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

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

Опр. 1. 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.

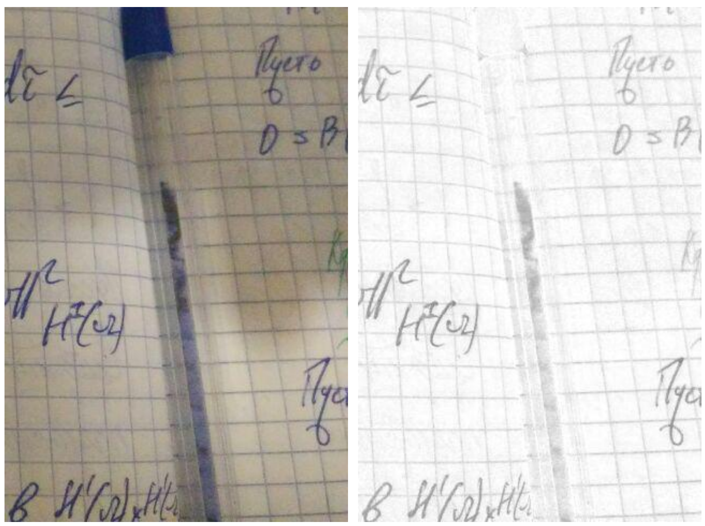

Рис. 1: Исходное изображение (слева) и разность между исходным изображением и размыканием (справа).

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. 2. 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.

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

В общем случае преобразование Хафа ставит в соответствие каждому паттерну из заданного семейства сумму значений пикселей изображения, принадлежащих 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 $i = \mathrm{const}$ 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 $\left\lfloor \dfrac{i}{2} \right\rfloor$. 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):

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

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

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 (1). 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 значений, а остальные заведомо имеют нулевые значения.

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 ($k = 0, 1, \dots, \log_2 N - 1$). 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

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

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:

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

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

При помощи элементарных преобразований данные из (2) 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. 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:

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

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:

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t)e^{-2\pi i(\omega t)}dt, \tag{3}$$

$$\hat{F}(u, v) = \int_{\mathbb{R}^2} f(x, y)e^{-2\pi i(xu+yv)}dudv. \tag{4}$$

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 $\delta(t) = \delta(-t)$):

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

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

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

■

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 (3) 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**):

$$\left[\mathcal{B}(p_\theta(s))\right](x,y) = \int_0^\pi p_\theta\left(x\cos\theta + y\sin\theta\right) d\theta. \quad (5)$$

Выражение (5) не совпадает с $f(x,y)$, однако часто используется как его приближение.

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

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

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

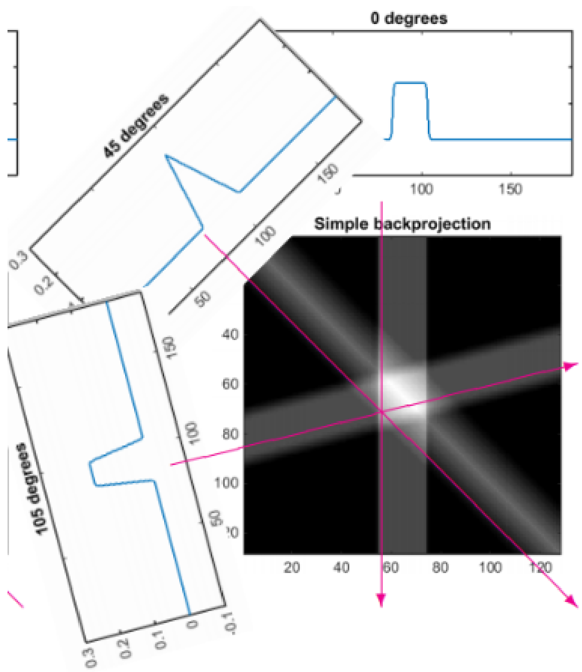

Рис. 2: Суть backprojection в одной картинке

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

$$\int_0^\pi \int_{\mathbb{R}} |\omega|\hat{p}_\theta(\omega)e^{2\pi i s\omega}d\omega d\theta = \int_0^\pi \left[\mathcal{F}^{-1}\left(|\omega|\hat{p}_\theta(\omega)\right)\right](s)d\theta. \tag{6}$$

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 (5).

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 (5) более правильной формулы, выведенной в (6):

$$\left[\mathcal{B}_f(p_\theta(s))\right](x,y) = \mathcal{F}^{-1}\left[|\omega|\hat{p}_\theta(\omega)\right](x\cos\theta + y\sin\theta)\,d\theta. \tag{7}$$

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**.

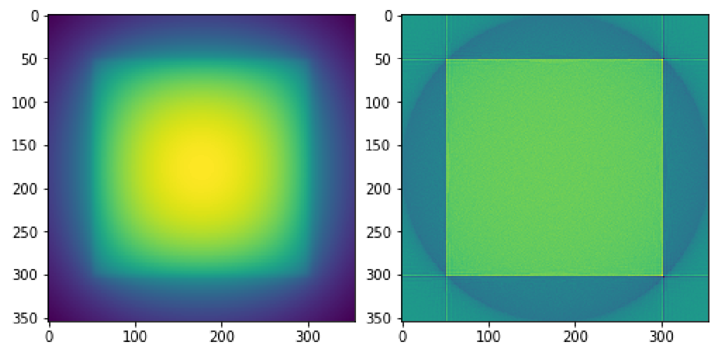

Рис. 3: Восстановление белого квадрата из синограммы при помощи backprojection (слева) и filtered backprojection (справа).

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

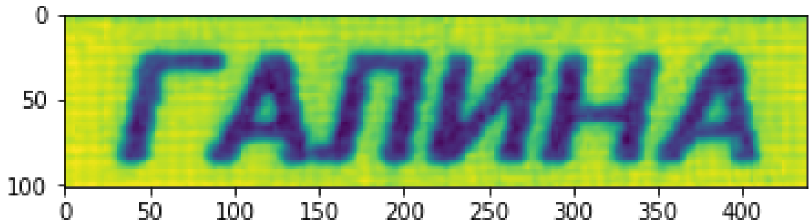

Рис. 4: Исходное изображение

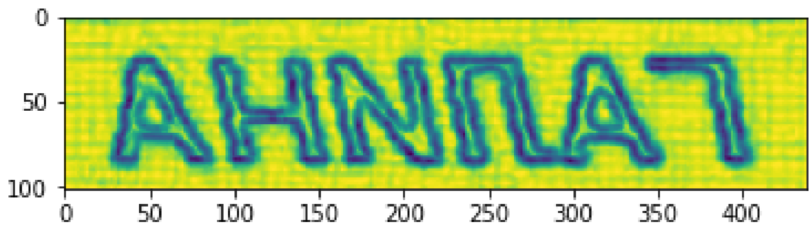

Рис. 5: Выделенные границы на изображении

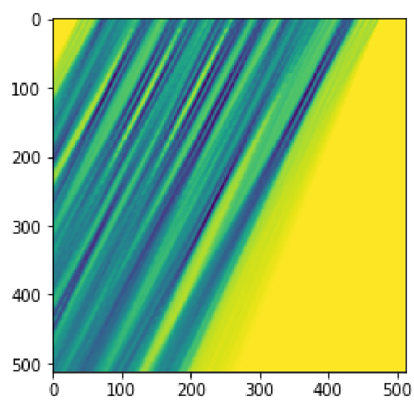

Рис. 6: Результат применения БПХ к изображению

с определенным наклоном $\theta_i = \arctan\left(\dfrac{n-1}{i}\right)$. 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-

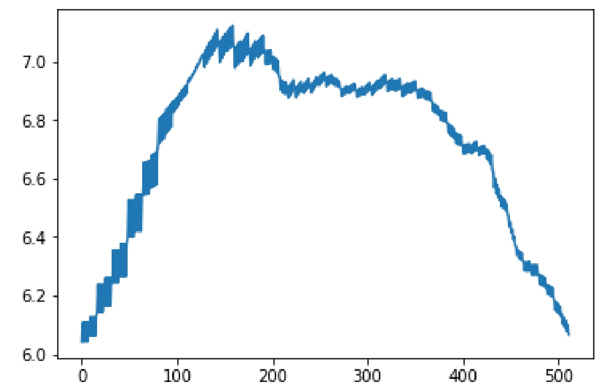

Рис. 7: График стандартных отклонений строк

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

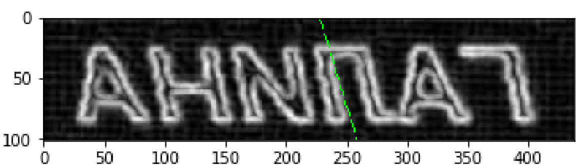

Рис. 8: Изображение со случайно выбранной прямой из найденного семейства

16. Способ использования БПХ для слепой компенсации радиальной дисторсии.

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.

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

20. Быстрая линейная бинарная кластеризация с помощью БПХ.

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