ЦИФРОВАЯ КУЛЬТУРА
УНИВЕРСИТЕТ ИТМО

Сравнение изображений: локальные признаки

Высшая школа цифровой культуры
Университет ИТМО
dc@itmo.ru

Содержание

1 Локальные признаки изображений: основные моменты

В начале нашего обсуждения видов признаков изображений мы упомянули, что их можно разделить на глобальные и локальные. Как глобальные, так и локальные признаки могут представлять определенные характеристики изображения - цвет, текстуру, форму, ориентацию краев. Однако, глобальные признаки характеризуют все изображение в целом, а локальные признаки описывают конкретный район, только часть изображения. Давайте теперь посмотрим, почему важны локальные признаки, и рассмотрим пару наиболее ярких примеров локальных признаков.

Когда глобальных признаков недостаточно Есть много случаев, когда глобальные признаки недостаточно устойчивы для сопоставления изображений. Например, если у нас имеется два изображения одного и того же объекта в разном масштабе, то сопоставить область одного изображения с другим изображением используя только глобальные признаки довольно сложно. Если изображения сняты с разных точек обзора, под разными условиями освещенности, с перекрытиями - нам нужно иметь возможность сопоставить локальные участки изображения одного изображения с локальными участками другого изображения.

Использование локальных признаков С помощью локальных признаков мы можем сопоставлять части изображений, даже если глобальные признаки не совпадают. Хорошо, было бы неплохо уметь сопоставлять не целые изображения, а их фрагменты. Но какие фрагменты нужно сравнивать? Как правильно выбирать наиболее информативные фрагменты? А также как лучше всего представить эти фрагменты? Нам нужно найти способ решить, какой фрагмент изображения выбрать, а затем как построить векторы признаков, представляющие эти фрагменты.

Как выбрать и сравнить фрагменты? Итак, как мы можем сопоставить фрагменты разных изображений? Допустим, мы хотим доказать, что объект на показанном на экране втором изображении меньшего размера соответствует виду объекта с первого изображения, но сзади. Простейший способ это сделать - сканировать оба изображения с помощью скользящих окон переменного размера и сравнить каждую возможную пару окон. Вероятно, это сработает, и мы найдем совпадающие фрагменты, пройдя по каждой возможной паре фрагментов. Но этот подход слишком медленный - только представьте, сколько возможных положений и размеров скользящего окна нам

придется проверить! Кроме того, нам нужно будет сравнить каждый возможный фрагмент на одном изображении со всеми возможными фрагментами на другом. Такой подход непомерно дорогостоящий с точки зрения вычислений.

Что, если мы выберем только подмножество фрагментов из обоих изображений и сделаем разреженное сопоставление? Очевидно, это будет быстрее! Мы можем выбрать фрагменты таким образом, чтобы они были изолированы, чтобы не было пересечения между разными фрагментами одного и того же изображения. Звучит неплохо. Но поскольку мы сравниваем не все возможные фрагменты, мы должны убедиться, что выбрали наиболее важные и репрезентативные фрагменты из всех возможных. Как это сделать? Как мы можем найти те части изображения, в которых мы можем устойчиво найти соответствия с другим изображением? Нам нужно найти такие области изображения, которые могут совпадать с областями на других изображениях. Участки без текстуры практически невозможно локализовать, поэтому в данном случае они не подходят. Участки со значительным изменением контраста легче локализовать. Нам нужно найти четкие области на изображениях, конкретные детали, такие как горные вершины, углы зданий, т.е. неоднородные, особые, информативные точки. Такие точки часто называют ключевыми точками или точками интереса.

Признаки хороших ключевых точек Каковы некоторые важные свойства хороших ключевых точек? Во-первых, их не должно быть слишком много. Нам нужно гораздо меньше ключевых точек, чем пикселей в изображении.

Далее, каждая ключевая точка должна быть информативной, уникальной. Если мы будем рассматривать много похожих окрестностей, будет сложно найти правильные совпадения на двух изображениях. Как в примере, который вы видите сейчас на экране - невозможно сказать, какая из трех красных точек на правом изображении соответствует красной точке на левом изображении.

Хорошие ключевые точки должны быть локальными. Ключевая точка должна занимать небольшую область изображения, и быть устойчивой к частичному перекрытию другим объекотом.

И, наконец, ключевые точки должны быть повторяемыми. Нам нужно, чтобы одна и та же точка находилась на разных изображениях, несмотря на геометрические и фотометрические изменения объекта съемки. Если совпадающих ключевых точек нет, сопоставить изображения будет невозможно. Необходимо, чтобы хотя бы подмножество ключевых точек одного изображения соответствовало подмножеству ключевых точек на другом. Кроме того, обнаружение ключевых точек должно происходить независимо на двух изображениях. Это означает, что используемый алгоритм обнаружения ключевых точек должен многократно обнаруживать одни и те же ключевые точки как

при применении к одному и тому же изображению, так и при применении к разным изображениям.

Области применения Обнаружение и сопоставление ключевых точек до сих пор являются важной задачей, встречающейся во многих сферах применения компьютерного зрения. Представьте, что мы хотим выровнять два изображения так, чтобы их можно было легко объединить в панорамный снимок. Чтобы сделать это автоматически, нам нужно определить ключевые точки на всех изображениях, которые мы хотим склеить, найти совпадающие местоположения и правильно их соединить.

Еще одна важная область применения - это 3D-реконструкция из двухмерных изображений. Имея ключевые точки мы можем установить такой набор соответствий, который позволит построить трехмерную модель или создать промежуточную модель.

Многие алгоритмы отслеживания движения используют ключевые точки. Обнаруживая и сопоставляя ключевые точки в соседних кадрах, можно обнаруживать и отслеживать движение.

Ключевые точки также используются в робототехнике для навигации роботов. Локальные признаки, основанные на ключевых точках, все еще используются в некоторых методиках поиска изображений. Еще десять лет назад они активно использовались для обнаружения и распознавания объектов, но теперь большинство приложений для обнаружения объектов (как и многие приложения для поиска изображений) пользуются преимуществами глубокого обучения.

1.1 Сопоставление изображений с ключевыми точками и локальные признаки

Процесс сопоставления изображений с использованием локальных признаков можно разделить на три отдельных этапа. Первый этап - обнаружение ключевых точек. На этом этапе на каждом изображении идет поиск таких областей, которые могут уверенно совпадать с областями на других изображениях. Второй этап - описание признаков. На этом этапе окрестности каждой обнаруженной ключевой точки преобразуются в более компактные дескрипторы, которые можно сопоставить с другими дескрипторами. Третий этап - это сопоставление признаков, когда алгоритм ищет подходящих кандидатов на двух изображениях.

Давайте рассмотрим каждый из этих этапов один за другим, начиная с обнаружения ключевых точек.

1.1.1 Использование угловых точек в качестве точек интереса

Интуитивно мы понимаем, что в угловых точках происходят быстрые изменения направления кривой края. Углы - очень эффективные признаки, поскольку они хорошо различимы и в разумной степени инвариантны к точке обзора. Благодаря этим характеристикам углы часто используются в качестве ключевых точек. Существует несколько алгоритмов обнаружения углов. Основная идея в большинстве алгоритмов - использоватнь небольшое окно и вычислить изменения яркости изображения при небольшом смещении окна.

Если пиксель находится в области с одинаковой яркостью, то значения в соседних положениях окна будут одинаковыми. Если пиксель находится на краю, то сдвигая окно в направлении, перпендикулярном краю, мы получим совсем иные значения, а сдвиг окна в направлении, параллельном краю, приведет лишь к небольшому изменению значений яркости. Если окно находится в угловой точке, то смещение в любом направлении приведет к значительному изменению интенсивности.

Один из самых ранних алгоритмов обнаружения углов, алгоритм Моравица, основан на вычислении суммы квадратов разностей между двумя соседними положениями окна. Харрис и Стивенс улучшили детектор углов Моравица, используя градиенты изображения в каждой точке вместо сдвига окна. По сути, они преобразовали эту простую идею об изменении интенсивности в небольшой окрестности в математическую форму.

1.1.2 Детектор Харриса: математика

. Детектор угла Харриса находит разность в интенсивности значений внутри окна, когда оно сдвигается на (u, v). Пусть $I(x, y)$ обозначает интенсивность изображения I в точке (x, y). Рассмотрим окно в области (x, y) и сместим его на (u, v). Взвешенная сумма квадратов разностей (WSSD) между этими двумя окнами определяется как:

$$E(u, v) = \sum_{x,y} w(x, y) \left[I(x + u, y + v) - I(x, y)\right]^2 \tag{1}$$

где $w(x, y)$ - это оконная (или весовая) функция, которая присваивает веса пикселям внутри его. Обычно это либо прямоугольное окно, когда оно равно 1 внутри окна и 0 где-либо еще, либо гауссово окно. Прямоугольное окно используется, когда важна скорость вычислений и уровень шума невысок. Окно Гаусса используется, когда важно сглаживание данных.

$I(x, y)$ - это интенсивность исходных точек, а $I(x + u, y + v)$ - интенсивность смещенных точек.

Для областей с практически постоянной интенсивностью, когда значения интенсивности в соседних окнах практически не меняются, $E(u, v)$ бу-

дет около 0. Для областей, в которых интенсивность сильно различается при небольшом смещении окна, $E(u, v)$ будет больше. Следовательно, нам нужны окна, в которых $E(u, v)$ велико.

Для малых (u, v), $I(x + u, y + v)$ может быть аппроксимировано рядом Тейлора. Пусть I_x и I_y - частные производные, если I такое, что

$$I(x + u, y + v) \approx I(x, y) + I_x(x, y)u + I_y(x, y)v \tag{2}$$

Это дает приближение для $E(u, v)$:

$$E(u, v) \approx \sum_{x,y} w(x, y) \left[I_x(x, y)u + I_y(x, y)v\right]^2 \tag{3}$$

Это уравнение может быть представлено в матричной форме:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{4}$$

где

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{5}$$

Матрицу M иногда называют матрицей Харриса.

Что показывает эта матрица? Как с ее помощью можно определить наличие области с монотонной интенсивностью, края или угла? Давайте сначала рассмотрим выровненный по оси угол. Таким образом, изменения интенсивности происходят только по осям x и y. Это означает, что произведение частных производных $I_x I_y$ равно нулю для всех позиций (x, y). А матрица M имеет вид

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & 0 \\ 0 & I_y^2 \end{bmatrix} = \sum_{x,y} w(x, y) \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{6}$$

Поскольку значения равны 0 везде, кроме главной диагонали, это означает, что главная диагональ содержит собственные значения этой матрицы.

Угловые участки на изображении будут характеризоваться большими значениями обоих собственных значений, что означает изменения интенсивности в обоих направлениях, x и y. Если какое-либо собственное значение близко к 0, то значит, эта позиция не угловая.

В этом случае мы предположили, что угол выровнен по оси. Но что если у нас есть угол, не совпадающий с осями изображения?

Поскольку M симметрична, мы можем представить ее следующим образом:

$$M = \sum_{x,y} w(x,y) X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T \tag{7}$$

Поскольку M является вещественной симметричной матрицей, ее собственные векторы указывают в направлении максимального разброса данных, а соответствующие собственные значения пропорциональны количеству данных, распределенных в направлении собственных векторов. Таким образом, собственные значения показывают величину изменения интенсивности в двух основных ортогональных направлениях градиента в окне.

Детектор Харриса: классификация по собственным значениям Теперь мы можем проанализировать собственные значения M, чтобы обнаружить углы. Угол (или вообще точка интереса) характеризуется большим изменением E во всех направлениях вектора $\begin{pmatrix} u & v \end{pmatrix}$, что соответствует большим значениям собственных значений M. M должен иметь два «больших» собственных значения, чтобы рассматриваемая точка являлась точкой интереса. На основании величин собственных значений можно сделать следующие выводы:

- Если оба собственных значения λ_1 и λ_2 близки к 0, то E почти постоянна во всех направлениях, и это означает, что пиксель (x, y) не имеет интересующих нас свойств и принадлежит к плоской однородной области.

- Если одно из собственных значений имеет большое положительное значение, а другое близко к 0, или если одно из собственных значений намного больше другого ($\lambda_1 >> \lambda_2$ or $\lambda_2 >> \lambda_2$), то это означает, что найден край.

- И, наконец, когда оба собственных значения имеют большие положительные значения, но отличаются друг от друга незначительно, то это означает, что E велико во всех направлениях, и найден угол.

Детектор Харриса: мера отклика для угла Таким образом, мы видим, что собственные значения матрицы, сформированной из производных на фрагменте изображения, могут использоваться для определения плоских областей, краев и углов. Но Харрис и Стивенс отметили, что точное вычисление собственных значений требует больших вычислительных ресурсов. Поэтому вместо этого они предложили использовать следующую функцию для измерения отклика угла:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det M - k(\mathrm{trace}M)^2 \quad (8)$$

Дело в том, что след квадратной матрицы равен сумме ее собственных значений, а ее определитель равен произведению ее собственных значений. Таким образом, нам не нужно фактически вычислять разложение по собственным значениям матрицы M, а вместо этого, чтобы найти углы или, скорее, точки интереса в целом, достаточно вычислить определитель и след M.

k - это эмпирическая константа. Диапазон ее значений зависит от области применения. В литературе считаются допустимыми значения в диапазоне $0.04\tilde{}0.15$. Можно интерпретировать k как «коэффициент чувствительности»: чем он меньше, тем больше вероятность, что детектор найдет углы.

Мера R принимает большие положительные значения, когда оба собственных значения велики, что указывает на наличие угла; она принимает отрицательные значения, когда одно собственное значение большое, а другое маленькое, что указывает на край; и ее абсолютное значение мало, когда оба собственных значения малы, что указывает на то, что рассматриваемый фрагмент изображения является плоским. Обычно R используется с порогом T. Мы говорим, что в определенной области изображения был обнаружен угол, только если $R > T$ для рассматриваемой области.

Детектор Харриса На этом слайде описаны все этапы работы детектора Харриса.

1. Вычислить производные изображения по горизонтали и вертикали I_x и I_y в каждой точке изображения с использованием Гауссова сглаживания. Как мы знаем из предыдущих лекций, это можно сделать, свернув исходное изображение с производными от гауссианов.

2. Вычислить матрицу M в гауссовом окне для каждого пикселя.

3. Вычислить меру отклика угла R.

4. Использовать порог R для того, чтобы сохранить только точки с достаточно большими значениями R.

5. Найти локальные максимумы меры отклика, используя немаксимальное подавление (nonmaximum suppression).

Схема детектора угла Показанная на экране схема работы является общей для многих детекторов угловых точек. Мы начинаем с входного изображения, применяем оператор угла, который создает карту «угловых точек».

Затем мы передаем эту карту угловых точек в оператор пороговой обработки и получаем карту порогов угловых точек. Эта карта затем передается оператору немаксимального подавления, который выводит положения угловых точек.

Пример 1 На экране представен один из примеров применения рассмотренной схемы к уже знакомому нам изображению Лена. Сначала мы переходим от исходного изображения к карте угловых точек, пороговой карте и, наконец, карте координат углов. Последнее изображение на экране показывает положение обнаруженных угловых точек, наложенных на входное изображение.

Пример 2 Рассмотрим еще один пример. Предположим, у нас есть два изображения одного и того же объекта, но одно представляет собой повернутую версию другого. Условия освещения тоже разные.

Давайте вычислим меру отклика угла Харриса R для обоих этих изображений. Красные точки на изображениях соответствуют высокому значению R, а синие точки соответствуют низким значениям R, оранжевые, желтые и зеленые точки имеют промежуточные значения между красным и синим.

Следующий шаг - оставить точки со значениями R, которые превышают выбранный порог. Таким образом, мы оставляем все точки, которые были красными или оранжевыми на предыдущем слайде, и отбрасываем желтые, зеленые и синие.

И последний шаг - выполнить немаксимальное подавление - найти и сохранить точки локальных максимумов R.

Теперь давайте наложим ключевые точки, обнаруженные с помощью детектора угловых точек Харриса, на исходные изображения. Вы можете заметить, что было найдено много пар соответствующих ключевых точек, и это именно то, чего мы добивались. На обоих изображениях есть ключевые точки, обнаруженные в местах расположения глаз и ноздрей, по границам черных пятен, а также во многих других выделяющихся областях.

Свойства детектора Харриса Давайте теперь рассмотрим свойства детектора Харриса. В идеале мы хотели бы, чтобы детекторы ключевых точек были инвариантны относительно поворота, изменения интенсивности и масштаба, чтобы одни и те же ключевые точки можно было найти в повернутых, осветленных или затемненных, отмасштабированных версиях изображений. А как насчет детектора Харриса? Инвариантен ли он ко всем этим изменениям?

Детектор Харриса инвариантен относительно поворота, как мы только

что видели на примере с повернутой игрушкой. Собственные значения остаются неизменными для повернутых версий изображения.

А как насчет сдвига интенсивности? Да, детектор Харриса инвариантен и к этим изменениям тоже. Поскольку для вычисления углового отклика используются только производные, он инвариантен к сдвигу интенсивности. А изменение шкалы интенсивности можно обрабатывать с помощью адаптивного определения порога.

А что насчет масштаба? К сожалению, детектор Харриса не инвариантен к масштабу. В зависимости от масштаба один и тот же фрагмент изображения может рассматриваться как содержащий только края или содержащий углы. Посмотрим на пример, представленный на экране. На изображении слева все окна по контуру должны были бы определить угол, но углов в окне нет. Если уменьшить тот же фрагмент до изображения, представленного справа, видно, что теперь у нас есть угол, попадающий в окно.

Инвариантность к изменению масштаба Одним из решений проблемы является извлечение признаков в различных масштабах. Если мы сможем изменить масштаб окна (или масштаб изображения), то мы сможем найти такие масштабы, в которых фрагменты будут совпадать. Та же самая кривая слева, что и на предыдущем слайде, может считаться краем и будет соответствовать кривой справа, если выбран правильный масштаб.

Вариант 2. Одним из решений проблемы является извлечение признаков в различных масштабах. Если мы сможем изменить масштаб окна (или масштаб изображения), то мы сможем найти такие масштабы, в которых фрагменты будут совпадать. Та же самая кривая слева, что и на предыдущем слайде, может считаться углом и будет соответствовать кривой справа, если выбран правильный масштаб.

Это можно сделать, выполнив ту же операцию определения углов с несколькими разрешениями в пирамиде изображений, полученных путем сглаживания и субдискретизации входных изображений.

Однако, необходимо выполнить поиск углов для нескольких разрешений, а затем для поиска лучшего совпадения выполнить $N \times N$ попарных сравнений для N масштабов, что обойдется дорого с точки зрения вычислений. Вместо того, чтобы извлекать признаки в разных масштабах и затем сопоставлять их все, более эффективно извлекать признаки, которые стабильны как по местоположению, так и по масштабу. Детектор Харриса является примером детектора, инвариантного относительно положения. Что же делать с масштабом? Как выбрать лучший масштаб для каждого фрагмента?

1.1.3 Выбор масштаба

Итак, как можно автоматически определить этот лучший масштаб? Нужно выбрать функцию, инвариантную к изменению масштаба, чтобы значение этой функции было одинаковым для соответствующих фрагментов, даже если эти фрагменты имеют разный масштаб. Одним из примеров такой функции является средняя интенсивность. Она будет одинаковой как для увеличенной, так и для уменьшенной версии одного и того же фрагмента изображения. Назовем ее характеристической функцией.

Затем мы можем рассмотреть эту функцию как на функцию от изменения размера области. Например, если изображение 2 является уменьшенной копией изображения 1 с коэффициентом масштабирования $1/2$, тогда такая функция для изображений 1 и 2 будет выглядеть примерно так, как графики, показанные на экране.

Затем давайте найдем локальный максимум этой функции и размер фрагмента, на котором достигается данный локальный максимум. Локальный максимум инвариантен к масштабу, поэтому точка, в которой достигается локальный максимум, является точкой характеристического размера.

Характеристический размер Давайте посмотрим на пример, представленный на экране: мы видим два изображения в разном масштабе. Возьмем точку на левом изображении, отмеченную желтым крестом. Теперь давайте изменим размер окна с центром в этой точке и вычислим некоторую функцию f для окон разного размера. Ниже приведен график этой функции f в зависимости от размера окна. Мы видим, что сначала она возрастает, когда мы увеличиваем размер области вокруг этого желтого креста, но затем она начинает убывать. Теперь выполним аналогичные действия с изображением справа. Теперь давайте посмотрим, при каком размере окна значение f было наибольшим для левого изображения и при каком размере окна f принимало наибольшее значение для правого изображения. Эта точка локального максимума является точкой характеристического размера. А соотношение размеров окон, соответствующих локальным максимумам на левом и правом изображениях, является соотношением масштабов этих изображений.

Реализация Вместо вычисления функции f для окон все большего размера, мы можем реализовать поиск характеристического размера, используя фиксированный размер окна и пирамиду Гаусса. Пирамида Гаусса - это такой вид представления изображения в разных масштабах, когда изображение подвергается многократному сглаживанию и субдискретизации. В пирамиде Гаусса последующие изображения получаются с использованием размытия по Гауссу и уменьшением масштаба.

Как выбрать характеристческую функцию f - часть 1 Как выбрать правильную «характеристическую» функцию f? Для определения масштаба часто используется нормированный по шкале лапласиан (LoG). Как вы, возможно, помните из предыдущих лекций, LoG - это оператор Лапласса $\nabla^2 f$, примененный к гауссовской функции G. Двумерная функция Гаусса определяется как

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{9}$$

$\frac{1}{2\pi\sigma^2}$ перед двумерным гауссовским ядром является нормирующей костантой. При использовании нормирующей костанты такое гауссовское ядро является нормированным ядром, т.е. его интеграл по всей области определения равен единице для каждого σ.

LoG определяется как

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \tag{10}$$

LoG нормированного Гауссиана не инвариантен к масштабу - он зависит от σ или величины размытия. Отклик производной гауссовского фильтра на идеальный край уменьшается с увеличением σ. Чтобы реакция оставалась неизменной (инвариантность к масштабу), мы должны умножить производную Гауссиана на σ. Лапласиан - это вторая производная Гауссиана, поэтому ее необходимо умножить на σ^2. Итак, чтобы получить независимость от масштаба, нужно использовать нормированного по масштабу LoG:

$$L(x, \sigma) = \sigma^2 \nabla^2 G \tag{11}$$

.

Как выбрать характеристическую функцию f - часть 2 Детектор Лапласиана Гауссиана представляет собой нормированный по масштабу LoG. Как вы можете видеть на экране, маска фильтра LoG имеет концентрическую структуру вокруг центральной точки, с положительными весами в центральной области и отрицательными весами в окружающем кольце. Таким образом, отклик этого фильтра будет максимальным, если применить его к окрестности изображения, которое имеет аналогичную (округлую или каплеобразную) структуру в соответствующем масштабе. Если мы посмотрим на окружности, обозначенные img1, img2 и img3, они будут обнаружены фильтром LoG соответствующего масштаба.

Характеристический размер и Лапласиан Таким образом, чтобы определить для данной точки изображения характеристический размер при помощи лапласиана, нужно рассмотреть отклики лапласиана в разном масштабе. И тот масштаб, при котором достигается пик отклика Лапласиана, и является характеристическим размером.

1.1.4 Детектор LoG

Обратите внимание, что для больших объектов округлой формы повторяющееся местоположение ключевой точки также может быть определено как центр большого объекта округлой формы. Таким образом, LoG можно применять как для поиска характеристического размера для данной области изображения, так и для прямого обнаружения инвариантных к масштабу областей путем поиска трехмерных (местоположение + масштаб) экстремумов LoG. Эта процедура показана на экране. Изображение свернуто с помощью фильтров LoG разного масштаба, и создает трехмерную карту отклика. Точки максимума на такой трехмерной карте отклика будут указывать на размер и расположение ключевых точек-капель. Детектор LoG также иногда называют детектором капель, потому что он обнаруживает в качестве областей интереса капли, а не углы, как детектор Харриса.

Пример: детектор больших объектов округлой формы Сейчас на экране показан пример работы детектора больших объектов округлой формы LoG. Обратите внимание на разные масштабы и расположение ключевых точек.

Аппроксимация LoG при помощи разности гауссианов (DoG) Вычисление производных второго порядка в лапласиане гауссиана требует больших вычислительных ресурсов. Но можно аппроксимировать LoG разностью двух гауссианов (DoG):

$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2) \tag{12}$$

Если мы вычтем один гауссиан из другого, причем первый будет иметь большее стандартное отклонение, чем второй, мы получим кривую, очень похожую на LoG. На экране вы можете видеть графики двух гауссианов, один с $\sigma = 3.2$ показан голубым цветом, а другой с $\sigma = 2.0$ показан темно-синим цветом, их разность, нанесенная красным, очень похожа на функцию мексиканской шляпы (или LoG), показанную на графике справа.

Наилучшее приближение к LoG может быть получено, когда стандартное отклонение первого гауссиана примерно в 1.6 раз больше стандартного

отклонения второго, поэтому $\sigma_1 / \ sigma a_2 = 1.6$, как и для гауссианов на графике, показанном слева.

Этот подход был предложен Дэвидом Лоу и является частью разработанного им алгоритма масштабно-инвариантного преобразования признаков (SIFT).

Разность по Гауссу Как и в случае с лапласианом гауссиана, разность гауссианов может быть использована для увеличения видимости краев изображения. Сейчас на экране показаны результаты свертки изначального изображения с двумя разными ядрами Гаусса, а также с разностью между ними (DoG). Обратите внимание, что свертка изображения с помощью ядра DoG дает тот же результат, что и вычитание результатов свертки изображения с двумя исходными гауссовскими ядрами:

$$I(x, y) \star G_1 - I(x, y) \star G_2 = I(x, y) \star (G_1 - G_2) \tag{13}$$

Размытие изображения с использованием ядра Гаусса подавляет высокочастотную пространственную информацию. Вычитание одного ядра из другого сохраняет пространственную информацию, которая находится в том диапазоне частот, которые сохраняются при использовании двух исходных фильтров. Таким образом, DoG представляет собой пространственный полосовой фильтр.

1.1.5 Обнаружение объектов округлой формы с помощью DoG

Итак, как мы только что убедились, LoG можно аппроксимировать при помощи DoG. Это означает, что различия гауссианов также можно использовать для обнаружения больших объектов округлой формы при использовании метода масштабно-инвариантного преобразования признаков. Вычитание смежных по масштабу уровней пирамиды Гаусса является эффективным методом в данном случае. Давайте посмотрим, как именно это можно сделать.

Детектор DoG использует подход каскадной фильтрации для обнаружения областей в изображении, являющихся инвариантными к изменению масштаба. Он ищет экстремумы в пространстве масштабов, причем пространство масштабов является представлением изображения как набора сглаженных изображений. Сглаженные изображения имитируют потерю деталей, которая может произойти при уменьшении масштаба изображения. Сглаживанием управляет параметр, определяющий размер масштаба. Для реализации сглаживания используются ядра Гаусса, а параметром масштаба является стандартное отклонение, σ. Таким образом, пространство масштабов представляет собой пирамиду Гаусса: входное изображение последовательно сворачивается с помощью ядер Гаусса, имеющих стандартные отклонения σ_1,

$k\sigma_1$, $k^2\sigma_1$, $k^3\sigma_1$, . . ., чтобы создать «стек» отфильтрованных по Гауссу (сглаженных) изображений, разделенных постоянным коэффициентом k, как показано на экране.

Пространство масштабов разделено на октавы, где каждая октава соответствует удвоению σ, так же как в музыке октава соответствует удвоению частоты звукового сигнала. Затем каждая октава делится на целое число интервалов s так, чтобы $k = 2^{1/s}$. В стеке размытых изображений для каждой октавы создаются $s + 3$ изображений. На экране октава 1 состоит из пяти изображений, поэтому $s = 2$ и $k = \sqrt{2}$. Входное изображение последовательно сглаживается сверткой с гауссианами со стандартными отклонениями σ_1, $\sqrt{2}\sigma_1$, $2\sigma_1$ и т.д. На экране показан полученный набор изображений пространства масштабов. Затем производится вычитание соседних гауссовых изображений, чтобы получить изображение с разностью гауссианов (DoG), показанное справа. После обработки полной октавы мы передискретизируем гауссовское изображение, которое имеет двойное начальное значение σ, взяв каждый второй пиксель из каждой строки и столбца - это будут 2 изображения с вершины стека.

Можно показать, что когда два соседних масштаба шкалы разделены постоянным множителем k, то необходимая нормировка масштаба уже выполнена. Таким образом, в отличие от LoG, дополнительная нормировка масштаба не требуется.

Как и в случае с детектором LoG, области интереса в соответствии с DoG определяются как области, которые одновременно являются экстремумами в плоскости изображения и вдоль координаты функции масштаба $D(x, \sigma)$. Такие точки находятся путем сравнения значения $D(x, \sigma)$ каждой точки с 8 ее ближайшими соседями на том же уровне масштаба и с 9 ближайшими соседями на каждом из двух смежных уровней, выше и ниже, как показано на экране (справа). Подводя итог, детектор DoG ищет экстремумы трехмерного пространства функции DoG.

Пример Сейчас на экране показаны изображения первых трех октав шкалы масштаба. В таблице показаны значения стандартного отклонения, используемые в каждой шкале каждой октавы. Например, стандартное отклонение, используемое для масштаба 2 октавы 1 равен $k\sigma_1$, что равно 1.

Сейчас на экране показаны примеры изображений DoG. Мы имеем всего $s + 3$ изображений с использованием фильтра Гаусса, $s + 2$ разностей гауссианов, полученных в каждой октаве из смежных пар изображений с гауссовой фильтрацией в данной октаве. Разности можно рассматривать как изображения, и для каждых трех октав на экране показан один образец такого изображения. Как и следовало ожидать, уровень детализации этих изображений уменьшается по мере движения вверх в пространстве масштабов.

Обнаружение ключевых точек: выводы относительно инвариантности Давайте подытожим, что же мы узнали об обнаружении ключевых точек. Напомним, что для двух изображений, содержащих одну и ту же сцену, целью любого алгоритма обнаружения ключевых точек является найти одинаковые ключевые точки (или точки интереса) в каждом изображении независимо друг от друга. В идеале мы хотели бы, чтобы этот алгоритм был инвариантен к таким параметрам, как изменение освещения, поворот, масштаб и точка обзора. Мы подробно обсудили три алгоритма. Во-первых, детектор углов Харриса, который определяет углы в качестве ключевых точек. Мы выяснили, что этот детектор инвариантен к изменениям освещения и повороту, но не инвариантен к масштабу.

Затем мы обсудили два инвариантных к масштабу детектора: LoG и DoG. Оператор Лапласа редко используется сам по себе, так как он похож на DoG, но более сложен с точки зрения вычислений. Однако, он иногда используется в сочетании с оператором Харриса. Оператор Харриса-Лапласа был предложен для увеличения различительной способности по сравнению с операторами LoG или DoG. Он сочетает в себе специфику оператора Харриса для определения угловых точек с механизмом выбора масштаба LoG.

Как упоминалось ранее, детектор DoG является частью предложенного Дэвидом Лоу алгоритма извлечения признаков SIFT, поэтому его также часто называют детектором SIFT.

Масштабно-инвариантный поиск: выводы Если сравнить два рассмотренных масштабно-инвариантных детектора, Харриса-Лапласа и SIFT, оба они работают в тех случаях, когда мы рассматриваем изображения одной и той же сцены с большой разницей в масштабе, и оба этих метода ищут максимумы определенной функции (или функций) в данных пространствах масштабов в изображениях.

Метод Харриса-Лапласа строит два отдельных пространства масштабов для функции Харриса и лапласиана. Первая версия этого метода использует функцию Харриса для локализации точек-кандидатов на каждом уровне шкалы и выбирает те точки, для которых лапласиан одновременно достигает экстремума по шкалам. Полученные точки устойчивы к изменениям масштаба, повороту изображения, изменениям освещения и шуму фотоаппарата. Кроме того, они очень хорошо находят точки интереса. В качестве недостатка можно упомянуть, что настоящий детектор Харриса-Лапласа обычно возвращает гораздо меньшее количество точек, чем детекторы LoG или DoG. Это результат использования дополнительного ограничения, которое заключается в том, что каждая точка должна одновременно удовлетворять двум различным условиям максимума. Для некоторых практических применений поиска объектов меньшее количество областей интереса может быть недостат-

ком, поскольку это снижает устойчивость к частичному перекрытию. По этой причине была предложена обновленная версия детектора Харриса-Лапласа, основанная на менее строгом критерии. Вместо того, чтобы искать точки, в которых максимумы достигаются одновременно, этот алгоритм выбирает максимумы масштаба лапласиана в таких точках, в которых Функция Харриса также достигает максимума на любой шкале. В результате, модифицированный таким образом детектор дает больше точек интереса при немного более низкой точности, что приводит к повышению производительности для приложений, где необходимо найти как можно больше областей интереса.

Детектор DoG (или SIFT) ищет максимумы разности гауссианов как в масштабе, так и в пространстве.

1.1.6 Локальные дескрипторы - сравнение элементов

Теперь мы знаем, как независимо определять ключевые точки на разных изображениях. Следующий вопрос - как сопоставить эти точки, т.е. как найти соответствующие пары. Нам нужны локальные дескрипторы для ключевых элементов, чтобы их можно было сравнивать на двух изображениях с помощью некоторой функции подобия. Как и в случае с глобальными признаками, мы хотим, чтобы локальные дескрипторы были инвариантны к изменениям освещения, положения, масштаба и т.д. Но в то же время эти дескрипторы должны быть хорошо различимыми, чтобы одному признаку можно было с большой вероятностью найти правильное соответствие в большой базе данных признаков.

Как мы уже знаем, масштабно-инвариантные детекторы ключевых точек могут дать нам информацию о масштабе. Таким образом, мы можем использовать эту информацию, чтобы выбрать окрестность правильного масштаба в местоположении ключевой точки для вычисления дескриптора. Глобальные дескрипторы, которые мы обсуждали ранее, такие как гистограммы цвета или интенсивности, отклики фильтра и другие, также могут использоваться в таком случае, если применить их к локальному участку изображения заданного масштаба и в заданном положении, указанном алгоритмом обнаружения ключевых точек. Но многие из этих признаков либо чувствительны даже к небольшим изменениям положения и позы, либо плохо работают, либо и то, и другое. Один из наиболее удачных локальных дескрипторов был предложен Дэвидом Лоу вместе с детектором ключевых точек DoG. Это хорошо известный дескриптор SIFT.

1.1.7 Дескриптор SIFT: основные этапы

Основные этапы работы дескриптора SIFT перечислены ниже:

1. Обнаружение экстремумов в пространстве масштабов. Мы подробно обсудили этот этап. Он реализован с использованием разности гауссианов.

2. Локализация ключевых точек. В каждом потенциальном местоположении используется детализированная модель для определения местоположения и масштаба. Ключевые точки выбираются на основе мер их стабильности, краевые отклики устраняются.

3. Назначение ориентации. Каждому местоположению ключевой точки назначается одна или несколько ориентаций на основе локальных направлений градиента изображения. Все будущие операции выполняются над данными изображения, которые были преобразованы относительно назначенной ориентации, масштаба и местоположения каждого признака.

4. Дескриптор ключевой точки. Градиенты локального изображения вычисляются в выбранном масштабе в области вокруг каждой ключевой точки. Затем они преобразуются в представление, допускающее изменения уровней освещенности и значительные искажения локальных форм.

Рассмотрим подробнее этапы 3 и 4.

Ориентация ключевых точек Чтобы охарактеризовать изображение в каждом потенциальном расположении ключевой точки, на каждом уровне пирамиды сглаженное изображение обрабатывается для получения градиентов и ориентации изображения. Для каждого пикселя A_{ij}, вычисляются значения градиента изображения, M_{ij}, и ориентации, R_{ij}, при использовании разности пикселей:

$$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i,j+1})^2} \tag{14}$$

$$R_{ij} = atan2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij}) \tag{15}$$

Для каждого положения ключевой точки выбирается каноническая ориентация таким образом, чтобы дескрипторы изображения были инвариантны к повороту. Чтобы сделать ее максимально устойчивой к изменениям освещения или контрастности, ориентация определяется как пик на гистограмме ориентаций локальных градиентов изображения. Такая гистограмма формируется из ориентаций градиента точек выборки в окрестности каждой ключевой точки. Гистограмма имеет 36 интервалов, охватывающих на плоскости изображения диапазон ориентации 360∘. Каждый образец, добавленный к гистограмме, взвешивается по величине градиента и круговой функции Гаусса

со стандартным отклонением, в 1,5 раза превышающим масштаб ключевой точки. Пики на гистограмме соответствуют доминирующим локальным направлениям локальных градиентов. После чего обнаруживается самый высокий пик на гистограмме, причем любой другой локальный пик, находящийся в пределах 80% от самого высокого пика также используется для создания другой ключевой точки с такой же ориентацией. Таким образом, для областей, содержащих несколько пиков одинаковой величины будет создано несколько ключевых точек в одном месте и в одном масштабе, но с разной ориентацией.

Изображение на экране показывает ключевые точки, наложенные на изображение, причем ориентация ключевых точек показана стрелками. Обратите внимание на согласованность ориентации похожих наборов ключевых точек на изображении. Например, посмотрите на ключевые точки с правой стороны здания в вертикальном направлении. Длина стрелок варьируется в зависимости от освещения и содержания изображения, но их направление безошибочно согласовано.

Дескриптор SIFT Итак, мы присвоили каждой ключевой точке признаки ее расположения, масштаба и ориентации. Эти параметры создают повторяемую локальную двумерную систему координат, в которой описывается локальная область изображения, и, следовательно, обеспечивают неизменность этих параметров. Следующим шагом является вычисление дескриптора для выделяющейся области локального изображения, которая, в то же время, является максимально инвариантной по отношению к остальным изменениям, таким как изменение освещения или трехмерной точки обзора.

Подход, который используется для вычисления дескрипторов в алгоритме SIFT, основан на экспериментальных результатах, предполагающих, что локальные градиенты изображения, по-видимому, выполняют функцию, аналогичную той, которую выполняет человеческое зрение для сопоставления и распознавания трехмерных объектов с разных точек зрения. Сейчас на экране представлено вычисление дескриптора ключевой точки.

Сначала вокруг местоположения ключевой точки выбираются величина градиента изображения и направление ориентации, причем для выбора уровня размытия изображения по Гауссу используется масштаб ключевой точки. Чтобы достичь инвариантности ориентации, координаты дескриптора и ориентации градиента поворачиваются относительно ориентации ключевой точки. Для повышения эффективности градиенты предварительно вычисляются для всех уровней пирамиды и используются как для определения ориентации ключевых точек, так и для вычисления дескриптора. В центре экрана маленькими стрелками показаны градиенты изображения для каждой выборки. Для присвоения веса каждой точке выборки используется гауссовская

функция с σ, равная половине ширины окна дескриптора, что и показано в круглом синем окне в центре экрана, хотя, в реальности, вес снижается более плавно. Назначение этого гауссовского окна - избежать внезапных изменений дескриптора при небольших изменениях положения окна и сделать меньший акцент на градиентах, которые находятся далеко от центра дескриптора.

Дескриптор ключевой точки показан в правой части экрана. Гистограммы ориентации создаются по выборкам 4x4. На рисунке показаны восемь направлений для каждой гистограммы ориентации, причем длина каждой стрелки соответствует величине этой гистограммы. Градиент, показанный на центральном изображении, может сдвигаться по 4 положениям, при этом он будет по-прежнему вносить вклад в ту же гистограмму справа, поскольку такой дескриптор допускает локальные позиционные сдвиги. Дескриптор формируется из вектора, содержащего значения всех записей гистограммы ориентации, соответствующие длинам стрелок на изображении справа. На рисунке справа показан массив гистограмм ориентации 2x2, но, согласно мнению автора метода, наилучшие результаты достигаются при использовании массива гистограмм 4x4 с 8 ячейками ориентации в каждой. Следовательно, дескриптор SIFT обычно представляет собой вектор признаков 4x4x8 = 128 элементов для каждой ключевой точки.

1.2 Какую модель выбрать?

Ключевые точки и локальные дескрипторы - полезные инструменты для обнаружения и описания небольших выделяющихся областей изображения. Но эти дескрипторы часто зашумлены и мало что могут сказать о картине большего масштаба. Часто мы заранее знаем формы интересующих нас объектов. Одним из примеров наиболее распространенных форм, обнаружение которых может быть нам интересно, являются прямые линии. Мир, созданный человеком, полон прямых линий. Обнаружение и сопоставление прямых может быть полезно во множестве приложений. Например, для определения полосы движения и понимания общей дорожной разметки как для помощи водителю, так и для беспилотных автомобилей. Например, поиск места для парковки. Среди других распространенных примеров можно назвать обнаружение границ поля в анализе спортивных видеозаписей, обнаружение прямых при архитектурном моделировании, оценивании положения камеры в городских условиях, анализе макетов печатных документов и многих других сферах. В число распространенных простых форм также входят круги. В показанном на экране примере задача состоит в том, чтобы автоматически сосчитать монеты, и в таком случа, можно было бы извлечь пользу из знания, что монеты должны быть круглыми.

В случаях, когда мы заранее знаем, какой тип формы мы ищем, мы можем выбрать параметрическую модель для представления набора ключевых

точек, формирующих интересующий нас объект, а затем определить правильные параметры модели, чтобы она подходила для данных наблюдений.

1.2.1 Параметрическая модель

Так что же это за параметрическая модель? Параметрическая модель - это функция от данных изображения и некоторых параметров. Эта функция определяет класс фигур, которые мы ищем на изображении. Но точное расположение этой формы и другие свойства этой формы, такие как, например, размер, могут изменяться ти задаваться параметрами. Цель процесса выбора модели - найти такие параметры, чтобы данная модель наилучшим образом соответствовала данным наблюдения.

1.3 Параметры модели для поиска прямой

Например, мы ищем прямую, и нам дается набор точек - это данные изображения, наша цель состоит в том, чтобы найти прямую, которая наилучшим образом аппроксимирует данные точки. В таком случае, параметрическая модель представляет собой линейное уравнение: $y = mx+b$. Нам известно много пар (x, y) - ключевых точек, являющихся данными изображения. Наша задача - найти такие значения параметров m и b, чтобы линейное уравнение $y = mx + b$ точно соответствовало всем (или большинству) ключевых точек.

Давайте представим, что вы обнаружили красные ключевые точки на изображении, показанном на экране, и ваша задача - найти лучшую прямую, которая может соответствовать этим точкам - например, это может быть прямая, показаная синим цветом на экране.

1.3.1 Как выбрать параметры модели: общие рекомендации

Существует множество различных методов и алгоритмов выбора параметров модели, но все они работают по одному алгоритму.

- Во-первых, нужно выбрать параметрическую модель, которая, как ожидается, лучше всего соответствует имеющимся данным. Если вы ищете прямые на изображении, вашей параметрической моделью будет прямая. Если вы ищете круги (и ожидаете найти их на имеющемся изображении), выберите параметрическое уравнение круга или эллипса в качестве модели.

- После того, как вы выберете модель, следующим шагом будет подгонка этой модели к вашим данным, поиск таких значений для всех параметров модели, чтобы она хорошо аппроксимировала ваши данные. Имейте в виду, что реальные данные всегда неточны. Вы будете иметь дело с

шумом в местах обнаруженных ключевых точек, с выбросами, с отсутствующими в результате перекрытия точками. Модель должна быть устойчивой ко всем этим проблемам и хорошо описывать фактические данные. Невозможно точно подогнать модель ко всем имеющимся у вас точкам данных. Но лучшей моделью будет та, которая подходит для описания максимально возможного количества точек. Таким образом, вам необходимо максимизировать количество данных, «вписывающихся» в модель и минимизировать количество «выбросов».

- Когда у вас уже есть параметрическая модель и параметры, которые лучше всего подходят для ваших данных, вы можете отфильтровать те точки данных, которые не соответствуют модели. Это выбросы.

Различные алгоритмы подгонки модели по-разному подходят к шагу номер два. Это самый важный шаг из перечисленных трех. Итак, как найти «оптимальные» параметры для выбранной модели?

1.3.2 Методы поиска модели

В качестве примера рассмотрим задачу поиска модели для прямой линии. Один из самых простых алгоритмов подгонки модели - это метод наименьших квадратов. Его можно использовать, когда мы знаем, какие точки принадлежат прямой, и перед нами стоит задача найти «оптимальные» параметры прямой. Его можно использовать также для других типов моделей. Но этот метод очень чувствителен к выбросам, поэтому он работает хорошо только тогда, когда исходные данные не зашумлены и вы ожидаете, что все точки данных впишутся в модель и не будут выбросами.

В случае, когда в данных есть выбросы, лучше выбрать более устойчивый алгоритм подгонки. Один из самых популярных алгоритмов в этой категории - RANSAC (консенсус случайной выборки), т.е. стабильный метод оценки параметров модели на основе случайных выборок.

Если вы ожидаете, что у вас несколько экземпляров объекта или несколько разных прямых (в случае, если наша задача - поиск прямой, и задача состоит в том, чтобы найти все линии), тогда можно использовать такие методы, как RANSAC или преобразование Хафа (Hough transform).

Теперь давайте поближе познакомимся с этими популярными методами подгонки.

Поиск модели для поиска прямой методом наименьших квадратов Начнем с метода наименьших квадратов. Этот метод является стандартным подходом в регрессионном анализе для аппроксимации решения переопределенных систем, когда уравнений больше, чем неизвестных. Ключевая идея

метода - минимизировать сумму квадратов отклонений, полученных в результате каждого отдельного уравнения.

В применении к подгонке модели к данным, отклонение - это разница между наблюдаемым и подобранным значением, полученными с помощью модели. Метод наименьших квадратов - это особый вид минимизации, цель которого - минимизировать квадраты отклонений.

Давайте начнем с самого начала. Наша цель - найти параметры моделирующей функции, которые лучше всего подходят для имеющегося набора данных. Набор данных состоит из n точек (пар данных): $(x_1, y_1), \ldots, (x_n, y_n)$. Моделирующаяя функция имеет форму линейного уравнения: $y = mx + b$. Цель состоит в том, чтобы найти значения параметров (m, b) для модели, которые "наилучшим образом"будут соответствовать данным. Подгонка модели к точке данных (x_i, y_i) измеряется ее отклонением, определяемым как разность между фактическим значением y_i и значением, предсказанным моделью: $r_i = y_i - mx_i - b$. Наша цель - минимизировать сумму квадратов отклонений для всех точек данных:

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \tag{16}$$

Если мы переведем все в векторы и матрицы, мы можем представить это уравнение в матричной форме:

$$E = ||Y - XB||^2 \tag{17}$$

где

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \; X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \; B = \begin{bmatrix} m \\ b \end{bmatrix} \tag{18}$$

Это выражение можно переписать следующим образом:

$$E = ||Y - XB||^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB) \tag{19}$$

Минимум E можно найти, установив частную производную по B равной нулю:

$$\frac{\partial E}{\partial B} = 2X^T XB - 2X^T Y = 0 \tag{20}$$

Первый член в E, $Y^T Y$ не зависит от B, поэтому его частная производная по B равна нулю, второй член дает нам $-2X^T Y$, а последний дает $2X^T XB$.

Мы приходим к нормальному уравнению вида $X^T XB = X^T Y$. Такие уравнения называются нормальнымм, потому что они задают, что отклонение должно быть нормально (перпендикулярно) каждому вектору на отрезке X.

Подставив в уравнение, мы получим

$$B = (X^T X)^{-1} X^T Y \tag{21}$$

Традиционный метод наименьших квадратов: недостатки Хотя традиционный (также называемый обычным) метод наименьших квадратов очень распространен и прост в использовании, у него есть некоторые проблемы. Во-первых, он не инвариантен относительно вращения. Поскольку мы минимизируем отклонения по оси y, при повороте изображения отклонения по оси y изменятся. Во-вторых, он совсем не подходит для вертикальных линий.

Также отметим, что традиционный метод наименьших квадратов асимметричен. Мы рассматриваем Y и X как зависимые переменные, причем при этом предполагается, что независимая переменная не содержит ошибок.

Все эти проблемы могут быть решены с помощью другого варианта метода наименьших квадратов, называемого Метод наименьших полных квадратов (Total Least Squares).

Метод наименьших полных квадратов Метод наименьших квадратов и метод наименьших полных квадратов оценивают точность аппроксимации по-разному: метод наименьших квадратов минимизирует сумму квадратов вертикальных расстояний от точек данных до линии аппроксимации, а метод наименьших полных квадратов минимизирует сумму квадратов ортогональных расстояний от точек данных до аппроксимирующей линии.

Возвращаясь к нашей задаче подгонки модели прямой, теперь мы можем переформулировать ее с помощью метода наименьших квадратов следующим образом: минимизировать сумму расстояний между точками (x_i, y_i) и прямой $ax + by = d$, где (a, b) - отрезок, перпендикулярный прямой, поэтому $a^2 + b^2 = 1$. Наша цель - найти такие (a, b, d), которые минимизируют сумму перпендикулярных расстояний:

$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2 \tag{22}$$

Метод наименьших квадратов vs метод наименьших полных квадратов На экране показана разница между прямыми, полученными по методу наименьших квадратов и методу наименьших полных квадратов, аппроксимирующие прямыми один и тот же набор точек. Крестики, расположенные на прямых, представляют собой аппроксимацию данных с использованием этих двух подходов подгонки. В случае использования метода наименьших

квадратов аппроксимация данных достигается путем корректировки только второй координаты: исходные точки данных и их приближения на прямой имеют одинаковое значение x, и только значения y различны. В случае использования метода наименьших полных квадратов аппроксимация данных получается путем корректировки обеих координат. Метод наименьших полных квадратов в регрессии - это такой метод моделирования данных с использованием метода наименьших квадратов, когда ошибки в откликах и предикторах учитываются как для зависимых, так и для независимых переменных.

Метод наименьших квадратов: устойчивость к шуму При использовании методов наименьших квадратов, как традиционного, так и метода наименьших полных квадратов, возникает еще одна проблема. Они очень чувствительны к шуму. Эти методы очень хорошо подгоняют модель прямой линии к данным в том случае, когда все точки данных близки к линии, как показано сейчас на экране.

Но если в наших данных есть выбросы, аппроксимация методом наименьших квадратов будет далека от идеала. Среднеквадратичная ошибка сильно зависит от выбросов. Таким образом, ошибки в данных будут сильнее влиять при использовании метода наименьших полных квадратов, чем при использовании метода наименьших квадратов.

1.3.3 Устойчивые оценки

Когда имеется всего несколько выбросов, проблема может быть решена с помощью устойчивых оценок. Устойчивые оценки основываются на устойчивых штрафных функциях, которые более «снисходительны» к точкам данных, значения которых значительно отличаются от аппроксимированнх при помощи данной модели.

Общий подход состоит в том, чтобы найти параметры модели θ, которые минимизируют сумму значений устойчивой функции для всех точек данных x_i: $\sum_i \rho(r_i(x_i, \theta), \sigma)$. Устойчивая функция ρ является функцией от отклонений r_i и параметра масштаба σ. На экране показан пример такой устойчивой функции, $\rho(u, \sigma) = \frac{u^2}{\sigma^2+u^2}$. Она ведет себя как квадрат расстояния для малых значений отклонений u, но отлична от него при больших значениях u. Таким образом, если в ваших данных всего несколько выбросов, с помощью устойчивой оценки вы сможете подогнать модель к выбросам, и влияние выбросов на параметры модели будет минимизированно.

Устойчивая подгонка модели - это нелинейная оптимизация, которая выполняется итеративно (в отличие от метода наименьших квадратов, который может быть решен в аналитическом виде). Для инициализации можно ис-

пользовать метод наименьших квадратов.

К выбору параметра масштаба устойчивой функции следует отнестись с осторожностью - при слишком маленьком значении ошибка будет почти одинаковой для каждой точки, и поэтому будет трудно найти подходящую аппроксимацию. Если значение масштаба слишком велико, функция будет вести себя аналогично методу наименьших квадратов.

1.3.4 Консенсус случайной выборки (RANSAC)

Устойчивые M-оценки определенно могут помочь уменьшить влияние выбросов, однако, если выбросов слишком много, такими методами трудно достичь глобального оптимума. Более хорошим подходом часто является поиск начального набора исходных соответствий, то есть точек, которые согласуются с оценкой доминирующих параметров. Широко используемый подход для решения этого вопроса называется RANSAC или стабильный метод оценки параметров модели на основе случайных выборок. Этот метод - основа для подгонки модели при наличии выбросов.

Основные этапы этого метода заключаются в следующем.

- Нужно случайным образом выбрать некоторое равномерной подмножество точек
- Подогнать модель к этому подмножеству
- Найти все оставшиеся точки, которые «близки» к модели - подсчитать количество точек, которые находятся в пределах ϵ от их положения, предсказанного моделью. Отбросить остальные точки как выбросы.
- Повторить эти шаги некоторое количество раз и выбрать лучшую модель с наибольшим количеством точек, хорошо описываемых моделью.

RANSAC: подгонка модели прямой Давайте посмотрим на пример подгонки прямой линии с помощью RANSAC. Предположим, наши точки данных выглядят так, как показано на экране. Вы можете видеть четкую диагональную линию, идущую слева направо вверх. Но изображение зашумлено и мы видим много выбросов.

Метод наименьших квадратов не сможет найти хорошее решение из-за большого количества выбросов.

С помощью RANSAC мы сначала случайным образом выбираем минимальное подмножество точек, показанное красной линией на экране. Поскольку мы хотим в качестве модели использовать прямую, то минимальное подмножество точек для определения прямой равно двум. На следующем шаге мы подбираем модель для этого подмножества. Затем мы вычисляем

функцию ошибок для этой модели и всех имеющихся у нас точек данных. Далее мы выбираем точки, которые соответствуют модели. Это точки, находящиеся на расстоянии ϵ от прямой, определенной моделью. Эти точки показаны на слайде зеленым цветом.

Далее, мы выбираем точки, соответствующие модели. Это точки на небольшом расстоянии ϵ от прямой, определенной нашей моделью. Такие точки показаны на слайде зеленым цветом.

Мы повторяем этот цикл, начиная со случайной выборки минимального подмножества точек, затем подгоняем модель под них и подсчитываем точки, соответствующие модели. Сейчас на экране показана еще одна гипотетическая прямая.

Повторим еще раз. Снова повторим. И еще раз.

Затем мы выберем такую модель из всех возможных гипотез, которой соответствует наибольшее количество точек.

Консенсус случайной выборки (RANSAC) для поиска прямой линии Подводя итог, консенсус случайной выборки для поиска прямой линии состоит из следующих шагов, повторяемых N раз.

- Случаным образом выберите s точек (равномерно). Обычно используется выборка из минимально возможного количества точек. Итак, для выбора модели прямой выберем две точки.
- Выберите прямую, подходящую к этим s точкам.
- Среди оставшихся точек выберите точки, которые описываются моделью: найдите точки, расстояние от которых до прямой меньше ϵ.
- Если вы получили d или больше точек, которые описываются моделью, эта прямая принимается и нужно заново ее пересчитать, используя все точки, принятые моделью.

Плюсы и минусы RANSAC Консенсус случайной выборки - это простой, общеизвестный и устойчивый метод оценки. С его помошью можно с высокой точностью оценить параметры модели даже в случае наличия большого количества выбросов в данных. Он широко используется и применяется для решения множества различных задач, не только для подбора моделей. Зачастую, он хорошо работает и на практике.

Но у него есть и недостатки. Как вы видели на предыдущем слайде на экране, он зависит от большого количества параметров. Существуют подходы для определения наилучшего количества итераций, количества точек,

которые нужно выбрать за итерацию, правильного расстояния для фильтрации выбросов и т.д. И зачастую выбор правильных параметров очень важен, т.к. неправильный выбор параметров может привести к слишком длительным вычислениям или некорректным результатам. Если количество итераций ограничено, полученный результат может быть неоптимальным, и может недостаточно точно описывать имеющиеся данные. RANSAC предлагает компромисс; при большом количестве итераций вероятность создания хорошей модели увеличивается.

Этот метод плохо работает, если выбросов слишком мало или слишком много. Консенсус случайной выборки не всегда может найти оптимальный набор даже для умеренно зашумленных наборов, и обычно он плохо работает, когда количество данных, хорошо описываемых моделью, меньше 50

Для некоторых моделей может быть сложно найти хорошую инициализацию модели на основе минимального количества элементов выборки.

Методы выбора и поиск модели Мы обсудили несколько самых популярных методов подбора модели. Метод наименьших квадратов и его модификации - одни из самых простых методов, но если выбросов слишком много, они не будут работать хорошо. В случае аппроксимации прямой метод наименьших квадратов хорошо работает, если большинство точек данных расположены близко к одной прямой. Если в данных много выбросов или если точки данных образуют несколько прямых, методы наименьших квадратов не работают.

Методы устойчивых оценок менее чувствительны к выбросам, но все равно не работают, если выбосов слишком много.

RANSAC - очень популярный подход, который достаточно хорошо справляется с выбросами. Но обычный консенсус случайной выборки может оценить только одну модель для определенного набора данных. Если данные предполагают существование двух (или более) моделей, консенсус случайной выборки в лучшем случае обнаружит только одну из них или даже может не найти ни одной.

1.3.5 Что если прямых несколько?

Как вы можете видеть на этом экране, методы наименьших квадратов совершенно не работают при наличии нескольких моделей и выбросов, а консенсус случайной выборки способен обнаруживать одну модель – одну прямую из пяти. Вероятно, эта прямая описывает большее количество данных, чем другие линии. Или этой прямой просто "повезло больше"и поэиому она была обнаружена в течение заданного количества итераций алгоритмом RANSAC.

Выбор модели Метод RANSAC может быть адаптирован к случаю, когда в данных можно обнаружить несколько моделей. Одна из наиболее простых модификаций RANSAC для случая с несколькими моделями обычно называется последовательным консенсусом случайной выборки. Еще один альтернативный устойчивый метод оценки - это преобразование Хафа, которой обычно используется, если в данных присутствует более одной модели. Существуют и другие подходы для случая, когда в данных присутствует несколько моделей, но сегодня мы сосредоточимся на двух вышеупомянутых методах, как на наиболее широко известных. Их обычно можно назвать процедурами голосования, поскольку они основаны на понятии «голосования» точек данных за конкретную модель.

1.3.6 Методы голосования

Основной принцип всех методов голосования одинаков. Каждая точка данных вносит свой вклад, т.е. голосует, за все модели, которые подходят для описания этой точки. Побеждает модель, которая наберет наибольшее количество голосов. В случае, если у нас несколькими моделей в изображении, мы можем выбрать не одну лучшую модель, а несколько, т.е. выбрать несколько моделей с количеством голосов выше определенного порога. Все выбранные модели будут лучшими и будут представлять собой несколько экземпляров одного и того же объекта, присутствующего на изображении.

Методы голосования предполагают, что точки выбросов или шума не будут голосовать последовательно за какую-либо отдельную модель, скорее их голоса будут распределяться более или менее равномерно по всем моделям. И «хорошая» модель получит больше голосов, потому что она получит голоса точек, хорошо описываемых моделью, в дополнение к случайным голосам точек шума.

Еще одним приятным свойством методов голосования является то, что они устойчивы к пропускам в данных. Если имеется достаточно точек данных для построения хорошей модели, отсутствующие данные не имеют значения.

1.3.7 Последовательный консенсус случайной выборки (RANSAC)

RANSAC по своей методике является методом голосования. Но, как мы видели, в первоначальной формулировке он подходит только для одной модели. Как мы можем изменить его, чтобы он обнаруживал и несколько экземпляров модели? Последовательный консенсус случайной выборки - отличный способ. Его идея очень проста. Мы будем использовать обычный консенсус случайной выборки для создания модели. Когда у нас появится первая модель, давайте удалим все точки данных, которые описываются этой моделью.

Как показано на экране, если консенсус случайной выборки обнаружил красную линию, давайте удалим все точки, описываемые этой прямой и начнем с нуля.

После удаления точек, соответствующих первой модели, второй запуск RANSAC найдет нам другую модель со следующим по количеству числом проголосовавших. Затем мы удалим точки данных, соответствующие этой модели, и снова запустим RANSAC.

Думаю, вы уловили идею. Удалив все точки данных, соответствующие уже обнаруженным моделям, мы используем RANSAC для поиска следующей модели. Мы можем продолжать до тех пор, пока количество голосов за лучшую модель не опустится ниже определенного порога, или до тех пор, когда больше не будет явного победителя, т.е. когда все кандидаты в модели получают примерно одинаковое количество голосов.

1.3.8 Преобразование Хафа

Другой очень популярный метод подбора модели, основанный на методе голосования, - это преобразование Хафа. Вероятно, это наиболее известный метод обнаружения прямых, и также его можно использовать для обнаружения других форм. Давайте посмотрим, как он работает.

В исходной формулировке каждая точка голосует за все возможные прямые, проходящие через нее, а затем прямые, соответствующие большим значениям счетчика или ячейки, исследуются на предмет того, являются ли они моделью.

Пусть (x_1, y_1) обозначает точку на плоскости xy. Рассмотрим общее уравнение прямой с угловым коэффициентом: $y = mx + b$. Бесконечно много прямых проходит через точку (x_1, y_1), но все они удовлетворяют уравнению $y_1 = mx_1 + b$ для различных значений m и b. Перепишем это уравнение в виде $b = -x_1m + y_1$ и рассмотрим mb-плоскость. Это пространство параметров - пространство всех возможных значений параметров m и b. Мы получили уравнение одной прямой в этом пространстве с фиксированными значениями (x_1, y_1).

Теперь добавим вторую точку на плоскости xy, (x_2, y_2). С ней также связана одна прямая в пространстве параметров, $b = -x_2m + y_2$. Эта прямая пересекает прямую, связанную с первой точкой (x_1, y_1) в некоторой точке (m_0, b_0) в пространстве параметров mb. m_0 - угловой коэффициент, а b_0 - точка пересечения с осью ординат прямой, содержащей обе точки (x_1, y_1) и $(x - 2, y_2)$ в плоскости xy. Итак, единственной точке (m_0, b_0) в пространстве параметров соответствует прямая $y = m_0x + b_0$. И все точки на этой прямой в плоскости xy будут иметь прямые в пространстве параметров, которые пересекаются в (m_0, b_0).

Теперь легко понять основной принцип преобразования Хафа. Разобьем пространство параметров mb на ячейки. Затем для каждой характерной точки на изображении проголосуем за каждую ячейку в пространстве параметров, которая могла бы сгенерировать эту точку. И, наконец, найдем ячейки, за которые было отдано больше всего голосов. Значения m и b, соответствующие этим ячейкам, определяют обнаруженные на исходном изображении прямые.

Представление пространства параметров Этот подход легко понять и реализовать, но есть пара проблем с mb-пространством. Во-первых, оно имеет неограниченные области параметров. Оба параметра, m и b, могут иметь любые значения от $-\infty$ до $+\infty$. Это затрудняет разделение пространства на ограниченное количество ячеек. Во-вторых, m, угловой коэффициент, приближается к бесконечности, когда прямая в xy-пространстве приближается к вертикальному направлению.

Один из способов обойти эти трудности - использовать нормальное (или полярное) представление прямой: $x\cos(\theta) + y\sin(\theta) = \rho$. На слайде представлена геометрическая интерпретация параметров ρ и θ. Вертикальная прямая имеет $\theta = 0\circ$, где ρ - это пересечение с осью x при x больше нуля. Точно так же горизовнтальная прямая имеет $\theta = 90\circ$, где ρ - это пересечению с осью y при y больше нуля, или $\theta = -90\circ$, где ρ - это пересечение с осью y, при y меньше нуля.

Преобразование Хафа в полярных координатах Подобно пространству параметров mb, мы можем рассмотреть пространство параметров $\theta\rho$. Каждая точка в плоскости xy будет иметь соответствующую синусоидальную кривую в плоскости $\theta\rho$, и если взять несколько точек, лежащих на одной прямой в пространстве xy, их соответствующие синусоидальные кривые в пространстве $\theta\rho$ будут пересекаться в одной точке, и координаты (θ, ρ) этого пересечения задают уравнние прямой в пространстве xy.

Чтобы преобразование Хафа было привлекательным с вычислительной точки зрения, мы подразделяем пространство параметров $\theta\rho$ на ячейки-аккумуляторы. В отличие от пространства mb, теперь у нас есть четко определенные ожидаемые диапазоны значений θ и ρ, $(\theta_{min}, \theta_{max})$ и (ρ_{min}, ρ_{max}). Используя $\theta_{min} = -90\circ$ и $\theta_{max} = 90\circ$, мы можем представить все возможные углы наклона прямых. А с помощью $-D \leq \rho \leq D$, где D - максимальное расстояние между противоположными углами изображения, мы можем найти все возможные прямые на изображении.

Алгоритм преобразования Хафа Обрисуем алгоритм. Сначала зададим все значения ячейки-аккумуляторов равными нулю. Затем для каждой

характерной точки (x_k, y_k) в плоскости xy изображения мы перебираем все допустимые значения θ на оси θ и ищем соответствующие ρ используя уравнение $\rho = x_k \cos(\theta) + y_k \sin(\theta)$. Затем полученные значения ρ округляются до ближайшего допустимого значения ячейки по оси ρ. Затем увеличьте значение аккумулятора ячейки (θ, ρ) на единицу. В конце процедуры значение K в ячейке (θ_i, ρ_j) означает, что K точек в плоскости xy лежат на прямой $x \cos(\theta_i) + y \sin(\theta_i) = \rho_j$. После того, как мы повторили эту оперцию для всех точек интереса, найдем аккумуляторы, соответствующие локальным максимумам. Если ячейка (θ_i, ρ_j) соответствует локальному максимуму, то обнаруженная на изображении прямая задается уравнением $x \cos(\theta_i) + y \sin(\theta_i) = \rho_j$.

Количество делений в плоскости $\theta\rho$ (или размер сетки) определяет точность коллинеарности точек. Можно показать, что количество вычислений в этом методе линейно относительно n, т.е. количества характерных точек на плоскости xy изображения.

Хотя в данном случае мы искали прямые на изображении в качестве примера применения преобразования Хафа, оно применимо к любой функции вида $g(v, c) = 0$, где v - вектор координат, а c - вектор коэффициентов. Существуют также улучшенные версии этого метода, когда градиентные ориентации краевых точек учитываются при переборе возможных значений $theta$. До эры глубокого обучения метод обобщенного преобразования Хафа был одним из популярных методов обнаружения объектов с помощью шаблонов.