t_2)$, ..., $f(x, y, t_n)$ и допустим, что $f(x, y, t_1)$ является эталонным образом. Изображение аккумулятивной разности формируется в результате сравнения эталонного образа с каждым образом в данной последовательности. В процедуре построения изображения аккумулятивной разности имеется счетчик, предназначенный для учета расположения пикселов. Его значение увеличивается каждый раз, когда возникает различие в расположении соответствующих пикселов эталонного образа и образа из рассматриваемой последовательности. Таким образом, когда k-й кадр сравнивается с эталонным, запись в данном пикселе аккумулятивной разности означает, во сколько раз интенсивность пиксела k-го кадра отличается от интенсивности пиксела эталонного образа. Различия устанавливаются, например, с помощью уравнения (8.2-25).

Приведенные выше рассуждения иллюстрируются рис. 8.21. На рис. 8.21, $а$—$д$ приведены образы прямоугольного объекта (обозначенного нулями), движущегося вправо с постоянной скоростью 1 пиксел/кадр. Эти образы приведены в моменты времени, соответствующие одному перемещению пиксела. На рис. 8.21, $а$ изображен кадр эталонного образа, на рис. 8.21, $б$—$г$ со 2-го по 4-й кадры последовательности, а на рис. 8.21, $д$ — 11-й кадр. Рис. 8.21, $е$—$и$ соответствуют изображениям аккумулятивной разности, которые можно объяснить следующим образом. На рис. 8.21, $е$ левая колонка из 1 обусловлена различием между

объектом на рис. 8.21, *а* и фоном на рис. 8.21, *б*. Правая колонка из 1 вызвана различием между фоном эталонного образа и передним контуром движущегося объекта. Ко времени появления

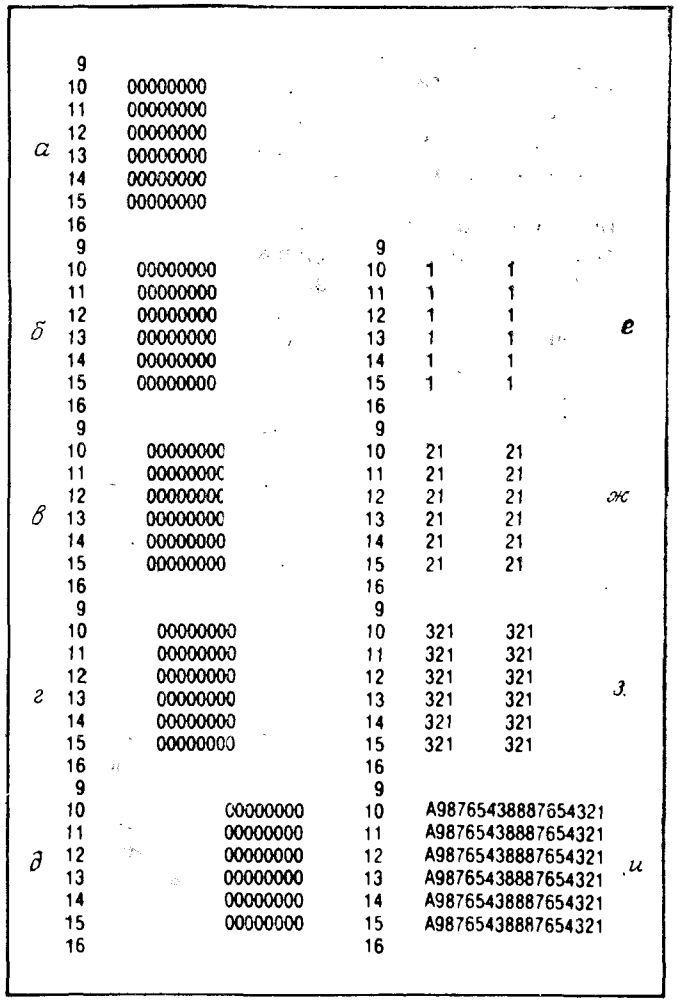


Рис. 8.21. Кадр эталонного образа (а), *б* — *д* соответственно 2-, 3-, 4- и 11-й кадры, *е* — *и* — изображения аккумулятивной разности для 2-, 3-, 4- и 11-го кадров [135].

4-го кадра (рис. 8.21, *г*), первый ненулевой столбец изображения аккумулятивной разности указывает на три отсчета, что соответствует трем основным различиям между этим столбцом в эталонном образе и соответствующим столбцом в последующих

кадрах. На рис. 8.21, *и* показано общее число из 10 (представленных «*A*» в шестнадцатеричной системе счисления) изменений этого положения. Остальные записи на этом рисунке объясняются аналогично.

Нередко полезно рассматривать три типа изображений аккумулятивной разности: абсолютное, положительное и отрицательное. Последние два получаются из уравнения (8.2-25), в

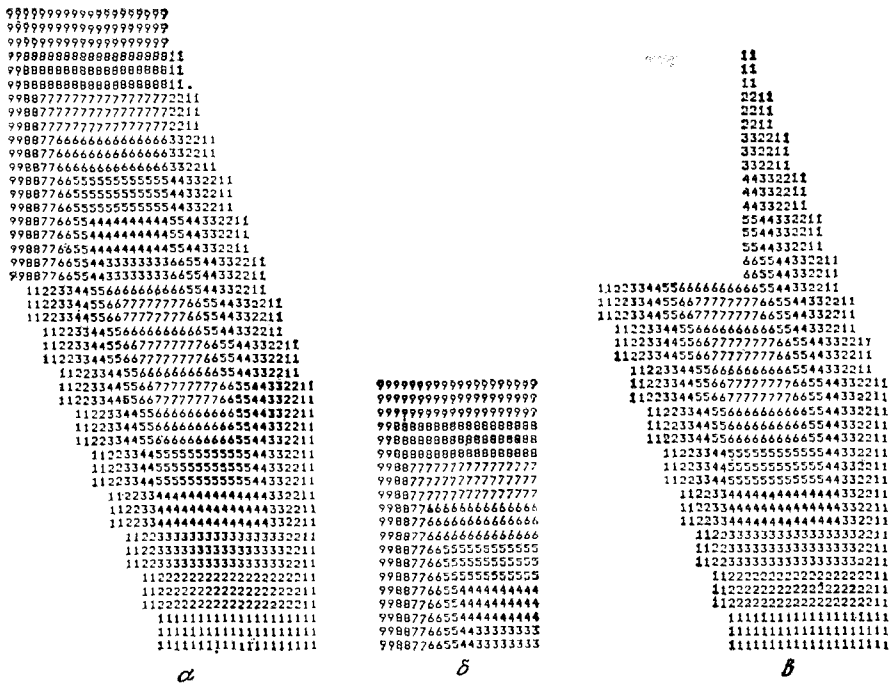


Рис. 8.22. Абсолютное (*а*), положительное (*б*) и отрицательное (*в*) изображения аккумулятивной разности для объекта размером 20 × 20 пикселов с интенсивностью, превышающей фон, движущегося в юго-восточном направлении [136].

котором нет модуля, а вместо $f(x, y, t_i)$ подставляется значение эталонного кадра. Предполагая, что числовые значения интенсивности объекта превышают значения фона в случае, когда разность положительна, она сравнивается с положительным значением порогового уровня; если отрицательна, сравнение выполняется с отрицательным значением порогового уровня. Это определение заменяется на противоположное, если интенсивность объекта меньше фона.

Пример. На рис. 8.22, *а* — *в* показаны абсолютное, положительное и отрицательное изображения аккумулятивной разности

размером 20×20 пикселов, причем интенсивность пикселов объекта больше интенсивности фона. Объект движется с постоянной скоростью в юго-восточном направлении. Важно отметить, что пространственное расширение положительного изображения прекращается, когда объект покидает начальное положение. Другими словами, когда объект, интенсивность которого больше фона, полностью выходит из положения, соответствующего положению на эталонном образе, то генерация новых записей для

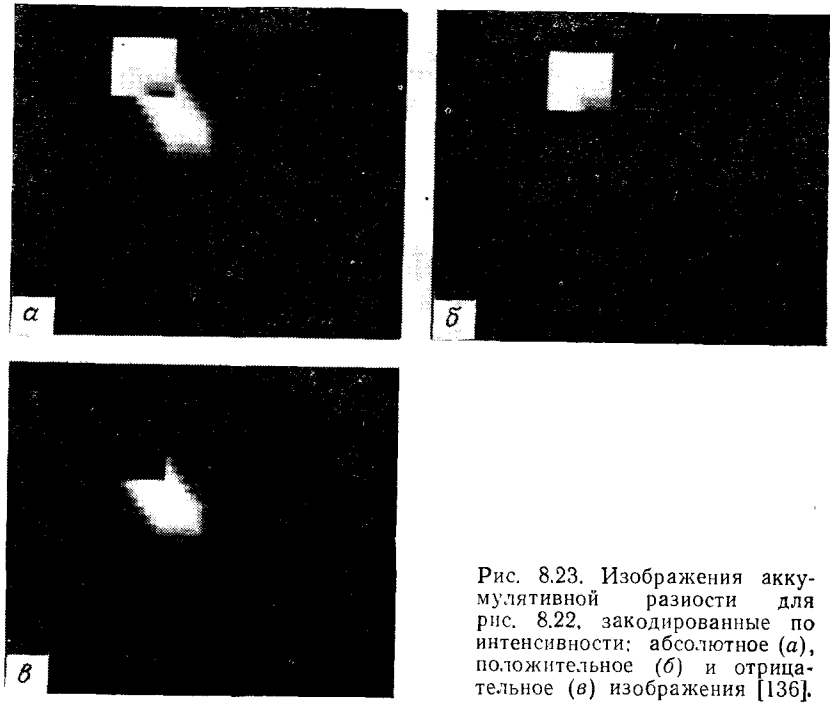


Рис. 8.23. Изображения аккумулятивной разности для рис. 8.22, закодированные по интенсивности: абсолютное (*а*), положительное (*б*) и отрицательное (*в*) изображения [136].

положительного изображения аккумулятивной разности прекращается. Таким образом, после завершения генерации положительное изображение аккумулятивной разности дает исходное расположение объекта на эталонном кадре. Как показано ниже, такое свойство полезно при создании эталона из движущейся последовательности образов. Из рис. 8.22 видно, что абсолютное изображение содержит области как положительного, так и отрицательного изображения, и записи в этих образах указывают скорость и направление движения объекта. Форма кодирования интенсивности образов, приведенных на рис. 8.22, дана на рис. 8.23.

Определение эталонного образа. Успех применения методов, изложенных в предыдущих двух разделах, зависит от эталонного образа, относительно которого проводятся дальнейшие сравнения. Как уже говорилось выше, различие между двумя образами в задаче распознавания движущихся объектов определяется путем исключения стационарных компонент при сохранении элементов, соответствующих шуму и движущимся объектам. Проблема выделения образа из шума решается методом фильтрации или с помощью формирования изображения аккумулятивной разности.

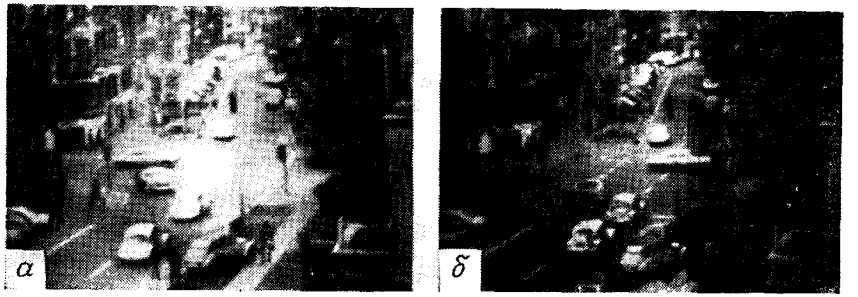


Рис. 8.24. Два кадра изображения сцены дорожного движения. Имеются два основных движущихся объекта: белый автомобиль в центре и пешеход в левой нижней части сцены [135].

На практике не всегда можно получить эталонный образ, имеющий только стационарные элементы, и это приводит к необходимости построения эталона из набора образов, содержащих один или более движущихся объектов. Это особенно характерно для ситуаций, описывающих сцены со многими быстроменяющимися объектами или в случаях, когда возникают частые изменения сцен. Рассмотрим следующую процедуру генерации эталонного образа. Предположим, что мы рассматриваем первый образ последовательности в качестве эталонного. Когда нестационарная компонента полностью вышла из своего положения в эталонном кадре, соответствующий фон в данном кадре может быть перенесен в положение, первоначально занимаемое объектом в эталонном кадре. Когда все движущиеся объекты полностью покинули свои первоначальные положения, в результате этой операции воссоздается эталонный образ, содержащий только стационарные компоненты. Перемещение объекта можно определить с помощью операции расширения положительного изображения аккумулятивной разности.

Пример. Иллюстрация описанного выше подхода приведена на рис. 8.24 и 8.25. На рис. 8.24 показаны два кадра изображе-

ния перекрестка. Первое изображение рассматривается в качестве эталонного, а второе воспроизводит эту же сцену в некоторый более поздний момент времени. Основными движущимися деталями являются автомобиль, движущийся слева направо, и пешеход, пересекающий улицу в нижней левой части картины.

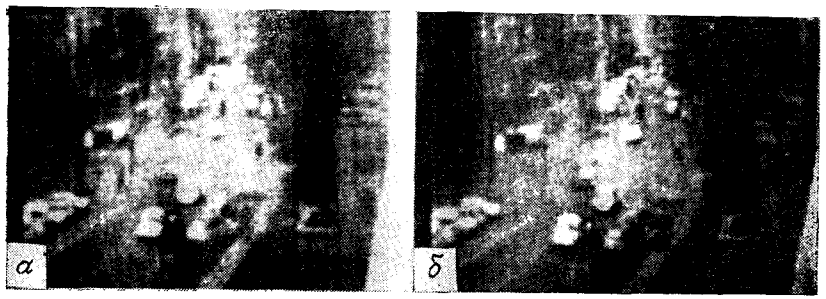


Рис. 8.25. Изображение, в котором устранен автомобиль и сохранен фон (*а*), и изображение, в котором устранен пешеход и сохранен фон. Последнее изображение можно использовать как эталонное [135].

Устранение движущегося автомобиля показано на рис. 8.25, *а*, а устранение пешехода — на рис. 8.25, *б*.

8.3. ОПИСАНИЕ

В системах технического зрения проблемой описания называется выделение свойств (деталей) объекта с целью распознавания. В идеальном случае дескрипторы не должны зависеть от размеров, расположения и ориентации объекта, но должны содержать достаточное количество информации для надежной идентификации объектов. Описание является основным результатом при конструировании систем технического зрения в том смысле, что дескрипторы должны влиять не только на сложность алгоритмов распознавания, но также и на их работу. В разд. 8.3.1, 8.3.2 и 8.4 мы рассмотрим три основные категории дескрипторов: дескрипторы границы, дескрипторы области и дескрипторы для описания трехмерных структур.

8.3.1. Дескрипторы границы

Цепные коды. Цепные коды применяются для представления границы в виде последовательности отрезков прямых линий определенной длины и направления. Обычно в основе этого представления лежит 4- или 8-связная прямоугольная решетка (рис. 8.26). Длина каждого отрезка определяется разрешением решетки, а направления задаются выбранным кодом. Отметим,

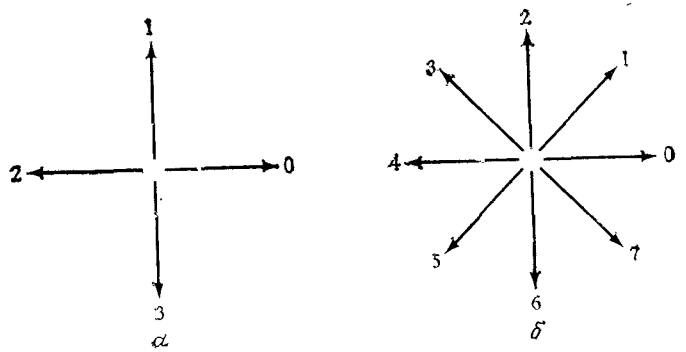


Рис. 8.26. 4-направленный цепной код (*а*) и 8-направленный цепной код (*б*).

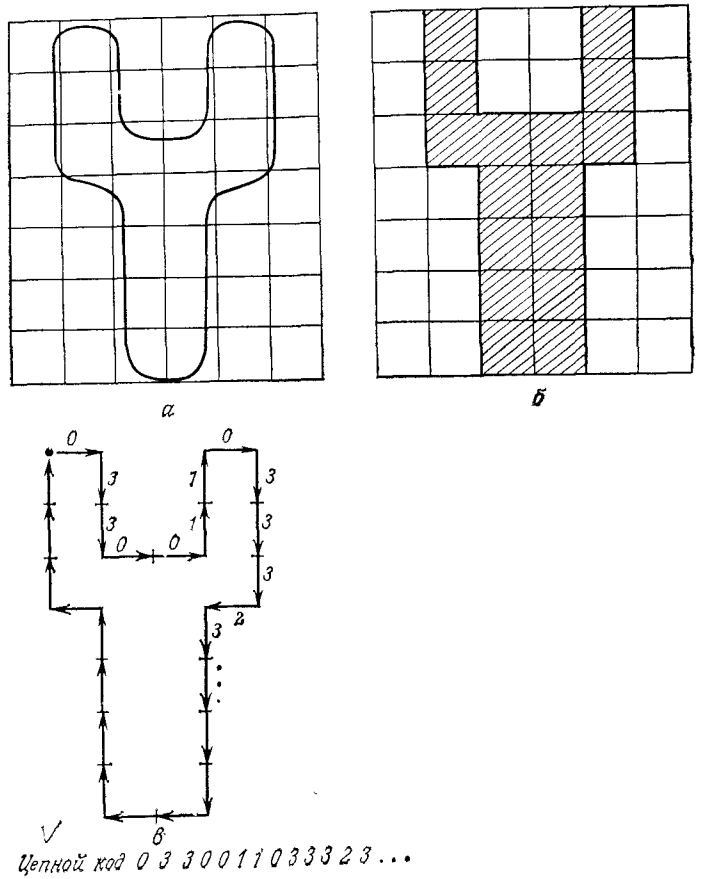


Рис. 8.27. Этапы получения цепного кода.

что для представления всех направлений в 4-направленном цепном коде достаточно 2 бит, а для 8-направленного цепного кода требуется 3 бит. Конечно, можно определить цепные коды с бóльшим числом направлений, но на практике наиболее часто используются коды, приведенные на рис. 8.26.

Для порождения цепного кода заданной границы сначала выбирается решетка (рис. 8.27, *а*). Тогда, если площадь ячейки, расположенной внутри границы, больше определенного числа (обычно 50 %), ей присваивается значение 1; в противном случае этой ячейке присваивается значение 0. Данный процесс иллюстрируется рис. 8.27, *б*, где ячейки со значением 1 отмечены темными линиями. Окончательно мы кодируем границу между двумя областями, используя направления, заданные решеткой на рис. 8.26, *а*. Результат кодирования в направлении по часовой стрелке с началом в месте, помеченном точкой, приведен на рис. 8.27, *в*. Альтернативная процедура состоит в разбиении границы на участки равной длины (каждый участок имеет одно и то же число пикселов) и соединении граничных точек каждого участка прямой линией, а затем присваивания каждой линии направления, ближайшего к одному из допустимых направлений цепного кода. Пример такого подхода, использующего четыре направления, приведен на рис. 8.28.

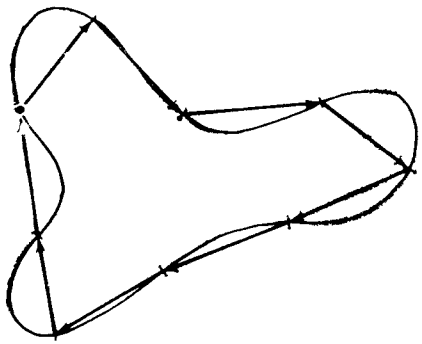


Рис. 8.28. Генерация цепного кода с помощью разбиения границы.

Важно отметить, что цепной код данной границы зависит от начальной точки. Однако можно нормировать код с помощью простой процедуры. Для создания цепного кода начальная точка на решетке выбирается произвольным образом. Рассматривая цепной код как замкнутую последовательность индексов направлений, мы вновь выбираем начальную точку таким образом, чтобы результирующая последовательность индексов была целым числом, имеющим минимальную величину. Также можно нормировать повороты, если вместо цепного кода рассматривать его первую разность. Первая разность вычисляется в результате отсчитывания (в направлении против часовой стрелки) числа направлений, разделяющих два соседних элемента кода. Например, первая разность для цепного кода с 4 направлениями 10103322 есть 3133030. Если рассматривать код как замкнутую последовательность, тогда первый элемент разности

можно вычислить, используя переход между последним и первым компонентами цепи. В данном примере результатом является 33133030. Нормирование можно осуществить путем разбиения всех границ объекта на одинаковое число равных сегментов и последующей подгонкой длин сегментов кода с целью их соответствия этому разбиению, как показано на рис. 8.28.

Изложенные методы нормирования являются точными только в том случае, когда сами границы инвариантны к повороту и изменению масштаба. Этот случай редко встречается на практике. Например, один и тот же объект, разбитый на элементы в двух различных направлениях. как правило, имеет разную

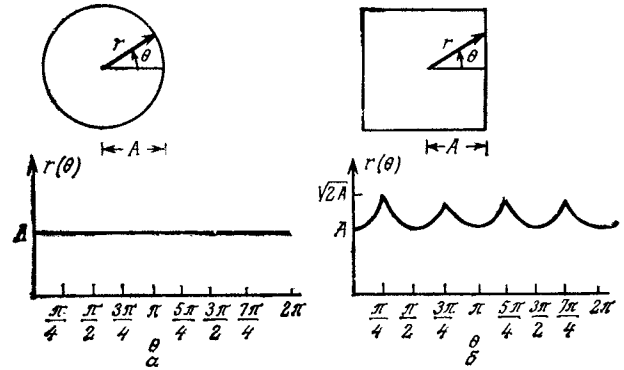


Рис. 8.29. Две простые формы границы и их сигнатуры как функции угла.

форму границы, причем степень различия пропорциональна разрешающей способности изображения. Этот эффект можно уменьшить, если выбирать длины элементов цепи бóльшими, чем расстояния между пикселами дискретного образа, или же выбирать ориентацию решетки (рис. 8.27) вдоль главных осей кодируемого объекта. Этот вопрос рассматривается ниже в разделе, посвященном индексам формы.

Сигнатуры. Сигнатурой называется одномерное функциональное представление границы. Известно несколько способов создания сигнатур. Одним из наиболее простых является построение отрезка из центра к границе как функции угла (рис. 8.29). Очевидно, что такие сигнатуры зависят от периметра области и начальной точки. Нормирование периметра можно осуществить, пронормировав кривую $r(\theta)$ максимальным значением. Проблему выбора начальной точки можно решить определив сначала цепной код границы, а затем применив метод, изложенный в предыдущем разделе. Конечно, расстояние, зависящее от угла, не является единственным способом определения сигнатуры. Например, можно провести через границу прямую линию и определить угол между касательной к гарнице и этой линией как функцию

положения вдоль границы [4]. Полученная сигнатура, хотя и отличается от кривой $r(\theta)$, несет информацию об основных характеристиках формы границы. Например, горизонтальные участки кривой соответствовали бы прямым линиям вдоль границы, поскольку угол касательной здесь постоянен. Один из вариантов этого метода в качестве сигнатуры использует так называемую *функцию плотности наклона* [206]. Эта функция представляет собой гистограмму значений угла касательной. Поскольку гистограмма является мерой концентрации величин, функция плотности наклона строго соответствует участкам границы с постоянными углами касательной (прямые или почти прямые участки) и имеет глубокие провалы для участков, соответствующих быстрому изменению углов (выступы или другие виды изгибов).

После получения сигнатуры мы должны ее описать таким образом, чтобы можно было различать сигнатуры, соответствующие различным формам границы. Как правило, эта проблема решается просто, поскольку теперь мы имеем дело с одномерными функциями. Метод, часто применяемый для определения отличительных признаков сигнатуры, состоит в определении ее *моментов*. Пусть a — дискретная случайная величина, означающая отклонения амплитуды сигнатуры, и пусть $p(a_i)$, $i = 1, 2, \ldots, K$ — соответствующие значения гистограммы, где K — число дискретных приращений амплитуды a. Тогда n-й момент μ от a относительно его среднего значения определяется выражением

$$\mu_n(a) = \sum_{i=1}^{K} (a_i - m)^n p(a_i), \qquad (8.3\text{-}1)$$

где

$$m = \sum_{i=1}^{K} a_i p(a_i). \qquad (8.3\text{-}2)$$

Величина m является средним значением a, а μ_2 — его дисперсией. Для установления различия между сигнатурами явно непохожих форм границы обычно требуется только несколько первых моментов.

Аппроксимация многоугольниками. Дискретную границу с произвольной точностью можно аппроксимировать многоугольниками. Для замкнутой кривой аппроксимация является точной, когда число сегментов в многоугольнике равно числу точек границы, так что каждая пара соседних точек определяет сегмент многоугольника. На практике целью аппроксимации многоугольниками является качественное определение формы границы с помощью минимального числа многоугольных сегментов. Хотя обычно эта проблема нетривиальна и довольно быстро сводится к итеративному поиску, требующему больших временных

затрат, имеется ряд методов аппроксимации многоугольниками, относительная простота которых и требования к обработке данных делают их пригодными для приложений в области технического зрения роботов. В этом разделе рассмотрено несколько таких методов.

Вначале рассмотрим метод, предложенный в работе [269], для определения многоугольников, имеющих минимальный периметр. Эту процедуру лучше всего объяснить на примере. Предположим, что мы заключили границу во множество связанных друг с другом элементов (рис. 8.30, *а*). Это вложение представляется в виде границ, соответствующих внешней и внутренней границам полосы элементов. Мысленно границу объекта можно представить в виде резиновой ленты, расположенной внутри границ. Если теперь предположить, что резиновая лента сжалась и приняла форму, показанную на рис. 8.30, *б*, то эта форма будет соответствовать многоугольнику минимального периметра, удовлетворяющего геометрии полосы элементов. Когда элементы расположены таким образом, что внутри

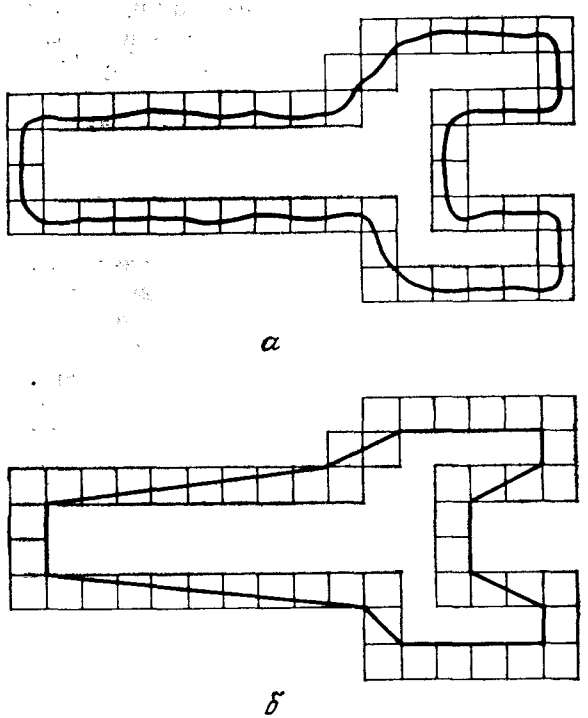


Рис. 8.30. Граница объекта, заключенная в элементы (*а*), и многоугольник минимального периметра (*б*).

каждого элемента находится только одна точка границы, то для каждого элемента максимальная ошибка между исходной границей и ее аппроксимацией резиновой лентой составит $\sqrt{2}\,d$, где d — расстояние между пикселами. Эту ошибку можно в 2 раза уменьшить, расположив пикселы в центрах соответствующих элементов.

В задаче аппроксимации многоугольниками применяются *методы объединения*, основанные на ошибке или других критериях. Один из подходов состоит в соединении точек границы линией по методу наименьших квадратов. Линия проводится до тех пор, пока ошибка аппроксимации не превысит ранее заданный порог. Когда порог превышается, параметры линии заносятся в память, ошибка полагается равной нулю и процедура повторяется; новые точки границы соединяются до тех пор, пока ошибка снова не превысит порог. В конце процедуры образуются вершины многоугольника в результате пересечения соседних линий. Одна из основных трудностей, связанная с этим подходом, состоит в том, что эти вершины обычно не соответствуют изгибам границы (таким, как углы), поскольку новая линия начинается только тогда, когда ошибка превысит порог. Если, например, длинная прямая линия пересекает угол, то числом (зависящим от порога) точек, построенных после пересечения, можно пренебречь ранее, чем будет превышено значение порогового уровня. Однако для устранения этой трудности наряду с методами объединения можно использовать методы разбиения.

Один из методов *разбиения сегментов границы* состоит в последовательном делении сегмента на две части до тех пор, пока удовлетворяется заданный критерий. Например, можно потребовать, чтобы максимальная длина перпендикуляра, проведенного от сегмента границы к линии, соединяющей две крайние точки этого сегмента, не превышала ранее установленного значения порогового уровня. Если это имеет место, наиболее дальняя точка становится вершиной, разделяя, таким образом, исходный сегмент на два подсегмента. Этот метод обладает тем преимуществом, что он адаптирован к наиболее подходящим точкам изгиба. Для замкнутой границы наилучшей начальной парой точек обычно являются точки, наиболее удаленные от границы. Соответствующий пример приведен на рис. 8.31. На рис. 8.31, *а* показана граница объекта, а на рис. 8.31, *б* — разбиение этой границы (сплошная линия) по наиболее удаленным точкам. В точке *c* перпендикуляр из вершины сегмента к линии *ab* имеет наибольшую длину. Аналогично точка *d* наиболее удалена от нижнего сегмента. На рис. 8.31, *в* показан результат применения процедуры разбиения с пороговым уровнем, равным 0,25 длины линии *ab*. Поскольку длина перпендикуляра, опущенного из любой точки новых сегментов границы до соответствующей прямой линии, не превышает выбранной величины порогового

уровня, то результатом работы процедуры является многоугольник, приведенный на рис. 8.31, *г*.

Отметим, что имеется большое число работ, посвященных комбинации методов объединения и разбиения. Обзор этих методов приведен в работе [232].

Индексы формы. Граница, закодированная с помощью цепного кода, имеет несколько первых разностей, зависящих от начальной точки. *Индекс формы* такой границы, основанный на

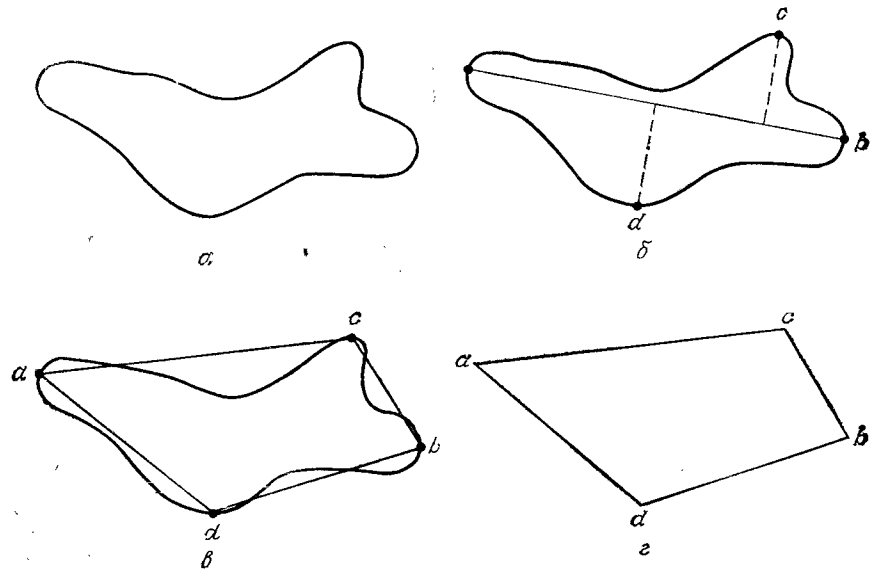


Рис. 8.31. Исходная граница (*а*), граница, разделенная по наиболее удаленным точкам (*б*), соединение вершин отрезками прямых линий (*в*) и получившийся в результате многоугольник (*г*).

коде из четырех направлений (рис. 8.26, *а*), определяется как первая разность наименьшего значения. *Порядок n* индекса формы равен числу цифр кода. Отметим, что n существует даже для замкнутой границы, и его значения ограничивает число всех возможных форм. На рис. 8.32 приведены все возможные формы 4-, 6- и 8-го порядков вместе с их цепными кодами, первыми разностями и индексами формы. Отметим, что при вычислении первых разностей цепные коды рассматривались как замкнутые последовательности. Хотя первая разность цепного кода не зависит от вращения, закодированная граница, вообще говоря, будет зависеть от ориентации решетки кода, показанной на рис. 8.27, *а*. Один из путей нормирования ориентации решетки следующий: *большой осью* границы является отрезок прямой линии, соединяющий две наиболее удаленные друг от

друга точки. *Малой осью* является перпендикуляр, восстановленный к большой оси и имеющий длину, достаточную для построения прямоугольника, описанного вокруг границы. Отношение большой оси к малой называется *эксцентриситетом* границы, а упомянутый выше прямоугольник — *базовым прямоугольником*. Во многих случаях при совмещении решетки цепного кода со сторонами базового прямоугольника получается только один индекс формы. В работе [85] приведен алгоритм построения базового прямоугольника для замкнутой кривой,

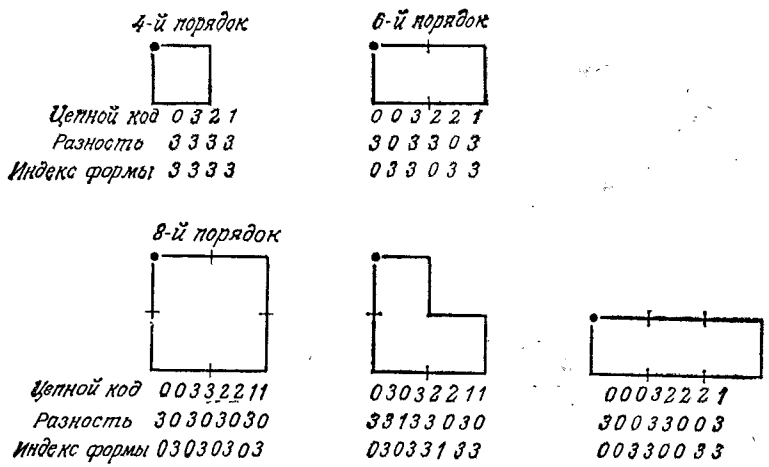


Рис. 8.32. Все формы 4-, 6- и 8-го порядков. Направления определяются согласно рис. 8.26, и точка в левом верхнем углу указывает начальную точку.

закодированной с помощью цепного кода. Задавая желаемый порядок формы границы области, мы получаем прямоугольник порядка n, эксцентриситет которого наилучшим образом соответствует эксцентриситету базового прямоугольника для установления размеров решетки. Например, если $n = 12$, то всеми прямоугольниками 12-го порядка (т. е. периметр которых равен 12) являются следующие: 2×4, 3×3 и 1×5. Если для данной границы эксцентриситет прямоугольника 2×4 наиболее близок к эксцентриситету базового прямоугольника, выбирается решетка 2×4, центр которой совпадает с центром базового прямоугольника, и для получения цепного кода применяется изложенная выше процедура. Индексы формы получаются из первых разностей этого кода с помощью только что приведенного метода. Хотя порядок полученного индекса формы обычно равен n (что определяется способом выбора размеров ячеек решетки) для границ, имеющих выемки, размеры которых сравнимы с размерами ячеек решетки, порядок индекса формы иногда будет больше n. В этом случае выбирается прямоугольник более низкого порядка и вся процедура повторяется, пока индекс формы, полученный в результате ее работы, не достигнет величины n.

Пример. Предположим, что для границы, показанной на рис. 8.33, *а* мы задали порядок $n = 18$. Для получения индекса формы такого порядка воспользуемся только что изложенной процедурой. Сначала мы находим базовый прямоугольник, как

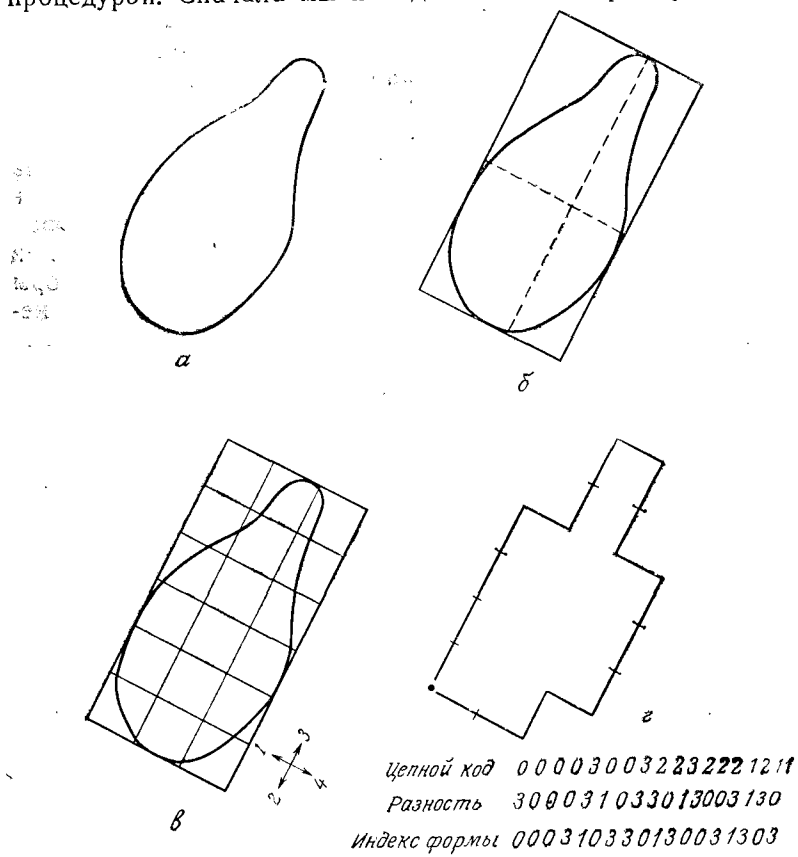


Рис. 8.33. Шаги генерации индекса формы.

показано на рис. 8.33, *б*. Ближайшим прямоугольником 18-го порядка является прямоугольник 3×6, поэтому базовый прямоугольник разбивается так, как показано на рис. 8.33, *в*, причем направления цепного кода совпадают с направлениями решетки. Затем мы определяем цепной код и его первую разность, по которой вычисляется индекс формы (рис. 8.33, *г*).

Дескрипторы Фурье. Дискретное одномерное преобразование Фурье, определяемое уравнением (7.6-4), часто можно исполь-

зовать для описания двумерной границы. Предположим, что граница состоит из M точек. Если рассматривать границу на комплексной плоскости (рис. 8.34), каждая точка (x, y) двумерной границы соответствует комплексному числу $x + jy$. Последовательность точек границы можно представить в виде функции, имеющей преобразование Фурье $F(u)$, $u = 0, 1, 2, \ldots$ $\ldots, M-1$. Если M является целым числом (степенью 2), $F(u)$ можно вычислить с помощью алгоритма быстрого преобразования Фурье (алгоритма БПФ), рассмотренного в разд. 7.6.1. Этот метод выбран потому, что для идентификации существенно разных форм обычно требуется только несколько первых компонент $F(u)$. Например, объекты, приведенные на рис. 8.35, можно различить, используя менее 10 % элементов полного преобразования Фурье их границ. Преобразование Фурье легко нормируется для размера, поворота и начальной точки границы. Для изме-

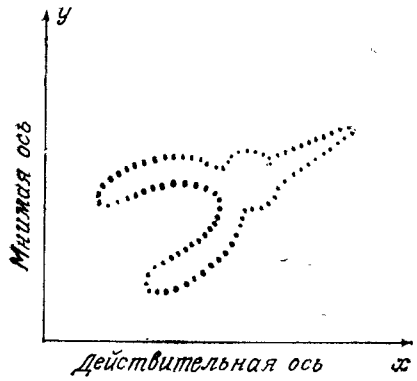


Рис. 8.34. Представление границы области на комплексной плоскости.

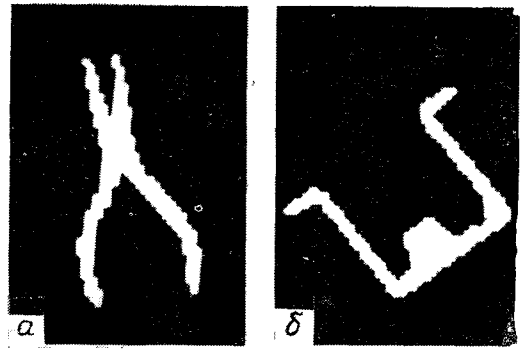


Рис. 8.35. Две формы, которые легко различить с помощью дескрипторов Фурье [234].

нения размера контура достаточно умножить компоненты преобразования Фурье на константу. В силу линейности преобразования Фурье это эквивалентно умножению границы на один и тот же множитель (масштабирование). Поворот на угол θ осуществляется умножением элементов $F(u)$ на $\exp(j\theta)$. Окончательно можно показать, что изменение начальной точки контура в пространственной области соответствует умножению k-й компоненты $F(u)$ на $\exp(jkT)$, где T лежит в интервале $[0, 2\pi]$. Поскольку значения T изменяются в пределах от 0 до 2π, начальная точка обходит весь контур только один раз. Эту информацию можно использовать как основу для нормирования [104].

8.3.2. Дескрипторы области

Область, представляющую интерес, можно описать формой ее границы (разд. 8.3.1) или же путем задания ее характеристик, как показано ниже. Важно отметить, что методы, рассмотренные в обоих разделах, применяются для описания областей.

Некоторые простые дескрипторы. Существующие системы технического зрения основываются на довольно простых дескрипторах области, что делает их более привлекательными с вычислительной точки зрения. Как следует ожидать, применение этих дескрипторов ограничено ситуациями, в которых представляющие интерес объекты различаются настолько, что для их идентификации достаточно несколько основных дескрипторов.

Площадь области определяется как число пикселов, содержащихся в пределах ее границы. Этот дескриптор полезен при сборе информации о взаимном расположении и форме объектов, от которых камера располагается приблизительно на одном и том же расстоянии. Типичным примером может служить распознавание системой технического зрения объектов, движущихся по конвейеру.

Большая и *малая* оси области (разд. 8.3.1) полезны для определения ориентации объекта. Отношение длин этих осей, называемое *эксцентриситетом* области, также является важным дескриптором для описания формы области.

Периметром области называется длина ее границы. Хотя иногда периметр применяется как дескриптор, чаще он используется для определения меры *компактности* области, равной квадрату периметра, деленному на площадь. Отметим, что компактность является безразмерной величиной (и поэтому инвариантна к изменению масштаба) и минимальной для поверхности, имеющей форму диска.

Связной называется область, в которой любая пара точек может быть соединена кривой, полностью лежащей в этой области. Для множества связных областей (некоторые из них имеют отверстия) в качестве дескриптора полезно использовать *число Эйлера*, которое определяется как разность между числом связных областей и числом отверстий. Например, числа Эйлера для букв A и B соответственно равны 0 и -1. Другие дескрипторы области рассматриваются ниже.

Текстура. Во многих случаях идентификацию объектов или областей образа можно осуществить, используя дескрипторы текстуры. Хотя не существует формального определения текстуры, интуитивно этот дескриптор можно рассматривать как описание свойств поверхности (однородность, шероховатость, регулярность). Некоторые примеры приведены на рис. 8.36. Двумя

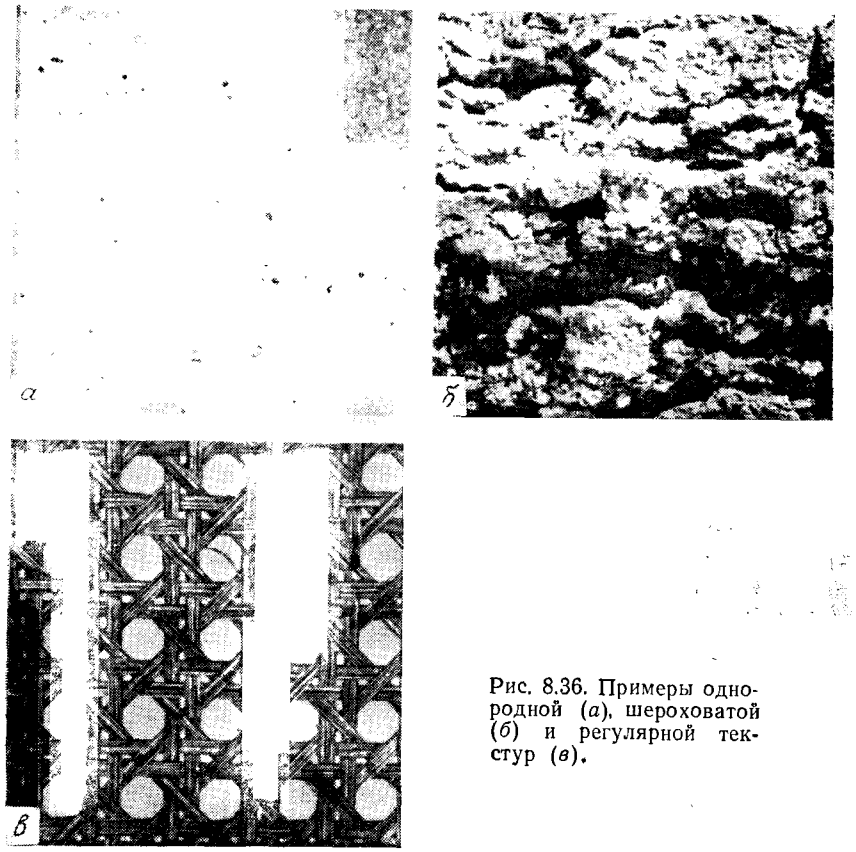


Рис. 8.36. Примеры однородной (*а*), шероховатой (*б*) и регулярной текстур (*в*).

основными подходами для описания текстуры являются статистический и структурный. Статистические методы дают такие характеристики текстуры, как однородность, шероховатость, зернистость и т. д. Структурные методы устанавливают взаимное расположение элементарных частей образа, как, например, описание текстуры, основанной на регулярном расположении параллельных линий.

Одним из наиболее простых методов описания текстуры является использование моментов гистограммы интенсивности

образа или области. Пусть z — случайная величина, обозначающая дискретную интенсивность образа, и $p(z_i)$, $i = 1, 2, \ldots, L$ — соответствующие значения гистограммы, где L — число уровней различной интенсивности. В разд. 8.3.1 показано, что n-й момент z относительно среднего значения определяется формулой

$$\mu_n(z) = \sum_{i=1}^{L} (z_i - m)^n p(z_i), \tag{8.3-3}$$

где m — среднее значение z (т. е. средняя интенсивность образа):

$$m = \sum_{i=1}^{L} z_i p(z_i). \tag{8.3-4}$$

Из уравнения (8.3-3) следует, что $\mu_0 = 1$ и $\mu_1 = 0$. Второй момент, называемый *дисперсией* и обозначаемый $\sigma^2(z)$, особенно важен для описания текстуры. Он является мерой контраста интенсивности и применяется для определения дескрипторов, описывающих однородность поверхности. Например, величина

$$R = 1 - \frac{1}{1 + \sigma^2(z)} \tag{8.3-5}$$

равна 0 для областей постоянной интенсивности [$\sigma^2(z) = 0$, если все z_i имеют одно и то же значение] и приближается к 1 для больших значений $\sigma^2(z)$. Третий момент является мерой асимметрии гистограммы, а четвертый момент — мерой ее относительной ровности. Пятый и шестой моменты не так легко связать с формой гистограммы, но они дают некоторую количественную информацию о виде текстуры.

Однако характеристики текстуры, вычисляемые только по гистограммам, не дают информации о взаимном расположении пикселов. Поэтому для получения такой информации в процессе анализа текстуры надо рассматривать не только распределение интенсивностей, но также положения пикселов с равными или почти равными значениями интенсивности. Пусть P — оператор положения и $\mathbf{A}$ — матрица размером $k \times k$, где элемент a_{ij} обозначает число появлений точек с интенсивностью z_i (в положении, определяемом P) относительно точек с интенсивностью z_j для $1 \leqslant i, j \leqslant k$. Например, рассмотрим образ с тремя уровнями интенсивности $z_1 = 0$, $z_2 = 1$ и $z_3 = 2$:

$$\begin{matrix} 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{matrix}$$

Если мы определим оператор положения как «один пиксел справа и затем один пиксел вниз», тогда мы получим матрицу **A** размерностью 3×3 следующего вида:

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix},$$

где, например, a_{11} — число появлений точки с уровнем интенсивности $z_1 = 0$, расположенной на один элемент по диагонали вниз от пиксела с такой же интенсивностью, а a_{13} — число появления точки с уровнем $z_1 = 0$ относительно точки с интенсивностью $z_3 = 2$. Отметим, что размерность матрицы **A** строго определяется числом различных уровней интенсивности входного образа. Таким образом, для практического применения этого метода обычно требуется разбиение диапазона интенсивности на несколько интервалов с целью сохранения приемлемой размерности матрицы **A**.

Обозначим через n общее число пар точек, удовлетворяющих P (в приведенном выше примере $n = 16$). Если ввести матрицу **C** с элементами $c_{ij} = a_{ij}/n$, тогда c_{ij} представляет собой оценку вероятности того, что пара точек, удовлетворяющих P, будет иметь значения (z_i, z_j). Матрица **C** называется *матрицей вероятности совместного появления уровней интенсивности*, где термин «уровень интенсивности» используется для обозначения интенсивности монохромного пиксела или образа. Поскольку **C** зависит от P, можно обнаружить наличие видов данной текстуры путем выбора соответствующего оператора положения. Так, оператор, используемый в предыдущем примере, чувствителен к полосам частот постоянной интенсивности, падающей под углом —45° (отметим, что в матрице **A** элементом с наибольшим значением является $a_{11} = 4$ отчасти благодаря точкам, имеющим интенсивность, равную нулю и падающую под углом —45°). В более общем случае возникает задача анализа матрицы **C** с целью описания текстуры области, для которой она была вычислена. Ниже приводится набор дескрипторов из [111].

1. Максимальная вероятность:

$$\max_{i,j} (c_{ij}).$$

2. Разностный момент k-го порядка:

$$\sum_i \sum_j (i-j)^k c_{ij}.$$

3. Обратный разностный момент k-го порядка:

$$\frac{\sum_i \sum_j c_{ij}}{(i-j)^k}, \quad i \neq j.$$

4. Энтропия:

$$-\sum_i \sum_j c_{ij} \log c_{ij}.$$

5. Однородность:

$$\sum_i \sum_j c_{ij}^2.$$

Основная идея состоит в описании «содержания» матрицы **C** с помощью этих дескрипторов. Например, первое свойство дает значение самого сильного отклика на оператор P (как в приведенном выше примере). Второй дескриптор имеет относительно низкое значение, когда большие значения элементов матрицы **C** расположены около главной диагонали, поскольку в этом случае разности $i - j$ весьма малы. Действие третьего

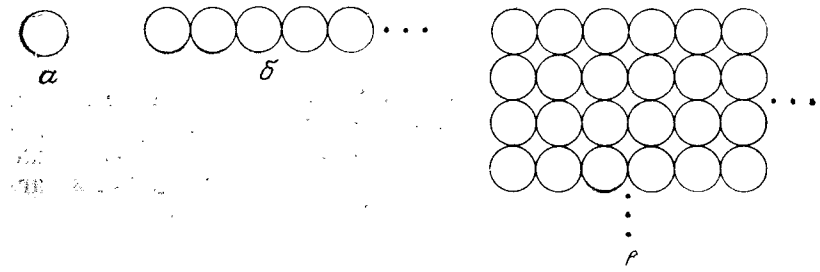


Рис. 8.37. Простейший элемент текстуры (*а*). Модель, созданная в результате применения правила $S \to aS$ (*б*). Двумерная модель текстуры, созданная с помощью этого и других правил (*в*).

дескриптора противоположно второму. Четвертый дескриптор является мерой беспорядка и достигает наибольшего значения, когда все элементы матрицы c_{ij} равны. Наоборот, пятый дескриптор имеет наименьшее значение, когда все c_{ij} равны. Метод, основанный на применении этих дескрипторов, состоит в том, чтобы «обучить» систему распознавать различные текстуры по соответствующим значениям дескрипторов. После того как вычислены дескрипторы области с неизвестной текстурой, в качестве искомой текстуры из памяти системы выбирается та текстура, дескрипторы которой наиболее близки к вычисленным. Более подробно этот метод рассматривается в разд. 8.4.

Методы, рассмотренные выше, являются статистическими. Как уже говорилось в начале этого раздела, ко второй главной категории методов описания текстуры относятся структурные методы. Предположим, что мы имеем правило вида $S \to aS$, которое означает, что символ S преобразуется в aS (например, три применения этого правила дадут цепочку $aaaS$). Если a является кругом (рис. 8.37, *а*), то, присвоив значение «круги

вправо» цепочке вида $aaa\ldots$ с помощью правила $S \to aS$, станет возможным генерировать модель текстуры, показанной на рис. 8.37, *б*.

Предположим, что к этой схеме мы добавили несколько новых правил: $S \to bA$, $A \to cA$, $A \to c$, $A \to bS$, $S \to a$, где b означает «круг вниз», а c — «круг влево». Теперь мы можем создать цепочку вида $aaabccbaa$, которая соответствует матрице из кругов размерностью 3×3. Различные модели текстур, одна из которых показана на рис. 8.37, *в*, можно легко создать аналогичным образом (однако с помощью этих правил можно создавать не только прямоугольные структуры).

Таким образом, для создания сложных моделей текстуры можно использовать «простейший элемент текстуры» (примитив) и набор правил, ограничивающих число возможных взаимных расположений этих элементов. Эти понятия лежат в основе структурного метода создания и распознавания структуры образа, который подробно рассмотрен в разд. 8.5.

Скелет области. Важным подходом для описания вида структуры плоской области является ее представление в виде графа. Во многих случаях для этого определяется схема (*скелет*) области с помощью так называемых *прореживающих* (или же *сокращающих*) алгоритмов. Прореживающие процедуры играют основную роль в широком диапазоне задач компьютерного зрения — от автоматической проверки печатных плат до подсчета асбестовых волокон в воздушных фильтрах. Скелет области можно определить через *преобразование средних осей* (ПСО), предложенное в работе [24]. ПСО области R с границей B определяется следующим образом. Для каждой точки p из R мы определяем ближайшую к ней точку, лежащую на B. Если p имеет больше одной такой точки, тогда о ней говорится, что она располагается на *средней оси* (скелете) области R. Важно отметить, что понятие «ближайшая точка» зависит от определения расстояния (разд. 7.5.3), и поэтому на результаты операции ПСО будет влиять выбор метрики. Некоторые примеры применения евклидовой метрики даны на рис. 8.38.

Хотя ПСО дает довольно удовлетворительный скелет области, его прямое применение затруднительно с вычислительной точки зрения, поскольку требуется определение расстояния между каждой точкой области и границы. Был предложен ряд алгоритмов построения средних осей, обладающих большей вычислительной эффективностью. Обычно это алгоритмы прореживания, которые итеративно устраняют из рассмотрения точки контура области так, чтобы выполнялись следующие ограничения:

1) не устранять крайние точки;
2) не приводить к нарушению связности;
3) не вызывать чрезмерного размывания области.

Хотя были сделаны некоторые шаги в применении скелетов для задач распознавания черно-белых образов с несколькими уровнями интенсивности [70], [256], этот тип представления обычно связывается с бинарными данными.

Ниже мы рассмотрим алгоритм, разработанный Накаши и Шингалом [204]. Эта процедура быстрая, проста в реализации и, как показано ниже, во многих случаях дает более хорошие результаты, чем другие алгоритмы прореживания. Сначала введем несколько определений. Используя бинарные данные,

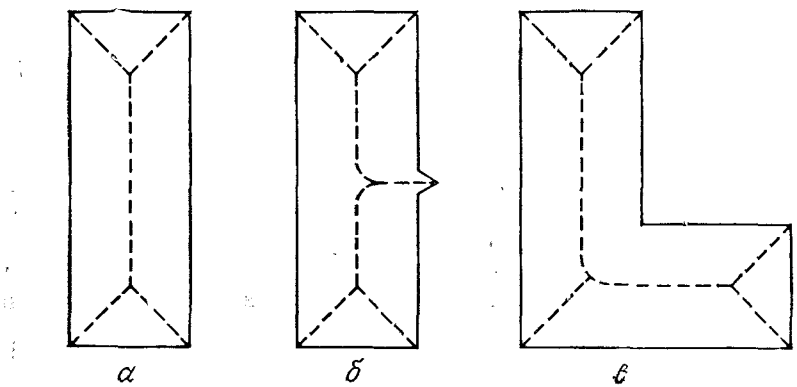


Рис. 8.38. Средние оси трех простых областей.

обозначим точки области единицами, а точки фона — нулями. Назовем их соответственно *темными* и *светлыми* точками. *Точкой контура* называется темная точка, которая имеет в своей окрестности, состоящей из четырех точек, по меньшей мере одну светлую. *Конечной точкой* называется темная точка, которая в своей окрестности, состоящей из восьми точек, имеет одну и только одну темную точку. *Точкой разрыва* называется темная точка, устранение которой привело бы к нарушению связности области. Для всех алгоритмов прореживания шум и другие ложные изменения границы могут приводить к значительным отклонениям определяемого скелета (наиболее явно это показано на рис. 8.38, *б*). Поэтому предполагается, что границы всех областей перед процессом прореживания были предварительно сглажены, например, с помощью процедуры, рассмотренной в разд. 7.6.2.

В соответствии с приведенным на рис. 8.39 расположением точек окрестности точка контура p, используемая в алгоритмах прореживания, может быть следующего вида: 1) *левая точка контура*, у которой левая соседняя точка n_4 светлая; 2) *правая точка контура*, у которой соседняя точка n_0 светлая; 3) *верхняя точка контура*, у которой соседняя точка n_2 свет-

лая; 4) *нижняя точка контура*, у которой соседняя точка n_6 светлая. Иногда точку можно одновременно отнести сразу к нескольким из этих видов. Например, темная точка p, у которой n_0 и n_4 светлые, будет одновременно правой и левой точками контура. Затем сначала рассматривается процесс идентификации левых точек контура, которые должны быть устранены, и уже потом процедура распространяется на другие виды точек.

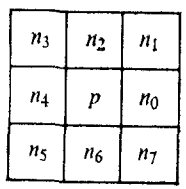


Рис. 8.39. Обозначение для соседних с точкой p пикселов, которое используется в алгоритме прореживания [204].

Точка контура p помечается, если она не является конечной точкой или точкой разрыва, а также если ее устранение не вызовет чрезмерного размытия (как показано ниже). Проверка этих условий осуществляется путем сравнения окрестности точки p, состоящей из восьми точек с окнами, приведенными

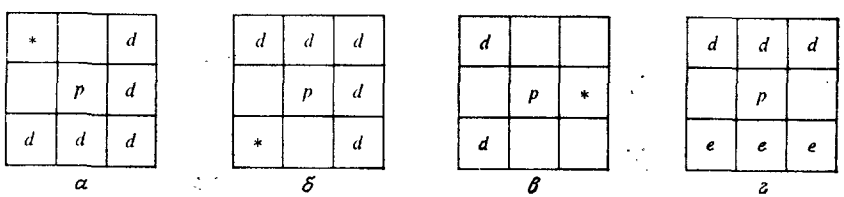


Рис. 8.40. Разметка окрестности из восьми соседних точек темной точки p. Для приведенных окон точка p не помечается. Звездочкой обозначена темная точка, а d и e могут быть либо темными, либо светлыми [204].

на рис. 8.40, где p и звездочка являются темными точками, а d и e могут быть либо темными, либо светлыми точками. Если окрестность точки p соответствует окнам, показанным на рис. 8.40, *а*—*в*, возможны два случая:

1. Если все точки d светлые, тогда p является конечной точкой.

2. Если хотя бы одна точка d темная, тогда p является точкой разрыва. В любом из этих случаев точка p не должна помечаться.

Анализ окна, представленного на рис. 8.40, *г*, более сложный. Если хотя бы одна из точек d и e темная, тогда точка p является точкой разрыва и не должна помечаться. Однако необходимо рассмотреть другое расположение точек. Предположим, что все точки d светлые, а точки e могут быть либо темными, либо светлыми. Это условие дает восемь вариантов, показанных на рис. 8.41. Для конфигураций *а* — *в* точка p конечная, а для конфигурации *г* она является точкой разрыва. Если бы p была устранена в конфигурациях *д* и *е*, то на примере легко показывается, что это устранение вызвало бы чрезмерное размытие наклонных областей шириной 2. На конфигурации *ж* точка p относится к так называемой *шпоре*, обычно появляющейся вследствие короткого ответвления или выступа области. Поскольку предполагается, что граница области вначале сглажена, появление шпоры во время процесса прореживания рассматривается как важный элемент описания формы области,

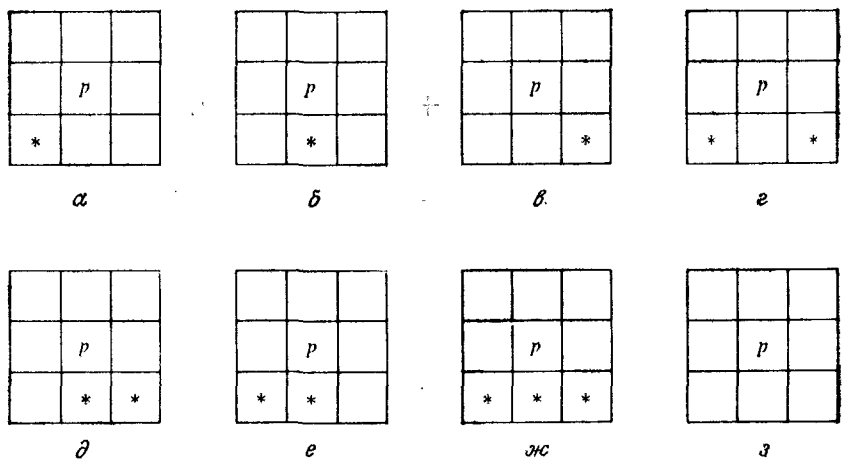


Рис. 8.41. Возможные конфигурации, когда на рис. 8.40 точка d светлая, а e может быть темной, * или светлой [204].

и поэтому точка p не должна быть устранена. Наконец, если все изолированные точки были устранены ранее, появление во время процесса прореживания конфигурации *з* означает, что область свелась к единственной точке; ее устранение означало бы ликвидацию последней остающейся части области. Аналогичные рассуждения применяются, когда точки d и e меняются местами или же когда они могут быть как темными, так и светлыми. Главным является то, что каждая левая точка контура p, имеющая окрестность из восьми точек, соответствующую окнам, показанным на рис. 8.40, не должна быть устранена.

Проверка окрестности точки p на соответствие четырем окнам (рис. 8.40) осуществляется по формуле булевой алгебры логики:

$$B_4 = n_0 \cdot (n_1 + n_2 + n_6 + n_7) \cdot (n_2 + \bar{n}_3) \cdot (\bar{n}_5 + n_6), \qquad (8.3\text{-}6)$$

где индекс у B означает, что n_4 является светлой (т. е. p — левая точка контура), точка — логическое И, плюс — логическое

ИЛИ, минус — логическое отрицание, а n_i, $i = 1, 2, 3, 4$ определены на рис. 8.39. Ранее непомеченным темным точкам присваивается значение 1 (ИСТИНА), светлым, а также помеченным точкам — значение 0 (ЛОЖЬ). Тогда, если $B_4 = 1$ (ИСТИНА), мы помечаем точку p. В противном случае точка p остается непомеченной. Легко показать, что эти условия, наложенные на B_4, одновременно выполняются для всех четырех окон, приведенных на рис. 8.40.

Подобные выражения получаются для правых точек контура

$$B_0 = n_4 \cdot (n_2 + n_3 + n_5 + n_6) \cdot (n_6 + \bar{n}_7) \cdot (\bar{n}_1 + n_2), \qquad (8.3\text{-}7)$$

для верхних точек контура

$$B_2 = n_6 \cdot (n_0 + n_4 + n_5 + n_7) \cdot (n_0 + \bar{n}_1) \cdot (\bar{n}_3 + n_4) \qquad (8.3\text{-}8)$$

и для нижних точек контура

$$B_6 = n_2 \cdot (n_0 + n_1 + n_3 + n_4) \cdot (n_4 + \bar{n}_5) \cdot (n_0 + \bar{n}_7). \qquad (8.3\text{-}9)$$

Используя эти уравнения, алгоритмы прореживания работают итеративным образом, два раза сканируя данные. Сканирование может осуществляться либо вдоль строк, либо вдоль столбцов образа, но этот выбор обычно влияет на конечный результат. В первом случае для пометки левых и правых точек контура применяются B_4 и B_0; при втором сканировании для пометки верхних и нижних точек контура применяются B_2 и B_6. Если после второго сканирования не появилось новых помеченных точек контура, алгоритм прекращает работу, а искомый скелет состоит из непомеченных точек; в противном случае процедура повторяется. Отметим, что предварительно помеченным темным точкам при вычислении булевых выражений присваивается значение 0. Альтернативная процедура состоит в присвоении любой помеченной точке значения 0 во время работы алгоритма, и следовательно, обработка точек фона и создание скелета происходят в конце работы процедуры. Этот метод более прост в реализации за счет отбрасывания всех других точек области.

Пример. На рис. 8.42, *а* приведена бинарная область, а на рис. 8.42, *б* — скелет, полученный с помощью описанного выше алгоритма. Для сравнения на рис. 8.42, *в* приведен скелет, полученный на основе тех же данных с помощью хорошо известного алгоритма прореживания [233]. Различие в результатах очевидно.

Инварианты моментов. В разд. 8.3.1 говорилось, что для описания границы области можно использовать дескрипторы Фурье, инвариантные относительно перемещения, поворота и изменения масштаба. В том случае, когда область задается

множеством точек, она может быть описана моментами, инвариантными относительно этих действий.

Обозначим через $f(x, y)$ интенсивность в точке области (x, y). Тогда для данной области *момент* $(p + q)$ порядка определяется выражением

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y), \qquad (8.3\text{-}10)$$

где суммирование проводится по всем пространственным координатам (x, y) точек области. *Центральный момент* порядка

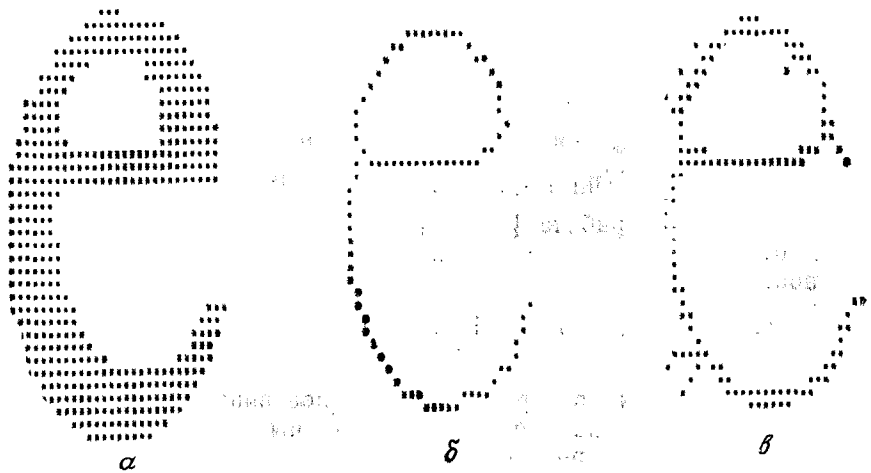


Рис. 8.42. Бинарная область (*а*). Скелет, полученный с помощью одного алгоритма прореживания (*б*), и скелет, полученный с помощью другого алгоритма прореживания (*в*) [204].

$(p + q)$ определяется следующей формулой:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y), \qquad (8.3\text{-}11)$$

где

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \qquad (8.3\text{-}12)$$

Нормированный центральный момент порядка $(p + q)$ определяется выражением

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\nu}}, \qquad (8.3\text{-}13)$$

где

$$\nu = \frac{p+q}{2} + 1 \quad \text{для} \quad (p+q) = 2, 3, \ldots . \qquad (8.3\text{-}14)$$

Используя только нормированные центральные моменты 2- и 3-го порядков, можно вывести следующий набор *инвариантов моментов*:

$$\phi_1 = \eta_{20} + \eta_{02}, \tag{8.3-15}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2, \tag{8.3-16}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2, \tag{8.3-17}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2, \tag{8.3-18}$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2], \tag{8.3-19}$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}), \tag{8.3-20}$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \tag{8.3-21}$$

Как показано в работе [128], этот набор моментов инвариантен относительно перемещения, поворота и изменения масштаба.

8.4. СЕГМЕНТАЦИЯ И ОПИСАНИЕ ТРЕХМЕРНЫХ СТРУКТУР

В предыдущих двух разделах основное внимание уделялось методам сегментации и описания двумерных структур. В этом разделе мы рассмотрим эти задачи применительно к трехмерным данным сцены.

Как уже говорилось в разд. 7.1, по существу зрение является трехмерной проблемой, поэтому в основе разработки многофункциональных систем технического зрения, пригодных для работы в различных средах, лежит процесс обработки информации о трехмерных сценах. Хотя исследования в этой области имеют более чем 10-летнюю историю, такие факторы, как стоимость, скорость и сложность, тормозят внедрение обработки трехмерной зрительной информации в промышленных приложениях.

Возможны три основные формы представления информации о трехмерной сцене. Если применяются датчики, измеряющие расстояние, то мы получаем координаты (x, y, z) точек поверхностей объектов. Применение устройств, создающих стереоизображение, дает трехмерные координаты, а также информацию об освещенности в каждой точке. В этом случае каждая точка представляется функцией $f(x, y, z)$, где значения последней в точке с координатами (x, y, z) дают значения интенсивности в этой точке (для обозначения точки в трехмерном пространстве и ее интенсивности часто применяется термин вокселей). Наконец, можно установить трехмерные связи на основе одного двумерного образа сцены, т. е. можно выводить связи между объектами, такие, как «над», «за», «перед». Поскольку точное трехмерное расположение точек сцены обычно не может быть вычислено на основе одного изображения, связи, полученные с помощью этого вида анализа, иногда относятся к так называемой 2,5-мерной информации.

8.4.1. Описание трехмерной сцены плоскими участками

Один из наиболее простых подходов для сегментации и описания трехмерных структур с помощью координат точек (x, y, z) состоит в разбиении сцены на небольшие плоские

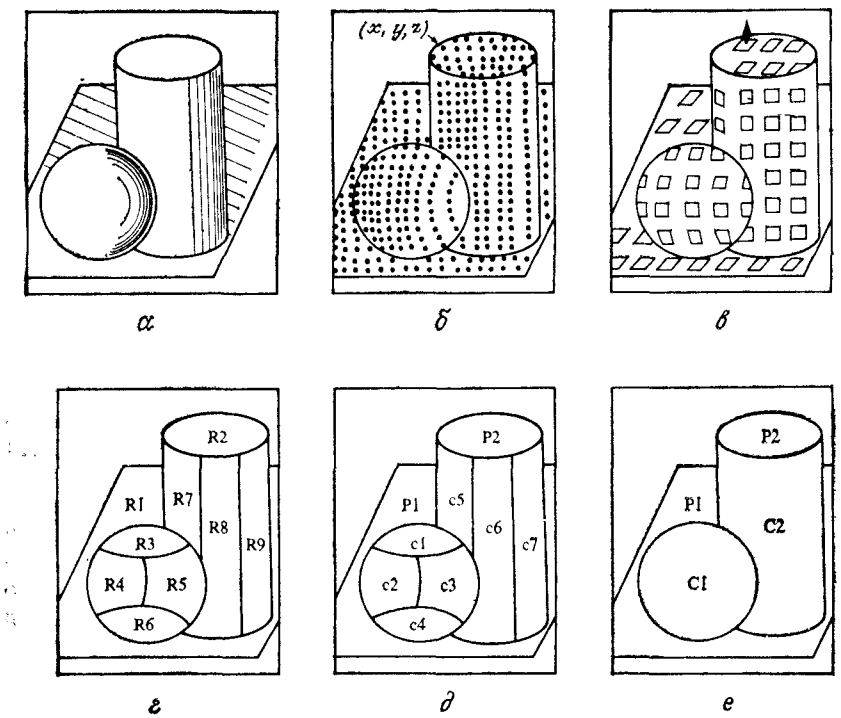


Рис. 8.43. Описание трехмерной поверхности, основанное на представлении в виде плоских участков [265].

«участки» с последующим их объединением в более крупные элементы поверхности в соответствии с некоторым критерием. Этот метод особенно удобен для идентификации многогранных объектов, поверхности которых достаточно гладкие относительно разрешающей способности.

Рассмотрим основные понятия, лежащие в основе этого метода, на примере рис. 8.43. На рис. 8.43, *а* приведена простая сцена, а на рис. 8.43, *б* — множество соответствующих трехмерных точек. Эти точки можно объединить в небольшие элементы поверхности, если, например, разбить трехмерное пространство на ячейки, в которых содержатся эти точки. Затем группе точек каждого элемента ставится в соответствие плоскость, для которой вычисляется единичный вектор (нормаль), проходящий через центр этого элемента.

Плоские участки получаются в результате пересечения плоскостей и сторон элементов, а их ориентация в пространстве задается единичным вектором (рис. 8.43, *в*.) Все плоские участки, направления которых в пределах задаваемого порога примерно одинаковы, объединяются в элементарные области (R) (рис. 8.43, *г*). Затем эти области классифицируются как плоские (P), кривые (C) или неопределенные (U) в зависимости от направления плоских участков каждой области (например, плоские участки плоской поверхности будут иметь одно и то же направление). Этот тип классификации областей показан на рис. 8.43, *д*. Окончательно (и это самый трудный шаг) основные поверхности получаются в результате объединения смежных областей одного класса (рис. 8.43, *е*). Отметим, что в результате работы этой процедуры сцена разделена на различные поверхности, а каждой поверхности присвоен дескриптор (т. е. поверхность является кривой или плоской).

8.4.2. Применение градиента

Когда сцена задана вокселами, ее можно описать плоскими участками (по аналогии с методом, рассмотренным в разд. 8.4.1) с помощью трехмерного градиента. В этом случае дескрипторы поверхности также получаются в результате объединения этих плоских участков. Как уже говорилось в разд. 7.6.4, вектор градиента указывает направление максимальной скорости изменения функции, а его величина соответствует величине этого изменения. Эти понятия применимы для трехмерного случая и также могут быть использованы для разбиения на сегменты трехмерных структур тем же способом, который применялся для двумерных данных.

Пусть задана функция $f(x, y, z)$, тогда ее градиент в точке (x, y, z)

$$\mathbf{G}[f(x, y, z)] = \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}. \qquad (8.4\text{-}1)$$

Величина градиента **G** определяется следующим образом:

$$G[f(x, y, z)] = (G_x^2 + G_y^2 + G_z^2)^{1/2}. \qquad (8.4\text{-}2)$$

Используя уравнение (7.6-39) для упрощения вычислений, G часто аппроксимируется абсолютными значениями:

$$G[f(x, y, z)] \approx |G_x| + |G_y| + |G_z|.$$

Применение трехмерного градиента основывается на использовании операторов, аналогичных операторам, рассмотренным в разд. 7.6.4. На рис. 8.44 приведен оператор размерностью

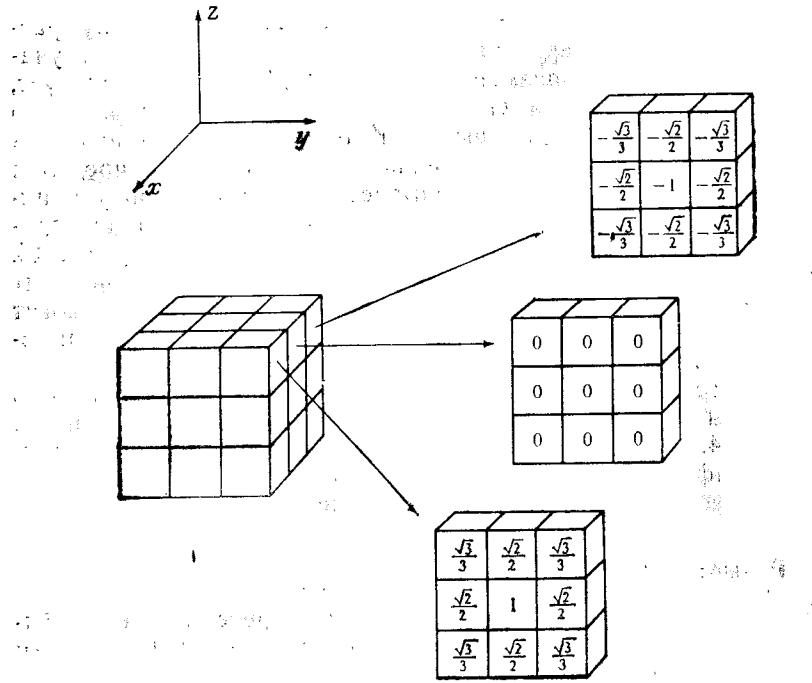


Рис. 8.44. Оператор размерностью $3 \times 3 \times 3$ для вычисления составляющей градиента G_x [324].

$3 \times 3 \times 3$, предложенный в работе [324] для вычисления компоненты градиента G_x. Такой же оператор, ориентированный вдоль оси y, используется для вычислений G_y, и ориентированный вдоль оси z — для вычисления G_z. Основное свойство этих операторов состоит в том, что они дают наилучший (по методу наименьших квадратов) плоский контур между двумя областями различной интенсивности в трехмерной окрестности.

Центр каждого оператора перемещается от воксела к вокселу, и операторы применяются так же, как их двумерные ана-

логи (разд. 7.6.4), т. е. результатом действия операторов в любой точке (x, y, z) являются G_x, G_y и G_z, которые затем подставляются в уравнение (8.4-1) для получения вектора градиента и в уравнение (8.4-2) или (8.4-3) для получения его величины. Отметим, что оператор, приведенный на рис. 8.44, дает нулевое значение градиента для области постоянной интенсивности размерностью $3 \times 3 \times 3$.

Для разбиения сцены на плоские участки, аналогичные тем, которые рассматривались в предыдущем разделе, имеется простая процедура, использующая метод градиента. Нетрудно показать, что вектор градиента к плоскости $ax + by + cz = 0$ имеет компоненты $G_x = a$, $G_y = b$ и $G_z = c$. Поскольку рассмотренные выше операторы дают оптимальный плоский участок в окрестности размерностью $3 \times 3 \times 3$, отсюда следует, что компоненты вектора **G** определяют направления плоского участка в каждой окрестности, в то время как величина **G** дает указание на резкое изменение интенсивности в пределах плоского участка, т. е. она указывает на наличие контура интенсивности в пределах этого участка. Соответствующий пример применения градиентного оператора приведен на рис. 8.45. Поскольку поверхность каждого планарного участка проходит через центр воксела, границы этих участков не всегда могут совпадать. На рис. 8.45 совпадающие участки показаны широкими однородными областями.

После того как получены плоские участки, они могут быть объединены в основные поверхности методом, рассмотренным в разд. 8.4.1. Отметим, что теперь мы располагаем дополнительной информацией об интенсивности и разрывах интенсивности, облегчающей процесс объединения и описания.

8.4.3. Разметка линий и соединений

Итак, контуры в трехмерной сцене определяются разрывами в данных о координатах и/или интенсивности. После того как был определен набор поверхностей и контуров, располагающихся между ними, окончательное описание сцены может быть получено путем разметки линий, которые соответствуют контурам, и соединений, которые эти контуры образуют.

Основные виды линий приведены на рис. 8.46. Выпуклая линия (помеченная +) образуется в результате пересечения двух поверхностей выпуклого тела (например, линия, образованная в результате пересечения двух сторон куба). Вогнутая линия (помеченная —) образуется в результате пересечения двух поверхностей, принадлежащих двум различным телам (например, пересечение стороны куба с полом). Скрытые линии (помеченные стрелками) представляют собой контуры невидимых поверхностей. Поверхности, закрывающие другие части объекта, располагаются справа от линии, если смотреть в направлении стрелок, а невидимые поверхности располагаются слева. После того как линии сцены размечены, их соединения дают ключ к пониманию природы трехмерных объектов сцены. Физические ограничения допускают лишь несколько возможных комбинаций меток линий в соединении. Например, сцена в виде многогранника не имеет линий, метки которых могут меняться между вершинами. Нарушение этого правила приводит к объектам, не имеющим физического смысла (рис. 8.47).

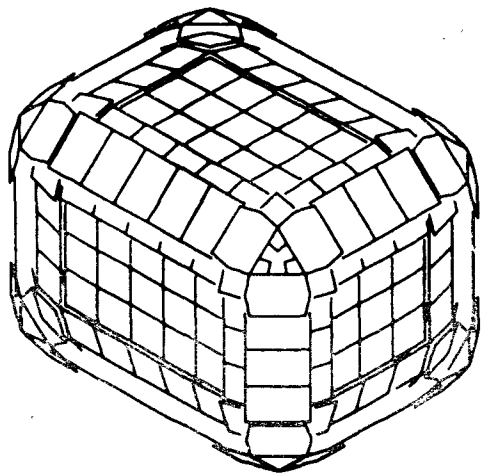


Рис. 8.45. Аппроксимация куба плоскими участками с использованием градиента[324]

В основе метода анализа соединений лежит образование словаря из допустимых типов соединений. Например, легко показать, что словарь соединений, приведенный на рис. 8.48, содержит все значимые помеченные вершины трехгранных тел

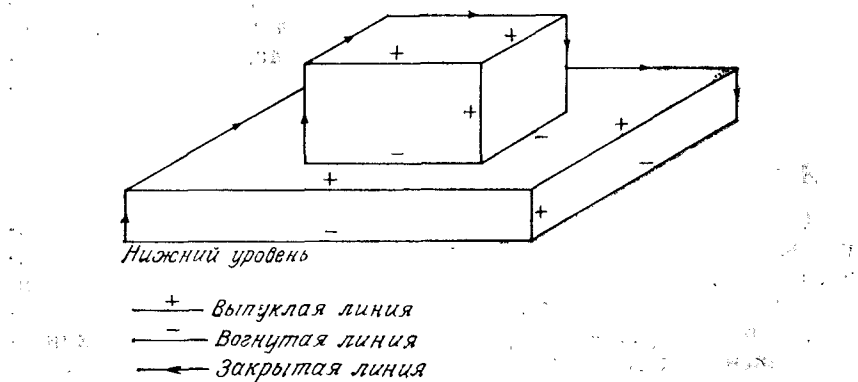


Рис. 8.46. Три основные разметки линий.

(т. е. тела, у которых три плоские грани сходятся к одной вершине). После того как произведена классификация соединений в соответствии со словарем, возникает задача объединения различных поверхностей в объекты. Как правило, она решается с помощью набора эвристических правил, построенных для

интерпретации помеченных линий и последовательностей из соседних соединений. Основные понятия, лежащие в основе этого метода, могут быть проиллюстрированы рис. 8.49. Из рис. 8.49, *б* следует, что капля полностью составлена из скрытой невидимой границы, за исключением короткой вогнутой линии, указывающей место, где она касается базовой детали. Поскольку перед каплей ничего нет, она может быть выделена из сцены. Отметим также, что имеется вершина типа (10) из словаря, приведенного на рис. 8.48. Очевидно, что для трехгранных объектов три поверхности, сходящихся к этой вершине, образуют кубическую поверхность. Подобные рассуждения применимы к базовой детали, после того как устранены кубические поверхности. После устранения базовой детали на фоне остается единственный объект, на котором завершается декомпозиция сцены.

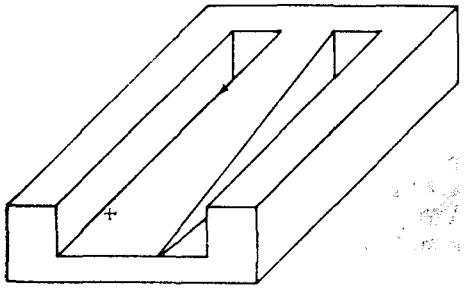


Рис. 8.47. Объект, не имеющий физического смысла. Отметим, что одна из линий меняет метку из-за того, что загорожена выпуклость между двумя вершинами.

Хотя этот пример и дает достаточно полное представление о применении анализа линий и соединений для описания трехмерных объектов сцены, тем не менее создание алгоритма для обработки более сложных сцен является далеко не тривиальной задачей. Работы, посвященные исследованиям в этой области, приводятся в конце главы.

8.4.4. Обобщенные конусы

Обобщенным конусом (или *цилиндром*) называется поверхность, получаемая в результате перемещения плоского поперечного сечения вдоль произвольной пространственной кривой (*хребта*) под постоянным к ней углом, причем поперечное сечение преобразуется по правилу *заметания объема*. В техническом зрении метод обобщенных конусов независимо от других методов позволяет создавать образы трехмерных структур, что полезно при моделировании и для проверки соответствия построенных моделей исходным данным.

Пример работы процедуры генерации обобщенных конусов приведен на рис. 8.50. На рис. 8.50, *а* поперечным сечением является кольцо, хребтом — прямая линия, а правило заметания объема состоит в перемещении поперечного сечения перпендикулярно хребту, в то время как его диаметр остается постоян-

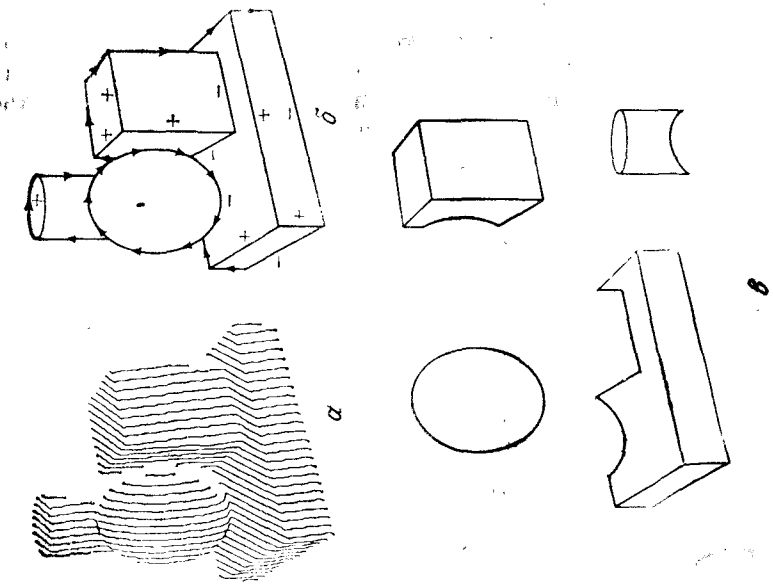


Рис. 8.49. Сцена (*а*), помеченные линии (*б*) и декомпозиция на основе анализа линий и соединений (*в*) [265].

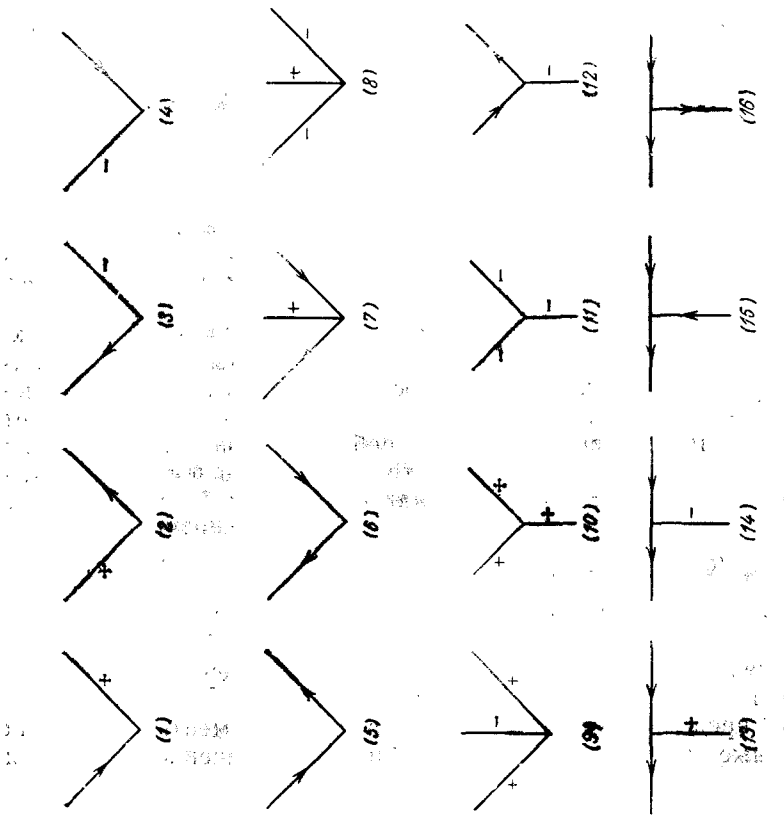


Рис. 8.48. Словарь соединений для трехгранных тел.

ным. В результате получается полый цилиндр. На рис. 8.50, *б* приведена та же ситуация, за исключением того, что сначала правило заметания объема состоит в том, чтобы поддерживать диаметр поперечного сечения постоянным, а затем после прохождения поперечным сечением средней точки хребта диаметр линейно возрастает.

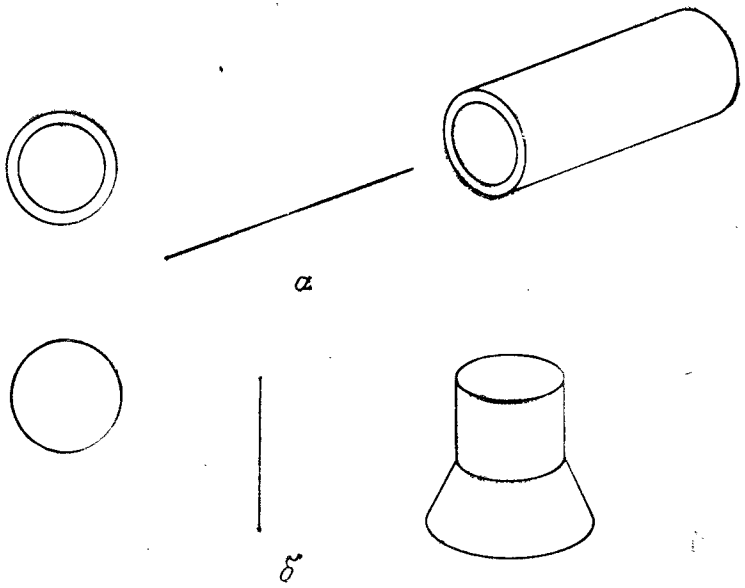


Рис. 8.50. Поперечные сечения и соответствующие им обобщенные конусы.
а — поперечное сечение остается постоянным во время заметания объема; *б* — диаметр поперечного сечения линейно увеличивается после прохождения средней точки хребта.

Когда устанавливается соответствие между набором трехмерных точек и набором известных обобщенных конусов, сначала определяется центральная ось точек, а затем набор поперечных сечений, который наилучшим образом соответствует данным при перемещении вдоль хребта. Обычно для получения удовлетворительного результата требуется большое число испытаний, особенно при неполных данных.

8.5. РАСПОЗНАВАНИЕ

Распознаванием называется процесс разметки, т. е. алгоритмы распознавания идентифицируют каждый объект сцены и присваивают ему метки (гаечный ключ, перемычка, болт). Обычно в большинстве промышленных систем технического зрения предполагается, что объекты сцены сегментированы как отдельные элементы. Другое общее ограничение относится

к расположению устройств сбора информации относительно исследуемой сцены (обычно они располагаются перпендикулярно рабочей поверхности). Это приводит к уменьшению отклонений в характеристиках формы, а также упрощает процесс сегментации и описания в результате уменьшения вероятности загораживания одних объектов другими. Управление отклонениями в ориентации объекта производится путем выбора дескрипторов, инвариантных к вращению, или путем использования главных осей объекта для ориентирования его в предварительно определенном направлении.

Современные методы распознавания делятся на две основные категории: *теоретические* и *структурные* методы. Как показано ниже, теоретические методы основываются на количественном описании (например, статистическая текстура), а в основе структурных методов лежат символические описания и их связи (например, последовательности направлений в границе, закодированной с помощью цепного кода). За несколькими исключениями, рассмотренные в этом разделе процедуры используются для распознавания образов двумерных объектов.

8.5.1. Теоретические методы

Теоретические методы распознавания объектов основываются на применении *решающих функций* (*дескриминанта*). Пусть $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ — *вектор модели объекта* (*модельный вектор*) с действительными компонентами, где x_i — i-й дескриптор данного объекта (например, площадь, средняя интенсивность, длина периметра). Тогда, если заданы M классов объектов $\omega_1, \omega_2, \ldots, \omega_M$, основной задачей распознавания является определение M решающих функций $d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots, d_M(\mathbf{x})$, таких, что для любого модельного вектора $\mathbf{x}^*$, принадлежащего классу ω_i, выполняются следующие неравенства:

$$d_i(\mathbf{x}^*) > d_j(\mathbf{x}^*), \quad j = 1, 2, \ldots, M; \quad j \neq i. \tag{8.5-1}$$

Другими словами, неизвестный объект, представленный вектором $\mathbf{x}^*$, относится к i-му классу (т. е. распознается), если при подстановке $\mathbf{x}^*$ во все решающие функции $d_i(\mathbf{x}^*)$ имеет наибольшее значение.

В промышленных системах технического зрения решающие функции используются главным образом для установления *соответствия* (*подбора*). Предположим, что мы представляем каждый класс объектов с помощью *вектора прототипа* (*усредненного вектора*):

$$\mathbf{m}_i = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k, \quad i = 1, 2, \ldots, M, \tag{8.5-2}$$

где $\mathbf{x}_k$ — модельные векторы, о которых известно, что они относятся к классу ω_i. Тогда, чтобы определить, к какому классу объектов относится неизвестный вектор $\mathbf{x}^*$, надо найти его ближайший прототип. Если для определения близости к прототипу

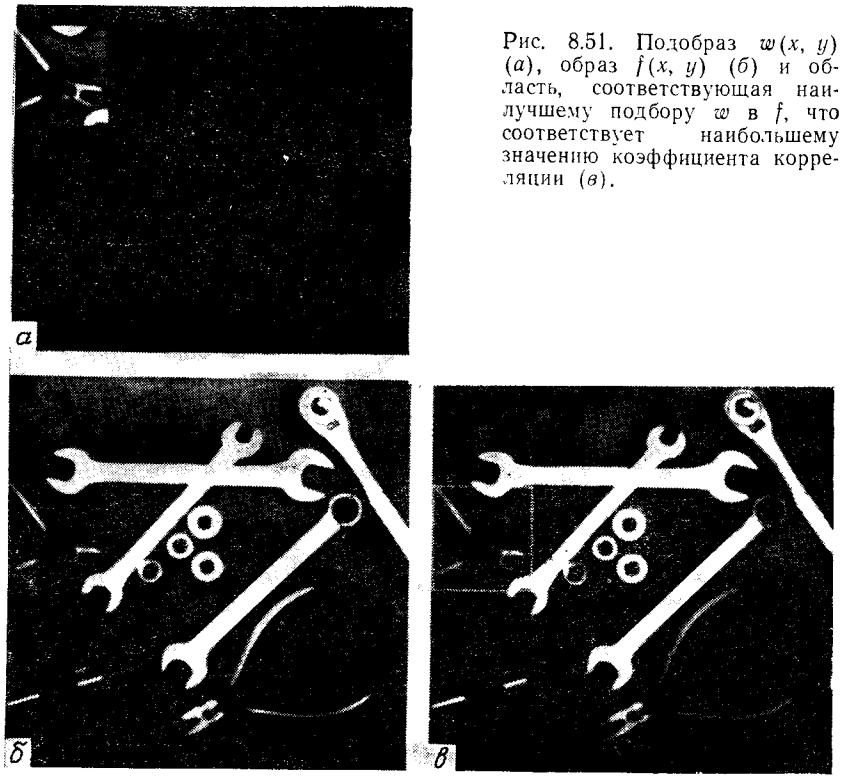


Рис. 8.51. Подобраз $w(x, y)$ (*а*), образ $f(x, y)$ (*б*) и область, соответствующая наилучшему подбору w в f, что соответствует наибольшему значению коэффициента корреляции (*в*).

используется евклидово расстояние, задача сводится к вычислению следующих выражений:

$$D_j(\mathbf{x}^*) = \|\mathbf{x}^* - \mathbf{m}_j\|, \quad j = 1, 2, \ldots, M; \tag{8.5-3}$$

где $\|\mathbf{a}\| = (\mathbf{a}^T\mathbf{a})^{1/2}$ — евклидова норма. В этом случае вектор x^* относится к классу ω_i, если $D_i(x^*)$ является наименьшим расстоянием. Нетрудно показать, что это эквивалентно вычислению значений функций

$$d_j(\mathbf{x}^*) = (\mathbf{x}^*)^T\mathbf{m}_j - 1/2\mathbf{m}_j^T\mathbf{m}_j, \quad j = 1, 2, \ldots, M \tag{8.5-4}$$

и выбору наибольшего значения, что совпадает с определением решающей функции (выражение 8.5-1). Другое применение соответствия состоит, например, в нахождении подобраза $w(x, y)$

на более большом образе $f(x, y)$. Для каждого участка (x, y) образа $f(x, y)$ *коэффициент корреляции* определяется выражением

$$v(x, y) = \frac{\sum_s \sum_t [w(s, t) - m_w][f(s, t) - m_f]}{\left\{\sum_s \sum_t [w(s, t) - m_w]^2 \sum_s \sum_t [f(s, t) - m_f]^2\right\}^{1/2}}, \tag{8.5-5}$$

где предполагается, что $w(s, t)$ центрируется в каждой точке (x, y). Суммирование проводится по всем координатам, общим для обоих областей, m_w — средняя интенсивность w и m_f — средняя интенсивность f в области, совпадающей с w. Отметим, что обычно значение $v(x, y)$ лежащее в диапазоне $[-1, 1]$, сильно изменяется при переходе от одного участка образа к другому, причем значение 1 соответствует наилучшему подбору. Затем для каждого участка (x, y) определяется $v(x, y)$, и для установления наилучшего соответствия w в f выбирается его наибольшее значение. [Процедура движения $w(x, y)$ по $f(x, y)$ аналогична рассмотренной в разд. 7.6.1 (рис. 7.20).]

Качество подбора можно регулировать, принимая в рассмотрение только те коэффициенты корреляции, значения которых превышают некоторое предварительно установленное значение (например, 0,9). Поскольку этот метод заключается в непосредственном сравнении двух областей, он весьма чувствителен к отклонениям ориентации и размера объекта. Отклонения интенсивности нормируются знаменателем в уравнении (8.5-5). Пример подбора с помощью коэффициента корреляции показан на рис. 8.51.

8.5.2. Структурные методы

Методы, рассмотренные в разд. 8.5.1, основываются на количественных моделях, в которых пренебрегают геометрическими связями, присущими форме объекта. В структурных же методах объект описывается с помощью этих связей.

В основе структурных методов распознавания образов лежит декомпозиция объекта на *простейшие элементы* (примитивы) (рис. 8.52). На рис. 8.52, *а* показана простая граница объекта, а на рис. 8.52, *б* — набор простейших элементов определенной длины и направления. Начиная с верхней левой точки в направлении по часовой стрелке, мы кодируем границу простейшими элементами, как показано на рис. 8.52, *в*. Таким образом, граница представляется следующей цепочкой: *aaabcbbbcdddcd*. Таким образом, зная длины и направления простейших элементов, а также порядок их расположения относительно друг друга, мы можем описать структуру объекта. В этом разделе описаны подобные методы, предназначенные

для создания и работы с такими видами структурных описаний моделей объектов.

Подбор индексов формы. Рассмотрим процедуру сравнения двух границ объектов, описанных с помощью индексов формы. Она аналогична процедуре, изложенной в разд. 8.5.1, для векторного представления объектов. В соответствии с разд. 8.3.1 *степенью схожести* k между двумя границами объектов A и B назовем наибольшее значение совпадающих индексов формы,

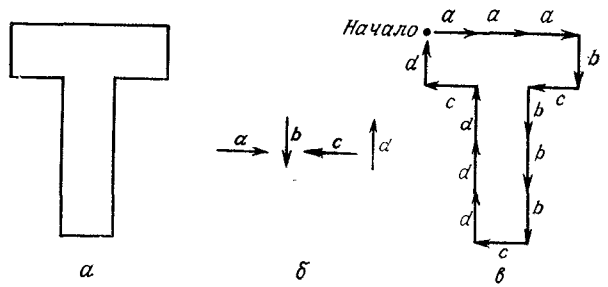


Рис. 8.52. Граница объекта (*а*), простейшие элементы (*б*) и граница, закодированная простейшими элементами (*в*). Соответствующая цепочка имеет вид *aaabcbbbcdddcd*.

т. е. мы имеем $s_4(A)=s_4(B)$, $s_6(A)=s_6(B)$, $s_8(A)=s_8(B)$, ..., $s_k(A)=s_k(B)$, $s_{k+2}(A)\neq s_{k+2}(B)$, $s_{k+4}(A)\neq s_{k+4}(B)$, ..., где s является индексом формы, а нижний индекс означает порядок. *Расстояние* между двумя формами границы A и B определяется как величина, обратная степени схожести:

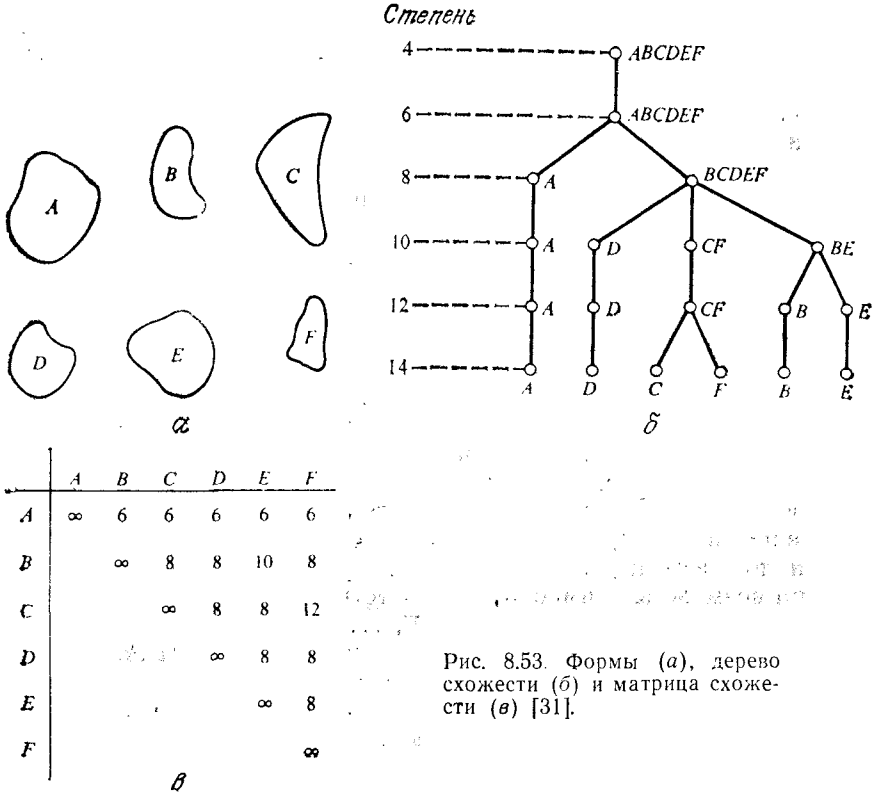
$$D(A,\ B)=\frac{1}{k}. \qquad (8.5\text{-}6)$$

Это расстояние удовлетворяет следующим свойствам:

$$\begin{aligned}
&\text{а)}\quad D(A,\ B)\geqslant 0;\\
&\text{б)}\quad D(A,\ B)=0,\ \text{если}\ A=B;\\
&\text{в)}\quad D(A,\ C)\leqslant \max[D(A,\ B),\ D(B,\ C)].
\end{aligned} \qquad (8.5\text{-}7)$$

Для сравнения двух форм границы используется либо k, либо D. Если используется степень схожести, из сказанного выше следует, что чем больше k, тем формы оказываются более близкими. Обратное справедливо, когда применяется расстояние.

Пример. Для иллюстрации приведенных понятий предположим, что мы хотим определить, какая из пяти форм (A, B, D, E, F) (рис. 8.53, *а*) наилучшим образом соответствует форме C. Это аналогично предположению о том, что свойства пяти форм известны, и требуется определить, какая из них наилучшим образом соответствует форме с неизвестными свойствами. Поиск можно представить наглядно с помощью *дерева схожести*, показанного на рис. 8.53, *б*. Корень дерева соответствует наименьшей рассматриваемой степени схожести, которая в данном примере равна 4. Из рис. 8.53, *б* видно, что все формы являются идентичными вплоть до шестой степени (и до восьмой степени, если исключить форму A). Таким образом, степень схожести формы A по отношению ко всем другим формам равна 6. Продолжая исследование, мы находим, что степень схожести формы D по отношению к остающимся формам равна 8 и т. д.



	A	B	C	D	E	F
A	∞	6	6	6	6	6
B		∞	8	8	10	8
C			∞	8	8	12
D				∞	8	8
E					∞	8
F						∞

в

Рис. 8.53. Формы (*а*), дерево схожести (*б*) и матрица схожести (*в*) [31].

В данном случае только форма F наиболее соответствует форме C, поскольку их степени схожести выше, чем для любых других форм. Если бы неизвестной предполагалась форма E, также была бы найдена единственная форма, но с более низкой степенью схожести. Если была бы неизвестной форма A, применяя этот метод, мы могли бы сказать, что она подобна другим пяти формам со степенью схожести, равной 6. Всю информацию можно объединить в *матрицу схожести* (рис. 8.53, *в*).

Установление соответствия с помощью цепочек. Предположим, что два контура C_1 и C_2 закодированы цепочками a_1,

$a_2 \ldots a_n$ и $b_1 b_2 \ldots b_m$ соответственно. Пусть A является числом соответствий между двумя цепочками, и мы говорим, что соответствие появляется в j-й позиции, если $a_j = b_j$. Число символов, между которыми нет соответствия, дается выражением

$$B = \max(|C_1|, |C_2|) - A, \qquad (8.5\text{-}8)$$

где $|C|$ является длиной (числом символов) цепочки C. Можно показать, что $B = 0$ только тогда, когда C_1 и C_2 идентичны.

Простая мера схожести между цепочками C_1 и C_2 определяется отношением

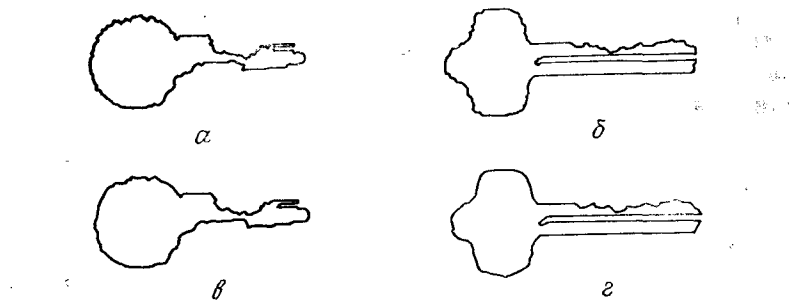
$$R = \frac{A}{B} = \frac{A}{\max(|C_1|, |C_2|) - A}. \qquad (8.5\text{-}9)$$

Поскольку в знаменателе стоит B, то R равно бесконечности для идеального соответствия и нулю, когда между символами в C_1 и C_2 нет ни одного соответствия (т. е. в этом случае $A = 0$). Поскольку соответствие устанавливается в результате поочередного сравнения соответствующих символов, то при создании цепочки, описывающей границу, важно местоположение начальной точки. В противном случае если для каждой границы выбраны произвольные начальные точки, то надо сдвигать одну из цепочек (по кругу) и определять R по уравнению (8.5-9) для каждого сдвига. Число сдвигов, требуемое для выполнения всех необходимых сравнений, равно $\max(|C_1|, |C_2|)$.

Пример. На рис. 8.54, *а* и *б* показаны эталонные границы для двух классов объектов. Границы были аппроксимированны многоугольниками (рис. 8.54, *в* и *г*). Затем в результате вычисления внутренних углов между сегментами многоугольника были сформированы цепочки, причем многоугольник обходился в направлении по часовой стрелке. Углы были закодированы одним из восьми возможных символов, соответствующих приращению 45°: s_1: $0° < \theta \leqslant 45°$, s_2: $45° < \theta \leqslant 90°$, ..., s_8: $315° < \theta \leqslant 360°$.

Результаты вычисления R для эталонной границы объекта 1 и его пяти моделей показаны на рис. 8.54, *д*, где записи соответствуют значениям $R = A/B$ (например, обозначение 1.c соответствует третьей цепочке для объекта класса 1). На рис. 8.54, *е* показаны результаты для цепочек объекта второго класса. Наконец, на рис. 8.54, *ж* показана таблица значений R, полученных в результате сравнения цепочек обоих классов. Важно отметить, что в последней таблице все значения R существенно меньше, чем в предыдущих двух таблицах. Из этого следует, что значения R соответствуют высокой степени различия между двумя классами объектов. Например, если бы цепочка 1.a соответствовала неизвестной форме границы, то ее наименьшее значение при сравнении с другой цепочкой класса 1 было бы равно 4,67. Наоборот, ее наибольшее значение при сравнении с цепочкой второго класса было бы 1,24. Отсюда следует, что эта цепочка относится к первому классу.

Синтаксические методы. Среди структурных методов синтаксические методы наиболее часто применяются для задач распознавания образов. Основная идея, лежащая в основе синтаксических методов, состоит в подробном описании структурной модели простейшими элементами в соответствии с набором правил (*грамматикой*). Сначала рассмотрим грамматики цепочек, а затем распространим эти идеи на грамматики более высокой размерности.



A/B	1.a	1.b	1.c	1.d	1.e
1.b	16,0				
1.c	9,6	26,3			
1.d	5,07	8,1	10,3		
1.e	4,67	7,2	10,3	14,2	
1.f	4,67	7,2	10,3	8,5	23,7

д

A/B	2.a	2.b	2.c	2.d	2.e
2.b	33,5				
2.c	4,75	5,8			
2.d	3,6	4,23	19,3		
2.e	2,83	3,25	9,17	18,3	
2.f	2,63	3,0	7,71	13,5	27,0

е

A/B	1.a	1.b	1.c	1.d	1.e	1.f
2.a	1,24	1,5	1,32	1,47	1,55	1,48
2.b	1,18	1,43	1,32	1,47	1,55	1,48
2.c	1,02	1,18	1,19	1,32	1,39	1,48
2.d	1,02	1,18	1,19	1,32	1,39	1,40
2.e	0,93	1,07	1,08	1,19	1,24	1,25
2.f	0,89	1,02	1,02	1,14	1,11	1,18

ж

Рис. 8.54. Границы объектов, принадлежащих двум различным классам (*а*, *б*), соответствующие им аппроксимации многоугольниками (*в*, *г*) и таблицы $R = A/B$ (*д* — *ж*) [276].

Грамматики цепочек. Предположим, что мы имеем два класса объектов ω_1 и ω_2, которые представляются цепочками из простейших элементов (разд. 8.5.2). Каждый простейший элемент можно интерпретировать как допустимый символ некоторой грамматики, где под *грамматикой* подразумевается набор синтаксических правил (отсюда название — синтаксическое распознавание образов). С помощью этих правил из допустимых символов можно создавать *предложения*. В данном случае предло-

жениями являются цепочки символов, которые в свою очередь служат для описания моделей объектов. Рассмотрим две грамматики G_1 и G_2. Правила грамматики G_1 позволяют создавать предложения только для объектов класса ω_1, а правила грамматики G_2 — только для объектов класса ω_2. Набор предложений, созданный грамматикой G, называется *языком* и обозначается $L(G)$.

После определения двух грамматик G_1 и G_2 процесс синтаксического распознавания образов в принципе довольно прост. Если дано предложение, представляющее неизвестную модель объекта, задача состоит в определении языка, к которому это предложение относится. Если предложение относится к языку $L(G_1)$, мы говорим, что модель описывает объект класса ω_1. Аналогично мы говорим, что объект относится к классу ω_2, если данное предложение из языка $L(G_2)$. Однако предложение может одновременно относиться сразу к двум языкам $L(G_1)$ и $L(G_2)$. В этом случае оно отбрасывается.

Когда имеется больше двух классов моделей объектов, подход синтаксической классификации тот же, что описанный выше. Различие заключается в том, что рассматривается большее число грамматик (по меньшей мере одна на класс). В этом случае модель объекта относится к классу ω_i, если она является предложением *только* из языка $L(G_i)$. Если предложение не относится ни к одному языку или же относится сразу к нескольким, то модель, соответствующая этому предложению, исключается из рассмотрения Для цепочки определим грамматику как четверку

$$G=(N,\ \Sigma,\ P,\ S), \qquad (8.5\text{-}10)$$

где N — конечный набор *нетерминальных* символов или переменных, Σ — конечный набор *терминальных* символов или констант, P — конечный набор *производящих правил* (правил вывода). Символ S, принадлежащий N, является *начальным символом*.

Наборы N и Σ не должны пересекаться. Далее нетерминальные символы обозначаются прописными буквами: A, B, ..., ..., S, ... Константы обозначаются строчными буквами начала алфавита: a, b, c, а цепочки из констант — строчными буквами конца алфавита: v, w, x, y, z. Цепочки, состоящие из констант и переменных, обозначаются строчными буквами греческого алфавита: α, β, θ, ... *Пустое предложение* (предложение, не имеющее символов) обозначается λ. Наконец, если задан набор символов V, то V^* будет обозначать все предложения, составленные из элементов V.

Грамматики цепочек характеризуются в основном видом их производящих правил. Определенный интерес для синтаксического распознавания образов представляют *регулярные грамматики*, производящие правила которых всегда имеют вид $A\to aB$ или $A\to a$, причем A и B принадлежат N, а a принадлежит $\sum$. Представляют интерес также *контекстно-свободные* грамматики с производящими правилами вида $A\to\alpha$, A принадлежит N, а α принадлежит набору $(N\cup\sum)^*-\lambda$; т. е. α может быть любой цепочкой, состоящей из констант и переменных за исключением пустой цепочки.

Пример. Смысл рассмотренных понятий поясним на следующем примере. Предположим, что объект, показанный на рис. 8.55, *а*, представляется своим скелетом. Для описания структуры подобных скелетов введем простейшие элементы, показанные на рис. 8.55, *б*. Рассмотрим грамматику $G=(N, \Sigma, P, S)$ с $N=\{A, B, S\}$, $\sum=\{a, b, c\}$ и производящими правилами

1) $S\to aA$,
2) $A\to bA$,
3) $A\to bB$,
4) $B\to c$,

где константы a, b и c приведены на рис. 8.55, *б*. Как уже говорилось выше, S является начальным символом, с которого создаются все цепочки в языке $L(G)$. Если, например, после правила 1 мы два раза применим правило 2, то получим $S\Rightarrow aA\Rightarrow abA\Rightarrow abbA$, где $\Rightarrow$ означает порядок вывода цепочки, начиная с S и применяя производящие правила из P. Отметим, что производящие правила $S\to aA$ и $A\to bA$ трактуются как «из S может быть выведено aA» и «из A может быть выведено bA». Поскольку в цепочке $abbA$ имеется переменная, то, применяя соответствующие правила, можно продолжить вывод. Например, если к ней два раза применить правило 2, а затем правила 3 и 4, то мы получим цепочку, которая соответствует структуре, приведенной на рис. 8.55, *в*. Отметим, что после применения правила 4 цепочка не содержит переменных, и поэтому вывод прекращается. Нетрудно видеть, что приведенная выше грамматика имеет язык $L(G)=\{ab^nc\,|\,n\geqslant 1\}$, где b^n означает n повторений символа b. Другими словами, с помощью G можно

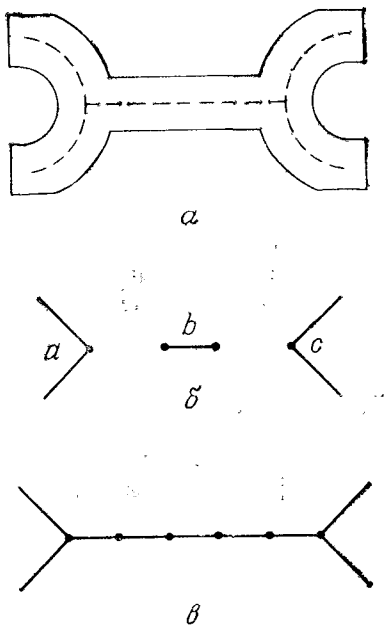


Рис. 8.55. Объект, представленный своим скелетом (*а*), простейшие элементы (*б*) и структура, созданная с помощью регулярной грамматики (*в*).

создавать скелеты структур типа гаечных ключей произвольной длины в пределах разрешения, определяемого длиной простейшего элемента b.

Применение семантик. В рассмотренном выше примере мы неявно предполагали, что взаимосвязь между простейшими элементами устанавливается только в точках, показанных на рис. 8.55, *б*. В более сложных ситуациях правила, устанавливающие связи между элементами, а также другая информация относительно таких особенностей, как длина и направление простейших элементов, число применений правил, должна быть представлена в явном виде. Обычно это делается с помощью *семантик*. Как правило, синтаксис определяет структуру объекта или выражения, а семантика передает ее смысл. Например, предложение на Фортране $A = B/C$ синтаксически правильно, но семантически корректно, только если $C \neq 0$. Для более ясного понимания этих понятий введем семантическую информацию для рассмотренной выше грамматики, предназначенной для описания формы гаечных ключей. Эту информацию вместе с производящими правилами представим следующим образом:

Производящие правила	Семантическая информация
$S \to aA$	Связь с a можно осуществить только в месте, помеченном точкой. Направление a, которое обозначается θ, определяется направлением перпендикуляра, опущенного на линию, соединяющую концы двух отрезков, не помеченных точками. Длина каждого отрезка равна 3 см
$A \to bA$	Связь с b осуществляется только в местах, помеченных точками. Многократные связи недопустимы. Направление b совпадает с направлением a, и длина b равна 0,25 см. Это производящее правило можно применять не более 10 раз
$A \to bB$	Направления a и b должны совпадать. Связи должны быть простыми и осуществляться только в местах, помеченных точками
$B \to c$	Направления c и a должны совпадать. Связи должны быть простыми и осуществляться в местах, помеченных точками

Отметим, что, используя синтаксическую информацию, можно применить несколько синтаксических правил для описания относительно широкого класса моделей. Например, определив направление θ, мы сможем не определять простейшие элементы для каждой возможной ориентации объекта. Аналогично, требуя, чтобы все простейшие элементы имели одинаковое направление, мы исключаем из рассмотрения структуры гаечных ключей, лишенные физического смысла.

Распознавание. До сих пор мы рассматривали грамматики как *генераторы* моделей образов. Ниже мы рассмотрим задачу распознавания, если дана цепочка, описывающая модель. При

этом будем предполагать, что цепочка относится к языку $L(G)$, созданному грамматикой G. Основные понятия, лежащие в основе синтаксического распознавания, можно проиллюстрировать математическими моделями вычислительных машин, называемых *автоматами*. Если на вход такого автомата подается цепочка, описывающая модель, то он способен распознать, относится ли эта модель к определенному языку или классу. Мы рассмотрим только *конечные автоматы*, которые распознают языки, создаваемые регулярными грамматиками.

Конечный автомат определяется как пятерка

$$A = (Q, \Sigma, \delta, q_0, F), \quad (8.5\text{-}11)$$

где Q — конечное непустое множество *состояний*, Σ — конечный входной *алфавит*, функция перехода δ устанавливает соответствие между множеством $Q \times \Sigma$ (набором упорядоченных пар, образованных из элементов Q и Σ) и набором из всех подмножеств Q, q_0 — *начальное состояние* и F (подмножество из Q) — множество *конечных* или *допустимых состояний*. Эту терминологию и обозначения, связанные с выражением (8.5-11), поясним на простом примере.

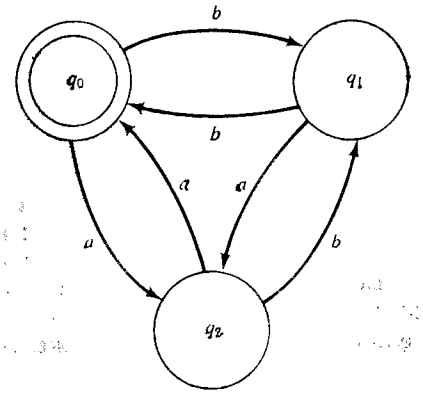


Рис. 8.56. Конечный автомат.

Пример. Рассмотрим автомат, заданный выражением (8.5-11), где $Q = \{q_0, q_1, q_2\}$, $\sum = \{a, b\}$, $F = \{q_0\}$ и функция переходов состояния определяется выражением $\delta = (q_0, a) = \{q_2\}$, $\delta(q_0, b) = \{q_1\}$, $\delta(q_1, a) = \{q_2\}$, $\delta(q_1, b) = \{q_0\}$, $\delta(q_2, a) = \{q_0\}$, $\delta(q_2, b) = \{q_1\}$. Если, например, автомат находится в состоянии q_0 и на входе его символ a, то он перейдет в состояние q_2. Аналогично, если после символа a на входе появляется символ b, автомат перейдет в состояние q_1 и т. д. Отметим, что в этом случае начальное и конечное состояния одно и то же.

Диаграмма состояний для рассмотренного выше автомата приведена на рис. 8.56. На этой диаграмме каждое состояние обозначено вершиной, а направленные дуги указывают возможные переходы между состояниями. Конечное состояние обозначено двойной окружностью, и каждая дуга помечена символом, который вызывает этот переход. О цепочке w терминальных символов говорится, что она *принимается* или *распознается* автоматом, если, начиная из состояния q_0, последовательность символов в w переводит автомат в конечное состояние, после того как на его входе был принят последний символ из w. На-

пример, автомат, приведенный на рис. 8.56, распознает цепочку $w = abbabb$, но отбрасывает цепочку $w = aabab$.

Между регулярными грамматиками и конечным автоматом имеется взаимно-однозначное соответствие, т. е. язык распознается конечным автоматом тогда и только тогда, когда он порождается регулярной грамматикой. Процедура получения автомата, соответствующего данной регулярной грамматике, довольно проста. Пусть имеется грамматика $G = (N, \sum, P, X_0)$, где $X_0 \equiv S$, и предположим, что в N входят X_0 и n нетерминальных символов $X_1, X_2, \ldots, X_n$. Множество состояний Q для автомата образуется в результате введения $n+2$ состояний $\{q_0, q_1, \ldots, q_n, q_{n+1}\}$, таких, что q_i соответствует X_i для $0 \leqslant i \leqslant n$ и q_{n+1} является конечным состоянием. Множество входных символов идентично множеству терминальных символов в G. Функция перехода δ определяется двумя правилами, основанными на правилах из G. Для каждого i и j, $0 \leqslant i \leqslant n$, $0 \leqslant j \leqslant n$:

1. Если $X_i \to aX_j$ находится в P, то $\delta(q_i, a)$ содержит q_j.
2. Если $S_i \to a$ находится в P, то $\delta(q_i, a)$ содержит q_{n+1}.

Если же задан конечный автомат $A = (Q, \Sigma, \sigma, q_0, F)$, то в соответствующей ему регулярной грамматике $G = (N, \Sigma, P, X_0)$ величина N является множеством состояний Q с начальным символом X_0, соответствующим q_0, а производящие правила G получаются следующим образом:

1. Если q_j находится в $\delta(q_i, a)$, то в P имеется производящее правило $X_i \to aX_j$.
2. Если состояние из F находится в $\delta(q_i, a)$, то в P имеется производящее правило $X_i \to a$.

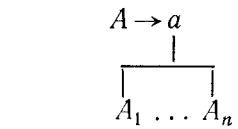
Пример. Конечный автомат, соответствующий приведенной выше грамматике, для описания моделей гаечных ключей можно построить, записав производящие правила следующим образом: $X_0 \to aX_1$, $X_1 \to bX_1$, $X_1 \to bX_2$, $X_2 \to c$. Тогда согласно изложенному выше, мы имеем $A = (Q, \sum, \delta, q_0, F)$, где $Q = \{q_0, q_1, q_2, q_3\}$, $\sum = \{a, b, c\}$, $F = \{q_3\}$, и соответствие, устанавливаемое функцией переходов состояния $\delta(q_0, a) = \{q_1\}$, $\delta(q_1, b) = \{q_1, q_2\}$, $(q_2, c) = \{q_3\}$. Для полноты можно записать $\delta(q_0, b) = \delta(q_0, c) = \delta(q_1, a) = \delta(q_1, c) = \delta(q_2, a) = \delta(q_2, b) = \phi$, где ϕ — пустое множество, указывающее, что для данного автомата такие переходы не определены.

Грамматики более высокой размерности. Рассмотренные выше грамматики пригодны для приложений, где связь между простейшими элементами удобно выражать цепочками. Ниже мы рассмотрим два примера грамматик, способных описывать более общие связи между простейшими элементами и частями моделей.

Древовидная грамматика определяется как пятерка

$$G = (N, \sum, P, r, S), \qquad (8.5\text{-}12)$$

где N и Σ — соответственно множества терминальных и нетерминальных символов; S — начальный символ, который в общем случае может быть деревом; P — набор производящих правил вида $T_i \to T_j$, где T_i и T_j являются деревьями; r — *функция ранжирования*, которая указывает число прямых потомков вершины, помеченной терминальным символом грамматики. *Расширенная* древовидная грамматика имеет производящие правила вида

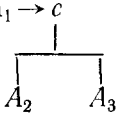


где $A, A_1, \ldots, A_n$ — нетерминальные символы, и a — терминальный символ.

Пример. Скелет структуры, показанной на рис. 8.57, *а*, можно создать с помощью древовидной грамматики, имеющей следующие правила:

1) $S \to a$  A_1

2) $A_1 \to b$  A_1

3) $A_1 \to c$  $A_2 \quad A_3$

4) $A_2 \to d$  A_2

5) $A_2 \to e$

6) $A_3 \to e$  A_3

7) $A_3 \to a$

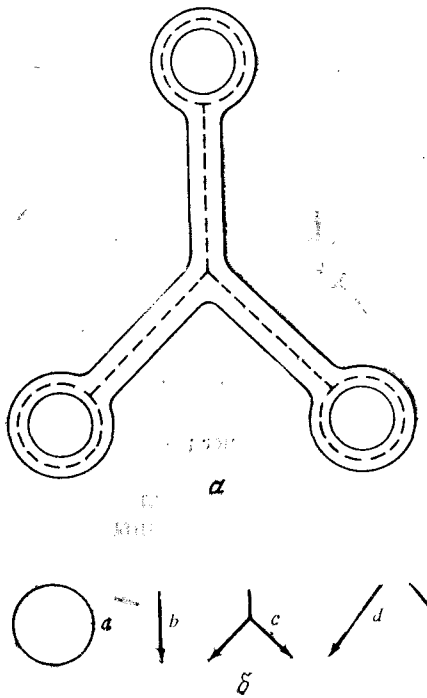


Рис. 8.57. Объект (*а*) и простейшие элементы, использованные для представления скелета с помощью древовидной грамматики (*б*).

где между простейшими элементами из прямых линий устанавливаются связи типа «начало одного с концом другого», а для простейшего элемента в виде окружности связи могут быть установлены в любом месте этой окружности. В этом случае фуикция ранжирования имеет следующий вид: $r(a) = \{0, 1\}$, $r(b) = r(d) = r(e) = \{1\}$, $r(c) = \{2\}$. Отметим, что ограничение, состоящее в том, чтобы правила 2, 4 и 6 были применены одинаковое число раз, приведет к структуре, у которой все три ветви будут иметь одинаковую длину. Аналогично, требование, чтобы правила 4 и 6 были применены одинаковое число раз,

приведет к структуре, симметричной относительно вертикальной оси фигуры, изображенной на рис. 8.57, *а*.

В заключение рассмотрим кратко грамматику, предложенную в работе [95] для построения трехмерных объектов, состоящих из кубических структур. Как и выше, основным моментом

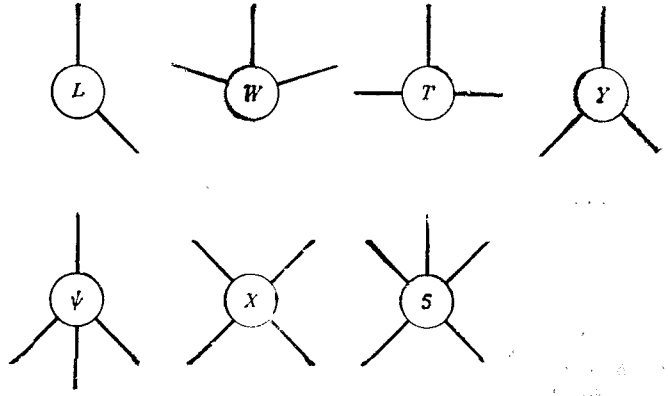


Рис. 8.58. Простейшие элементы в виде вершин [95].

при создании объекта с помощью синтаксических методов является подробное описание набора простейших элементов и их взаимосвязей. В этом случае простейшими элементами являются вершины, показанные на рис. 8.58. В зависимости от

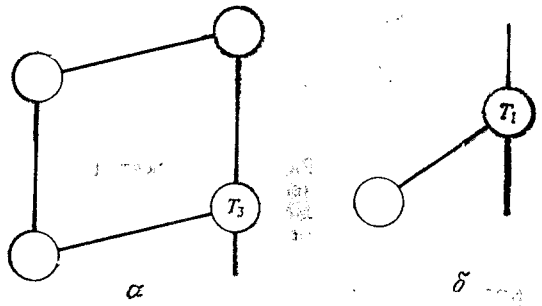
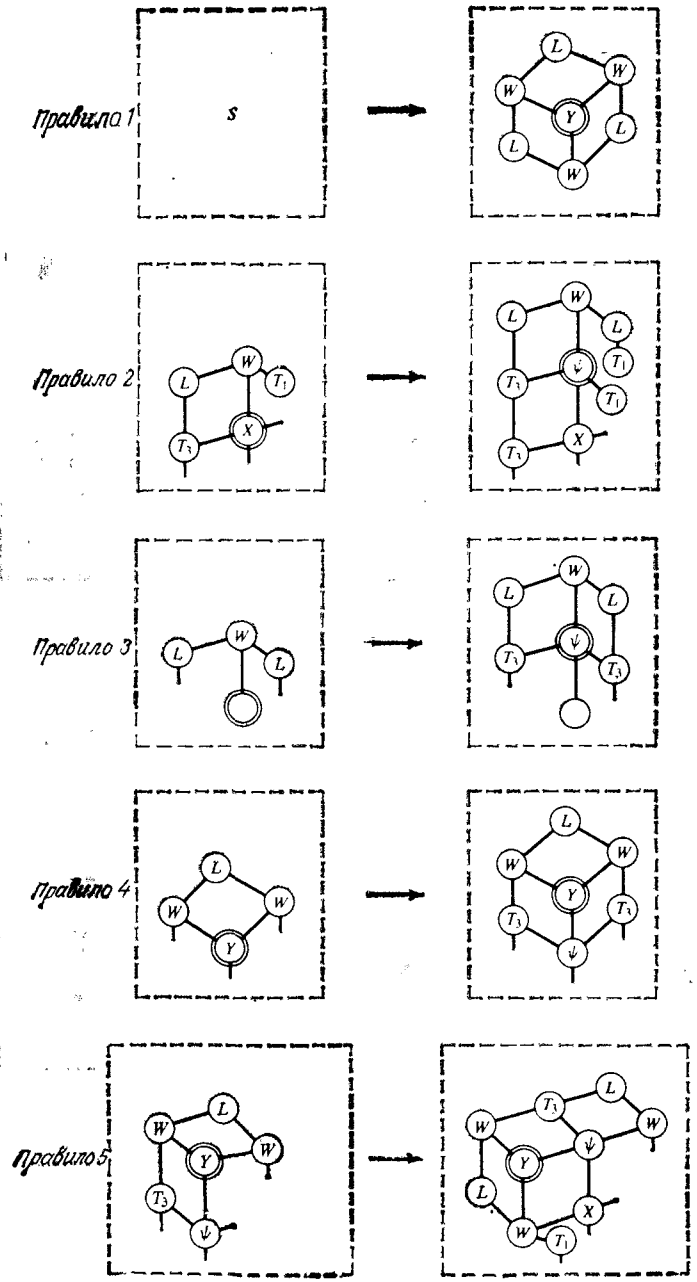


Рис. 8.59. Классификация вершин типа Т.

локальной информации вершина типа T может быть либо вершиной T_1, либо T_3. Если вершина T содержится среди вершин параллелограмма, она относится к типу T_3, в противном случае она относится к вершине T_1. Эта классификация приведена на рис. 8.59.

Грамматические правила заключаются в описании взаимосвязей между структурами (рис. 8.60). Вершины, обозначенные



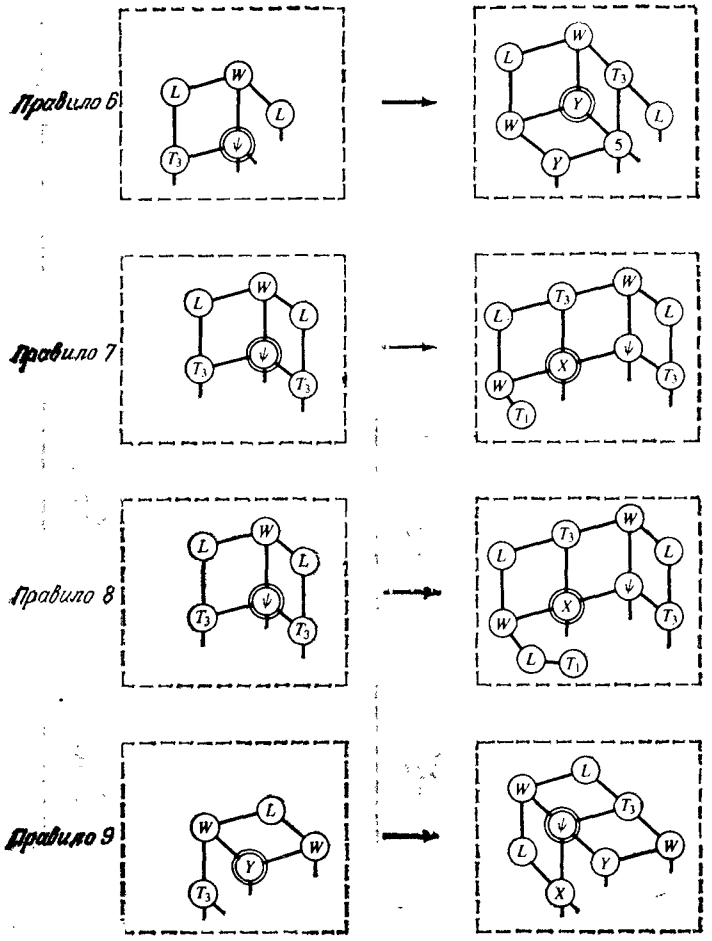


Рис. 8.60. Правила, используемые для генерации трехмерных структур. Пустые круги означают, что допустимо несколько типов вершин [95].

двойными окружностями, обозначают центральные вершины конечного куба объекта, откуда можно осуществлять дальнейшие связи. Это иллюстрируется рис. 8.61, на котором показан

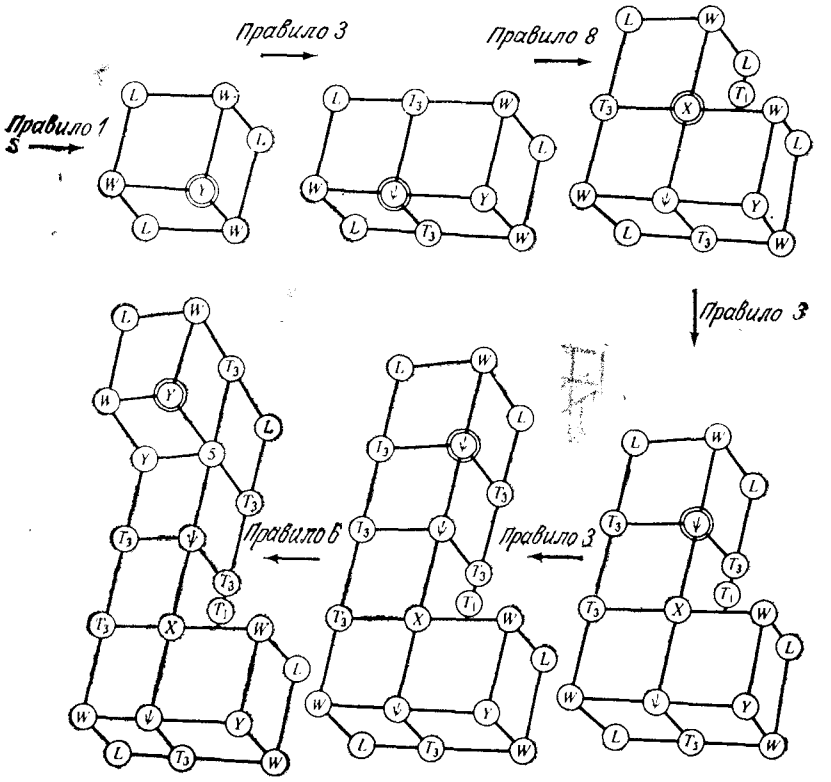


Рис. 8.61. Построение модели с помощью правил, приведенных на рис. 8.60 [95].

типичный вывод, использующий правила, приведенные на рис. 8.60. На рис. 8.62 приводятся виды структур, которые могут быть построены с помощью этих правил.

8.6. ИНТЕРПРЕТАЦИЯ

В этом разделе мы рассматриваем интерпретацию как процесс, который позволяет системе технического зрения приобрести более глубокие знания об окружающей среде по сравнению со знаниями, полученными с помощью методов, изложенных выше. Рассматриваемая с этой точки зрения интерпретация охватывает данные методы как неотъемлемую часть процесса

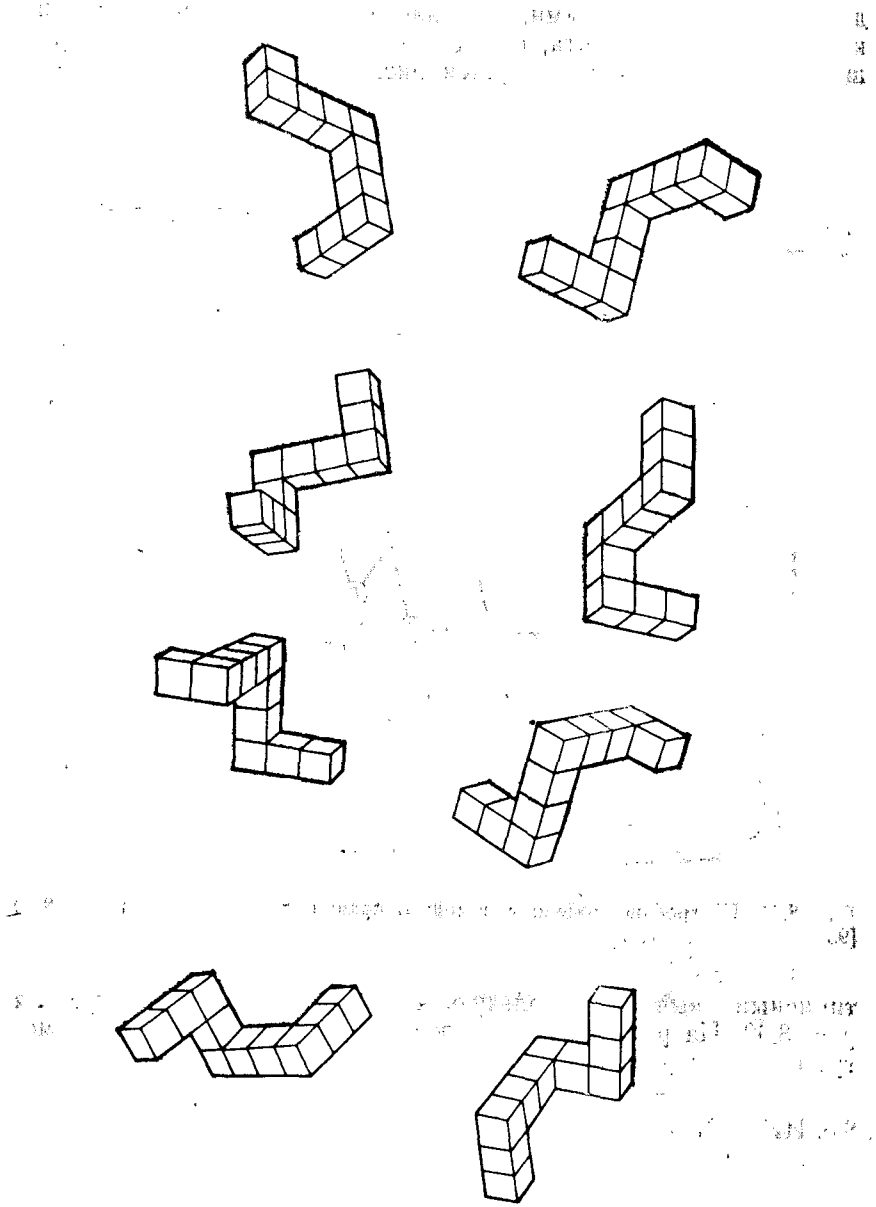


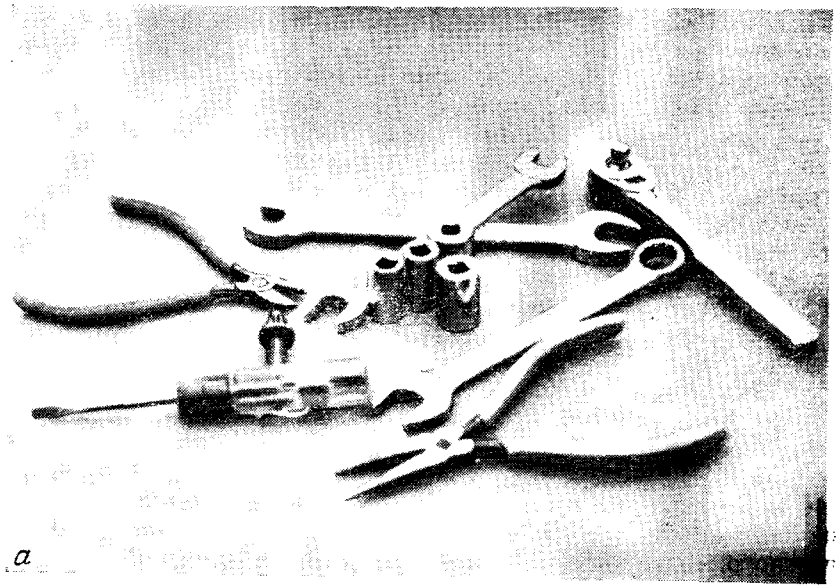
Рис. 8.62. Модели трехмерных структур, созданные с помощью правил, приведенных на рис. 8.60 [95].

понимания зрительной сцены. Хотя в области технического зрения она и является объектом активных исследований, достижения пока весьма незначительны. Ниже мы кратко рассмотрим проблемы, представляющие современные исследования в этой области технического зрения.

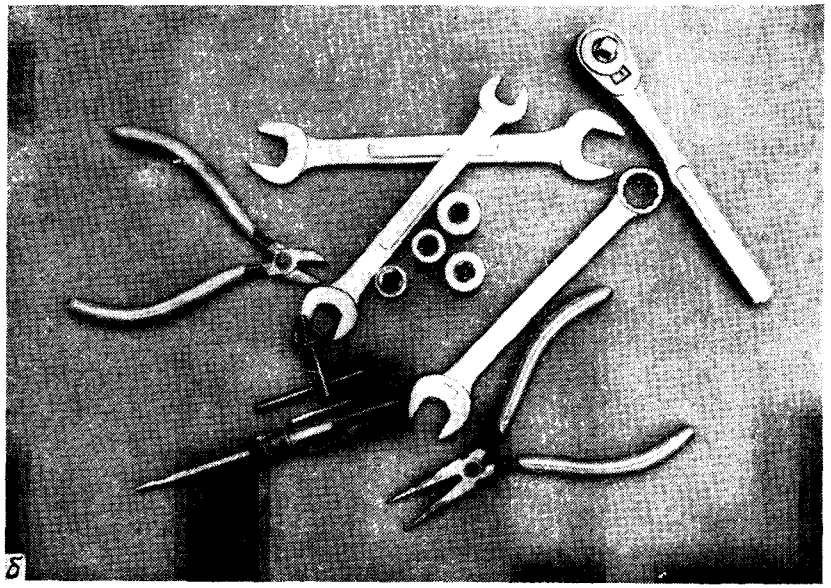
Мощность системы технического зрения определяется ее способностью выделять из сцены значимую информацию при различных условиях наблюдения и использовании минимальных знаний об объектах сцены. По ряду причин (неравномерное освещение, наличие тел, загораживающих объекты, геометрия наблюдения) этот тип обработки представляет трудную задачу. В разд. 7.3 много внимания уделено методам уменьшения разброса в интенсивности. Способы обратного и структурированного освещения, рассмотренные в разд. 7.3, позволяют устранить трудности, связанные с произвольным освещением рабочего пространства. К этим трудностям относятся теневые эффекты, усложняющие процесс определения контуров, и неоднородности на гладких поверхностях. Это часто приводит к тому, что они распознаются как отдельные объекты. Очевидно, многие из этих проблем обусловлены тем, что относительно мало известно о моделировании свойств освещения и отражения трехмерных сцен. Методы разметки линий и соединений, изложенные в разд. 8.4, представляют собой некоторые попытки в этом направлении, но они не в состоянии количественно объяснить эффекты взаимодействия освещения и отражения. Более перспективный подход основан на математических моделях, описывающих наиболее важные связи между освещением, отражением и характеристиками поверхности, такими, как ориентация [124, 142, 189].

Проблема загораживания одних объектов другими имеет место, когда рассматривается большое число объектов в реальном рабочем пространстве. Например, рассмотрим сцену, показанную на рис. 8.63, *а*. Человек без труда определил бы, что за втулками находятся два гаечных ключа. Однако для машины интерпретация этой сцены происходит совершенно по-другому. Даже если бы система была способна идеально выделить группу объектов из фона, то все ранее рассмотренные двумерные процедуры описания и распознавания дали бы плохой результат для большинства загороженных объектов. Применение трехмерных дескрипторов, рассмотренных в разделе 8.4, было бы более успешным, но даже они дали бы неполную информацию. Например, некоторые из втулок оказались бы частично цилиндрическими поверхностями и средний гаечный ключ был бы представлен двумя отдельными объектами.

Для обработки таких сцен, какая показана на рис. 8.63, *а* требуются описания, которые должны содержать информацию о формах и объемах объектов, а также процедуры для установ-



а



б

8.63. Два вида одной трехмерной сцены.

ления связей между этими описаниями, даже когда они не являются полными. Несомненно, эти проблемы будут решены только путем разработки методов, позволяющих обрабатывать трехмерную информацию (полученную либо в результате непосредственных измерений, либо с помощью геометрических методов вывода) и устанавливать (необязательно количественно) трехмерные связи на основе информации об интенсивности образа.

В качестве примера читатель может дать подробную интерпретацию объектов, приведенных на рис. 8.63, *а*, за исключением объекта, загороженного отверткой. Знание о том, в каких случаях интерпретация сцены или части сцены является невозможной, так же важно, как и правильный анализ сцены. Просмотр сцены из различных точек (рис. 8.63, *б*) решил бы эту проблему и был бы естественной реакцией интеллектуального наблюдателя.

В этом направлении одним из наиболее перспективных подходов являются исследования в области технического зрения, основанного на моделях [33]. Основной идеей метода является интерпретация сцены на основе обнаружения отдельных случаев соответствия между данными образа и трехмерными моделями простейших объемных элементов или же целых объектов, представляющих интерес. Зрение, основанное на трехмерных моделях, имеет другое важное преимущество: оно дает возможность обрабатывать несоответствия в геометрии наблюдения. Изменчивость образа объекта, наблюдаемого из различных положений, является одной из наиболее серьезных проблем технического зрения. Даже для двумерных случаев, где определена геометрия наблюдения, ориентация объекта может сильно влиять на процесс распознавания, если он не управляется соответствующим образом (см. замечания в разд. 8.3). Одно из преимуществ подхода, основанного на моделях, состоит в том, что в зависимости от известной геометрии наблюдения можно подбирать ориентацию трехмерных моделей с целью упрощения соответствия между неизвестным объектом и тем, что система видит из данной точки наблюдения.

8.7. ЗАКЛЮЧЕНИЕ

Основное внимание в главе уделено понятиям и методам технического зрения, применяемым в промышленных приложениях. Как указывалось в разд. 8.2, сегментация является одним из наиболее важных процессов на ранней стадии распознавания образов системой технического зрения. Поэтому значительная часть главы посвящена этой задаче. Следующей задачей систе-

мы технического зрения является образование набора дескрипторов, который полностью идентифицирует объекты определенного класса. Как отмечалось в разд. 8.3, обычно стремятся выбирать дескрипторы, наименее зависящие от размеров объекта, его ориентации и расположения. Хотя зрение и является трехмерной задачей, большинство современных промышленных систем работает с данными, которые часто упрощаются с помощью методов специального освещения или строго определенной геометрии наблюдения. Сложности возникают, когда эти ограничения ослабляются (разд. 8.4 и 8.6).

Методы распознавания, изложенные в главе, служат как бы введением в эту широкую область исследований, и на эту тему были написаны десятки книг и тысячи статей. Литература, приведенная в конце главы, дает возможность более глубоко ознакомиться как с теоретическими, так и структурными методами распознавания образов и связанными с ними задачами.

Литература

Дополнительный материал о методах локального анализа, рассмотренных в разд. 8.2.1, можно найти в книге Розенфельда и Кака [252]. Преобразование Хоуга было впервые предложено Хоугом [127] в патенте США и позже изложено в более доступной форме Дудой и Хартом [65]. Обобщение преобразования Хоуга для определения произвольной формы объекта было предложено Баллардом [10]. Графотеоретические методы рассмотрены в двух статьях Мартелли [191, 192]. Другой интересный подход, основанный на поиске пути минимальной стоимости, изложен в работе [243]. Дополнительный материал о методах поиска на графе можно найти в работах [216, 217]. Определение контуров можно осуществить с помощью методов динамического программирования. Более подробно эти задачи изложены в работе [11].

Оптимальные пороги, рассмотренные в разд. 8.2.2, впервые были использованы Шу и Канеко [45] для определения границ на рентгеновских снимках сердца, в которое было введено красящее вещество. Дополнительный материал по оптимальному распознаванию можно найти в работе Фу и Гонсалеса [290]. В книге [252] приводится ряд методов выбора и оценки порогов. Применение характеристик границ для определения порогов рассматривается в статье Уайта и Роурера [307]. Метод использования нескольких переменных для определения порогов предложен Гонсалесом и Винцем [104].

Обзор методов сегментации, основанной на рассмотрении областей (разд. 8.2.3), дан в статье Закера [323]. Более подробно эти задачи изложены в работах [14, 32, 125, 220]. Метод

квадродерева был сначала назван методом регулярной декомпозиции в работах Клингера [146, 147]. Материал разд. 8.2.4 основан на двух статьях Джайна [135, 136]. Другие методы для анализа динамических сцен можно найти в работах [1, 205, 242, 303, 302].

Цепные коды, рассмотренные в разд. 8.3.1, впервые предложены Фриманом [83, 84]. Дополнительный материал по сигнатурам можно найти в работах [4, 11, 206]. В книге Павлидиса [232] подробно изложены методы аппроксимации многоугольниками. Индексы формы рассмотрены в работах [30, 31]. Материал по фурье-дескрипторам можно найти в работах [104, 234, 322]. Трехмерные дескрипторы Фурье рассмотрены в работе [299].

Материал по разд. 8.3.2 можно найти в работе Гонсалеса и Винца [104]. В последние несколько лет дескрипторам текстуры уделялось большое внимание. Статистический подход к исследованию текстуры рассмотрен в работах [9, 51, 111, 112], а структурный подход — в работах [178, 286]. Методы построения скелетов, изложенные в этой главе, приведены в статье Накаши и Шингала [204], в которой имеется большой набор ссылок на другие работы в этой области. Дэвис и Пламмер [53] приводят некоторые фундаментальные результаты по методам прореживания. Определение скелета с помощью дескрипторов Фурье изложено в работе [234]. Инвариантный к моментам подход предложен Ху [128]. Этот метод был обобщен на трехмерный случай Саджади и Холлом [255].

Метод, рассмотренный в разд. 8.4.1, был использован Шираи [265] для сегментации данных о расположении объектов сцены. Метод градиентного оператора, изложенный разд. 8.4.2, разработан Закером и Хаммелом [324]. Одной из ранних работ по анализу сцены на основе метода разметки линий и соединений являются работы [109 и 249]. Более широкое применение этих идей можно найти у Вальца [301, 392].

Современный обзор исследований в этих областях приведен в работе [15]. Для более подробного знакомства с обобщенными конусами имеются работы [2, 209, 189, 262].

Более подробно метод теоретического решения, рассмотренный в разд. 8.5.1, изложен в книге Ту и Гонсалеса [290]. Материал разд. 8.5.2, связанный с подбором индексов формы, основан на статье Брибиеску и Гузмана [31]. Синтаксические методы изложены в работе [276]. Дополнительный материал по структурным методам распознавания образов можно найти в книгах Павлидиса [232], Гонсалеса и Томасона [103] и Фу [88, 89].

По разд. 8.6 смотрите работы Додда и Россола [60], а также работы Балларда и Брауна [11]. Обзор статей по проблемам, рассмотренным в этом разделе, сделан в работе [28].

Упражнения

8.1. 1) Разработайте общую процедуру для получения нормального представления линии, заданной уравнением $y = ax + b$. 2) Найдите нормальное представление линии $y = -2x + 1$.

8.2. 1) Наложите на рис. 8.7 все возможные контуры, задаваемые с помощью графа, приведенного на рис. 8.8. 2) Определите стоимость пути минимальной стоимости.

8.3. Найдите контур, соответствующий пути минимальной стоимости для подобраза, приведенного на рис. 8.64, где цифры в скобках указывают интенсивность. Предположите, что контур начинается на первом столбце и заканчивается на последнем.

	0	1	2
0	(2)	(1)	(0)
1	(1)	(1)	(7)
2	(6)	(8)	(2)

Рис. 8.64.

8.4. Предполагая, что образ имеет распределение интенсивности, показанное на рис. 8.65, где $p_1(z)$ соответствует интенсивности объектов, а $p_2(z)$ — интенсивности фона, найдите оптимальный порог между пикселами объекта и фона при условии что, $P_1 = P_2$.

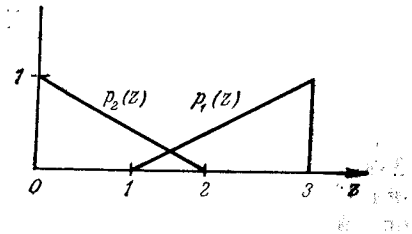


Рис. 8.65.

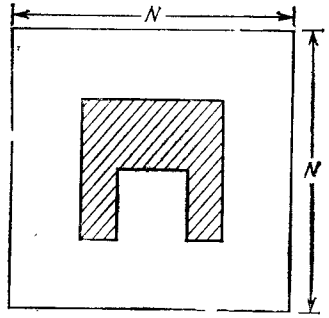


Рис. 8.66.

8.5. Сегментируйте образ, приведенный на рис. 8.66, используя процедуру разбиения и объединения, рассмотренную в разд. 8.2.3. Пусть $P(R_i) =$ ИСТИНА, если все пикселы в R_i имеют одну и ту же интенсивность. Постройте квадродерево, соответствующее вашей процедуре сегментации.

8.6. 1) Покажите, что новый выбор начальной точки цепного кода таким образом, чтобы результирующая последовательность цифр образовывала целое число минимальной величины, делает код независимым от того, с какой точки мы начнем обход границы 2) Какой будет нормированная начальная точка цепного кода 11076765543322?

8.7. 1) Покажите, что применение первой разности цепного кода нормирует его по отношению к повороту как объяснено в разд. 8.3.1. 2) Вычислите первую разность кода 0101030303323232212111.

8.8. 1) Начертите сигнатуру границы квадрата, используя метод тангенса угла, рассмотренный в разд. 8.3.1. 2) Повторите эту процедуру для функции плотности наклона. Предположите, что стороны квадрата совпадают с осями x и y, и возьмите ось x в качестве линии отсчета. Начните с угла, ближайшего к началу координат.

8.9. Задайте несколько дескрипторов-моментов, необходимых для дифференциации форм областей, приведенных на рис. 8.29.

8.10. 1) Покажите, что метод аппроксимации многоугольниками с помощью резиновой ленты, рассмотренный в разд. 8.3.1, дает прямоугольник минимального периметра. 2) Покажите, что если каждый элемент соответствует граничному пикселу, то максимальная ошибка для этого элемента будет $\sqrt{2}d$, где d — расстояние между пикселами (для данной решетки).

8.11. 1) Каким бы был результирующий многоугольник, если в методе объединения, рассмотренном в разд. 8.3.1, положить ошибку, равной нулю. 2) Каким был бы в этом случае результат применения метода разбиения?

8.12. 1) Чему равен порядок индекса формы для каждой из фигур, приведенных на рис. 8.67. 2) Найдите индекс формы для четвертой фигуры.

8.13. Вычислите среднее значение и дисперсию для 4-уровневого образа, имеющего гистограмму $p(z_1) = 0{,}1$, $p(z_2) = 0{,}4$, $p(z_3) = 0{,}3$, $p(z_4) = 0{,}2$. Предположите, что $z_1 = 0$, $z_2 = 1$, $z_3 = 2$ и $z_4 = 3$.

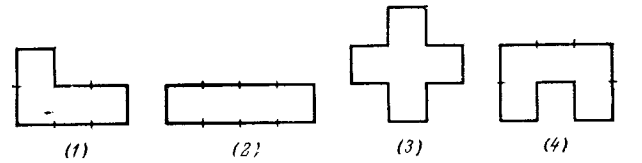


Рис. 8.67.

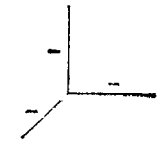


Рис. 8.68.

8.14. Для образа размерностью 5×5, составленного из 1 и 0, меняющихся в шахматном порядке, составьте матрицу вероятностей совместного появления уровней интенсивности, если 1) P определяется как «один пиксел справа» и 2) «два пиксела справа». Предположите, что верхний левый пиксел имеет значение 0.

8.15. Рассмотрите образ, который состоит из белых и черных квадратов размером $m \times m$, расположенных в виде шахматной доски. Найдите оператор положения, который дал бы диагональную матрицу вероятностей совместного появления уровней освещенности.

8.16. 1) Покажите, что средней осью круга является его центр. 2) Начертите среднюю ось прямоугольника, области между двумя концентрическими окружностями и равностороннего треугольника.

8.17. 1) Докажите, что уравнение (8.3-6) реализует условия, задаваемые четырьмя окнами, приведенными на рис. 8.40. 2) Нарисуйте окна, соответствующие B_0 в уравнении (8.3-7).

8.18. Нарисуйте трехгранный объект, имеющий соединения следующего вида (рис. 8.68).

8.19. Докажите, что использование уравнения (8.5-4) для определения того, к какому классу принадлежит объект, описываемый модельным вектором, эквивалентно использованию уравнения (8.5-3).

8.20. Докажите, что $D(A, B) = \frac{1}{k}$ удовлетворяет трем условиям, задаваемым уравнением (8.5-7).

8.21. Покажите, что $B = \max(|C_1|, |C_2|) - A$ в уравнении (8.5-7) равно нулю только тогда, когда C_1 и C_2 являются идентичными цепочками.

Глава 9.
ЯЗЫКИ ПРОГРАММИРОВАНИЯ РОБОТОВ

> Одна и та же физическая реальность не приводит различных наблюдателей к построению одинаковой картины мира, если они говорят на языках, между которыми не установлено соответствие.
>
> *Бенджамин Л. Ворф*

9.1. ВВЕДЕНИЕ

В предыдущих главах обсуждались вопросы кинематики, динамики, управления, планирования траекторий, очувствления и зрения для манипуляторов, управляемых ЭВМ. Алгоритмы, предназначенные для выполнения этих задач, обычно реализуются в виде управляющих программных модулей. Основная трудность использования манипуляторов в качестве универсальных сборочных машин заключается в том, что пока не создана эффективная связь между пользователем и робототехнической системой, которая необходима для того, чтобы пользователь мог управлять манипулятором при выполнении задачи.

Основными средствами реализации связи между пользователем и роботом являются: распознавание отдельных слов, обучение с последующим воспроизведением роботом рабочей программы, а также языки программирования высокого уровня.

Современные системы распознавания речи весьма примитивны и обычно зависят от говорящего. Они могут распознавать ряд отдельных слов из ограниченного словаря и обычно требуют, чтобы слова отделялись друг от друга паузой. Хотя сейчас можно распознавать отдельные слова в реальном времени благодаря быстродействующим вычислительным средствам и эффективным алгоритмам обработки, однако способность распознавания отдельных слов для описания задачи робота довольно ограничена. Более того, распознавание речи обычно требует большого объема оперативной памяти или вторичной памяти (на внешних запоминающих устройствах) для хранения речевых данных, что в свою очередь требует дополнительного времени для обучения построению эталонов речи, необходимых при распознавании.

Обучение с целью последующего воспроизведения роботом рабочей программы, известное также как «программирование обучением», является наиболее распространенным методом управления современными промышленными роботами. По этому

методу робот «обучается» тем движениям, которые затем выполняет в автоматическом режиме. Программирование обучением выполняется за несколько шагов: 1) ведение робота в медленном режиме при ручном управлении для выполнения технологической операции и запись углов между звеньями робота в соответствующих положениях с целью повторного воспроизведения движения; 2) корректировка и воспроизведение обучающего движения; 3) если обучающее движение правильно, робот запускается в работу на соответствующей скорости для выполнения повторяющихся операций.

При ведении робота в режиме обучения обычно можно использовать ручной привод, пульт независимого управления или специальную систему с кнопочной клавиатурой отслеживания принудительного движения робота. В настоящее время для обучения наиболее часто употребляется ручной пульт с кнопочной клавиатурой. В этом случае пользователь ведет робот вручную в пространстве и нажимает на соответствующую кнопку, чтобы записать любое желаемое угловое положение манипулятора. Угловые положения записываются в память ЭВМ в виде ряда точек траектории, через которые проходит манипулятор. Затем по этим точкам численными методами производится интерполяция, и робот воспроизводит программный режим вдоль сглаженной траектории. В режиме корректировки программного движения пользователь может произвести коррекцию записанных угловых положений и удостовериться, что робот не столкнется с препятствиями во время выполнения задачи. В рабочем режиме робот будет выполнять рабочие циклы по откорректированной и сглаженной траектории. При изменении условий задачи повторяются названные выше шаги. Преимущества этого метода заключаются в простоте обучения и в относительно малом объеме памяти ЭВМ, требуемом для записи угловых положений звеньев робота. Основной недостаток — трудность использования этого метода для включения информации с датчиков обратной связи в систему управления.

Языки программирования высокого уровня дают более общий подход к решению проблемы связи человек — робот. За прошедшее десятилетие роботы были успешно применены в таких областях, как дуговая сварка и окраска, причем в обоих случаях, как правило, используется программирование обучением [71]. Эти задачи не требуют взаимодействия между роботом и окружающей средой и могут быть легко запрограммированы. Однако применение роботов для реализации задач сборки требует методов программирования на языках высокого уровня, так как сборочный робот обычно имеет сенсорную обратную связь, и этот тип нестандартного взаимодействия робота с окружающей средой может быть осуществлен только программными методами.

Программирование робота существенно отличается от традиционного программирования. При выборе метода программирования робота необходимо иметь в виду следующие соображения: объекты, которыми манипулирует робот, являются трехмерными и имеют набор физических свойств; роботы функционируют в пространственно-сложной среде; описание и представление трехмерных объектов в ЭВМ — неточное; информация

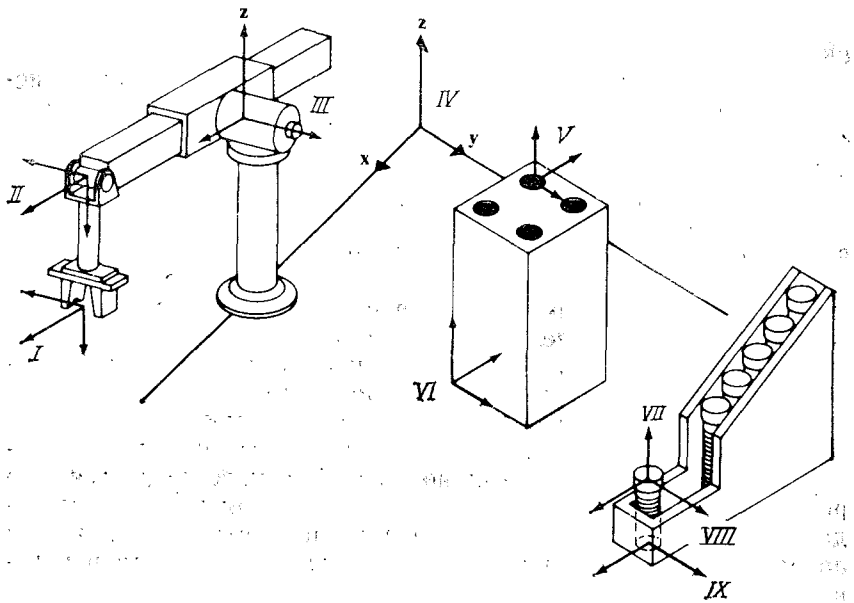


Рис. 9.1. Задача установки роботом болта в отверстие.

I — система координат, связанная со схватом (E); II — система координат, связанная с сочленением (T_6); III — система координат, связанная с корпусом; IV — неподвижная система координат; V — система координат, связанная с отверстием детали; VI — система координат, связанная с деталью; VII — направление захвата болта; VIII — система координат, связанная с головкой болта; IX — система координат, связанная с загрузочным устройством.

с устройств очувствления должна быть собрана, обработана и соответствующим образом использована.

Современные подходы к программированию роботов можно разделить на две категории: программирование, ориентированное на робот (роботоориентированное программирование), и проблемно-ориентированное программирование, или программирование на уровне задачи.

В роботоориентированном программировании задача сборки описывается в виде последовательности движений. Робот управляется программой в течение выполнения всей задачи, причем каждый шаг программы примерно соответствует одному действию робота. При программировании на уровне задачи сборка

описывается в виде последовательности целей, которая определяет неявное движение робота. Эти подходы обсуждаются подробно в следующих двух разделах.

9.2. ХАРАКТЕРИСТИКИ РОБОТООРИЕНТИРОВАННЫХ ЯЗЫКОВ

Наиболее общий подход при создании роботоориентированных языков состоит в расширении возможностей существующих языков высокого уровня (чтобы соответствовать требованиям программирования робота). Этот подход специфичен, и не всегда имеются общие правила, как осуществлять расширение языка. Мы можем легко определить несколько основных характеристик, которые являются общими для всех языков, ориентированных на роботы, с помощью анализа шагов разработки программы движения робота. Рассмотрим задачу установки болта в отверстие детали роботом (рис. 9.1). Робот должен приблизиться к загрузочному устройству, извлечь болт, переместиться к детали и вставить болт в одно из отверстий.

Обычно для реализации этой программы требуются следующие шаги:

1. Определение рабочего пространства и фиксация объектов в зажимных приспособлениях и загрузочных устройствах.
2. Описание на языке программирования расположения объектов (*загрузочного устройства*, *детали* и т. д.) в пространстве и их отношений (*деталь_отверстие*, *болт_схват*, и т. д.) [1]
3. Разделение задачи сборки на последовательность действий, таких, как движение робота, захват объектов, установка болта.
4. Добавление команд с устройств очувствления для определения нестандартных ситуаций (таких, как невозможность определения местонахождения болта во время захвата) и для управления выполнением задачи сборки.
5. Отладка и совершенствование программы с помощью повторения шагов 2—4.

Важными характеристиками, которые мы определили, являются описания положения (шаг 2), движения (шаг 3) и очувствления (шаг 4). Эти характеристики подробно обсуждаются ниже.

В качестве примеров рассмотрим языки *AL* [200] и *AML* [283]. Выбор этих языков не является произвольным. *AL* повлиял на создание многих роботоориентированных языков и до сих пор активно развивается.

---

[1] Символ подчеркивания обычно используется в языках программирования для обеспечения соответствия в названиях переменных.

Он обладает широким набором команд для удовлетворения требованиям программирования робота и также свойствами языка программирования высокого уровня. Язык *AML* в настоящее время предназначен для управления роботами фирмы IBM. *AML* построен на иных принципах, чем *AL*. С помощью этих языков воссоздается окружающая среда робототехнической системы, в которой могут быть построены различные программируемые интерфейсы для роботов. Таким образом, имеется богатый набор основных элементов для описания операций робота, что дает возможность пользователям реализовывать команды высокого уровня в соответствии с их требованиями. В настоящее время *AL* и *AML* являются наиболее распространенными роботоориентированными языками. Они описаны в табл. 9.1.

Таблица 9.1. **Краткая характеристика роботоориентированных языков програмирования AL и AML**

AL разработан в Стандфордском университете. В настоящее время он может быть реализован на ЭВМ типа VAX, и управление роботом в реальном времени осуществляется с одиночной стойки PDP-11.

- Язык программирования высокого уровня с характерными чертами Алгола и Паскаля.
- Оснащен спецификациями как на уровне робота, так и на уровне задачи.
- Переведен на язык программирования низкого уровня и реализован на машине, осуществляющей управление в реальном времени.
- Имеет конструкции языка программирования в реальном времени (синхронизация, совместное выполнение и условные переходы).
- Представление данных и структур управления осуществляется, как и в языке Алгол.
- Оснащен средствами для моделирования рабочего пространства.

AML разработан фирмой IBM. Он предназначен для управления роботами типа IBM RS-1 и реализован на ЭВМ SERIES-1 (или персональном компьютере фирмы IBM), которая также допускает возможность управления роботом. Робот RS-1 представляет собой манипулятор, работающий в декартовой системе координат и имеющий 6 степеней свободы. Первые три сочленения робота — призматические и три последние — вращательные.

- Обеспечивает описание окружающей среды, в которой могут быть построены различные интерфейсы пользователя.
- Обладает свойствами и конструкциями, подобными свойствам и конструкциям языка LISP, и способен к агрегированию данных.
- Имеет возможность планирования траекторий сочленений в пространстве, допускает ограничения по положению и скорости, а также позволяет описывать абсолютные и относительные движения.
- Обеспечивает управление от датчиков для возможного прерывания движения робота.

9.2.1. Определение положения

Обычно при роботизированной сборке взаимное расположение робота и деталей строго определено. Детали закрепляются в технологических приспособлениях так, чтобы минимизировать

отклонения от заданного положения. Для сборки произвольно расположенных деталей требуются системы технического зрения, и поэтому она еще не получила широкого распространения в промышленности.

Наиболее общим подходом для описания ориентации и положения объектов в рабочем пространстве является подход, предусматривающий использование систем координат и соответствующих им структур данных, называемых фреймами. Фреймы представляются матрицами однородного преобразования размерностью 4×4. Фрейм состоит из подматрицы размерностью 3×3 (определяющей ориентацию) и вектора (определяющего положение), которые в свою очередь определены по отношению к некоторой базовой системе координат (базовому фрейму). В табл. 9.2 приведены описания на языках *AL* и *AML* для трех фреймов, соответствующих системам координат, связанным с *роботом*, *деталью* и *загрузочным устройством*, показанных на рис. 9.1.

Таблица 9.2. **Определение основных фреймов на языках AL и AML**

AL:

Робот ← FRAME(nilrot, VECTOR (508, 0, 381) * мм);
Деталь ← FRAME (ROT (Z, 90 * град), VECTOR (508, 381, 0) * мм);
Загрузочное устройство ← FRAME(nilrot, VECTOR (635, 508, 0) * мм);

Примечания: 1) nilrot — предварительно определенный фрейм, который имеет значение ROT(Z, 0*град);
2) ← — оператор присваивания в языке AL;
3) точка с запятой обозначает конец программного выражения;
4) звездочка (*) является оператором умножения, который зависит от типа переменных. Здесь она используется, чтобы добавить единицы измерения к элементам вектора.

AML:

Робот = ⟨⟨508, 0, 381⟩, EULERROT (⟨0, 0, 0⟩)⟩;
Деталь = ⟨⟨508, 381, 0⟩, EULERROT (⟨0, 0, 90⟩)⟩;
Загрузочное устройство = ⟨⟨635, 508, 0⟩, EULERROT (⟨0, 0, 0⟩)⟩;

Примечание. EULERROT — подпрограмма, формирующая матрицу вращения, которая дает требуемые значения углов.

Язык *AL* позволяет формировать структуры данных декартовых систем координат |*FRAME*|, матриц вращения |*ROT*| и векторов |*VECTOR*|. С другой стороны, язык *AML* предусматривает обобщенную структуру, называемую *агрегатом*, которая позволяет пользователю конструировать свои собственные структуры данных. В табл. 9.2 приведена структура данных для представления декартовой системы координат, имеющая формат ⟨вектор, матрица⟩. В этой структуре данных вектор положения робота является агрегатом, состоящим из трех скаляров, а матрица представляет собой агрегат из трех векторов ориентации.

Рассмотрим более подробно обозначения в табл. 9.2. Первый *AL*-оператор описывает систему координат *робота*, главные оси которой (*nilrot* обозначает отсутствие вращения) параллельны осям неподвижной системы координат. Начало системы координат, связанной с роботом, смещено на расстояние (508, 0, 381) мм от начала неподвижной системы координат. Второй *AL*-оператор описывает систему координат, связанную с *деталью*, главные оси которой повернуты на 90° вокруг оси *Z* относительно неподвижной системы координат. Начало координат расположено на расстоянии (508, 0, 381) мм от начала неподвижной системы координат. Третий оператор имеет то же значение, что и первый, за исключением расположения. Смысл трех операторов на языке *AML* тот же, что для операторов на языке *AL*. Таким образом в этих языках положение в пространстве определяется с помощью фрейма, ориентированного относительно базового фрейма.

Преимущество использования матрицы однородного преобразования состоит в том, что фреймы, определенные относительно базового, могут быть получены путем умножения матрицы преобразования на базовый фрейм. В табл. 9.3 приводится сравнение операторов на языках *AL* и *AML*, которые используются для определения систем координат и отношений *болт_головка болта*, *болт_схват* и *деталь_отверстие* (рис. 9.1). Для описания матриц преобразования язык *AL* предусматривает оператор умножения матриц * и структуру данных *TRANS*-преобразование, которое состоит из операций вращения и перемещения. Язык

Таблица 9.3. **Описание основных систем координат на языках AL и AML**

На языке AL:

```
T6 ← робот*TRANS ((ROT, 180*град), VECTOR(381, 0, 0)*мм);
E ← T6*TRANS(nilrot, VECTOR(0, 0, 127)*мм);
болт_головка болта ← загрузочное устройство*TRANS(nilrot, nilvect)
болт_схват ← болт_головка болта*TRANS(nilrot, VECTOR(0, 0, 25.4)*мм);
деталь_отверстие ← деталь*TRANS(nilrot, VECTOR(0, 30.8, 76.2)*мм);
```

Примечание. Nilvect — предварительно определенный вектор, имеющий значение VECTOR(0, 0, 0)*мм.

На языке AML:

```
T6=DOT(робот, ≪381, 0, 0>, EULERROT(<180, 0, 0>)>);
E=DOT(T6, ≪0, 0, 127>, EULERROT(<0, 0, 0>)>);
болт_головка болта=DOT (загрузочное устройство, ≪0, 0, 0>,
                                        EULERROT(<0, 0, 0>)>);
болт_схват=DOT (болт_головка болта <0, 0, 25.4>,
                                        EULERROT(<0, 0, 0>)>);
деталь_отверстие=DOT (деталь, ≪0, 50.8, 76.2>,
                                        EULERROT(<0, 0, 0>)>);
```

Примечание. DOT — подпрограмма перемножения двух матриц

AML не имеет встроенного оператора матричного умножения, но оснащен системной подпрограммой *DOT*.

Рассмотрим более подробно обозначения в табл. 9.3. Первый *AL*-оператор описывает систему координат *T*6, главные оси которой повернуты на 180° относительно оси *X* системы координат, связанной с роботом. Начало системы координат *T*6 расположено на расстоянии (381, 0, 0) мм от начала системы координат, связанной с роботом. Второй оператор описывает систему координат *E*, главные оси которой параллельны (*nilrot* обозначает отсутствие вращения) главным осям системы координат *T*6, а начало расположено на расстоянии (0, 0, 127) мм от начала системы координат *T*6. Аналогичные рассуждения применимы ко всем другим трем операторам на языке *AL*. Такой же смысл имеют операторы на языке *AML*. Связь между фреймами, которые мы определили в табл. 9.2 и 9.3, представлена на рис. 9.2, *а*.

Отметим, что в языке *AL* не требуются фреймы, определяющие положение руки робота, поскольку в нем используется неявный фрейм для представления конечного положения схвата и нет доступа к промежуточным фреймам *T*6 и *E*. Так как в процессе сборки детали перемещаются или взаимодействуют с другими объектами, то последовательность фреймов в программе должна соответствовать текущим состояниям рабочего пространства (рис. 9.2, *б*).

Другой способ определения ориентации и положения объекта состоит в том, что схват робота в интерактивном режиме перемещается от одной фиксированной точки до другой. В качестве примера приведем систему *POINTY* [107], использующую язык *AL*. Эта система позволяет пользователю вести робот вручную или с помощью подвесного пульта управления через рабочее пространство. При этом когда пользователь направляет руку робота, оснащенную специальным инструментом, к объекту, то система генерирует команды на языке *AL*, подобные командам, приведенным в табл. 9.2 и 9.3. Это устраняет необходимость измерять расстояния и углы между системами координат, что весьма трудоемко.

Хотя система декартовых координат широко применяется для представления конфигураций робота, она имеет некоторые ограничения. Более удобным способом представления конфигураций робота по сравнению с декартовой системой координат является система обобщенных координат. Поскольку обратная задача кинематики не дает единственного решения, конфигурация робота определяется неоднозначно в декартовом пространстве. По мере увеличения числа операций и объектов связи между системами координат становятся сложными и трудными для управления. Кроме того, число требуемых вычислений также существенно увеличивается.

9.2.2. Определение движения

Наиболее общей операцией в роботизированной сборке является операция типа «взять и установить». Она включает дви-

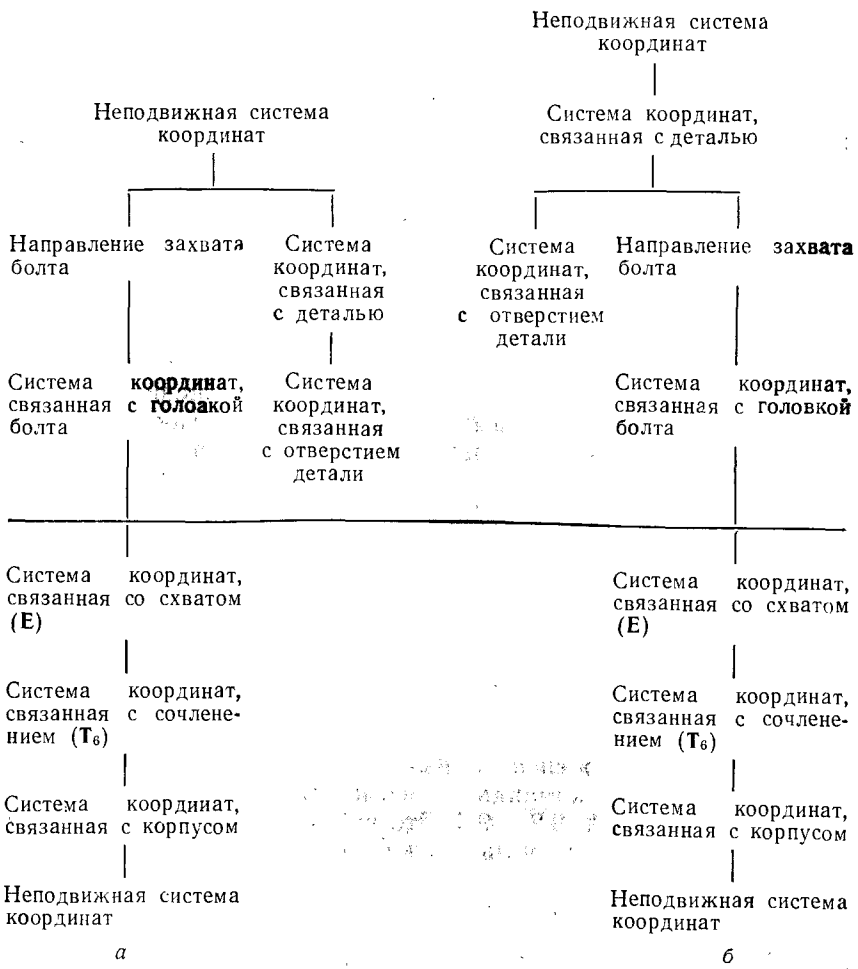


Рис. 9.2. Связи между фреймами.

жение робота от начального положения к положению схватывания, захват объекта и движение в конечное положение. Движение обычно определяется как последовательность промежуточных положений, которые должен достигнуть робот. Однако определения только начального и конечного положения не достаточно.

Вначале управляющая система планирует траекторию движения робота без учета расположения объектов в рабочем пространстве и возможных препятствий на планируемой траектории. Для того чтобы управляющая система могла генерировать траекторию, свободную от столкновений с препятствиями, программист должен задавать достаточно часто промежуточные точки траектории. Например, как показано на рис. 9.3, если

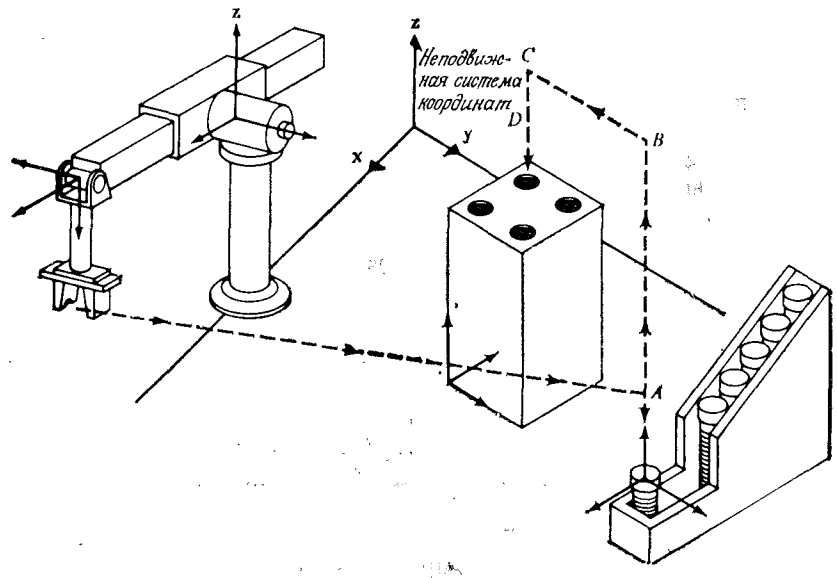


Рис. 9.3. Траектория движения робота.

было бы задано движение по прямой линии от точки *A* к точке *C*, то робот столкнулся бы с деталью. Поэтому для обеспечения безопасности движения робота необходимо ввести промежуточную точку *B*. Промежуточные положения робота могут быть определены либо в обобщенной, либо в декартовой системе координат в зависимости от языка программирования. В языке *AL* движение определяется с помощью команды *MOVE*, которая указывает систему координат требуемого положения руки робота. Промежуточные точки определяются с помощью ключевого слова *VIA*, за которым следует фрейм промежуточной точки (табл. 9.4). На языке *AML* пользователь имеет возможность определять движение робота в обобщенных координатах и составлять собственные программы для представления движения в декартовых координатах. Сочленения определяются номерами сочленений (от 1 до 6), и движение может быть либо относительным, либо абсолютным (табл. 9.4).

Таблица 9.4. **Примеры описания движеиия руки робота на языках AL и AML**

На языке AL:

{перемещать руку из состояния покоя к системе координат A и затем к системе координат *болт_схват*}

MOVE barm TO A;
MOVE barm TO *болт_схват*;

{другое описание движения робота}

MOVE barm TO *болт_схват* VIA A;

{двигаться вдоль текущей оси Z на 25,4 мм (относительное движение)}

MOVE barm TO ⊗ —25,4*Z*мм;

Примечания: barm — идентификатор, обозначающий руку робота;
⊗ указывает текущее положение руки, эквивалентное выражению: робот *T6*E.

Выражения в скобках {...} являются комментариями.
На языке AML:

• "Переместить сочленения 1 и 4 на 234 мм и 20° соответственно (абсолютное движение)"

MOVE (<1,4>, <234,20>);

• "Переместить сочленения 1, 3 и 6 на 25,4 мм, 50,8 мм и 5° соответственно (относительное движение)"

DMOVE (<1,3,6>, <25,4, 50,8>);

Примечание: Выражения в кавычках "..." являются комментариями.

К недостаткам этого типа описания относится то, что программист должен предварительно спланировать все движения робота, чтобы выбрать промежуточные точки. В результате робот может совершать лишние движения по траектории. Более того, описание сложной траектории как последовательности точек приводит к неоправданно длинной программе.

При движении руки робота из начального положения в конечное имеются физические ограничения, например такие, как места закрепления звеньев, которые требуют, чтобы рука шла вдоль оси. Кроме того, имеются ограничения на движение робота в рабочем пространстве, связанные с расположением объектов. В обоих случаях это может препятствовать требуемому движению робота.

Для обеспечения безопасного движения программист должен управлять такими параметрами, как скорость, ускорение, замедление, направление движения. Обычно эти параметры рассматриваются как ограничения, которым должно удовлетворять программное движение. Язык *AL* оснащен ключевым словом *WITH*, с помощью которого к командам движения робота присоединяются операторы ограничения. В табл. 9.5 приведены команды движения робота из состояния «*болт_схват* к точке *A*

Таблица 9.6. **Примеры описаиия движения руки робота иа языках AL и AML**

На языке AL:
{Перемещать руку от системы координат *болт_схват* к системе координат A}

MOVE barm TO A
WITH DEPARTURE=Z WRT *загрузочное устройство*
WITH DURATION=5*с.;

{открыть схват на 63,5 мм}

OPEN bhand TO 63,5*мм

Примечание. С помощью команды WRT (по отношению к) создается вектор в указанной системе координат.

На языке AML:

• Переместить сочленения 1 и 4 на 254 мм и 20° со скоростью 25,4 мм/с.
• Ускорение и замедление равны 25,4 мм/с².

MOVE (<1,4>, <254,20>, <25,4, 25,4, 25,4>);

• Открыть схват на 63,5 мм

MOVE (GRIPPER, 63.5);

с направлением начала движения вдоль положительного направления оси $+Z$ *загрузочного устройства* и временной длительностью 5 с, т. е. медленное движение. В языке *AML* агрегаты типа ⟨скорость, ускорение, замедление⟩ могут быть добавлены к команде *MOVE*, чтобы определить скорость, ускорение и замедление робота.

Обычно движения схвата должны рассчитываться в зависимости от поставленной задачи и окружающих условий зоны рабочего пространства. В большинстве языков предусмотрены простые команды движений схвата, из которых строятся сложные движения. Для двупалого схвата могут быть предусмотрены команды «открыть» (разжатие пальцев) или «закрыть» (сжатие пальцев). Оба языка *AL* и *AML* используют предварительно определенные переменные для обозначения схвата (в *AL* — *bhand*, в *AML* — *GRIPPER*). Используя команду *OPEN* (в *AL*) и *MOVE* (в *AML*), можно запрограммировать требуемое раскрытие схвата (табл. 9.5).

9.2.3. Очувствление и управление

Расположение и размеры объектов в рабочем пространстве, как правило, определяются только с некоторой степенью точности. Чтобы робот выполнял задачи при наличии этих ограничений, должно быть реализовано очувствление. Информация, собранная с датчиков очувствления, также действует как обратная связь с рабочим пространством, что помогает роботу исследо-

вать и проверять состояние сборки. Очувствление в программировании робота условно делится на 3 типа:

1. *Позиционное очувствление.* Применяется для определения текущего положения робота. Оно обычно дается кодировщиками, чтобы измерять углы в сочленениях и вычислять соответствующее положение руки робота в рабочем пространстве.
2. *Силовое и тактильное очувствление.* Применяется для определения наличия объектов в рабочем пространстве. Силовое очувствление используется при управлении упругой податливостью, чтобы обеспечить обратную связь в управлении движением по силе, а тактильное очувствление — для определения проскальзывания во время схватывания объекта.
3. *Системы технического зрения.* Применяются для идентификации объектов и грубой оценки их положения.

На сегодняшний день нет общего соглашения, как строить команды очувствления, и каждый язык программирования роботов имеет свой собственный синтаксис. *AL* снабжен командами типа *FORCE* (ось) и *TORQUE* (ось) для силового очувствления, определяемых в командах управления как условия. Например: $FORCE(Z) > 0{,}84 * H$. *AML* оснащен оператором *MONITOR*, который вводится в команды управления для определения асинхронных событий. Программист может ввести в программу описание датчиков и управлять движением робота, используя информацию от датчиков (табл. 9.6). Также имеются

Таблица 9.6. **Силовое очувствление и податливое движение**

На языке AL:

```
{Тест на определение отверстия с помощью силового очувствления}
MOVE barm TO ⊗ —25,4*Z*мм ON FORCE(Z)<2,8*H
   DO ABORT ("NO HOLE");

{Установить болт, действуя силой вниз, пока нет сопротивления}
MOVE barm TO деталь_отверстие

WITH FORCE(Z)=—2,8*H WITH FORCE(X)=0*H
WITH FORCE(Y)=0*H WITH DURATION=3*c
```

На языке AML:

```
•Определить команду управления для датчиков силы SLP и SRP. Если
значения сигналов с датчиков выходят из интервала [0, F], mofmons = 1,
в противном случае 0:

fmons=MONITOR(<SLP, SRP>, 1,0, F);
• Переместить сочленение 3 на 25,4 мм и остановить, если fmons = 1.
DMOVE (<3>, <1>, fmons);
```

Примечание. Синтаксис команды управления:
MONITOR (датчики, тип теста, 1-й предел, 2-й предел)

операторы типа *OPOSITION*, которые дают информацию о текущем положении сочленений [1].

Большинство языков программирования роботов ориентировано на применение в системах технического зрения, и пользователь имеет возможность самостоятельно разрабатывать программные модули для обработки зрительной информации.

Одно из основных применений информации очувствления состоит в том, чтобы начать или завершить выполнение операции. Например, деталь, находящаяся на ленте конвейера, попадая в поле зрения оптического датчика может быть захвачена роботом по сигналу с этого датчика. В случае возникновения ненормальных условий функционирования выполнение команды прекращается. Пример, иллюстрирующий использование информации о силовом очувствлении для определения требуемого положения схвата относительно отверстия, приведен в табл. 9.6.

При движении руки робота вниз сила, действующая вдоль оси Z системы координат, связанной со схватом, фиксируется с помощью оператора $FORCE(Z)$. Если сила превышает 2,8 н, это означает, что схват не разместил деталь в отверстии и задача прерывается.

Обычно информация с датчиков управляет программным движением робота. Большинство языков программирования роботов оснащено конструкциями типа *if_then_else_*, *case_*, *do_until*, *while_do_* для управления программой движения робота при различных условиях. При выполнении некоторых задач требуется, чтобы движение робота удовлетворяло внешним ограничениям. Например, для установки детали в отверстие необходимо, чтобы схват двигался только в одном направлении.

Любые боковые силы могут вызывать трение, которое препятствует движению схвата с деталью. Для того чтобы осуществить податливое движение в процессе сборки, необходимо очувствление. В табл. 9.6 приведен пример, демонстрирующий применение команд силового очувствления для выполнения задачи установки детали в отверстие при наличии податливости. Податливое движение определяется величиной силы, допустимой в каждом из направлений системы координат, связанной со схватом. В данном случае внешняя сила прикладывается только вдоль оси этой системы координат.

9.2.4. Системные средства программирования

Язык без системных средств программирования (редактор, отладчик и т. д.) бесполезен для пользователя. Сложный язык должен сопровождаться соответствующим программным обес-

[1] Представляет собой агрегат вида ⟨1,5⟩, который определяет сочленения 1 и 5.

Таблица 9.7. **Программа установки болта в отверстие на языке AL**

```
BEGIN установка

{Набор переменных}
болт_диаметр ← 12,7*мм;
болт_высота ← 25,4*мм;
попытки ← 0;
схвачено ← ложь;

{Определение базовых фреймов}
деталь ← FRAME(ROT(Z, 90*град), VECTOR(508, 381, 0)*мм);
загрузочное устройство ← FRAME (nilrot, VECTOR(635, 508, 01)*мм);

{Определение основных фреймов}
болт_схват ← загрузочное устройство*TRANS(nilrot, nilvect);
болт_головка болта ← болт_схват*TRANS(nilrot, VECTOR(0, 0, 12,7)*мм);
деталь_отверстие ← деталь*TRANS(nilrot, VECTOR(0, 0, 25,4)*мм);

{Определение фреймов для промежуточных точек траектории}
A ← загрузочное устройство*TRANS(nilrot, VECTOR (0, 0, 127)*мм);
B ← загрузочное устройство*TRANS(nilrot, VECTOR(0, 0, 203,2)*мм);
C ← деталь_отверстие*TRANS(nilrot, VECTOR(0, 0, 127)*мм);
D ← деталь_отверстие*TRANS(nilrot, болт_высота*Z);

{Открытие схвата}
OPEN bhand TO болт_диаметр+25,4*мм

{Расположение схвата точно над болтом}
MOVE barm TO болт_схват VIA A
    WITH APPROACH=—Z WRT загрузочное устройство

{Попытка захвата болта}
DO
   CLOSE bhand TO 0,9*болт_диаметр:
   IF bhand <болт_диаметр THEN BEGIN
   {неудача при захвате болта, новая попытка}
   OPEN bhand TO болт_диаметр + 25,4 * мм;
   MOVE barm TO ⊗—25,4*Z*мм; END ELSE схвачено ← истина;
   попытки ← попытки+1;
UNTIL схвачено OR (попытки > 3);

{Прекращение операции, если болт не схвачен за 3 попытки}
IF NOT схвачено THEN ABORT («неудача при захвате болта»);

{Перемещать руку к точке B}
MOVE barm TO B
   VIA A
   WITH DEPARTURE=Z WRT загрузочное устройство;

{Перемещать руку к точке D}
MOVE barm TO D VIA C
   WITH APPROACH=—Z WRT деталь_отверстие;

{Проверить, есть ли отверстие}
MOVE barm TO ⊗ —2,54*Z*мм ON FORCE(Z)>2,8*H
   DO ABORT («нет отверстия»);

{Установка болта в отверстие с помощью системы силового очувствления}
MOVE barm TO деталь_отверстие DIRECTLY
   WITH FORCE(Z)=—2,8*H
   WITH FORCE(X)=0*H
   WITH FORCE(Y)=0*H
   WITH DURATON=5*1c;
END установка болта
```


печением для пользователя. Сложные программы для робота трудны как для разработки, так и для отладки. Более того, программирование робота накладывает следующие дополнительные требования на средства разработки и отладки программ:

1. *Модификация on_line и непосредственный повторный запуск.* Поскольку задачи робота требуют сложных движений и длительного времени выполнения, то не всегда можно вновь запустить программу в случае аварийного останова. Система программирования робота должна иметь возможность модификации программ в режиме *on_line* и воспроизводить повторный запуск в любое время.

2. *Выходные сигналы с датчиков и программные траектории.* Взаимодействие в реальном времени между роботом и окружающей средой не всегда воспроизводимо. Поэтому отладчик должен быть способен записывать значения сигналов с датчиков вдоль всей программной траектории.

3. *Моделирование.* Это свойство позволяет тестировать программы без физической реализации робота и рабочего пространства. Следовательно, различные программы будут отлаживаться с большей эффективностью.

Читатель должен понять, что программирование на роботоориентированном языке утомительно и громоздко. Это иллюстрируется следующим примером.

Пример. В табл. 9.7 приводится программа на языке *AL* для выполнения операции установки болта в отверстие (рис. 9.1). Необходимые разъяснения даны в предыдущих разделах. Примите во внимание, что предложение на языке *AL* не считается завершенным до тех пор, пока не встретится точка с запятой.

9.3. ХАРАКТЕРИСТИКИ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ

Совершенно другой подход к программированию роботов применяется при использовании проблемно-ориентированных языков. В задачах сборки более естественно описывать не движения робота, а объекты, которыми робот манипулирует. Этот факт лежит в основе программирования на проблемно-ориентированных языках, применение которых упрощает задачу сборки.

Система программирования на уровне задачи позволяет пользователю описать задачу на языке высокого уровня (описание задачи). Затем планировщик задачи будет обращаться к базе данных (моделям рабочего пространства) и преобразовывать описание задачи в программу на уровне робота (синтез программы для робота), которая и будет управлять ходом выполнения задачи сборки.

Основываясь на этом описании, можно мысленно разделить планирование задачи на три этапа: моделирование рабочего пространства, описание задачи и синтез программы. Следует отметить, что эти три этапа не являются полностью независимыми; фактически они связаны между собой на уровне вычислений.

С помощью декомпозировщика [1] исходное описание задачи раскладывается на последовательность подзадач и выделяется такая информация, как первоначальное и исходное состояния,

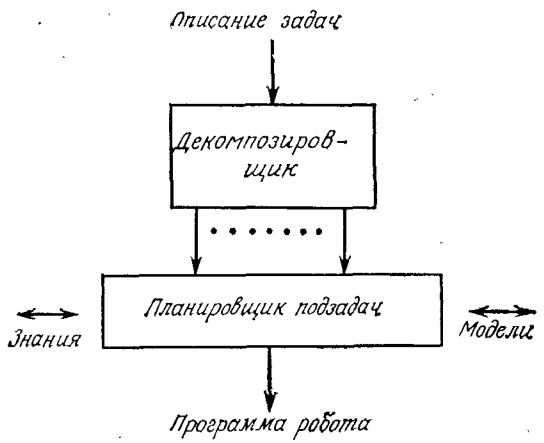


Рис. 9.4. Планировщик задач.

положения схватывания, связи закрепления. Затем подзадачи поступают на планировщик подзадач, который генерирует требуемую программу для управления движением робота (рис. 9.4).

Концепция планирования задачи до некоторой степени подобна идее автоматического создания программ в системе искусственного интеллекта. Пользователь задает требования на вход и выход желаемой программы, после чего генератор программы создает программу, удовлетворяющую требованиям пользователя на вход и выход [13]. Программирование на уровне задачи, подобное автоматической генерации программ, находится на стадии исследования, и многие вопросы до сих пор остаются нерешенными. В последующих разделах мы обсудим проблемы, возникающие при планировании задачи, и возможные пути их решения.

[1] Под декомпозировщиком подразумевается программный модуль, предназначенный для разбиения исходной задачи на подзадачи. — *Прим. перев.*

9.3.1. Моделирование рабочего пространства

Для описания геометрических и физических свойств объектов, в том числе робота и представления процесса сборки в рабочем пространстве, проводится предварительное моделирование.

Геометрические и физические модели. Для того чтобы планировщик мог генерировать программу, предназначенную для выполнения роботом данной задачи, он должен иметь информацию об объектах и самом роботе. Она содержит геометрические и физические свойства объектов, которые представляются соответствующими моделями. Геометрическая модель дает информацию о размерах, объемах и формах объектов в рабочем пространстве. Как уже говорилось в гл. 8, для моделирования трехмерных объектов разработаны соответствующие численные методы [8, 245]. Для создания моделей объектов обычно применяется подход, называемый «пространственная геометрия воспроизведения объектов». При этом модели объектов рабочего пространства с помощью набора упорядоченных операций типа объединения и пересечения представляются в виде совокупности элементарных объектов, таких, как куб, цилиндр и т. д. Элементарные объекты можно реализовать различными способами, например:

1) набором контуров;
2) набором поверхностей;
3) телами вращения;
4) сеточной структурой.

В системе *AUTOPASS* [171] объекты моделируются с помощью процессора геометрического конструирования (ПГК) [306], в котором реализованы определенные процедуры для описания объектов. Основная идея состоит в том, что каждому объекту соответствует название процедуры и ряд параметров. Внутри такой процедуры форма объекта устанавливается с помощью обращений к другим процедурам, представляющим различные объекты, или к допустимым операциям. В системе ПГК заложен ряд элементарных объектов: куб, цилиндр, конус, клин, полусфера, пластина и тела вращения. В свою очередь элементарные объекты представляются в виде списка поверхностей, контуров и точек, которые определяются параметрами в соответствующей процедуре. Например, обращение

CALL SOLID(*CUBOUD*, «*Block*», *xlen*, *ylen*, *zlen*);

вызовет процедуру *SOLID*, с помощью которой определяется прямоугольный ящик, называемый *Block*, с размерами *xlen*, *ylen* и *zlen*. Более сложные объекты можно определить путем обращения к другим процедурам и применением к ним процедуры *MERGE*. Описание операции установки болта в отверстие, приведенной в разд. 9.2, дано в табл. 9.8.

Таблица 9.8. **Описание болта на языке системы ПГК**

```
Болт: PROCEDURE (стержень_высота, стержень_радиус, стержень_грани,
      головка_высота, головка_радиус, головка_грани);

/*определение параметров*/
DECLARE

    стержень_высота    /*высота стержня*/
    головка_высота     /*высота головки*/
    стержень_радиус    /*радиус стержня*/
    головка_радиус     /*радиус головки*/
    стержень_грани     /*число граней стержня*/
    головка_грани      /*число граней головки*/

/*описание вышеупомянутых переменных как переменных с плавающей точкой*/
FLOAT;

/*определение формы стержня*/
CALL SOLID(CYLIND, «стержень», стержень_высота, стержень_радиус, стержень_грани);

/*определение формы головки*/
CALL SOLID(CYLIND, «головка», головка_высота, головка_радиус, головка_грани);

/*формирование описания болта*/
CALL MERGE(«стержень», «головка», «объединение»);

END Болт
```

Примечание. Выражения в /*...*/ являются комментариями.

Физические свойства тела, такие, как инерция, масса и коэффициенты трения, могут ограничивать тип движений, осуществляемых роботом. Эти свойства не требуется хранить в памяти, поскольку они выводятся из модели объекта. Так как ни одна модель не является точной на 100 %, то, следовательно, тождественные части модели и объекта имеют незначительные различия в физических свойствах. Поэтому в модель необходимо вводить допуски [246].

Представление состояний рабочего пространства. Для того чтобы генерировать программу робота, планировщик задач должен быть способен сформулировать все этапы сборки. Каждый этап сборки можно кратко представить текущим состоянием рабочего пространства. Один из путей представления этих состояний заключается в описании взаимодействия всех объектов в рабочем пространстве в каждый момент времени. В языке *AL* предусмотрен оператор связи *AFFIX*, который позволяет присоединять фреймы друг к другу. Это эквивалентно физическому присоединению одной детали к другой, и если одна из деталей движется, то присоединенная деталь также будет двигаться.

Язык *AL* позволяет автоматически изменять расположение систем координат, проводя соответствующие преобразования.

Например:

AFFIX деталь_отверстие TO деталь RIGIDLY;
деталь_отверстие- FRAME(*nilrot*, *VECTOR*(1, 0, 0)*мм);

означает, что система координат деталь_отверстие присоединяется к системе координат, связанной с деталью.

В системе *AUTOPASS* для представления состояний рабочего пространства применяется граф. Вершины графа обозначают объекты, а дуги — связи между ними. Связи могут быть следующих видов:

1. *Соединение.* Объект может быть закреплен жесткой связью, присоединен гибкой связью или условно соединен с другим объектом. Первым двум видам соединений соответствует оператор *AFFIX* в языке *AL*. Условное соединение означает, что объект удерживается силой тяжести, но не строго закреплен.

2. *Ограничение.* Связи типа ограничения представляют собой физические ограничения между объектами, которые могут двигаться поступательно или совершать вращательные движения.

3. *Компонент сборки.* Он используется, чтобы указать, что подграф, связанный дугой этого типа, является этапом сборки и может быть рассмотрен в качестве объекта. В процессе сборки граф изменяется, чтобы отражать текущее состояние сборки во времени.

9.3.2. Описание задачи сборки

Описание задачи сборки дается на языке высокого уровня. Наилучшим вариантом для пользователя было бы общение с системой на естественном языке, что привело бы к исключению описания этапов сборки. Тогда задача сборки водяного насоса могла бы быть определена командой «собрать водяной насос». Однако до такого уровня еще очень далеко и пока нельзя обойтись без подробного описания последовательности сборки.

Современный подход представляет собой использование языка программирования с хорошо разработанным синтаксисом и семантикой, что облегчает пользователю подробно описывать последовательность сборки. Задачу сборки можно представить в виде последовательности состояний модели рабочего пространства, которые определяются взаимным расположением всех объектов в рабочем пространстве.

Один из путей описания взаимного расположения объектов заключается в использовании пространственных связей между этими объектами. Например, рассмотрим рабочее пространство состояний из блоков, показанное на рис. 9.5. Чтобы указать, что две поверхности касаются друг друга, введем пространственную связь *AGAINST*. Тогда для описания двух ситуаций, изображенных на рис. 9.5, могут быть использованы выражения из

табл. 9.9. Если мы предположим, что состояние A — начальное и состояние B — конечное, тогда их можно использовать для описания задачи подъема блока 3 и размещения его на верхней грани блока 2. Если же состояние A — конечное и состояние B — начальное, мы имеем задачу съема блока 3 и размещения его на столе. Преимущество использования этого типа представления заключается в том, что состояния рабочего пространства легко интерпретируются человеком и, более того, легко определяются и модифицируются. Существенным ограничением этого

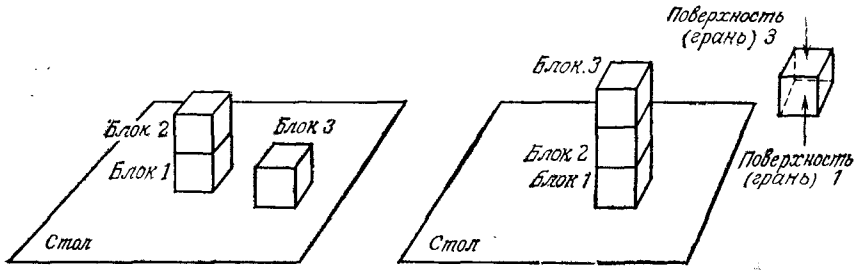


Рис. 9.5. Рабочее пространство из блоков.

метода является то, что он не дает всю необходимую информацию для описания операции. Например, вращающий момент, требуемый для затягивания болта, не может быть включен в описание состояния.

Таблица 9.9. **Описание состояний рабочего пространства из модельных блоков**

Состояние A
(*блок 1_грань 1* AGAINST *стол*)
(*блок 1_грань 3* AGAINST *блок 2_грань 1*)
(*блок 3_грань 1* AGAINST *стол*)
Состояние B
(*блок 1_грань 1* AGAINST *стол*)
(*блок 1_грань 3* AGAINST *блок 2_грань 1*)
(*блок 2_грань 3* AGAINST *блок 3_грань 1*)

Альтернативный подход заключается в описании задачи в виде последовательности символических операций на объектах. Обычно также имеется ряд пространственных ограничений на объекты, устраняющих любую неопределенность. Эта форма описания подобна форме, используемой в схеме промышленной сборки. Большинство языков, ориентированных на роботы, приспособлено к этому типу описания.

Язык *AL* обладает ограниченными возможностями описания задачи с помощью этого метода. С помощью оператора *AFFIX* объектный фрейм может быть присоединен к идентификатору

barm для указания того, что рука робота удерживает объект. В этом случае движение объекта к другой точке представляется движением системы координат, связанной не с рукой, а с объектом. Например, процесс установки болта в отверстие (рис. 9.1) определяется следующим образом:

AFFIX болт_головка болта TO рука;
MOVE болт_головка болта TO деталь_отверстие;

В работе [238] предложен язык, называемый *RAPT*, в котором применяются операторы *AGAINST*, *FIT* и *COPLANAR* для описания связи между свойствами объектов, которыми могут быть: плоские или сферические поверхности, цилиндрические стержни и отверстия, контуры и вершины. Эти свойства определяются с помощью фреймов, подобных тем, которые используются в языке *AL*. Например, две операции в рабочей зоне с блоками (рис. 9.5) могут быть описаны следующим образом:

PLACE блок 3 SO THAT (блок 2_ грань 3 AGAINST блок 3_ грань 1)
PLACE блок 3 SO THAT (блок 3_ грань 1 AGAINST стол)

Таблица 9.10. **Определение изменения состояний рабочего пространства и команды для описания инструмента в системе AUTOPASS**

Определение изменений состояний рабочего пространства

PLACE ⟨объект⟩, ⟨предлог⟩ ⟨схватывание⟩ ⟨конечное состояние⟩ ⟨ограничения⟩ ⟨тогда-держать⟩

где

⟨объект⟩	— символическое имя объекта
⟨предлог⟩	— либо "IN", либо "ON" используется для определения типа операции
⟨схватывание⟩	— определяет способ захвата объекта
⟨ограничение⟩	— определяет ограничения
⟨тогда-держать⟩	— указывает, что схват должен оставаться в прежнем положении после завершения команды.

Команды для описания инструмента

OPERATE ⟨инструмент⟩ ⟨загрузить-список⟩ ⟨в положении⟩ ⟨приспособление⟩ ⟨конечное состояние⟩ ⟨инструмент-параметры⟩ ⟨тогда-держать⟩

где

⟨инструмент⟩	— определяет инструмент, который должен быть использован
⟨загрузить-список⟩	— определяет список вспомогательных принадлежностей
⟨в положении⟩	— определяет рабочее положение инструмента
⟨приспособление⟩	— определяет новое приспособление
⟨конечное-состояние⟩	— определяет конечное состояние, которое должно быть достигнуто по завершении команды
⟨инструмент-параметры⟩	— определяет рабочие параметры инструмента, такие, как скорость и направление вращения
⟨тогда-держать⟩	— означает, что рука должна оставаться в прежнем положении после завершения команды

Затем выделяются пространственные связи для определения ограничений на взаимное расположение объектов, которые требуются для выполнения задачи сборки.

В системе *AUTOPASS*, имеющей более совершенный синтаксис, также используется этот тип описания, и команды, связанные со сборкой, делятся на три группы:

1. *Команды изменения состояния.* Описывают операцию сборки как перемещение и установку деталей.

2. *Команды описания инструмента.* Описывают тип инструмента, который необходимо использовать.

3. *Команды закрепления.* Описывают операции закрепления деталей. Синтаксис этих команд более сложный (табл. 9.10). Например, для описания операции установления болта в отверстие и его завинчивания использовано следующее предложение:

PLACE болт ON деталь SUCH THAT болт_головка болта IS ALIGNED WITH деталь_отверстие;

DRIVE IN болт_ AT болт_схват SUCH THAT TORQUE IS EQ 12.0 IN-LSB USING воздух_устройство для завинчивания;

9.3.3. Синтез программы

Синтез программы для управления движением робота на основе описания задачи является одним из наиболее важных и трудных этапов планирования задачи сборки. Этот этап включает следующие действия: планирование захвата детали, планирование движения и проверка плана.

Прежде чем планировщик задачи сможет осуществить планирование, он должен преобразовать символическое описание задачи в приемлемую форму. Один из путей — получение ограничений на взаимное расположение объектов из символических соотношений. Интерпретатор *RAPT* выделяет символические соотношения и формирует ряд матричных уравнений с неизвестными параметрами связей между объектами. Эти уравнения затем упрощаются с помощью набора специальных правил преобразования. Полученный результат представляет собой набор ограничений на форму и взаимное расположение объектов, необходимый для выполнения операции.

Возможно, что планирование захвата детали является наиболее важной проблемой при планировании задачи сборки, поскольку траектория движения руки робота при захвате детали влияет на все последующие операции. Траектория, по которой робот производит захват объекта, зависит от геометрии захватываемого объекта и наличия других объектов в рабочем пространстве. Обычно существует единственное положение захвата детали, которое достижимо и устойчиво. Робот должен быть способен захватить намеченный объект, не сталкиваясь с другими объектами в рабочем пространстве, и в свою очередь захваченный объект должен надежно удерживаться в схвате во время последующих движений робота.

Обычно способ захвата детали и соответствующая ему конфигурация манипулятора выбираются на основе следующих процедур:

1. Возможные способы захвата детали выбираются исходя из геометрии объекта (так, например, для схвата с параллельным движением пальцев место, удобное для захвата, расположено параллельно направлению движения пальцев со стороны их внутренних или внешних поверхностей, устойчивости (одна из возможностей захвата предусматривает, чтобы центр масс объекта находился между пальцами) и уменьшения неопределенности.

2. Затем число возможных способов захвата детали сокращается в зависимости от того, могут ли они быть реализованы роботом или привести к столкновению с другими объектами.

3. Окончательная конфигурация манипулятора при захвате выбирается среди оставшихся (если таковые имеются) так, чтобы она привела к наиболее устойчивому захватыванию и вероятность столкновения с препятствиями была минимальной при наличии ошибок позиционирования.

Большинство современных методов для планирования захвата детали направлено на нахождение достижимых роботом положений захватывания, причем не все ограничения на взаимное расположение объектов принимаются во внимание. Поскольку при наличии неопределенностей захватывание наиболее трудно осуществимо, то в этом случае часто используется очувствление.

После того как объект схвачен, робот должен его перенести в требуемое место рабочего пространства и затем выполнить операцию. Это движение может быть разделено на 4 этапа:

1. Отход с предосторожностями от текущего положения.

2. Движение, свободное от столкновений, к желаемому положению.

3. Подход с предосторожностями к месту выполнения операции.

4. Податливое движение для достижения конечного положения. При этом одной из важных проблем является планирование движения робота без столкновений с объектами рабочего пространства. Было предложено несколько алгоритмов для планирования траектории, свободной от столкновений, которые могут быть объединены в 3 класса:

1. *Гипотеза и тест.* В этом методе выбирается возможная траектория и проверяется на столкновения на ряде выбранных взаимных расположений объектов и робота. Если происходит столкновение, вводится коррекция для обхода препятствий [170]. Основным преимуществом данного метода является его

простота и то, что имеются достаточно эффективные системы геометрического моделирования. Однако расчет коррекции бывает затруднителен, когда рабочее пространство заполнено препятствиями.

2. *Штрафные функции.* Это метод построения штрафных функций, значения которых зависят от близости препятствий. Штрафные функции обладают тем свойством, что их значения возрастают при приближении робота к препятствиям. Полная штрафная функция является суммой всех отдельных штрафных функций, причем иногда имеется член, характеризующий степень близости к оптимальной траектории. Затем берутся производные полной штрафной функции по параметрам взаимного расположения объектов. Найдя локальный минимум полной штрафной функции, можно определить траекторию, свободную от столкновения с препятствиями. Метод имеет то преимущество, что позволяет учитывать все препятствия и ограничения. Однако сами штрафные функции довольно трудно определять.

3. *Алгоритмы свободного пространства.* Было предложено несколько алгоритмов этого типа. В работе [174] рассматривается представление свободного пространства (пространства, свободного от препятствий) в терминах конфигураций робота (конфигурационное пространство). Эта идея эквивалентна преобразованию руки робота, держащей объект, в точку и соответственно увеличению числа препятствий в рабочем пространстве. Тогда нахождение траектории, свободной от столкновений, равнозначно нахождению траектории, которая не пересекает ни одно из препятствий. Этот алгоритм работает достаточно хорошо, когда рассматривается только поступательное движение. Для вращательного движения должны быть сделаны приближения, чтобы определить конфигурационное пространство, а это требует существенного увеличения объема вычислений.

В работах [34] и [35] предложен другой метод, в котором свободное пространство представляется в виде пересечения обобщенных конусов, а объем, охватываемый движущимся объектом, функцией их ориентации. Тогда траекторию, свободную от столкновений, можно определить путем сравнения объема, охватываемого объектом, с объемом свободного пространства.

Другой трудной и важной проблемой является генерация податливого движения. В настоящее время ведутся исследования, направленные на то, чтобы допустимые конфигурации робота лежали на С-поверхности [193] [1] в пространстве реализаций различных конфигураций робота. Тогда осуществление податливых движений эквивалентно нахождению комбинированной стратегии управления по положению/силе для того, чтобы траектория робота оставалась на С-поверхности.

[1] С-поверхность определяется в С-системе координат. Она представляет собой конфигурацию робота, допускающую движение только по некоторым степеням свободы. Вдоль С-поверхности реализуются степени свободы по положению, а перпендикулярно к ней — по силе. — С-система координат представляет собой ортогональную систему координат в декартовом пространстве. Система координат выбрана так, чтобы поступательное движение робота осуществлялось вдоль главных осей, а вращательное движение вокруг них,

9.4. ЗАКЛЮЧЕНИЕ

Мы рассмотрели характеристики языков, ориентированных на роботы, и проблемно-ориентированных языков. В роботоориентированных языках задача сборки явно описывается как последовательность движений робота. Программист управляет движением робота в ходе выполнения всей задачи, причем каждый шаг программы примерно соответствует одному действию робота.

Проблемно-ориентированные языки программирования позволяют представлять задачу сборки как последовательность промежуточных положений объектов, а не как последовательность движений робота, необходимую для достижения этих положений. Таким образом, в этом случае движение робота задается неявно. В качестве примеров роботоориентированных языков были приведены языки программирования *AL* и *AML*.

Мы считаем, что роботоориентированный язык программирования труден для практической эксплуатации, поскольку пользователь вынужден подробно программировать каждое движение робота. Проблемно-ориентированные языки намного легче использовать. Однако многие сложности, возникающие при планировании задачи, моделировании объекта, обходе препятствий, планировании траектории, использовании информации очувствления и конфигурации схватывания должны быть решены, прежде чем проблемно-ориентированные языки смогут эффективно применяться.

В заключение приведем таблицу сравнения различных языков (табл. 9.11, *а* и *б*).

Литература

Подробные сведения о робото-ориентированном программировании можно найти в работах [27, 93, 108, 175, 219, 225, 227, 229, 238, 239, 263, 270, 277, 283]. Информация о программировании на уровне задачи (проблемно-ориентированное программирование) содержится также в работах [23, 52, 80, 169, 198]. Языки программирования для описания объектов приводятся в работах [13, 107, 171, 306]. В работе [277] приводится матричное уравнение однородного преобразования для описания последовательности выполнения задачи манипулятором.

Различные алгоритмы обхода препятствий, реализованные в языках программирования в виде модулей, приведены в ра-

Таблица 9.11, а. **Сравнение различных языков программированиия роботов**

Язык	AL	AML	AUTOPASS	HELP	JARS	MAPLE
Институт	Станфорд	ИБМ	ИБМ	GE	JPL	ИБМ
Робот	Пума, станфордский манипулятор	Манипулятор ИБМ	Манипулятор ИБМ	Аллегро	Пума, станфордский манипулятор	Манипулятор ИБМ
Робото- или проблемноориентированный	Робото- и проблемноориентированный	Роботоориентированный	Проблемно-ориентированный	Роботоориентированный	Роботоориентированный	Роботоориентированный
Базис языка	Concurrent Pascal	Лисп, АПЛ, Паскаль	ПЛ/1	Паскаль	Паскаль	ПЛ/1
Компилятор или интерпретатор	И то и другое	Интерпретатор	И то и другое	Интерпретатор	Компилятор	Интерпретатор
Тип представления данных	Фрейм	Агрегат	Модель	Нет	Фрейм	Нет
Описание движения	Фрейм	Сочленение	Неявное описание	Сочленение	Сочленение, фрейм	Перемещение, вращение
Структура управления	Паскаль	Паскаль	ПЛ/1	Паскаль	Паскаль	ПЛ/1
Команды очувствления	Положение, сила	Положение	Сила, тактильная информация	Сила, зрение	Близость, зрение	Сила, близость
Параллельная обработка	COBEGIN семафоры	Нет	IN PARALLEL	Семафоры	Нет	IN PARALLEL
Многосуставный робот	Да	Нет	Нет	Да	Нет	Да
Литература	[200]	[283]	[171]	[325]	[48]	[52]

Таблица 9.11, б. **Сравнение различных языков программирования роботов**

Язык	MCL	PAL	RAIL	RPL	VAL
Институт	Mc. Donnell Douglas	Purdue	Automatix	SRI	Unimate
Робот	Цинциннати Милакрон Т3	Станфордский манипулятор	Манипулятор, проектируемый по заказу	Пума	Пума
Робото- или проблемноориентированный	Роботоориентированный	Роботоориентированный	Роботоориентированный	Роботоориентированный	Роботоориентированный
Базис языка	APT	Изменяющийся базис	Паскаль	Фортран, Лисп	Бейсик
Компилятор или интерпретатор	Компилятор	Интерпретатор	Интерпретатор	И то и другое	Интерпретатор
Тип представления данных	Фрейм	Фрейм	Фрейм	Нет	Фрейм
Описание движения	Перемещение, вращение	Фрейм	Сочленения, фрейм	Сочленения	Сочленения, фрейм
Структура управления	if-then-else while-do	if-then-else	Паскаль	Фортран	if-then
Команды очувствления	Положение	Сила	Сила, зрение	Положение, зрение	Положение, сила
Параллельная обработка	INPAR	Нет	Нет	Нет	Семафоры
Многосуставный робот	Да	Нет	Нет	Нет	Нет
Литература	[219]	[277]	[81]	[225]	[296]

ботах [34, 35, 36, 170, 170, 175, 177]. Для обхода препятствий в рабочем пространстве, заполненном различными объектами, в работах [174 и 176] предложены методы планирования задачи в пространстве конфигураций робота.

Будущие языки программирования роботов должны содержать методы искусственного интеллекта [13] и использовать знания [33] для планирования выполнения роботизированной сборки и других технологических задач.

Упражнения

9.1. Напишите программу на языке *AL* для определения системы координат схвата, которая может быть получена вращением системы координат «блок» на угол 65° вокруг оси y и затем перемещением его на 101,6 и 152,4 мм вдоль осей x и y соответственно.

9.2. Напишите программу для решения задачи упр. 9.1 на языке *AML*.

9.3. Напишите программу на языке *AL* для складирования роботом 9 деталей из загрузочного устройства в палету, представляющую собой матрицу из 3×3 ячеек. Предполагается, что расположение загрузочного устройства и палеты известны. Программа должна фиксировать заполнение каждой ячейки и сигнализировать пользователю о заполнении всей палеты.

9.4. Напишите программу на языке *AML* для решения задачи упр. 9.3.

9.5. Напишите программу на языке *VAL* для решения задачи упр. 9.3.

9.6. Напишите программу на языке системы *AUTOPASS* для решения задачи упр. 9.3.

9.7. Ханойская башня. Три стержня A, B, C, системы координат которых соответственно (x_A, y_A, z_A), (x_B, y_B, z_B) и (x_C, y_C, z_C) имеют известные расположения относительно неподвижной системы координат (x_0, y_0, z_0), как показано ниже на рис. 9.6. Первоначально, на штифте A находятся два диска

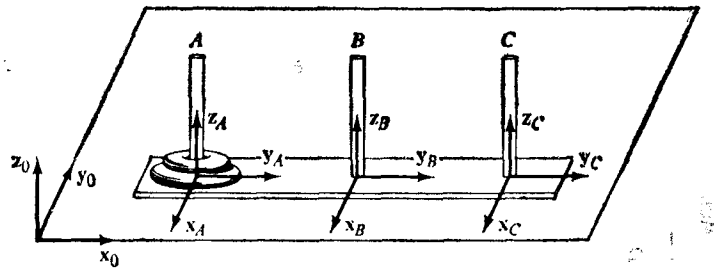


Рис. 9.6.

разных размеров, причем диск меньшего диаметра всегда должен располагаться на диске большего диаметра. Требуется написать программу на языке *AI* для управления роботом, оснащенного специальным схватом присоской (для захватывания диска), чтобы двигать два диска от стержня A к стержню C; причем в любой момент времени диск меньшего диаметра должен всегда располагаться на диске большего диаметра. Каждый диск имеет одинаковую толщину 25,4 мм.

9.8. Напишите программу на языке *AML* для решения задачи в упр. 9.7.

Глава 10.
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ПЛАНИРОВАНИЕ ЗАДАЧ В РОБОТОТЕХНИКЕ

То, что воспринято разумом, — незыблемо, а то, что постигается только чувствами, — всегда зыбко и неопределенно.

Из «Диалогов» Платона

10.1. ВВЕДЕНИЕ

Одной из основных проблем робототехники является *планирование* движения для решения поставленной задачи с последующим *контролем* выполнения роботом команд, необходимых для осуществления этих действий. Под планированием здесь подразумевается принятие решения о ходе действия перед его выполнением.

Действия робота могут быть определены системой планирования действий робота, которая будет искать пути достижения поставленной цели, исходя из некоторой первоначальной ситуации. В этом случае необходим план, представляющий собой последовательность действий робота для достижения цели.

Исследования процесса построения решений задач робототехники привели ко многим идеям относительно системы построения решения задач в области искусственного интеллекта. В обычной постановке задачи мы имеем робот, оснащенный датчиками и способный выполнять элементарные действия в несложном рабочем пространстве. Действия робота преобразуют одно состояние (или конфигурацию) рабочего пространства в другое. Например, в рабочем пространстве несколько объектов могут располагаться на столе или друг на друге, а робот, состоящий из телевизионной камеры, манипулятора и схвата, поднимает или передвигает эти объекты. В некоторых задачах робототехники робот представляет собой подвижную тележку с телевизионной камерой, выполняющую такие операции, как, например, перемещение объектов в рабочем пространстве. В этой главе мы кратко изложим несколько методов принятия решений и их применение к планированию действий робота.

10.2. ПОИСК В ПРОСТРАНСТВЕ СОСТОЯНИЙ

Один из методов решения задачи состоит в последовательной проверке различных возможных вариантов до тех пор, пока мы случайно не получим желаемое решение. Такая попытка

включает поиск типа «проб и ошибок». Для обсуждения методов решения этого типа введем понятие состояний задачи (проблемных состояний) и операторов. *Проблемным состоянием*, или просто *состоянием*, является конкретная конфигурация робота. Набор всех возможных конфигураций образует пространство проблемных состояний или *пространство состояний*. Оператор, применяемый к состоянию, преобразует его в другое состояние. Решение поставленной задачи представляет собой последовательность операторов, которые преобразуют начальное состояние в требуемое конечное состояние.

Пространство состояний, достижимых из начального, удобно представлять в виде графа, вершины которого соответствуют состояниям, а дуги — операторам. Решение задачи можно получить, применяя операторы к начальному состоянию и получая в результате новые состояния. В полученным новым состояниям в свою очередь применяются операторы до тех пор, пока не будет достигнуто целевое состояние. Методы организации такого поиска для определения целевого состояния обычно описываются в терминах теории графов.

10.2.1. Вводные примеры

Прежде чем приступить к обсуждению методов поиска на графе, рассмотрим кратко несколько базовых примеров, чтобы ознакомить читателя с понятиями, применяемыми в этой главе.

Рабочее пространство из модельных объектов. Будем считать, что рабочее пространство состоит из поверхности T

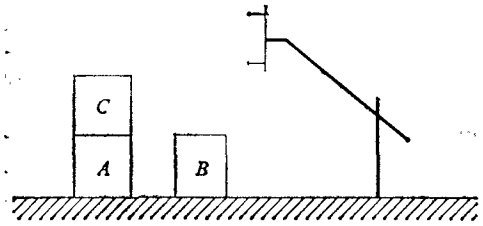


Рис. 10.1. Взаимное расположение робота и модельных объектов.

и трех объектов A, B и C. Начальное состояние рабочего пространства следующее: объекты A и B находятся на поверхности, а объект C расположен на верхней грани объекта A (рис. 10.1). Робот должен преобразовать начальное состояние в целевое состояние, в котором объекты расположены друг на друге следующим образом: объект A наверху, объект B в середине и объект C внизу. Робот может использовать единственный оператор MOVE (двигать) X от Y к Z, который перемещает объект X с вершины объекта Y на объект Z. Для применения оператора требуется 1) чтобы перемещаемый объект X

был объектом, на вершине которого ничего нет; 2) если Z — объект, на нем ничего не должно быть расположено.

Нетрудно использовать граф вместо рис. 10.1 для описания возможных состояний. Оператор MOVE X от Y к Z имеет вид MOVE(X, Y, Z). Граф поиска в пространстве состояний приведен на рис. 10.2. Если мы устраним в графе штриховые линии (предназначенные для того, чтобы оператор не мог быть использован для выполнения одной и той же операции более одного раза), мы получим дерево поиска в пространстве со-

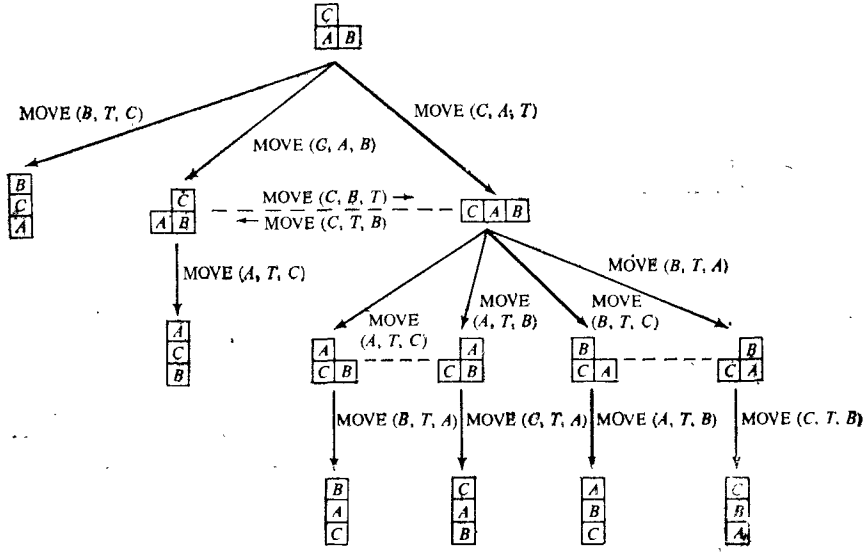


Рис. 10.2. Граф поиска в пространстве состояний.

стояний. Из рис. 10.2 видно, что решение задачи состоит из следующей последовательности операторов: MOVE(C, A, T), MOVE(B, T, C), MOVE(A, T, B).

Выбор пути. Предположим, что мы хотим перемещать длинный и тонкий объект A через заполненное препятствиями рабочее пространство (рис. 10.3). Для планирования движений объекта, находящегося в схвате робота, мы можем выбрать представление пространства состояний в виде тройки (x, y, α), где x — горизонтальная координата объекта: $1 \leqslant x \leqslant 5$; y — вертикальная координата объекта: $1 \leqslant y \leqslant 3$; α — ориентация объекта:

$$\alpha = \begin{cases} 0, & \text{если объект } A \text{ параллелен оси } x, \\ 1, & \text{если объект } A \text{ параллелен оси } y. \end{cases}$$

Положение и ориентация объекта являются дискретными величинами. Операторы или команды робота следующие:

MOVE в направлении $\pm x$ на одну единицу,

MOVE в направлении $\pm y$ на одну единицу,

ROTATE 90° (повернуть на 90°).

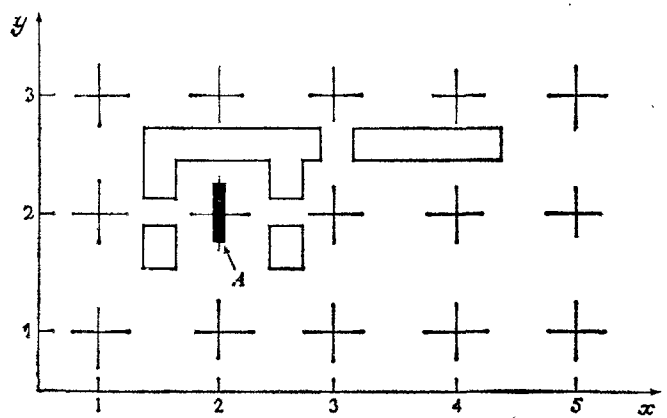


Рис. 10.3. Рабочее пространство.

Пространство состояний показано на рис. 10.4. Мы предполагаем, что каждое «поступательное движение» имеет длину 2 и «вращательное движение» имеет длину 3. Пусть объект A

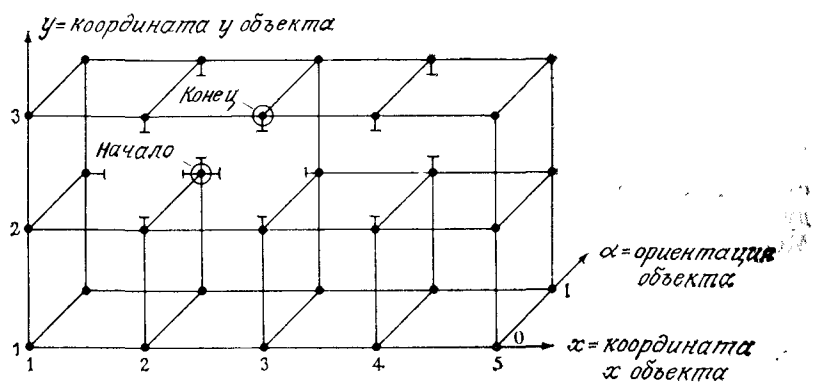


Рис. 10.4. Пространство состояний.

первоначально находится в точке (2, 2) и ориентирован параллельно оси y. Задача состоит в том, чтобы передвинуть объект A в точку (3, 3) так, чтобы он был ориентирован параллельно оси x. Тогда начальное состояние будет (2, 2, 1), а целевое — (3, 3, 0).

Два пути равной длины, приводящих к решению, показаны на рис. 10.5 и 10.6. На первый взгляд эти два пути не кажутся наиболее короткими. Однако более подробное исследование показывает, что если передвигать вдоль них объект из целевого состояния, то можно избежать двух вращений за счет несколько большего расстояния.

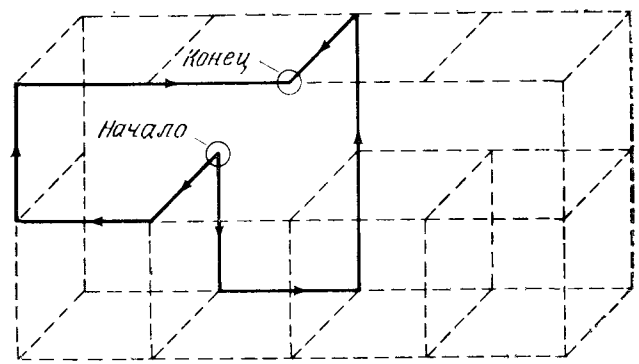


Рис. 10.5.

Задача «Обезьяна и бананы». В комнате находятся обезьяна [1], ящик и гроздь бананов (рис. 10.7). Бананы подвешены под потолком вне досягаемости обезьяны. Каким образом обезьяна может достать бананы?

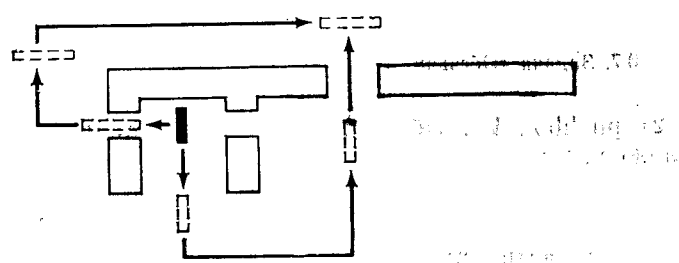


Рис. 10.6.

Для описания состояния воспользуемся списком из 4-х элементов (W, x, Y, z), где

W — горизонтальное положение обезьяны;

$x = 1$ или 0 в зависимости от того, находится обезьяна на вершине ящика или нет соответственно;

Y — горизонтальное положение ящика;

[1] Подразумевается, что обезьяной может быть мобильный робот.

$z = 1$ или 0 в зависимости от того, достала обезьяна бананы или нет соответственно.

Операторы для решения этой задачи — следующие:

1) goto(U). Обезьяна двигается к горизонтальному положению U, или в форме производящего правила,

$$(W,\ 0,\ Y,\ z) \xrightarrow{\text{goto }(U)} (U,\ 0,\ Y,\ z).$$

Таким образом, состояние $(W, 0, Y, z)$ может быть преобразовано в состояние $(U, 0, Y, z)$ с помощью оператора goto(U).

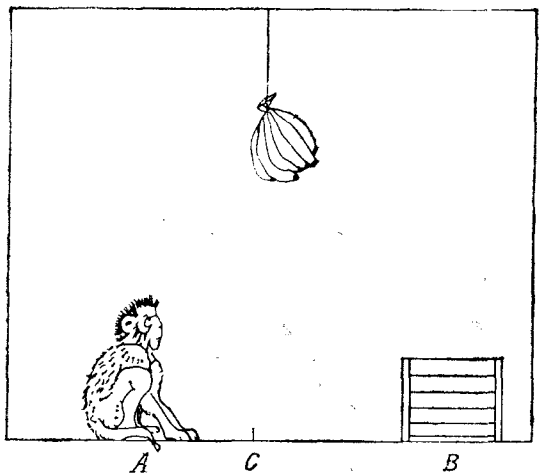


Рис. 10.7. Задача «Обезьяна и бананы».

2) pushbox(V). Обезьяна двигает ящик к горизонтальному положению V, или

$$(W,\ 0,\ W,\ z) \xrightarrow{\text{pushbox }(V)} (V,\ 0,\ V,\ z).$$

Из левой части производящего правила следует, что оператор pushbox(V) может быть применен, если обезьяна расположена в том же положении W, что и ящик, но не на ящике. Такое условие, наложенное на область применения оператора, называется *предусловием* (предпосылкой) производяшего правила.

3) climbbox. Обезьяна залезает на ящик, или

$$(W,\ 0,\ W,\ z) \xrightarrow{\text{climbbox}} (W,\ 1,\ W,\ z).$$

Необходимо отметить, что для применения оператора climbbox обезьяна должна быть в том же положении W, что и ящик, но не на ящике.

4) grasp. Обезьяна захватывает бананы, или

$$(C,\ 1,\ C,\ 0) \xrightarrow{\text{grasp}} (C,\ 1,\ C,\ 1),$$

где C — положение на полу непосредственно под бананами. Следует отметить, что для применения оператора grasp обезьяна и ящик должны быть в положении C, причем обезьяна предварительно должна находиться на ящике.

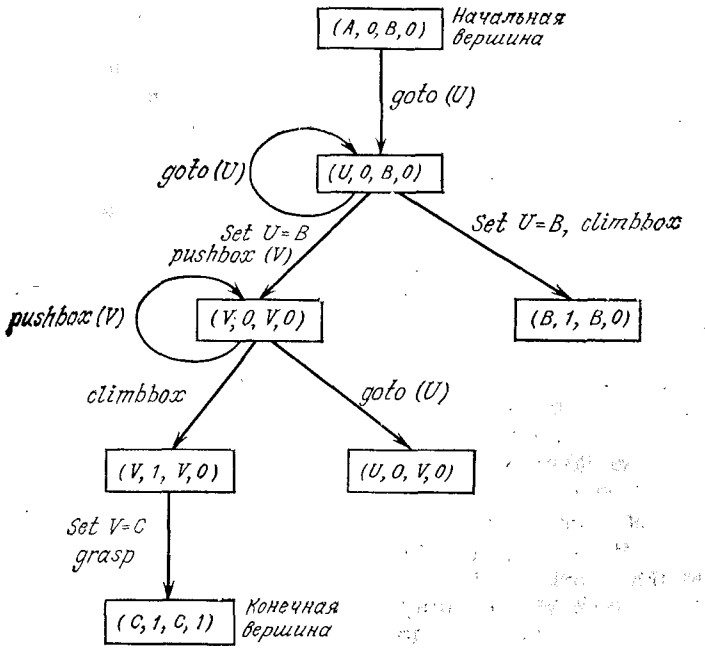


Рис. 10.8. Представление задачи «Обезьяна и бананы» в виде графа.

Подразумевается, что как область допустимых значений, так и результаты действия операторов описываются с помощью производящих правил. Например, в правиле 2 оператор pushbox(V) применяется только тогда, когда выполнено его предусловие. В результате действия этого оператора обезьяна передвинула ящик в положение V. При такой постановке ряд целевых состояний описывается одним списком, последний элемент которого есть 1.

Пусть начальное состояние $(A, 0, B, 0)$. Единственным оператором, который можно применить, является goto(U). Он приводит к следующему состоянию $(U, 0, B, 0)$. Теперь применимы 3 оператора: goto(U), pushbox(V) и climbbox (если $U = B$). Продолжая применять все возможные операторы к каждому состоянию, мы создадим граф пространства состояний, показан-

ный на рис. 10.8. Нетрудно видеть, что последовательность операторов, преобразующих начальное состояние в целевое, состоит из goto(B), pushbox(C), climbbox и grasp.

10.2.2. Методы поиска на графе

Для небольших графов, один из которых показан на рис. 10.8, путь из начального состояния в конечное может быть легко найден непосредственной проверкой. В случае более сложных графов для определения пути в пространстве состояний из начального состояния в конечное требуется формализованный процесс поиска. Один из способов описания процесса поиска пути на графе заключается в использовании *производящих систем*. Производящая система включает следующее:

1. *Базу данных*, которая содержит информацию, относящуюся к определенной задаче.

2. Набор *правил*, действующих над базой данных. Каждое правило состоит из левой части, которая определяет применимость правила (предусловие), и правой части, описывающей действие, которое должно быть выполнено в случае применения правила. Применение правил изменяет базу данных.

3. *Стратегию управления*, которая определяет, какие правила должны быть применены, а также прекращение вычислений, когда для базы данных выполняется конечное условие.

На языке производящих систем граф, показанный на рис. 10.8, создается стратегией управления. Различные базы данных, полученные с помощью правил, на графе обычно представляются в виде вершин. Таким образом, стратегия управления поиском на графе подразумевает нахождение пути на графе из начальной вершины, представляющей собой начальную базу данных, в целевую вершину. Целевая вершина в свою очередь является базой данных, удовлетворяющей конечному (или целевому) условию производящей системы.

Процедура поиска на графе может быть описана следующим образом:

Шаг 1. Создать граф поиска G, состоящий только из начальной вершины s. Занести s в список, называемый OPEN.

Шаг 2. Создать список, называемый CLOSED, который первоначально является пустым.

Шаг 3. LOOP: если OPEN пуст, выход на аварийный останов.

Шаг 4. Выбрать первую вершину из OPEN, удалить ее из OPEN и занести ее в CLOSED. Назвать эту вершину n.

Шаг 5. Если n является целевой вершиной, выход на останов с выдачей решения, полученного с помощью отслеживания пути, помеченного указателями, в направлении от n к s в G (указатели [1] определяются на шаге 7).

[1] В данном случае под указателем понимается направление дуги от одной вершины графа к другой. — *Прим. перев.*

Шаг 6. Расширить вершину n, создав набор M из преемников, не являющихся предшественниками n. Определить вершины набора M в качестве преемников n в G.

Шаг 7. Установить указатели в направлении n от тех вершин набора M, которые еще не были в списках OPEN или CLOSED. Добавить эти вершины из набора M в OPEN. Для каждой вершины набора M, который уже был в OPEN или CLOSED, решить, надо ли восстановить его указатель в направлении n. Для каждой вершины из M, уже находящейся в CLOSED, определить для каждого из его потомков в G, следует ли восстанавливать его указатель [1].

Шаг 8. Переупорядочить список OPEN по некоторому произвольному критерию или эвристическому правилу.

Шаг 9. Перейти к выполнению шага 3.

Если никакая эвристическая информация не используется при упорядочении вершин из списка OPEN, на шаге 8 должен быть применен некоторый произвольный критерий. Результирующая процедура поиска называется *слепой*. Первый тип процедуры слепого поиска упорядочивает вершины в OPEN по мере возрастания глубины их расположения в дереве поиска [2]. Такой поиск называется поиском в ширину первого типа. Было показано, что этот метод обеспечивает нахождение пути минимальной длины к целевой вершине при условии, что этот путь существует.

Второй тип слепого поиска упорядочивает вершины в OPEN по мере уменьшения глубины их расположения в дереве поиска. Вершины, располагающиеся на наибольшей глубине, заносятся в список первыми. Вершины равной глубины располагаются в произвольном порядке. Назовем этот метод поиском в глубину первого типа. Для предотвращения процесса поиска вдоль некоторого бесполезного пути устанавливается ограничение на глубину. Ни одна вершина, глубина которой превышает ограничения, не порождается.

Описанных выше методов слепого поиска достаточно для нахождения путей от начальной вершины к целевой. Во многих случаях допустимо использование дополнительной информации, зависящей от вида решаемой задачи и помогающей сократить поиск. Этот класс процедур поиска называется *эвристиками*

[1] Если граф, подлежащий исследованию, является деревом, то ни один из преемников, созданных на шаге 6, не был создан ранее. Таким образом, вершины из набора M уже не входят либо в OPEN, либо в CLOSED. В этом случае каждая вершина из набора M добавляется в OPEN и размещается в исследуемом дереве в качестве преемника n. Если граф, подлежащий исследованию, не является деревом, возможно, что некоторые из вершин набора M уже были созданы, т. е. они могли уже быть в OPEN или CLOSED.

[2] Для обеспечения наиболее раннего завершения целевые вершины должны быть занесены в начало OPEN.

или *наилучшим поиском первого типа*, а используемая информация, зависящая от вида решаемой задачи, называется *эвристической информацией*. На шаге 8 процедуры поиска на графе эвристическая информация может быть использована для упорядочивания вершин в списке OPEN так, чтобы поиск осуществлялся среди наиболее перспективных секторов графа. Рассмотрим важный метод, который использует функцию оценивания для вычисления «перспективы» вершин. Вершины в списке OPEN упорядочиваются по мере увеличения значения функции оценивания. Связи среди вершин устанавливаются произвольно, но всегда в пользу целевых вершин. Результаты поиска существенно зависят от выбора функции оценивания. Полезный алгоритм наилучшего поиска первого типа, так называемый A*-алгоритм, приводится ниже.

Пусть функция оценивания для любой вершины n имеет следующий вид:

$$f(n) = g(n) + h(n),$$

где $g(n)$ — мера стоимости прохода от начальной вершины к вершине n, а $h(n)$ — оценка стоимости прохода от вершины n к целевой вершине. Таким образом, $f(n)$ представляет собой оценку стоимости прохода от начальной вершины к целевой вдоль пути, проходящего через вершину n.

A*-алгоритм

Шаг 1. Вначале список OPEN содержит только начальную вершину. Присвоить g значение 0, h — произвольное значение, а f — присвоить значение $h+0$ или h. Список CLOSED считать пустым.

Шаг 2. До тех пор, пока целевая вершина не будет найдена, повторять следующую процедуру: если в списке OPEN нет ни одной вершины, выдать сообщение об аварийном останове. В противном случае выбрать из OPEN вершину с наименьшим значением f. Присвоить ей имя BESTNODE (наилучшая вершина), удалить из списка OPEN и занести в список CLOSED. Проверить, является ли BESTNODE целевой вершиной. Если да, то выход на останов с сообщением о решении (либо выдать целевую вершину BESTNODE, либо путь между начальной вершиной и целевой вершиной BESTNODE). В противном случае создать вершины — преемники вершине BESTNODE, но не определять указатели между вершиной BESTNODE и вершинами-преемниками. (Сначала необходимо установить, имеются ли среди них ранее созданные вершины.) Для каждой такой вершины, имеющей имя SUCCESSOR (преемник), выполнить следующее:

а) Установить указатель от вершины SUCCESSOR к BESTNODE. Такие обратные связи дают возможность восстановить путь в случае нахождения решения.

б) Вычислить g(SUCCESSOR) $= g$(BESTNODE) + стоимость прохода от BESTNODE к SUCCESSOR.

в) Проверить, совпадает ли SUCCESSOR с какой-либо вершиной из списка OPEN (т. е. была ли она уже создана, но не обработана). Если да, назовем эту вершину OLD. Поскольку данная вершина уже существует в графе, мы можем отбросить SUCCESSOR и добавить OLD в список преемников BESTNODE. Теперь решим, надо ли определять путь от вершины OLD к BESTNODE. Поскольку OLD и SUCCESSOR по существу являются одной и той же вершиной, надо сравнить их значение g. Если путь к OLD через его предшественника дешевле, чем путь к SUCCESSOR через BESTNODE, то ничего не надо делать. Если путь к вершине SUCCESSOR дешевле, вновь устанавливаем путь от вершины OLD к BESTNODE, присваиваем значению g(OLD) значение, соответствующее более дешевому пути, и вычисляем новое значение f(OLD).

г) Если вершина SUCCESSOR не входит в список OPEN, проверяем, входит ли она в список CLOSED. Если входит, этой вершине из списка CLOSED присваиваем имя OLD и заносим ее в список преемников вершины BESTNODE. Проверяем так же, как на шаге 2*в*, какой путь лучше — новый или старый, и аналогично определяем путь и значения f и g. Если мы только что определили лучший путь к вершине OLD, мы должны распространить улучшение на всех преемников этой вершины. Таким образом, вершина OLD определяет своих преемников, а эти преемники в свою очередь определяют своих преемников. Так продолжается до тех пор, пока каждая ветвь не закончится вершиной, которая либо находится в списке OPEN, либо не имеет преемников. Чтобы распространить новую стоимость вниз, осуществляем поиск первого типа в глубину на дереве, начинающемся с вершины OLD, меняем для каждой вершины значение g (и соответственно значение f). Для каждой ветви графа поиск заканчивается, когда вы либо достигли вершины, не имеющей преемников, либо вершины, для которой уже был найден эквивалентный или же лучший путь. Это условие легко проверить. Поскольку мы идем вниз к вершине, проверяем, имеется ли путь от предшественника вершины к ней. Если да, продолжаем распространение. Если нет, значение g уже определяет лучшую часть пути. Поэтому на этом шаге распространение может быть прекращено. Но, возможно, что с новым значением g, полученным далее, путь, вдоль которого мы следовали, окажется лучшим, чем путь через текущего предшественника. Поэтому они оба сравниваются между собой. Если путь через текущего предшественника все же лучше, прекращаем распространение. Если же путь, вдоль которого распространялось улучшение, все же лучше, вновь определяем предшественника и продолжаем распространение.

д) Если вершина SUCCESSOR не находится ни в OPEN, ни в CLOSED, заносим ее в OPEN и добавляем в список преемников вершины BESTNODE.

$$\text{Вычисляем } f(\text{SUCCESSOR}) = \\ = f(\text{SUCCESSOR}) + h(\text{SUCCESSOR}).$$

Легко видеть, что A*-алгоритм представляет собой алгоритм поиска на графе, который использует $f(n)$ в качестве функции оценивания для упорядочивания вершин. Отметим, что, поскольку $g(n)$ и $h(n)$ складываются, важно, чтобы значение $h(n)$ являлось мерой стоимости прохода от вершины n к целевой вершине.

Целью процедуры поиска является определение пути в пространстве состояний от начального состояния к целевому. Имеются два направления, в которых такой поиск может осуществляться: 1) вперед из начального состояния и 2) назад из целевого состояния. В модели производящей системы имеется набор правил как для вывода набора промежуточных состояний вперед, так и для вывода назад. При выводе вперед левые части, или предусловия, соответствуют текущим состояниям, а правые части (результаты) используются для генерации новых вершин до тех пор, пока не будет достигнуто целевое состояние. При выводе назад правые части соответствуют текущим состояниям, а левые части используются для генерации новых вершин, представляющих собой новые целевые состояния, которые необходимо достичь. Это продолжается до тех пор, пока одно из целевых состояний не будет соответствовать начальному состоянию.

Описав процесс поиска как последовательное применение ряда правил, легко создать алгоритмы поиска, не зависящие от направления поиска. Конечно, имеется возможность комбинации поиска вперед и назад. В этом случае одновременно отыскивается путь как из целевого состояния в конечное, так и путь из конечного состояния в целевое до тех пор, пока они не пересекутся. Такая стратегия называется *стратегией двунаправленного поиска.*

10.3. ПРОБЛЕМА СВЕДЕНИЯ ЗАДАЧИ К ПОДЗАДАЧАМ

Другой подход к решению задачи основан на *проблеме сведения задачи к подзадачам.* Основная идея заключается в разбиении задачи на подзадачи, которые в свою очередь разбиваются на подзадачи, и так до тех пор, пока исходная задача не сведется к ряду тривиальных задач, имеющих очевидные решения. Оператор сведения задачи преобразует описание задачи в описание ряда подзадач. К данному описанию задачи могут применяться многие операторы. Применение каждого

оператора также приводит к необходимости решения ряда подзадач. Поскольку не все из них в свою очередь могут быть решены, необходимо применять несколько операторов для решения всех возникающих подзадач. Это снова требует организации процесса поиска.

Разбиение задачи на ряд подзадач обычно представляется в виде графа. Предположим, что задачу A можно решить, если будут решены три подзадачи B, C и D (рис. 10.9). Дуги, ведущие из вершины A к вершинам B, C и D, назовем И-дугами. Тогда вершины B, C и D называются И-вершинами. С другой

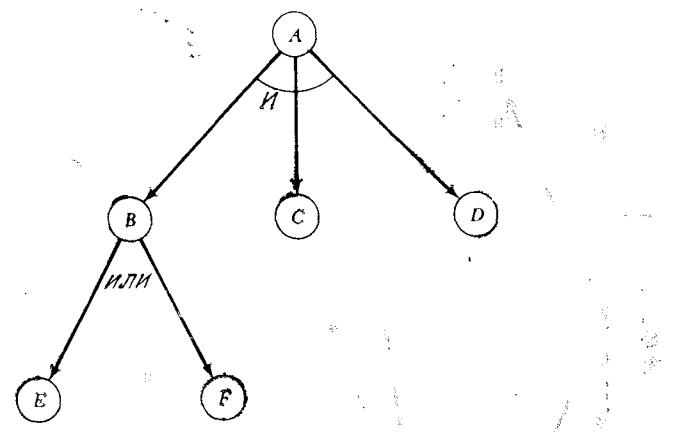


Рис. 10.9. Граф И/ИЛИ.

стороны, если задача B может быть решена в случае решения одной из подзадач E и F, то в этом случае будем использовать дугу ИЛИ. Полученный граф называется графом И/ИЛИ (рис. 10.9). Легко видеть, что методы поиска, рассмотренные в разд. 10.2, сводятся к графу И/ИЛИ, с помощью которого мы хотим найти единственный путь из начальной вершины к целевой.

Пример. Граф И/ИЛИ для задачи «Обезьяна и бананы» приведен на рис. 10.10. В данном случае задача представляется тройкой (S, F, G), где S — набор начальных состояний, F — набор операторов и G — набор целевых состояний. Поскольку для данной задачи набор операторов F является постоянным и начальное состояние представляется в виде $(A, 0, B, 0)$, то можно исключить символ F и описать задачу как $(\{A, 0, B, 0\}, G)$. Один из путей выбора операторов сведения задачи заключается в применении понятия *различие.* Короче говоря, различие для тройки (S, F, G) представляет собой список причин, по которым ни один из элементов набора S не входит в набор G. (Если

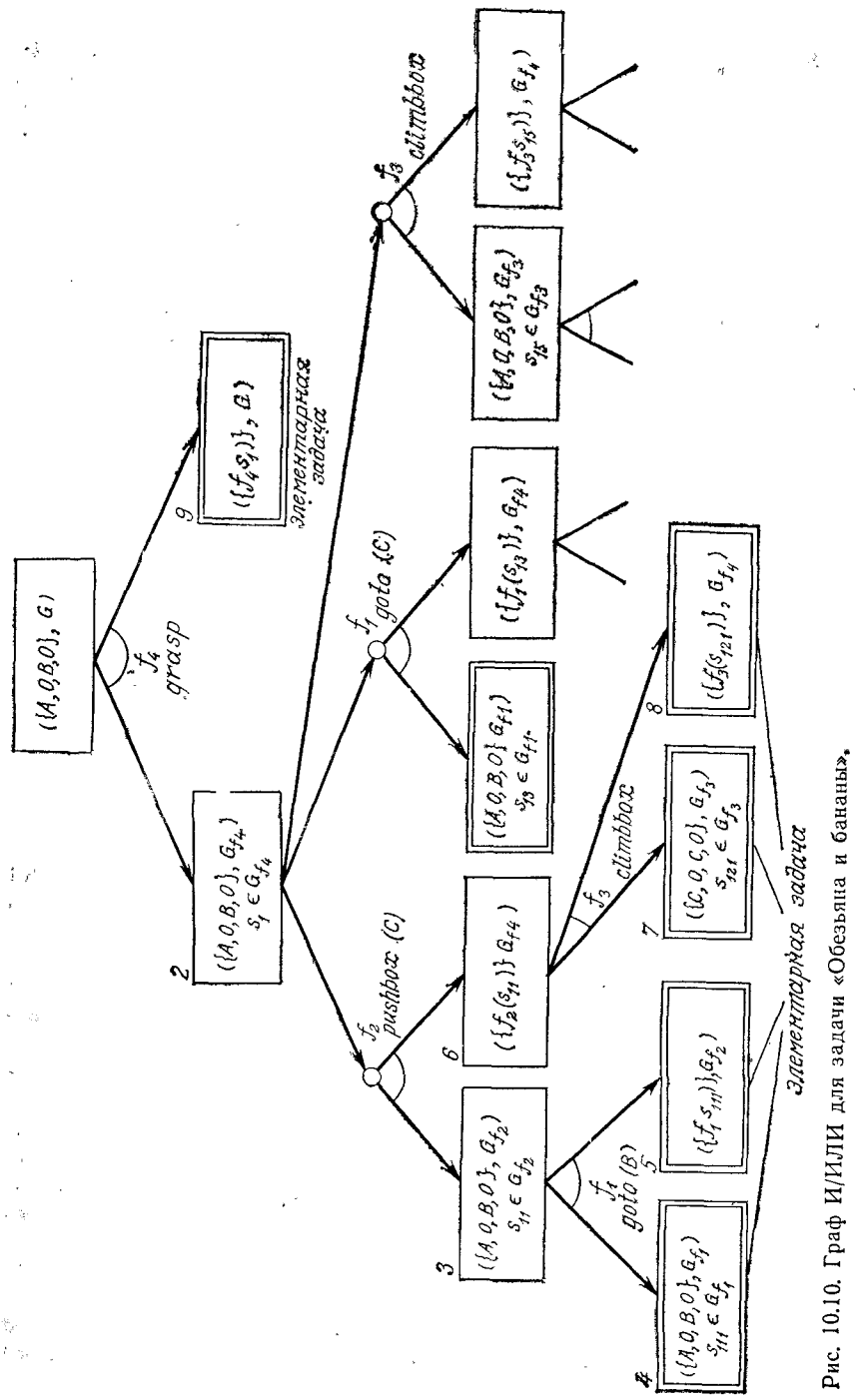


Рис. 10.10. Граф И/ИЛИ для задачи «Обезьяна и бананы».

хотя бы один из элементов набора S входит в G, задача решается и различие отсутствует.)

В примере из разд. 10.2.1 $F = \{f_1, f_2, f_3, f_4\} = \{\text{goto}(U), \text{pushbox}(V), \text{climbbox}, \text{grasp}\}$. Сначала мы вычисляем различие для начальной задачи. Дело в том, что список $(A, 0, B, 0)$ не может удовлетворять решению, поскольку в нем последний элемент не равен 1.

Оператором, уменьшающим это различие, является оператор $f_4 = \text{grasp}$. Используя f_4 для разбиения начальной задачи, мы получаем следующую пару подзадач: $(\{(A, 0, B, 0)\}, G_{f_4})$ и $(\{f_4(s_1)\}\ G)$, где G_{f_4} представляет собой набор описаний состояний, к которым применим оператор f_4, и s_1 — состояние, входящее в G_{f4}, полученное при решении подзадачи $(\{(A, 0, B, 0)\}), G_{f_4})$.

Для решения задачи $(\{(A, 0, B, 0)\}, G_{f_4})$ сначала надо вычислить ее различие. Состояние, описываемое $(\{(A, 0, B, 0)\}, G_{f_4})$, не входит в G_{f_4}, поскольку:

1) ящик не находится в точке C;
2) обезьяна не находится в точке C;
3) обезьяна не находится на ящике.

Операторами для уменьшения этих различий являются $f_2 = \text{pushbox}(C)$, $f_1 = \text{goto}(C)$ и $f_3 = \text{climbbox}$ соответственно. Применяя оператор f_2, получаем подзадачи $(\{(A, 0, B, 0)\}, G_{f2})$ и $(f_2(s_{11}), G_{f2})$, где $s_{11} \in G_{f2}$ вытекает из решения первой подзадачи.

Поскольку сначала должна быть решена подзадача $(\{(A, 0, B, 0)\}, G_{f2})$, то для нее вычисляется различие. Различие состоит в том, что обезьяна не находится в точке B и оператор, который устраняет его, есть $f_1 = \text{goto}(B)$. Этот оператор затем используется для сведения задачи к паре подзадач $(\{(A, 0, B, 0)\}, G_{f1})$ и $(f_1(s_{111}), G_{f2})$. Теперь первая из этих задач является элементарной. Ее различие равно 0, поскольку $(A, 0, B, 0)$ входит в область определения оператора f_1, с помощью которого можно решить эту задачу. Отметим, что $f_1(s_{111}) = (B, 0, B, 0)$, поэтому возникает вторая задача $(\{(B, 0, B, 0)\}, G_{f_2})$. Эта задача также является элементарной, поскольку $(B, 0, B, 0)$ входит в область определения оператора f_2, с помощью которого можно решить эту задачу. Процесс последовательного решения подзадач должен продолжаться до тех пор, пока исходная задача не будет решена.

В графе И/ИЛИ одна из вершин, называемая *начальной вершиной состояния*, соответствует описанию исходной задачи. Вершины графа, соответствующие описаниям элементарных задач, называются *конечными вершинами*. Целью процесса поиска на графе является доказательство того, что задача, соответствующая начальной вершине, имеет решение, т. е. что начальная вершина является решаемой. Рекурсивное определе-

ние решаемой вершины может быть дано следующим образом:

1. Конечные вершины являются решаемыми вершинами, поскольку они соответствуют элементарным задачам.

2. Если вершина, не являющаяся конечной, имеет преемников типа ИЛИ, она является решаемой вершиной только тогда, когда по меньшей мере один из ее преемников является решаемым.

3. Если вершина, не являющаяся конечной, имеет преемников типа И, она является решаемой только тогда, когда все преемники являются решаемыми.

Граф решения, который является подграфом, состоящим из решаемых вершин, показывает, что начальная вершина является решаемой. Целью производящей системы или процесса поиска является нахождение графа решения от начальной вершины к конечным вершинам. Граф решения, определяющий путь от вершины n к набору вершин N графа типа И/ИЛИ, в некоторой степени аналогичен пути на обычном графе. Он может быть получен, если начало пути определить в вершине n и каждый раз выбирать только одну выходящую из нее дугу. Для каждой вершины-преемника, к которой направлена эта дуга, мы продолжаем выбирать только одну выходящую из нее дугу. И так до тех пор, пока в конце концов каждый преемник, созданный таким образом, не войдет в набор N.

Для определения решений на графе И/ИЛИ нам необходим алгоритм, подобный алгоритму А*, но способный выбирать И-дуги соответствующим образом. Такой алгоритм для реализации эвристического поиска на графе И/ИЛИ называется АО*-алгоритмом.

АО*-алгоритм

Шаг 1. В G входит только одна вершина, соответствующая начальному состоянию. (Назовем эту вершину INIT.) Вычисляется h(INIT).

Шаг 2. До тех пор пока INIT не будет помечена как SOLVED или значение h(INIT) не превзойдет FUTILITY, повторять следующие действия:

а) Идти из вершины INIT вдоль помеченных дуг и выбрать для процесса расширения одну из вершин на этом пути, ранее не подвергавшуюся расширению. Назвать выбранную вершину NODE.

б) Создать преемников вершины NODE. Если их нет, присвоить FUTILITY и h значение NODE. Это означает, что NODE не является решаемой вершиной. Если преемники имеются для каждого из них (названного SUCCESSOR), который также не является предшественником NODE, делать следующее:

1) добавить SUCCESSOR к G;

2) если SUCCESSOR является конечной вершиной, пометить его SOLVED и присвоить его значению h значение 0;

3) Если SUCCESSOR не является конечной вершиной, вычислить его значение h.

в) Распространять вновь полученную информацию по графу следующим образом:

Пусть S является набором вершин, которые были помечены SOLVED или значения h которых были изменены. И поэтому необходимо знать значения h вершин их предшественников. Начнем рассмотрение S с вершины NODE. До тех пор пока S не станет пустым, повторять следующие действия:

1) Выбрать из S вершину, у которой ни один из потомков, входящих в G, не входит в S. (Другими словами, надо удостовериться, что для каждой вершины, подлежащей обработке, она будет обработана раньше любого из ее предшественников.) Назовем эту вершину CURRENT и устраним ее из S.

2) Вычислить стоимость каждой дуги, выходящей из CURRENT. Стоимость дуги равна сумме значений h каждой вершины на конце дуги плюс стоимость самой дуги. Присвоить новому значению h вершины CURRENT минимальное значение среди только что вычисленных стоимостей дуг, выходящих из нее.

3) Отметить лучший путь из вершины CURRENT, пометив дугу, которая имеет минимальную стоимость, вычисленную на предыдущем шаге.

4) Пометить вершину CURRENT SOLVED, если все вершины, связанные с ней через вновь помеченную дугу, были помечены SOLVED.

5) Если CURRENT уже была отмечена SOLVED или если стоимость вершины CURRENT только что была изменена, добавляем к S всех предшественников вершины CURRENT.

Отметим, что вместо двух списков OPEN и CLOSED, которые были использованы в А*-алгоритме, в АО*-алгоритме применяется единая структура G. Эта структура представляет собой часть графа поиска, которая к этому времени уже была создана. Каждая вершина на графе указывает как вниз (на своих преемников), так и вверх (на своих непосредственных предшественников). Каждой вершине графа присвоено значение h для оценивания стоимости пути от нее самой к набору вершин-решений. Значение g (стоимость прохода от начальной вершины к текушей) не сохраняется как в А*-алгоритме, поскольку h служит оценкой качества вершины. Величина FUTILITY необходима. Если оцениваемая стоимость решения становится больше, чем значение FUTILITY, поиск прекращается. Значение FUTILITY должно быть выбрано так, чтобы соответствовать порогу, за пределами которого любое решение, даже если бы оно могло быть найдено, было бы слишком громоздким, чтобы его можно было применять на практике.

Алгоритм поиска в ширину первого типа может быть получен из АО*-алгоритма, если $h \equiv 0$.

10.4. ПРИМЕНЕНИЕ ЛОГИКИ ПРЕДИКАТОВ

Для решения задач, связанных с управлением робота, должна быть реализована возможность представления исправления и преобразования наборов утверждений. Для этого может быть использован язык логики или, более точно, язык логики предикатов первого порядка. Логический формализм представляется привлекательным, поскольку является мощным средством выведения новых знаний из старых (методом математической дедукции). По этому методу новое утверждение истинно, если можно доказать, что оно следует из утверждения, о котором уже известно, что оно истинно. Эту идею доказательства, можно применять для получения ответов на вопросы и решения задач в робототехнике [1].

Рассмотрим сначала возможности логики высказываний как способа представления знаний. Она имеет то преимущество, что ее аппарат достаточно прост и для представления знаний имеются соответствующие операции. На языке логики высказываний легко описывать явления окружающего мира в виде набора правильно построенных формул (ППФ). Например:

Идет дождь.
RAINING
Светит солнце.
SUNNY
Стелется туман.
FOGGY
Если идет дождь, то не светит солнце.
$\text{RAINING} \rightarrow \sim \text{SUNNY}$

Используя эти утверждения, мы, например, можем вывести, что не светит солнце, если идет дождь. Но очень быстро мы сталкиваемся с ограничениями логики высказываний. Предположим, мы хотим на ее языке представить очевидный факт, констатируемый следующим предположением:

ДЖОН — человек

Мы можем написать

JOHNMAN

Но аналогично можно сказать

Пол — человек

или

PAULMAN

[1] Читателям, не знакомым с логикой высказывания и логикой предикатов, мы можем рекомендовать просмотреть введение в логику прежде, чем читать оставшуюся часть главы. Читатели, желающие более полно ознакомиться с материалом этого раздела, должны просмотреть книгу [43].

что могло бы быть совершенно другим утверждением и нам было бы трудно вывести какое-либо заключение о степени схожести между Джоном и Полом.

Было бы намного лучше представить это в виде:

MAN(JOHN)
MAN(PAUL)

поскольку теперь структура представления отражает структуру самих знаний. Возникает еще больше трудностей, если мы попытаемся на языке логики высказываний описать следующее утверждение: «все люди смертны», так как потребуются утверждения для представления соотношений количества, если мы не собираемся по отдельности описывать смерть каждого известного человека.

Поэтому придется воспользоваться аппаратом логики предикатов, поскольку на ее языке можно описывать события, не поддающиеся описанию на языке логики высказываний. На языке логики предикатов можно представить события окружающего мира в виде ППФ. Но главная причина выбора логики предикатов состоит в том, что с ее помощью можно не только представить знания, но и получить из них новые знания.

В этом разделе мы кратко опишем язык и методы логики предикатов. Основными компонентами языка логики предикатов являются: *предикаты, переменные, функции* и *константы.* Предикаты используются для установления связи в рассуждениях. Например, для описания факта «Робот в комнате r_1» достаточно простой *формулы-атома*:

$$\text{INROOM}(\text{ROBOT}, r_1)$$

В этой формуле ROBOT и r_1 являются константами. Обычно формулы-атомы состоят из предикатов и термов. Константа является простейшим видом терма для представления объектов или вещей при рассуждениях. Переменные также являются термами для описания объектов, относящихся к некоторому классу, как, например, $\text{INROOM}(x, y)$. Функции применяются для определения связей между объектами, подлежащими рассмотрению. Например, функция *mother* (мать) применяется для установления связи между человеком и его (ее) родителем по материнской линии. Можно воспользоваться следующей формулой-атомом для представления утверждения «Мать Джона замужем за отцом Джона»:

$$\text{MARRIED}[\text{father}(\text{JOHN}), \text{mother}(\text{JOHN})].$$

Формула-атом имеет значение И (истина), когда соответствующее утверждение истинно, и Л (ложь), когда соответствующее утверждение ложно. Таким образом INROOM(RO-

ВОТ, r_1) имеет значение И, а INROOM(ROBOT, r_2) имеет значение Л. Формулы-атомы являются элементарными формулами, из которых строятся более сложные утверждения на языке логики предикатов. Из формул-атомов можно строить более сложные ППФ, применяя операции $\wedge$ (и) $\vee$ (или) и $\Rightarrow$ (следование). Формулы, построенные путем связи одних формул с другими символом $\wedge$, называются *конъюнкциями*, а символом $\vee$ — *дизъюнкциями*. Символ $\Rightarrow$ применяется для представления утверждения «если—тогда». Например, «если обезьяна на ящике, она схватит бананы»:

$$\text{ON(MONKEY, BOX)} \Rightarrow \text{GRASP(MONKEY, BANANAS)}.$$

Символ $\sim$ означает отрицание, т. е. он изменяет значение ППФ с И на Л и наоборот. (Истинное) утверждение «робот не находится в комнате r_2» может быть представлено в виде

$$\sim \text{INROOM(ROBOT}, r_2)$$

Иногда формула-атом $P(x)$ имеет значение И для всех возможных значений x. Это свойство описывается с помощью квантора всеобщности $(\forall x)$, который ставится перед $P(x)$. Если $P(x)$ имеет значение И хотя бы для одного значения x, это свойство представляется с помощью квантора существования $(\exists x)$, который ставится перед $P(x)$. Например, утверждение «все роботы серые» можно выразить в виде

$$(\forall x)\,[\text{ROBOT}(x) \Rightarrow \text{COLOR}(x, \text{GRAY})],$$

а «в комнате r_1 находится объект»

$$(\exists x)\,\text{INROOM}\,(x, r_1)$$

Если P и Q являются ППФ, истинные значения выражений, составленных из этих формул, приводятся ниже в таблице:

P	Q	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$\sim P$
И	И	И	И	И	Л
Л	И	И	Л	И	И
И	Л	И	Л	Л	Л
Л	Л	Л	Л	И	И

Если два истинных значения ППФ одинаково интерпретируются, эти ППФ являются эквивалентными. Используя таблицу истинных значений, мы можем установить следующие эквивалентности:

$\sim (\sim P)$ эквивалентно P
$P \vee Q$ эквивалентно $\sim P \Rightarrow Q$

Законы де Моргана:

$\sim (P \vee Q)$ эквивалентно $\sim P \wedge \sim Q$
$\sim (P \wedge Q)$ эквивалентно $\sim P \vee \sim Q$

Дистрибутивные законы:

$P \wedge (Q \vee R)$ эквивалентно $(P \wedge Q) \vee (P \wedge R)$
$P \vee (Q \wedge R)$ эквивалентно $(P \vee Q) \wedge (P \vee R)$

Коммутативные законы:

$P \wedge Q$ эквивалентно $Q \wedge P$
$P \vee Q$ эквивалентно $Q \vee P$

Ассоциативные законы:

$(P \wedge Q) \vee R$ эквивалентно $P \wedge (Q \wedge R)$
$(P \vee Q) \vee R$ эквивалентно $P \vee (Q \vee R)$

Контропозитивный закон:

$P \Rightarrow Q$ эквивалентно $\sim Q \Rightarrow \sim P$

И, кроме того, мы имеем:

$\sim (\exists x) P(x)$ эквивалентно $(\forall x)\,[\sim P(x)]$
$\sim (\forall x) P(x)$ эквивалентно $(\exists x)\,[\sim P(x)]$

В логике предикатов имеются правила вывода, которые можно применять к определенным ППФ и наборам ППФ для получения новых ППФ. Важным правилом вывода является *модус поненс*, являющийся правилом вывода ППФ W_2 из ППФ вида W_1 и $W_1 \Rightarrow W_2$. Другое правило вывода — *универсальная специализация* позволяет выводить ППФ $W(A)$ из ППФ $(\forall x)\,W(x)$, где A является константой. Используя совместно модус поненс и универсальную специализацию, можно, например, вывести ППФ $W_2(A)$ из ППФ $(\forall x)\,[W_1(x) \Rightarrow W_2(x)]$ и $W_1(A)$.

В логике предикатов ППФ формулы, полученные с помощью правил вывода, называются *теоремами*, а последовательность правил вывода — *доказательством* теоремы. В искусственном интеллекте некоторые задачи могут быть рассмотрены как задачи доказательства теорем. Последовательность правил вывода дает решение требуемой задачи.

Пример. Пространство состояний задачи «Обезьяна и бананы» можно описать ППФ. Мы предполагаем, что используются 3 оператора: grasp (захват), climbbox (прыжок на ящик), pushbox (передвижение ящика).

Пусть начальное состояние s_0 описывается следующим набором ППФ:

$$\sim \text{ONBOX}$$
$$\text{AT(BOX}, B)$$
$$\textbf{AT}(\text{BANANAS};\ C)$$
$$\sim \text{HB}$$

Предикат ONBOX имеет значение И(истинно), только когда обезьяна находится на вершине ящика, а предикат HB имеет значение И, только когда у обезьяны имеются бананы.

Действия трех операторов можно описать следующими ППФ:

1) grasp

$$(\forall s)\{\text{ONBOX}(s) \wedge \text{AT}(\text{BOX}, C, s) \Rightarrow \text{HB}(\text{grasp}(s))\}$$

что означает: «для любого s, если обезьяна находится на ящике и ящик расположен в точке C в состоянии s, обезьяна завладеет бананами, если оператор grasp применить к состоянию s».

Отметим, что значением оператора $\text{grasp}(s)$ является новое состояние после того, как этот оператор был применен к состоянию s.

2) climbbox

$$(\forall s)\{\text{ONBOX}(\text{climbbox}(s))\}$$

что означает: «для любого s обезьяна будет находиться на ящике в состоянии, которое получается в результате применения оператора climbbox к состоянию s».

3) pushbox

$$(\forall x \forall s)\{\sim \text{ONBOX}(s) \Rightarrow \text{AT}(\text{BOX}, x, \text{pushbox}(x, s))\}$$

что означает: «для любого x и s, если обезьяна не находится на ящике в состоянии s, ящик будет в положении x в состоянии, полученном в результате применения оператора $\text{pushbox}(x)$ к состоянию s».

Целевой ППФ является

$$(\exists s)\, HB(s)$$

Чтобы показать, что обезьяна может достать бананы, эту задачу можно решить как доказательство теоремы [216].

10.5. АНАЛИЗ КОНЕЧНЫХ ЗНАЧЕНИЙ

Выше мы рассмотрели несколько методов поиска вперед и назад, но для конкретной задачи должно быть выбрано одно из двух возможных направлений. Однако для решения задачи часто применяется поиск в двух направлениях. Такая смешанная стратегия давала бы возможность решить основные части задачи методом поиска вперед. Затем можно было бы идти назад, чтобы решить вспомогательные задачи, возникающие при соединении вместе отдельных частей решения. Это можно осуществить, применяя метод *анализ конечных значений*. Метод основывается на установлении различия между текущим и целевым состояниями. Если различие установлено, необходимо найти оператор, который может его уменьшить. Возможно, что этот оператор неприменим к текущему состоянию. Поэтому возникает подзадача получения состояния, в котором он может применяться. Также возможно, что оператор не приводит точно к целевому состоянию. Тогда мы имеем вторую подзадачу получения состояния, применив к которому данный оператор, мы перейдем в целевое состояние.

Если различие определено правильно и результат применения оператора действительно уменьшает это различие, легче решить две подзадачи, чем исходную задачу. Анализ конечных значений рекурсивно применяется к подзадачам. С этой точки зрения анализ конечных значений можно было бы рассматривать как метод сведения задачи к подзадачам. Для того чтобы нацелить систему на решение больших задач, различиям должны быть присвоены приоритетные уровни. Различия с высоким приоритетом рассматриваются раньше, чем различия с низким приоритетом. Наиболее важной структурой данных, используемой в анализе конечных значений, является «цель». Цель представляет собой кодировку текущей проблемной ситуации, желаемой ситуации и историю прежних попыток превращения текущей ситуации в желаемую. Имеются цели трех основных типов:

Тип 1. *Преобразование* объекта A в объект B.

Тип 2. *Уменьшение* различия между объектом A и объектом B посредством модификации объекта A.

Тип 3. *Применение* оператора Q к объекту A.

Для достижения целей указанных типов имеются соответствующие процедуры (рис. 10.11), которые могут быть интерпретированы как операторы *сведения* исходной задачи к подзадачам, соответствующим И-вершинам (в случае преобразования объекта или применения оператора) либо ИЛИ-вершинам (в случае уменьшения различия).

Первой программой, применяющей анализ конечных значений, был общий решатель задач GPS (general problem solver). Причиной его создания послужило частое использование этого метода для решения задач. Для GPS исходной задачей является достижение цели *преобразования* объекта A в объект B, где A является исходным объектом или состоянием, а B — желаемым объектом или целевым состоянием. GPS прекращает работу, если для цели *преобразование объекта* нет различия между A и B или же для цели *применение оператора* оператор Q непосредственно переводит A в B. Для цели *уменьшение различия* процесс вычисления может завершиться остановом в ситуациях, когда нельзя применить ни один из имеющихся операторов.

При *преобразовании объекта* A в объекте B для обнаружения различия между двумя объектами используется процесс сравнения. В первую очередь уменьшаются различия с более высокими приоритетами. Операторы, соответствующие уменьшению каждого различия, заносятся в таблицу различие — оператор.

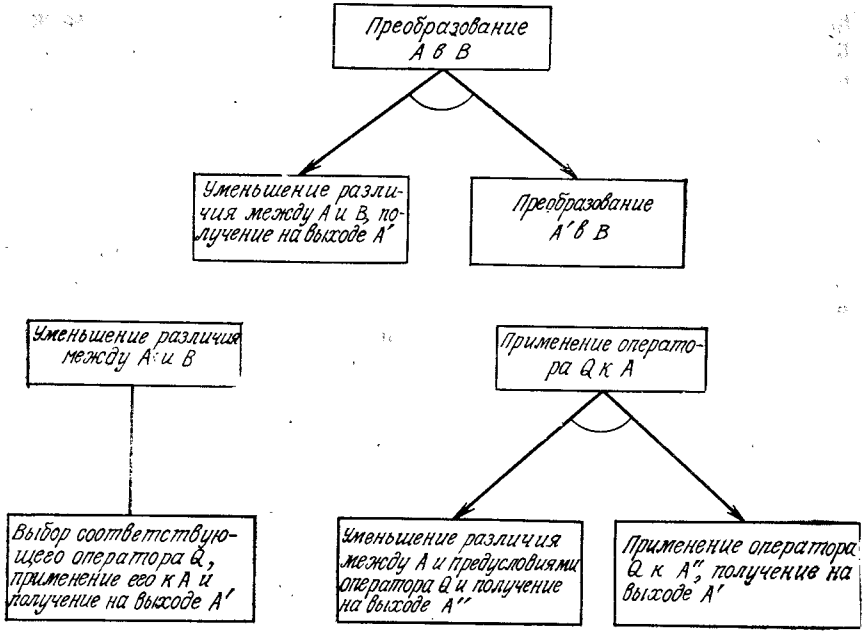


Рис. 10.11. Методы анализа конечных значений.

Рассмотрим простую задачу, операторы для решения которой приводятся ниже:

Предусловия	Оператор	Результат
1. *AT* (*ROBOT*, *OBJ*) ∧ *LARGE* (*OBJ*) ∧ *CLEAR* (*OBJ*) ∧ *HANDEMPTY*	*PUSH* (*OBJ*, *LOC*) →	*AT* (*OBJ*, *LOC*) ∧ *AT* (*ROBOT*, *LOC*)
2. *AT* (*ROBOT*, *OBJ*) ∧ *SMALL* (*OBJ*)	*CARRY* (*OBJ*, *LOC*) →	*AT* (*OBJ*, *LOC*) ∧ *AT* (*ROBOT*, *LOC*)
3. *None*	*WALK* (*LOC*) →	*AT* (*ROBOT*, *LOC*)
4. *AT* (*ROBOT* *OBJ*)	*PICKUP* (*OBJ*) →	*HOLDING* (*OBJ*)
5 *HOLDING* (*OBJ*)	*PUTDOWN* (*OBJ*) →	~ *HOLDING* (*OBJ*)
6. *AT* (*ROBOT*, *OBJ*2) ∧ *HOLDING* (*OBJ*1)	*PLACE* (*OBJ*1, *OBJ*2) →	*ON* (*OBJ*1, *OBJ*2)

где: *PUSH* (*OBJ*, *LOC*) — переместить объект *OBJ* в положение *LOC*;
CARRY (*OBJ*, *LOC*) — переместить объект *OBJ* в положение *LOC*;
WALK (*LOC*) — переместить робот в положение *LOC*;
PICKUP (*OBJ*) — поднять объект *OBJ*;
PUTDOWN (*OBJ*) — опустить объект *OBJ*;
PLACE (*OBJ*1, *OBJ*2) — разместить объект *OBJ*1 на объекте *OBJ*2.
Утверждения в предусловиях и результатах действия операторов имеют следующий смысл:

1. *AT* (*ROBOT*, *OBJ*)	Робот *ROBOT* и объект *OBJ* в одном месте
2. *AT* (*OBJ*, *LOC*)	Объект *OBJ* в месте *LOC*
3. *AT* (*ROBOT*, *LOC*)	Робот *ROBOT* в месте *LOC*
4. *AT* (*ROBOT*, *OBJ*2)	Робот *ROBOT* и объект *OBJ*2 в одном месте
5. *LARGE* (*OBJ*)	Объект *OBJ* большой
6. *SMALL* (*OBJ*)	Объект *OBJ* маленький
7. *CLEAR* (*OBJ*)	На объекте *OBJ* нет другого объекта
8. *HANDEMPTY*	В схвате робота нет объекта
9. *NONE*	Оператор не имеет предусловий
10. *HOLDING* (*OBJ*)	В схвате робота находится объект *OBJ*
11. *HOLDING* (*OBJ*1)	В схвате робота объект *OBJ*1
12. *ON* (*OBJ*1, *OBJ*2)	Объект *OBJ*1 на объекте *OBJ*2

В таблице на рис. 10.12 отмечены случаи применения тех или иных операторов. Заметим, что иногда могут использоваться

Различие	Оператор					
	PUSH	CARRY	WALK	PICKUP	PUTDOWN	PLACE
Переместить объект в определенное положение	+	+				
Переместить робот в определенное положение			+			
Снять один объект с другого объекта				+		
Положить один объект на другой						+
Освободить схват робота от объекта					+	+
Держать объект в схвате робота				+		

Рис. 10.12. Таблица «различие — оператор».

несколько операторов, которые уменьшают данное различие, или же один оператор может уменьшать несколько различий.

Предположим, что робот должен перенести платформу с двумя объектами на ней из одной комнаты в другую. Объекты на платформе также можно передвигать. Основным различием

между начальным и целевым состояниями является расположение платформы. Для уменьшения различия можно выбрать оператор PUSH или CARRY. Если сначала выбирается оператор CARRY, следует определить его предусловие. Это приводит к двум различиям, которые должны быть уменьшены, а именно: расположение робота и размер платформы. Расположение робота можно изменить с помощью оператора WALK, но нет операторов, которые могли бы изменить размер объекта, (платформы), поэтому этот вариант оказывается тупиковым. Рассмотрим другой вариант, т. е. попытаемся применить оператор PUCH. Он имеет три предусловия, два из которых определяют различия между начальным и целевым состояниями. Поскольку платформа предполагается большой, ни одно из предусловий не создает различия. Робот можно перевести в требуемое положение оператором WALK, и поверхность платформы можно освободить, применяя дважды оператор PICKUP. Но после первого применения оператора PICKUP попытка применить его второй раз приводит к другому различию: схват должен быть свободным. Для уменьшения этого различия можно использовать оператор PUTDOWN.

Применение оператора PUCH почти приводит к целевому состоянию. Объекты должны быть вновь размещены на платформе.

Это осуществляется с помощью оператора PLACE. Но он не может применяться непосредственно. Должно быть устранено другое различие, поскольку робот должен взять объекты. Это осуществляется с помощью оператора PICKUP. Чтобы применить оператор PICKUP, робот должен находиться в месте расположения объектов. Это различие можно уменьшить, применив оператор WALK. Поскольку робот находится в месте расположения двух объектов, можно воспользоваться операторами PICKUP и CARRY для передвижения объектов в другую комнату.

Очень важен порядок рассмотрения различий; например, в первую очередь должны быть уменьшены наиболее существенные различия. В разд. 10.6 описана система STRIPS, предназначенная для решения задач роботом и использующая анализ конечных значений.

10.6. РЕШЕНИЕ ЗАДАЧИ

Самая простая система для составления плана решения задачи роботом — производящая система, построенная на основе описания состояния в виде базы данных. Описание состояний и целей для задачи, решаемой роботом, может быть составлено из логических утверждений. В качестве примера рассмотрим схват робота и конфигурацию объектов, показанных на

рис. 10.1. Эту ситуацию можно описать конъюнкцией из следующих утверждений:

CLEAR(B)	На вершине объекта B ничего нет
CLEAR(C)	На вершине объекта C ничего нет
ON(C, A)	Объект C на объекте A
ONTABLE(A)	Объект A на столе
ONTABLE(B)	Объект B на столе
HANDEMPTY	В схвате робота нет объекта
HOLDING(X)	В схвате робота объект X

Цель заключается в построении пирамиды из объектов, в которой объект B находится на объекте C и объект A — на объекте B. В терминах логических утверждений эта цель выглядит следующим образом: $\mathrm{ON}(B, C) \wedge \mathrm{ON}(A, B)$.

Действия робота переводят одно состояние или конфигурацию рабочего пространства в другое. Один из простых и полезных методов для описания действий робота используется системой планирования решений задачи робота, которая называется STRIPS [78]. Для описания действий робота используется набор правил. Каждое правило состоит из трех компонент. Первым из них является *предусловие*, которое должно быть истинным, прежде чем правило может быть применено. Предусловие обычно составляет левую часть правила. Вторым компонентом является список предикатов, называемый *списком исключений.* Когда правило применяется к описанию состояний или базе данных, то из списка исключений, находящегося в базе данных, проводится исключение соответствующих утверждений. Третий компонент называется *списком добавлений.* Когда применяется правило, утверждение вводится в список добавлений, расположенный в базе данных. Описание действия MOVE для примера построения пирамиды из объектов A, B, C приводится ниже:

MOVE(X, Y, Z):	Перемещать объект X от Y к Z
Предусловие:	CLEAR(X), CLEAR(Z), ON(X, Y)
Список исключений:	ON(X, Y), CLEAR(Z)
Список добавлений:	ON(X, Z), CLEAR(Y)

Если MOVE является единственным имеющимся в наличии оператором или действием робота, в этом случае генерируется граф (или дерево), показанный на рис. 10.2.

Рассмотрим более подробно пример работы системы STRIPS с исходным состоянием базы данных, приведенном на рис. 10.1, и 4 действиями или операциями:

1. PICKUP(X) — поднять объект X

Предусловие и список исключений: ONTABLE(X), CLEAR(X), HANDEMPTY

Список добавлений: HOLDING(X)

2. PUTDOWN(X) — опустить объект X

Предусловие и список исключений: HOLDING(X)

Список добавлений: ONTABLE(X), CLEAR(X), HANDEMPTY

3. STACK(X, Y)— положить объект X на объект Y

Предусловие и список исключений: HOLDING(X), CLEAR(Y)

Список добавлений: HANDEMPTY, ON(X, Y), CLEAR(X)

4. UNSTACK(X, Y)— снять объект X с объекта Y

Предусловие и список исключений: HANDEMPTY, CLEAR(X), ON(X, Y)

Список добавлений: HOLDING(X), CLEAR(Y)

Предположим, что нашей целью является ON(B, C)$\wedge$ $\wedge$ ON(A, B). Начиная работу с описания начального состояния,

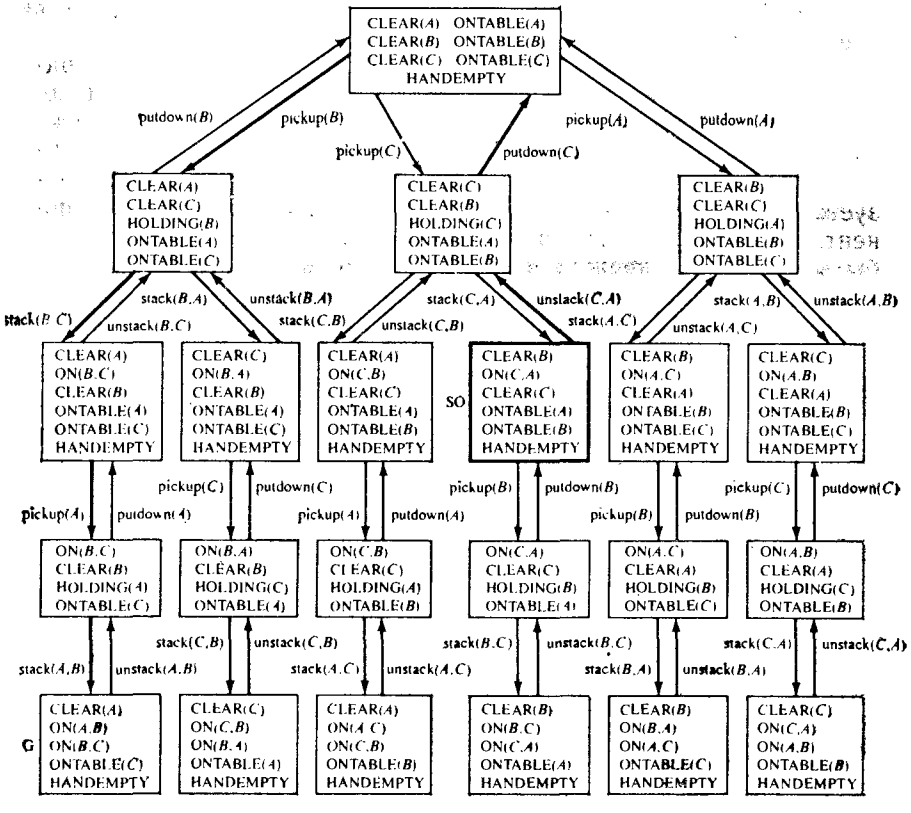


Рис. 10.13. Пространство состояний для задачи робота.

приведенного на рис. 10.1, мы получаем полное пространство состояний (рис. 10.13) и путь (решение) между начальным и целевым состояниями, отмеченный полужирными линиями. Последовательность действий для решения задачи имеет вид {UNSTACK(C, A), PUTDOWN(C), PICKUP(B), STACK(B, C), PICKUP(A), STACK(A, B)} и называется «планом» достижения цели.

Если системе известно, как каждый оператор изменяет состояние рабочего пространства или базы данных и известны предусловия оператора, который должен выполняться, она может применять анализ конечных значений для решения задачи. Короче говоря, по этому методу рассматриваются различия между текущими и целевыми состояниями и предпринимается попытка найти оператор, который уменьшит это различие. Таким оператором является тот, чей список добавлений содержит формулы, которые устранили хотя бы некоторую часть этого различия. Поиск продолжается рекурсивным образом до тех пор, пока не будет достигнуто целевое состояние. Система STRIPS и большинство других планировщиков используют анализ конечных значений.

Выше мы видели, как система STRIPS составляет план для решения конкретной задачи движения робота. Следующий шаг заключается в обобщении плана путем замены констант новыми параметрами. Другими словами, мы хотим расширить план до *схемы плана*. Требования обобщения плана появляются в обучающихся системах. Для сохранения планов (т. е. чтобы их части могли быть использованы в дальнейшем процессе планирования) должны быть известны предусловия и результаты действия любой части плана. С этой целью планы хранятся в *треугольной таблице* со строками и столбцами, соответствующими операторам плана. Треугольная таблица содержит структуру плана в такой форме, которая позволяет использовать части плана при решении сходных задач.

Пример треугольной таблицы приведен на рис. 10.14. Левый столбец назовем нулевым, тогда j-й столбец означает j-й оператор в последовательности плана. Пусть верхняя строка имеет номер 1. Если имеется N операторов в последовательности, представляющей собой план, тогда последняя строка будет иметь номер $N+1$.

Записи в (i, j)-х элементах таблицы для $j > 0$ и $i < N+1$ являются теми утверждениями, добавленными к описанию состояния j-м оператором, которые остались в качестве предусловий i-го оператора. Записями в $(i, 0)$-х элементах для $i < N+1$ являются те утверждения в описании начального состояния, которые остались как предусловия i-го оператора. Записи в $(N+1)$-й строке таблицы являются предложениями в описании первоначального состояния и предложениями, добавленными различными операторами, которые являются компонентами цели.

Треугольные таблицы легко строятся из описания начального состояния, последовательности операторов и описания цели. Эти таблицы являются четким и удобным представлением

для планов действия робота. Записи в строках слева от i-го оператора являются предусловием оператора. Записи в столбце снизу i-го оператора являются предложениями того оператора, которые требуются последующим операторам или которые являются компонентами цели.

i-е ядро можно получить из треугольной таблицы путем вычеркивания из нее верхних $i-1$ строк и всех столбцов, начиная с i-го.

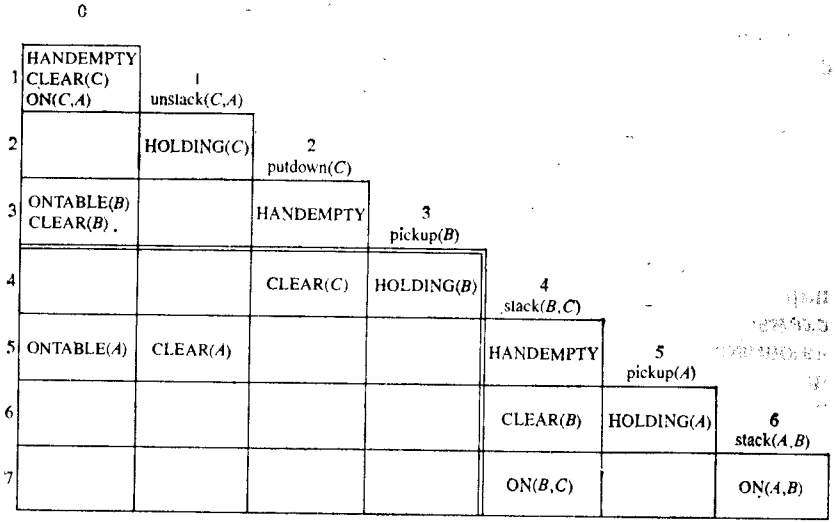


Рис. 10.14. Треугольная таблица.

На рис. 10.14 четвертое ядро обведено двойными линиями. Записи в i-м ядре определяют условия, которые должны соответствовать описанию состояния таким образом, чтобы последовательность, составленная из i-го и последующего операторов, была выполнимой и приводила к цели. Таким образом, первое ядро (т. е. нулевой столбец) содержит те условия начального состояния, которые необходимы последующим операторам и цели; $(N+1)$-е ядро (т. е. $(N+1)$-я строка) содержит условия цели. Эти свойства треугольной таблицы весьма полезны для управления ходом выполнения плана движения робота.

Поскольку планы робота в реальном рабочем пространстве должны выполняться механическими устройствами, система управления должна предусматривать возможность того, что планируемые действия робота не будут реализованы, поскольку механические допуски могут вводить ошибки при реализации плана. Во время выполнения роботом операций незапланированные действия могут неожиданно приблизить нас к цели либо сбить с пути. Эти проблемы можно было бы решить путем генерации нового плана (основанного на модернизированном описании состояний) после каждого действия робота, но очевидно, что такая стратегия была бы слишком дорогой, поэтому мы приведем схему, с помощью которой можно управлять ходом действия робота согласно данному плану.

Ядра треугольных таблиц содержат только информацию, необходимую для реализации такой системы выполнения плана. В начале выполнения плана мы знаем, что полный план выполним и соответствует достижению цели, поскольку предложения в первом ядре соответствуют описанию первоначального состояния, которое было использовано при создании плана. (Здесь мы предполагаем, что рабочее пространство является статическим, т. е. рабочее пространство изменяют только действия самого робота.) Теперь предположим, что система выполнила первые $i-1$ действий последовательности операторов плана. Тогда, чтобы оставшаяся часть плана (состоящая из i-го и последующих действий) была выполнимой и приводила к достижению цели, предложения в i-м ядре должны соответствовать описанию нового текущего состояния. (Мы предполагаем, что во время выполнения плана система очувствления непрерывно модернизирует описание состояний так, чтобы это описание точно моделировало текущее состояние рабочего пространства.) Фактически мы можем не просто проверить, соответствует ли ядро описанию состояния рабочего пространства после действия робота, мы можем выбрать соответствующее ядро с наибольшим номером. Тогда, если непредвиденные эффекты приближают нас к цели, нам необходимо только выполнить оставшиеся действия; и если ошибка выполнения нарушает результаты предыдущих действий, соответствующие действия могут быть выполнены вновь. Чтобы найти соответствующее ядро, мы проверяем ядра в треугольной таблице по мере уменьшения их номеров, начиная с большего (ядро, соответствующее большому номеру, является последней строкой таблицы). Если целевое ядро (последняя строка таблицы) приводит к достижению конечного состояния, то система заканчивает работу. В противном случае предполагаем, что соответствующим ядром с наибольшим номером является ядро с номером i. Тогда мы знаем, что i-й оператор применим к описанию текущего состояния. В этом случае система осуществляет действия, соответствующие i-му оператору, и проверяет результаты, снова отыскивая соответствующее ядро с наибольшим номером. В идеальном рабочем пространстве эта процедура выполняется просто согласно последовательности операторов плана. В реальном же рабочем пространстве эта процедура помогает избежать ненужных действий, а также устранить некоторые виды аварийного завершения путем повторного выполнения соответствующих действий. Перепланирование осуществляется, когда больше нет соответствующих ядер.

В качестве примера, иллюстрирующего этот процесс, рассмотрим снова задачу построения пирамиды из объектов A, B и C и план, представленный в виде треугольной таблицы на рис. 10.14. Предположим, что система выполняет действия, соответствующие первым четырем операторам, и результаты этих действий заранее планируются. Теперь предположим, что система пытается осуществить захват блока A, но программа выполнения (в это время) меняет блок A на блок B, и поэтому вместо блока A робот захватывает блок B. [Снова предполагаем, что система очувствления точно модернизирует описание состояния, добавляя утверждение HOLDING(B) и исключая утверждение ON(B, C); в частности, она не добавляет утверждение HOLDING(A).] Если бы не было ошибки выполнения, шестое ядро было бы теперь соответствующим; в результате ошибки соответствующим ядром с наибольшим номером становится четвертое ядро. Вновь выполняется действие, соответствующее оператору STACK(B, C), и система продолжает работу.

То, что ядра треугольной таблицы частично перекрываются, можно использовать для улучшения поиска соответствующего ядра с наибольшим номером. Начиная с нижней строки, мы просматриваем таблицу слева направо, определяя первый элемент, содержащий предложение, не соответствующее описанию текущего состояния. Если мы просмотрели всю строку и не обнаружили такого элемента, целевое ядро является соответствующим. Если же мы нашли такой элемент в i-м столбце, наибольший номер соответствующего ядра не может превышать i. В этом случае мы устанавливаем границу по i-му столбцу, переходим к следующей снизу строке и начинаем просматривать ее слева направо до столбца i. Если мы находим элемент, содержащий несоответствующее утверждение, мы вновь устанавливаем границу по столбцу, начинаем просматривать следующую строку и т. д. Процесс завершается определением k-го соответствующего ядра (наибольшим номером соответствующего k-й просматриваемой строке и проведением границы по k-му столбцу, соответствующему этой строке).

Пример. Рассмотрим простую задачу переноса ящика мобильным роботом из соседней комнаты. Начальное состояние модели рабочего пространства робота приведено на рис. 10.15. Предположим, что имеются два оператора GOTHRU и PUSHTHRU.

GOTHRU (d, r_1, r_2) Робот двигается через дверь d из комнаты r_1 в комнату r_2

Предусловие: INROOM(ROBOT, r_1) $\wedge$ CONNECTS(d, r_1, r_2) робот в комнате r_1 и дверь d соединяет комнаты r_1 и r_2

Список исключений: INROOM(ROBOT, S) для любого значения S

Список добавлений: INROOM(ROBOT, r_2)

PUSHTHRU(b, d, r_1, r_2) Робот переносит объект b через дверь d из комнаты r_1 в комнату r_2

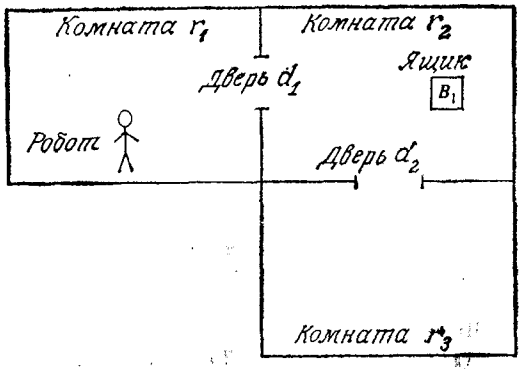


Рис. 10.15. Начальная модель рабочего пространства.

Начальная база данных M_0
INROOM (ROBOT, R_1) — робот в комнате R_1
CONNECTS (D_1, R_1, R_2) — дверь D_1 соединяет комнаты R_1 и R_2
CONNECTS (D_2, R_2, R_3) — дверь D_2 соединяет комнаты R_2 и R_3
BOX (B_1) — ящик B_1
INROOM (B_1, R_2) — ящик B_1 в комнате R_2
$\forall_x \forall_y \forall_z$ [CONNECTS (x, y, z) $\rightarrow$ CONNECTS (x, y, z)]

Цель G_0

$\exists^x$ [BOX (x) $\wedge$ INROOM (x, R_1)]

Различие \ Оператор	GOTHRU	PUSHTHRU
Расположение ящика		V
Расположение робота	V	
Расположение ящика и робота		V

Рис. 10.16. Таблица «действие — оператор».

Предусловие: INROOM(b, r_1) $\wedge$ INROOM(ROBOT, r_1) $\wedge$ $\wedge$ CONNECTS(d, r_1, r_2)

Список исключений: INROOM(ROBOT, S) INROOM(b, S)

Список добавлеиий: INROOM(ROBOT, r_2), INROOM (b, r_2)

Таблица «различие — оператор» приведена на рис. 10.16.

Когда система STRIPS приступает к решению задачи, она сначала пытается достигнуть цели G_0 из начального состояния M_0. Эта задача не может быть решена сразу. Однако, если начальная база данных содержит утверждение INROOM(B_1, R_1) процесс решения задачи может продолжаться. Система STRIPS находит оператор PUSHTHRU(B_1, d, r_1, R_1), действие которого может привести к желаемому утверждению. Предусловием G_1 для оператора PUSHTHRU является следующее:

$$G_1: \text{INROOM}(B_1, r_1) \wedge \text{INROOM}(\text{ROBOT}, r_1) \wedge \wedge \text{CONNECTS}(d, r_1, R_1)$$

В анализе конечных значений это предусловие является подцелью и STRIPS пытается реализовать его из состояния M_0.

Поскольку нет непосредственного решения этой задачи, STRIPS находит, что если $r_1 = R_2$, $d = D_1$ и текущая база данных содержит оператор INROOM(ROBOT, R_2), то процесс может продолжаться. Вновь STRIPS находит оператор GOTHRU(d, r_1, R_2, результат действия которого может привести к желаемому состоянию. Его предусловие является следующей подцелью:

$$G_2: \text{INROOM}(\text{ROBOT}, r_1) \wedge \text{CONNECTS}(d, r_1, R_2)$$

Используя подстановки $r_1 = R_1$ и $d = D_1$, система STRIPS способна выполнить операцию G_2. Применяя оператор GOTHRU(D_1, R_1, R_2) к состоянию M_0, имеем

$$M_1: \text{INROOM}(\text{ROBOT}, R_2), \text{CONNECTS}(D_1, R_1, R_2)$$
$$\text{CONNECTS}(D_2, R_2, R_3), \text{BOX}(B_1)$$
$$\text{INROOM}(B_1, R_2), \ldots$$
$$(\forall x)\ (\forall y)\ (\forall z)\ [\text{CONNECTS}(x, y, z) \Rightarrow \text{CONNECTS}(x, z, y)]$$

Теперь STRIPS пытается достигнуть подцели G_1 из новой базы данных M_1. Она находит оператор PUSHTHRU(B_1, D_1, R_2, R_1) с подстановками $r_1 = R_2$ и $d = D_1$. Применяя этот оператор к M_1, получаем

$$M_2: \text{INROOM}(\text{ROBOT}, R_1), \text{CONNECTS}(D_1, R_1, R_2)$$
$$\text{CONNECTS}(D_1, R_2, R_3), \text{BOX}(B_1)$$
$$\text{INROOM}(B_1, R_1), \ldots$$
$$(\forall x \forall y \forall z)\ [\text{CONNECTS}(x, y, z) \Rightarrow \text{CONNECTS}(x, z, y)]$$

На следующем шаге система STRIPS пытается реализовать исходную цель G_0 из состояния M_2. Эта попытка оказывается успешной, и окончательная последовательность операторов имеет вид:

$$\text{GOTHRU}(D_1, R_1, R_2), \text{PUSHTHRU}(B_1, D_1, R_2, R_1)$$

Желательно создать план, который не зависел бы от констант D_1, R_1, R_2 и B_1 и можно было бы использовать в ситуациях

с произвольными дверями, комнатами, ящиками. Треугольная таблица для конкретного плана приведена на рис. 10.17, а треугольная таблица для обобщенного плана — на рис. 10.18. Следовательно, обобщенный план мог бы иметь следующий вид:

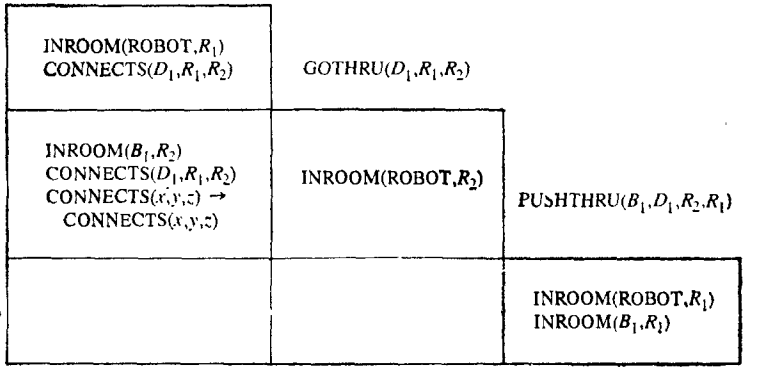


Рис. 10.17. Треугольная таблица для конкретного плана.

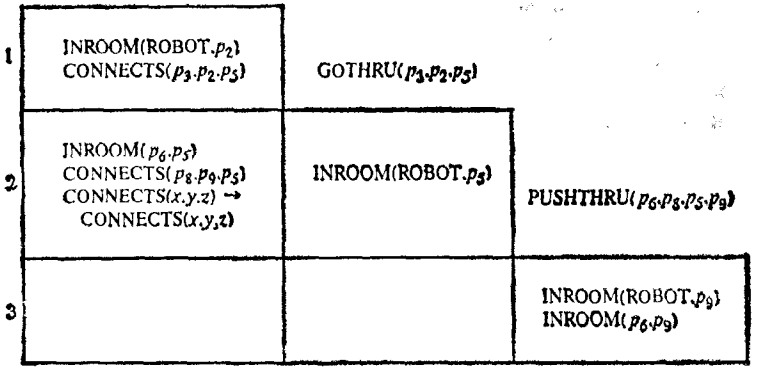


Рис. 10.18. Треугольная таблица для обобщенного плана.

$$\text{GOTHRU}(d_1, r_1, r_2)$$
$$\text{PUSHTHRU}(b, d_2, r_2, r_3)$$

и мог быть использован для перехода из первой комнаты во вторую и переноса ящика в третью комнату.

10.7. ОБУЧЕНИЕ РОБОТА

Мы рассмотрели применение треугольных таблиц обобщенных планов для управления ходом выполнения плана действий робота. Треугольные таблицы для обобщенных планов также

могут быть использованы системой STRIPS для выделения соответствующей последовательности операторов во время последовательного процесса планирования. В принципе можно составить единую треугольную таблицу для представления семейства обобщенных операторов. Из списка добавлений, определяемого системой STRIPS, мы должны выбрать наиболее экономичный по параметрам оператор. Напомним, что $(i+1)$-я строка треугольной таблицы (исключая первый элемент) представляет собой список добавлений $A_1, \ldots, {}_i$ i-й части плана, т. е. последовательность $OP_1, \ldots, OP_i$. n-й шаг плана дает системе STRIPS n альтернативных списков добавлений, каждый из которых может быть использован для уменьшения различия, возникающего во время нормального процесса планирования. Система STRIPS обычным образом проверяет соответствие каждого из списков добавлений обобщенного плана и выбирает из них те, которые обеспечивают наибольшее уменьшение различия. Часто данный набор подходящих утверждений будет появляться в нескольких строках таблицы. В этом случае выбирается строка с наименьшим номером, поскольку данный выбор приводит к наиболее короткой последовательности операторов, с помощью которых создаются требуемые утверждения.

Предположим, что система STRIPS выбирает i-й список добавлений $A_1 \ldots, A_i$, $i < n$. Поскольку этот список является результатом работы последовательности операторов $OP_1, \ldots, \ldots, OP_i$, то очевидно, что не надо интересоваться их применением и тем более установлением их предусловий. Как правило, некоторые шаги плана требуются только для установления предусловий последующих шагов. Если нас не интересует завершающая часть плана, то в соответствующем примере обобщенного плана могут не содержаться те операторы, единственной целью которых является определение предусловий для завершающей части плана. Поэтому обычно система STRIPS использует ограниченный набор из списка $A_1, \ldots, A_i$ для установления соответствия i-й начальной части плана. Каждый из первых i операторов, которые не добавляют утверждений в этот ограниченный набор или помогают установить предусловия для некоторого оператора, добавляющего утверждение в этот набор, не требуется в соответствующем примере обобщенного плана.

Для того чтобы получить систему планирования действий робота, которая может не только ускорять процесс планирования, но также улучшать способность системы решать более сложные задачи, можно было создать систему, имеющую способность к обучению. Система STRIPS использует схему общих правил машинного обучения. Другой формой обучения могло бы быть обновление информации в таблице «различие — оператор» на основе опыта системы.

Мощным средством является обучение по аналогии, которое также применяется для планирования действий робота. Обучающаяся система планирования действий робота, названа PULP-I, была предложена в работе [279]. Система использует аналогию между текущей незапланированной задачей и любыми известными сходными задачами для сокращения поиска решения. Для внутреннего представления задач в этой системе вместо логики предикатов используется семантическая сеть. Вначале в качестве знаний, основанных на простом опыте, в систему заводится ряд примеров типовых задач. Для определения сходства между двумя задачами используется аналогия, которая определяется семантической процедурой соответствия. Алгоритм соответствия определяет семантическую «близость», самую маленькую величину (самое близкое значение). Основанный на семантическом определении соответствия прошлый опыт применяется для формирования плана-кандидата. Затем проверяются предусловия операторов каждого плана-кандидата на применимость их к текущему состоянию рабочего пространства. Если план не применим, он исключается из списка кандидатов. В результате проверки на применимость может оказаться несколько возможных планов-кандидатов. Эти планы-кандидаты упорядочиваются согласно их оценкам семантических соответствий. План с наименьшим значением семантического соответствия имеет высший приоритет и должен располагаться в начале списка кандидатов. Конечно, если ни один кандидат не найден, система заканчивает работу аварийным остановом. Моделирование с помощью системы PULP-I на ЭВМ показало значительное улучшение процесса планирования, которое касается не только скорости планирования, но и способности формирования сложных планов на основе типовых обучающих примеров.

10.8. ПЛАНИРОВАНИЕ ЗАДАЧИ ДВИЖЕНИЯ РОБОТОВ

Предназначенные для роботов планировщики, рассмотренные в предыдущем разделе, требуют описания только начального и конечного состояний данной задачи. Эти системы планирования обычно не определяют подробно движения робота, необходимые для выполнения операции, а формируют такие команды, как PICKUP(A) и STACK(X, Y), не определяя траекторию движения робота. Однако в обозримом будущем планировщики задач должны давать более подробную информацию о промежуточных состояниях, чем та, которую сейчас эти системы выдают. От них будет требоваться создание намного более подробной программы движения робота. Другими словами, планировщик задачи должен быть способен преобразовать детализацию на уровне задачи в детализацию на уровне манипулятора. Для выполнения этого преобразования планировщик задачи

должен иметь описания объектов, подлежащих манипулированию, рабочего пространства, где выполняется задача, схемы выполнения роботом задачи, начального состояния рабочего пространства и желаемого конечного (целевого) состояния. На выходе планировщика задачи должна быть программа достижения роботом конечного состояния из определенного начального состояния.

Планирование задачи осуществляется в 3 этапа: моделирование, детализация задачи и синтез программы движения манипулятора. Модель рабочего пространства, в котором выполняется задача, должна содержать следующую информацию·

1) геометрическое описание всех объектов и робота;
2) физическое описание всех объектов;
3) кинематическое описание всех связей;
4) описание робота и характеристик сенсоров.

Модели состояний задачи должны включать взаимное расположение всех объектов и связей в модели рабочего пространства.

10.8.1. Моделирование

Геометрическое описание объектов является важной частью модели рабочего пространства. Основным источником информации о геометрических моделях являются системы автоматизированного проектирования (САПР) и техническое зрение. Имеются три основных типа схем представления трехмерных объектов [247]:

1) граничное представление;
2) шаблонное представление;
3) объемное представление.

Объемное представление объекта может задаваться тремя способами:

1) занимаемой пространственной областью;
2) разбиением на элементарные ячейки;
3) пространственной геометрией воспроизведения объектов (ПГВО).

Для задач планирования была предложена пространственная геометрия воспроизведения объектов, основной идеей которой является построение сложных тел из нескольких типов простых с помощью набора операций. Объект на рис. 10.19, *а* может быть описан с помощью структуры, приведенной на рис. 10.19, *б*. Возможные движения объекта ограничиваются наличием других объектов в рабочем пространстве, и форма ограничений зависит от формы объектов. По этой причине планировщику задачи необходимы геометрические описания объектов. Имеются дополнительные ограничения на движение, накладываемые кинематической структурой самого робота. Кинематические модели дают

планировщику задачи информацию, требуемую для планирования движений манипулятора. Эта информация является внешним ограничением.

Многие физические характеристики объектов играют важную роль при планировании операций робота. Например, от массы и моментов инерции зависит скорость передвижения деталей или величина силы, которую надо приложить, чтобы их поднять и перенести. Другим важным аспектом является очувствление. Для планирования задачи зрение дает возможность

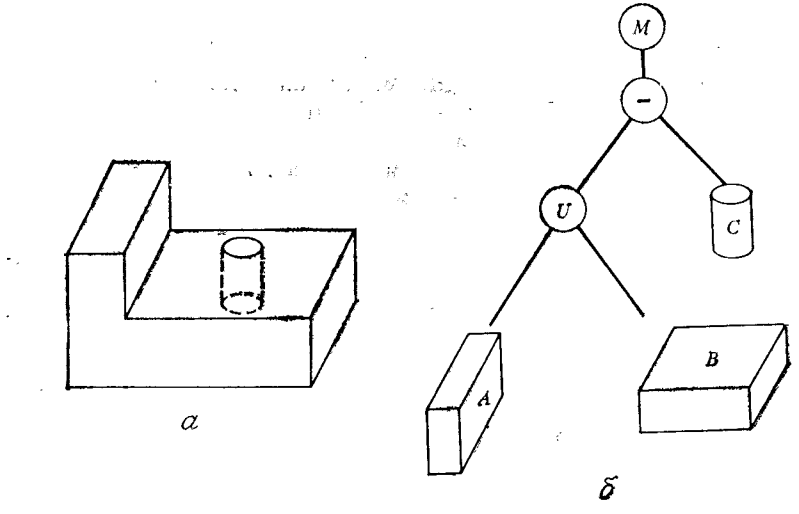


Рис. 10.19. Пространственная геометрия воспроизведения объектов.
Атрибуты *A* и *B*: длина, ширина, высота
Атрибуты *A* и *B*: длина, ширина, высота. Атрибуты *C*: радиус, высота. Набор операторов-связей: ∪ — объединение, ∩ — пересечение, — разность.

роботу с некоторой точностью получать информацию о расположении объектов; силовое очувствление допускает использование упругой податливости; тактильное очувствление позволяет реализовать обе возможности. Помимо очувствления необходимо описывать много индивидуальных характеристик манипуляторов, например граничные значения скорости и ускорения и точность позиционирования каждого из сочленений.

10.8.2. Подробное описание задачи

Состояние модели определяется взаимным расположением (конфигурациями) всех объектов в рабочем пространстве; задачи обычно определяются последовательностью состояний модели рабочего пространства. Для определения взаимного расположения можно использовать, во-первых, САПР (для распо-

ложения моделей объектов в желаемые конфигурации), во-вторых, робот (для определения его конфигураций и расположения частей объектов) и, в-третьих, пространственные символические связи между частями объекта (для введения ограничений на конфигурации объектов). Первые две возможности позволяют создавать численные описания конфигураций, которые трудны для интерпретации и модификации. В последнем случае конфигурация описывается набором символических пространственных связей, которые необходимо соблюдать между объектами в этой конфигурации.

Поскольку состояниями модели являются наборы конфигураций и подробное описание задачи определяется последовательными состояниями, выражаемыми символическими пространственными связями конфигураций, то мы в состоянии определять задачи. Предположим, что в модель входят имена и свойства объектов. Первым шагом процесса планирования задачи является преобразование символических пространственных связей между свойствами объектов в уравнение параметров конфигураций объектов в этой модели. Затем эти уравнения должны быть насколько возможно упрощены, чтобы можно было определить ранги конфигураций всех объектов. Во время синтеза программы также используется символическая форма связей.

10.8.3. Синтез программы движения манипулятора

Синтез программы манипулятора на основе подробного описания задачи является решающим этапом планирования. Основные шаги этого этапа следующие: планирование захвата, планирование движения и обнаружение ошибок. Программа, полученная в результате синтеза, состоит из команд захвата, нескольких видов определения движения и тестов на ошибки. Обычно эта программа составлена на роботоориентированном языке для определенного манипулятора и допускает многократное использование без предварительного перепланирования.

10.9. ОСНОВНЫЕ ПРОБЛЕМЫ ПЛАНИРОВАНИЯ ЗАДАЧ

10.9.1. Символические пространственные связи

Основными этапами получения из символических пространственных связей ограничений на конфигурации являются:

1) Определение системы координат для объектов и частей объектов.

2) Определение уравнений параметров конфигурации объекта по каждой пространственной связи.

3) Объединение уравнений для каждого объекта.

4) Решение уравнений для параметров конфигурации каждого объекта.

Рассмотрим пример, приведенный на рис. 10.20:

PLACE Block 1 (f_1 against f_3) and (f_2 anainst f_4)

(разместить блок 1 так, чтобы грань f_1 располагалась напротив грани f_3 и грань f_2 — напротив грани f_4).

Требуется получить набор уравнений, которые связывают конфигурацию блока 1 с известной конфигурацией блока 2, т. е. грань f_1 блока 2 должна быть напротив грани f_3 блока 1, а грань f_2 блока 2 — напротив грани f_4 блока 1.

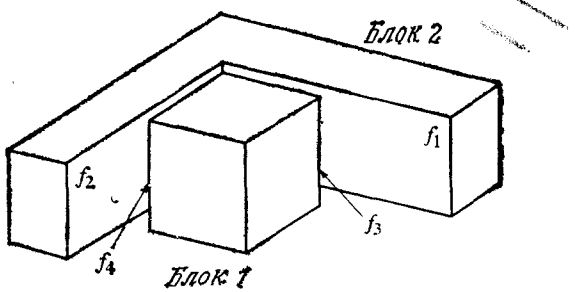


Рис. 10.20. Иллюстрация пространственной связи между объектами.

Каждый объект и его части имеют связанные с ним системы координат (рис. 10.21). Конфигурации описываются матрицами преобразования размерностью 4×4:

$$f_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix},$$

$$f_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad f_4 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Пусть $\mathrm{twix}(\theta)$ — матрица преобразования, описывающего поворот на угол θ вокруг оси x, $\mathrm{trans}(x, y, z)$ — матрица перемещения вдоль осей x, y и z и M — матрица поворота вокруг оси y, причем $M = M^{-1}$. Каждая связь типа *напротив* между двумя гранями, например гранью f объекта A и гранью g объекта B, дает следующее ограничение на конфигурацию двух объектов:

$$A = f^{-1} M \,\mathrm{twix}(\theta)\,(0,\ y,\ z)\, gB. \qquad (10.9\text{-}1)$$

Две связи типа *напротив* в примере на рис. 10.20 дают следующие уравнения

$$\text{Block1} = f_3^{-1} M\, \text{twix}(\theta_1)\, \text{trans}(0,\ y_1,\ z_1)\, f_1\, \text{Block2},$$
$$\text{Block1} = f_4^{-1} M\, \text{twix}(\theta_2)\, \text{trans}(0,\ y_2,\ z_2)\, f_2\, \text{Block2}. \quad (10.9\text{-}2)$$

Уравнение (10.9-2) состоит из двух независимых ограничений на конфигурацию блока 1, которые должны одновременно

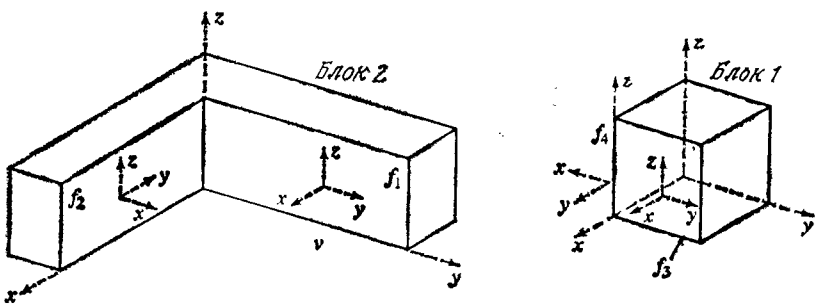


Рис. 10.21. Оси координат, связанные с объектами и их частями.

выполняться. Приравнивая оба уравнения (10.9-2) и устраняя общий член Block 2, мы получаем

$$f_3^{-1} M\, \text{twix}(\theta_1)\, \text{trans}(0,\ y_1,\ z_1)\, f_1 = f_4^{-1} M\, \text{twix}(\theta_2)\, \text{trans}(0,\ y_2,\ z_2)\, f_2. \quad (10.9\text{-}3)$$

Выражение (10.9-3) можно преобразовать следующим образом:

$$f_3^{-1} M\, \text{twix}(\theta_1)\, \text{trans}(0,\ y_1+1,\ z_1+1)\, (f_2')^{-1} \times$$
$$\times\, \text{trans}(0,\ -y_1,\ -z_2)\, \text{twix}(-\theta_2)\, M^{-1} f_4 = I, \quad (10.9\text{-}4)$$

где главные матрицы обозначают вращательный компонент преобразования, полученный подстановкой в последнюю строку матрицы [0, 0, 0, 1]. Можно показать, что для компоненты вращения и перемещения уравнения этого типа могут быть решены независимо. Уравнение вращения может быть получено заменой каждой из матриц trans на единичную и использованием только вращательных компонент других матриц. Уравнение вращения для уравнения (10.9-4) имеет вид

$$(f_3')^{-1} M\, \text{twix}(\theta_1)\, (f_2')^{-1}\, \text{twix}(-\theta_2)\, M\, (f_4') = \text{I}. \quad (10.9\text{-}5)$$

Поскольку $f_3' = \text{I}$, уравнение (10.9-5) можно записать в виде

$$\text{twix}(\theta_1)\, (f_2')^{-1}\, \text{twix}(-\theta_2) = M\, (f_4')^{-1} M, \quad (10.9\text{-}6)$$

так как

$$(f_2') = M\, (f_4')^{-1} M = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Равенство (10.9-6) выполняется, и мы можем выбрать $\theta_2 = 0$. Полагая $\theta_2 = 0$ в уравнении (10.9-6), мы получаем

$$\text{twix}(\theta_1) = M\, (f_4')^{-1} M\, (f_2') = \text{I}. \quad (10.9\text{-}7)$$

Из уравнения (10.9-7) следует, что $\theta_1 = 0$. Таким образом, уравнение 10.9-2 имеет вид

$$\text{Block1} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & y_1 & 2+z_1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 2-y_2 & 0 & 2+z_2 & 1 \end{bmatrix}. \quad (10.9\text{-}8)$$

Сравнивая соответствующие элементы матрицы, имеем

$$2 - y_2 = 1,$$
$$y_1 = 0,$$
$$2 + z_1 = 2 + z_2.$$

Следовательно, $y_2 = 1$, $y_1 = 0$ и $z_1 = z_2$, т. е. блок 1 имеет одну степень свободы, соответствующую перемещению вдоль оси z.

Метод, используемый в предыдущем примере, был предложен в работе [5]. Контактные связи, применяемые в этом методе, делятся на связи типа *напротив*, *соответствовать* и *параллельность* для деталей, имеющих плоские или сферические поверхности, цилиндрические стержни и отверстия, острые края и вершины. В работе [281] этот подход распространен на неконтактные связи, такие, как стержень и отверстие с диаметром большим, чем диаметр стержня, объект в ящике или деталь рядом с другой деталью. Эти связи повышают число ограничений на параметры конфигурации. Например, они могут быть использованы для моделирования положения конца отвертки в схвате робота, ошибок положения в сочленениях и проскальзывания отвертки в схвате. После упрощения равенств и неравенств определяется набор линейных ограничений путем линеаризации вращений вблизи исходной конфигурации. Значения параметров

конфигурации, удовлетворяющей ограничениям, могут быть получены методом линейного программирования применительно к линеаризованным уравнениям ограничений.

10.9.2 Обход препятствий

Наиболее общими движениями робота являются движения связанные с переносом объектов, для которых единственное ограничение состоит в том, чтобы робот и переносимый объект не столкнулись с объектами в рабочем пространстве. Поэтому для планирования задачи важно планировать движения с учетом обхода препятствий. В разных областях были предложены различные алгоритмы обхода препятствий. Ниже мы кратко изложим эти алгоритмы, предназначенные для обхода роботом препятствий в трехмерном пространстве. Эти алгоритмы можно разделить на 3 класса: 1) гипотеза — тест; 2) штрафная функция; 3) явно свободное пространство.

Раньше других был предложен метод гипотезы и теста, состоящий из трех основных шагов:

1. Предлагается гипотеза относительно пути-кандидата между начальной и конечной конфигурациями манипулятора робота.
2. Набор конфигураций вдоль этого пути тестируется на возможность столкновений.
3. Если столкновение оказывается возможным, с целью определения пути обхода исследуется препятствие, которое может вызвать это столкновение. Весь процесс повторяется для модифицированного движения.

Главное преимущество метода «гипотеза — тест» заключается в его простоте. Основными вычислительными операциями метода являются определение возможных столкновений и модификация путей для предотвращения столкновений. Первая операция равнозначна способности определять ненулевое геометрическое пересечение между моделями манипулятора и препятствий. Эта возможность, как правило, имеется в большинстве систем геометрического моделирования. В разд. 10.8 мы указывали, что вторая операция — модифицирование предложенного пути — может быть очень трудной. Обычные предложения модификации пути основываются на аппроксимации препятствий в виде замкнутых сфер. Эти методы хорошо работают, когда препятствия расположены редко. Когда же пространство заполнено объектами, попытки избежать столкновения с одним препятствием обычно приводят к столкновению с другим. В таких условиях определение возможных столкновений может быть осуществлено более точно, если применить систему технического зрения и/или другие системы очувствления.

Второй класс алгоритмов обхода препятствий основывается на определении штрафной функции для конфигурации манипулятора, с помощью которой кодируется наличие объектов. Обычно для конфигурации, которая приводит к столкновениям, значение штрафа равно бесконечности и резко падает по мере увеличения расстояния от препятствий. Полная штрафная функция представляет собой сумму штрафов отдельных препятствий, к которой иногда добавляется штрафной член отклонения от кратчайшего пути. Для каждой конфигурации мы можем вычислить значение штрафной функции и оценить ее частные

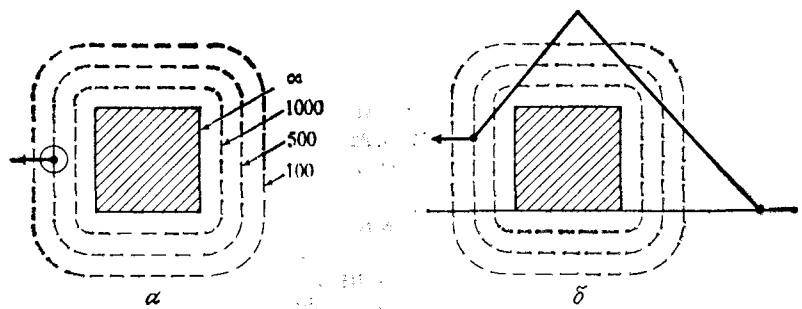


Рис. 10.22. Штрафная функция для простого робота с одной вращательной степенью свободы (*а*) и для двузвенного манипулятора (*б*). (Числа на рисунке указывают значения штрафной функции.)

производные по отношению к параметрам конфигурации. На основе этой локальной информации с помощью функции поиска пути необходимо выбрать последовательность конфигураций. Решение должно соответствовать локальному минимуму штрафной функции, т. е. компромиссу между увеличением длины пути и максимальным приближением к препятствиям. Интерес к методам штрафных функций основан на том, что они обеспечивают относительно простой путь комбинирования ограничений от различных объектов. Однако это справедливо лишь для робота с вращательной степенью свободы; в этом случае вид штрафной функции легко получить исходя из формы препятствия. Для большинства реальных роботов, таких, как двузвенный манипулятор, штрафную функцию можно было бы определить через преобразование конфигурации препятствия в пространстве. В противном случае движения робота, уменьшающие значения штрафной функции, не были бы безопасными. Различие между этими двумя типами штрафных функций иллюстрируется на рис. 10.22. Отметим, что на рис. 10.22, *а* движение вдоль умень-

шающихся значений штрафной функции является безопасным, в то же время на рис. 10.22, *б* подобное движение конца руки манипулятора приводит к столкновению.

Метод, описанный в работе [144], является промежуточным и использует штрафную функцию, которая удовлетворяет определению потенциального поля. Градиент поля в точке робота интерпретируется как сила отталкивания в этой точке, которая суммируется с силой притяжения, действующей из конечного положения робота. Движение робота является результатом взаимодействия этих двух сил при наличии кинематических ограничений. Используя несколько точек робота, можно избежать ситуаций, приведенных на рис. 10.22.

Основным недостатком использования штрафных функций для планирования безопасных путей является точная локальная информация, которую они дают для поиска пути. Попытки определения пути по локальным минимумам штрафной функции могут привести к тупиковым ситуациям, когда дальнейший поиск пути становится невозможным. В этом случае алгоритм должен выбрать предыдущую конфигурацию и продолжить поиск в другом направлении. Такие точки возврата трудны для идентификации на основе локальной информации. Поэтому полезно комбинировать метод штрафных функций с более общим методом гипотеза — тест. Штрафные функции более удобны в в тех случаях, когда требуются только небольшие модификации известного пути.

Третий класс алгоритмов обхода препятствий в явном виде определяет наборы конфигураций робота, которые свободны от столкновений с объектами (*свободное пространство*). Обход препятствия трактуется здесь как поиск пути в пределах этих наборов, связывающего начальную и конечную конфигурации. Эти алгоритмы в основном различаются самими наборами, составляющими свободное пространство, и путями представления этих наборов. Преимущество методов свободного пространства состоит в том, что они используют явную характеристику свободного пространства, позволяющего определить метод поиска, который гарантирует нахождение путей в том случае, если они существуют, в пределах известного набора конфигураций из свободного пространства. Более того, эти методы часто позволяют определить самый короткий из всех безопасных путей. К недостаткам метода относится то, что процесс вычисления свободного пространства может быть длительным. В частности, другие методы могут быть более эффективными в случае пространства с небольшим числом объектов. Однако для пространства, заполненного большим числом объектов, другие методы либо оказываются неработоспособными, либо слишком трудоемкими.

10.9.3. Планирование захвата

Типичная операция робота начинается с захвата объекта: остальные операции зависят от выбора, сделанного во время захвата. Существует несколько вариантов выбора конфигураций захвата объекта без столкновений, однако другие аспекты общей проблемы планирования движений захвата разработаны слабо. В этом разделе термин *объект-цель* относится к объекту, который должен быть схвачен. Поверхности робота, используемые для захвата (например, внутренние стороны пальцев), называются *поверхностями захвата*. Конфигурация манипулятора, в которой он производит операцию захвата объекта, называется *начальной конфигурацией захвата*. Конфигурация, в которой робот переносит объект-цель к месту его назначения, называется *конечной конфигурацией захвата*.

Существуют три основных критерия выбора конфигурации захвата для объектов с известными конфигурациями. Первым критерием является *безопасность*: робот должен обеспечить безопасность в начальной и конечной конфигурациях захвата. Вторым критерием — *достижимость*: робот должен быть способен достичь начальной конфигурации захвата и с объектом в руке найти свободный от столкновений путь к конечной конфигурации захвата. Третьим критерием является *надежность*: захват должен быть *надежным* при наличии сил, действующих на захватываемый объект во время движений, перемещений и некоторых других операций. Если начальная конфигурация объекта-цели является существенно неопределенной, дополнительным критерием является *определенность*: двнжение при захвате должно уменьшить неопределенность положения объекта цели.

Выбор безопасных и достижимых конфигураций манипулятора, обеспечивающий захват, относится к проблеме обхода препятствий. Однако при решении этой задачи следует иметь в виду, что при планировании захвата определяется одна конфигурация, а не траектория.

Кроме того, при планировании захвата необходимо подробно рассматривать взаимодействие манипулятора и объекта-цели. Отметим, что возможными конфигурациями, обеспечивающими захват, являются конфигурации, при которых руки захватного устройства находятся в контакте с объектом-целью и одновременно выполняется условие предотвращения столкновений между манипулятором и другими объектами. И наконец, при планировании процесс выбора конфигурации маиипулятора, обеспечивающей захват, связан с ограничениями, накладываемыми последующими операциями при захвате объекта. Поэтому большинство существующих методов планирования захвата рассматривается независимо от проблемы обхода препятствий.

Большинство подходов к выбору безопасного захвата основано на следующем методе:

1. Выбрать набор кандидатов конфигураций захвата исходя из геометрии объектов, стабильности или уменьшения неопределенности. Для схватов с параллельными зажимами обычно выбирается та конфигурация захвата, которая размещает схваты в контакте с параллельными поверхностями объекта-цели. Критерием при выборе поверхностей является минимизация вращающих моментов вокруг оси между зажимами схвата.

2. Затем набор кандидатов конфигураций захвата сокращается за счет устранения тех конфигураций, которые недостижимы роботом или приводят к столкновениям. Существующие подходы к планированию захвата отличаются главным образом ограничениями на предотвращение столкновений:

а) возможные столкновения схвата и граничных объектов при начальной конфигурации захвата;

б) p-выпуклости (вся масса тела вблизи геометрического центра находится с одной стороны определенной плоскости);

в) существование свободного от столкновений пути к начальной конфигурации захвата;

г) возможны столкновения всего манипулятора и соседних объектов при начальной конфигурации захвата, столкновения схвата и соседних объектов при конечной конфигурации захвата, столкновения всего манипулятора и соседних объектов при конечной конфигурации захвата и существование свободных от столкновений путей из начальной конфигурации захвата к конечной.

3. После сокращения набора кандидатов выбор конфигурации может быть любым. Одной из возможностей служит выбор конфигурации, ведущей к наиболее стабильному захвату. Другой возможностью является выбор конфигурации, имеющей наименьшую вероятность столкновения при наличии ошибки положения или неопределенности.

Легко видеть, что информация очувствления (зрительная, тактильная, вращающий момент или сила) весьма полезна для определения стабильной и свободной от столкновений конфигурации захвата.

10.10. ЭКСПЕРТНЫЕ СИСТЕМЫ И СИСТЕМЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Большинство методов в области искусственного интеллекта далеко не полностью охватывают мыслительную деятельность человека и даже животного. Компьютерные системы распознавания образов, отдельных звуков и речи имеют пока весьма скромные успехи. Однако в одном из направлений искусственного интеллекта, связанном с получением новых знаний из старых в ограниченной предметной области, компьютерные программы могут не только приближаться к результатам работы человеческого мозга, но в некоторых случаях превосходят его. Эти программы используют набор фактов, правил просмотра и другие знания из данной предметной области, которые совместно с методами применения этих правил позволяют из старых знаний получать новые знания.

Они решают задачи в таких специализированных областях, как медицинская диагностика, добыча полезных ископаемых и эксплуатация нефтяных скважин. Они существенно отличаются от обычных компьютерных программ, поскольку их задачи не имеют алгоритмического решения, и, кроме того, эти программы часто должны делать заключения, основанные на неполной или неопределенной информации. Разрабатывая такого рода системы, исследователи обнаружили, что именно от накопления большого количества знаний, а не от сложных алгоритмов вывода зависит производительность системы.

10.10.1. Построение экспертной системы

В настоящее время не во всех областях знаний можно построить экспертные системы. Для задачи «представления знаний» должны быть выполнены следующие условия:

1. Должен быть по меньшей мере один человек-эксперт, который хорошо понимает задачу.

2. Основным источником деятельности эксперта служат специальные знания, оценки и опыт.

3. Эксперт должен быть способен четко сформулировать эти специальные знания, оценки и опыт, а также объяснить методы, которые надо применить при решении данной задачи.

4. Задача должна иметь четко ограниченную область применения.

Иногда экспертные системы можно строить, не следуя точно указанным выше условиям. Например, в постановке задачи могут участвовать несколько экспертов.

Структура экспертной системы является модульной. Факты и другие знания о данной области отделяются от процедур вывода или структуры управления, предназначенной для применения этих фактов, в то время как другая часть системы — глобальная база данных — является моделью «мира», связанного с решаемой задачей, с его статусом и историей. Желательно (хотя это еще не является общепринятым) иметь интерфейс на естественном языке, который облегчает как разработку системы, так и ее применение в данной области. В некоторые сложные системы входит модуль, дающий разъяснения о принятых системой решениях и позволяющий пользователю оспаривать ре-

шения системы и исследовать лежащие в основе процессы рассуждений, ведущие к этим решениям.

Экспертные системы отличаются от большинства обычных компьютерных программ несколькими важными аспектами. В обычных компьютерных программах знания и методы для использования этих знаний заложены таким образом, что изменить программу весьма сложно. В экспертной системе обычно имеется четкое разделение между общими знаниями о задаче (база знаний) и информацией о текущей задаче (входные данные), а также методами (правилами вывода), необходимыми для применения общих знаний к конкретной задаче. В результате программу можно видоизменять путем простой модификации базы знаний. Это особенно верно для систем, основанных на правилах, где система может быть изменена путем простого добавления или извлечения правил из базы знаний.

10.10.2. Системы, основанные на правилах

Наиболее распространенным подходом представления знаний предметной области (как фактов, так и эвристик), необходимых для экспертной системы, являются производящие правила (относящиеся к так называемым правилам СИТУАЦИЯ—ДЕЙСТВИЕ или ЕСЛИ—ТОГДА). Приведем простой пример производящего правила: ЕСЛИ источник питания на «Спейс шаттл» отказал И имеется в наличии запасной, И причина неисправности на первом источнике существует недолго, ТОГДА включить запасной источник питания. Работа системы, основанной на правилах, состоит в применении правил, анализе результатов и применении новых правил к изменившейся ситуации. Эти системы могут использовать прямой логический вывод, начинать с начальной ситуации и идти вперед к решению, либо начинать с гипотез решений и идти назад к начальной ситуации, или применять метод дедукции, начиная с начальной ситуации и основываясь на определенных гипотезах.

Одной из наиболее ранних и наиболее часто используемых экспертных систем является система Dendral [13]. Она разработана в конце 1960 г. Эдвардом Фейгенбаумом и Джошуа Ледербергом в Станфордском университете для генерации правдоподобного структурного представления органических молекул на основе данных масс-спектрального анализа. Система обеспечивает:

1) получение ограничений на основе данных;
2) создание структур кандидатов;
3) рекомендации на основе спектрограмм для кандидатов;
4) сравнение результатов с данными.

Эта система, основанная на правилах и строящая молекулы исходя из данных, иллюстрирует весьма общий в искусственном

интеллекте подход к решению задач, называемый «генерация и тест». Больше 15 лет химики-органики использовали систему Dendral в качестве консультанта. В настоящее время она признана как эксперт в области масс-спектрального анализа.

Одной из наиболее известных экспертных систем является также система MYCIN [13], созданная Эдвардом Шортлифом в Станфордском университете в середине 70-х годов. Она является диалоговой системой, диагностирующей виды бактериальной инфекции и рекомендующей терапию из антибиотиков. MYCIN выдает экспертную оценку в виде правил «условие — заключение», связывая данные о пациенте с гипотезами об инфекции, и в то же время дает «экспертную достоверность» каждого правила. Она идет от предполагаемых диагнозов, чтобы оценить факторы достоверности заключений, основанные на факторах достоверности их предшественников с целью проверки диагноза. Если для уменьшения числа гипотез не хватает информации, система запрашивает у врача дополнительные данные, исключающие применение всех гипотез. По окончанию работы MYCIN выдает курс лечения применительно ко всем диагнозам, которые имеют высокую степень достоверности.

Другая основанная на правилах система R1 успешно применялась для разработки конфигураций компьютерных систем VAX на основе заказов потребителей, включающих требование на систему и вспомогательное оборудование. Первая версия R1 была разработана Джоном Мак-Дермоттом в 1979 г. в Университете Карнеги—Меллона для фирмы Digital Equipment Corp. Используя принципы системного подхода, R1 разбивает исходную задачу на следующие подзадачи:

1) исправление ошибок в заказе;
2) размещение плат и других компонентов в стойках центрального процессора;
3) размещение модулей в стойках общей шины и размещение компонентов в модулях;
4) размещение распределительных щитов в стойках общей шины;
5) составление плана расположения (размещения) системы;
6) осуществление проводки кабелей.

На каждом шаге разработки конфигурации для дальнейших действий обычно пригодны несколько правил. Из возможных применяемых правил R1 выбирает правило, имеющее наибольшее число предложений ЕСЛИ при условии, что это правило наиболее соответствует текущей ситуации. (R1 написана на языке OPS 5, специально предназначенном для обработки производящих правил.) В настоящее время система имеет около 1200 правил для VAX вместе с информацией о нескольких 1000 компонентов VAX. Вся система имеет около 3000 правил и систему знаний о компонентах PDP-11 и VAX.

10.10.3. Замечания

Области применения экспертных систем включают медицинскую диагностику и определение курса лечения, автоматизацию медицинских знаний, интерпретацию химических данных, химический и биологический синтез, разработку нефти и полезных ископаемых, планирование и составление расписаний, интерпретацию сигналов, оценку военной угрозы, тактическое планирование, космическую оборону, управление воздушным движением, анализ целей, СБИС-конструирование, оценку повреждений в различных структурах, диагностику неисправного оборудования, выбор конфигурации компьютерных систем, распознавание речи, консультации с помощью компьютеров, доступ к базам знаний и различного рода управление, планирование и контроль процесса производства и проектирование экспертных систем.

Однако существуют ограничения на использование экспертных систем. Поэтому меняются методы их проектирования и конструирования. Пределы возможностей систем, основанных на правилах, очевидны: не все знания могут быть структурированы на основе эмпирических ассоциаций. Такие ассоциации не способствуют выявлению причинных связей и также не соответствуют выдвигаемым на первый план структурам, системам знаний и их функциям. Очень мало экспертных систем, которые содержат знания о причинности и структуре. Подобные системы перспективны и могут давать правильные ответы достаточно часто, что делает возможным их применение в автономных системах. Однако они не смогут все-таки заменить человека.

Существует тенденция конструировать системы, не основанные на производящих правилах и использующие семантические сети, фреймы и другие структуры представления знаний, часто более пригодные для причинного моделирования. Обеспечивая более соответствующее представление знаний для решения конкретной задачи, эти системы также упрощают процедуру вывода. Некоторые экспертные системы используют подход «черной доски», комбинирование основанных и не основанных на правилах различных частей системы, для получения решения наилучшим способом, причем каждый сегмент программы осуществляет свою собственную экспертизу.

10.11. ЗАКЛЮЧЕНИЕ

В этой главе рассмотрены вопросы решения задач планирования применительно к роботам. Планировщик пытается найти путь из начального состояния рабочего пространства робота в конечное состояние. Путь состоит из последовательности операций, считающихся элементарными для системы. Решение задачи

может быть основано на соответствующей последовательности реальных действий в реальном рабочем пространстве. Планирование должно рассматриваться как интеллектуальная функция робота.

В конце 1971 и начале 1972 г.г. были предложены два основных подхода к планированию действий робота. Первый подход, типичным представителем которого является система STRIPS, заключается в создании для робота универсального планировщика, способного решать задачи в различных рабочих пространствах. Второй подход состоит в выборе определенного рабочего пространства и в написании применительно к нему компьютерной программы решения задач. Первый подход, подобный многим другим процессам решения задач в искусственном интеллекте, обычно требует большого объема вычислений для решения задач, возникающих при функционировании робота в реальном рабочем пространстве, и, следовательно, чрезвычайно трудно реализуем с вычислительной точки зрения. Второй подход имеет общий недостаток, состоящий в том, что для каждых действий робота необходимо писать свой набор программ, что значительно ограничивает гибкость функционирования робота в реальном рабочем пространстве.

Помимо планирования задач робота на высоком уровне требуется более подробная количественная информация о рабочем пространстве робота. Существующие методы планирования задач непригодны с вычислительной точки зрения для практических применений в реальном времени. Требуются более мощные и эффективные алгоритмы. Кроме того, для ускорения вычислений и удовлетворения требованиям реального времени могут быть использованы специализированные компьютеры.

Планирование действий робота, подразумевающее интеллектуальные действия и способность решения задач, все еще является активной сферой исследований. Для функционирования робота в реальном времени до сих пор необходимы мощные и эффективные алгоритмы планирования, которые будут реализованы на высокоскоростных специализированных компьютерных системах.

Литература

Общие вопросы, относящиеся к тематике этой главы, могут быть найдены в работах [13, 216, 217, 248]. Материалы разд. 10.2 и 10.3 основаны на работах [308, 309, 216, 312]. Материалы по разд. 10.4 можно найти в работе [43]. Дополнительную информацию по разд. 10.5 и 10.6 можно найти в работах [78, 248], а по разд. 10.7 в работе [279]. Ранними работами, рассматривающими вопросы планирования задачи робо-

том (разд. 10.8 и 10.9), являются работы [5, 61, 77, 267, 281]. Более поздними — работы [55, 144, 247]. Дополнительную информацию по разд. 10.10 можно найти в работах [117, 208, 304].

Упражнения

10.1. Предположим, что три миссионера и три людоеда собираются переплыть на лодке с одного берега реки на другой. Максимальная вместимость лодки 2 человека. Если в любой момент времени число людоедов превзойдет число миссионеров, людоеды начнут есть миссионеров. Предложите компьютерную программу для решения задачи безопасной переправы людей на другой берег. *Указание.* Используйте представление пространства состояний и методы поиска, приведенные в разд. 10.2. Пространство состояний можно представить в виде (N_m, N_c), где N_m, N_c — соответственно число миссионеров и людоедов на одном из берегов. Начальным состоянием является (0, 0), целевым — (3, 3) и возможными промежуточными состояниями — (0, 1), (0, 2), (0, 3), (1, 1), (2, 2), (3, 0), (3, 1), (3, 2).

10.2. Представьте, что вы хотите доказать теорему: «Диагонали параллелограмма делятся в точке пересечения пополам». Используйте граф И/ИЛИ для описания шагов поиска доказательства. Укажите подграф решения, содержащий доказательство теоремы.

10.3. Представьте следующие предложения на языке логики предикатов.

а) Формула, в которой главной связкой является $\Rightarrow$, эквивалентна некоторой формуле, в которой главной связкой является $\vee$.

б) Робот является интеллектуальным, если он может решить задачу, причем, если ее решает человек, ему для этого необходим интеллект.

в) Если объект находится на столе, он не находится на другом объекте.

10.4. Покажите, как можно представить задачу «Обезьяна и бананы», чтобы система STRIPS могла создать план, состоящий из следующих действий: идти к ящику, подвинуть ящик под бананы, влезть на ящик, схватить бананы. Используйте анализ конечных значений в качестве стратегии управления.

10.5. Покажите, как можно применить анализ конечных значений для решения задачи планирования действий робота, приведенной в примере в конце разд. 10.4.

Приложение А.
ВЕКТОРЫ И МАТРИЦЫ

Ниже дается обзор основных положений векторного и матричного исчислений. Векторы обозначены полужирными строчными буквами, а матрицы — полужирными прописными буквами.

А.1. Скалярные и векторные величины

Физические параметры можно разделить на два класса: параметры, которые характеризуются только величиной, и параметры, которые характеризуются величиной и направлением. Параметры, имеющие только величину, называются *скалярами* (например, время, масса, плотность, длина, координаты). Скалярная величина обычно записывается в виде действительного числа с соответствующей размерностью. Сравнение скалярных величин можно производить, только если они имеют одинаковую размерность.

Параметры, имеющие величину и направление, называются *векторами* (например, сила, момент, скорость, ускорение). Вектор обычно представляется графически в виде стрелки, длина и направление которой соответствуют величине и направлению рассматриваемого параметра. Сравнение векторных величин можно производить только, если они описывают один физический параметр и имеют одинаковую размерность.

Два вектора $\mathbf{a}$ и $\mathbf{b}$ равны, если они имеют одинаковые длину и направление. Запись $-\mathbf{a}$ обозначает, что вектор имеет величину, равную вектору $\mathbf{a}$, но противоположно направлен. С вектором связан положительный скаляр, равный его величине. Эта величина записывается в виде $|\mathbf{a}|$. Если a является величиной или длиной вектора $\mathbf{a}$, то

$$a = |\mathbf{a}|. \tag{A.1}$$

Единичный вектор $\mathbf{a}$ имеет единичную длину в заданном направлении

$$|\mathbf{a}| = 1. \tag{A.2}$$

Любой вектор $\mathbf{a}$ в трехмерном пространстве может быть приведен к единичному вектору следующим образом:

$$\frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{\mathbf{a}}{\sqrt{a_x^2 + a_y^2 + a_z^2}}, \tag{A.3}$$

где a_x, a_y и a_z — элементы вектора **a** вдоль основных осей координат.

A.2. Сложение и вычитание векторов

Сложение двух векторов **a** и **b** является коммутативным, т. е.

$$\mathbf{a}+\mathbf{b}=\mathbf{b}+\mathbf{a}. \tag{A.4}$$

Это легко проверить путем рассмотрения параллелограмма

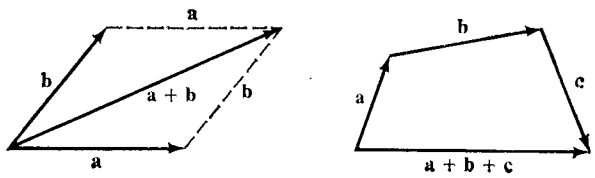


Рис. А.1. Сложение векторов.

со сторонами **a** и **b** (рис. А.1). Сложение трех и более векторов ассоциативно (рис. А.1):

$$(\mathbf{a}+\mathbf{b})+\mathbf{c}=\mathbf{a}+(\mathbf{b}+\mathbf{c})=\mathbf{a}+\mathbf{b}+\mathbf{c}. \tag{A.5}$$

Разность между двумя векторами **a** и **b** записывается как $\mathbf{a}-\mathbf{b}$ и определяется как вектор, проведенный из конца вектора

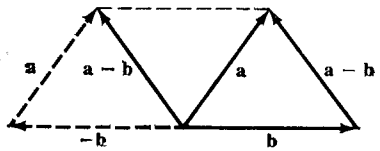


Рис. А.2. Вычитание векторов.

b в конец вектора **a** при совмещении начальных точек векторов **a** и **b** (рис. А.2).

A.3. Умножение векторов на скалярные величины

Умножение вектора **a** на скаляр m соответствует увеличению длины вектора **a** в $|m|$ раз в том же направлении, если $m>0$, и в обратном направлении, если $m<0$. Таким образом,

$$\mathbf{b}=m\mathbf{a} \tag{A.6}$$

и

$$|\mathbf{b}|=|m||\mathbf{a}|. \tag{A.7}$$

При умножении векторов на скалярные величины используются следующие правила:

1) $m(n\mathbf{a})=mn\mathbf{a}$;

2) $m(\mathbf{a}+\mathbf{b})=m\mathbf{a}+m\mathbf{b}$;

3) $(m+n)\mathbf{a}=m\mathbf{a}+n\mathbf{a}$,

где m и n — скалярные величины. (A.8)

A.4. Линейное векторное пространство

Линейным векторным пространством V называется непустое множество векторов, определенных на области действительных чисел F, которые удовлетворяют следующим условиям сложения векторов и их умножения на скалярные величины:

1. Сумма двух любых векторных элементов из V также является векторным элементом, принадлежащим V.

2. Сложение двух любых векторных элементов V является коммутативным.

3. Сложение трех любых векторных элементов V является ассоциативным.

4. Существует единственный элемент, называемый нулевым вектором пространства V (обозначается через **0**), такой, что для каждого элемента $\mathbf{a}\in V$:

$$\mathbf{0}+\mathbf{a}=\mathbf{a}+\mathbf{0}=\mathbf{a}.$$

5. Для каждого векторного элемента $\mathbf{a}\in V$ существует единственный вектор $(-\mathbf{a})\in V$, такой, что

$$\mathbf{a}+(-\mathbf{a})=\mathbf{0}.$$

6. Для каждого векторного элемента $\mathbf{a}\in V$ и для каждого скаляра $m\in F$ произведение m на **a** также является вектором пространства V. Если $m=1$, то

$$m\mathbf{a}=1\mathbf{a}=\mathbf{a}1=\mathbf{a}.$$

7. Для любых скаляров m и n из области F и любых векторов **a** и **b** из пространства V умножение на скалярные величины является дистрибутивным:

$$m(\mathbf{a}+\mathbf{b})=m\mathbf{a}+m\mathbf{b},$$
$$(m+n)\mathbf{a}=m\mathbf{a}+n\mathbf{a}.$$

8. Для любых скаляров m и n из области F и любого вектора **a** из пространства V

$$m(n\mathbf{a})=(mn)\mathbf{a}=mn\mathbf{a}.$$

Примерами линейного векторного пространства являются множества из всех действительных одно-, двух- или трехмерных векторов.

А.5. Линейная зависимость и линейная независимость

Конечное множество векторов $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ в пространстве V является линейно зависимым тогда и только тогда, когда существуют n скаляров $\{c_1, c_2, c_3, \ldots, c_n\}$ в области F (не все равные нулю), такие, что

$$c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + c_3\mathbf{x}_3 + \ldots + c_n\mathbf{x}_n = \mathbf{0}. \qquad (A.9)$$

Если единственным условием справедливости этого уравнения является тождественное равенство всех скаляров c_i нулю, то множество векторов $\{\mathbf{x}_i\}$ является линейно независимым.

Два линейно зависимых вектора в трехмерном пространстве коллинеарны, т. е. они находятся на одной линии. Три линейно зависимых вектора в трехмерном пространстве компланарны, т. е. они лежат в одной плоскости.

Пример. Пусть $\mathbf{a} = (1, 2, 0)^T$, $\mathbf{b} = (0, 3, 2)^T$, а $\mathbf{c} = (3, 0, -4)^T$ — векторы трехмерного векторного пространства. Эти три вектора составляют линейно зависимое множество в трехмерном векторном пространстве, поскольку

$$3\mathbf{a} - 2\mathbf{b} - \mathbf{c} = \mathbf{0}.$$

Они являются также компланарными.

А.6. Линейные комбинации, базисные векторы и размерность

Если существует подмножество векторов $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ в пространстве V и множество скаляров $\{e_1, e_2, \ldots, e_n\}$ в пространстве F, такие, что каждый вектор $\mathbf{x}$ в V может быть выражен как

$$\mathbf{x} = c_1\mathbf{e}_1 + c_2\mathbf{e}_2 + \ldots + c_n\mathbf{e}_n = \sum_{i=1}^{n} c_i\mathbf{e}_i, \qquad (A.10)$$

то вектор $\mathbf{x}$ является линейной комбинацией векторов $\{\mathbf{e}_i\}$. В этом случае говорят, что множество векторов $\{\mathbf{e}_i\}$ охватывает векторное пространство V.

Базисными векторами в векторном пространстве V является множество линейно независимых векторов, охватывающих векторное пространство V. Другими словами, базисные векторы — это минимальное число векторов, охватывающих векторное пространство V. При выбранном множестве базисных векторов для векторного пространства V каждый вектор $\mathbf{x} \in V$ может быть однозначно записан в виде линейной комбинации базисных векторов.

Размерность векторного пространства V равна числу базисных векторов, описывающих это пространство. Таким образом, n-мерное линейное векторное пространство имеет n базисных векторов. Для представления векторного пространства размерностью n используется обозначение V_n.

А.7. Декартовы системы координат

Из разд. А.6 следует, что при данном множестве из n базисных векторов $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ в векторном пространстве V_n любой вектор $\mathbf{r} \in V_n$ может быть однозначно записан в виде линейной комбинации базисных векторов:

$$\mathbf{r} = r_1\mathbf{e}_1 + r_2\mathbf{e}_2 + \ldots + r_n\mathbf{e}_n. \qquad (A.11)$$

В частности, при $n = 3$ в качестве базисных векторов могут служить три любых некомпланарных вектора.

В трехмерном векторном пространстве, если множество базисных векторов $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ имеет общее начало O, они образуют косоугольную систему координат с осями OX, OY и OZ,

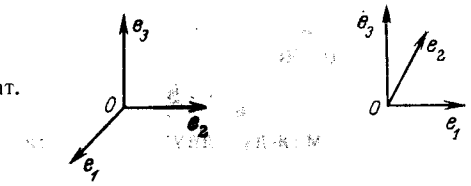


Рис. А.3. Системы координат.

направленными вдоль базисных векторов (рис. А.3). Соответствующим выбором направления базисных векторов можно формировать различные системы координат, которые обычно используются в инженерной практике.

Если базисные векторы $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ расположены ортогонально друг другу, т. е. они пересекаются в начале координат O под

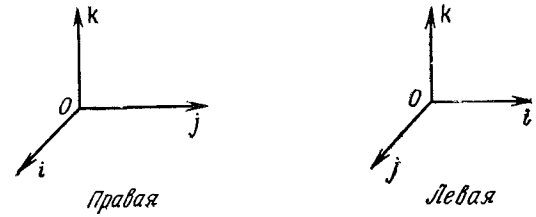


Рис. А.4. Декартова система координат.

прямыми углами, то они образуют прямоугольную или декартову систему координат. Кроме того, если каждый из базисных векторов имеет единичную длину, то система координат называется ортонормальной. В этом случае обычно для обозначения базисных векторов вместо $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ используется запись $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ (рис. А.4).

Если базисные векторы $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ ортонормальной системы координат направлены вдоль основных координатных осей и для совмещения оси OX с осью OY требуется поворот вправо на 90° относительно оси OZ, такая система координат называется

правой. Если же направление базисных векторов для аналогичного совмещения требует поворота влево, соответствующая система координат называется *левой*. В этой книге используется только правая система координат.

A.8. Произведение двух векторов

Кроме произведения скаляра и вектора важными являются еще два типа произведений. Первый тип называется *скалярным* (внутренним) произведением, а второй — *векторным* (внешним) произведением.

A.8.1. Скалярное (внутреннее) произведение

Внутреннее произведение двух векторов **a** и **b** является скаляром и определяется выражением

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta, \tag{A.12}$$

где θ — угол между двумя векторами (рис. A.5). Скалярная величина

$$b = |\mathbf{b}|\cos\theta \tag{A.13}$$

является компонентом вектора **b** вдоль вектора **a**, т. е. b численно равняется проекции **b** на **a**. Эта величина положительна,

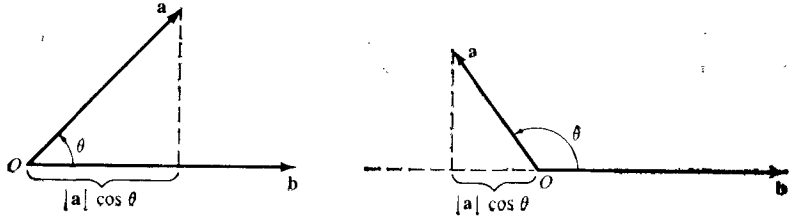


Рис. A.5. Скалярное произведение.

если проекция имеет одинаковое с вектором **a** направление, и отрицательна, если проекция имеет противоположное направление.

Скалярное произведение

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \cdot |\mathbf{b}|\cos\theta = |\mathbf{a}|b = |\mathbf{b}||\mathbf{a}|\cos\theta = |\mathbf{b}|a \tag{A.14}$$

равно произведению величины вектора **a** на компоненту вектора **b** вдоль вектора **a**. Оно также равно произведению величины вектора $|\mathbf{b}|$ на компоненту вектора **a** вдоль вектора **b**. Следовательно, скалярное произведение обладает свойством коммутативности:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}. \tag{A.15}$$

Если скалярное произведение **a** и **b** равно нулю, то один из векторов или оба вектора равны нулю, либо они перпендикулярны друг другу и $\cos(\pm 90°) = 0$. Таким образом, два ненулевых вектора **a** и **b** ортогональны тогда и только тогда, когда их скалярное произведение равно нулю.

Поскольку внутреннее произведение может равняться нулю, когда ни один из векторов не равен нулю, то деление на вектор не определяется. Таким образом, если

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a} \cdot \mathbf{c}, \tag{A.16}$$

то не $\mathbf{b} = \mathbf{c}$, а просто $\mathbf{a} \cdot (\mathbf{b} - \mathbf{c}) = 0$ и $\mathbf{b} - \mathbf{c}$ является нулевым вектором или вектором, ортогональным к **a**.

Если **a** и **b** имеют одно направление и $\theta = 0°$, то $\cos\theta = 1$ и $\mathbf{a} \cdot \mathbf{b}$ равно произведению длин этих двух векторов. В частности, если $\mathbf{a} = \mathbf{b}$, то

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a} \cdot \mathbf{a} = |\mathbf{a}||\mathbf{a}| = |\mathbf{a}|^2, \tag{A.17}$$

т. е. равно квадрату длины вектора **a**.

Скалярное произведение векторов является дистрибутивным относительно сложения, т. е.

$$\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c} \tag{A.18}$$

и

$$(\mathbf{b} + \mathbf{c}) \cdot \mathbf{a} = \mathbf{b} \cdot \mathbf{a} + \mathbf{c} \cdot \mathbf{a}. \tag{A.19}$$

A.8.2. Векторное (внешнее) произведение

Векторное (или внешнее) произведение **двух** векторов **a** и **b** определяется как вектор **c**:

$$\mathbf{c} = \mathbf{a} \times \mathbf{b}, \tag{A.20}$$

который ортогонален к векторам **a** и **b** и **имеет** величину

$$|\mathbf{c}| = |\mathbf{a}||\mathbf{b}|\sin\theta. \tag{A.21}$$

Вектор **c** направлен так, что вращение вправо вокруг **c** на угол θ, меньший 180°, совмещает **a** с **b**, где θ — угол между **a** и **b** (рис. A.6).

Поскольку $h = |\mathbf{b}|\sin\theta$ (рис. A.6), произведение $a \times b$ имеет величину, равную площади параллелограмма со сторонами **a** и **b**. Векторное произведение $\mathbf{a} \times \mathbf{b}$ может быть рассмотрено как результат проецирования вектора b на плоскость $WXYZ$, перпендикулярную плоскости векторов **a** и **b**, вращения проекции на 90° в положительном направлении вокруг вектора **a** и умножения полученного результата на $|\mathbf{a}|$.

Векторное произведение $\mathbf{b} \times \mathbf{a}$ имеет ту же величину, что и $\mathbf{a} \times \mathbf{b}$, но противоположное направление, т. е.

$$\mathbf{b} \times \mathbf{a} = -(\mathbf{a} \times \mathbf{b}). \tag{A.22}$$

Векторное произведение не является коммутативным. Если векторы $\mathbf{a}$ и $\mathbf{b}$ параллельны, то θ равен 0° или 180° и

$$|\mathbf{a} \times b| = |\mathbf{a}||\mathbf{b}| \sin\theta = 0. \tag{A.23}$$

И наоборот, если векторное произведение равно нулю, то один или оба вектора равны нулю или параллельны.

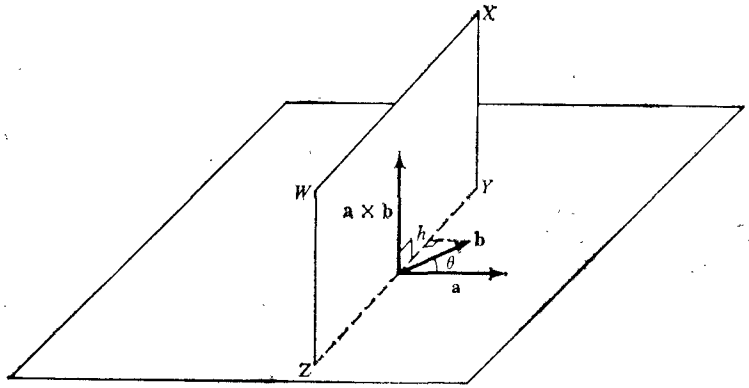


Рис. А.6. Векторное произведение.

Отметим также, что векторное произведение дистрибутивно относительно сложения, т. е.

$$\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c} \tag{A.24}$$

и

$$(\mathbf{b} + \mathbf{c}) \times \mathbf{a} = \mathbf{b} \times \mathbf{a} + \mathbf{c} \times \mathbf{a}. \tag{A.25}$$

Для скалярного и векторного произведения единичных векторов $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, направленных вдоль основных осей правой декартовой системы координат, имеем

$$\begin{aligned} \mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1, \\ \mathbf{i} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{i} = 0, \\ \mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = \mathbf{0}, \\ \mathbf{i} \times \mathbf{j} = -\mathbf{j} \times \mathbf{i} = \mathbf{k}, \\ \mathbf{j} \times \mathbf{k} = -\mathbf{k} \times \mathbf{j} = \mathbf{i}, \\ \mathbf{k} \times \mathbf{i} = -\mathbf{i} \times \mathbf{k} = \mathbf{j}. \end{aligned} \tag{A.26}$$

Используя определение составляющих элементов вектора и уравнение (А.26), скалярное произведение $\mathbf{a}$ и $\mathbf{b}$ можно записать в виде

$$\mathbf{a} \cdot \mathbf{b} = (a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}) \cdot (b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}) = a_1b_1 + a_2b_2 + a_3b_3 = \mathbf{a}^T\mathbf{b}, \tag{A.27}$$

где $\mathbf{a}^T$ обозначает транспонирование $\mathbf{a}$ (разд. А.12). Векторное произведение $\mathbf{a}$ и $\mathbf{b}$ можно записать в виде детерминанта (разд. А.15)

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}. \tag{A.28}$$

А.9. Произведение трех и более векторов

Для скалярных и векторных произведений трех и более векторов обычно выделяют следующие типы произведений:

$$(\mathbf{a} \cdot \mathbf{b})\mathbf{c}, \quad \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}), \quad \mathbf{a} \times (\mathbf{b} \times \mathbf{c}). \tag{A.29}$$

Произведение $(\mathbf{a} \cdot \mathbf{b})\mathbf{c}$ является простым произведением скаляра $(\mathbf{a} \cdot \mathbf{b})$ и вектора $\mathbf{c}$. Результирующий вектор имеет величину $|\mathbf{a} \cdot \mathbf{b}||\mathbf{c}|$ и направление, совпадающее с направлением вектора $\mathbf{c}$ или противоположное ему в зависимости от знака $(\mathbf{a} \cdot \mathbf{b})$.

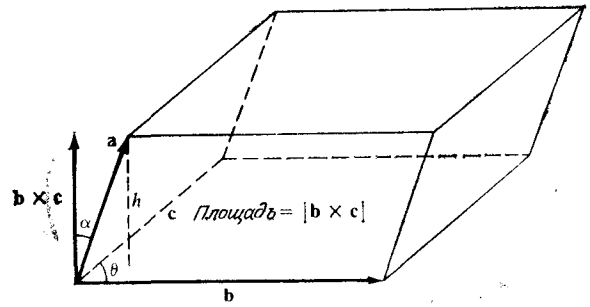


Рис. А.7. Смешанное произведение.

Смешанное произведение $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ представляет собой скаляр с величиной, равной объему параллелепипеда с гранями, образованными векторами $\mathbf{a}$, $\mathbf{b}$ и $\mathbf{c}$ (рис. А.7), т. е.

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = |\mathbf{a}||\mathbf{b}||\mathbf{c}| \sin\theta\cos\alpha = hA = \text{объем параллелепипеда}, \tag{A.30}$$

где h и A — соответственно высота и площадь основания параллелепипеда. Представляя векторы в виде составляющих их элементов в трехмерном векторном пространстве, получим

$$\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})=(a_x\mathbf{i}+a_y\mathbf{j}+a_z\mathbf{k})\cdot\begin{vmatrix}\mathbf{i}&\mathbf{j}&\mathbf{k}\\ b_x&b_y&b_z\\ c_x&c_y&c_z\end{vmatrix}=$$

$$=a_x(b_yc_z-b_zc_y)+a_y(b_zc_x-b_xc_z)+a_z(b_xc_y-b_yc_x)=$$

$$=\begin{vmatrix}a_x&a_y&a_z\\ b_x&b_y&b_z\\ c_x&c_y&c_z\end{vmatrix}. \qquad (A.31)$$

Отметим, что скобки, выделяющие векторное произведение $\mathbf{b}\times\mathbf{c}$, могут быть сняты, поскольку не имеет смысла представление $\mathbf{a}\cdot\mathbf{b}\times\mathbf{c}$ в виде $(\mathbf{a}\cdot\mathbf{b})\times\mathbf{c}$.

Видно также (рис. А.7), что объем параллелепипеда не зависит от плоскости, выбранной за его основание.

Таким образом,

$$\mathbf{a}\cdot\mathbf{b}\times\mathbf{c}=\mathbf{b}\cdot\mathbf{c}\times\mathbf{a}=\mathbf{c}\cdot\mathbf{a}\times\mathbf{b} \qquad (A.32)$$

и

$$\mathbf{a}\cdot\mathbf{b}\times\mathbf{c}=-\mathbf{a}\cdot\mathbf{c}\times\mathbf{b}=-\mathbf{c}\cdot\mathbf{b}\times\mathbf{a}=-\mathbf{b}\cdot\mathbf{a}\times\mathbf{c}. \qquad (A.33)$$

Эти результаты могут быть получены из свойств детерминантов (разд. А.15). Запись $\mathbf{a}\cdot\mathbf{b}\times\mathbf{c}\equiv\mathbf{a}\times\mathbf{b}\cdot\mathbf{c}$ может быть

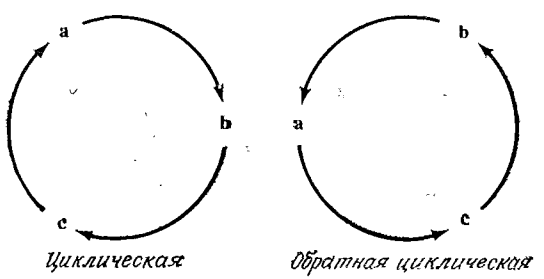


Рис. А.8. Циклическая перестановка.

представлена в виде $(\mathbf{abc})$, поскольку смысл данного выражения не меняется от использования операции, обозначенной точкой или косым крестом. Уравнение (А.32), выражает циклическую перестановку этих векторов, а уравнение (А.33) выражает их обратную циклическую перестановку. Иллюстрация циклической перестановки приведена на рис. А.8. Уравнение (А.32) получается путем следования по часовой стрелке вдоль окружности. Таким же образом, но в обратном направлении, получается уравнение (А.33). Смешанное произведение может быть использовано для проверки линейной зависимости трех компланарных векторов. Если три вектора $\mathbf{a}$, $\mathbf{b}$ и $\mathbf{c}$ — компланарны, то $(\mathbf{abc})=0$. Следовательно, если два или три вектора равны, то смешанное произведение будет равно нулю. Из этого следует, что если $\mathbf{e}_1$, $\mathbf{e}_2$ и $\mathbf{e}_3$ являются базисными векторами в векторном пространстве V_3, то $(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)\neq 0$, и они образуют правую систему координат при $(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)>0$ и левую систему координат при $(\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3)<0$.

Двойное векторное произведение $\mathbf{a}\times(\mathbf{b}\times\mathbf{c})$ представляет собой вектор, перпендикулярный $(\mathbf{b}\times\mathbf{c})$ и лежащий в плоскости векторов $\mathbf{b}$ и $\mathbf{c}$ (рис. А.9). Предположим, что векторы $\mathbf{a}$, $\mathbf{b}$

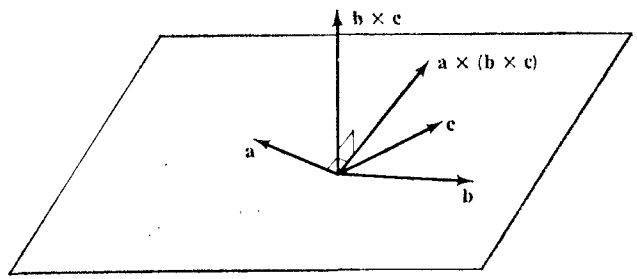


Рис. А.9. Двойное векторное произведение.

и $\mathbf{c}$ неколлинеарны. Тогда вектор $\mathbf{a}\times(\mathbf{b}\times\mathbf{c})$, находящийся в плоскости векторов $\mathbf{b}$ и $\mathbf{c}$, может быть выражен в виде линейной комбинации $\mathbf{b}$ и $\mathbf{c}$, т. е.

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c})=m\mathbf{b}+n\mathbf{c}. \qquad (A.34)$$

Поскольку вектор $\mathbf{a}\times(\mathbf{b}\times\mathbf{c})$ перпендикулярен вектору $\mathbf{a}$, то при скалярном умножении левой и правой частей уравнения (А.34) на вектор $\mathbf{a}$ получим 0:

$$\mathbf{a}\cdot[\mathbf{a}\times(\mathbf{b}\times\mathbf{c})]=m(\mathbf{a}\cdot\mathbf{b})+n(\mathbf{a}\cdot\mathbf{c})=0. \qquad (A.35)$$

Таким образом,

$$\frac{m}{\mathbf{a}\cdot\mathbf{c}}=\frac{-n}{\mathbf{a}\cdot\mathbf{b}}=\lambda, \qquad (A.36)$$

где m, n и λ — скаляры, и уравнение (А.34) примет вид

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c})=\lambda[(\mathbf{a}\cdot\mathbf{c})\mathbf{b}-(\mathbf{a}\cdot\mathbf{b})\mathbf{c}]. \qquad (A.37)$$

Можно показать, что $\lambda\equiv 1$, поэтому двойное векторное произведение будет равно

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c})=(\mathbf{a}\cdot\mathbf{c})\mathbf{b}-(\mathbf{a}\cdot\mathbf{b})\mathbf{c}. \qquad (A.38)$$

Используя уравнение (А.22), получим

$$(\mathbf{a}\times\mathbf{b})\times\mathbf{c}=-\mathbf{c}\times(\mathbf{a}\times\mathbf{b})=-(\mathbf{c}\cdot\mathbf{b})\,\mathbf{a}+(\mathbf{c}\cdot\mathbf{a})\,\mathbf{b}. \quad (A.39)$$

Более сложные случаи умножения четырех и более векторов можно упростить с помощью двойных произведений. Например,

$$(\mathbf{a}\times\mathbf{b})\times(\mathbf{c}\times\mathbf{d})=(\mathbf{a}\times\mathbf{b}\cdot\mathbf{d})\,\mathbf{c}-(\mathbf{a}\times\mathbf{b}\cdot\mathbf{c})\,\mathbf{d}=(\mathbf{abd})\,\mathbf{c}-(\mathbf{abc})\,\mathbf{d} \quad (A.40)$$

и

$$(\mathbf{a}\times\mathbf{b})\cdot(\mathbf{c}\times\mathbf{d})=\mathbf{a}\cdot\mathbf{d}\times(\mathbf{c}\times\mathbf{d})=\mathbf{a}\cdot[(\mathbf{b}\cdot\mathbf{d})\,\mathbf{c}-(\mathbf{b}\cdot\mathbf{c})\,\mathbf{d}]=$$
$$=(\mathbf{b}\cdot\mathbf{d})(\mathbf{a}\cdot\mathbf{c})-(\mathbf{b}\cdot\mathbf{c})(\mathbf{a}\cdot\mathbf{d}). \quad (A.41)$$

А.10. Производные векторных функций

Производная вектора $\mathbf{r}(t)$ определяется как

$$\frac{d\mathbf{r}}{dt}\triangleq\lim_{\Delta t\to 0}\frac{\mathbf{r}(t+\Delta t)-\mathbf{r}(t)}{\Delta t}=\lim_{\Delta t\to 0}\frac{\Delta\mathbf{r}}{\Delta t}. \quad \textbf{(A.42)}$$

Из уравнений (А.11) и (А.42) следует, что

$$\frac{d\mathbf{r}}{dt}=\left(\frac{dr_x}{dt}\right)\mathbf{i}+\left(\frac{dr_y}{dt}\right)\mathbf{j}+\left(\frac{dr_z}{dt}\right)\mathbf{k} \quad \textbf{(A.43)}$$

и m-я производная от $\mathbf{r}(t)$ равна

$$\frac{d^n\mathbf{r}}{dt^n}=\left(\frac{d^nr_x}{dt^n}\right)\mathbf{i}+\left(\frac{d^nr_y}{dt^n}\right)\mathbf{j}+\left(\frac{d^nr_z}{dt^n}\right)\mathbf{k}. \quad (A.44)$$

Из уравнения (А.42) вытекают следующие правила дифференцирования векторных функций:

$$1)\ \frac{d}{dt}(\mathbf{a}\pm\mathbf{b})=\frac{d\mathbf{a}}{dt}\pm\frac{d\mathbf{b}}{dt}; \quad (A.45)$$

$$2)\ \frac{d}{dt}(m\mathbf{a})=m\frac{d\mathbf{a}}{dt},\ \text{где } m-\text{скаляр};$$

$$3)\ \frac{d}{dt}(\mathbf{a}\cdot\mathbf{b})=\left(\frac{d\mathbf{a}}{dt}\right)\cdot\mathbf{b}+\mathbf{a}\cdot\left(\frac{d\mathbf{b}}{dt}\right);$$

$$4)\ \frac{d}{dt}(\mathbf{a}\times\mathbf{b})=\left(\frac{d\mathbf{a}}{dt}\right)\times\mathbf{b}+\mathbf{a}\times\left(\frac{d\mathbf{b}}{dt}\right);$$

$$5)\ \frac{d}{dt}(\mathbf{abc})=\left[\left(\frac{d\mathbf{a}}{dt}\right)\mathbf{bc}\right]+\mathbf{a}\left[\left(\frac{d\mathbf{b}}{dt}\right)\mathbf{c}\right]+\left[\mathbf{ab}\left(\frac{d\mathbf{c}}{dt}\right)\right];$$

$$6)\ \frac{d}{dt}[\mathbf{a}\times(\mathbf{b}\times\mathbf{c})]=\left[\frac{d\mathbf{a}}{dt}\times(\mathbf{b}\times\mathbf{c})\right]+\left[\mathbf{a}\times\left(\frac{d\mathbf{b}}{dt}\times\mathbf{c}\right)\right]+$$
$$+\left[\mathbf{a}\times\left(\mathbf{b}\times\frac{d\mathbf{c}}{dt}\right)\right].$$

А.11. Интегрирование векторных функций

Если $d\mathbf{a}/dt=\mathbf{b}(t)$, то интеграл вектора $\mathbf{b}(t)$ представляет собой

$$\mathbf{a}(t)=\int\mathbf{b}(\tau)\,d\tau+\mathbf{c}, \quad (A.46)$$

где $\mathbf{c}$ — постоянный вектор.

Если $\mathbf{b}(t)$ можно записать в прямоугольной системе координат, то

$$\begin{aligned}a_x(t)&=\int b_x(\tau)\,d\tau+c_x,\\ a_y(t)&=\int b_y(\tau)\,d\tau+c_y,\\ a_z(t)&=\int b_z(\tau)\,d\tau+c_z.\end{aligned} \quad (A.47)$$

А.12. Матричная алгебра

Ниже мы рассмотрим еще один важный математический инструмент — матрицы, которые составляют основу математического аппарата при анализе роботов. Матрица $\mathbf{A}$ (или $\mathbf{A}_{m\times n}$) порядка $m\times n$ представляет собой прямоугольную систему действительных или комплексных чисел, называемых элементами матрицы, которые расположены в m строк и n столбцов:

$$\mathbf{A}=[a_{ij}]=\begin{bmatrix}a_{11}&a_{12}&\dots&a_{1n}\\ a_{21}&a_{22}&\dots&a_{2n}\\ \vdots&&&\vdots\\ a_{m1}&a_{m2}&\dots&a_{mn}\end{bmatrix},\quad \begin{aligned}&i=1,\ 2,\ \dots,\ m,\\ &j=1,\ 2,\ \dots,\ n.\end{aligned} \quad (A.48)$$

Кроме специальных случаев будем предполагать, что $\mathbf{A}$ является действительной матрицей. Матрица, состоящая из одного столбца (строки), называется матрицей-столбцом (строкой). Матрица-столбец и матрица-строка часто рассматриваются как векторы.

Транспонированная матрица $\mathbf{A}$, обозначаемая $\mathbf{A}^T$, представляет собой матрицу, номер строки в которой совпадает

с номером столбца матрицы **A**. Другими словами, если

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad \begin{matrix} i=1,\ 2,\ \dots,\ m, \\ j=1,\ 2,\ \dots,\ n, \end{matrix} \qquad \textbf{(A.49)}$$

то

$$\mathbf{A}^T=\begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}, \quad \begin{matrix} i=1,\ 2,\ \dots,\ n, \\ j=1,\ 2,\ \dots,\ m. \end{matrix} \qquad \textbf{(A.50)}$$

В частности, при транспонировании матрицы-столбца получается матрица-строка, и наоборот.

Квадратная матрица порядка n имеет одинаковое число строк и столбцов (т. е. $m=n$). Диагональная матрица порядка n является квадратной матрицей с нулевыми недиагональными элементами, т. е. элементы

$$a_{ij}=0 \text{ при } i\neq j \quad \text{для} \quad i,\ j=1,\ 2,\ \dots,\ n. \qquad \text{(A.51)}$$

Единичная матрица порядка n представляет собой диагональную матрицу с единичными диагональными элементами, т. е. $a_{ij}=1$, если $i=j$, и $a_{ij}=0$, если $i\neq j$. Эта матрица называется тождественной и обозначается через $\mathbf{I}_n$ или $\mathbf{I}_{n\times n}$.

Симметрической матрицей порядка n называется квадратная матрица, которая не изменяется при транспонировании, т. е. $\mathbf{A}=\mathbf{A}^T$ или $a_{ij}=a_{ji}$ для всех i и j. Если элементы квадратной матрицы такие, что

$$a_{ij}=-a_{ij} \quad \text{и} \quad a_{ii}=0, \qquad i,\ j=1,\ 2,\ \dots,\ n, \qquad \text{(A.52)}$$

то матрица называется *кососимметрической*. Отметим, что если **A** — кососимметрическая матрица, то $\mathbf{A}=-\mathbf{A}^T$.

Любая несимметрическая матрица **A** может быть преобразована в симметрическую матрицу **C** следующим образом:

$$\mathbf{C}=\frac{\mathbf{A}+\mathbf{A}^T}{2}. \qquad \text{(A.53)}$$

Нулевой матрицей называется матрица, все элементы которой тождественно равны нулю. Две матрицы одного порядка равны, если равны их соответствующие элементы, т. е. если $a_{ij}=b_{ij}$ для всех i и j, то $\mathbf{A}=\mathbf{B}$.

A.13. Сложение матриц

Сложение (вычитание) матриц **A** и **B** одного порядка дает результирующую матрицу **C** того же порядка, элементы которой представляют собой сумму (разность) соответствующих элементов матриц **A** и **B**. Таким образом,

$$\mathbf{A}+\mathbf{B}=\mathbf{C} \quad \text{или} \quad a_{ij}+b_{ij}=c_{ij} \quad \text{для всех } i,\ j \qquad \text{(A.54)}$$

и

$$\mathbf{A}-\mathbf{B}=\mathbf{C} \quad \text{или} \quad a_{ij}-b_{ij}=c_{ij} \quad \text{для всех } i,\ j. \qquad \text{(A.55)}$$

Сложение матриц имеет те же свойства, что и сложение действительных чисел:

1) $\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$;
2) $(\mathbf{A}+\mathbf{B})+\mathbf{C}=\mathbf{A}+(\mathbf{B}+\mathbf{C})$;
3) $\mathbf{A}+\mathbf{O}=\mathbf{A}$ ($\mathbf{O}$ — нулевая матрица); (A.56)
4) $\mathbf{A}+(-\mathbf{A})=\mathbf{O}$.

A.14. Умножение матриц

Произведение скаляра и матрицы получают путем умножения каждого элемента матрицы **A** на скаляр. Таким образом,

$$k\mathbf{A}=\mathbf{A}k=[ka_{ij}]=[a_{ij}k], \quad i=1,\ 2,\ \dots,\ m,\ \ j=1,\ 2,\ \dots,\ n.$$

При умножении произвольной матрицы порядка $(m\times n)$ на произвольный скаляр справедливы следующие правила:

1) $a(\mathbf{A}+\mathbf{B})=a\mathbf{A}+a\mathbf{B}$;
2) $(a+b)\mathbf{A}=a\mathbf{A}+b\mathbf{A}$;
3) $a(b\mathbf{A})=(ab)\mathbf{A}$; (A.57)
4) $1\mathbf{A}=\mathbf{A}$,

где a и b — скаляры.

Две матрицы могут быть перемножены, только если они соответствуют друг другу, т. е. если $\mathbf{AB}=\mathbf{C}$, то число столбцов **A** должно быть равно числу строк **B**, при этом результирующая матрица **C** имеет число строк и столбцов, соответственно равное их числу в **A** и **B**. Таким образом,

$$(\mathbf{A}_{m\times n})(\mathbf{B}_{n\times p})=\mathbf{C}_{m\times p} \quad \text{или} \quad c_{ij}=\sum_{k=1}^{n} a_{ik}b_{kj}. \qquad \text{(A.58)}$$

Относительно уравнения (A.58) можно сказать, что или **A** стоит перед **B**, или **B** стоит после **A** при формировании **C**. Для получения элемента i-й строки и j-го столбца **C** суммируются произведения членов соответствующих элементов в i-й строке **A** и j-м столбце **B** по уравнению (A.58). Другими словами, производится умножение i-й строки **A** на j-й столбец **B**. Как

правило, умножение матриц не является коммутативным, даже если они соответствуют друг другу, т. е. для квадратных матриц **A** и **B** порядка n имеем $\mathbf{AB} \neq \mathbf{BA}$.

Если $\mathbf{AB} = \mathbf{BA}$, то матрицы коммутативны. Единичная матрица коммутативна относительно любой квадратной матрицы:

$$\mathbf{IA} = \mathbf{AI} = \mathbf{A}. \tag{A.59}$$

Умножение матриц ассоциативно и дистрибутивно по отношению к сложению матриц, т. е.

$$\begin{aligned} &1)\ (k\mathbf{A})\mathbf{B} = k(\mathbf{AB}) = \mathbf{A}(k\mathbf{B});\\ &2)\ \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C};\\ &3)\ (\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC};\\ &4)\ \mathbf{C}(\mathbf{A}+\mathbf{B}) = \mathbf{CA}+\mathbf{CB} \end{aligned} \tag{A.60}$$

в предположении, что умножение осуществимо. Из правила 2) следует, что при умножении трех матриц можно или умножить **B** на **C** или умножить **A** на **B**, а потом умножить результат на оставшуюся матрицу. Обычно $\mathbf{AB} = \mathbf{O}$ не означает, что $\mathbf{A} = \mathbf{O}$ или $\mathbf{B} = \mathbf{O}$. Сформулируем следующие правила умножения матриц:

1) $(\text{Матрица})_{m\times n}\ (\text{Матрица})_{n\times p} = (\text{Матрица})_{m\times p}$;
2) $(\text{Матрица})_{m\times n}\ (\text{Матрица-столбец})_{n\times 1} = (\text{Матрица-столбец})_{m\times 1}$;
3) $(\text{Матрица-строка})_{1\times n}\ (\text{Матрица-столбец})_{n\times 1} = \text{Скаляр}$;
4) $(\text{Матрица-строка})_{1\times m}\ (\text{Матрица})_{m\times n} = (\text{Матрица-строка})_{1\times n}$;
5) $(\text{Матрица-столбец})_{m\times 1}\ (\text{Матрица-строка})_{1\times n} = (\text{Матрица})_{m\times n}$.

Иногда при сложении и умножении матриц удобно разбивать матрицы на подматрицы для применения правил матричной алгебры к подматрицам.

A.15. Детерминанты

Детерминант матрицы **A** размерностью $n \times n$ записывается в виде

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \tag{A.61}$$

и равняется сумме произведений элементов произвольной строки или столбца на их соответствующие алгебраические дополнения, т. е.

$$|\mathbf{A}| = \sum_{j=1}^{n} a_{ij}A_{ij} = \sum_{i=1}^{n} a_{ij}A_{ij}. \tag{A.62}$$

Здесь A_{ij} — дополнение элемента a_{ij}, которое определяется выражением

$$A_{ij} = (-1)^{i+j} M_{ij}, \tag{A.63}$$

где M_{ij} — дополнительный минор, полученный исключением элементов в i-й строке и j-м столбце детерминанта $|\mathbf{A}|$. Другими словами, если

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ a_{n1} & a_{n2} & \dots & a_{nj} & \dots & a_{nn} \end{vmatrix},$$

то, исключая элемент, i-й строки и j-го столбца, получим

$$|\boldsymbol{M}_{ij}| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1,j-1} & a_{1,j+1} & \dots & a_{1n} \\ \vdots & & & & & & \vdots \\ a_{i-1,1} & a_{i-1,2} & \dots & a_{i-1,j-1} & a_{i-1,j+1} & \dots & a_{i-1,n} \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,j-1} & a_{i+1,j+1} & \dots & a_{i+1,n} \\ \vdots & & & & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{n,j-1} & a_{n,j+1} & \dots & a_{nn} \end{vmatrix}.$$

Из приведенного определения следует, что детерминант порядка n зависит от n детерминантов порядка $n-1$, каждый из которых в свою очередь зависит от $n-1$ детерминантов порядка $n-2$, и так далее до детерминанта порядка 1, который является скаляром.

Для вычисления детерминантов второго и третьего порядков может быть использован простой диагональный метод. Для $n = 2$ имеем

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}. \tag{A.64}$$

Детерминант третьего порядка находится следующим образом:

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{32}a_{21} - a_{31}a_{22}a_{13} - a_{12}a_{21}a_{33} - a_{11}a_{32}a_{23}. \quad \text{(A.65)}$$

Для упрощения вычислений детерминантов используются следующие свойства:

1. Если все элементы любой строки (или столбца) матрицы **A** равны нулю, $|\mathbf{A}| = 0$.
2. $|\mathbf{A}| = |\mathbf{A}^T|$.
3. Если две любые строки (или столбца) матрицы **A** поменять местами, изменится знак соответствующего детерминанта.
4. Если **A** и **B** имеют порядок n, то $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$.
5. Если все элементы любой строки (или столбца) матрицы **A** умножить на скаляр k, значение детерминанта увеличится в k раз.
6. Если ранг (разд. A.16) матрицы **A** порядка n меньше n, ее детерминант равен нулю.
7. Если кратное любой строки (или столбца) сложить с другой строкой (или столбцом), детерминант не изменится.

Пример. Пусть

$$\mathbf{A} = \begin{bmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{bmatrix}.$$

Тогда

$$|\mathbf{A}| = \begin{vmatrix} 1 & a & a^2 \\ 1 & b & b^2 \\ 1 & c & c^2 \end{vmatrix} = \begin{vmatrix} 1 & a & a^2 \\ 0 & b-a & b^2-a^2 \\ 0 & c-a & c^2-a^2 \end{vmatrix} = (b-a)(c^2-a^2) - (c-a)(b^2-a^2) = (a-b)(b-c)(c-a).$$

Этот детерминант называется детерминантом Вандермонда третьего порядка.

A.16. Ранг матрицы

Если строки квадратной матрицы **A** порядка n линейно независимы, детерминант такой матрицы не равен нулю, а матрица называется невырожденной. Если детерминант квадратной матрицы порядка n равен нулю, то матрица — вырожденная, а ее строки не являются линейно независимыми. Таким образом, детерминант может использоваться для характеристики особенности матрицы.

Ранг матрицы **A** порядка $m \times n$ равен порядку наибольшей подматрицы **A** с ненулевым детерминантом. Следовательно, матрица порядка $m \times n$ может иметь ранг, равный наименьшему из значений m и n или ниже. Ранг матрицы определяет число линейно независимых строк (или столбцов) в матрице.

A.17. Присоединение и обратные матрицы

Если **A** — квадратная матрица, а A_{ij} — алгебраическое дополнение элемента a_{ij} в детерминанте $|\mathbf{A}|$, то транспонированная матрица, полученная из дополнений $\mathbf{A}_{ij}$, называется присоединенной к **A** матрицей:

$$[A_{ij}]^T = [A_{ji}] \quad i, j = 1, 2, \ldots, n. \quad \text{(A.66)}$$

Иногда присоединение к **A** записывается в виде $adj\,\mathbf{A}$.

Обратная матрица $\mathbf{A}^{-1}$ к невырожденной квадратной матрице **A** равна присоединенной к **A** матрице, деленной на детерминант **A**, т. е.

$$\mathbf{A}^{-1} = \frac{[A_{ij}]^T}{|\mathbf{A}|} = \frac{adj\mathbf{A}}{|\mathbf{A}|}. \quad \text{(A.67)}$$

Произведение невырожденной матрицы **A** порядка $n \times n$ на свою обратную матрицу равно тождественной матрице $\mathbf{I}_n$, т. е.

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n. \quad \text{(A.68)}$$

Таким образом, из уравнений (A.67) и (A.68) получаем

$$(adj\mathbf{A})\,\mathbf{A} = \mathbf{A}\,(adj\mathbf{A}) = |\mathbf{A}|\,\mathbf{I}_n \quad \text{(A.69)}$$

и

$$|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}. \quad \text{(A.70)}$$

Если $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ — квадратные матрицы порядка n, обратная матрица их произведения равна произведению их обратных матриц в обратном порядке:

$$(\mathbf{A}_1\mathbf{A}_2 \ldots \mathbf{A}_n)^{-1} = \mathbf{A}_n^{-1}\mathbf{A}_{n-1}^{-1} \ldots \mathbf{A}_2^{-1}\mathbf{A}_1^{-1}. \quad \text{(A.71)}$$

Аналогично, если существует матрица произведения $\mathbf{A}_1\mathbf{A}_2 \ldots \mathbf{A}_n$, то транспонированная матрица этого произведения равна произведению транспонированных матриц в обратном порядке:

$$(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 \ldots \mathbf{A}_n)^T = (\mathbf{A}_n)^T(\mathbf{A}_{n-1})^T \ldots (\mathbf{A}_2)^T(\mathbf{A}_1)^T. \quad \text{(A.72)}$$

В общем случае матрица порядка 2×2

$$\mathbf{A} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

имеет обратную матрицу, равную

$$\mathbf{A}^{-1}=\frac{1}{ad-bc}\begin{bmatrix} d & -c \\ -b & a \end{bmatrix}.$$

Аналогично матрица порядка 3×3

$$\mathbf{A}=\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

имеет обратную матрицу, равную

$$\mathbf{A}^{-1}=\frac{1}{aei+dhc+gfb-afh-dbi-gec}\times$$
$$\times\begin{bmatrix} (ei-fh) & -(bi-ch) & (bf-ce) \\ -(di-fg) & (ai-cg) & -(af-cd) \\ (dh-ge) & -(ah-bg) & (ae-bd) \end{bmatrix}.$$

Важный результат, называемый леммой об обращении, формулируется следующим образом:

$$[\mathbf{A}^{-1}+\mathbf{B}^T\mathbf{C}\mathbf{B}]^{-1}=\mathbf{A}-\mathbf{A}\mathbf{B}^T[\mathbf{B}\mathbf{A}\mathbf{B}^T+\mathbf{C}^{-1}]^{-1}\mathbf{B}\mathbf{A}. \quad \text{(A.73)}$$

Доказательство этого результата предполагается выполнить в качестве упражнения.

A.18. След матрицы

Следом квадратной матрицы **A** порядка n является сумма ее диагональных элементов

$$Trace\mathbf{A}\equiv Tr(\mathbf{A})=\sum_{i=1}^{n}a_{ii}. \quad \text{(A.74)}$$

Некоторыми важными свойствами следа матрицы являются следующие свойства:

1) $Tr(\mathbf{A})=Tr(\mathbf{A}^T)$; (A.75)
2) $Tr(\mathbf{A}+\mathbf{B})=Tr(\mathbf{A})+Tr(\mathbf{B})$; (A.76)
3) $Tr(\mathbf{AB})=Tr(\mathbf{BA})$; (A.77)
4) $Tr(\mathbf{ABC}^T)=Tr(\mathbf{CB}^T\mathbf{A}^T)$. (A.78)

Более подробное изложение материала, рассмотренного в данном приложении, можно найти в работах [21, 82, 218, 237, 285].

Приложение Б.
ЯКОБИАН МАНИПУЛЯТОРА

При раздельном (независимом) управлении движением (гл. 5) необходимо определять, насколько бесконечно малое движение каждого сочленения манипулятора влияет на бесконечно малое движение всего манипулятора. Одним из преимуществ раздельного движения является существование линейного соответствия между пространством бесконечно малого движения сочленений и пространством бесконечно малого движения манипулятора. Это соответствие определяется с помощью якобиана. Ниже рассматриваются три метода получения якобиана для шестизвенного манипулятора с поступательными и вращательными сочленениями.

Б.1. Метод векторного произведения

Пусть векторы положения, линейной и угловой скорости манипулятора (конечного звена) относительно базовой системы координат (x_0, y_0, z_0) определяются соответственно в виде

$$\begin{aligned}\mathbf{p}(t)&\triangleq[p_x(t),\ p_y(t),\ p_z(t)]^T,\\ \mathbf{v}(t)&\triangleq[v_x(t),\ v_y(t),\ v_z(t)]^T,\\ \mathbf{\Omega}(t)&\triangleq[\omega_x(t),\ \omega_y(t),\ \omega_z(t)]^T,\end{aligned} \quad \text{(Б.1)}$$

где, как и выше, знак T обозначает операцию транспонирования. Основываясь на идее подвижной системы координат [310], линейную и угловую скорости манипулятора можно получить из скоростей входящих в манипулятор сочленений:

$$\begin{bmatrix}\mathbf{v}(t)\\ \mathbf{\Omega}(t)\end{bmatrix}=\mathbf{J}(\mathbf{q})\dot{\mathbf{q}}(t)=[\mathbf{J}_1(\mathbf{q}),\ \mathbf{J}_2(\mathbf{q}),\ \ldots,\ \mathbf{J}_6(\mathbf{q})]\dot{\mathbf{q}}(t), \quad \text{(Б.2)}$$

где $\mathbf{J}(\mathbf{q})$ — матрица порядка 6×6, в которой i-й столбец вектора $\mathbf{J}_i(\mathbf{q})$ определяется [310] в виде

$$\mathbf{J}_i(\mathbf{q})=\begin{cases}\begin{bmatrix}\mathbf{z}_{i-1}\times{}^{i-1}\mathbf{p}_6\\ \mathbf{z}_{i-1}\end{bmatrix}, & \text{если } i\text{-е сочленение — вращательное,}\\ \begin{bmatrix}\mathbf{z}_{i-1}\\ 0\end{bmatrix}, & \text{если } i\text{-е сочленение — поступательное.}\end{cases} \quad \text{(Б.3)}$$

где $\dot{\mathbf{q}}(t)=[\dot{q}_1(t),\ \ldots,\ \dot{q}_6(t)]^T$ — вектор скорости сочленения манипулятора, $\times$ — знак векторного произведения, ${}^{i-1}\mathbf{p}_6$ — положе-

ние начала системы координат конечного звена манипулятора относительно системы координат $(i-1)$-го звена, записанное в базовой системе координат, а $\mathbf{z}_{i-1}$ — единичный вектор вдоль оси движения i-го сочленения, записанный в базовой системе координат.

Для шестизвенного манипулятора с шарнирными сочленениями якобиан может быть найден в виде

$$\mathbf{J}(\theta)=\begin{bmatrix} \mathbf{z}_0\times{}^0\mathbf{p}_6 & \mathbf{z}_1\times{}^1\mathbf{p}_6 & \dots & \mathbf{z}_5\times{}^5\mathbf{p}_6 \\ \mathbf{z}_0 & \mathbf{z}_1 & \dots & \mathbf{z}_5 \end{bmatrix}, \qquad \text{(Б.4)}$$

Для манипулятора робота Пума (рис. 2.11) и матриц преобразования координат его звеньев (рис. 2.13) элементы якобиана определяются следующим образом:

$$\mathbf{J}_1(\theta)=$$
$$=\begin{bmatrix} -S_1[d_6(C_{23}C_4S_5+S_{23}C_5)+S_{23}d_4+a_3C_{23}+a_2C_2]-C_1(d_6S_4S_5+d_2) \\ C_1[d_6(C_{23}C_4S_5+S_{23}C_5)+S_{23}d_4+a_3C_{23}+a_2C_2]-S_1(d_6S_4S_5+d_2) \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\mathbf{J}_2(\theta)=\begin{bmatrix} d_6C_1S_4S_5+d_2C_1 \\ d_6S_1S_4S_5+d_2S_1 \\ J_{2z} \\ -S_1 \\ C_1 \\ 0 \end{bmatrix},$$

где

$$J_{2z}=-S_1[d_6S_{23}C_4S_5-d_6C_{23}C_5-d_4C_{23}+a_3S_{23}+a_2S_2]$$
$$-C_1[d_6C_{23}C_4S_5+d_6S_{23}C_5+d_4S_{23}+a_3C_{23}+a_2C_2],$$

$$\mathbf{J}_3(\theta)=\begin{bmatrix} d_6C_1S_4S_5 \\ d_6S_1S_4S_5 \\ J_{3z} \\ -S_1 \\ C_1 \\ 0 \end{bmatrix},$$

где

$$J_{3z}=-S_1[d_6S_3C_4S_5-d_6C_3C_5-d_4C_3+a_3S_4]-$$
$$-C_1[d_6C_3C_4S_5+d_6S_3C_5+d_4S_3+a_3C_3],$$

$$\mathbf{J}_4(\theta)=\begin{bmatrix} S_1S_{23}(d_6C_5+d_4)-d_6C_{23}S_4S_5 \\ d_6C_{23}C_4S_5-C_1S_{23}(d_6C_5+d_4) \\ d_6C_1S_{23}S_4S_5-d_6S_1S_{23}C_4S_5 \\ C_1S_{23} \\ S_1S_{23} \\ C_{23} \end{bmatrix},$$

$$\mathbf{J}_5(\theta)=\begin{bmatrix} d_6S_{23}S_4C_5 \\ d_6S_{23}S_4S_5 \\ d_6C_1C_{23}S_4C_5+d_6S_1C_4C_5+d_6S_1C_{23}S_4S_5-d_6C_1C_4S_5 \\ -C_1C_{23}S_4-S_1C_4 \\ -S_1C_{23}S_4+C_1C_4 \\ S_{23}S_4 \end{bmatrix},$$

$$\mathbf{J}_6(\theta)=\begin{bmatrix} d_6(S_1C_{23}C_4+C_1S_4)S_5+d_6S_1S_{23}C_5 \\ -d_6(C_1C_{23}C_4-S_1S_4)S_5-d_6C_1S_{23}C_5 \\ 0 \\ (C_1C_{23}C_4-S_1S_4)S_5+C_1S_{23}C_5 \\ (S_1C_{23}C_4+C_1S_4)S_5+S_1S_{23}C_5 \\ -S_{23}C_4S_5+C_{23}C_5 \end{bmatrix},$$

где $S_i=\sin\theta_i$, $C_i=\cos\theta_i$, $S_{ij}=\sin(\theta_i+\theta_j)$

и

$$C_{ij}=\cos(\theta_i+\theta_j).$$

Если требуется управлять манипулятором вдоль или вокруг осей системы координат манипулятора, необходимо выразить линейную и угловую скорости в системе координат манипулятора. Это можно сделать умножением $\mathbf{v}(t)$ и $\boldsymbol{\Omega}(t)$ на матрицу вращения $[{}^0\mathbf{R}_6]^T$ размерностью 3×3, где ${}^0\mathbf{R}_6$ — матрица вращения манипулятора, связывающая положение системы координат конечного звена манипулятора с базовой системой координат. Таким образом,

$$\begin{bmatrix} {}^6\mathbf{v}_0(t) \\ {}^6\boldsymbol{\Omega}_0(t) \end{bmatrix}=\begin{bmatrix} [{}^0\mathbf{R}_6]^T & \mathbf{0} \\ \mathbf{0} & [{}^0\mathbf{R}_6]^T \end{bmatrix}\begin{bmatrix} \mathbf{v}(t) \\ \mathbf{R}(t) \end{bmatrix}=\begin{bmatrix} {}^6\mathbf{R}_0 & \mathbf{0} \\ \mathbf{0} & {}^6\mathbf{R}_0 \end{bmatrix}[\mathbf{J}(\mathbf{q})]\,\dot{\mathbf{q}}(t), \qquad \text{(Б.5)}$$

где $\mathbf{0}$ — нулевая матрица размерностью 3×3.

Б.2. Метод дифференциального перемещения и вращения

В работе [229] используется однородная матрица преобразования для нахождения дифференциального перемещения и вращения относительно системы координат, из которых определяется якобиан манипулятора. Для заданной системы координат $\mathbf{T}$ звена манипулятора дифференциальное изменение в $\mathbf{T}$ соответствует дифференциальному перемещению или вращению вдоль или вокруг базовых осей координат, т. е.

$$\mathbf{T}+d\mathbf{T}=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-\delta_z&\delta_y&0\\\delta_z&1&-\delta_x&0\\-\delta_y&\delta_x&1&0\\0&0&0&1\end{bmatrix}\mathbf{T} \tag{Б.6}$$

или

$$d\mathbf{T}=\left[\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-\delta_z&\delta_y&0\\\delta_z&1&-\delta_x&0\\-\delta_y&\delta_x&1&0\\0&0&0&1\end{bmatrix}-\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\right]\mathbf{T}=\Delta\mathbf{T}, \tag{Б.7}$$

где

$$\Delta\triangleq\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-\delta_z&\delta_y&0\\\delta_z&1&-\delta_x&0\\-\delta_y&\delta_x&1&0\\0&0&0&1\end{bmatrix}-\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}, \tag{Б.8}$$

$\delta=(\delta_x,\ \delta_y,\ \delta_z)^T$ — дифференциальное вращение вокруг основных осей базовой системы координат, а $\mathbf{d}=(d_x,\ d_y,\ d_z)^T$ — дифференциальное перемещение вдоль основных осей базовой системы координат. Аналогично дифференциальное изменение в $\mathbf{T}$ может быть записано в соответствии с дифференциальными перемещением и вращением вдоль и вокруг системы координат $\mathbf{T}$:

$$\mathbf{T}+d\mathbf{T}=\mathbf{T}\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-\delta_z&\delta_y&0\\\delta_z&1&-\delta_x&0\\-\delta_y&\delta_x&1&0\\0&0&0&1\end{bmatrix} \tag{Б.9}$$

или

$$d\mathbf{T}=\mathbf{T}\left[\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-\delta_z&\delta_y&0\\\delta_z&1&-\delta_x&0\\-\delta_y&\delta_x&1&0\\0&0&0&1\end{bmatrix}-\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\right]\triangleq(\mathbf{T})({}^T\Delta), \tag{Б.10}$$

где ${}^T\Delta$ имеет ту же структуру, что и в уравнении (В.8) с учетом, что определения δ и $\mathbf{d}$ — различны. $\delta=(\delta_x,\ \delta_y,\ \delta_z)^T$ — дифференциальное вращение вокруг основных осей системы координат $\mathbf{T}$, а $\mathbf{d}=(d_x,\ d_y,\ d_z)^T$ — дифференциальное перемещение вдоль основных осей системы координат $\mathbf{T}$. Связь между Δ и ${}^T\Delta$ получаем из уравнений (Б.7) и (В.10):

$$\Delta\mathbf{T}=(\mathbf{T})({}^T\Delta)$$

или

$${}^T\Delta=\mathbf{T}^{-1}\Delta\mathbf{T}. \tag{Б.11}$$

С учетом уравнения (2.2-27) уравнение (Б.11) имеет вид

$${}^T\Delta=\mathbf{T}^{-1}\Delta\mathbf{T}=$$

$$=\begin{bmatrix}\mathbf{n}\cdot(\delta\times\mathbf{n})&\mathbf{n}\cdot(\delta\times\mathbf{s})&\mathbf{n}\cdot(\delta\times\mathbf{a})&\mathbf{n}\cdot(\delta\times\mathbf{p})+\mathbf{d}\\\mathbf{s}\cdot(\delta\times\mathbf{n})&\mathbf{s}\cdot(\delta\times\mathbf{s})&\mathbf{s}\cdot(\delta\times\mathbf{a})&\mathbf{s}\cdot(\delta\times\mathbf{p})+\mathbf{d}\\\mathbf{a}\cdot(\delta\times\mathbf{n})&\mathbf{a}\cdot(\delta\times\mathbf{s})&\mathbf{a}\cdot(\delta\times\mathbf{a})&\mathbf{a}\cdot(\delta\times\mathbf{p})+\mathbf{d}\\0&0&0&0\end{bmatrix}, \tag{Б.12}$$

где $\delta=(\delta_x,\ \delta_y,\ \delta_z)^T$ — дифференциальное вращение вокруг основных осей базовой системы координат, а $\mathbf{d}=(d_x,\ d_y,\ d_z)^T$ — дифференциальное перемещение вдоль основных осей базовой системы координат. Используя векторные тождества

$$\mathbf{x}\cdot(\mathbf{y}\times\mathbf{z})=-\mathbf{y}\cdot(\mathbf{x}\times\mathbf{z})=\mathbf{y}\cdot(\mathbf{z}\times\mathbf{x})$$

и

$$\mathbf{x}\cdot(\mathbf{x}\times\mathbf{y})=0,$$

уравнение (Б.12) можно записать в виде

$$
{}^{T}\Delta = \begin{bmatrix} 0 & -\delta\cdot(\mathbf{n}\times\mathbf{s}) & \delta\cdot(\mathbf{a}\times\mathbf{n}) & \delta\cdot(\mathbf{p}\times\mathbf{n})+\mathbf{d}\cdot\mathbf{n} \\ \delta\cdot(\mathbf{n}\times\mathbf{s}) & 0 & -\delta\cdot(\mathbf{s}\times\mathbf{a}) & \delta\cdot(\mathbf{p}\times\mathbf{s})+\mathbf{d}\cdot\mathbf{s} \\ -\delta\cdot(\mathbf{a}\times\mathbf{n}) & \delta\cdot(\mathbf{s}\times\mathbf{a}) & 0 & \delta\cdot(\mathbf{p}\times\mathbf{a})+\mathbf{d}\cdot\mathbf{a} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{Б.13}
$$

Поскольку координатные оси **n**, **s**, **a** — ортогональны, имеем

$$
\mathbf{n}\times\mathbf{s}=\mathbf{a},\quad \mathbf{s}\times\mathbf{a}=\mathbf{n},\quad \mathbf{a}\times\mathbf{n}=\mathbf{s},
$$

и уравнение (Б.13) запишется в виде

$$
{}^{T}\Delta = \begin{bmatrix} 0 & -\delta\cdot\mathbf{a} & \delta\cdot\mathbf{s} & \delta\cdot(\mathbf{p}\times\mathbf{n})+\mathbf{d}\cdot\mathbf{n} \\ \delta\cdot\mathbf{a} & 0 & -\delta\cdot\mathbf{n} & \delta\cdot(\mathbf{p}\times\mathbf{s})+\mathbf{d}\cdot\mathbf{s} \\ -\delta\cdot\mathbf{s} & \delta\cdot\mathbf{n} & 0 & \delta\cdot(\mathbf{p}\times\mathbf{a})+\mathbf{d}\cdot\mathbf{a} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{Б.14}
$$

Если задать элементы ${}^{T}\Delta$ в виде

$$
{}^{T}\Delta = \begin{bmatrix} 0 & -\mathbf{T}_{\delta_z} & \mathbf{T}_{\delta_y} & \mathbf{T}_{d_x} \\ \mathbf{T}_{\delta_z} & 0 & -\mathbf{T}_{\delta_x} & \mathbf{T}_{d_y} \\ -\mathbf{T}_{\delta_y} & \mathbf{T}_{\delta_z} & 0 & \mathbf{T}_{d_z} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{Б.15}
$$

то, приравнивая элементы матриц из уравнений (Б.14) и (Б.15), получим

$$
\begin{aligned}
{}^{T}d_x &= \delta\cdot(\mathbf{p}\times\mathbf{n})+\mathbf{d}\cdot\mathbf{n}=\mathbf{n}\cdot[(\delta\times\mathbf{p})+\mathbf{d}],\\
{}^{T}d_y &= \delta\cdot(\mathbf{p}\times\mathbf{s})+\mathbf{d}\cdot\mathbf{s}=\mathbf{s}\cdot[(\delta\times\mathbf{p})+\mathbf{d}],\\
{}^{T}d_z &= \delta\cdot(\mathbf{p}\times\mathbf{a})+\mathbf{d}\cdot\mathbf{a}=\mathbf{a}\cdot[(\delta\times\mathbf{p})+\mathbf{d}],\\
{}^{T}\delta_x &= \delta\cdot\mathbf{n},\\
{}^{T}\delta_y &= \delta\cdot\mathbf{s},\\
{}^{T}\delta_z &= \delta\cdot\mathbf{a}.
\end{aligned} \tag{Б.16}
$$

Записывая эти уравнения в матричной форме, имеем

$$
\begin{bmatrix} {}^{T}d_x \\ {}^{T}d_y \\ {}^{T}d_z \\ {}^{T}\delta_x \\ {}^{T}\delta_y \\ {}^{T}\delta_z \end{bmatrix} = \begin{bmatrix} [\mathbf{n},\ \mathbf{s},\ \mathbf{a}]^T & [(\mathbf{p}\times\mathbf{n}),\ (\mathbf{p}\times\mathbf{s}),\ (\mathbf{p}\times\mathbf{a})]^T \\ \mathbf{0} & [\mathbf{n},\ \mathbf{s},\ \mathbf{a}]^T \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix}, \tag{Б.17}
$$

где **0** — нулевая подматрица размерностью 3×3. Уравнение (Б.17) дает связь дифференциальных перемещения и вращения в базовой системе координат и дифференциальных перемещения и вращения в системе координат **T**.

Применяя уравнение (Б.10) к кинематическому уравнению серийного шестизвенного манипулятора, получим дифференциал ${}^{0}\mathbf{T}_6$:

$$
d^{0}\mathbf{T}_6 = {}^{0}\mathbf{T}_6\,{}^{T_6}\Delta. \tag{Б.18}
$$

В случае шестизвенного манипулятора дифференциальное изменение движения i-го звена вызовет соответствующее изменение в ${}^{0}\mathbf{T}_6\,{}^{T_6}\Delta$:

$$
d^{0}\mathbf{T}_6 = {}^{0}\mathbf{T}_6\,{}^{T_6}\Delta = {}^{0}\mathbf{A}_1\,{}^{1}\mathbf{A}_2\ldots{}^{i-2}\mathbf{A}_{i-1}\;{}^{i-1}\Delta_i\,{}^{i-1}\mathbf{A}_i\ldots{}^{5}\mathbf{A}_6, \tag{Б.19}
$$

где ${}^{i-1}\Delta_i$ определяется как преобразование дифференциального изменения вдоль или вокруг оси движения i-го сочленения:

$$
{}^{i-1}\Delta_i = \begin{cases} \begin{bmatrix} 0 & -d\theta_i & 0 & 0 \\ d\theta_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & \text{если } i\text{-е сочленение — вращательное,} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & dd_i \\ 0 & 0 & 0 & 0 \end{bmatrix}, & \text{если } i\text{-е сочленение — поступательное} \end{cases}
$$

Используя уравнение (Б.19), получим ${}^{T_6}\Delta$ из дифференциального изменения движения i-го сочленения:

$${}^{T_6}\Delta = \left({}^{i-1}\mathbf{A}_i\,{}^{i}\mathbf{A}_{i+1}\ldots{}^{5}\mathbf{A}_6\right)^{-1}\,{}^{i-1}\Delta_i\left({}^{i-1}\mathbf{A}_i\,{}^{i}\mathbf{A}_{i+1}\ldots{}^{5}\mathbf{A}_6\right) = \mathbf{U}_i^{-1}\,{}^{i-1}\Delta_i\mathbf{U}_i, \tag{Б.21}$$

где

$$
\mathbf{U}_i = {}^{i-1}\mathbf{A}_i\,{}^{i}\mathbf{A}_{i+1}\ldots{}^{5}\mathbf{A}_6.
$$

Записывая $\mathbf{U}_i$ в виде общей однородной матрицы преобразования, получим

$$\mathbf{U}_i = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad \text{(Б.22)}$$

С учетом уравнений (Б.20) и (Б.22) для i-го вращательного сочленения, уравнение (Б.21) примет вид

$${}^{T_6}\Delta = \begin{bmatrix} 0 & -a_z & s_z & p_x n_y & -p_y n_x \\ a_z & 0 & -n_z & p_x s_y & -p_y s_x \\ -s_z & n_z & 0 & p_x a_y & -p_y a_x \\ 0 & 0 & 0 & & 0 \end{bmatrix} d\theta_i. \qquad \text{(Б.23)}$$

Для случая призматического i-го сочленения уравнение (Б.21) запишется в виде

$${}^{T_6}\Delta = \begin{bmatrix} 0 & 0 & 0 & n_z \\ 0 & 0 & 0 & s_z \\ 0 & 0 & 0 & a_z \\ 0 & 0 & 0 & 0 \end{bmatrix} dd_i. \qquad \text{(Б.24)}$$

Для элементов ${}^{T_6}\Delta$, которые были определены в уравнении (Б.15), приравнивая элементы матриц в уравнениях (Б.15) и (Б.23) [или (Б.24)], получим

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = \begin{cases} \begin{bmatrix} p_x n_y - p_y n_x \\ p_x s_y - p_y s_x \\ p_x a_y - p_y a_x \\ n_z \\ s_z \\ az \end{bmatrix} d\theta_i, & \text{если } i\text{-е сочленение — вращательное,} \\ \begin{bmatrix} n_z \\ s_z \\ a_z \\ 0 \\ 0 \\ 0 \end{bmatrix} dd_i, & \text{если } i\text{-сочленение — поступательное.} \end{cases} \qquad \text{(Б.25)}$$

Таким образом, якобиан манипулятора может быть получен из уравнения (Б.25) для $i = 1, 2, \ldots, 6$:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = \mathbf{J}(\mathbf{q}) \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix}, \qquad \text{(Б.26)}$$

где столбцы матрицы якобиана получаются из уравнения (Б.25). Для манипулятора робота Пума (рис. 2.11) и его матриц преобразования звеньев (рис. 2.13) якобиан определяется в виде

$$\mathbf{J}_1(\theta) = \begin{bmatrix} J_{1x} \\ J_{1y} \\ J_{1z} \\ -[S_{23}(C_4C_5C_6 - S_4S_6) + C_{23}S_5C_6], \\ S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \\ -S_{23}C_4S_5 + C_{23}C_5 \end{bmatrix},$$

где

$$J_{1x} = [d_6(C_{23}C_4S_5 + S_{23}C_5) + d_4S_{23} + a_3C_{23} + a_2C_2](S_4C_5C_6 + C_4S_6) - \\ - (d_6S_4S_5 + d_2)[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6]),$$

$$J_{1y} = [d_6(C_{23}C_4S_5 + S_{23}C_5) + d_4S_{23} + a_3C_{23} + a_2C_2](-S_4C_5S_6 + C_4C_6) - \\ - (d_6S_4S_5 + d_2)[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6],$$

$$J_{1z} = [d_6(C_{23}C_4S_5 + S_{23}C_5) + d_4S_{23} + a_3C_{23} + a_2C_2](S_4S_5) - \\ - (d_6S_4S_5 + d_2)(C_{23}C_4S_5 + S_{23}C_5);$$

$$\mathbf{J}_2(\theta) = \begin{bmatrix} J_{2x} \\ J_{2y} \\ J_{2z} \\ S_4C_5C_6 + C_4S_6 \\ -S_4C_5S_6 + C_4S_6 \\ S_4S_5 \end{bmatrix},$$

где

$$J_{2x} = (d_6S_3C_5 + d_6C_3C_4S_5 + d_4S_3 + a_3C_3 + a_2)(S_5C_6) - \\ - (-d_6C_3C_5 + d_6S_3C_4S_5 - d_4C_3 + a_3S_3)(C_4C_5C_6 - S_4S_6),$$

$$J_{2y} = -(d_6S_3C_5 + S_6C_3C_4S_5 + d_4S_3 + a_3C_3 + a_2)(S_5S_6) + \\ + (-d_6C_3C_5 + d_6S_3C_4S_5 - d_4C_3 + a_3S_3)(C_4C_5S_6 + S_4C_6),$$

$$J_{2z} = -(d_6S_3C_5 + d_6C_3C_4S_5 + d_4S_3 + a_3C_3 + a_2)C_5 - \\ - (-d_6C_3C_5 + d_6S_3C_4S_5 - d_4C_3 + a_3S_3)(C_4S_5);$$

$$\mathbf{J}_3(\theta) = \begin{bmatrix} (a_3 + d_6C_4S_5)(S_5C_6) + (d_4 + d_6C_5)(C_4C_5C_6 - S_4S_6) \\ -(a_3 + d_6C_4S_5)(S_5S_6) - (d_4 + d_6C_5)(C_4C_5S_6 + S_4C_6) \\ -(a_3 + d_6C_4S_5)C_5 + (d_4 + d_6C_5)C_4C_5 \\ S_4C_5C_6 + C_4S_6 \\ -S_4C_5S_6 + C_4S_6 \\ S_4S_5 \end{bmatrix},$$

$$\mathbf{J}_4(\theta) = \begin{bmatrix} d_6S_5S_6 \\ d_6S_5C_6 \\ 0 \\ -S_5C_6 \\ S_5S_6 \\ C_5 \end{bmatrix},$$

$$\mathbf{J}_5(\theta) = \begin{bmatrix} d_6C_6 \\ -d_6S_6 \\ 0 \\ S_6 \\ C_6 \\ 0 \end{bmatrix}, \quad \mathbf{J}_6(\theta) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Б.3. Определение якобиана из уравнений движения Ньютона — Эйлера

В рассмотренных двух методах якобиан был получен в аналитической форме. Возможно численное определение элементов якобиана во времени в явном виде из уравнений движения Ньютона — Эйлера. Это определение основывается на том, что отношения малых величин ускорений конечного звена манипулятора к малым величинам ускорений сочленений являются элементами якобиана, если исключить нелинейные компоненты ускорений из уравнений движения Ньютона — Эйлера. Ускорения манипулятора могут быть получены путем нахождения производной по времени от вектора скорости из уравнения (Б.2):

$$\begin{bmatrix} \dot{\mathbf{v}}(t) \\ \dot{\Omega}(t) \end{bmatrix} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}}(t) + \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}(t), \qquad (Б.27)$$

где $\ddot{\mathbf{q}}(t) = [\ddot{q}_1(t), \ldots, \ddot{q}_6(t)]^T$ — вектор ускорения сочленения манипулятора. Первый член уравнения (Б.27) дает линейную связь между ускорениями манипулятора и сочленений. Второй член характеризует нелинейные компоненты ускорений и является функцией скорости сочленений. Таким образом, линейная связь между ускорениями манипулятора и ускорениями сочленений может быть найдена из уравнений движения Ньютона — Эйлера с помощью первого члена в уравнении (Б.27). Из табл. 3.3 имеем следующие рекурсивные кинематические выражения (здесь рассматриваются только манипуляторы с шарнирными сочленениями):

$${}^{i}\mathbf{R}_0\omega_i = {}^{i}\mathbf{R}_{i-1}({}^{i-1}\mathbf{R}_0\omega_{i-1} + \mathbf{z}_0\dot{q}_i), \qquad (Б.28)$$

$${}^{i}\mathbf{R}_0\dot{\omega}_i = {}^{i}\mathbf{R}_{i-1}[{}^{i-1}\mathbf{R}_0\dot{\omega}_{i-1} + \mathbf{z}_0\ddot{q}_i + ({}^{i-1}\mathbf{R}_0\omega_{i-1}) \times \mathbf{z}_0\dot{q}_i], \qquad (Б.29)$$

$${}^{i}\mathbf{R}_i\dot{\mathbf{v}}_i = ({}^{i}\mathbf{R}_0\dot{\omega}_i) \times ({}^{i}\mathbf{R}_0\mathbf{p}_i^*) + ({}^{i}\mathbf{R}_0\omega_i) \times \\ \times [({}^{i}\mathbf{R}_0\omega_i) \times ({}^{i}\mathbf{R}_0\mathbf{p}_i^*)] + {}^{i}\mathbf{R}_{i-1}({}^{i-1}\mathbf{R}_0\dot{\mathbf{v}}_{i-1}). \qquad (Б.30)$$

Третий член уравнения (Б.29) и второй член уравнения (Б.30), представляют нелинейные кориолисово и центробежное ускорения. Исключая эти члены в уравнениях (Б.29) и (Б.30), получим линейную связь между ускорениями манипулятора и ускорениями сочленений. Если использовать входной единичный вектор ускорения $(\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_6)^T = (1, 0, 0, \ldots, 0)^T$, $(\dot{q}_1, q_2, \ldots, q_6)^T = (0, 1, 0, \ldots, 0)^T$, $(\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_6)^T = (0, 0, 0, \ldots, 1)^T$ и т. д., то столбцы якобиана могут быть выделены, так как первый член в уравнении (Б.27) является линейным, а второй (нелинейный) член исключен. Этот численный метод требует около $24n(n+1)/2$ операций умножения и $19n(n+1)/2$ операций сложения, где n — число степеней свободы. Кроме того, необходимо произвести $18n$ операций умножения и $12n$ операций сложения для преобразования ускорений манипулятора из системы координат его звеньев в систему координат манипулятора.

Хотя эти три метода эквивалентны с точки зрения нахождения якобиана, последний метод удобен для устройства управления, использующего уравнения движения Ньютона — Эйлера. Поскольку для определения моментов в сочленениях из уравнений движения Ньютона — Эйлера разработаны способы параллельного вычисления [158], то якобиан может быть также вычислен этими способами. Однако последний метод дает только численные значения якобиана и не позволяет получить его аналитическую форму.

Более подробно вопросы, затронутые в данном приложении, можно найти в работах [222, 229, 310].

ЛИТЕРАТУРА

1. Aggarwal J. K., Badler N. I. (eds.) Motion and Time Varying Imagery, Special Issue. IEEE Trans. Pattern Anal. Machine Intelligence. PAMI-2, No. 6, pp. 493—588, 1980.
2. Agin G. J. Representation and Description of Curved Objects, Memo AIM-173, Artificial Intelligence Laboratory, Stanford University, Palo Alto, Calif., 1972.
3. Albus J. S. A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller, Trans. ASME, J. Dynamic Systems, Measurement and Control, pp. 220—227, 1975.
4. Ambler A. P. A Versatile System for Computer Controlled Assembly, Artificial Intelligence, 6, No. 2, pp. 129—156, 1975.
5. Ambler A. P., Popplestone R. J. Inferring the Positions of Bodies from Specified Spatial Relationships, Artificial Intelligence, 6, No. 2, pp. 157—174, 1975.
6. Armstrong W. M. Recursive Solution to the Equations of Motion of an N-link Manipulator, Proc. 5th World Congr., Theory of Machines, Mechanisms, 2, pp. 1343—1346, 1979.
7. Astrom K. J., Eukhoff P. System Identification — A Survey, Automatica, 7, pp. 123—162, 1971.
8. Baer A., Eastman C., Henrion M. Geometric Modelling: A Survey, Computer Aided Design, 11, No. 5, pp. 253—272, 1979.
9. Bajcsy R., Lieberman L. Texture Gradient as a Depth Cue, Comput. Graph. Image Proc., 5, No. 1, pp. 52—67, 1976.
10. Ballard D. H. Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recog. 13, No. 2, pp. 111—122, 1981.
11. Ballard D. H., Brown C. M. Computer Vision, Prentice-Hall, Englewood Cliffs, N. J., 1982.
12. Barnard S. T., Fischler M. A. Computational Stereo, Computing Surveys. 14, No. 4, pp. 553—572, 1982.
13. Barr A., Cohen P., Feigenbaum E. A. The Handbook of Artificial Intelligence, vols 1, 2, and 3, William Kaufmann, Inc., Los Altos, Calif, 1981—1982.
14. Barrow H. G., Tenenbaum J. M. Experiments in Model Driven Scene Segmentation, Artificial Intelligence, 8, No. 3, p. 241—274, 1977.
15. Barrow H. G., Tenenbaum J. M. Interpreting Line Drawings as Three-Dimensional Surfaces, Artificial Intelligence, 17, pp. 76—116, 1981.
16. Bejczy A. K. Robot Arm Dynamics and Control, Technical Memo 33—669, Jet Propulsion Laboratory, Pasadena, Calif, 1974.
17. Bejczy A. K. Dynamic Models and Control Equations for Manipulators, Technical Memo 715—19, Jet Propulsion Laboratory, Pasadena, Calif, 1979.

18. Bejczy A. K. Sensors, Controls, and Man — Machine Interface for Advanced Teleoperation, Science, 208, pp. 1327—1335, 1980.
19. Bejczy A. K., Lee S. Robot Arm Dynamic Model Reduction for Control, Proc. 22nd IEEE Conf. on Decision and Control, San Antonio, Tex., pp. 1466—1476, 1983.
20. Bejczy A. K., Paul R. P. Simplified Robot Arm Dynamics for Control. Proc. 20th IEEE Conf. Decision and Control, San Diego, Calif, pp. 261—262, 1981.
21. Bellman R. Introduction to Matrix Analysis, 2d edition, McGraw-Hill, New York, 1970.
22. Beni G., et al. Dynamic Sensing for Robots: An Analysis and Implementation, Intl. J. Robotics Res. 2, No. 2, pp. 51—61, 1983.
23. Binford T. O. The AL Language for Intelligent Robots, in Proc. IRIA Sem. Languages and Methods of Programming Industrial Robots (Rocquencourt, France), pp. 73—87, 1979.
24. Blum H. A Transformation for Extracting New Descriptors of Shape, in Models for the Perception of Speech and Visual Form (Q. Wathen-Dunn, ed.), MIT Press, Cambridge, Mass, 1967.
25. Bobrow J. E., Dubowsky S. Gibson J. S. On the Optimal Control of Robot Manipulators with Actuator Constraints, Proc. 1983 American Control Conf., San Francisco, Calif., pp. 782—787, 1983.
26. Bolles R., Paul R. An Experimental System for Computer Controlled Mechanical Assembly, Stanford Artificial Intelligence Laboratory Memo AIM-220, Stanford University, Palo Alto, Calif, 1973.
27. Bonner S., Shin K. G. A Comparative Study of Robot Languages, IEEE Computer, 15, No. 12, pp. 82—96, 1982.
28. Brady J. M. (ed.) Computer Vision, North-Holland Publishing Co., Amsterdam, 1981.
29. Brady J. M., et al. Robot Motion: Planning and Control, MIT Press, Cambridge, Mass., 1982.
30. Bribiesca E. Arithmetic Operations Among Shapes Using Shape Numbers, Pattern Recog., vol. 13, no. 2, pp. 123—138, 1981.
31. Bribiesca E. Guzman A. How to Describe Pure Form and How to Measure Differences in Shape Using Shape Numbers, Pattern Recog., 12, No. 2, pp. 101—112, 1980.
32. Brice C., Fennema C. Scene Analysis Using Regions, Artificial Intelligence, vol. 1, no. 3, pp. 205—226, 1970.
33. Brooks R. A. Symbolic Reasoning Among 3-D Models and 2D Images, Artificial Intelligence, 17, pp. 285—348., 1981.
34. Brooks R. A. Solving the Find-Path Problem by Good Representation of Free Space, IEEE Trans. Systems, Man. Cybern., vol. SMC-13, pp. 190—197, 1983a.
35. Brooks R. A. Planning Collision-Free Motion for Pickand Place Operations, Intl. J. Robotics Res., vol. 2, no. 4, pp. 19—44, 1983b.
36. Brooks R. A., and Lozano — Perez T. «A Busdivision Algorithm in Configuration Space for Find-Path with Rotation», Proc. Intl. Joint Conf. Artificial Intelligence (Karlsure, W. Germany), pp. 799—808, 1983.
37. Bryson A. E., and Ho Y. C. Applied Optimal Control, John Wiley, New York, 1975.
38. Canali C. et al., Sensori di Prossimita Electtronici, Fisica e Technologia, 4, no. 2, pp. 95—123, 1981.
39. Canali C. et al. An Ultrasonic Proximity Sensor Operating in Air, Sensors and Actuators, 2, no. 1, pp. 97—103, 1981.
40. Catros J. Y., Espiau B., Use of Optical Proximity Sensors in Robotics, Nouvel Automatisme, 25, No. 14, pp. 47—53, 1980.
41. Chase M. A. Vector Analysis of Linkages, Trans. ASME, J. Engr. Industry, Series B, 85, pp. 289—297, 1963.

42. Chase M. A., Bayazitoglu Y. O. Development and Application of a Generalized d'Alembert Force for Multifreedom Mechanical Systems. Trans. ASME, J. Engr. Industry, Series B. 93, p. 317—327, 1971.
43. Chang C. L., Lee R. C. T. Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.
44. Chaudhuri B. B. A Note on East Algorithms for Spatial Domain Techniques in Image Processing IEEE Trans. Systems, Man, Cybern., SMC-13, No. 6, pp. 1166—1169, 1983.
45. Chow C. K., Kaneko T. Automatic Boundary Detection of the Left Ventricle from Cineangiograms, Comput. and Biomed. Res., 5, p. 388—410, 1972.
46. Chung M. J. Adaptive Control Strategies for Computer-Controlled Manipulators, Ph. D. Dissertation, The Computer Information, and Control Engineering Program, University of Michigan, Ann Arbor, Mich. 1983.
47. Cowart A. E., Snuder W. E., Ruedger W. H. 1983. The Detection of Unresolved Targets Using the Hough Transform, Comput. Vision, Graphics, and Image Proc., 21, pp. 222—238, 1983.
48. Craig J. J. JARS: JPL Autonomous Robot System, Robotics and Teleoperators Group, Jet Propulsion Laboratory, Pasadena, Calif, 1980.
49. Craig J. J. Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, Mass, 1986.
50. Crandall S. H., Karnopp D. C., Kurtz E. F. et al. Dynamics of Mechanical and Electromechanical Systems, McGraw-Hill, New York, 1968.
51. Cross G. R., Jain A. K. Markov Random, Field Texture Models, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5, No. 1, pp. 25—39, 1983.
52. Darringer J. A., Blasgen M. W. MAPLE: A High Level Language for Research, in Mechanical Assembly, IBM Research Report RC 5606, IBM T. J. Watson Research Center, Yorktown Heights, N. Y., 1975.
53. Davies E. R., Plummer A. P. N. Thinning Algorithms: A Critique and a New Methodology, Pattern Recog., 14, pp. 53—63, 1981.
54. Davis L. S. A Survey of Edge Detection Techniques, Comput. Graphics Image Proc., 4, pp. 248—270, 1975.
55. Davis R. H., Comacho M. The Application of Logic Programming to the Generation of Plans for Robots», Robotica, 2, pp. 137—146, 1984.
56. Denavit J. Description and Displacement Analysis of Mechanisms Based on 2×2 Dual Matrices, Ph. D. Thesis, Mechanical Engineering, Northwestern U., Evanston, III, 1965.
57. Denavit J., Hartenberg R. S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, J. Appl. Mech., 77, pp. 215—221, 1955.
58. Derksen J., Rulifson J. F., Waldinger R. J. The QA4 Langauge Applied to Robot Planning», Tech. Note 65, Stanford Research Institute, Menlo Park, Calif., 1972.
59. Dijkstra E. A Note on Two Problems in Connection with Graphs, Numerische Mathematik, 1. pp. 269—271, 1959.
60. Dodd G. G., Rossol L., (eds.) Computer Vision and Sensor-Based Robots, Plenum, 1979.
61. Doran J. E. Planning and Robots", in Machine Intelligence, 5 (B: Meltzer, D. Michie eds.), American Elsevier, pp. 519—532, 1970.
62. Dorf R. C. Robotics and Automated Manufacturing, Reston Publishin Co., Reston, Va., 1983.
63. Drake S. H. Using Compliance in Lieu of Sensory Feedback for Automatic Assembly, Report T-657, C. S. Draper Laboratory, Cambridge, Mass., 1977.
64. Dubowsky S., DesForges D. T. The Application of Model Referenced Adaptive Control to Robotic Manipulators. Trans. ASME, J. Dynamic Systems, Measurement and Control. 101. pp. 193—200, 1979.
65. Duda R. O., Hart P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures", Comm. ACM. 15, Nol. 1, pp. 11—15, 1972.
66. Duda R. O., Hart P. E. Pattern Classification and Scene Analysis, John Wiley, 1973.

67. Duda R. O., Nitzan D., Barrett P. Use of Range and Reflectance Data to Find Planar Surface Regions, IEEE Trans. Pattern Anal. Machine Intell., PAMI-1, No. 3, pp. 259—271, 1979.
68. Duffy J. Analysis of Mechanisms and Robot Manipulators, John Wiley, 1980.
69. Duffy J., Rooney J. A. Foundation for a Unified Theory of Analysis of Spatial Mechanisms, Trans. ASME, J. Engr. Industry, 97, No. 4, Series B, pp. 1159—1164, 1975.
70. Dyer C. R., Rosenfeld A. Thinning Algorithms for Grayscale Pictures, IEEE Trans. Pattern Anal. Machine Intelligence PAMI-1, No. 1, pp. 88—89, 1979.
71. Engelberger J. F. Robotics in Practice, AMOCOM, N. Y. 1980.
72. Ernst H. A. MH—1, 0 Computer-Oriented Mechanical Hand", Proc. 1962 Spring Joint Computer Conf. San Francisco, Calif., pp. 39—51, 1962.
73. Fahlman S. E. A Planning System for Robot Construction Tasks", Artifial Intelligence 5, No. 1, pp. 1—49, 1974.
74. Fairchild CCD Imaging Catalog. Fairchild Corp., Palo Alto, Calif., 1983.
75. Falb P. L., Wolovich W. A. Decoupling in the Design and Synthesis of Multivariable Control Systems", IEEE Trans. Automatic Control, 12, No. 6, pp. 651—655, 1967.
76. Featherstone R. The Calculation of Robot Dynamics Using Articulated-Body Inertia, Intl. J. Robotics Res., 2, No. 1, pp. 13—30, 1983.
77. Fikes R. E., Hart P. E., Nilsson N. J. Learning and Executing Generalized Robot Plans, Artificial Intelligence, 3, No. 4, pp. 251—288, 1972.
78. Fikes R. E., Nilsson N. J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, 2, No. 3/4, pp. 189—208, 1971.
79. Fink D. G. (ed.) Television Engineering Handbook, McGraw-Hill, N. Y. 1957.
80. Finkel R. et al. An Overview of AL, A Programming Language for Automation, Proc. 4th Intl. Joint Conf. Artificial Intelligence, pp. 758—765, 1975.
81. Franklin J. W. Vanderburg G. J. Programming Vision and Robotics Systems with RAIL", SME Robots VI, pp. 392—406, 1982.
82. Frazer R. A., Duncan W. T., Collan A. R. Elementary Matrices, Cambridge University Press, Cambridge, England, 1960.
83. Freeman H. On the Encoding of Arbitrary Geometric Configurations. IEEE Trans. Elec. Computers, vol. ES-10, pp. 260—268, 1961.
84. Freeman H. Computer Processing of Line Drawings", Comput. Surveys, 6, pp. 57—97, 1974.
85. Freeman H. Shapira R. Determining the Minimum — Area Encasing Rectangle for an Arbitrary Closed Curve", Comm. ACM, 18. No. 7, pp. 409—413, 1975.
86. Freund E. Fast Nonlinear Control with Arbitrary Pole Placement for Industrial Robots and Manipulators", Intl. J. Robotics. Res., 1, No. 1, pp. 65—78, 1982.
87. Fu K. S. Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control, IEEE Trans. Automatic Control. AC-16, No. 2, pp. 70—72, 1971.
88. Fu K. S. Syntactic Pattern Recognition and Applications, Prentice — Hall, Englewood Cliffs. N. J., 1982.
89. Fu K. S. (ed.) Special Issue of Computer on Robotics and Automation, 15, No. 12, 1982.
90. Fu K. S., and Mui J. K. A Survey of Image Segmentation", Pattern Recog., 13, No. 1, pp. 3—16, 1981.
91. Galey B., Hsia P. A Survey of Robotics Sensor Technology, Proc. 12th Annual Southeastern Symp. System Theory, pp. 90—93, 1980.
92. Gantmacher F. R. The Theory of Matrices, 2 vols., Chelsea N. Y., 1959.

93. Geschke C. C. A System for Programming and Controlling Sensor-Based Robot Manipulators IEEE Trans. Pattern Anal. Machine Intell., PAMI-5, No. 1, pp. 1—7, 1983.
94. Gilbert E. G., Ha I. J. An Approach to Nonlinear Feedback Control with Application to Robotics", IEEE Trans. Systems, Man, Cybern., SMC-14, No. 2, pp. 101—109, 1984.
95. Gips J. A Syntax — Directed Program that Performs a Three-Dimensional Perceptual Task, Pattern Recog., 6, pp. 189—200, 1974.
96. Goldstein H. Classical Mechanics, Addison — Wesley, Reading, Mass., 1950.
97. Gonzalez R. C. How Vision Systems See, Machine Design, 55 No. 10, pp. 91—96, 1983.
98. Gonzales R. C. Industrial Computer Vision, in Computer — Based Automation (J. T. Tou, ed.), Plenum N. Y., pp. 345—385, 1985.
99. Gonzalez R. C., Computer Vision, McGraw-Hill Yearbook of Science and Technology, McGraw-Hill, 1985.
100. Gonzalez R. C. Digital Image Enhacement and Restoration, in Handbook of Pattern Recognition and Image Processing, (T. Young and K. S. Fu, eds.), Academic Press, New York, pp. 191—213, 1986.
101. Gonzalez R. C., Fittes B. A. Gray — Level Transformations for Interactive Image Enhacement, Mechanisms and Machine Theory, 12, pp. 111—122, 1977.
102. Gonzalez R. C., Safabakhsh R. Computer Vision Techniques for Industrial Applications and Robot Control", Computer, 15, No. 12, pp. 17—32, 1982.
103. Gonzalez R. C., Thomason M. G. Syntactic Pattern Recognition: An Introduction, Addison — Wesley, Reading, Mass, 1978.
104. Gonzalez R. C., Wintz P. Digital Image Processing, Addison — Wesley, Reading, Mass., 1977.
105. Goodman J. W. Introduction to Fourier Optics, McGraw-Hill, 1968.
106. Green C. Application of Theorem Proving to Problem Solving. Proc. 1st Intl. Joint Conf. Artificial Intelligence, Washington D. C., 1969.
107. Grossman D. D., Taylor R. H. Interactive Generation of Object Models with a Manipulator, IEEE Trans. Systems. Man. Cybern. SMC-8 No. 9, pp. 667—679, 1978.
108. Gruver W. A. et al. Industrial Robot Programming Languages: A Comparative Evaluation, IEEE Trans. Systems. Man. Cybern. SMC-14 No. 4, pp. 321—333, 1984.
109. Guzman A., Decomposition of a Visual Scene into Three — Dimensional Bodies, in Automatic Interpretation and Classification of Images (A. Grasseli. ed.), Academic Press. N. Y., 1969.
110. Hackwood S., et al. A Torque — Sensitive Tactile Array for Robotics, Intl. J. Robotics Res., 2, No. 2, pp. 46—50, 1983.
111. Haralick R. M. Statistical and Structural Approaches to Texture, Proc. 4th Intl. Joint Conf. Pattern Recog., pp. 45—60, 1979.
112. Haralick R. M., Shanmugan R., Dinstein I. Textural Features for Image Classification, IEEE Trans. Systems, Man. Cybern. SMC-3 No. 6, pp. 610—621, 1973.
113. Harmon L. D. Automated Tactile Sensing, Intl, J. Robotics Res. 1, No. 2, pp. 3—32, 1982.
114. Harris J. L. 1982. Constant Variance Enhancement — A Digital Processing Techniques, Appl. Optics, 16, pp. 1268—1271, 1977.
115. Hart P. E., Nilsson N. J., Raphael B. A Formal Basis for the Heuristic Determination of Minimum-Cost Paths, IEEE Trans. Systems, Man. Cybern, SMC—4, pp. 100—107, 1968.
116. Hartenberg R. S., Denavit J. Kinematic Synthesis of Linkages, McGraw-Hill, V. Y., 1964.
117. Hayer Roth R. Waterman D., Lenat D. (eds.) Building Expert Systems, Addison Wesley, Reading Mass, 1983.

118. Hemami H., Camana P. C. Nonlinear Feedback in Simple Locomotion Systems, IEEE Trans. Automatic Control. AC-19, pp. 855—860, 1976.
119. Herrick C. N. Television Theory and Servicing, 2d ed., Reston Publishers, Reston, Va., 1976.
120. Hillis D. W. A High Resolution Imaging Touch Sensor", Intl. J. Robotics Res. vol. 1, no. 2, pp. 33—44, 1982.
121. Holland S. W., Rossol L., Ward M. R. CONSIGHT—1: A Vision-Controlled Robot System for Transferring Parts from Belt Conveyors, in Computer Vision and Sensor-Based Robots (G. G. Dobb and L. Rossol, eds.), Plenum N. Y. 1979.
122. Hollerbach J. M. A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE Trans. Systems, Man, Cybern., SMC—10, No. 11, pp. 730—736, 1980.
123. Hollerbach J. M. Dynamic Scalling of Manipulator Trajectories", Trans. ASME J. Dyn. Systems, Measurement and Control. 106, pp. 102—106, 1984.
124. Horn B. K. P. Understanding Image Intensities, Artificial Intelligence, 8, pp. 201—231, 1977.
125. Horowitz S. L., Pavlidis T. Picture Segmentation by a Directed Split-and-Merge Procedure, Proc. 2d Intl. Joint Conf. Pattern Recog., pp. 424—433, 1974.
126. Horowitz R., Tomizuka R. An Adaptive Control Scheme for Mechanical Manipulators — Compensation of Nonlinearity and Decoupling Control. Trans. ASME J. Dynamic Systems. Measurement and Control. 1986.
127. Hough, P. V. C. Methods and Means for Recognizing Complex Patterns, U. S. Patent 3,069,654, 1962.
128. Hu M. K. Visual Pattern Recognition by Moment Invariants", IEEE Trans. Inform. Theory, 8, pp. 179—187, 1962.
129. Huang T. S., Yang G. T., Tang G. Y. "A Fast Two-Dimensional Median Filtering Algorithms", IEEE Trans. Acoust., Speech, Signal Proc., ASSP—27, pp. 13—18, 1979.
130. Huston R. L., Kelly F. A. The Development of Equations of Motion of Single — Arm Robots", IEEE Trans. Systems, Man Cybern, SMC—12, No. 3, pp. 259—266, 1982.
131. Huston R. L., Passerello C. E., Harlow M. W. Dynamics of Multirigid Body Systems. J. Appl. Mech., 45, pp. 889—894, 1978.
132. Inoue H. Force Feedback in Precise Assembly Tasks, MIT Artificial Intelligence Laboratory Memo 308, MIT, Cambridge, Mass., 1974.
133. Ishizuka M., Fu K. S., Yao J. T. P. A Rule — Based Damage Assessment Systems for Existing Structures, SM Archives, 8, pp. 99—118, 1983.
134. Itkis U. Control Systems of Variable Structure, John Wiley N. Y. 1976.
135. Jain R. Dynamic Scene Analysis Using Pixel — Based Processes, Computer, 14, No. 8, pp. 12—18, 1981.
136. Jain R. Segmentation of Frame Sequences Obtained by a Moving Observer, Report GMR—4247, General Motors Research Laboratories, Warren, Mich. 1983.
137. Jarvis R. A. A Perspective on Range Finding Techniques for Computer Vision IEEE Trans. Pattern Anal. Machine Intell., PAMI—5, No. 2, pp. 122—139, 1983.
138. Jarvis R. A. A Laser Time — of — Flight Range Scanner for Robotic Vision, IEEE Trans. Pattern Anal. Machine Intell., PAMI—5, No. 5, pp. 505—512, 1983.
139. Johnston A. R. Proximity Sensor Technology for Manipulator End Effectors, Mechanism and Machine Theory, 12, Nol 1, pp. 95—108, 1977.
140. Kahn M. E., Roth B. The Near — Minimum — Time Control of Open-Loop Articulated Kinematic Chains, Trans. ASME, J. Dynamic Systems, Measurement and Control 93., pp. 164—172, 1971.

141. Kane T R., Levinson D. A. The Use of Kane's Dynamical Equations in Robotics,", Intl. J. Robotics. Res , 2, No. 3, pp. 3—21, 1983.
142. Katushi I., Horn B. K. P. Numerical Shape from Shading and Occluding Boundaries", Artificial Intelligence, 17, pp. 141—184, 1981.
143. Ketcham D. J. Real — Time Image Enhancement Techniques. Proc. Soc. Photo — Optical Instrum. Engr., 74, pp. 120—125, 1976.
144 Khatib O. Commande Dynamique dans Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles", Docteur Ingenieur Thesis, L'Ecole Nationale Superieure de l'Aeronautique et de l'Espace, Toulouse, France, 1980.
145. Kirk D. E. Optimal Control Theory, An Introduction, Prentice-Hall, Englewood Cliffs, N. J. 1970.
146. Klinger A. Patterns and Search Statistics", in Optimizing Methods in Statistics (J. S. Rustagi, ed.), Academic Press, N. Y., pp. 303—339, 1972.
147. Klinger A. Experiments in Picture Representation Using Regular Decomposition", Comput. Graphics Image Proc., 5, pp. 68—105, 1976.
148. Kohler R. J., Howell H. K. Photographic Image Enhancement by Superposition of Multiple Images, Phot. Sci. Engr., 7, No. 4, pp. 241—245, 1963.
149. Kohli D., Soni A. H. Kinematic Analysis of Spatial Mechanisms via Successive Screw Displacement, J. Engr. for Industry, Trans. ASME, 2, series B pp. 739—747, 1975.
150. Koivo A. J., Guo T. H. Adaptive Linear Controller for Robotic Manipulators. IEEE Trans. Automatic Control. AC-28, No. 1, pp. 162—171, 1983.
151. Landau Y. D. Adaptive Control — The Model Reference Approach, Marcel Dekker, N. Y. 1979.
152. Lee B. H. An Approach to Motion Planning and Motion Control of Two Robots in a Common Workspace, Ph. D. Dissertation, Computer Information and Control Engineering Program, University of Michigan, Ann Arbor, Mich., 1985.
153. Lee C. C. Elimination of Redundant Operations for a Fast Sobel Operator, IEEE Trans. Systems. Man. Cyybern, SMC—13, No. 3, pp. 242—245, 1983.
154. Lee C. S. G. Robot Arm Kinematics, Dynamics, and Control. Compute 15 No. 12, pp. 62—80, 1982.
155. Lee C. S. G. On the Control of Robot Manipulators", Proc. 27th Soc. Photooptical Instrumentation Engineers, 442, San Diego, Calif., pp. 58—83, 1983.
156. Lee C. S. G. Robot Arm Kinematics and Dynamics, in Advances in Automation and Robotics: Theory and Applications (G. N. Saridis, ed.), JAI Press, Conn., pp. 21—63, 1985.
157. Lee C. S. G., Chang P. R. A Maximum Piplined CORDIC Architecture for Robot Inverse Kinematics Computation, Technical Report TR-EE-86-5, School of Electrical Engineering, Purdue University, West Lafayette, Ind., 1986.
158. Lee C. S. G., Chang P. R. Efficient Parallel Algorithm for Robot Inverse Dynamics Computation, IEEE Trans. System, Man, Cybern, SMC—16, No. 4. 1986.
159. Lee C. S. G., Chung M. J. An Adaptive Control Strategy for Mechanical Manipulators, IEEE Trant. Automatic Control, AS-29, No. 9, pp. 837—840, 1984.
160. Lee C. S. G., Chung M. J. Adaptive Perturbation Control with Feedforward Compensation for Robot Manipulators, Simulation, 44, No. 3, pp. 127—136, 1985.
161. Lee C. S. G., Chung M. J., Lee B. H. An Approach of Adaptive Control for Robot Manipulators, J. Robotic Systems, 1, No. 1, pp. 27—57. 1984.
162. Lee C. S. G., Chung M. J., Mudge T. N., On the Control of Mechanical Manipulators, Proc. 6th IFAC Conf. Estimation and Parameter Identification, Washington D. C., pp. 1454—1459, 1982.

163. Lee C. S. G., Gonzalez R. C., Fu K. S. Tutorial on Robotics, 2d ed. IEEE Computer Press, Silver Spring, Md, 1986.
164. Lee C. S. G., Huang D. A Geometric Approach to Deriving Position/Force Trajectory in Fine Motion, Proc. 1985, IEEE Intl. Conf. Robotics and Automation, St. Louis, Mo, pp. 691—697, 1985.
165. Lee C. S. G., Lee B. H. Resolved Motion Adaptive Control for Mechanical Manipulators, Trans. ASME J. Dynamic Systems, Measurement and Control 106, No. 2, pp. 134—142, 1984.
166. Lee C. S. G., Lee B. H., Nigam R. Development of the Generalized d'Alembert Equations of Motion for Mechanical Manipulators, Proc. 2nd Conf. Decision and Control, San Antonio, Tex., pp. 1205—1210, 1983.
167. Lee C. S. G., Mudge T. N., Turney J. L. Hierarchical Control Structure Using Special Purpose Processors for the Control of Robot Arms, Proc. 1982, Pattern Recognition and Image Processing Conf., Las Vegas, Nev., pp. 634—640, 1982.
168. Lee C. S. G., Ziegler M. A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots, IEEE Trans. Aerospace and Electronic Systems, AES-20, No. 6, pp. 695—706, 1984.
169. Lewis R. A. Autonomous Manipulation on a Robot: Summary of Manipulators Software Functions, Technical Memo 33—679, Jet Propulsion Laboratory, Pasadena, Calif., 1974.
170. Lewis R. A., Bejczy A. K. Planning Considerations for a Roving Robot with Arm." Proc. 3rd Int. Joint Conf. Artificial Intelligence, Stanford University, Palo Alto, Calif., 1973.
171. Lieberman L. I., Wesley M. A. AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly", IBM J. Res. Devel., 21, No. 4, pp. 321—333, 1977.
172. Lin C. S., Chang P. R., Luh J. Y. S. Formulation and Optimization of Cubic Polynomial Joint Trejectories for Industrial Robots. IEEE Trans. Automatic Control. AC-28, No. 12, pp. 1066—1073, 1983.
173. Lozano — Perez T. Automatic Planning of Manipulator Transfer Movements, IEEE Trans. Systems. Man. Cybern., SMC—11, Nol 11, pp. 691—698, 1981.
174. Lozano — Perez T. Spatial Planning, A Configuration Space Apace Approach, IEEE Trans. Comput C-32, No. 2, pp. 108—120, 1982.
175. Lozano — Perez T., Robot Programming, Proc. IEEE, 71, No. 7, pp. 821—841, 1983.
176. Lozano — Perez T. Task Planning in Robot Motion: Planning and Control. (M. Brady et al., eds.), MIT Press, Cambridge, Mass., 1983.
177. Lozano — Perez T., Wesley M. A. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstracles, Comm. ACM 22, No. 10, pp. 560—570, 1979.
178. Lu S. Y., Fu K. S. A Syntactic Approach to Texture Analysis, Comput. Graph. Image Proc., 7, No. 3, pp. 303—330, 1978.
179. Luh J. Y. S. An Anatomy of Industrial Robots and their Control, IEEE Trans. Automatic Control. AC-28, No. 2, pp. 133—153, 1983.
180. Luhi J. Y. S. Conventional Controller Design for Industrial Robots — A Tutorial, IEEE Trans. Systems, Man, and Cybern, SMC-13, No. 3, pp. 298, 316, 1983.
181. Luh J. Y. S., Lin C. S. Optimum Path Planning for Mechanical Manipulators. Trans. ASME, J. Dynamic Systems, Measurement and Control. 102, pp. 142—151, 1981.
182. Luh J. Y. S., Lin C. S. Automatic Generation of Dynamic Equations for Mechanical Manipulators. Proc. Joint Automatic Control Conf., Charlottesvile, Va., pp. TA-2D, 1981.
183. Luh J. Y. S., Lin C. S. Scheduling of Parallel Computation for a Computer Controlled Mechanical Manipulator, IEEE Trans. Systems, Man, Cybern, SMC-12, pp. 214—234, 1982.

184. Luh J. Y. S., Lin C. S. Approximate Joint Trajectories for Control of Industrial Rebots Along Cartesian Path, IEEE Trans. Sustems, Man, Cybern., SMC-14 No. 3, pp. 444—450, 1984.
185. Luh J. Y. S., Walker M. W., Paul R. P. On Line — Computational Scheme for Mechanical Manipulators. Trans. ASME, J. Dynamic Systems, Measurements and Control. 120, pp. 69—76. 1980.
186. Luh J. Y. S., Walker M. W., Paul R. P. Resolved — Acceleration Control of Mechanical Manipulators, IEEE Trans. Automatic Control. AC-25, No. 3, pp. 468—474, 1980.
187. Marck V. Algorithms for Complex Tactile Information Processing, Proc. Intl. Joint Conf. Artificial Intelligence, pp. 773—774, 1981.
188. Markiewicz B. R. Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer Controlled Manipulator, Technical Memo 33—601, Jet Propulsion Laboratory, Pasadena, Calif., 1973.
189. Marr. D. Visual Information Processing: The Structure and Creation of Visual Representations, Proc. Inti. Joint. Conf. Artificial Intelligence, Tokyo, Japan, pp. 1108—1126, 1979.
190. Marr D. Vision Freeman, San Francisco, Calif., 1982.
191. Martelli A. Edge Detection Using Heuristic Search Methods, Comput. Graphics Image Proc., 1. pp. 169—182, 1972.
192. Martelli A. An Application of Heuristic Search Methods to Edge and Contour Detection, Comm. ACM. 19, No. 2, pp. 73—83, 1976.
193. Mason M. T. Compliance and Force Control for Computer Controlled Manipulator IEEE Trans. Systems, Man, Cybern, SMC—11 No. 6, pp. 418—432, 1981.
194. McCarthy J. et al. A Computer with Hands, Eyes, and Ears, 1968 Fall Joint Pomputer Conf., AFIPS Proceeddings, pp. 329—338, 1968.
195. McDermott J. Sensors and Transducers EDN 25, No. 6, pp. 122—137, 1980.
196. Meindl J. D., Wise K. D. Special Issue on Solid — State Sensors, Actuators, and Interface Electronics, IEEE Trans. Elect. Devices, 26, pp. 1861—1978, 1979.
197. Merritt R. Industrial Robots: Getting Smarter All The Time, Instruments and Control Systems, 55, No. 7, pp. 32—38, 1982.
198. Milenkovic V., Huang B. Kinematics of Major Robot Linkages, Proc. 13th Intl. Symp. Industrial Robots, Chicago, III, pp. 16—31, 1983.
199. Mujtaba S. M., Goldman R. AL User's Manual, 3d ed., STAN-CS-81 889 CSD, Stanford University, Palo Alto, Calif., 1981.
200. Mujtaba M. S., Goldman R. A., Binford T. The AL Robot Programming Language", Comput. Engr., 2, pp. 77—86, 1982.
201. Mundy J. L. Automatic Visual Inspection", Proc. 1977, Conf. Decision and Control, pp. 705—710, 1977.
202. Murray J. J., Neuman C. P. ARM: An Algebraic Robot Dynamic Modeling Program, Proc. Intl. Conf. Robotics, Atlanta, Ga., pp. 103—113, 1984.
203. Myers W. Industry Begins to Use Visual Pattern Recognition Computer, 13, No. 5, pp. 21—31, 1980.
204. Naccache N. J. Shinghai R. SPTA: A Proposed Algorithm for Thinning Binary Pattersm, IEEE Trans. Systems. Man, Cybern., SMC—14, No. 3, pp. 409—418, 1984.
205. Nagel H. H. Representation of Moving Rigid Objects Based on Visual Observations", Computer, 14, No. 8, pp. 29—30, 1981.
206. Nahim P. J. The Theory of Measurement of a Silhouette Description for Image Processing and Regonition, Pattern Recog., 6, No. 2, pp. 85—95, 1974.
207. Narendra P. M., Fitch R. C. Real-Time Adaptive Contrast Enhancement, IEEE Trans. Pattern Anal, Machine Inteil., PAMI-3, No. 6, p. 655—661, 1981.
208. Nau D. S. Expert Computer Systems, Computer 16, pp. 63—85, 1983.

209. Nevatia R., Binford T. O. Description and Recognition of Curved Objects, Artificial Intelligence, 8, pp. 77—98, 1977.
210. Nevins J. L., Whitney D. E. Computer — Controlled Assembly, Sci. Am., 238, No. 2, pp. 62—74, 1978.
211. Nevins J. L., Whitney D. E., et al. Exploratory Research in Industrial Modular Assembly, NSF Project Reports 1 to 4 C. S. Draper Laboratory, Cambridge, Mass., 1974—1976.
212. Newman W. M., Sproull R. F. Principles of Interactive Computer Graphics, McGraw — Hill, New York, 1979.
213. Neuman C. P., Tourassis V. D. Discrete Dynamic Robot Modelling, IEEE Trans. Systems. Man. Cybern, SMC—15, pp. 193—204, 1985.
214. Neumann C. P., Tourassis V. D. Robot Control: Issues and Insight, Proc. Third Yale Workshop on Applications of Adaptive Systems Theory, Yale University, New Haven, Conn., pp. 179—189, 1983.
215. Nigam R., Lee C. S. G. Multiprocessor — Based Controller for the Control of Mechanical Manipulators, IEEE J. Robotics and Automation, RA-1 No. 4, pp. 173—182, 1985.
216. Nilsson N. J., Problem — Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.
217. Nilsson N. J. Principles of Artificial Intelligence, Tioga Pub. Palo Alto, Calif. 1980.
218. Noble B. Applied Linear Algebra, Prentice Hall, Englewood Cliffs, N. J. 1969.
219. Oldroyd A. MCL: An APT Approach to Robotic Manufacturing SHARE 56, Houston, Tex., March 9—13, 1981.
220. Ohlander R., Price K., Reddy D. R. Picture Segmentation Using a Recursive Region Splitting Methods, Comput. Graphics Image Proc., 8, No. 3, pp. 313—333, 1979.
221. Orin D. E. Pipelined Approach to Inverse Plant Plus Jacobian Control of Robot Manipulators, Proc. Intl. Conf. Robotics, Atlanta, Ga., pp. 169—175, 1984.
222. Oring D. E. Schrader W. W. Efficient Computation of the Jacobian for Robot Manipulators. Intls. J. Robotics Res., 3, No. 4, pp. 66—75, 1984.
223. Orin D. E., McGhee R. B., Vukobratovic M., Hartoch G. Kinematic and Kinetic Analysis of Open — Chain Linkages Utilizing Newton — Euler Methods, Math Biosci., 43, pp. 107—130, 1979.
224. Papoulis A., Probability, Random Varaibles, and Stochastic Processes, McGraw — Hill, 1965.
225. Park W. T. The SRI Robot Programming System (RPS): An Executive Summary SRI International, Menlo Park, Calif. 1981.
226. Paul R. P. Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm. Memo AIM-177; Stanford Artificial Intelligence Laboratory, Palo Alto, Calif., 1972.
227. Paul R. P. WAVE: A Model — Based Language for Manipulator Control Technical Paper MR76-615, Society of Manufacturing Engineers, Dearborn, Mith., 1976; The Industrial Robot, 4, pp. 10—17, 1977.
228. Paul R. P. Manipulator Cartesian Path Contro. IEEE Trans. Systems, Man. Cubern, SMC—9, No. 11, pp. 702—711; 1979.
229. Paul R. P. Robot Manipulator: Mathematics, Programming and Control, MIT Press, Cambridge, Mass. 1981.
230. Paul R. P. Shimano B. Compliance and Control Proc. Joint Automatic Control Conference, Purdue University, West Lafayette Ind., 1976.
231. Paul R. P., Shimano B. E,, Mayer G. Kinematic Control Equations for Simple Manipulators IEEE Trans. Systems. M. Cybern., SMC-11, No. 6, pp. 449—455, 1981.
232. Pavlidis T. Structural Pattern Recognition, Springer-Verlag, 1977.
233. Pavlidis T. Algorithms for Graphics and Image Processing, Computer Science Press., Rockville, Md., 1982.

234. Persoon E., Fu K. S. Shape Descrimination Using Fourier Descriptors, IEEE Trans. Systems. Man, Cybern., SMC-7, No. 2, pp. 170—179, 1977.
235. Pieper D. L. The Kinematics of Manipulators under Computer Control, Artificial Intelligence Project Memo No. 72, Computer Science Department, Stanford University, Palo Alto, Calif., 1968.
236. Pieper D. L., Roth B. The Kinematics of Manipulators under Computer Control. Proc. III Intl. Congr. Theory of Machines and Mechanisms, 2, pp. 159—168, 1969.
237. Pipes L. A. Matrix Methods in Engineering, Prentice-Hall, Englewood Cliffs, N. J., 1963.
238. Popplestone R. J., Ambler A. P., Bellos I. RAPT, A Language for Describing Assembles, Industrial Robot, 5, No. 3, pp. 131—137, 1978.
239. Popplestone R. J. Ambler A. P., Bellos I. An Interpreter for a Language Descri ing Assembles, Artificial Intelligence, 14, No. 1. pp. 79—107, 1980.
240. Raibert M. H., Craig J. J. Hybrid Position/Force Control of Manipulators, Trans. ASME, J. Dynamic Systems, Measurement, and Control. 102, pp. 126—133, 1981.
241. Raibert M. H., Tanner J. E. Design and Implementation of a VLSI Tactile Sensing Computer, Intl. J. Robotics Res., 1. No. 3, pp. 3—18, 1982.
242. Rajala S. A., Riddle A. N., Snyder W. E. Application of the One-Dimensional Fourier Transform for Tracking Moving Objects in Noisy Environments, Comput. Vision, Graphics, Image Proc., 2, pp. 280—293, 1983.
243. Ramer U. Extraction of Line Structures from Photographs of Curved Object, Comput. Graphics Image Proc., 4, pp. 81—103, 1975.
244. Reddy D. R., Hon R. W. Computer Architectures for Vision, Computer Vision and Sensor-Based Robots (G. G. Dodd and L. Rossol, eds.), Plenum N. Y., 1979.
245. Requicha A. Representation for Rigif Solids: Theory, Methods, and Systems, Computing Surveys, 12, No. 4, pp. 437—464, 1980.
246. Requicha A. Towards a Theory of Geometric Telerancing, Intl. J. Robotics Res., 2, No. 4, pp. 45—60, 1983.
247. Requicha A., Voelcker H. B. Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Comput, Graphics and Applications, 2, No. 2, pp. 9—24, 1982.
248. Rich A. Artificial Intelligence, McGraw-Hill, 1983.
249. Roberts L. G. Machine Perception of three-Dimensional Solids, Optical and Electro-Optical Information Processing, (J. P. Tippett et al., eds.), MIT Press, Cambridge, Mass, 1965.
250. Rocher F., Keissling A. Methods for Analyzing Three-Dimensional Scenes, Proc. 4th Intl. Joint Conf. Artificial Intelligence, pp. 669—673, 1975.
251. Rosen C. A., Nitzan D. Use of Sensors in Programmable Automation, Computer, 10. No. 12, pp. 12—23, 1977.
252. Rosenfeld A., Kak A. C. Digital Picture Processing, 2d ed., Academic Press, N. Y. 1982.
253. Roth B., Rastegar J., Scheinman V. On the Design of Computer Controlled Manipulators, 1st CISM-IFTMM Symp. Theory and Practice of Robots and Manipulators, pp. 93—113, 1973.
254. Sacerdoti E. D. A Structure for Plans and Behavior, Elsevier, N. Y. 1977.
255. Sadjadi F. A., Hall E. L. Three-Dimensional Moment Invariants, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-2, No. 2, pp. 127—136, 1980.
256. Salari E., Siy P. The Ridge-Seeking Method for Obtaining the Skeleton of Digital Images," IEEE Trans. Systems Man, Cybern., SMC—14, No. 3, pp. 524—528, 1984.
257. Saridis G. N. Intelligent Robotic Control. IEEE Trans. Automatic Control. AC-28, No. 5, pp. 547—557, 1983.
258. Saridis G.N., Lee C. S. G. An Approximation Theory of Optimal Control for Trainable Manipulators, IEEE Trans. Systems. Man, Cybern, SMC—9. No. 3, pp. 152—159, 1979.

259. Saridis G. N., Lobbia R. N. Parameter Identification and Control of Linear Discrete-Time Systems, IEEE Trans. Automatic Control. AC-17, No. 1, pp. 52—60, 1972.
260. Saridis G. N., Stephanou H. E. A Hierarchical Approach to the Control of a Prosthetic Arm IEEE Trans. Systems. Man. Cybern., SMC—7, No. 6, pp. 407—420, 1977.
261. Scheinman V. D. Design of a Computer Manipulator, Artificial Intelligence Laboratory Memo AIM-92, Stanford University, Palo Alto, Calif., 1969.
262. Shani U. A 3-D Model — Driven System for the Recognition of Abdominal Anatomy from CT Scans, Proc. 5th Intl. Joint Conf. Pattern Recog., pp. 585—591, 1980.
263. Shimano B. VAL: A Versatile Robot Programming and Control System, Proc. 3rd Intl. Computer Software Applications Conf., Chicago, III, pp. 878—883, 1979.
264. Shimano B. E., Roth B. On Force Sensing Information and its Use in Controlling Manipulators, Proc. 9th Intl. Symp. on Industrial Robots, Washington, D. C., pp. 119—126, 1979.
265. Shirai Y. Three-Dimensional Computer Vision", in Computer Vision and Sensor-Based Robots (G. G. Dodd and L. Rossol, eds.), Plenum N. Y., 1979.
266. Siklossy L. Modelled Exploration by Robot, Tech. Rept. L, Computer Science Department, University of Texas, Austin, Tex., 1972.
267. Siklossy L., Dreussi J. An Efficient Robot Plannor which Generates its Own Procedures, Proc. 3rd Intl. Joint Conf. Artificial Intelligence, pp. 423, 430, 1973.
268 Silver W. M. On the Equivalence of the Lagrangian and Newton—Euler Dynamics for Manipulators, Intl. J. Robotics, Res., 1, No. 2, pp. 60—70, 1982.
269. Sklansky J., Chazin R. L., Hansen B. J. Minimum-Perimeter Polygons of Digitized Silhouettes, IEEE Trans. Comput., C-21, No. 3, pp. 260—268, 1972.
270. Snyder W. E. Industrial Robots: Computer Interfacing and Control. Prentice — Hall, Englewood Cliffs, N. Y., 1985.
271. Spencer J. D. Versatile Hall-Effect Devices Handle Movement-Control Tasks, EDN 25, No. 3, pp. 151—156, 1980.
272. Stepanenko Y., Vukobratovic M. Dynamics of Articulated Open-Chain Active Mechanisms, Math. Biosci., 28, pp. 137—170, 1976.
273. Suh C. H., Radcliffe C. W. Kinematics and Mechanisms Design, John Wiley, N. Y. 1978.
274. Sussman G. J., Winograd T., Charniak E. Micro — Planner Reference Manual," AI Memo 203, MIT Press, Cambridge, Mass., 1970.
275. Symon K. R., Mechanics, Addison — Wesley, Reading, Mass., 1971.
276. Sze T. W., Yang Y. H. A Simple Contour Matching Algorithms, IEEE Trans. Pattern Anal. Math. Intell., PAMI-3, No. 6, pp. 676—678, 1981.
277. Takase K., Paul R. P., Berg E. J. A Structural Approach to Robot Programming and Teaching, IEEE Trans. Systems, Man, Cybern., SMC—11, No. 4, pp. 274—289, 1981.
278. Takegaki M., Arimoto S. A New Feedback Method for Dynamic Control of Manipulators, Trans. ASME, J. Dynamic Systems, Measurement and Control, 102, pp. 119—125, 1981.
279. Tangwongsan S., Fu K. S. An Application of Learning to Robotic Planning. Intl. J. Computer and Information Sciences, 8, No. 4, pp. 303—333, 1979.
280. Tarn T. J. et al. Nonlinear Feedback in Robot Arm Control, Proc. 1984, Conf. Decision ond Control. Las Vegas, Nev., pp. 736—751, 1984.
281. Taylor R. H. The Synthesis of Manipulator Control Programs from

Task — Level Specifications, Report AIM-282, Artificial Intelligence Laboratory, Stanford University, Palo Alto, Calif., 1976.
282. Taylor R. H. Planning and Execution of Straight Line Manipulator Trejectories, IBM J. Res. Devel., 23, No. 4, pp. 424—436, 1979.
283. Taylor R. H., Summers P. D., Meyer J. M. AML: A Manufacturing Language", Intl. J. Robotics Res., 1, No. 3, pp. 19—41, 1983.
284. Thompson W. B., Barnard S. T. Lower — Level Estimation and Interpretation of Visual Motion, Computer 14, No. 8, pp. 20—28, 1981.
285. Thrall R. M., Tornheim L. Vector Spaces and Matrices John Wiley N. Y. 1963.
286. Tomita F., Shirai Y., Tsuji S. Description of Texture by a Structural Analysis, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-4, No. 2, pp. 183—191, 1982.
287. Tomovic R., Boni G. An Adaptive Artificial Hand., IRE Trans. Automatic Control. AC-7, 3, pp. 3—10, 1962.
288. Toriwaki J. I., Kato N., Fukumura T. Parallel Local Operations for a New Distance Transformation of a Line Pattern and Their Applications, IEEE Trans. Systems, Man, Cybern, SMC—9, No. 10, pp. 628—643, 1979.
289. Tou J. T. (ed.) Computer — Based Automation, Plenum, 1985.
290. Tou J. T., Gonzalez R. C. Pattern Recognition Principles, Addison—Wesley, Reading, Mass., 1974.
291. Turney J. L., Mudge T. N., Lee C. S. G. Eguivalence of Two Formulations for Robot Arm Dynamics. SEL Report 142, ECE Department, University of Michigan, Ann Arbor, Mich., 1980.
292. Turney J. L., Mudge T N., Lee C. S. G. Connection Between Formulations of Robot Arm Dynamics with Applications to Simulation and Control. CRIM Technical Report No. RSD—TR-4-82, the University of Michigan, Ann Arbor, Mich, 1982.
293. Uicker J. J. On the Dynamic Analysis of Spatial Linkages using 4×4 Matrices, Ph. D. disseration, Northwestern University. Evanston, III, 1965.
294. Uicker J. J., Jr., Denavit J., Hartenberg R. S. An Iterative Methods for the Displacement Analysis of Spatial Mechanisms, Trans. ASME, J. Appl. Mech., 31, Series E, pp. 309—314, 1964.
295. Unger S. H. Pattern Detection and Recognition", Proc. IRE, 47. No. 10, pp. 1737—1752, 1959.
296. User's Guide to VAL Version 11, 2d ed., Unimation, Inc., Danbury, Conn. 1979.
297. Utkin V. I. Variable Structure Systems with Sliding Mode: A Survey, IEEE Trans. Automatic Control, AS-22, pp. 212—222, 1977.
298. Vukobratovic M., Stokic D. Contribution to the Decoupled Control of Large — Scale Mechanical Systems, Automatica, 16, pp. 16—21, 1980.
299. Walker M. W., Orin D. E. Efficient Dynamic Computer Simulation of Robotic Mechanisms, Trans. ASME, J. Systems, Measurement and Control. 104, pp. 205—211, 1982.
300. Wallace T. P., Mitchell O. R. Analysis of Three — Dimensional Movements Using Fourier Descriptors, IEEE Trans. Pattern Anal. Machine Intell., PAMI-2, No. 6, pp. 583—588, 1980.
301. Waltz D. I. Generating Semantic Descriptions from Drawings, of Scenes with Shadows, Ph. D. Dissertation, Artificial Intelligence Lab., MIT, Cambridge, Mass., 1972.
302. Waltz D. I. Automata Theoretical Approach to Visual Information Processing, Applied Computation Theory (R. T. Yeh, ed.), Prentice-Hall, Englewood Cliffs, N. J., 1976.
303. Webb J. A., Aggarwal J. K. Visually Interpreting the Motion of Objects in Space Computer 14, No. 8, pp. 40—49, 1981.
304. Weiss S. M,. Kukowski C. A. A Practical Guide to Designing Expert Systems, Rowman and Allanheld, New Jersey., 1984.
305. Weska J. S. A Survey of Threshold Selection Techniques, Comput. Graphics Image Proc., 7, pp. 259—265, 1978.
306. Wesley M. A. A Geometric Modeling System for Automated Mechanical Assembly IBM J. Res. Devel., 24, No. 1, pp. 64—74, 1980.
307. White J. M., Rohrer G. D. Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction, IBM J. Res. Devel., 27, No. 4, pp. 400—411, 1983.
308. Whitney D. E. Resolved Motion Rate Control of Manipulators and Human Prostheses, IEEE Trans. Man — Machine Systems, MMS-10, No. 2, pp. 47—53, 1969.
309. Whitney D. E. State Space Models of Remote Manipulation Tasks. Proc. Intl. Joint Conf. Artificial Intelligence, Washington, D. C., pp. 495—598, 1969.
310. Whitney D. E. The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators, Trans. ASME, J. Dynamic Systems, Measurement and Control. 122, pp. 303—309, 1972.
311. Will P., Grossman D. An experimental System for Computer Controlled Mechanical Assembly, IEEE Trans. Comput., C-24, No. 9, pp. 879—888, 1975.
312. Winston P. H. Artificial Intelligence, 2d ed., Addison — Wesley, Reading, Mass., 1984.
313. Wise K. D., (ed.) Special Issue on Solid — State Sensors, Actuators, and Interface Electronics, IEEE Trans. Elect. Devices, 29, pp. 42—48, 1982.
314. Wolfe G. J. Mannos J. L. Fast Median Filter Implementation, Proc. Soc. Photo-Optical Inst. Engr., 207, pp. 154—160, 1979.
315. Woods R. E., Gonzalez R. C. Real-Time Digital Image Enhancement, Proc. IEEE 69, No. 5, pp. 643—654, 1981.
316. Wu C. H., Paul R. P. Resolved Motion Force Control of Robot Manipulator IEEE Trans. Systems, Man, Cybern. SMC—12 No. 3, pp. 266—275, 1982.
317. Yakimovsky Y., Cunningham R. A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras, Comput, Graphics Image Proc., 7, pp. 195—210, 1979.
318. Yang A. T. Displacement Analysis of Spatial Five — link Mechanism Using 3×3 Matrices with Dual — Number Elements, Trans. ASME J. Engr. Industry, 91, No. 1. Series V, pp. 152—157, 1969.
319. Yang A. T., Freudenstein R. Application of Dual Number Quaternian Algebra to the Analysis of Spatial Mechanism", Trans. ASME, J. Appl. Mech., 31, series E, pp. 152—157, 1964.
320. Young K. K. D. Controller Design for a Manipulator Using Theory Variable Structure Systems, IEEE Trans. Systems, Man, Cybern., SMC—8, No. 2, pp. 101—109, 1978.
321. Yuan M. S. C., Freudensten R. Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates, Trans. ASME, J. Engr. Industry, 93, No. 1, pp. 61—73, 1971.
322. Zahn C. T., Roskies R. Z. Fourier Descriptors for Plane Closed Curves, IEEE Trans. Comput., C-21, No. 3, pp. 269—281, 1972.
323. Zucker S. W. Region Growing: Childhood and Adolescence, Comput. Graphics Image Proc., 5, pp. 382—399, 1976.
324. Zucker S. W., Hummel R. A. A Three — Dimensional Edge Operator, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-3, No. 3, pp. 324—331, 1981.
325. Automation Systems Assembly Robot Operator's Manual, P50VE025, General Electric Co., Bridgeport, Conn., February 1982.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ