# ТЕХНИЧЕСКОЕ ЗРЕНИЕ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Для более полного взаимодействия человека с компьютером, как инструментом своей деятельности, ясно, что компьютер должен обладать способностью видеть. Для автономного робота зрение совершенно необходимо. Процесс преобразования в числовую форму сигнала телевизионной камеры не относится к области ИИ, а вот процесс интерпретации этих сигналов как раз да. Здесь будут рассмотрены несколько способов интерпретации этих сигналов компьютером.

Предполагается, что видео сигнал уже преобразован в цифровую форму и необходимо распознать образы, содержащиеся в этом сигнале. Под видео сигналом понимается некоторая картинка, кадр, на котором надо что-либо узнать.

Первоначально рассмотрим некоторые ключевые вопросы распознавания образов и обработки изображений.

## 1. Два подхода к распознаванию

Существует два подхода, на которых основывается работа системы технического зрения. В соответствии с первым подходом изображение упрощается до линий, которые формируют контур каждого объекта. Этот метод использует различные фильтры, которые удаляют лишнюю информацию с изображения, усиливают контраст, чтобы сделать все части изображения либо черными, либо белыми. Иногда это называют бинарным (двоичным) изображением, потому что нет серых областей - каждая точка в изображении является или черной, или белой. Процедура создания бинарного изображения не относится к проблеме интерпретации исходного изображения. Это выполняет препроцессор, не связанный с процессом интерпретации изображения. Он может быть выполнен в виде аналого-цифрового устройства, осуществляющего последовательно несколько раз фильтрацию и повышающего контраст изображения. Преимущество двоичных изображений состоит в том, что они четко выделяют контуры объектов, которые компьютер может легко распознавать, используя довольно простые алгоритмы. Другими словами, компьютеру ясно, где каждый объект начинается и где заканчивается. Высоко-контрастное изображение наиболее распространено в управляемых средах, в которых заранее известно, что будет опознаваться только несколько выбранных объектов. Бинарные изображения большей частью используют двумерные системы обработки изображений.

Второй подход, который применяется для разработки систем технического зрения, позволяет сделать восприятие образов компьютером более сходным с восприятием образов человеком. В соответствии с этим подходом компьютер анализирует информацию о яркости частей изображения. Это позволяет получать две важные характеристики изображения, которые невозможно получить при первом подходе (т.е. используя высоко-контрастное изображение) - это информация о поверхности (степень ее гладкости) и тени. Теперь компьютер может работать с трехмерной информацией об изображении и разрешать конфликты, когда один объект частично блокирует другой. Этот тип обработки изображения обычно используется в трехмерных системах технического зрения.

Замечание относительно цветных изображений.

Фактически системы технического зрения обычно используют черно-белые изображения вместо цветных по двум причинам: во-первых, цвет, в общем случае, не является необходимым, и, во-вторых, добавление информации о цвете предъявляет еще больше требований как к компьютеру, так и к программному обеспечению, которое обрабатывает изображение. Поэтому в учебных целях будем предполагать, что все изображения черно-белые.

## 2. Двумерные системы

Двумерные системы технического зрения требуют строго контролируемую и ограниченную среду функционирования, потому что они обрабатывают изображения так, как если бы эти образы были плоскими. Иногда двумерные системы называют системами обработки плоских образов. Двумерные системы вполне подходят к таким средам как линии автоматизированной сборки, где важным является ориентация, местоположение или распознавание определенных двумерных форм.

Заметим, что сами же реальные объекты не являются плоскими, они обычно трехмерные. Единственное требование к системе состоит в том, что трехмерное изображение может быть сведено к двумерному без потери индивидуальности, т.е. каждый трехмерный объект должен иметь свою двумерную проекцию, отличную от других распознаваемых трехмерных объектов. Например, представим себе фабрику, которая производит только четыре типа блоков: квадраты, прямоугольники, треугольники и цилиндры. По мере изготовления блоки помещаются на конвейерную ленту для автоматической сортировки. Телевизионная камера установлена над конвейером. Она воспринимает двумерное

изображение каждого блока и передает в компьютер. Компьютер должен определить, к какому типу относится блок, и направить его в соответствующую корзину. Эта ситуация показана рисунке 1.

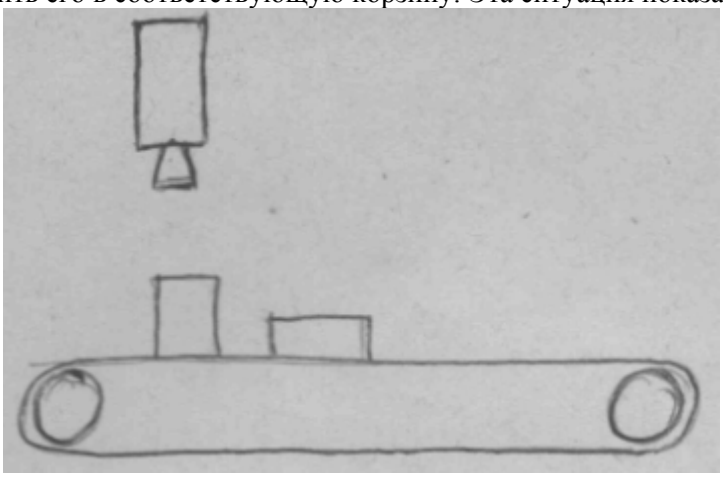


Рис.1. Сортировочная линия

Как только блок проходит под камерой, она воспринимает вид объекта сверху и затем происходит преобразование трехмерной поверхности в двухмерную. Таким образом, компьютер видит изображение такое, как показано на рисунке 2. Хотя одна размерность полностью отброшена, компьютер все же имеет достаточно информации для правильного распознавания каждого блока.

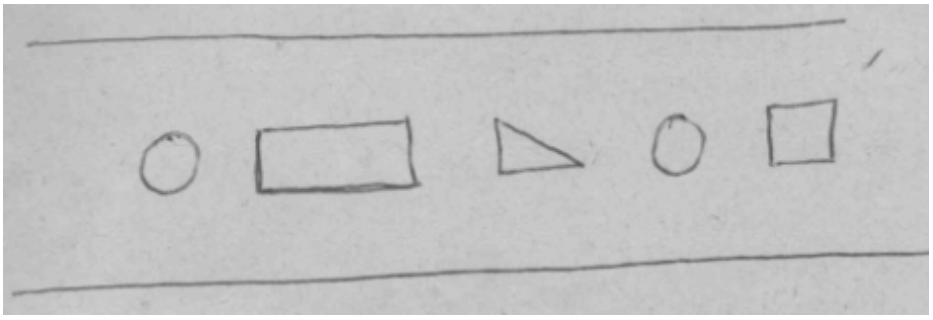
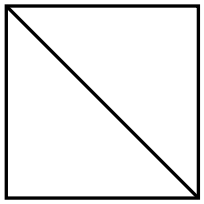


Рис.2. Двумерные образы, которые «видит» компьютер

Для успеха простой двумерной системы технического зрения необходимо, чтобы наблюдаемые объекты четко подавались на ленту и не появлялись непредусмотренные формы. Например, если подается на сортировку пирамида с квадратной основой, то ее двумерное изображение появится как квадрат, и компьютер поместит ее в лоток квадратных блоков. Если необходимо распознавать и пирамиду, то необходимо изменить позицию камеры, или добавить вторую двух-размерную систему, чтобы получить вид с боку.

Общая проблема двумерных систем состоит в том, что у них могут возникнуть трудности с распознаванием объекта, когда тот частично блокирован или покрыт другим объектом. Так как камера дает компьютеру информацию о двумерном изображении, то эта информация может быть противоречива или ошибочна; не всегда возможно правильно разрешить такой конфликт. Представим, что треугольный блок был случайно помещен сверху квадратного блока. Компьютер видел бы следующее:



Компьютер не может определить, состоит ли изображение из двух треугольников, помещенных рядом друг с другом, которые формируют квадратную форму, или это изображение - треугольник на верхней части квадрата. Такие проблемы требуют работать с трехмерными образами.

## 3. Трехмерные системы

По существу, трехмерные системы технического зрения позволяют разрешать все конфликты, которые появляются в двумерных системах, когда объекты, например, накладываются друг на друга, или один объект находится сверху другого. Трехмерные системы также позволяют работать с топографической информацией, например, когда компьютер генерирует геофизические карты по наблюдениям со спутника.

Трехмерная обработка изображения занимает важнейшее место в теории ИИ, потому что существуют проблемы, которые должны быть преодолены.

На самом деле системе технического зрения достаточно одной камеры для многих приложений, требующих трехмерной информации. Чтобы понять почему, закройте один глаз на мгновение и посмотрите вокруг. Вы можете также легко распознавать объекты вокруг Вас! Причина этого проста для объяснения, но трудна для выполнения на компьютере. Человек может видеть и одним глазом, потому что наша система зрения получает намного больше информации, чем только границы объектов. Она имеет информацию относительно цвета, оттенков, яркости и расстояния. Мы все получаем трехмерное представление мира, используя один глаз, еще и потому, что человек может полагаться на другую информацию (по крайней мере, частично) чтобы восполнить потерю бинокулярного видения. Далее предполагаем, что в системе используется только одна камера.

Трехмерная система более сложная, чем двумерная. Трехмерное изображение содержат намного больше информации. В высоко-контрастном двоичном изображении, компьютер может сохранять каждый пиксель в одном бите, потому что он является или черным или белым. (Пиксель - одна дискретная отметка видео изображения, точка экрана). Однако, чтобы получить трехмерное восприятие, требуется информация о яркости каждого пикселя. Пусть различается 256 оттенков серого цвета, тогда каждый пиксель требует 1 байт для хранения. Это в восемь раз больше, чем нужно для хранения черно-белого высоко-контрастного изображения. А для программы требуется существенно больше времени, чтобы проанализировать трехмерное изображение, в отличие от изображения в высоко-контрастном режиме.

Более важная проблема возникает при попытке заставить компьютер использовать всю информацию, которую содержит изображение. Когда человек смотрит, он не задумывается о том, как именно он видит. Наши глаза незначительно отличаются от телевизионной камеры. Следовательно, мозг должен произвести значительное количество работы, чтобы интерпретировать все изображения, которые мы видим каждый день. З**адача ИИ** состоит в том, чтобы воссоздать в компьютере способ, которым человек обрабатывает изображения. Чтобы понять трудность интерпретации образов реального мира, рассмотрим следующие задачи и способы их решения.

### 3.1. Установление направления изменения уровня поверхности

Фотография горного хребта, выполненная со спутника, напоминает эскиз, приведенный в Рис. 3. Если использовать компьютер для анализа этой фотографии, то, как бы компьютер определил, что это фотография горной цепи, а не большого ущелья? Чтобы интерпретировать изображение гор правильно, компьютер должен знать направление изменения уровней поверхностей. Определять направление поверхности можно по способу отражения света. Рис. 4 показывает вид сбоку горы и ущелья, когда солнце находится прямо сверху. Свет, падающий на гору, отражается от ее склона и уходит в сторону. А свет, падающий в ущелье, сначала отражается на противоположный склон и затем возвращается обратно в небо. Программа, анализирующая изображение, может использовать относительную яркость поверхностей, чтобы определить, рассматривает ли компьютер горный хребет или ущелье. Этот анализ - довольно сложная задача даже с простым изображением.

Рис. 3. Эскиз вида гор со спутника

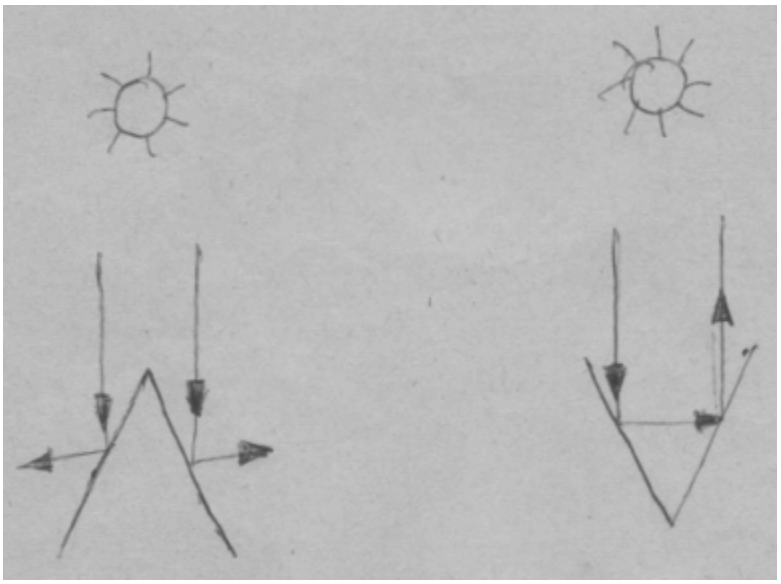


Рис. 4. Отражение света на горе и в долине

### 3.2. Определение строения поверхности

Человек, в общем, может определять, является ли объект гладким или шершавым, рассматривая его. Например, пушистый клубок пряжи выглядит мягким, а стеклянный или мраморный – твердым даже на расстоянии. Вид хорошо отполированного металла или стекла отличается от естественной древесины. Таким образом, Вы можете определять строение поверхности, рассматривая ее. Ключ к различию гладкой поверхности от шероховатой снова в способе отражения света.

Как показано на рисунке 5, гладкая поверхность, которая является характерной для жесткого объекта, отражает падающий свет во всех точках в одном направлении, а шероховатая поверхность, которая является характерной для мягкого объекта, свет рассеивает. Следовательно, относительная яркость мягкого объекта меньше, чем жесткого. Компьютер может просто использовать относительную яркость каждого объекта, чтобы определить, является ли поверхность его гладкой или шероховатой. Этот тип обработки изображения обычно используется на сборочных линиях автомобилей, где такие свойства, как сварочный шов и расцветка, проверяются на однородность.

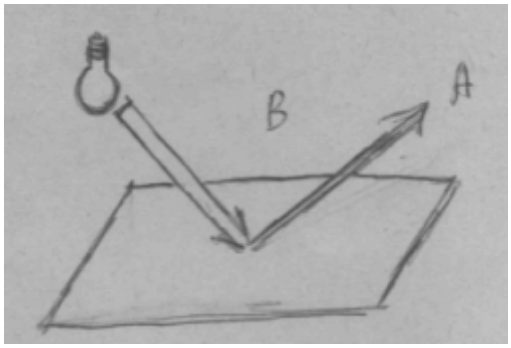



Рис. 5. Отражение света от гладких и шероховатых поверхностей

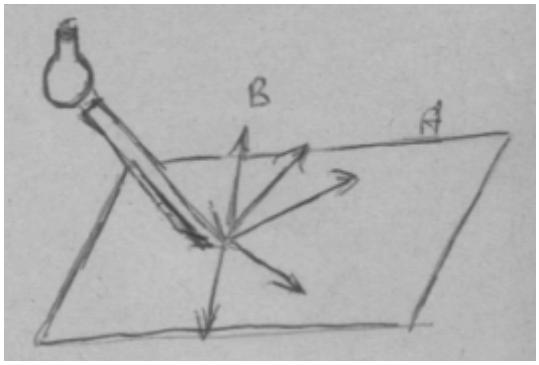



Рис. 6. Объекты с гладкой и шероховатой поверхностью, наблюдаемые из двух точек: A и B.

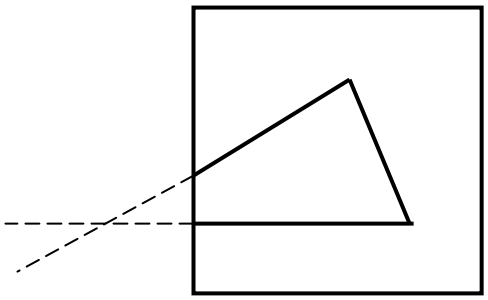
Однако, чтобы компьютер отличил шероховатую поверхность от гладкой в реальном мире, требуется больше, чем просто сравнение относительных яркостей объектов. Причина в том, что на относительную яркость объекта также воздействует цвет и рефлексивные качества материалов объекта. Следовательно, определение степени шероховатости требует двух или большего количества изображений объектов, освещенных с различных точек. Рассмотрим рисунок 6, когда на объект смотрят из точки A. Объект с гладкой поверхностью выглядит ярким, потому что почти весь свет отражается на смотрящего. Объект с шероховатой поверхностью отражает только часть света на смотрящего. Однако, когда на объект смотреть из точки B, то объект с гладкой поверхностью отражает только малое количество света, в то время как объект с шероховатой поверхностью отражает почти то же количество света, что и в точку A. Следовательно, компьютер может различать степень шероховатости поверхности, сравнивая изменение яркости. Большое изменение указывает на гладкую поверхность, а малое - шероховатую.

## 4. Общие проблемы распознавания

### 4.1. Перекрывающиеся объекты

Одна из самых сложных проблем, с которыми приходится сталкиваться при попытке создать систему технического зрения - это распознавание накладывающихся друг на друга объектов. Трудность не в том, что компьютер не может сообщить, что один объект находится перед другим; здесь тени и различия в оттенках дают достаточные сведения. Главная трудность состоит в программировании компьютера распознавать отдельные объекты, видя их только частично. Например, если заложить в компьютер, что треугольник имеет три стороны и три вершины, и если компьютер видит треугольник, одна вершина которого затенена другим объектом (или, возможно, часть его находится за полем зрения камеры), то как компьютер узнает, что это треугольник, когда третья вершина отсутствует?

Существует много подходов к решению этой проблемы, но наиболее интересное решение и самое близкое к тому, как это делает человек, называется управляемой галлюцинацией. В этом методе компьютер, опираясь на начальную информацию и постулаты, проверяет, видит ли он треугольник, для чего используются некоторые средства, например, типа вычисление того, пересекутся ли две линии где-нибудь в затененной области. Это сложный процесс для реализации, так как это программирование находчивости.



### 4.2. Распознавание классов объектов

Другая трудная задача, возникает при программировании компьютера, распознающего классы объектов, то есть распознавания того, что эта яблоня есть дерево, или того, что это конкретное здание есть дом. Гораздо проще заставить компьютер распознать какой-то отдельный объект как объект, чем распознать объект как экземпляр класса. Причина этого в том, что отдельным объектам можно приписывать строгий набор параметров и ограничений, которым должно соответствовать изображение, но при распознавании объекта как экземпляра класса необходимо хранить описание класса совершенно общее и позволяющее охватывать все небольшие различия между экземплярами данного класса.

### 4.3. Оптические обманы

Многие оптические обманы, которые вводят в заблуждение людей, абсолютно не воздействуют на компьютерный анализ того же самого изображения. Например, линии A и B на рисунке 7 одной и той же длины, но, A кажется более длинной. Однако, компьютер не сделал бы эту ошибку. Но, верна и противоположная ситуация: компьютер может быть введен в заблуждение изображениями, которые люди могут правильно интерпретировать. Например, если смотреть на длинную прямую дорогу, то кажется, что она сужается и исчезает, превратившись в точку: то есть объект выглядит тем меньше, чем дальше он находится. Мы привыкли к этому эффекту и мало об этом думаем. Однако, в несовершенных системах технического зрения, компьютер, вероятно, будет думать, что отдаленные объекты просто маленькие.

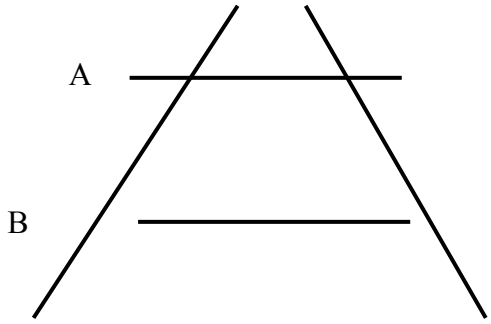



Рис. 7. Простой оптический обман

Таким образом, для компьютера и человека наборы оптических обманов различны. Более того, многие свойства, которые мы считаем само собой разумеющимися, должны явно программироваться для того, чтобы компьютер смог правильно интерпретировать изображение.

## 5. Двухмерное распознавание образа (к лабор. работе)

Из-за сложности трехмерных изображений и методов, которые позволяют распознать их, здесь ограничимся двухмерными высоко-контрастными плоскими изображениями. Но некоторые из рассматриваемых здесь методов можно применять к распознаванию и трехмерных образов.

Существует множество способов, которыми компьютер может распознавать объекты. Причем одни из них в определенных ситуациях работают лучше, чем другие. Некоторые методы работают только со специфическими объектами, в то время как другие могут работать с объектами более общего класса. Следовательно, трудно разработать объективный набор критериев, который определял бы, является ли один подход лучше другого. Однако, ответы на следующие вопросы помогут понять, в какой ситуации можно применять конкретный метод распознавания и каковы его ограничения.

1. Позволяет ли метод правильно распознавать объекты, перекрываемые другими объектами или помещенные один сверху другого?
2. Влияет ли ориентация объекта на результат его распознавания?
3. Влияют ли конкретные размеры объекта на результат его распознавания?
4. Насколько эффективен метод?
5. Может ли метод заблуждаться и допускает ли ошибки?

В русле этих вопросов рассмотрим три метода распознавания изображений.

## 5.1. Ситуация

Необходимо разработать программы распознавания образов, которые правильно идентифицируют треугольники и квадраты. Сначала, будет распознаваться только один тип треугольника - равнобедренный. Затем, добавляется правильный треугольник. Эта задача может казаться слишком простой, но она иллюстрирует многие трудности создания программ распознавания образов.

## 5.2. Моделирование видеообраза

Для моделирования визуального изображения используется видеопамять дисплея компьютера.

В IBM PC и аналогах каждой символьной позиции на экране соответствует байт в зарезервированной части оперативной памяти. Все что попадает в эту область памяти - отображается на экране. Имеет место также и обратная ситуация: все что отображается на экране - попадает в эту часть памяти.

Чтобы моделировать изображение, получаемое от системы технического зрения, сначала очищается экран, а затем отображаются контуры простых объектов. Это есть высоко-контрастное, плоское изображение. Затем программа просматривает видеопамять и пытается идентифицировать объекты. Программы не знают, что именно помещено на экран. Видеоизображение моделируется в текстовом режиме, то есть используется 24x80 позиций (байтовых) экрана для создания изображения. Каждый объект на экране изображается звездочками и пробелами. Если позиция содержит звездочку, то компьютер принимает, что это часть контура; если позиция содержит пробел, то - к контуру она не относится.

Программа просматривает изображение, используя BIOS, прерывание 10H, функцию 8, которая возвращает символ, находящийся в текущей позиции. Использование изображений в текстовом режиме позволяет пользователям без использования графических возможностей компьютера работать с представленными здесь программами распознавания образов.

Все рассматриваемые программы распознавания образов осуществляют поиск точек, составляющих контур. Поскольку программа не знает, в каком месте экрана может находиться объект, то необходимо использовать функцию find_point() для поиска позиции, содержащей звездочку. Каждый раз как вызывается find_point(), она начинает искать звездочки, стартуя с координаты startx, starty, и продвигается слева направо и верху вниз. Когда она находит звездочку, то возвращает ее позицию x и y.

## 5.3. Программа, распознающая по углу

В сильно управляемой среде, правильно идентифицировать треугольник или квадрат возможно, измеряя угол вершины. Ограничившись только одним типом треугольников, равнобедренным, становится просто различать треугольник и квадрат: необходимо только проверить две точки в любой вершине. Если точки - под прямым углом, то объект должен быть квадратом. Если точки - под острым углом, то объект - треугольник.

Один из самых простых способов распознавания по углам состоит в проверке местоположения точек смежных с вершиной. Например, в любой вершине треугольника будет смежная звездочка на диагонали. Аналогично идентифицируется квадрат, только звездочки должны быть под прямым углом.

Функции istriangle() и issquare() проверяют, соответственно, является ли изображение треугольником или квадратом, когда уже известно положение фигуры на экране.

Для идентификации треугольника программа вызывает istriangle() каждый раз как find_point() возвращает позицию звездочки. Если треугольник обнаружен, то выполнение заканчивается; если программа не находит треугольник, она делает попытку в другой позиции. В конечном счете, треугольник обнаруживается.

## 5.5. Программа распознавания по ключевым точкам

Поскольку прямоугольный треугольник может быть спутан программой распознавания по углу, то в этом разделе добавим прямоугольный треугольник к равнобедренному треугольнику и квадрату, и рассмотрим программу, идентифицирующую все три объекта.

Данный подход, при котором программа может идентифицировать все объекты, требует, чтобы всегда они имели тот же самый размер. Тогда исследуется только несколько ключевых точек. Ключевые точки выбираются таким образом, что каждый их набор строго определяет только один объект. На рисунке 8 показаны три объекта, которые должна распознавать программа, а также их ключевые точки.

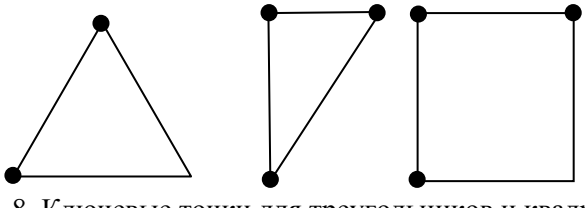


Рис. 8. Ключевые точки для треугольников и квадрата

Поскольку теперь два различных треугольника с различными ключевыми точками, то простейшее решение - создать отдельную программу для распознавания каждого типа. Таким образом, в дополнение к используемой issquare() для определения квадрата и istriangle() - для равнобедренного треугольника, добавляется isright() - для прямоугольного треугольника.

## 5.6. Распознаватель Дельта-Д

Последний метод распознавания, рассматриваемый в этой главе, основан на подсчете количества изменений направления. Например, начиная с любой точки треугольника осуществляется проход по всем его сторонам с возвратом в отправную точку, направление при этом изменяется три раза. При проходе по сторонам квадрата направление меняется четыре раза. Количество изменений направления соответствует количеству вершин формы. (Круг можно воспринимать как форму с бесконечным числом изменений направления или как имеющую нулевое изменение). Каждый раз при прохождении вершины, происходит изменение направления. В математике такие изменения часто обозначаются как дельта. Поскольку данная программа, представленная здесь, основана на изменении направления, то этот метод иногда называют распознавателем Дельта-Д, где D - происходит от слова direction (направление).

Распознаватель Дельта-Д должен уметь проходить по контуру объекта, то есть программа должна быть способной проходить по прямой линии до ее пересечения с другой прямой линией. В точке пересечения программа должна правильно найти следующую прямую линию и пойти по ней. Решение, представленное здесь, использует базу данных, названную oldp, в которой хранится список всех точек, используемый для того, чтобы программа знала, когда она вернулась в стартовую точку, то есть проследовала весь контур объекта.

Функция assert_oldp() сохраняет в базе местоположение точек; функция find() проверяет наличие в базе конкретной точки.

Основными функциями распознавателя Дельта-Д являются follow() и find_direction(). Зададим начальную точку. Функция follow() вызывает функцию find_direction() для определения значений приращений incx и incy. Затем происходит движение вдоль линии, определяемой этими приращениями. Если следующая позиция не содержит звездочки, которая показывает поворот, то follow() вызывает find_direction(), чтобы определить новый набор приращений. Процесс заканчивается, когда программа снова вернется в стартовую точку. Функция find_direction() просматривает все смежные позиции, чтобы найти звездочку. Функция find() возвращает "истина", если точка уже находится в базе oldp и "ложь" - если ее там нет. Поэтому в базе нет точек, которые были уже пройдены. Этот шаг предотвращает продвижение по уже пройденному пути, то есть исключает зацикливание.

Функция istriangle() и issquare() изменяются. В отличие от распознавания по ключевым точкам Дельта-Д распознаватель требует только одну функцию для распознавания треугольгика, так как все треугольники имеют три стороны. Точный размер и расположение - не имеет значения.

Если объект - треугольник, то направление изменится дважды (не три раза, так как эта версия функции istriangle() не рассматривает изменение начального направления). Следовательно, если follow() возвращает 2, то это показывает, что найден треугольник. Аналогично, если follow() возвращает 3 - распознан квадрат.