5 ЛАБОРАТОРНАЯ РАБОТА №5. ПРОСТРАНСТВЕННАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ

ЦЕЛЬ РАБОТЫ – Изучение принципов пространственной обработки и фильтрации изображений; реализация методов пространственной обработки изображений на языке Python.

5.1 Теоретические сведения

5.1.1 Техника обработки изображений в пространственной области

Обработку изображения $s(i,j)$ в пространственной области можно описать выражением

$$d(i,j) = T\big(s(i,j)\big), \tag{5.1}$$

где T – некоторый оператор (преобразование) над s, который определен в некоторой окрестности точки (i,j).

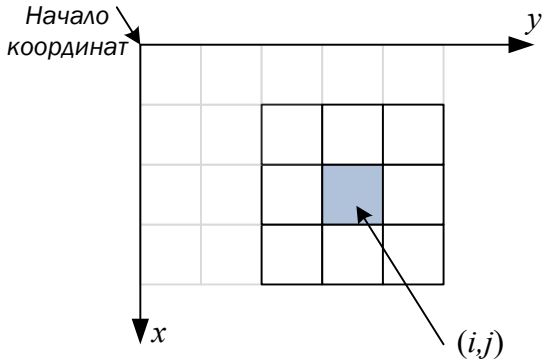


Рисунок 19 – Окрестность размера 3 × 3 вокруг точки (i,j)

Окрестность вокруг точки (i,j) как правило выбирается в виде квадратной или прямоугольной области с центром в точке (i,j), как показано на рис. 19. Центр заданной шаблонной подобласти перемещается от пиксела к пикселу, начиная, как правило, из верхнего левого угла, и на своем пути он покрывает все изображение. Преобразование T применяется в каждой точке (i,j), давая в результате выходное (обработанное) значение $d(i,j)$ для данной точки.

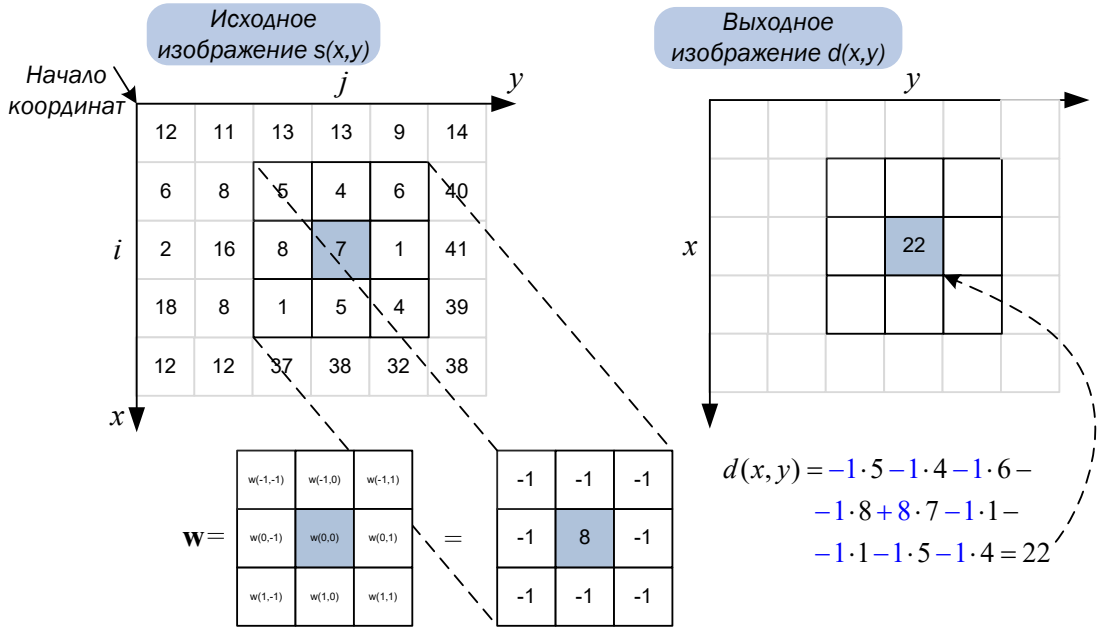
5.1.2 Свертка

Исходя из описанной выше концепции значение пикселя (i,j) на обработанном изображении является линейной комбинацией значений пикселей, находящихся в окрестности точки (i,j) на исходном изображении. Значения на которые умножаются пиксели в окрестности точки (i,j) образуют матрицу w размером $n \times m$, которую называют ***фильтром*** или ***ядром*** (реже используют термины ***шаблон*** и ***маска***). Обработка изображения при помощи пространственного фильтра $\mathrm{w} = [w(i,j)]$ описывается следующим уравнением свертки:

$$d(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) \cdot s(x+i, y+j), \qquad (5.2)$$

где значения индексов $i = 0$, $j = 0$ соответствуют центру фильтра; $a = \frac{m-1}{2}$; $b = \frac{n-1}{2}$. Ширина (m) и высота (n) ядра всегда выбираются нечетными.

На рис. 20 приведена иллюстрация, поясняющая процесс вычисления свертки изображения с ядром 3×3.



$$d(x,y) = w(-1,-1)s(x-1,y-1) + w(-1,0)s(x-1,y) + w(-1,1)s(x-1,y+1) +$$
$$+ w(0,-1)s(x,y-1) + w(0,0)s(x,y) + w(0,1)s(x,y+1) +$$
$$+ w(1,-1)s(x+1,y-1) + w(1,0)s(x+1,y) + w(1,1)s(x+1,y+1)$$

Рисунок 20 – Процесс вычисления свертки изображения с ядром 3×3

5.1.3 Сглаживающий фильтр

Из предыдущего раздела нам известно, что для формирования линейного пространственного фильтра размерами $m \times n$ требуется задать $m \cdot n$ коэффициентов маски $w(i,j)$. Выбор этих коэффициентов основан на том, какие действия ожидаются от фильтра.

Выход простейшего линейного сглаживающего (усредняющего) пространственного фильтра есть среднее значение элементов по окрестности, покрытой маской фильтра. Идея применения сглаживающих фильтров заключается в уменьшение «резких» переходов уровней яркости, поскольку случайный шум как раз характеризуется резкими скачками яркости.

Ниже приведен пример сглаживающего фильтра по окрестности 3×3:

$$w = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \quad (5.3)$$

Данный вариант дает обычное средние значение яркости по окрестности 3×3. Заметим, что коэффициенты указаны как единицы, место $1/9$. Такой вариант является более эффективным с вычислительной точки зрения. В общем случае маска размерами $m \times n$ будет иметь нормировочный коэффициент $1/mn$. Такой пространственный фильтр, все коэффициенты которого одинаковы, иногда называют ***однородным усредняющим фильтром***.

Рассмотрим ещё один сглаживающий фильтр, называемый ***взвешенным средним***:

$$w = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \quad (5.4)$$

В данном случае значения элементов умножаются на разные коэффициенты, что позволяет им присвоить как бы разные «важности» (веса) по сравнению с другими.

5.1.4 Фильтры, основанные на порядковых статистиках

Базовые понятия

Фильтры, основанные на порядковых статистиках, относятся к методам нелинейно обработки изображений. Процесс обработки заключается в том, что пиксели, располагающиеся в окрестности точки с координатами (i, j), рассматриваются, как (одномерный) массив данных на основе которого производится вычисление некоторой порядковой статистики. Наиболее распространенными являются медианный фильтр, min- и max-фильтры.

Медианная фильтрация

Вначале вспомним определение того, что именно в статистике называют медианой. ***Медиана*** набора чисел есть такое число ξ, что половина чисел из набора меньше или равны ξ, а другая половина — больше или равны ξ. Чтобы выполнить медианную фильтрацию для элемента изображения, необходимо:

1. Упорядочить по возрастанию значения пикселей внутри окрестности.

2. Найти значение медианы. Для окрестности 3×3 элементов медианой будет пятое значение по величине, для окрестности 5×5 — тринадцатое значение и т.д.

3. Присвоить полученное значение обрабатываемому элементу.

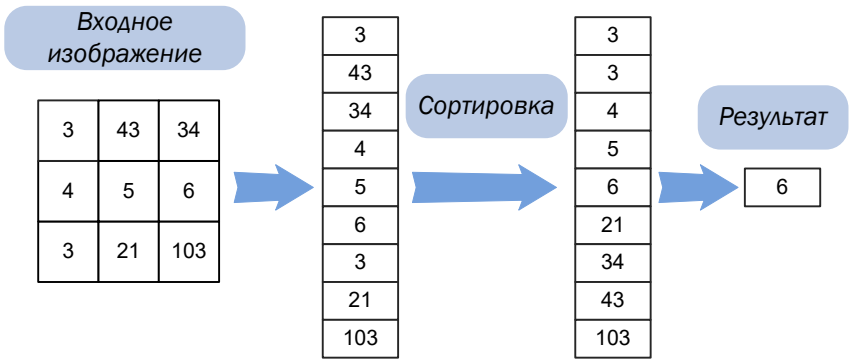


Рисунок 21 – Идея медианной фильтрации в окне 3 × 3

Ранговая фильтрация

Медианный фильтр является частным случаем класса фильтров, называемых ранговыми или порядковыми. Ранговый фильтр порядка r ($1 \leq r \leq N$, где N – число элементов в окрестности) выбирает из полученного ряда элемент с номером r и присваивает его значение как результат фильтрации пикселя исходного изображения. Если число N нечетное и $r = (N + 1)/2$, фильтр становится медианным. Если $r = 1$, фильтр выбирает минимальное значение яркости в окне и называется min-фильтром. Если $r = N$, фильтр выбирает максимальное значение яркости в окне и называется max-фильтром.

5.1.5 Фильтр повышения резкости изображения

Главная цель повышения резкости заключается в том, чтобы усилить перепады и другие разрывы (например, шумы) и не подчеркивать области с медленными изменениями уровней яркостей. Простейшим оператором повышения резкости, основанным на производных, является лапласиан (оператор Лапласа), который в случае функции двух переменных $f(x, y)$ определяется как:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + \\ + f(x,y+1) + f(x,y-1) - 4f(x,y). \tag{5.5}$$

Данное выражение можно описать в терминах свертки (5.2). В этом случае ядро $w(i, j)$ будет иметь следующий вид:

$$\mathrm{w}_{\text{Lapl-1}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{5.6}$$

Диагональные элементы также могут быть включены в формулу дискретного лапласиана:

$$\mathrm{w}_{\text{Lapl-2}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{5.7}$$

Поскольку оператор Лапласа по сути является второй производной, его применение подчеркивает разрывы уровней яркости на изображении и подавляет области со слабыми уровнями яркости.

Метод использования лапласиана для улучшения изображения сводится к следующему

$$\begin{aligned} d(x,y) = {} & s(x,y) - \nabla^2 s(x,y) \\ = {} & s(x,y) - \mathrm{w}_{\mathrm{Lapl}} * s(x,y). \end{aligned} \tag{5.8}$$

5.1.6 Выделение границ на изображении

Выделение границ на изображении выполняется при помощи фильтров, которые аппроксимируют операцию вычисления производной. Ядра таких фильтров устроены таким образом, чтобы возвращать большие значения в тех местах, где изображение имеет резкие переходы (разрывы). Таким образом выполняется детектирование границ (англ. *edge detection*).

Оператор выделения границ на изображении обычно имеет следующее описание:

$$\begin{aligned} G &= \sqrt{{G_x}^2 + {G_y}^2}, \\ \theta &= \mathrm{arctg}\left(\frac{G_y}{G_x}\right). \end{aligned} \tag{5.9}$$

Здесь G – это оценка градиента изображения, который подчеркивает имеющиеся границы. В свою очередь θ – ориентация градиента изображения.

x- и y-компоненты градиента вычисляются путем свертки изображения с ядром, аппроксимирующем вычисление производной вдоль соответствующей оси:

$$G_x = S * D_x, \qquad G_y = S * D_y. \tag{5.10}$$

На практике чаще всего используются следующие ядра для дифференцирования:

Таблица – Фильтры для детектирования границ

Название оператора	D_x	D_y
Робертс	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
Прюитт	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Собель	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Перекрестный оператор Робертса быстро вычисляется (из-за минимального размера ядер), но он очень чувствителен к шуму. Детекторы Прюитта и Собела не так чувствительны к шуму, но используют более сложные ядра.

Ядра Прюитта/Собеля, как правило, предпочтительнее подхода Робертса, поскольку градиент не сдвигается на половину пикселя в обоих направлениях Ключевое различие между операторами Собеля и Прюитта заключается в том, что ядро Собеля реализует дифференцирование в одном направлении и (приблизительное) усреднение по Гауссу в другом. Преимущество этого заключается в том, что он сглаживает краевую область, снижая вероятность того, что зашумленные или изолированные пиксели будут доминировать в отклике фильтра.

5.1.7 Модели шума

Гауссовый шум

Наиболее часто возникающий тип шума это аддитивный гауссовый шум. Он широко используется для моделирования тепловых эффектов и многого другого. Функция плотности вероятности гауссова шума q со средним значением μ и дисперсией σ^2 описывается выражением

$$p_q(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}. \quad (5.11)$$

На рис. 22 показан пример сгенерированного гауссового шума, который имеет среднее значение $\mu = 20$, а СКО $\sigma = 5$.

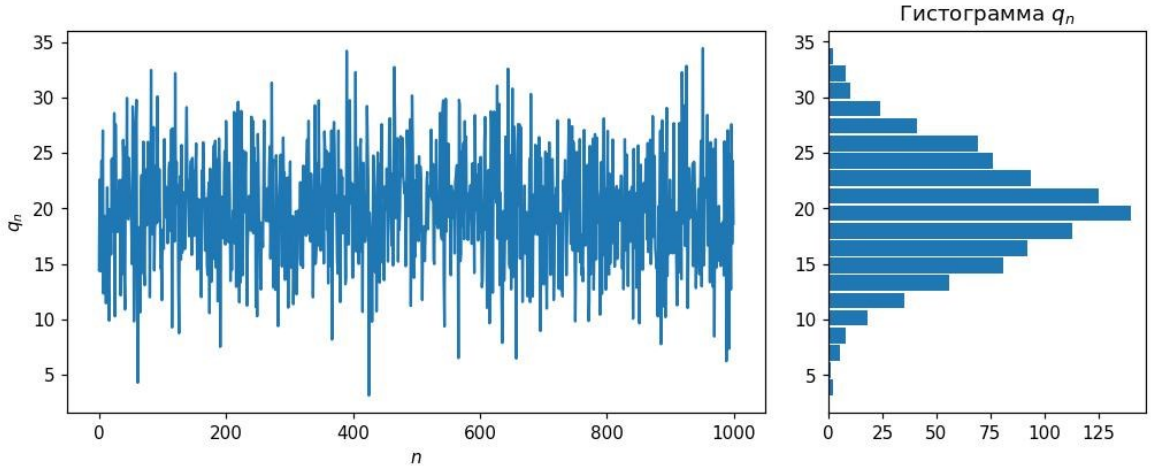


Рисунок 22 – Гауссовый шум: $\mu = 20$, $\sigma = 5$

Слева на рис. 22 приведена гистограмма сгенерированного шума, которая по форме напоминает функцию плотности вероятности (5.11).

Гауссов шум используется в качестве естественного приближения в тех случаях, когда сенсорные детекторы изображения работают на пороге чувствительности.

В Python для генерации приведенного выше примера гауссового шума использовался следующий код

```
[1]: mean = 20
[2]: sigma = 5
[3]: q = np.random.normal(mean, sigma, 1000)
```

Шум «соль и перец» (импульсный шум)

Шум типа «соль и перец» возникает в устройствах с ошибочной коммутацией. Этот шум характеризуется тем, что на изображении возникает множество случайных белых («соль») и черных («перец») пикселей.

Функция плотности распределения вероятностей импульсного шума задаётся выражением

$$p(x) = \begin{cases} P_p \text{ при } x = p, \\ P_s \text{ при } x = s, \\ 0 \text{ в остальных случаях.} \end{cases} \quad (5.12)$$

Если $s > p$, то пиксель с яркостью s выглядит как светлая точка на изображении. Пиксель с яркостью s выглядит, наоборот, как темная точка.

Следующая функция Python иллюстрирует, как можно добавить импульсный шум на изображение

```
# Adding salt & pepper noise to an image
def salt_pepper(img, prob_s, prob_p):
    # prob_s – “salt” probability
    # prob_p – “pepper” probability

    output = np.copy(img).reshape((img.size))

    # Apply salt noise on each pixel individually
    num_salt = np.ceil(prob_s * img.size)
    coords = np.random.randint(0, img.size - 1, int(num_salt))
    output[coords] = 255

    # Apply pepper noise on each pixel individually
    num_pepper = np.ceil(prob_p * img.size)
    coords = np.random.randint(0, img.size - 1, int(num_pepper))
    output[coords] = 0

    return output.reshape(img.shape)
```

На рис. 23 показан пример добавления шума типа «соль и перец» на изображение. На данном случае вероятность появления ярких точек («соли») в 10 раз меньше, чем темных («перца»), поэтому визуально на изображении больше черных точек.

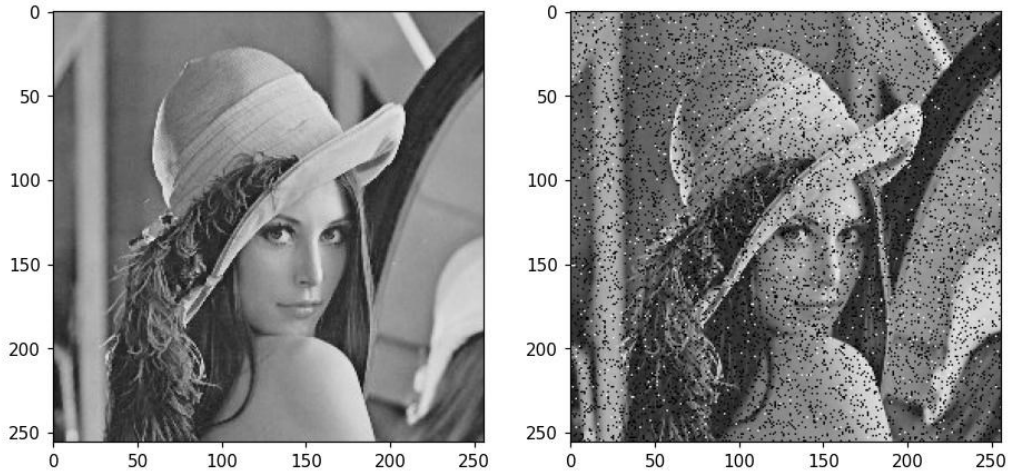


Рисунок 23 – Добавление к изображению шума «соль и перец» ($P_s = 0{,}01$ – вероятность «соли», $P_p = 0{,}1$ – вероятность «перца»)

Добавление шума к изображению

Для добавления шума к изображению необходимо вначале сгенерировать число отсчетов шума равное числу пикселей на изображении, а затем применить к полученному массиву метод `reshape` после чего сложить изображение с шумом.

5.2 Порядок выполнения работы

1) Взять монохромное изображение размером не более 1000x1000 пикселей и добавить к нему гауссовый шум с параметрами, приведенными в таблице 1.

Таблица 1 – Варианты для задания 1

Номер варианта	μ	σ
1	20	5
2	40	9
3	60	17
4	70	20
5	30	11
6	50	22
7	10	10
8	50	14

На экране отобразить 1) исходное изображение; 2) сгенерированный шум и 3) изображение с добавленным шумом.

2) Написать функцию на языке Python для сглаживающей фильтрации в соответствии с вариантом (таблица 2).

Таблица 2 – Варианты для задания 2

Номер варианта	Размер окрестности	Тип фильтра
1	3×3	Однородный усредняющий
2	5×5	Взвешенное среднее
3	7×3	Однородный усредняющий
4	5×7	Взвешенное среднее
5	7×7	Однородный усредняющий
6	9×5	Взвешенное среднее
7	11×9	Однородный усредняющий
8	13×13	Взвешенное среднее

3) Взять монохромное изображение размером не более 1000x1000 пикселей и добавить к нему шум типа соль и перец с параметрами, приведенными в таблице 3.

Таблица 3 – Варианты для задания 3

Номер варианта	P_s	P_p
1	0,05	0,05
2	0,01	0,09
3	0,12	0,00
4	0,03	0,08
5	0,00	0,15
6	0,10	0,08
7	0,08	0,10
8	0,02	0,12

4) Написать функцию на языке Python для ранговой фильтрации в соответствии с вариантом (таблица 4).

Таблица 4 – Варианты для задания 3

Номер варианта	Размер окрестности	Порядок фильтра r
1	5×5	10
2	7×7	25
3	9×9	41
4	11×11	121
5	17×17	501
6	19×19	201
7	15×15	111
8	13×13	89

Примените разработанную функцию фильтрации к изображению из задания 3. Отобразите на экране результат фильтрации.

5) Разработать функцию, реализующую метод повышения резкости (5.8) на изображении в соответствии с вариантом (таблица 5).

Таблица 5 – Варианты для задания 5

Номер варианта	Тип лапласиана
1	$\mathbf{w}_{\text{Lapl-2}}$
2	$\mathbf{w}_{\text{Lapl-2}}$
3	$\mathbf{w}_{\text{Lapl-2}}$
4	$\mathrm{w}_{\text{Lapl-1}}$
5	$\mathbf{w}_{\text{Lapl-2}}$
6	$\mathbf{w}_{\text{Lapl-1}}$
7	$\mathbf{w}_{\text{Lapl-1}}$
8	$\mathbf{w}_{\text{Lapl-1}}$

Применить разработанную функцию к изображению. Вывести на экран исходной изображение, изображение после обработки его лапласианом и изображение с повышенной резкостью.

6) Разработать функцию, реализующую метод выделения краев на изображении в соответствии с вариантом (таблица 6).

Таблица 6 – Варианты для задания по детектированию границ

Номер варианта	Тип оператора
1	Робертс
2	Прюитт
3	Собель
4	Робертс
5	Прюитт
6	Собель
7	Прюитт
8	Собель

Применить разработанную функцию к изображению. Вывести на экран исходной градиент изображения G и ориентацию градиента θ.

7) Оформить отчет в соответствии с СТП 01-2017.