1. Изображения как структура данных. Базовые операции над изображением. Свертка изображения

Изображения в целом разделяются на растровые и векторные. В данном случае рассматриваются только растровые изображения, загруженные полностью, то есть избавленные от сжатия.

Изображение как структура данных обычно состоит из двух частей:

1. Линейный массив, где последовательно записаны значения всех компонент последовательно для каждого пикселя. В случае цветового пространства RGB это три числа, представляющих собой значения интенсивности соответствующего канала для рассматриваемой точки.

2. Заголовок – небольшая структура постоянного размера. В этой структуре хранится информация о размерах изображения, о том, смещению на сколько байт в памяти соответствует смещение на один пиксель по каждому из направлений массива, а также ссылка на начало массива, соответствующее первому пикселю и служебные данные, необходимые для управления владением массивом.

За счет хранения в заголовке информации о размерах и сдвигах можно быстро получать произвольные срезы изображения (например, выбор произвольной прямоугольной области на изображении), в том числе разреженные (можно взять каждый второй пиксель по одной из координат и каждый третий по другой) или отраженные (поскольку сдвиг может быть отрицательным). Полученные структуры будут использовать тот же самый массив, что и исходное изображение, то есть будет скопирован только заголовок, но не сами данные. При этом сдвиги (stride) и размеры массива могут не соответствовать друг другу, то есть с точки зрения полученного среза вполне возможно, что описываемое изображение будет храниться в памяти с разрывами.

Опр. 1. Свертка. Пусть задано небольшое одноканальное небольшое изображение, называемое **ядром свертки**, размерность которого совпадает с размерностью изображения. Также у ядра свертки должен быть выделен один элемент, называемый **якорем**. Чаще всего используют ядра размера 3×3 с якорем в центре, такие, что сумма всех значений ядра равна 1 или 0. **Сверткой** изображения с ядром той же размерности называется операция преобразования изображения, при которой новое значение каждого пикселя вычисляется следующим образом:

1. Ядро совмещается с изображением таким образом, чтобы якорь ядра был совмещен с рассматриваемым пикселем.

2. Вычисляются все попарные произведения значений пикселей ядра и значений совмещенных с ними значений ядра. При этом существует несколько различных стратегий, какие значения следует принять для пикселей, выходящих за границы изображения: константные значения, отражение изображения относительно границы и т. д.

3. Сумма этих произведений считается значением соответствующего пикселя свертки.

2. Морфологические операции. Эрозия, дилатация, замыкание и размыкание: что это и для чего могут быть использованы.

Опр. 2. Морфологическая операция Пусть дано изображение I в оттенках серого и некоторый структурный элемент S – небольшое черно-белое изображение, на котором выделена некоторая начальная точка (как правило, в центре изображения). Тогда под **морфологической операцией** понимается преобразование I в выходное изображение B такого же размера, где значение каждой точки B_{ij} определяется по следующему правилу:

1. Структурный элемент совмещается с исходным изображением так, чтобы точка I_{ij} совпала с начальной точкой S;

2. Из исходного изображения выделяется набор точек, на которые накладываются белые точки структурного элемента;

3. B_{ij} определяется как некоторая заданная функция от значений выделенного набора точек (например, среднее, максимум/минимум)

Выделяют следующие стандартные операции:

Эрозия. В случае эрозии заданная функция – это минимум, поэтому эта операция также называется «оконным минимумом». Эта операция полезна для удаления небольших объектов, в том числе шумов (в предположении, что объекты светлее фона), однако она затирает части объектов вблизи границы. Обозначение: $B = I \ominus S$.

Дилатация. Заданная функция – максимум. В случае, если исходное изображение – бинарное, эта операция эквивалентна смазу, при котором в качестве point spread function используется S. Обозначение: $B = I \oplus S$.

Взятие разности $I - (I \oplus S)$ может использоваться для выделения на изображении границ.

Замыкание (closing). Замыкание – операция, которая задается как комбинация дилатации и эрозии (в таком порядке): $B = (I \oplus S) \ominus S$.

Размыкание (opening). Размыкание – операция, которая задается как комбинация эрозии и дилатации (в таком порядке): $B = (I \ominus S) \oplus S$. Эта операция используется для выделения темного фона: сначала при помощи эрозии удаляются шумы и маленькие объекты, а затем при помощи дилатации восстанавливаются удаленные границы. При помощи вычитания размыкания из исходного изображения можно получить объекты, очищенные от фона. Это полезно, например, для выделения текста на изображении с неравномерным освещением, что можно видеть на изображении 1.

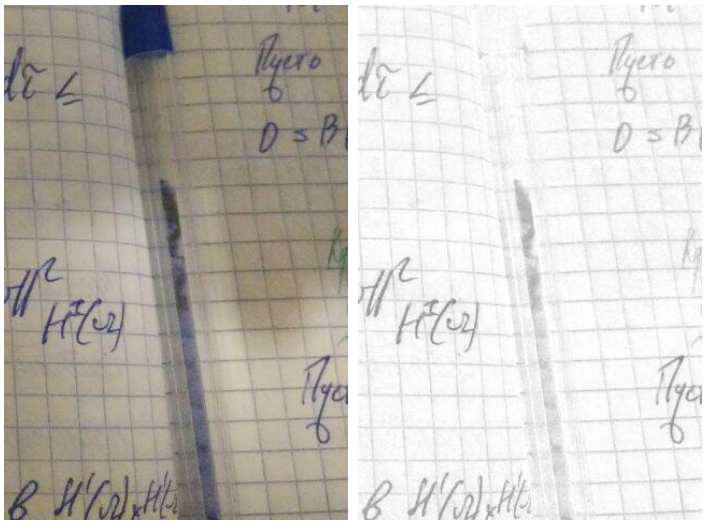

Рис. 1: Исходное изображение (слева) и разность между исходным изображением и размыканием (справа).

3. Фильтр границ Канни, для чего используется, какие параметры за что отвечают.

Фильтр границ Канни (Кэнни) – алгоритм, который среди всех пикселей изображения в градациях серого выделяет множество пикселей, которые образуют границы между объектами. Содержит следующие шаги:

1. Сглаживание изображения с целью устранения шума. Сглаживание выполняется путем сворачивания изображения с гауссовым ядром фиксированного размера: $I = H(\sigma) * I_0$. Слишком маленькие значения σ не смогут убрать шум, что приведет к множеству ложноположительных срабатываний, а слишком большие превратят все изображение в слабо меняющийся градиент и уничтожат все границы.

2. Вычисление градиента, то есть полей частных производных яркости по двум координатам. Как правило, используется разностная схема размера 3×3, то есть свертка с ядром оператора Собеля:

$$D_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad D_y = D_x^T;$$
$$G_x = D_x * I, \quad G_y = D_y * I.$$

   Однако этот шаг может быть объединен с предыдущим: для этого нужно сворачивать исходное изображение с дифференцированным гауссовым ядром, то есть с дискретными приближениями $\partial_x h(0, \sigma), \partial_y h(0, \sigma)$.

3. По полученным значениям вычисляется массив абсолютных значений градиента ($G = \sqrt{G_x^2 + G_y^2}$, поэлементно) и массив направлений. Все направления округляются до одного из основных: вертикаль, горизонталь или одна из двух диагоналей, при этом сторона (влево или вправо) значения не имеет.

4. Все пиксели помечаются как границы или как не-границы по следующим правилам в указанном порядке:

   (a) Если элемент не является локальным максимумом в направлении *своего* градиента, то элемент отмечается как не-граница. Например,

если округленное направление градиента в I_{ij} – вертикаль, то $G_{ij} \leqslant G_{i-1,j} \wedge G_{ij} \leqslant G_{i+1,j} \implies B_{ij} = 0$, где B – выходной булев массив того же размера, что и изображение.

(b) Если модуль градиента $G_{ij} < \theta_{\text{low}}$, то $B_{ij} = 0$.

(c) Если $G_{ij} > \theta_{\text{high}}$, то $B_{ij} = 1$.

(d) В противном случае, если пиксель является локальным максимумом, но его абсолютное значение лежит между двумя пороговыми значениями, то он считается границей, если хотя бы один из 8 соседних был определен как граница с использованием предыдущего правила.

Параметры $\theta_{\text{low}}, \theta_{\text{high}}$ регулируют количество ошибок обоих родов, но придать им какой-либо физический смысл довольно сложно.

4. Преобразование Радона. Дискретное преобразование Радона. Оценка сложности.

Опр. 3. Преобразование Радона Пусть $l_{\theta,s}$ – прямая, направляющий вектор которой направлен под углом θ ($\theta = 0$ соответствует горизональной прямой) и удаленная от начала координат на расстояние s. Тогда преобразованием Радона функции $f(x, y)$ называется интеграл этой функции по параметризованной прямой:

$$[\mathcal{R}f](\theta, s) = \int_{l_{\theta,s}} f(x, y)dl = \int_{\mathbb{R}^2} f(x, y)\delta(x\cos\theta + y\sin\theta - s)dxdy =$$
$$= \int_{\mathbb{R}} f(s\cos\theta + z\sin\theta, s\sin\theta - z\cos\theta)dz.$$

Рассмотрим теперь дискретную версию $f(x, y)$, заданную, например, как среднее по квадратной области:

$$\hat{f}(i, j) = \frac{1}{h^2}\int_{ih-\frac{h}{2}}^{ih+\frac{h}{2}}\int_{jh-\frac{h}{2}}^{jh+\frac{h}{2}} f(x, y)dydx.$$

Для определения дискретного преобразования Радона также необходим некоторый алгоритм дискретизации прямой $\Omega(s, \alpha)$, возвращающий множество координат пикселей $\left\{(i_k, j_k)_{k=1}^m\right\}$, приближающих непрерывную прямую с соответствующими параметрами.

Тогда **дискретное преобразование Радона** определяется следующим образом:

$$\left[\hat{\mathcal{R}}\hat{f}\right](\alpha, s) = \sum_{(i,j)\in\Omega(\alpha,s)} \hat{f}(i,j).$$

Однако такое определение обладает проблемой, связанной с неравномерностью приближения длины прямой количеством пикселей. Например, вертикальная непрерывная прямая, проходящая через центр квадрата со стороной N имеет длину пересечения N, а диагональная – $N\sqrt{2}$. Но обе дискретные версии будут иметь в пересечении с квадратом N пикселей.

Оценим сложность преобразования Радона наивным способом. В разумной параметризации прямых количество дискретных прямых, которые проходят через изображение, составляет $O(n^2)$. Нужно вычислить и сохранить для каждой из них сумму по содержащимся в ней пикселям, количество которых $O(n)$. Считая, что Ω также обладает не более чем линейной сложностью по n, получаем, что для вычисления сумм требуется $O(n^3)$ операций и $O(n^2)$ памяти.

В случае черно-белых (бинарных) преимущественно черных изображений возможно изменить алгоритм: достаточно перебрать все белые точки и для каждой из них прибавлять единицу к сумме по всем прямым, проходящим через эту точку. Таким образом сложность понижается до $O(Cn^2)$, где C – количество белых точек на изображении. Однако необходимо учесть, что для этой вариации требуется уметь находить прямые, проходящие через заданную точку не более чем за $O(n^2)$ операций, что может быть затруднительно для некоторых видов параметризации.

5. Виды параметризации прямых на изображении и их свойства. Повторное вычисление преобразования Хафа и связь этой процедуры с поиском точки схода.

Количество прямых на дискретном изображении размера $n \times n$ можно оценить как $\Theta(n^2)$, хотя точное количество зависит от выбранной параметризации. Обычно рассматривают следующие способы:

- (ρ, ϕ)-параметризация. Прямая, описанная такими параметрами, принадлежит к семейству параллельных прямых, направляющий вектор которых имеет угол ϕ, а расстояние от начала координат до этой прямой

равно ρ. Таким образом, ортогональная проекция начала координат на эту прямую в полярных координатах имеет координаты $\left(\phi + \frac{\pi}{2}, \rho\right)$.

Преобразование построенной непрерывной прямой в дискретную обычно делается при помощи алгоритма Брезенхема. Для преимущественно горизонтальных прямых ($\left|\frac{dy}{dx}\right| \leqslant 1$) в каждом столбце выбирается один пиксель, центр которого ближе всего к пересечению прямой с соответствующей вертикалью. Если прямая – преимущественно вертикальная, то используется аналогичная стратегия с заменой строк на столбцы и наоборот.

Такая параметризация ограничена, поскольку ϕ изменяется от 0 до π, а ρ – от $-\frac{n}{\sqrt{2}}$ до $+\frac{n}{\sqrt{2}}$ (или $[0, 2\pi)$ и $\left[0, \frac{n}{\sqrt{2}}\right]$ соответственно), но нелинейна, так как множество пар (ρ, ϕ), проходящих через заранее выбранную точку, образует не прямую, а синусоиду.

- (k, b)-параметризация. Описывает непрерывную прямую, заданную уравнением $y = kx + b$. В этой параметризации из рассмотрения исключаются вертикальные прямые. Параметризация линейна, так как при фиксированных (x, y) параметры k, b связаны линейным соотношением, но не ограничена, поскольку k будет принимать неограниченно большие значения для прямых, близких к вертикальным. Однако недостаток этой параметризации состоит в ее неравномерности: разница между $k = 1$ и $k = 2$ намного более значительна, чем между $k = 100$ и $k = 101$.

  Так же, как и в предыдущем пункте, дискретизация проводится при помощи алгоритма Брезенхема.

- (s, t)-параметризация. Эта параметризация учитывает только преимущественно вертикальные прямые с наклоном вправо ($\frac{dx}{dy} \in [0, 1]$). Прямая, соответствующая паре параметров (s, t), проходит через середины пикселей с координатами $(s, 0)$ и $(s + t, n - 1)$. Так как оба параметра изменяются от 0 до $n - 1$, второй пиксель в этой паре может выходить за пределы изображения, но это не должно быть проблемой, поскольку сетку координат всегда можно продлить. Дискретизация выполняется либо при помощи алгоритма Брезенхема, либо при помощи диадических паттернов.

Такая параметризация является ограниченной, что было отмечено раньше, и линейной, так как точка x, y принадлежит описываемой прямой, если выполняется следующее условие:

$$x = s \cdot \left(\frac{y}{n-1}\right) + (s+t) \cdot \left(1 - \frac{y}{n-1}\right) = s + t \cdot \frac{y}{n-1}, \tag{1}$$
$$s = \frac{y}{n-1}t - x.$$

Поскольку быстрое преобразование Хафа использует (s, t)-параметризацию, которая является линейной, то на Хаф-образе пучка прямых, проходящих через одну точку, заметна выделяющаяся прямая, то есть прямая с экстремально высокой суммой значений по содержащимся пикселям. Найдя параметры этой прямой, при помощи уравнения (1) можно определить точку пересечения пучка. Прямую с таким свойством можно найти, если еще раз применить к линограмме преобразование Хафа.

Следует отметить, что на настоящем изображении, на котором нужно выделить точку схода, после выделения границ будет множество шумовых линий, но все же большое количество линий проходит через одну точку, поэтому описанный алгоритм потенциально можно использовать. Рассмотрим условный искусственный пример.

На изображении 2 приведен пучок прямых, имитирующий линии разметки, проходящие через точку схода.

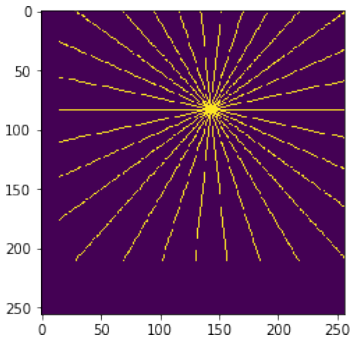

Рис. 2: Пучок прямых на изображении размера 256×256

На линограмме этого изображения 3 видно четыре яркие точки, соответствующие тем прямым, которые обладают нужным наклоном и могут быть определены при помощи БПХ. При этом эти точки лежат на одной прямой, поскольку соответствующие прямые пересекаются.

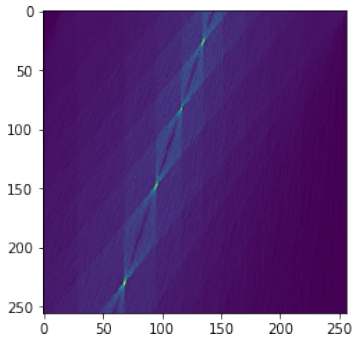

Рис. 3: Линограмма изображения 2

Поскольку эта прямая имеет неправильный наклон, применим преобразование Хафа к линограмме, предварительно отразив ее по вертикали. На полученной линограмме 4 заметна одна яркая точка, соответствующая целевой точке схода, и четыре прямых, соответствующих некоторым прямым из исходного пучка. Однако координаты точки не соответствуют исходным и требуют пересчета в соответствии с (1).

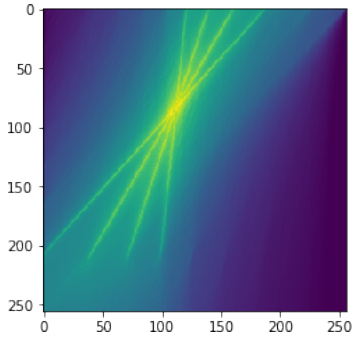

Рис. 4: Линограмма изображения 3

6. Преобразование Хафа и быстрое преобразование Хафа. Описание работы алгоритмов и их вычислительных характеристик.

В общем случае преобразование Хафа ставит в соответствие каждому патерну из заданного семейства сумму значений пикселей изображения, принад-

лежащих этому паттерну. В данном случае рассматривается преобразование Хафа для дискретных прямых на двумерном изображении, которое также может быть названо дискретным преобразованием Радона.

Быстрое преобразование Хафа – это вариация алгоритма, позволяющая понизить сложность алгоритма с $\Theta(n^3)$ до $\Theta(n^2 \log n)$ за счет использования диадических паттернов для приближения прямых.

Диадические паттерны описывают преимущественно вертиальные прямые с наклоном вправо, то есть в каждой строке $i = \text{const}$ содержат ровно один пиксель и $j(i)$ нестрого возрастает. Они задаются следующим образом:

1. Существует один паттерн высоты 1, состоящий из одного пикселя

2. Существует 2^n паттернов высоты 2^n (или порядка n), $n > 0$. Паттерн под номером i (нумерация начинается с нуля) состоит из двух состыкованных по вертикали паттернов высоты 2^{n-1} под номером $\left\lfloor \frac{i}{2} \right\rfloor$. Если i четно, то нижний пиксель верхнего паттерна располагается над верхним пикселем нижнего паттерна. Если i нечетно, то верхний паттерн дополнительно сдвигается на один пиксель вправо.

В силу построения диадические паттерны содержит множество пересечений. В частности, любой паттерн высоты больше 1 делит как верхнюю, так и нижнюю половину с другим паттерном той же высоты. Поэтому для вычисления сумм по всем n^2 паттернам некоторого порядка достаточно вычислить n^2 сумм по паттернам меньшего порядка, а не $2n^2$, как было бы для наивной реализации.

Рассмотрим алгоритм БПХ более подробно. Пусть имеется квадратное изображение со стороной $N = 2^n$. Изначальное изображение можно рассматривать как набор сумм по паттернам порядка 0. Чтобы получить в том же изображении набор сумм по паттернам порядка 1, проделаем следующую операцию:

- Разделим изображение на 2^{n-0-1} групп, каждая из которых содержит 2^{0+1} последовательных строк.

- Каждую группу разделим на верхнюю и нижнюю половины. Затем во все строки группы с номером $2i + r$ нужно записать поэлементную сумму i-х строк верхней и нижней половины, сместив строку из верхней половины вправо на $i + r$ элементов, где i меняется от 0 до 2^0 невключительно (то есть, на начальной итерации $i = 0$), а $r \in \{0, 1\}$ – остаток от деления на 2.

- В силу определения диадических паттернов, после предыдущего шага в i-й строке каждый группы записаны суммы по диадическим паттернам 1 порядка с номером i, причем все они начинаются с различных элементов нижней строки группы.

Так как полученное в конце условие очень похоже на изначальное (набор сумм по паттернам порядка 1), то неудивительно, что продолжная аналогичную операцию с соответственно увеличивающимися размерами групп после n шагов мы получим массив, содержащий суммы по паттернам высоты $2^n = N$, начинающимся с 0 строки изображения. Это и есть суммы по диадическим прямым определенного наклона. Для получения суммы по всем прямым, проходящим через изображение, следует воспользоваться БПХ для транспонированных/отраженных/... копий изображения и объединить полученные результаты.

Оценим сложность алгоритма быстрого преобразования Хафа. На каждой итерации значения каждого пикселя обновляются как сумма двух значений. При этом выполняется ровно n итераций. Следовательно, итоговая сложность составляет $\Theta(N^2 \cdot n) = \Theta(N^2 \log N)$, где N – сторона изображения.

7. Трехмерное быстрое преобразование Хафа для плоскостей. Параметризация, описание работы, вычислительная сложность.

Трехмерное быстрое преобразование Хафа для плоскостей – это алгоритм, позволяющий быстро подсчитать сумму по всем плоскостям с определенным наклоном в изображении размера $N \times N \times N$. При этом плоскость рассматривается как объект, составленный из диадических паттернов и проходящий через точки со следующими координатами (для удобства приведены четыре точки, хотя для задания плоскости достаточно любых трех):

$$(s, 0, 0), \quad (s + t_1, 0, N - 1), \quad (s + t_2, N - 1, 0), \quad (s + t_1 + t_2, N - 1, N - 1), \tag{2}$$

$$0 \leqslant s \leqslant N - 1, 0 \leqslant t_1 \leqslant N - 1, 0 \leqslant t_2 \leqslant N - 1.$$

При помощи наивного алгоритма Хафа можно вычислить суммы по N^3 плоскостей, содержащих $O(N^2)$ вокселей, за $O(N^5)$ операций. Как и в двумерном случае, эту асимптотику можно улучшить.

Рассмотрим плоскость, образованную диадическими паттернами в соответствии с (2). В каждом сечении $z = \text{const}$ образ этой плоскости представляет собой диадический паттерн с $\hat{t} = t_2, \hat{s} = s + I_{t_1}(z)$, где $I_t(k)$ – смещение вокселя в строке k в паттерне со склонением t. Нам требуется вычислить суммы по всем вокселям в объединении этих паттернов. В качестве первого шага можно вычислить суммы по каждому из этих диадических паттернов в отдельности.

Если в исходном изображении применить к каждому массиву в сечении по оси Z двумерное быстрое преобразование Хафа, то смысл осей изменится: теперь координаты в массиве будут представлять не (x, y, z), а $(\hat{s}, \hat{t}, z)$. Таким образом, искомая сумма преобразуется в сумму по вокселям $\left\{\left(s + I_{t_1}(z), t_2, z\right)_{z=1}^{N-1}\right\}$, что в свою очередь является диадическим паттерном в плоскости $\hat{t} = \text{const}$.

Применив к каждому массиву в сечении по оси $\hat{T}$ двумерное быстрое преобразование Хафа, в точке (s, t_2, t_1) получаем искомую сумму.

Поскольку в ходе работы алгоритма мы вычислили $2N$ двумерных БПХ, общая сложность алгоритма составляет $\Theta\left(N^3 \log N\right)$. Можно сказать, что за счет переиспользования вычисленных сумм сумма по каждой плоскости вычисляется за $O(\log N)$ в среднем.

8. Трехмерное быстрое преобразование Хафа для прямых. Параметризация, описание работы, вычислительная сложность.

Трехемерное быстрое преобразование Хафа для прямых позволяет вычислять сумму по дискретным прямым в изображении размера $N \times N \times N$ вокселей. Рассматриваемые прямые – преимущественно ориентированные вдоль оси z. Прямая параметризуется 4 параметрами и проходит через следующие воксели на верхней и нижней гранях куба:

$$(s_1, s_2, 0), \quad (s_1 + t_1, s_2 + t_2, 0).$$

При этом считается, что все параметры могут меняться от 0 до $N-1$. Заметим, что проекции прямой на плоскости Oxz, Oyz представляют собой плоские диадические паттерны (в основном потому что именно так определяется трехмерная диадическая прямая).

Следовательно, для начала необходимо преобразовать изображение в четырехмерное. Исходное изображение располагается в пространстве с нулевой

координатой по 4 измерению, все остальные пространства заполняются нулевыми значениями. По аналогии с двумерным преобразованием, основная стратегия состоит в получении на каждом шаге отдельных «слоев», значения в каждом из которых описывают суммы по всем возможным паттернам данной высоты 2^k, начинающихся с низа данного слоя. Но если в двумерном случае каждая группа должна была иметь размер $2^k \times N$, то для данного случая потребуются группы $N \times N \times 2^k \times 2^k$. При этом первые две координаты практически не меняют своего смысла – они соответствуют s_1, s_2, то есть описывают, в какой точке x, y слоя `group[:, :, 0, 0]` (или, что то же самое, `image[:, :, 2**k * groupid, 0]`) начинается паттерн, описываемый вокселем. Третья координата z – это ось, вдоль которой распределяются и объединяются попарно группы. Четвертое измерение необходимо, чтобы было куда записывать получающиеся значения; при этом на k-й итерации используются только первые 2^k пространств, а остальные заведомо заполнены нулевыми значениями.

На каждом шаге две соседние группы объединяются в одну, и в двумерный срез с координатами `group[:, :, 2*i1+r1, 2*i2+r2]` записывается поэлементная сумма `group[:, :, i1, i2]` и `group[:, :, 2**k+i1, 2**k+i2]`, причем второй массив предварительно сдвигается на $(i_1 + r_1, i_2 + r_2)$ элементов по последним координатам. Например, после первой итерации группы будут иметь размер $N \times N \times 2 \times 2$, причем две последние координаты описывают тип паттерна (вертикальный, диагональный по одной координате, диагональный по другой координате или диагональный по обеим координатам).

На каждой итерации обрабатывается $\dfrac{N}{2^{k+1}}$ групп ($k = 0, 1, \ldots, \log_2 N - 1$). В каждой группе для всех значений i_1, i_2, r_1, r_2, то есть $2^k \cdot 2^k \cdot 2 \cdot 2 = 2^{2k+2}$ раз суммируются двумерные массивы из N^2 элементов. Таким образом, сложность алгоритма составляет

$$\Theta\left(\sum_{k=1}^{\log_2 N-1} \frac{N}{2^{k+1}} \cdot 2^{2k+2} N^2\right) = \Theta\left(2N^3 + 4N^3 + 8N^3 + \cdots + N^4\right) = \Theta\left(N^4\right). \quad (3)$$

Наивная реализация потребовала бы $O(N^5)$ операций. Можно сказать, что трехмерное БПХ для прямых вычисляет сумму по каждой прямой в среднем за константное время.

9. История развития томографии. Строение томографа.

10. Взаимодействие рентгеновского излучения с веществом. Сведение зарегистрированных данных к виду преобразования Радона.

Рассмотрим монохроматический пучок излучения (в данном случае – рентгеновского), идущего от далекого точечного источника и проходящего через однородный слой вещества толщиной D.

Утв. 1. Закон Бугера-Ламберта-Бера Интенсивность излучения после прохождения вещества описывается следующей формулой:

$$I = I_0 e^{-\mu_0 D},$$

где I_0 – интенсивность излучения перед слоем, μ_0 – линейный коэффициент поглощения, зависящий от вещества и от длины волны. Пренебрегая рассеянием, принимается, что для вакуума $\mu_0 = 0$.

Этот закон можно обобщить на случай неоднородного слоя следующим образом:

$$I = I_0 e^{-\int_0^D \mu(x)dx}$$

Рассмотрим помещенный в работающий по параллельной схеме томограф объект, обладающий неизвестной функцией поглощения $\mu(x, y)$. Тогда непосредственно регистрируемые томографом данные задаются следующими формулами:

$$I(\theta, s) = I_0 e^{-\int_{l_{\theta,s}} \mu(x,y)dl} = I_0 \exp\left(-\int_{\mathbb{R}^2} \mu(x, y)\delta(x\cos\theta + y\sin\theta - s)dxdy\right), \quad (4)$$

где I_0 – результат, получаемый на детекторе в отсутствие объекта.

При помощи элементарных преобразований данные из (4) сводятся к Радон-образу функции поглощения:

$$\ln\frac{I_0}{I(\theta, s)} = -\ln\frac{I(\theta, s)}{I_0} = \int_{\mathbb{R}^2} \mu(x, y)\delta(x\cos\theta + y\sin\theta - s)dxdy = [\mathcal{R}\mu](\theta, s).$$

Таким образом, взяв логарифм от ослабления сигнала и применив обратное преобразование, можно вычислить функцию поглощения.

11. Преобразование Радона. Синограмма.

Опр. 4. Преобразование Радона Пусть $l_{\theta,s}$ – прямая, направляющий вектор которой направлен под углом θ ($\theta = 0$ соответствует горизональной прямой) и удаленная от начала координат на расстояние s. Тогда преобразованием Радона функции $f(x, y)$ называется интеграл этой функции по параметризованной прямой:

$$[\mathcal{R}f](\theta, s) = \int_{l_{\theta,s}} f(x, y)dl = \int_{\mathbb{R}^2} f(x, y)\delta(x\cos\theta + y\sin\theta - s)dxdy =$$
$$= \int_{\mathbb{R}} f(s\cos\theta + z\sin\theta, s\sin\theta - z\cos\theta)dz.$$

Каждая точка Радон-образа функции представляет собой "сумму" по прямой с определенными параметрами. Как правило, по обоим параметрам рассматривается равномерная дискретная сетка значений, где θ меняется от 0 до π, s – от 0 до некоторого максимального значения, соответствующего размеру сцены. Полученный массив значений называется синограммой, так как Радон-образом точечной функции является синусоида.

12. Теорема о центральном сечении.

Преобразование Фурье функций от одной и от двух переменных задаются следующими формулами:

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t)e^{-2\pi i(\omega t)}dt, \tag{5}$$

$$\hat{F}(u, v) = \int_{\mathbb{R}^2} f(x, y)e^{-2\pi i(xu+yv)}dudv. \tag{6}$$

Также введем сокращенное обозначение для преобразования Радона, считая θ фиксированным параметром, а s – переменной:

$$p_\theta(s) := [\mathcal{R}f](\theta, s).$$

Теор. 1. *О центральном сечении. Преобразование Фурье от $p_\theta(s)$ совпадает со значениями двумерного преобразования Фурье от $f(x, y)$ на некоторой прямой:*

$$\hat{p}_\theta(\omega) = F(\omega\cos\theta, \omega\sin\theta)$$

□. Преобразуем определение преобразований Фурье и Радона, используя основное свойство дельта-функции (а также $\delta(t) = \delta(-t)$):

$$\widehat{p}_\theta(\omega) = \int_{\mathbb{R}} p_\theta(s) e^{-2\pi i \omega s} ds =$$

$$= \int_{\mathbb{R}^3} f(x,y) \cdot \delta(x\cos\theta + y\sin\theta - s) e^{-2\pi i \omega s} dx dy ds = \int_{\mathbb{R}^2} f(x,y) e^{-2\pi i \omega (x\cos\theta + y\sin\theta)} dx dy;$$

$$F(u = \omega\cos\theta, v = \omega\sin\theta) = \int_{\mathbb{R}^2} f(x,y) e^{-2\pi i (xu+yv)} \Big|_{\substack{u=\omega\cos\theta \\ v=\omega\sin\theta}} = \widehat{p}_\theta(s).$$

■

Таким образом, прямая, вырезанная из двумерного Фурье-образа исходной функции, проходящая через начало координат, фактически описывает интегралы этой функции вдоль всех прямых, параллельных вырезанной. Получить их можно при помощи обратного к (5) преобразования.

13. Алгоритм обратного проецирования (BP).

Рассмотрим задачу восстановления исходной функции по ее Радон-образу. Если рассматривать модель непрерывного мира, то $f(x,y)$ может быть восстановлена при помощи **обратного проецирования** (**Back-projection**):

$$\left[\mathcal{B}(p_\theta(s))\right](x,y) = \int_0^\pi p_\theta\left(x\cos\theta + y\sin\theta\right) d\theta. \tag{7}$$

Выражение (7) не совпадает с $f(x,y)$, однако часто используется как его приближение.

$$f(x,y) = \mathcal{F}_2^{-1}\left[F(u,v)\right](x,y) = \int_{\mathbb{R}^2} F(u,v) e^{2\pi i (xu+yv)} du dv = \ldots$$

Перейдем в последнем равенстве к полярным координатам по переменным интегрирования: $u = \omega\cos\theta, v = \omega\sin\theta, dudv = |\omega| d\omega d\theta$:

$$\cdots = \int_0^\pi \int_{-\infty}^{\infty} F(\omega\cos\theta, \omega\sin\theta) e^{2\pi i (x\cos\theta + y\sin\theta)\omega} |\omega| d\omega d\theta = \ldots$$

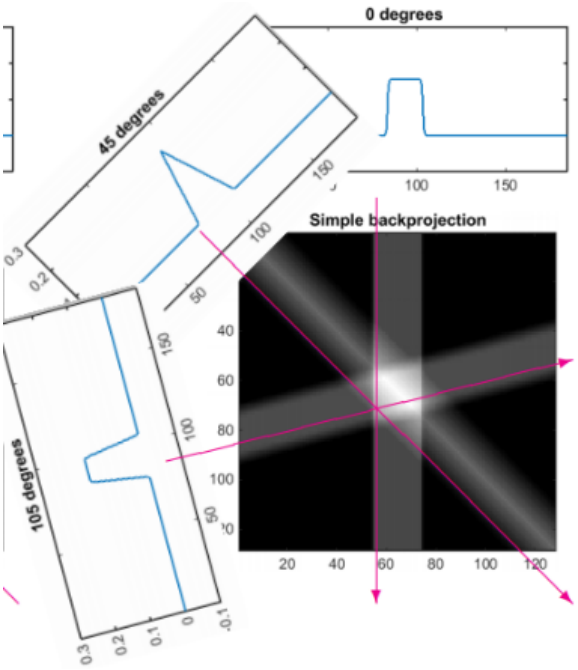

Рис. 5: Суть backprojection в одной картинке

Используем теорему 1 для замены $F(\omega\cos\theta, \omega\sin\theta) = \hat{p}_\theta(\omega)$. Также введем для краткости обозначение $s = x\cos\theta + y\sin\theta$.

$$\int_0^\pi \int_{\mathbb{R}} |\omega| \hat{p}_\theta(\omega) e^{2\pi i s\omega} d\omega d\theta = \int_0^\pi \left[\mathcal{F}^{-1}\left(|\omega|\hat{p}_\theta(\omega)\right)\right](s) d\theta. \tag{8}$$

Если удалить из последнего выражения якобиан $|\omega|$, оставив под интегралом вместо выражения $\mathcal{F}^{-1}\left[\hat{p}_\theta(\omega)\right](s) = p_\theta(s) = p_\theta\left(x\cos\theta + y\sin\theta\right)$ то получится в точности выражение (7).

14. Алгоритм FBP.

Как было показано ранее, алгоритм обратного проецирования легко описать и реализовать, но он не является вполне точным с математической точки зрения, поскольку опускает якобиан $|\omega|$. В результате этого, как правило, восстановленное изображение получается размытым, значения, близкие к началу координат, завышены, а далекие от начала координат, наоборот, занижены. Причина этого в том, что после дискретизации с равномерной сеткой по s и

θ через пиксели, близкие к началу координат, проходит большее количество дискретных прямых.

Алгоритм **filtered backprojection** состоит в использовании вместо (7) более правильной формулы, выведенной в (8):

$$\left[\mathcal{B}_f(p_\theta(s))\right](x,y) = \mathcal{F}^{-1}\left[|\omega|\hat{p}_\theta(\omega)\right](x\cos\theta + y\sin\theta)\,d\theta. \tag{9}$$

С точки зрения реализации следует добавить в алгоритм обратного проецирования дополнительный шаг: для каждого угла θ_i заменить «сырые» значения синограммы $p_{\theta_i}(s)$ на $\mathcal{F}^{-1}\left[|\omega|\hat{p}_{\theta_i}(\omega)\right](s)$. Множитель $|\omega|$ называется **Ramp filter**.

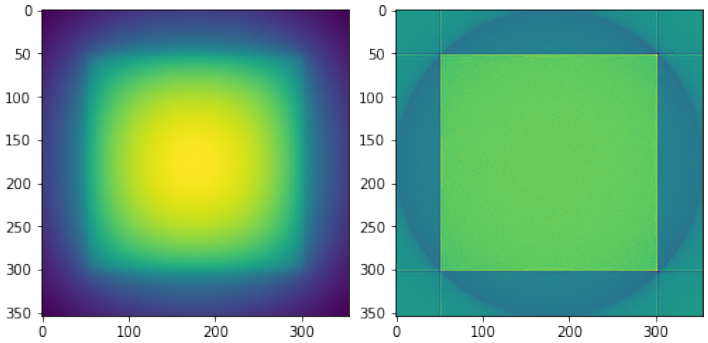

Рис. 6: Восстановление белого квадрата из синограммы при помощи backprojection (слева) и filtered backprojection (справа).

15. Способ использования БПХ для определения наклона шрифта.

Быстрое преобразование Хафа может быть использовано для определения наклона шрифта. Рассмотрим следующее изображение:

Выделим на изображении границы путем вычитания из изображения его дилатации. Также здесь изображение было отражено относительно вертикальной оси, поскольку шрифт, очевидно, наклонен вправо, а стандартная версия быстрого преобразования Хафа работает с прямыми, наклоненными влево с точки стандартной системы координат изображения.

Применим к полученному изображению БПХ:

В полученном массиве каждая точка описывает сумму значений по некоторой дискретной прямой, причем i-я строка соответствует семейству прямых

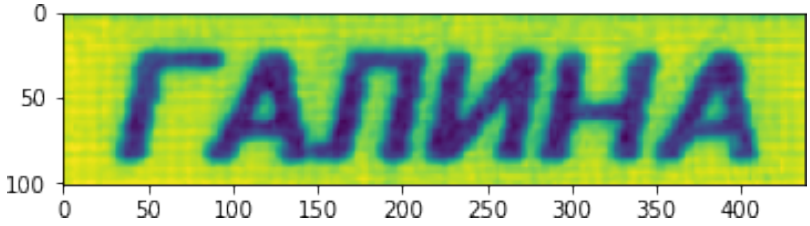

Рис. 7: Исходное изображение

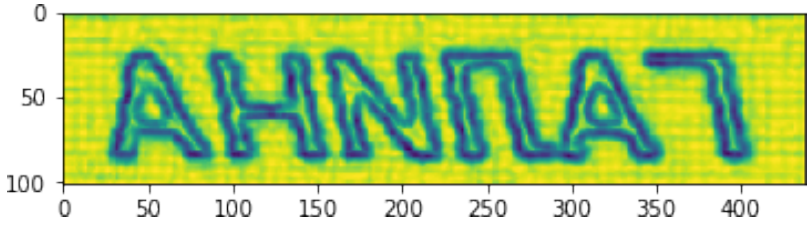

Рис. 8: Выделенные границы на изображении

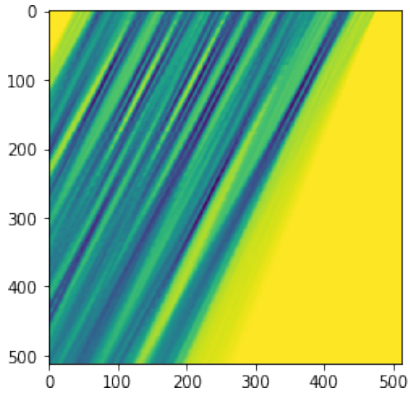

Рис. 9: Результат применения БПХ к изображению

с определенным наклоном $\theta_i = \arctan\left(\frac{n-1}{i}\right)$. Ясно, что среди всех таких семейств то, которое накладывается на особенности шрифта, будет иметь наибольшую изменчивость: от 0 в промежутках между буквами до высоты строки в случае наложения на «вертикальный» элемент буквы. Следовательно, нужно выбрать строку, значения в которой обладают наибольшей дисперсией (или наибольшим стандартным отклонением):

В данном случае максимальной дисперсией обладает строка $i = 158$, опи-

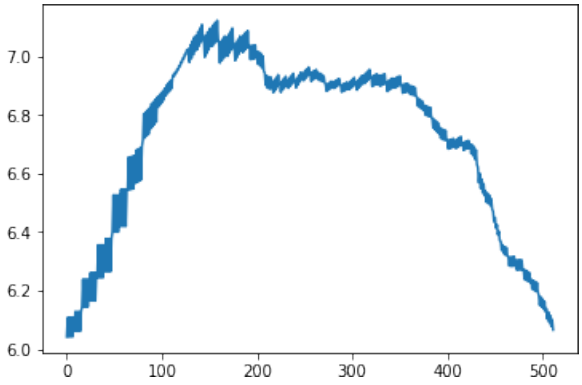

Рис. 10: График стандартных отклонений строк

сывающая прямые с наклоном $\theta_{158} = \arctan\left(\frac{511}{158}\right) \approx 73^\circ$.

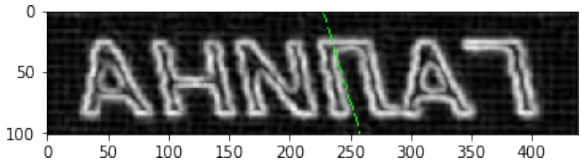

Рис. 11: Изображение со случайно выбранной прямой из найденного семейства

16. Способ использования БПХ для слепой компенсации радиальной дисторсии.

Радиальная дисторсия – это класс искажений изображения на фотографии, связанный с оптической системой камеры, особенно характерное для камер с широкоугольным объективом. При этом искажении прямые линии на изображении искривляются по следующим формулам:

$$\begin{cases} x' = x \cdot f(r), \\ y' = y \cdot f(r), \\ f(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots, \\ r = \sqrt{x^2 + y^2}, \end{cases} \tag{10}$$

где (x, y) – координаты точки без дисторсии, (x', y') – наблюдаемые координаты точки в результате действия дисторсии. Значения параметров k_i на практи-

ке достаточно быстро убывают, поэтому не приведенные в (10) члены обычно отбрасываются. Пример искажения показан на изображении 12.

Рис. 12: Пример фотографии с радиальной дисторсией

Далее будем считать, что при $i > 3$ $k_i = 0$. Если параметры k_1, k_2, k_3 известны, то построение изображения, преобразование которого совпадает с полученной искаженной фотографией, не вызывает проблем. Необходимо создать изображение I', заполненное нулями, и для каждого пикселя искаженного изображения I_d выполняется следующая операция:

$$I'(x(x_d, y_d), y(x_d, y_d)) += I_d(x_d, y_d).$$

Преимущество такого преобразования состоит в том, что несмотря на то, что длины кривых не сохраняются, интегральная яркость вдоль каждой кривой остается той же самой. Но для такой реконструкции требуется уметь находить обратную функцию $f^{-1}(r)$ к (10).

Предположим, что оригинальная фотография без искажений содержит множество прямых линий и нужно восстановить эту фотографию по искаженной версии. В таком случае можно перебором в некотором пространстве параметров найти параметры, при которых некоторая метрика, описывающая «прямолинейность» изображения, достигает максимума. Эта целевая функция вычисляется при помощи быстрого преобразования Хафа.

Качество восстановления изображения с выбранными параметрами оценивается следующим образом:

1. С использованием численных методов находится функция $f^{-1}(r)$. При этом функция $f(r)$ должна быть монотонной на определенном отрезке, в противном случае предложенные параметры не рассматриваются.

2. На изображении выделяются границы при помощи оценки градиента яркости.

3. Стоится восстановленние изображение, соответствующее оцениваемым параметрам дисторсии.

4. Стоится преобразование Хафа для данного изображения. При этом размер матрицы увеличивается вчетверо, поскольку вычисляются суммы по прямым всех ориентаций.

5. Из Хаф-образа вычитается этот же образ, сглаженный Гауссовым фильтром; отрицательные значения заменяются на 0. Полученная матрица обозначается как H.

6. Вычисляется вектор F, содержащий дисперсии значений матрицы H по строкам. Каждый элемент F_i содержит дисперсию значений, соответствующих прямым с одинаковой тангентой $t = t_i$.

7. Пусть полученный вектор F имеет K уровней значений, частоты появления которых $P(F_k)$. Тогда в качестве целевой функции используется энтропия F:
$$E\left(F\right) = -\sum_{k=1}^{K} P(F_k) \log P(F_k) \to \min$$

17. Способ использования БПХ для определения степени сбития камеры. Эпиполярная геометрия.

18. Быстрое вычисление суммы по любому отрезку и четырехвершиннику на изображении с помощью БПХ.

Рассмотрим изображение размера $N \times N, N = 2^n$ в оттенках серого. Известно, что для любой пары пикселей существует по крайней мере одна содержащая их диадическая прямая. При этом все такие прямые приближают отрезок между ними одинаковым образом, поэтому задача вычисления суммы по отрезку, заданному двумя концами при помощи преобразования Хафа как

минимум корректно поставлена. Для ее решения требуется научиться находить параметры какой-либо диадической прямой, проходящей через эту пару точек и вычислять сумму по любому отрезку известной прямой.

Для определенности будем рассматривать только преимущественно горизонтальные отрезки с положительным наклоном, то есть такие, что $y_2 \geqslant y_1, x_2 \geqslant x_1, \Delta y \leqslant \Delta x$, где (x_1, y_1) и (x_2, y_2) – координаты концов отрезка.

Определение диадического паттерна. Известно, что любой диадический паттерн $y = I_p(x), p \in \{0, 1, \ldots, N-1\}$ может быть представлен в виде суммы некоторого подмножества множества $D = \{I_1(x), I_2(x), I_4(x), \ldots, I_N(x)\}$, где под суммой паттернов понимается паттерн, описываемый уравнением $y = I_{p_1}(x) + I_{p_2}(x)$. Отметим, что нам достаточно подобрать любой паттерн, удовлетворяющий условию $I_p(x_2) - I_p(x_1) = y_2 - y_1$. Для этого применим жадный алгоритм: выберем в качестве начального приближения горизонтальный паттерн $I_0(x)$ и на каждом шаге будем прибавлять самый «большой» паттерн из множества D, который не приводит к переполнению, то есть при котором $I_p(x_2) - I_p(x_1) \leqslant y_2 - y_1$. Данный алгоритм всегда приводит к решению, следовательно, после $\Theta(\log N)$ операций мы получим параметр t, гарантирующий, что если паттерн проходит через первую точку, то он проходит и через вторую. Параметр s подбирается очевидным способом так, чтобы паттерн действительно проходил через первую точку

Сумма по диадическому отрезку. Пусть задана диадическая прямая $y = s + I_t(x)$ и требуется вычислить сумму по всем точкам этой прямой на отрезке $0 \leqslant x_1 \leqslant x < x_2 \leqslant N$.

Заметим, что для некоторых пар (x_1, x_2) эта задача уже решается в процессе быстрого преобразования Хафа. А именно, на k-й итерации алгоритма в соответствующем массиве сохраняются такие суммы для всех $x_1 = m \cdot 2^k, x_2 = (m+1) \cdot 2^k$ по всем прямым. Несложно модифицировать алгоритм так, чтобы эти суммы были доступны после окончания вычислений, а не перезаписывались, для чего достаточно использовать для вычислений трехмерный массив. Это приведет к увеличению требований по памяти до $\Theta(N^2 \log N)$. При помощи этих значений сумма по любому отрезку с $x_1 = 0$ может быть вычислена при помощи жадного алгоритма, похожего на использованный в предыдущем пункте. Для произвольных значений (x_1, x_2) следует вычислить сумму по двум отрезкам и взять разность. Итоговая сложность алгоритма составляет $\Theta(N^2 \log N)$ операций на предподсчет (БПХ), после чего сумма по каждому отрезку вычисляется за $O(\log N)$ операций.

Рассмотрим, как эта же методика может быть применена для вычисления сумм по произвольному выпуклому четырехвершиннику. Для этого применим к каждому столбцу кумулятивное суммирование так, чтобы каждый пиксель хранил сумму всех пикселей исходного изображения, расположенных в том же столбце не выше, чем он сам. Это преобразование выполняется линейным проходом по каждому столбцу за $\Theta\left(N^2\right)$ операций. После такого преобразования сумма по произвольному отрезку фактически является суммой значений в исходном изображении по прямоугольной трапеции, одна из сторон которой обязательно находится на нижней строке изображения. Любой выпуклый четырехвершинник может быть представлен в виде суммы или разности не более чем четырех таких трапеций, следовательно, задача сводится к предыдущей и обладает такой же асимптотической сложностью: $\Theta\left(N^2 + N^2 \log N\right) = \Theta\left(N^2 \log N\right)$ операций на предподсчет и $O\left(4 \log N\right) = O\left(\log N\right)$ операций на вычисление суммы по четырехвершиннику.

19. Сочетание БПХ и принципа четырех русских для случаях прямых в трехмерном пространстве.

Как было показано в (3), сложность быстрого трехмерного преобразования Хафа для прямых составляет $\Theta(N^4)$ и вычисляет суммы по всем диадическим дискретным прямым. Но если требуется найти сумму не по всем прямым, а по $O(N^3)$ выбранных, асимптотическая сложность наивного суммирования совпадает со сложностью ТБПХ.

Алгоритм четырех русских – это способ ускорить вычисление сумм по $O(N^3)$ прямых за счет совмещения обоих описанных алгоритмов. В нем предлагается выполнить не все $n = \log_2 N$ шагов ТБПХ, а только x первых итераций (способ выбора x описан ниже), после чего для каждой из прямых непосредственно просуммировать $\dfrac{N}{2^x}$ значений, каждое из которых хранит сумму по части прямой высоты 2^x. Оценим сложность обоих шагов алгоритма:

1. Сложность выполнения первых x итераций ТБПХ:
$$\Theta\left(2N^3 + 4N^3 + \cdots + 2^x N^3\right) = \Theta\left(N^3 2^x\right).$$
2. Сложность суммирования по прямым:
$$O(N^3) \cdot \Theta\left(\frac{N}{2^x}\right) = O\left(N^4 2^{-x}\right)$$

Очевидно, что асимптотический минимум получается при $2^x = \sqrt{N}$, поскольку в противном случае одно из слагаемых будет иметь худшую асимптотику. Таким образом, необходимо сделать $\frac{1}{2}\log_2 N$ шагов алгоритма ТБПХ, после чего перейти к суммированию по прямым, и итоговая сложность алгоритма составит $O\left(N^{3.5}\right)$. Необходимо отметить, что рассуждения приводятся для случая, когда количество прямых составляет $\Theta(N^3)$, в противном случае оптимальное количество шагов и итоговая сложность могут оказаться другими.

20. Быстрая линейная бинарная кластеризация с помощью БПХ.

Рассмотрим некоторое изображение в градациях серого с темным фоном, светлые точки которого нужно разделить одной гиперплоскостью на два кластера наиболее «естественным» способом. Для простоты будем считать, что изображение двумерно и имеет размер $n \times n$, хотя данный алгоритм потенциально может быть использован для изображений любой размерности, в таком случае необходимо найти разделяющую прямую.

Будем обозначать множества, на которые прямая разделяет множество всех точек изображения Ω как A, B. Считая, что яркость точки задается как функция от координат $P\left(\vec{x}\right)$, введем следующие характеристики множеств:

$$\omega(X) = D^0(X) = \sum_{\vec{x}\in X} P\left(\vec{x}\right),$$

$$D^1_i(X) = \sum_{\vec{x}\in X} x_i P\left(\vec{x}\right) = \left(\sum_{\vec{x}\in X} \vec{x} P\left(\vec{x}\right)\right)_i,$$

$$D^2_{ij}(X) = \sum_{\vec{x}\in X} x_i x_j P\left(\vec{x}\right) = \left(\sum_{\vec{x}\in X} \vec{x}\otimes\vec{x} P\left(\vec{x}\right)\right)_{ij}.$$

Следует отметить, что, во-первых, все приведенные статистики являются аддитивными ($D^p\left(X\cup Y\right) = D^p(X) + D^p(Y)$ для любых непересекающихся X, Y), а во-вторых, матрица D^2 симметрична, что снижает количество элементов, которые нужно вычислять.

В качестве критерия оптимизации предлагается выбрать одну из следующих целевых функций, которые могут быть выражены через аддитивные

статистики:

$$\omega(A)\operatorname{tr}\sigma(A) + \omega(B)\operatorname{tr}\sigma(B) \to \min_{A,B},$$
$$\omega(A)\lambda_2(A) + \omega(B)\lambda_2(B) \to \min_{A,B},$$

где $\sigma(A)$ – внутриклассовая матрица ковариации, λ_2 – ее второе собственное значение.

Поскольку данные функционалы невыпуклые, единственным надежным способом выбора прямой является полный перебор всех прямых из некоторого ограниченого семейства. Для эффективного вычисления массива значений функционала нужно вычислить 6 (для двумерного изображения; в общем случае по количеству независимых значений среди всех D^p для произвольного фиксированного множества) массивов значений аддитивных статистик, вычисленных для всех полуплоскостей. Эта операция может быть выполнена при помощи быстрого преобразования Хафа.

Обозначим вычисляемую статистику как $T(X) \in \{D^0, D^1_1, D^1_2, D^2_{11}, D^2_{12}, D^2_{22}\}$. За $O(n^2)$ операций можно вычислить массив того же размера, что и изображение, где каждый элемент равен значению T на множестве из одного соответствующего пикселя.

Выполним к данному массиву операцию кумулятивного суммирования вдоль оси Oy. После этого каждый элемент по определению операции содержит сумму значений T по всем пикселям непосредственно под ним (включая сам пиксель), а сумма по полуплоскости, лежащей под прямой, заданной параметрами (s,t) теперь может быть вычислена как сумма всех элементов этой прямой. Применив к массиву быстрое двумерное преобразование Хафа, получим массив, где элемент I_{st} содержит статистику T, вычисленную по полуплоскости, лежащей ниже прямой с соответствующими параметрами. Статистика по второй полуплоскости может быть вычислена при помощи вычитания найденного значения из $T(\Omega) = \text{const}$. После вычисления всех статистик при помощи поэлементных операций над двумерными массивами вычисляются значения целевых функций для каждой прямой. Среди полученных значений находится минимум. Прямая, соответствующая положению этого минимума, является оптимальной разделяющей прямой.

Итоговая сложность алгоритма составляет $O(mn^2 \log n)$, где m – количество нужных статистик (для описанных критериев $m = 6 = \text{const}$), так как именно столько операций требуется на вычисление всех линограмм, а все остальные операции выполняются за $O(n^2)$.

21. Робастное решение задачи линейной регрессии путем вычисления М-оценок с помощью БПХ.

В практических приложениях линейной регрессии параметры прямой $y = kx + b$ оцениваются по известному набору значений $\left\{(x_i, y_i = kx_i + b + \varepsilon_i)_{i=1}^{n}\right\}$ путем минимизации следующей целевой функции:

$$\sum_{i=1}^{n} (y_i - (kx_i + b))^2 \to \min_{k,b}, \tag{11}$$

для чего существует хорошо описанное аналитическое решение. Однако модель (11) неустойчива к выбросам: если какая-то доля наблюдений не подчиняется модели $y_i = kx_i + b + \varepsilon_i$, то МНК выдает сильно искаженные результаты, как на рис. 13.

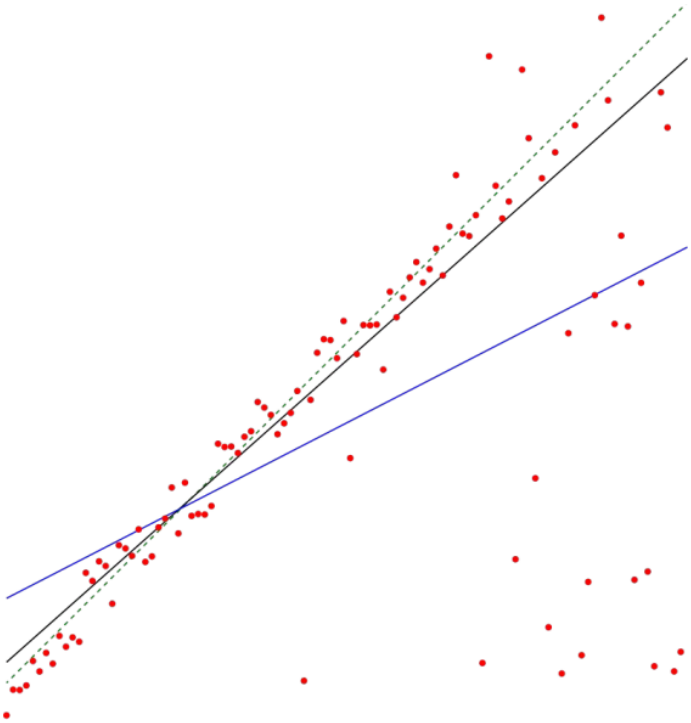

Рис. 13: Искажение результатов линейной регрессии небольшой долей выбросов

Причина такого поведения заключается в том, что квадрат отклонения быстро возрастает с ростом расстояния до моделируемой прямой, поэтому да-

же небольшое количество точек, полученных из другого распределения, приводят к большому штрафу для истинной прямой. Для улучшения результатов линейной регрессии следует заменить критерий (11) на другой, более устойчивый к выбросам.

Опр. 5. M-оценка. **M-оценкой** называется любая статистическая оценка, которая находится как решение уравнения следующего вида:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} \rho\left(\varepsilon(\theta, x_i)\right). \tag{12}$$

В случае линейной регрессии функция ε вычисляет расстояние от точки до прямой, заданной параметрами θ, а ρ является штрафной функцией. Для того, чтобы модель получилась устойчивой к выбросам, нужно, чтобы $\rho(t)$ сравнительно быстро возрастала в окрестности нуля, после чего возрастание замедлялось бы.

Предположим, что мы выбрали функцию $\rho(t)$ и хотим научиться минимизировать ее. Непосредственный перебор всех возможных параметров по некоторой дискретной равномерной сетке обладает высокой сложностью $\left(O\left(n_\rho n_\theta n\right)\right)$. Однако его можно ускорить при помощи преобразования Радона. Необходимо найти такое преобразование изображения, чтобы сумма результата по прямой была равна значению функции потерь для этой прямой с точностью до знака. После этого оптимальные параметры $\hat{\theta}$ можно будет найти следующим образом:

$$\hat{\theta} = \arg\max_{\theta} \left[\mathcal{R}\left(K_\rho\left(\|\vec{x}\|\right) * I\left(\vec{x}\right)\right)\right](\theta), \tag{13}$$

где $I\left(\vec{x}\right) = \sum_{i=1}^{n} \delta\left(\vec{x} - \vec{x}_i\right)$ – непрерывное изображение, соответствующее оцениваемому набору точек, $K_\rho\left(\|\vec{x}\|\right)$ – некоторое центрально-симметричное ядро, которое нужно найти. В соответствии в правилами свертки $K_\rho\left(\|\vec{x}\|\right) * I\left(\vec{x}\right) = \sum_{i=1}^{n} K_\rho\left(\|\vec{x} - \vec{x}_i\|\right)$.

Поскольку преобразование Радона линейно, достаточно рассмотреть случай, когда в наборе всего одна точка. Сравнивая выражения (12), (13), понимаем, что интеграл по прямой должен быть равен минус расстоянию от единственной точки до рассматриваемой прямой.

Пусть прямая задана таким образом, что ее точки задаются следующими

уравнениями (l – натуральный параметр на прямой):

$$\begin{cases} x_1(\theta, l) = -l\sin\theta_1 + \theta_0\cos\theta_1, \\ x_2(\theta, l) = l\cos\theta_1 + \theta_0\sin\theta_1. \end{cases}$$

Так как мы рассматриваем центрально-симметричные ядра, то преобразование Радона от ядра $f\left(\vec{x}\left(\theta, l\right)\right) =: F\left(\sqrt{x_1^2 + x_2^2}\right) = F\left(\sqrt{l_2^2 + \theta_0^2}\right)$ сводится к преобразованию Абеля:

$$\left[\mathcal{R}f\right]\left(\theta_0\right) = \int_{\mathbb{R}} F\left(\sqrt{l_2^2 + \theta_0^2}\right) dl = \int_{|\theta_0|}^{+\infty} \frac{F(r)r}{\sqrt{r^2 - \theta_0^2}} dr.$$

К счастью, Абель также нашел обратное преобразование, позволяющее находить функцию $F(r)$ в зависимости от желаемых значений интеграла (при условии, что она убывает достаточно быстро):

$$F(r) = -\frac{1}{\pi}\int_r^{\infty} \frac{\left[\mathcal{R}f\right]'\left(\theta_0\right)}{\sqrt{\theta_0^2 - r^2}} d\theta_0 = +\frac{1}{\pi}\int_r^{\infty} \frac{\rho'\left(\theta_0\right)}{\sqrt{\theta_0^2 - r^2}} d\theta_0.$$

Таким образом, для любой заданной функции ρ можно найти соответствующее ядро. Итоговый алгоритм записывается следующим образом:

- Дискретизовать изображение $I\left(\vec{x}\right)$ в двумерную гистограмму.
- Свернуть гистограмму с дискретизованным ядром, найденным при помощи обратного преобразования Абеля для желаемой целевой функции.
- Вычислить быстрое преобразование Хафа.
- Найти максимум в пространстве Хафа и вернуть параметры прямой, соответствующей этому максимуму.

Сложность алгоритма составляет $\Theta\left(N + w^2n^2 + n^2\log n\right)$, где N – количество точек, n – линейный размер квдаратной гистограммы, w – линейный размер используемого ядра.