

Федеральное государственное автономное образовательное
учреждение высшего образования
**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет систем управления и робототехники

Лабораторная работа №3
ФИЛЬТРАЦИЯ И ВЫДЕЛЕНИЕ КОНТУРОВ

Студенты: Овчинников П.А.
Румянцев А.А.
Чебаненко Д.А.

Поток: ТЕХ.ЗРЕНИЕ 2.1

Преподаватель: Шаветов С.В.

Санкт-Петербург
2024

Содержание

| | |
|---|-----------|
| Цель работы | 3 |
| Давайте пошумим! | 3 |
| Импульсный шум | 3 |
| Аддитивный шум | 3 |
| Мультиплкативный шум и спекл-шум | 3 |
| Гауссов шум | 4 |
| Шум квантования | 4 |
| Применение шумов к изображению | 4 |
| Низкочастотная фильтрация (или повесть «Прекращаем шуметь!») | 5 |
| Фильтр Гаусса | 5 |
| Контргармонический усредняющий фильтр | 6 |
| Нелинейная фильтрация (покупайте нелинейки, чтобы разобраться) | 11 |
| Медианный фильтр | 11 |
| Взвешенный медианный фильтр | 13 |
| Ранговый фильтр | 16 |
| Винеровский фильтр | 23 |
| Адаптивный медианный фильтр | 25 |
| Высокочастотная фильтрация (выстраиваем границы понимания) | 28 |
| Фильтр Робертса | 28 |
| Фильтр Превитта (Прюитта) | 28 |
| Фильтр Собела | 29 |
| Фильтр Лапласа | 30 |
| Алгоритм Кэнни | 31 |
| Вывод | 33 |
| Вопросы к защите | 33 |

Цель работы

В этой работе мы изучим разные типы шумов и фильтров. С помощью последних мы научимся фильтровать изображения от шумов. Также, используя фильтры, мы научимся выделять контуры в изображении. И всё это мы сможем сделать с помощью библиотек `OpenCV`, `scikit-image` и `SciPy` прямо в `Python`. Весь код будет приведён ссылкой на приватный гист в GitHub Gist, которую мы оставим в конце работы.

И перед её выполнением взглянем первый и последний раз на исходное изображение:



Рис. 1: Исходное изображение

Давайте пошумим!

Итак, начнём с шумов. По каждому из них я приведу теоретическую справку и только затем покажу результат их применения к изображению, дабы сэкономить пространство.

Импульсный шум

Первый, один из простых — импульсный шум, появляющийся, например, из-за ошибок декодирования. Такой шум ещё называется точечным шумом, потому что приводит к появлению на изображении белых и чёрных точек, которые ещё называют «солью» и «перцем». И работает он очень просто:

$$I_n(x, y) = \begin{cases} d, & \text{с вероятностью } p, \\ s_{x,y}, & \text{с вероятностью } (1 - p). \end{cases}$$

Здесь $s_{x,y}$ — интенсивность пикселя исходного изображения, I_n — зашумлённое изображение, p — вероятность появления шума в пикселе, d — вероятность появления белой точки (чем выше, тем больше вероятность, что появится белая точка, иначе появится чёрная). Выходит, мы можем регулировать как вероятность появления шума в принципе, так и отношение «соли» к «перцу».

Аддитивный шум

Аддитивный шум предполагает наличие распределения функции плотности вероятности, которая просто добавляется к исходному изображению:

$$I_n(x, y) = I(x, y) + \eta(x, y).$$

I — исходная функция, η — аддитивный шум с гауссовым или любым другим распределением функции плотности вероятности, а I_n — зашумлённое изображение.

Мультипликативный шум и спекл-шум

Мультипликативный шум — то же самое, что и аддитивный шум, но шум не добавляется к исходному изображению, а умножается на него. Зернистость плёнки или ультразвуковые изображение — примеры такого шума.

$$I_n(x, y) = I(x, y) \cdot \eta(x, y)$$

Изображения, снятые медицинскими сканерами или радарами, обладают светлыми крапинками (спеклами), разделёнными тёмными участками изображения — это частный случай мультипликативного шума, так называемый спекл-шум.

Гауссов шум

Когда сцена недостаточно освещена или высокая температура в помещении плохо влияет на матрицу фотокамеры, появляется гауссов шум или нормальный шум. Именно этот шум в небольшом количестве появляется на фотографиях и от него обычно стараются избавиться. Функция распределения плотности вероятности $p(z)$ случайной величины z описывается следующим выражением:

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}},$$

где z — интенсивность изображения, μ — среднее (или мат.ожидание) величины z , σ — среднеквадратичное отклонение.

Шум квантования

Шум квантования — ошибки, возникающие при квантовании сигнала, когда сигнал округляется или усекается. Такой шум не устранимся, поэтому нужно быть очень осторожным, при квантовании изображения. Одно из точных приближение к шуму квантования — распределение Пуассона, которое выглядит вот так:

$$p(z) = \frac{\lambda^z}{z!} e^{-\lambda}.$$

Здесь z — так же интенсивность изображения, а λ — мат.ожидание величины.

Применение шумов к изображению



Рис. 2: Импульсный шум



Рис. 3: Аддитивный шум

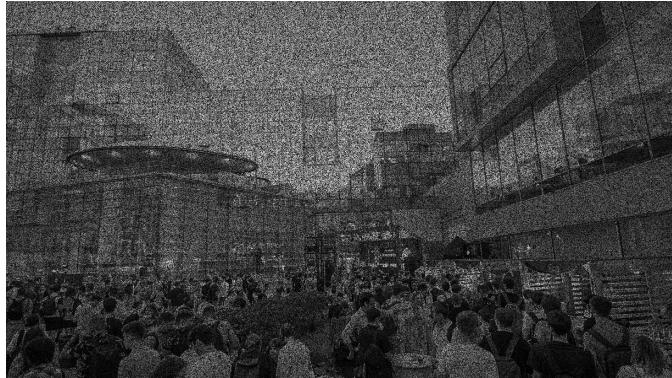


Рис. 4: Мультиплексивный шум

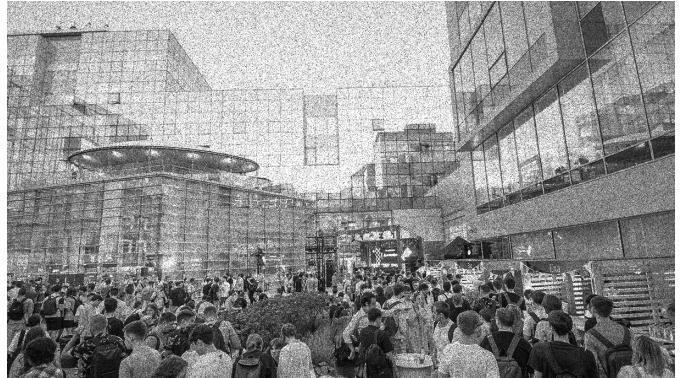


Рис. 5: Спекл-шум



Рис. 6: Гауссов шум



Рис. 7: Шум квантования

Низкочастотная фильтрация (или повесть «Прекращаем шуметь!»)

Низкочастотные фильтры ослабляют высокочастотные компоненты (области с сильным изменением интенсивностей — например, границы объектов) и оставляют низкочастотные компоненты без изменений. Именно такие фильтры снижают уровень шума, и в результате мы получаем сглаженное или размытое изображение.

Фильтр Гаусса

Один из простейших фильтров, который активно применяется во многих сферах и чаще известен как «размытие по Гауссу». Цифровые камеры многих смартфонов используют этот фильтр для ослабления шума, а некоторые графические приложения перед уменьшением размера изображения могут применить фильтр для предотвращения артефактов и алиасинга. И вот как выглядит двумерный фильтр Гаусса:

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-y^2}{2\sigma^2}}$$

С ростом σ изображение размывается сильнее.

Вот как фильтр Гаусса применяется к каждому из шумов (размера ядра 3×3):



Рис. 8: Импульсный шум + фильтр Гаусса



Рис. 9: Аддитивный шум + фильтр Гаусса



Рис. 10: Мультипликативный шум + фильтр Гаусса



Рис. 11: Спекл-шум + фильтр Гаусса



Рис. 12: Гауссов шум + фильтр Гаусса



Рис. 13: Шум квантования + фильтр Гаусса

Как мы видим, фильтр Гаусса хорошо подавляет аддитивный шум.

Контргармонический усредняющий фильтр

$$I_f(x, y) = \frac{\sum_{i=0}^m \sum_{j=0}^n I(i, j)^{Q+1}}{\sum_{i=0}^m \sum_{j=0}^n I(i, j)^Q}$$

Q — порядок фильтра. Контргармонический фильтр является обобщением усредняющих фильтров и может подавлять как шум типа «соль» ($Q < 0$), так и шум типа «перец» ($Q > 0$), но фильтр не способен удалить одновременно белые и чёрные точки.

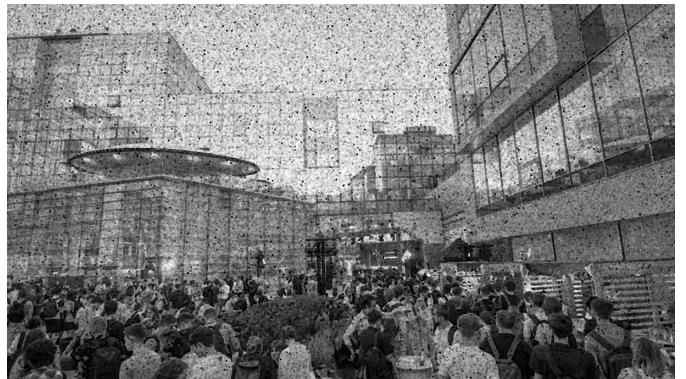
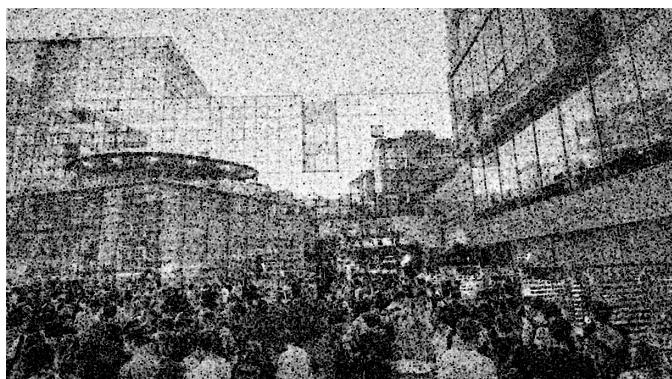
Рис. 14: Импульсный шум + фильтр ($Q = -1.5$)Рис. 15: Аддитивный шум + фильтр ($Q = -1.5$)Рис. 16: Мульти-ный шум + фильтр ($Q = -1.5$)Рис. 17: Спекл-шум + фильтр ($Q = -1.5$)Рис. 18: Гауссов шум + фильтр ($Q = -1.5$)Рис. 19: Шум квантования + фильтр ($Q = -1.5$)

Рис. 20: Импульсный шум + фильтр ($Q = -0.5$)Рис. 21: Аддитивный шум + фильтр ($Q = -0.5$)Рис. 22: Мульти-ний шум + фильтр ($Q = -0.5$)Рис. 23: Спекл-шум + фильтр ($Q = -0.5$)Рис. 24: Гауссов шум + фильтр ($Q = -0.5$)Рис. 25: Шум квантования + фильтр ($Q = -0.5$)

Рис. 26: Импульсный шум + фильтр ($Q = 0.5$)Рис. 27: Аддитивный шум + фильтр ($Q = 0.5$)Рис. 28: Мульти-шум + фильтр ($Q = 0.5$)Рис. 29: Спекл-шум + фильтр ($Q = 0.5$)Рис. 30: Гауссов шум + фильтр ($Q = 0.5$)Рис. 31: Шум квантования + фильтр ($Q = 0.5$)

Рис. 32: Импульсный шум + фильтр ($Q = 1.5$)Рис. 33: Аддитивный шум + фильтр ($Q = 1.5$)Рис. 34: Мульти-ний шум + фильтр ($Q = 1.5$)Рис. 35: Спекл-шум + фильтр ($Q = 1.5$)Рис. 36: Гауссов шум + фильтр ($Q = 1.5$)Рис. 37: Шум квантования + фильтр ($Q = 1.5$)

Работу контргармонического усредняющего фильтра хорошо видно на импульсном шуме. Действительно, на отрицательных значениях Q мы видим чёрные точки и при этом белые подавляются, а при положительных значениях Q всё наоборот. Но с аддитивным шумом фильтр работает плохо — тёмные или светлые контуры (в зависимости от знака Q) размываются.

Нелинейная фильтрация (покупайте нелинейки, чтобы разобраться)

Низкочастотные фильтры хороши, когда шум основан на нормальном распределении — они сглаживают изображение, оставляя только высокочастотные компоненты, но это не самый оптимальный метод. Ведь существуют нелинейные фильтры, которые могут устранять импульсные помехи другими способами.

Медианный фильтр

И начнём с одного из главного нелинейного фильтра — обычновенного медианного фильтра. Среди пикселей, попавших в окно фильтрации, выбирается медианный по интенсивности. Посмотрим на результат работы фильтра, применённого к зашумленным разными методами изображениям, с разными размерами окон:



Рис. 38: Импульсный шум + фильтр ($ksize = 3$)



Рис. 39: Аддитивный шум + фильтр ($ksize = 3$)

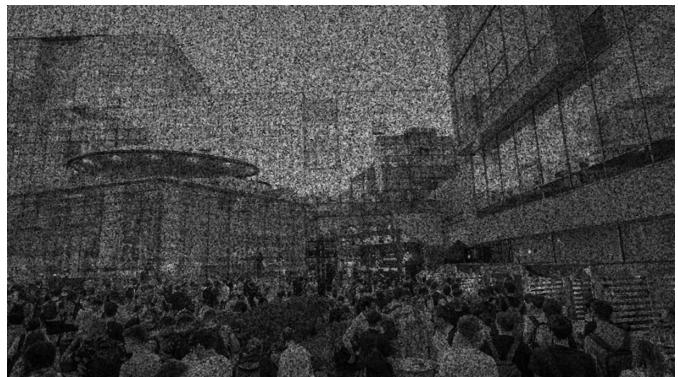


Рис. 40: Мульти-ний шум + фильтр ($ksize = 3$)

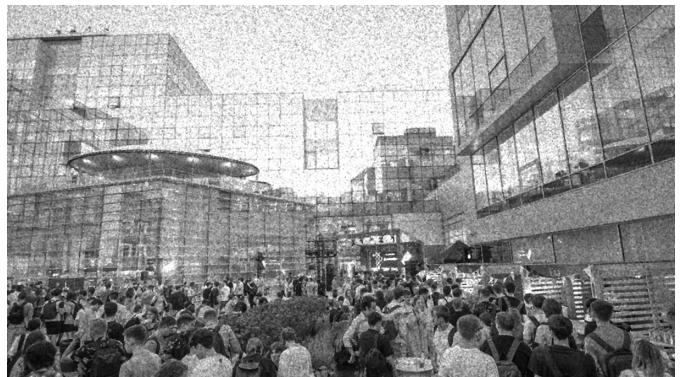


Рис. 41: Спекл-шум + фильтр ($ksize = 3$)



Рис. 42: Гауссов шум + фильтр ($ksize = 3$)



Рис. 43: Шум квантования + фильтр ($ksize = 3$)



Рис. 44: Импульсный шум + фильтр ($ksize = 5$)



Рис. 45: Аддитивный шум + фильтр ($ksize = 5$)



Рис. 46: Мульти-ный шум + фильтр ($ksize = 5$)



Рис. 47: Спекл-шум + фильтр ($ksize = 5$)



Рис. 48: Гауссов шум + фильтр ($ksize = 5$)



Рис. 49: Шум квантования + фильтр ($ksize = 5$)

Рис. 50: Импульсный шум + фильтр ($ksize = 9$)Рис. 51: Аддитивный шум + фильтр ($ksize = 9$)Рис. 52: Мульти-ний шум + фильтр ($ksize = 9$)Рис. 53: Спекл-шум + фильтр ($ksize = 9$)Рис. 54: Гауссов шум + фильтр ($ksize = 9$)Рис. 55: Шум квантования + фильтр ($ksize = 9$)

Как мы видим, медианный фильтр идеально удаляет импульсный шум. Конечно, мы потеряли некоторые детали. К примеру, электрические провода в верху фотографии уже не такие чёткие и разрываются. Также медианный фильтр неплохо справляется с аддитивным шумом. С ростом размера окна фильтрации мы наблюдаем всё большее и большее размытие очертаний объектов — отличие от фильтра Гаусса состоит в том, что объекты как будто теряют свою многоформерность и становятся менее полигональными. Но можно ли как-то улучшить медианную фильтрацию?

Взвешенный медианный фильтр

Да, есть. К примеру, взвешенная медианная фильтрация использует маску с весовыми коэффициентами. Эти коэффициенты подсказывают фильтру, какие пиксели должны больше влиять на результат фильтрации.



Рис. 56: Импульсный шум + фильтр ($ksize = 3$)



Рис. 57: Аддитивный шум + фильтр ($ksize = 3$)

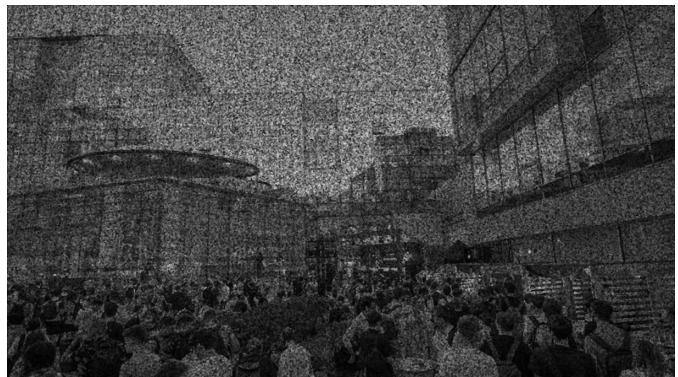


Рис. 58: Мульти-ний шум + фильтр ($ksize = 3$)

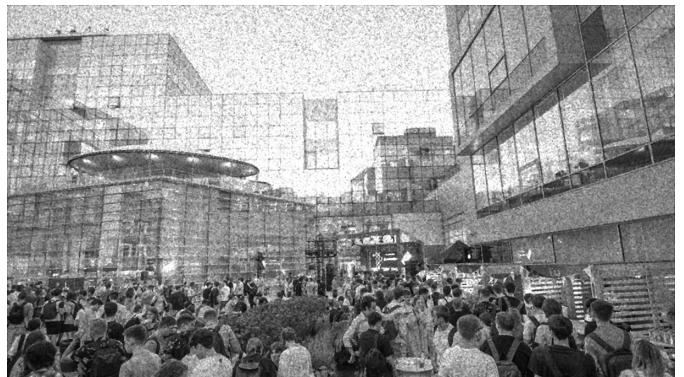


Рис. 59: Спекл-шум + фильтр ($ksize = 3$)



Рис. 60: Гауссов шум + фильтр ($ksize = 3$)



Рис. 61: Шум квантования + фильтр ($ksize = 3$)



Рис. 62: Импульсный шум + фильтр ($ksize = 5$)



Рис. 63: Аддитивный шум + фильтр ($ksize = 5$)



Рис. 64: Мульти-ний шум + фильтр ($ksize = 5$)



Рис. 65: Спекл-шум + фильтр ($ksize = 5$)



Рис. 66: Гауссов шум + фильтр ($ksize = 5$)



Рис. 67: Шум квантования + фильтр ($ksize = 5$)

Рис. 68: Импульсный шум + фильтр ($ksize = 9$)Рис. 69: Аддитивный шум + фильтр ($ksize = 9$)Рис. 70: Мульти-ный шум + фильтр ($ksize = 9$)Рис. 71: Спекл-шум + фильтр ($ksize = 9$)Рис. 72: Гауссов шум + фильтр ($ksize = 9$)Рис. 73: Шум квантования + фильтр ($ksize = 9$)

Во многом взвешенная медианная фильтрация отрабатывает так же, как и медианная — разве что немного меняются цвета (яркость) некоторых объектов.

Ранговый фильтр

Обобщает медианную фильтрацию, т.к. позволяет выбрать пиксель с любым *рангом* — это может быть как порядковый номер в векторе-столбце отсортированных интенсивностей пикселей окна, так и процент.

Мы посмотрим на результат применения фильтра с размерами окна 3×3 , 5×5 и для каждого из размеров с минимальным, близким к медианному и близким к максимальному рангом.



Рис. 74: Импульсный шум + фильтр ($k = 3$, $rank = 1$)



Рис. 75: Аддитивный шум + фильтр ($k = 3$, $rank = 1$)



Рис. 76: Мульти-ный шум + фильтр ($k = 3$, $rank = 1$)



Рис. 77: Спекл-шум + фильтр ($k = 3$, $rank = 1$)

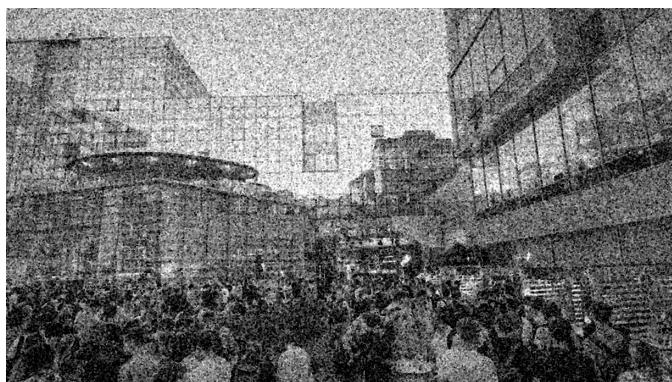


Рис. 78: Гауссов шум + фильтр ($k = 3$, $rank = 1$)



Рис. 79: Шум квантования + фильтр ($k = 3$, $rank = 1$)

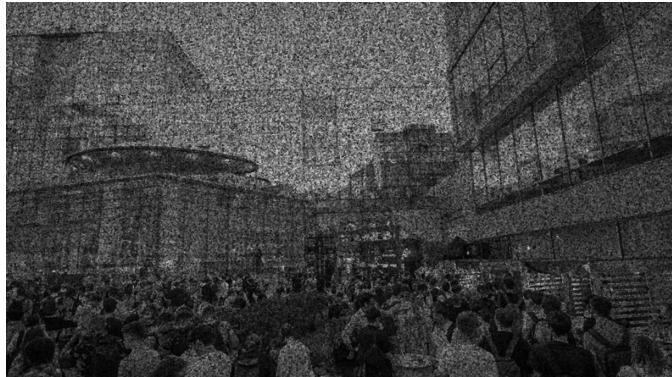
Рис. 80: Импульсный шум + фильтр ($k = 3$, $rank = 4$)Рис. 81: Аддитивный шум + фильтр ($k = 3$, $rank = 4$)Рис. 82: Мульти-ный шум + фильтр ($k = 3$, $rank = 4$)Рис. 83: Спекл-шум + фильтр ($k = 3$, $rank = 4$)Рис. 84: Гауссов шум + фильтр ($k = 3$, $rank = 4$)Рис. 85: Шум квантования + фильтр ($k = 3$, $rank = 4$)

Рис. 86: Импульсный шум + фильтр ($k = 3$, $rank = 7$)Рис. 87: Аддитивный шум + фильтр ($k = 3$, $rank = 7$)Рис. 88: Мульти-ный шум + фильтр ($k = 3$, $rank = 7$)Рис. 89: Спекл-шум + фильтр ($k = 3$, $rank = 7$)Рис. 90: Гауссов шум + фильтр ($k = 3$, $rank = 7$)Рис. 91: Шум квантования + фильтр ($k = 3$, $rank = 7$)

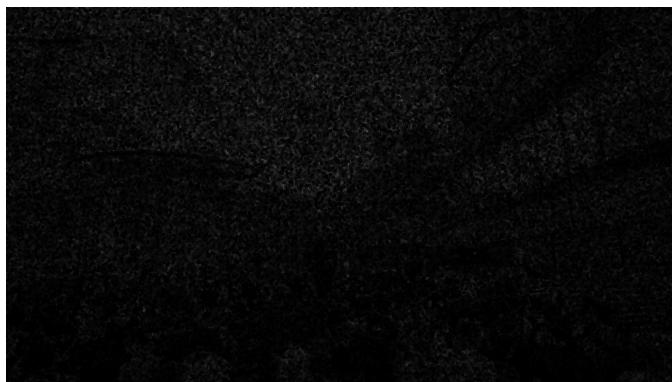
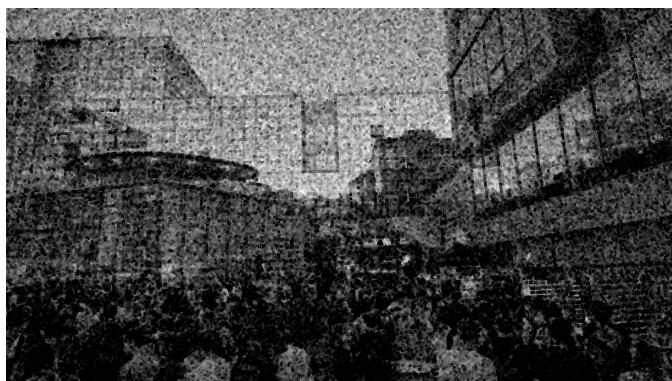
Рис. 92: Импульсный шум + фильтр ($k = 5$, $rank = 1$)Рис. 93: Аддитивный шум + фильтр ($k = 5$, $rank = 1$)Рис. 94: Мульти-шум + фильтр ($k = 5$, $rank = 1$)Рис. 95: Спекл-шум + фильтр ($k = 5$, $rank = 1$)Рис. 96: Гауссов шум + фильтр ($k = 5$, $rank = 1$)Рис. 97: Шум квантования + фильтр ($k = 5$, $rank = 1$)

Рис. 98: Импульсный шум + фильтр ($k = 5, rank = 11$)Рис. 99: Аддитивный шум + фильтр ($k = 5, rank = 11$)Рис. 100: Мульти-ний шум + фильтр ($k = 5, rank = 11$)Рис. 101: Спекл-шум + фильтр ($k = 5, rank = 11$)Рис. 102: Гауссов шум + фильтр ($k = 5, rank = 11$)Рис. 103: Шум кван-ия + фильтр ($k = 5, rank = 11$)

Рис. 104: Имп-ный шум + фильтр ($k = 5$, $rank = 21$)Рис. 105: Адди-ный шум + фильтр ($k = 5$, $rank = 21$)Рис. 106: Мульти-ный шум + фильтр ($k = 5$, $rank = 21$)Рис. 107: Спекл-шум + фильтр ($k = 5$, $rank = 21$)Рис. 108: Гауссов шум + фильтр ($k = 5$, $rank = 21$)Рис. 109: Шум кван-ия + фильтр ($k = 5$, $rank = 21$)

Винеровский фильтр

Винеровская фильтрация — фильтрация, использующая пиксельно-адаптивный метод Винера. Метод оценивает локальную окрестность пикселя и собирает о них статистические данные, которые в дальнейшем используются для фильтрации:

$$\left[\mu = \frac{1}{n \cdot m} = \sum_{i=0}^m \sum_{j=0}^n I(i, j) \quad \sigma^2 = \frac{1}{n \cdot m} = \sum_{i=0}^m \sum_{j=0}^n I^2(i, j) - \mu^2 \right] \Rightarrow I_f = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2} (I(x, y) - \mu)$$

Здесь μ — среднее в окрестности, σ^2 — дисперсия, ν^2 — дисперсия шума. В этом фильтре мы, как и в предыдущих, регулируем только размер окна. Посмотрим на получившиеся с фильтрацией изображения:



Рис. 110: Импульсный шум + фильтр ($k = 3$)



Рис. 111: Аддитивный шум + фильтр ($k = 3$)



Рис. 112: Мульти-ний шум + фильтр ($k = 3$)

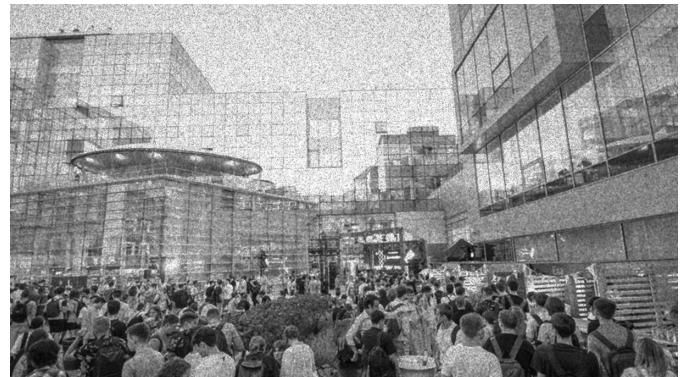


Рис. 113: Спекл-шум + фильтр ($k = 3$)



Рис. 114: Гауссов шум + фильтр ($k = 3$)



Рис. 115: Шум квантования + фильтр ($k = 3$)



Рис. 116: Импульсный шум + фильтр ($k = 5$)



Рис. 117: Аддитивный шум + фильтр ($k = 5$)



Рис. 118: Мульти-ний шум + фильтр ($k = 5$)



Рис. 119: Спекл-шум + фильтр ($k = 5$)



Рис. 120: Гауссов шум + фильтр ($k = 5$)



Рис. 121: Шум квантования + фильтр ($k = 5$)

Рис. 122: Импульсный шум + фильтр ($k = 9$)Рис. 123: Аддитивный шум + фильтр ($k = 9$)Рис. 124: Мульти-ний шум + фильтр ($k = 9$)Рис. 125: Спекл-шум + фильтр ($k = 9$)Рис. 126: Гауссов шум + фильтр ($k = 9$)Рис. 127: Шум квантования + фильтр ($k = 9$)

Наблюдаем, как винеровская фильтрация хорошо работает с импульсным шумом и хоть и не гасит его «перцо-вую» составляющую полностью на светлых участках изображения, но делает это хорошо на тёмных участках. Также фильтр неплохо гасит мультипликативный шум и в частности спекл-шум.

Адаптивный медианный фильтр

Ещё одна модификация медианного фильтра, в котором окно адаптивно увеличивается в зависимости от результата фильтрации. Алгоритм подразумевает увеличение размера окна до тех пор, пока алгоритм найдёт медианное значение, не являющееся импульсным шумом, или пока не достигнет максимального размера окна, которое мы задали. У нас будет два скользящих окна: одно размера 3×3 , расширяющееся до размера $s_{max} = 7$ и второе размера 5×5 с $s_{max} = 17$.

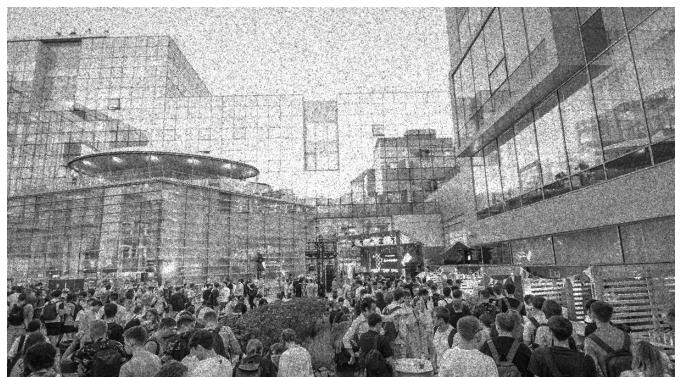
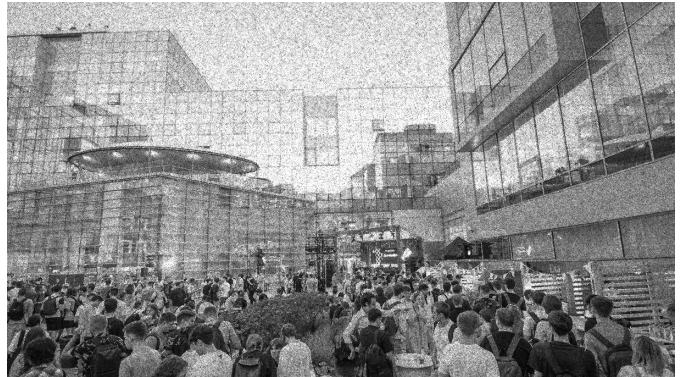
Рис. 128: Импульсный шум + фильтр ($s = 3, s_{max} = 7$)Рис. 129: Аддитивный шум + фильтр ($s = 3, s_{max} = 7$)Рис. 130: Мульти-шум + фильтр ($s = 3, s_{max} = 7$)Рис. 131: Спекл-шум + фильтр ($s = 3, s_{max} = 7$)Рис. 132: Гауссов шум + фильтр ($s = 3, s_{max} = 7$)Рис. 133: Шум квантования + фильтр ($s = 3, s_{max} = 7$)

Рис. 134: Импульсный шум + фильтр ($s = 5, s_{max} = 17$)Рис. 135: Аддитивный шум + фильтр ($s = 5, s_{max} = 17$)Рис. 136: Мульти-шум + фильтр ($s = 5, s_{max} = 17$)Рис. 137: Спекл-шум + фильтр ($s = 5, s_{max} = 17$)Рис. 138: Гауссов шум + фильтр ($s = 5, s_{max} = 17$)Рис. 139: Шум кван-ия + фильтр ($s = 5, s_{max} = 17$)

Как и медианный фильтр, аддитивный медианный фильтр хорошо удаляет импульсный шум. Есть плюсы — очертания объектов практически не размываются, но есть и минусы — не все белые и чёрные точки были удалены.

Высокочастотная фильтрация (выстраиваем границы понимания)

Говоря о высокочастотной фильтрации: она отлично подходит для выделения границ объектов, т.к. отфильтровывает низкие частоты (участки с низким изменением интенсивности пикселей) и оставляет только высокие частоты, т.е. контуры объектов.

Фильтр Робертса

И начнём с фильтра Робертса, который использует маски размером 2×2 , которые используются для нахождения градиента по осям OX и OY :

$$G_x = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \quad \text{или} \quad G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

И вот такой результат применения фильтра Робертса к оригинальному изображению мы получаем при использовании первой пары масок.



Рис. 140: Изображение с применённым фильтром Робертса

Получаем тёмное изображение, на котором серым хорошо отображены контуры изображения, похожие на выдавленный рельеф.

Фильтр Превитта (Прюйтта)

Фильтр Превитта использует маски размером 3×3 , которые используются для нахождения градиента по осям OX и OY :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

И результат фильтрации Превитта выглядит так:

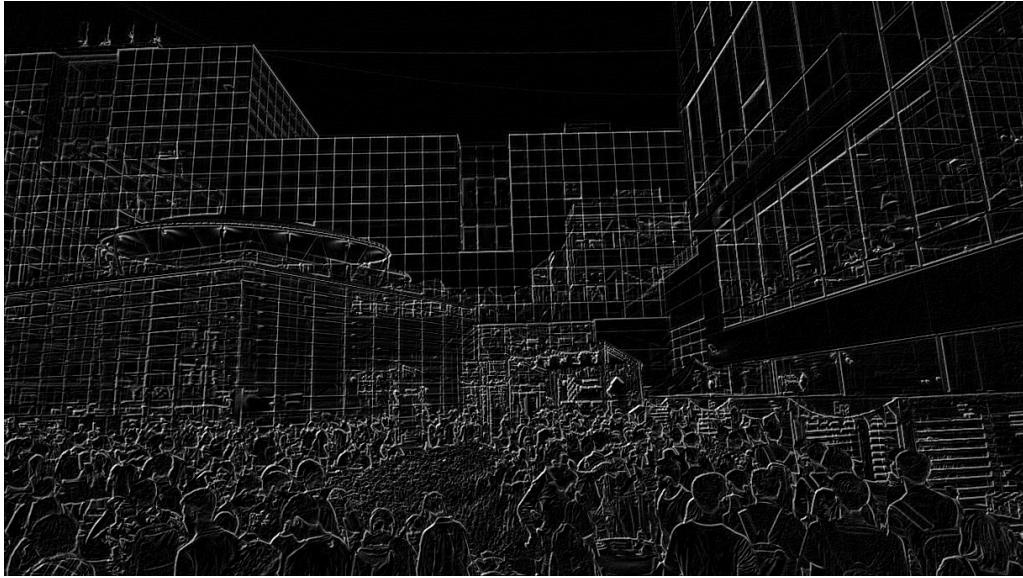


Рис. 141: Изображение с применённым фильтром Превитта

В сравнении с фильтром Робертса наблюдается большее количество контуров и сами по себе контуры более яркие.

Фильтр Собела

Фильтр Собела так же, как и предыдущий фильтр, использует маски 3×3 :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Применим фильтр к исходному изображению и посмотрим на результат:

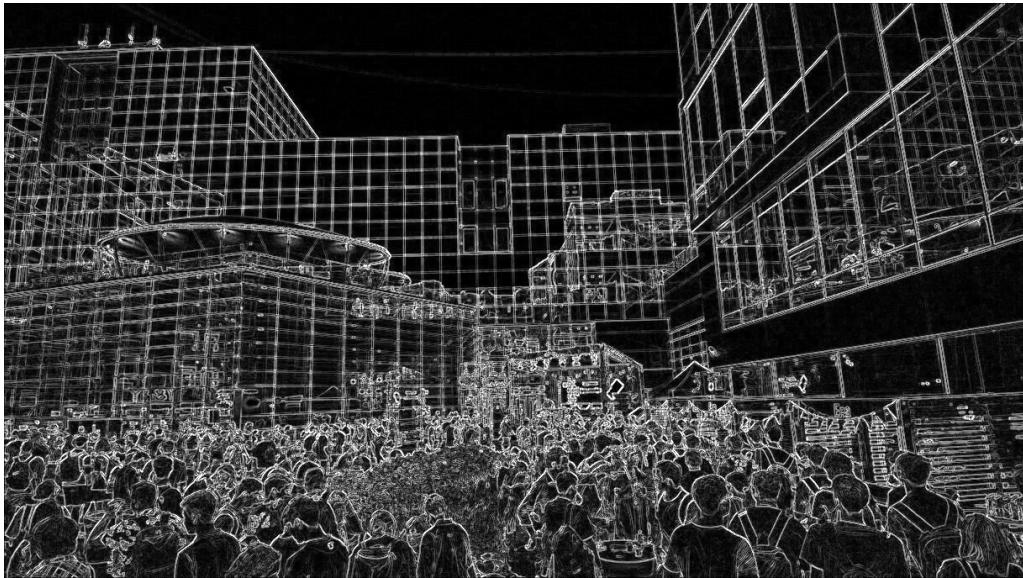


Рис. 142: Изображение с применённым фильтром Собела

Контуры объектов после применения фильтра Собела стали толще и их стало больше — уже видны шумы и некоторые незначительные детали, которые фильтр захватывает как контуры.

Фильтр Лапласа

Фильтр Лапласа отличается от предыдущих высокочастотных фильтров, которые мы рассматривали. Этот фильтр использует аппроксимацию вторых производных (а не первых, как это было с фильтрами выше). Обычно для фильтрации Лапласа используется одна общая маска:

$$w = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Посмотрим на то, как отрабатывает фильтр Лапласа на нашем изображении:

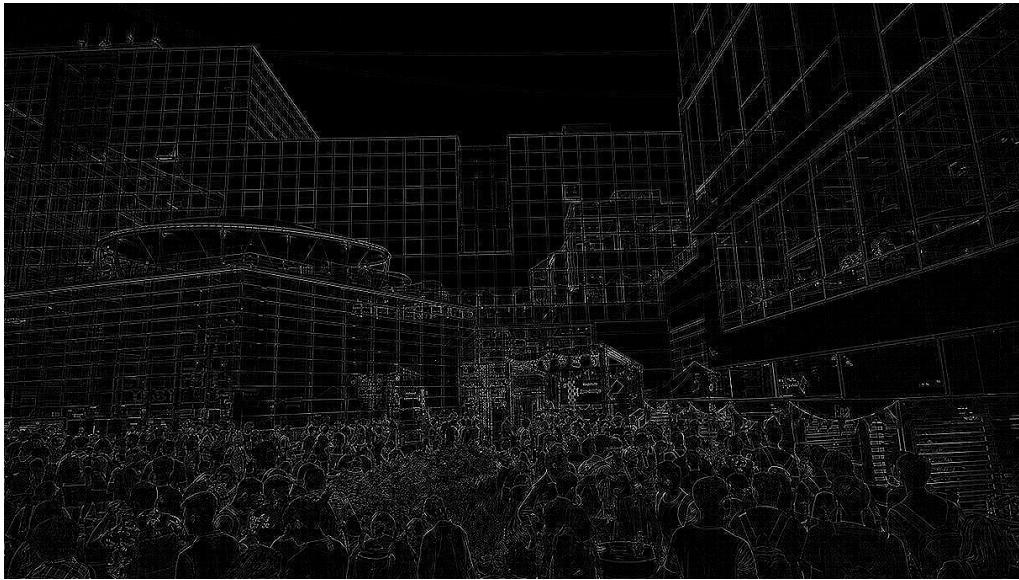


Рис. 143: Изображение с применённым фильтром Лапласа

Как мы видим, результат похож на фильтр Собела, но контуры чуть тоньше и в изображении меньше шумов. Однако же это всё ещё не идеальный вариант. Есть ли какой-то алгоритм, который даст нам точные контуры изображений без шумов и артефактов, которые появляются вследствие наложения маски?

Алгоритм Кэнни

Да, такой алгоритм есть — это алгоритм Кэнни. Это по сути комбинация сразу нескольких методик, которые приводят к тому, что мы получаем как бы перерисованное с нуля изображение, в котором перерисовали только контуры. Алгоритм Кэнни включает в себя и фильтр Гаусса для размытия, и фильтр Собела для определения значений модуля градиента всех пикселей, и двойную пороговую фильтрацию краевых пикселей. Все неоднозначные пиксели подавляются и шум в алгоритме Кэнни исключён. А благодаря возможности настраивать пропускную способность «фильтра», мы можем регулировать, насколько детальными должны быть контуры.

Выполним алгоритм с порогами модуля градиента $t_1 = 100$ и $t_2 = 200$ и посмотрим на результат:

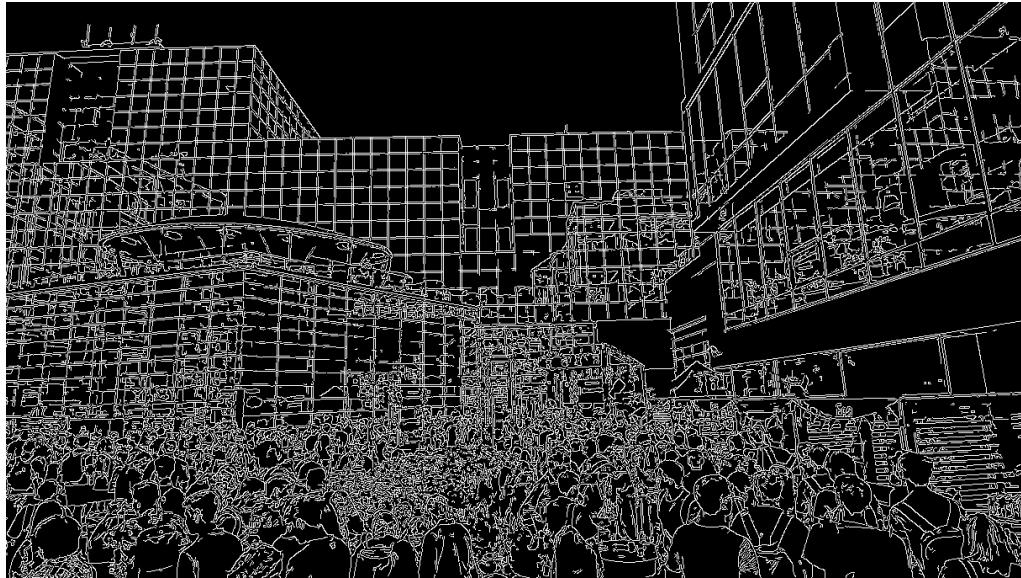


Рис. 144: Результат работы алгоритма Кэнни

Мы потеряли некоторые детали (к примеру, электропровода над крышами домов), но зато получили точные границы и контуры объектов.

Вывод

В этой лабораторной работе мы освоили методы наложения шумов и фильтрации изображений для того, чтобы вычленить из него важную информацию, начиная от среднего цвета всего изображения до выделения контуров объектов, что является нетривиальной задачей. Мы узнали о том, зачем нужны низкочастотные, нелинейные и высокочастотные фильтры, а также о том, какие из них лучше всего гасят определённые виды шумов. Ещё мы познакомились с возможностями `scikit-image` и `SciPy`, которые помогли нам реализовать работу фильтров.

Весь код, который мы использовали для генерации новых изображений, вы можно найти в [приватном гисте в GitHub](#).

Вопросы к защите

1. В чём заключаются основные недостатки адаптивных методов фильтрации изображений?

Адаптивные фильтры, несмотря на то, как хорошо они фильтруют импульсивный шум, чаще всего работают дольше по времени, т.к. корректируют свои параметры в ходе фильтрации. Поэтому, если встаёт вопрос о производительности, то, возможно, стоит выбрать фильтры, которые работают быстрее.
2. При каких значениях параметра Q контргармонический фильтр будет работать как арифметический, а при каких — как гармонический?

Как отмечено в теоретической справке в методическом пособии, при контргармонический фильтра работает как арифметический при $Q = 0$ и как гармонический при $Q = -1$.
3. Какими операторами можно выделить границы на изображении?

Дифференциальными операторами Робертса, Прюитта, Собела, а также оператором Лапласа.
4. Для чего на первом шаге выделения контуров, как правило, выполняется низкочастотная фильтрация?

Для того, чтобы погасить мелкие шумы и сместить акцент с мелких деталей, которые могли бы позже всплыть при выделении контуров. Так фильтр сможет хорошо выделить только самые важные контуры.