Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет Систем Управления и Робототехники

Дисциплина: Обработка цифровых изображений

ОТЧЁТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №3

Фильтрация и выделение контуров

Выполнил студент Топольницкий А.А.
Группа R4134c
Преподаватель Шаветов С.В.

Санкт-Петербург
2023 г.

Цель работы:

Освоение основных способов фильтрации изображений от шумов и выделения контуров

Краткое теоретическое обоснование:

Шум – разнообразные искажения на цифровых изображениях, обусловленные разного рода помехами. Основные виды шума:

- Импульсный шум – выбросы с большими отрицательными или положительными значениями. Белые точки – «соль», чёрные – «перец»;
- Аддитивный шум: $I_{new}(x, y) = I(x, y) + \eta(x, y)$,
  где $\eta(x, y)$ – шум с гауссовым или любым другим распределением функции плотности вероятности;
- Мультипликативный шум: $I_{new}(x, y) = I(x, y) * \eta(x, y)$ – зернистость фотоплёнки, ультразвуковые изображения;
- Гауссов (нормальный) шум;
- Шум квантования – зависит от выбранного шага квантования.

Фильтрация – процесс получения изображения на основе заданного, однако полученное по некоторым правилам. В первой части работы рассматриваются низкочастотные фильтры: арифметический усредняющий фильтр, фильтр Гаусса, медианный фильтр, минимальный, максимальный и фильтр Винера. Поскольку бóльшая часть работы посвящена фильтрации, давайте рассмотрим кратко формулы для каждого фильтра.

Арифметический усредняющий:

Усредняет значение интенсивности пикселя по окрестности с использованием маски с одинаковыми коэффициентами. Его формула приведена ниже.

$$I_{new}(x, y) = \frac{1}{m * n} \sum_{i=0}^{m} \sum_{j=0}^{n} I(i, j)\,, \qquad (1)$$

где $I_{new}(x, y)$ – значение интенсивности отфильтрованного пикселя, m, n – ширина и высота маски фильтра.

Геометрический усредняющий:

Используется для подавления высокочастотного аддитивного шума. Его формула:

$$I_{new}(x, y) = \left[\prod_{i=0}^{m} \prod_{j=0}^{n} I(i, j) \right]^{\frac{1}{m*n}} \quad (2)$$

Гармонический усредняющий:

Фильтр хорошо подавляет шумы типа «соль» и не работает с шумами типа «перец».

$$I_{new}(x, y) = e^{\frac{1}{m*n} \sum_{i=0}^{m} \sum_{j=0}^{n} \ln (I(i,j))} \quad (3)$$

Контргармонический усредняющий фильтр

Имеет порядок Q. При разных Q работает по-разному: если меньше 0, то подавляет шумы «соль», больше 0 – шумы «перец», равен 0 – это арифметический, равен – 1 – гармонический. Базируется на формуле:

$$I_{new}(x, y) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} I(i, j)^{Q+1}}{\sum_{i=0}^{m} \sum_{j=0}^{n} I(i, j)^{Q}} \quad (4)$$

Фильтр Гаусса

Позволяет учесть влияние пикселей, расположенных ближе к анализируемому, поскольку именно такие пиксели оказывают бóльшее влияние.

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}, \quad (5)$$

где σ – параметр, отвечающий за силу размытия. Чем больше, тем сильнее размывается изображение.

В работе также рассматриваются нелинейные фильтры, поскольку они позволяют лучше устранять импульсные помехи.

Медианная фильтрация – значения интенсивностей пикселей в окне представляются в виде вектора-столбца и сортируются по возрастанию.

Взвешенная медианная фильтрация – в маске используются весовые коэффициенты, чтобы отразить бóльшее влияние на результат фильтрации пикселей, расположенных ближе к фильтруемому элементу.

Адаптивная медианная фильтрация – отличается тем, что используется скользящее окно размера *s* x *s*, которое адаптивно увеличивается в зависимости от результата фильтрации. Размер окна увеличивается до тех пор, пока алгоритм не найдёт медианное значение (не шум) или пока не достигнет предельного размера окна.

Ранговая фильтрация – обобщение медианной фильтрации с порядком r. Если r = 1, то это *min-фильтр*; если r = N (число пикселей), то фильтр выбирает максимальное значение интенсивности – *max – фильтр*.

Винеровская фильтрация – использует пиксельно-адаптивный метод Винера, основанный на статистических данных, оценённых из локальной окрестности каждого пикселя.

Также коротко распишем методы высокочастотной фильтрации, применяемой в работе.

Суть их работы в усилении высокочастотных компонент (с сильным изменением интенсивности) и в ослаблении низкочастотных компонент. Используются для выделения перепадов интенсивностей и определения границ контуров на изображениях. Аппроксимируют вычисление производных по направлению, коэффициенты маски могут быть отрицательными и сумма всех коэффициентов равно нулю.

Фильтр Робертса – работает с маской 2 x 2, позволяет получить оценку градиента по направлениям.

Фильтр Превитта – две ортогональные маски размерами 3 x 3.

Фильтр Собела – аналогичен фильтру Превитта, но используются разные веса в масках.

Фильтр Лапласа – использует аппроксимацию вторых производных. Предыдущие использовали первую.

Алгоритм Кэнни – несколько шагов: 1) устраняет мелкие детали, сглаживая фильтром Гаусса; 2) использует оператор Собела для определения значений модуля градиента; 3) анализирует значение модулей градиента ортогональных исследуемому пикселей, если больше у исследуемого, то он краевой; 4) выполнение двойной пороговой фильтрации краевых пикселей; 5) подавление всех неоднозначных пикселей.

Ход работы

Лабораторная работа выполнялась в MATLAB.

Листинг 1. *Задание шумов разных типов*

```
clc
clear all

I = imread("nature.jpg");
I = rgb2gray(I);  % перевод в полутоновое

I_imp_noise = imnoise(I, 'salt & pepper');
I_speckle = imnoise(I,'speckle');
I_gauss = imnoise(I,"gaussian");
I_poisson = imnoise(I, 'poisson');

figure
subplot(3,2,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,2,2)
imshow(I_imp_noise)
title('Импульсный шум')
subplot(3,2,3)
imshow(I_speckle)
title('Мультипликативный')
subplot(3,2,4)
imshow(I_gauss)
title('Гауссов шум')
subplot(3,2,5)
imshow(I_poisson)
title('Шум квантования')
```

Комментарий: считываем наше изображение, переводим его в полутоновое, поскольку большего нам и не надо. С помощью функции *imnoise* и её параметров задаём импульсный шум, мультипликативный, Гауссов шум и шум квантования. Выводим на одной фигуре.

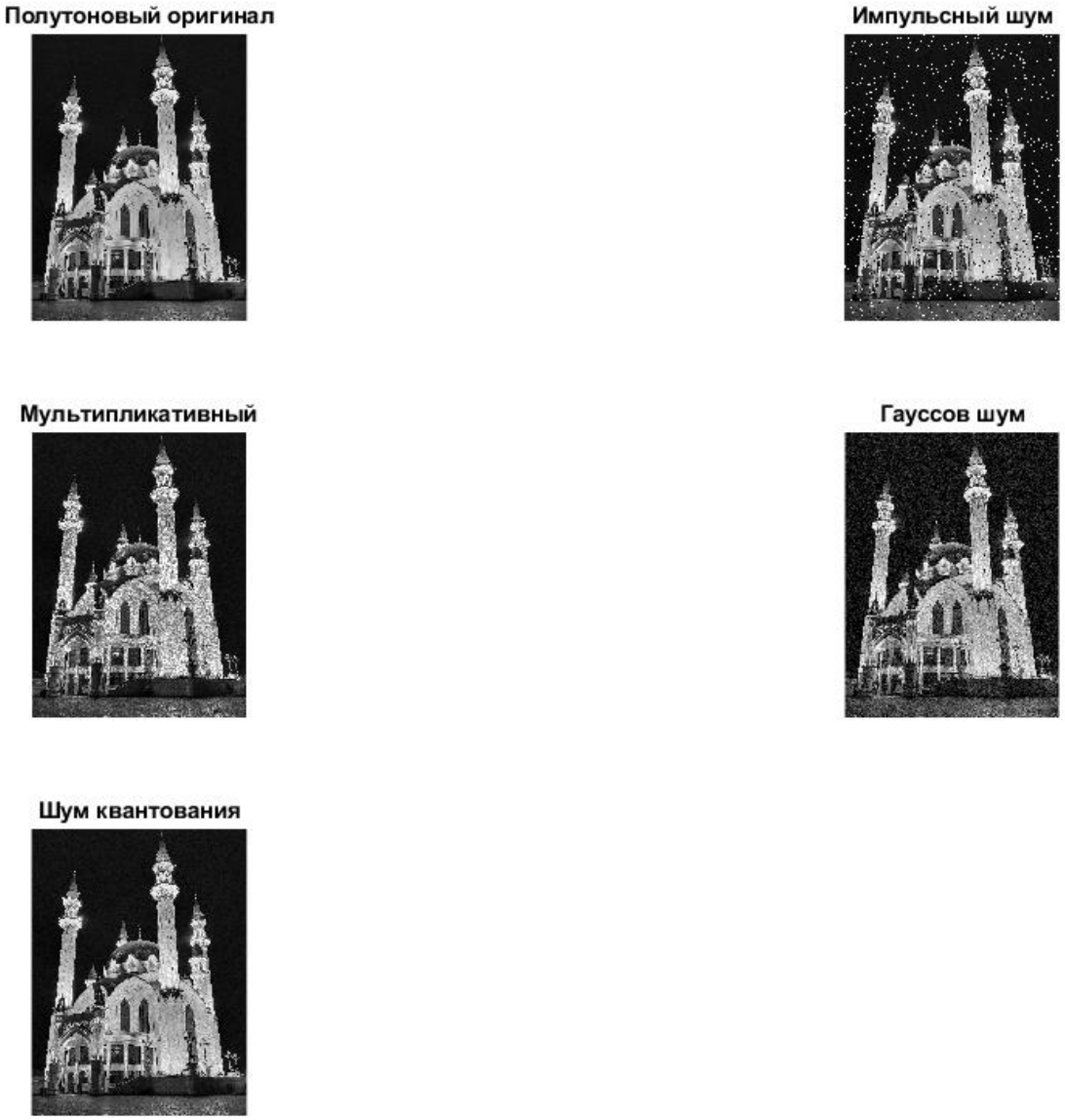

Рисунок 1. Задание шумов разных видов для исходного полутонового изображения

Листинг 2. *Пример низкочастотной фильтрации шума типа «соль» - «перец»*

```
I_f_mean = filter2(fspecial('average',3),I_imp_noise); % усредняющий
```

```
I_f_gauss = imgaussfilt(I_imp_noise); % Гауссов
I_f_median = medfilt2(I_imp_noise); % медианная
I_f_median_2 = ordfilt2(I_imp_noise, 5, ones(3,3)); % ранговый (обобщение медианного)
I_f_min = ordfilt2(I_imp_noise, 1, ones(3,3)); % минимальный
I_f_max = ordfilt2(I_imp_noise, 9, ones(3,3)); % максимальный
I_f_wiener = wiener2(I_imp_noise); % Винеровская фильтрация


figure

subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_imp_noise)
title('Импульсный шум')
subplot(3,3,3)
imshow(I_f_mean / 255)
title('Усредняющий')
subplot(3,3,4)
imshow(I_f_gauss)
title('Фильтр Гаусса')
subplot(3,3,5)
imshow(I_f_median)
title('Медианный')
subplot(3,3,6)
imshow(I_f_median_2)
title('Ранговый 50%')
subplot(3,3,7)
imshow(I_f_min)
title('Минимальный фильтр')
subplot(3,3,8)
imshow(I_f_max)
title('Максимальный фильтр')
subplot(3,3,9)
imshow(I_f_wiener)
```

Комментарий: описание работы каждого фильтра приведено в теории. Пробежимся по каждому. Усредняющий неплохо убирал как «соль», так и «перец», однако всё равно видна слабая «соль» на чёрном небе. Фильтр Гаусса не особо справился, только уменьшил количество шума. Медианный справился вполне неплохо как ранговый 50%, что и не удивительно, так как это одно и то же. Минимальный и максимальный фильтры убрали либо «соль», либо «перец» соответственно, но шум остался. Фильтр Винера отработал так себе, убрав лишь часть шумов.

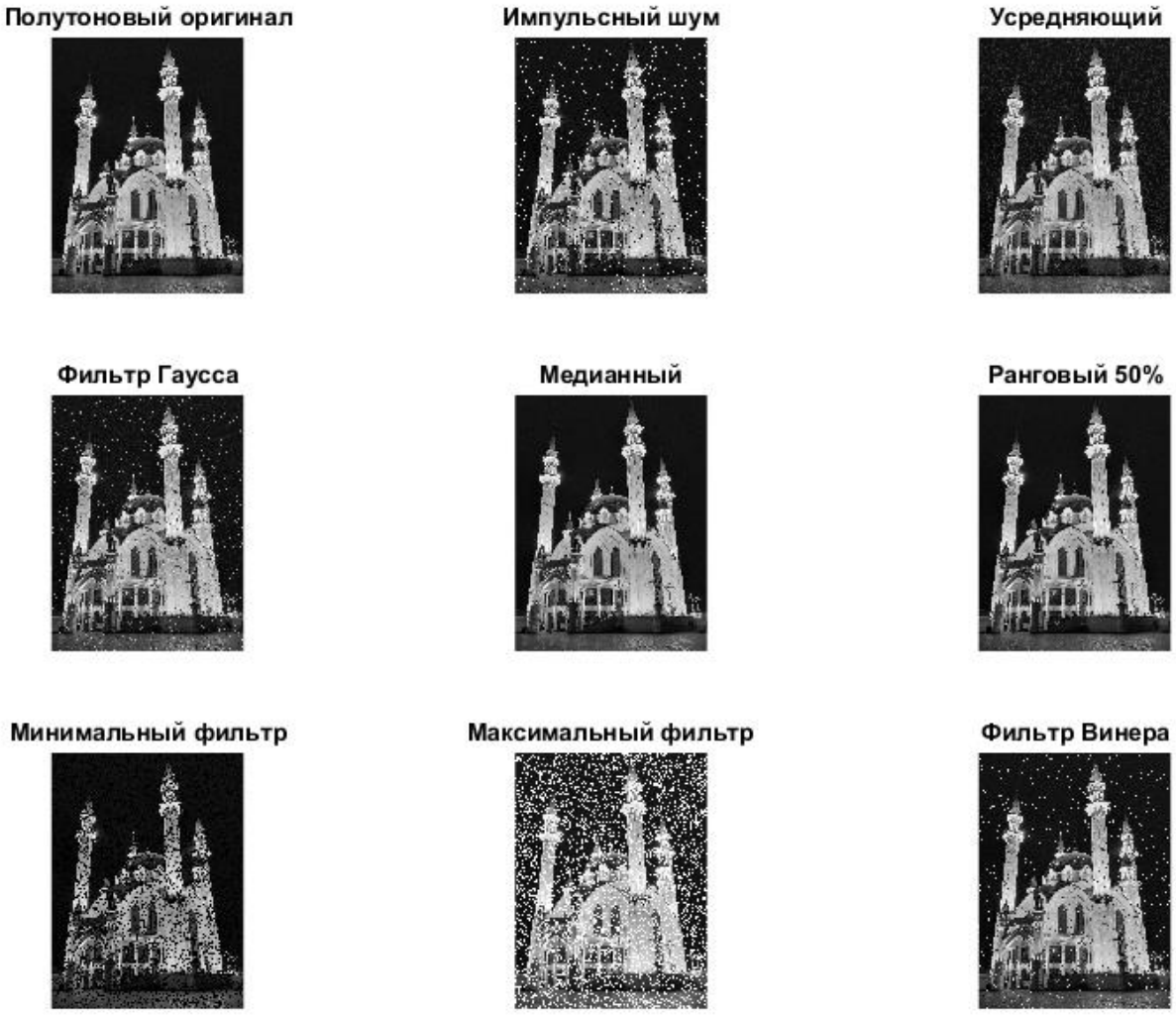

Рисунок 2. Низкочастотная фильтрация шума типа «соль» - «перец»

Листинг 3. *Пример низкочастотной фильтрации мультипликативного шума*

```
I_f_mean = filter2(fspecial('average',3),I_speckle); % усредняющий
```

```
I_f_gauss = imgaussfilt(I_speckle); % Гауссов
I_f_median = medfilt2(I_speckle); % медианная
I_f_median_2 = ordfilt2(I_speckle, 5, ones(3,3)); % ранговый (обобщение медианного)
I_f_min = ordfilt2(I_speckle, 1, ones(3,3)); % минимальный
I_f_max = ordfilt2(I_speckle, 9, ones(3,3)); % максимальный
I_f_wiener = wiener2(I_speckle); % Винеровская фильтрация

figure
subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_speckle)
title('Мультипликативный')
subplot(3,3,3)
imshow(I_f_mean / 255)
title('Усредняющий')
subplot(3,3,4)
imshow(I_f_gauss)
title('Фильтр Гаусса')
subplot(3,3,5)
imshow(I_f_median)
title('Медианный')
subplot(3,3,6)
imshow(I_f_median_2)
title('Ранговый 50%')
subplot(3,3,7)
imshow(I_f_min)
title('Минимальный фильтр')
subplot(3,3,8)
imshow(I_f_max)
title('Максимальный фильтр')
subplot(3,3,9)
imshow(I_f_wiener)
title('Фильтр Винера')
```

Комментарий: с мультипликативным шумом фильтры справились неплохо, особенно усредняющий и минимальный с максимальным, хоть последние два и изменили интенсивности – затемнил и засветил соответственно. Фильтр Гаусса для такого шума отработал лучше, чем для импульсного, но с усредняющим не сравнился. Фильтр Винера как и медианный отработали средне.

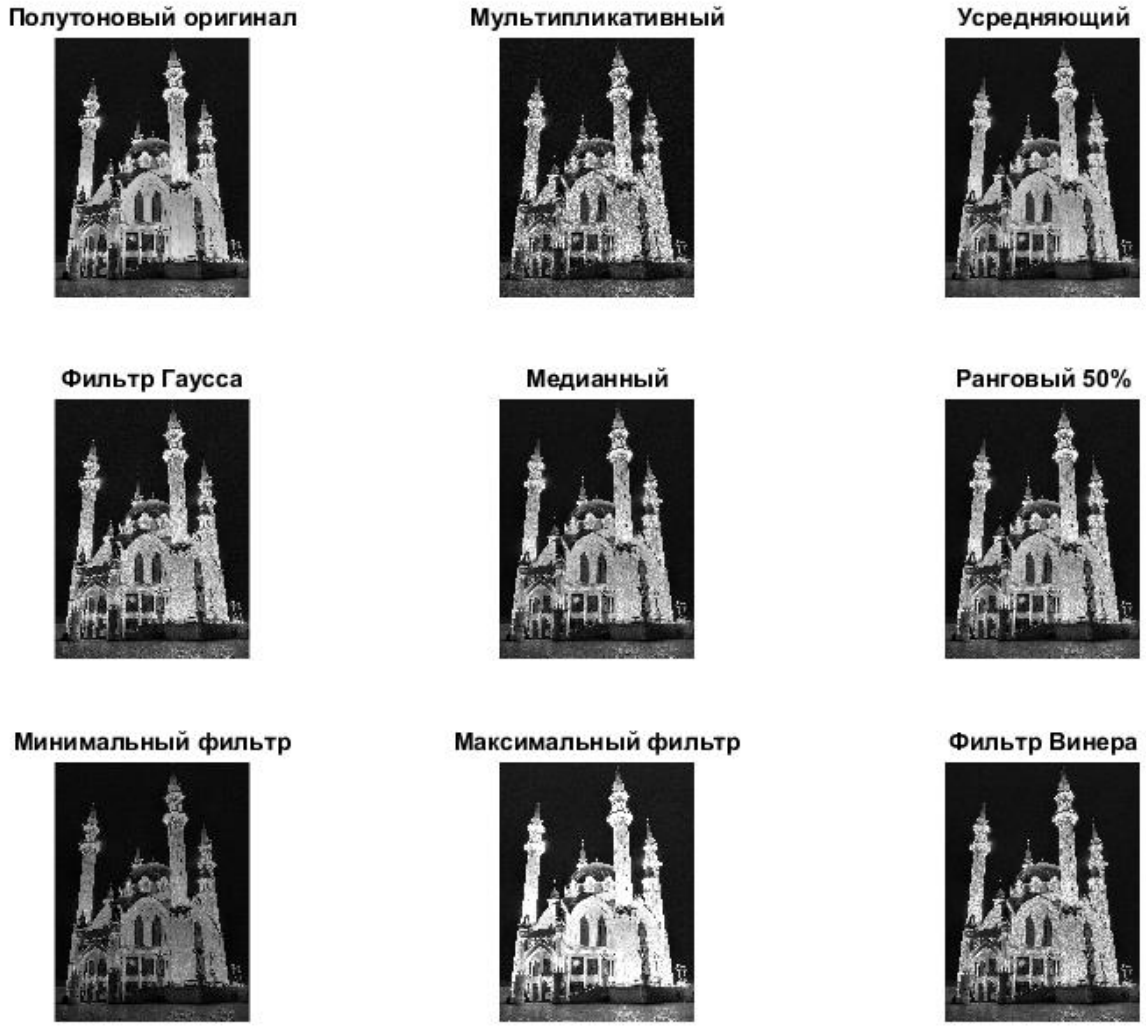

Рисунок 3. Пример низкочастотной фильтрации мультипликативного шума

Листинг 4. *Пример низкочастотной фильтрации Гауссова шума*

```
I_f_mean = filter2(fspecial('average',3),I_gauss); % усредняющий
I_f_gauss = imgaussfilt(I_gauss); % Гауссов
I_f_median = medfilt2(I_gauss); % медианная
```

```
I_f_median_2 = ordfilt2(I_gauss, 5, ones(3,3)); % ранговый (обобщение медианного)
I_f_min = ordfilt2(I_gauss, 1, ones(3,3)); % минимальный
I_f_max = ordfilt2(I_gauss, 9, ones(3,3)); % максимальный
I_f_wiener = wiener2(I_gauss); % Винеровская фильтрация

figure

subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_gauss)
title('Гауссов шум')
subplot(3,3,3)
imshow(I_f_mean / 255)
title('Усредняющий')
subplot(3,3,4)
imshow(I_f_gauss)
title('Фильтр Гаусса')
subplot(3,3,5)
imshow(I_f_median)
title('Медианный')
subplot(3,3,6)
imshow(I_f_median_2)
title('Ранговый 50%')
subplot(3,3,7)
imshow(I_f_min)
title('Минимальный фильтр')
subplot(3,3,8)
imshow(I_f_max)
title('Максимальный фильтр')
subplot(3,3,9)
imshow(I_f_wiener)
title('Фильтр Винера')
```

Комментарий: с Гауссовым шумом фильтр Винера справился неплохо, как и усредняющий. Фильтр Гаусса убрал часть шума, в основном с собора, но на небе шум остался. Медианный тоже неплохо отработал.

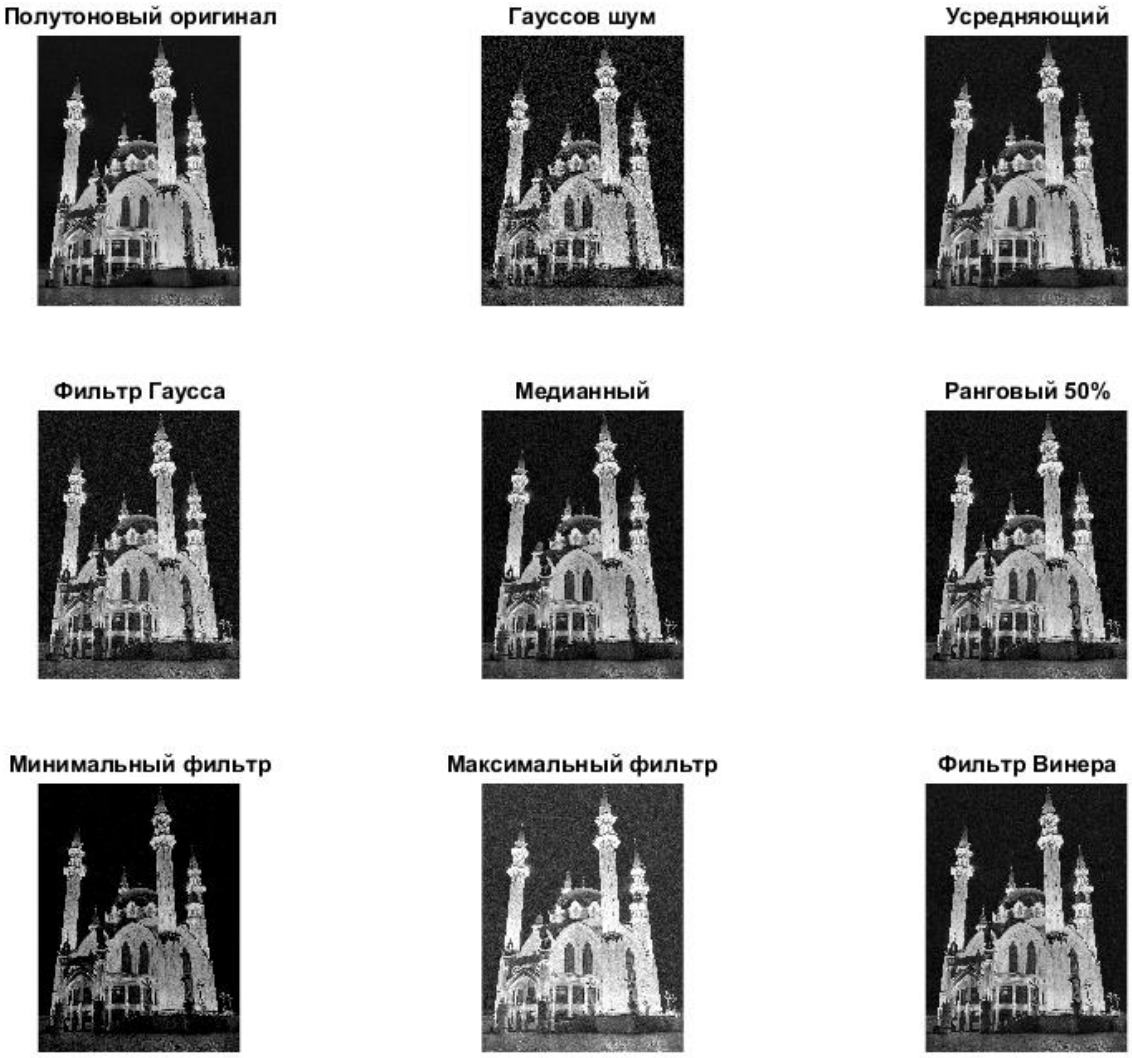

Рисунок 4. Пример низкочастотной фильтрации Гауссова шума

Листинг 5. *Пример низкочастотной фильтрации шума квантования*

```
I_f_mean = filter2(fspecial('average',3),I_poisson); % усредняющий
I_f_gauss = imgaussfilt(I_poisson); % Гауссов
I_f_median = medfilt2(I_poisson); % медианная
I_f_median_2 = ordfilt2(I_poisson, 5, ones(3,3)); % ранговый (обобщение медианного)
I_f_min = ordfilt2(I_poisson, 1, ones(3,3)); % минимальный
I_f_max = ordfilt2(I_poisson, 9, ones(3,3)); % максимальный
```

```
I_f_wiener = wiener2(I_poisson); % Винеровская фильтрация


figure
subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_poisson)
title('Шум квантования')
subplot(3,3,3)
imshow(I_f_mean / 255)
title('Усредняющий')
subplot(3,3,4)
imshow(I_f_gauss)
title('Фильтр Гаусса')
subplot(3,3,5)
imshow(I_f_median)
title('Медианный')
subplot(3,3,6)
imshow(I_f_median_2)
title('Ранговый 50%')
subplot(3,3,7)
imshow(I_f_min)
title('Минимальный фильтр')
subplot(3,3,8)
imshow(I_f_max)
title('Максимальный фильтр')
subplot(3,3,9)
imshow(I_f_wiener)
title('Фильтр Винера')
```

Комментарий: с шумом квантования справились все, но, справедливости ради, стоит заметить, что сам шум на фото не так уж бросается в глаза.

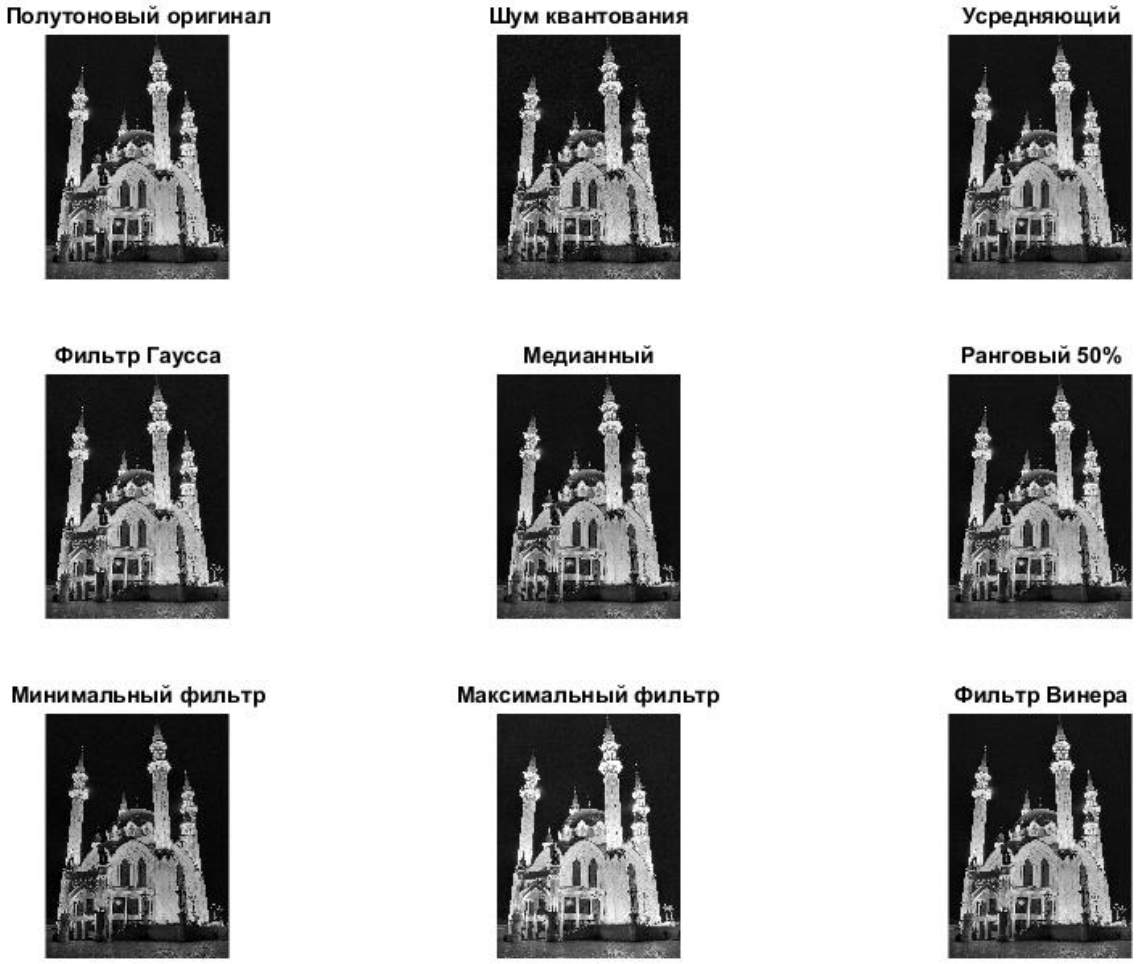

Рисунок 5. Пример низкочастотной фильтрации шума квантования

Листинг 6. *Реализация контргармонического усредняющего фильтра с различными значениями параметра Q*

```
function J = CHMF(I,m,q) % исходное изображение, размер маски, значение Q
I = im2double(I);
J = zeros(size(I));  % задаём пустой массив размером исходного изображения
mask_base = padarray(I,[floor(m / 2) floor(m / 2)],'symmetric'); % симметричным образом вокруг
% нашего массива добавляем элементы, которые симметричны границе

for i = 1:size(I,1)
   for j = 1:size(I,2)
      mask = mask_base(i : i + (m - 1), j : j + (m - 1)); % подаём в нашу функцию числа
      % от 1 до m,
      numerator = sum(mask.^(q + 1), 'all'); % считаем числитель
```

```
        deno = sum(mask.^(q), 'all'); % знаменатель
        if deno == 0  % если знаменатель 0, то, чтобы избежать деление на 0,
        % берём 0
            J(i,j) = 0;
        else
            J(i,j) = numerator / deno;
        end
    end
end

end

I = imread("church.jpg");
I = rgb2gray(I);  % перевод в полутоновое
I = imrotate(I,-90);

I_imp_noise = imnoise(I, 'salt & pepper');
I_speckle = imnoise(I,'speckle');
I_gauss = imnoise(I,"gaussian");
I_poisson = imnoise(I, 'poisson');

J_noise = CHMF(I_imp_noise,3,1.5);
J_speckle = CHMF(I_speckle,3,1.5);
J_gaussian = CHMF(I_gauss,3,1.5);
J_poisson = CHMF(I_poisson,3,1.5);


figure

subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_imp_noise)
title('Импульсный шум')
subplot(3,3,3)
imshow(I_speckle)
title('Мультипликативный')
subplot(3,3,4)
```

```
imshow(I_gauss)
title('Гауссов шум')
subplot(3,3,5)
imshow(I_poisson)
title('Шум квантования')
subplot(3,3,6)
imshow(J_noise)
title('Импульсный шум после фильтра')
subplot(3,3,7)
imshow(J_speckle)
title('Мультипликативный шум после фильтра')
subplot(3,3,8)
imshow(J_gaussian)
title('Гауссов шум после фильтра')
subplot(3,3,9)
imshow(J_poisson)
title('Шум квантования после фильтра')
```

Комментарий: в целом, в листинге присутствуют комментарии, но можно обобщить. Была создана функция для расчёта, поскольку подразумевается многократное (16 раз) использование одного и того же кода и нет смысла увеличивать объёмы. На вход в функцию подаётся изображение с шумом, значение размера окна *m* и значение параметра *q* (это Q из формулы). На выходе получаем готовый массив интенсивностей. В самой функции переводим изображение в double, чтобы избежать ошибок с округлением. Для того, чтобы на границе работать с маской корректно, будем, как указано в лекции, симметрично отражать элементы – это возможно с помощью функции *padarray* и её параметра *'symmetric'*. В подфункцию указываем размер нашего массива, а дальше указываем размерность добавления (будем добавлять по 1 строчке и столбцу с каждой стороны, если m = 3). Дальше в цикле начинаем перебор по символам и реализуем сложение и возведение в степень для каждого значения интенсивности. Чтобы избежать деления на 0, будем

полагать, что если знаменатель равен 0, то интенсивность пикселя равна 0. Возможно, это не самое лучшее решение, поскольку дальше будут видны шумы типа «перец».

Ниже приведены результаты работы фильтра для каждого шума при разных значениях параметра Q = {1.5, -0.5, -1, 0}.

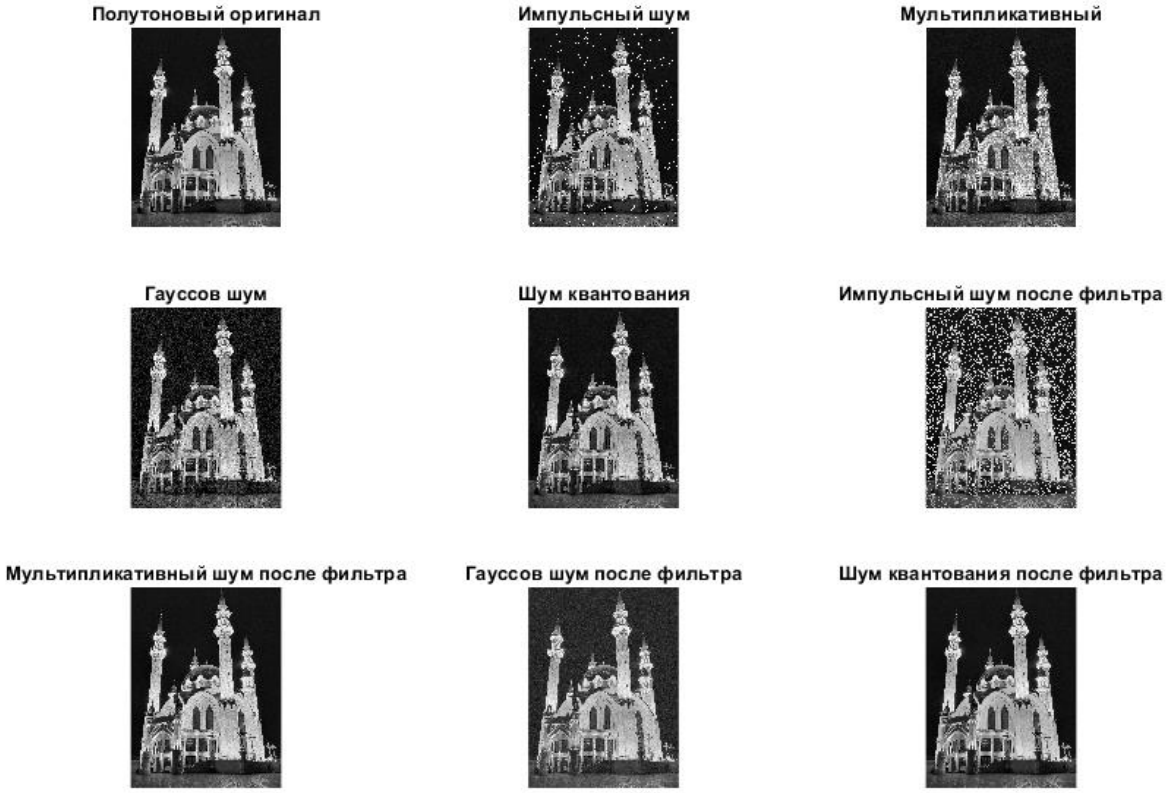

Рисунок 6. Использование контргармонического фильтра при параметре $Q = 1.5$

Комментарий: как и предполагалось, фильтр смог подавить шумы типа «перец» при заданном коэффициенте, однако шумы типа «соль» никуда не делись и их стало даже больше. С мультипликативным шумом фильтр справился неплохо, как и с шумом квантования.

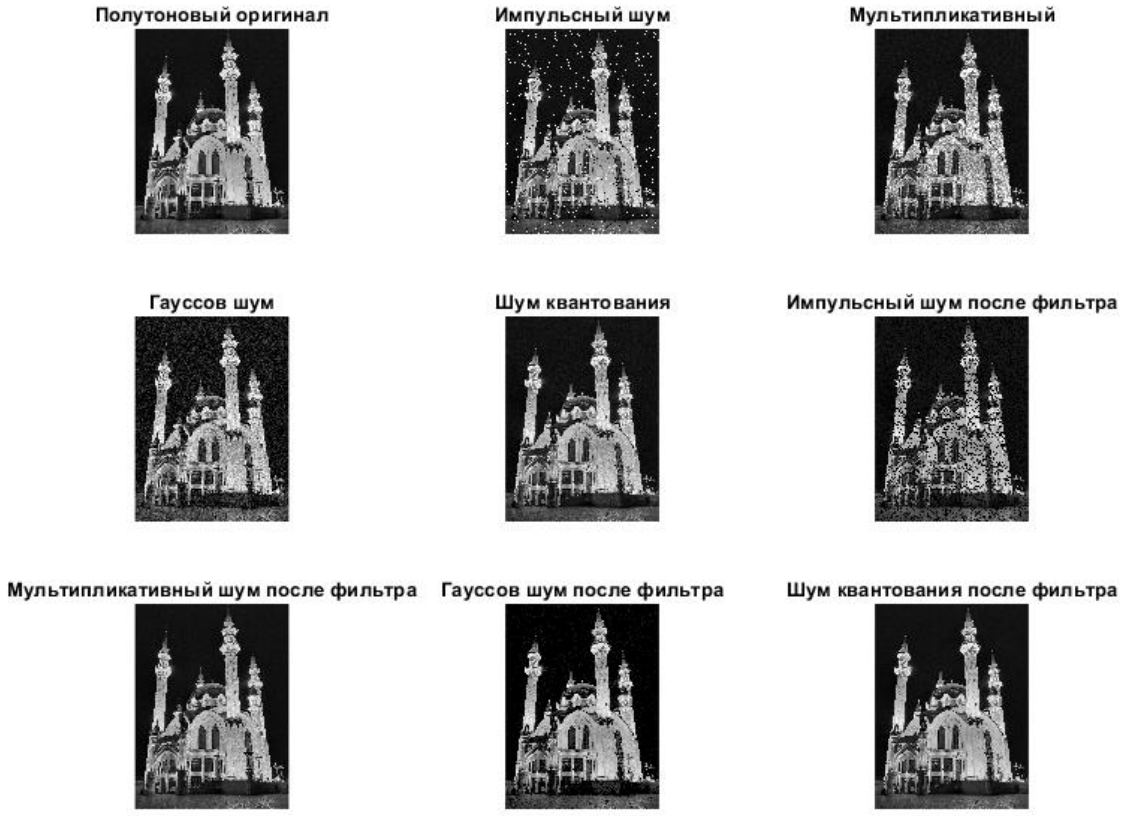

Рисунок 7. Использование контргармонического фильтра при параметре $Q = -0.5$

Комментарий: по рисунку видно, что, как и должно было быть, фильтр при отрицательном коэффициенте справился с шумом типа «соль», однако шума типа «перец» стало больше. С остальными шумами, особенно с Гауссовым шумом, фильтр справился хорошо.

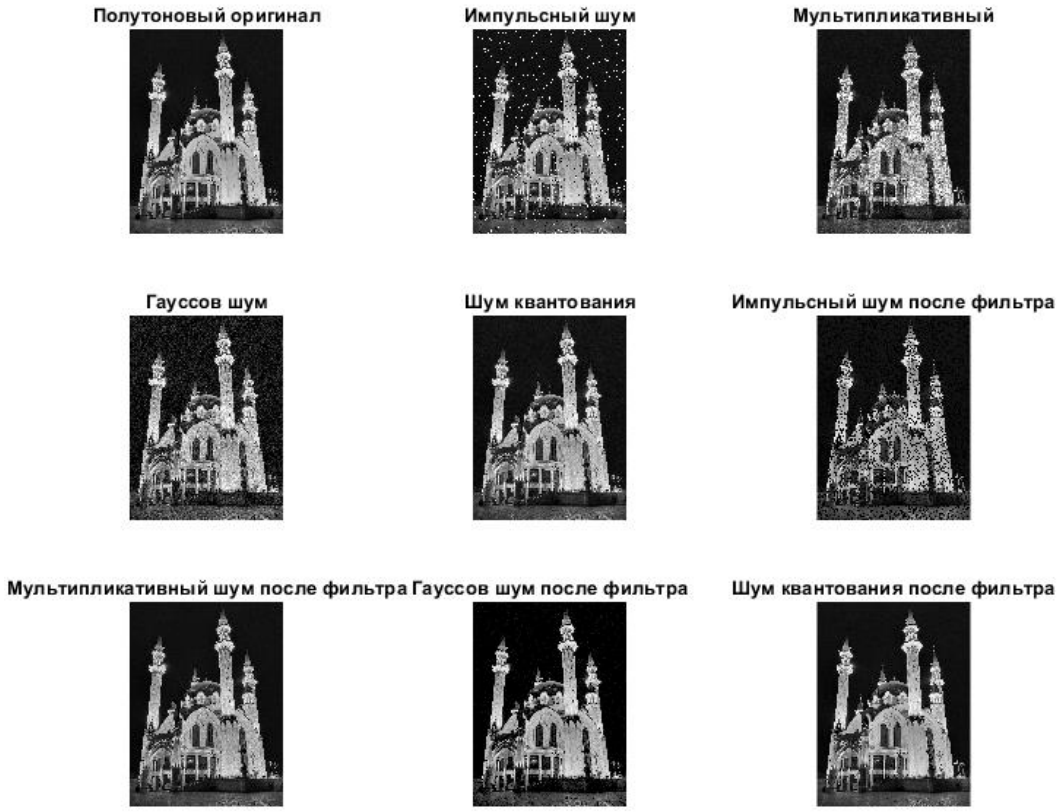

Рисунок 8. Использование контргармонического фильтра при параметре Q = - 1

Комментарий: при таком значении коэффициента видно, что фильтр также справился с шумами, но, как и следовало ожидать, шумы типа «перец» всё также остались.

Листинг 7. *Высокочастотная фильтрация для оригинального изображения*

```
figure;
subplot(2,3,1)
imshow(I)
title('Полутоновый оригинал')

I_roberst = edge(I,'roberts'); % работает с маской 2 x 2, 2 штуки
subplot(2,3,2)
imshow(I_roberst)
title('Фильтр Робертса')

I_prewitt = edge(I,'prewitt'); % 2 ортогональные маски размерами 3 x 3
subplot(2,3,3)
```

```
imshow(I_prewitt)
title('Фильтр Превитта')

I_sobel = edge(I,'sobel'); % отличается от Превитта наличием весов
subplot(2,3,4)
imshow(I_sobel)
title('Фильтр Собела')

I_laplas = edge(I,'log'); % использует аппроксимацию вторых производных
subplot(2,3,5)
imshow(I_laplas)
title('Фильтр Лапласа')

I_canny = edge(I,'canny');
subplot(2,3,6)
imshow(I_canny)
title('Алгоритм Кэнни')
```

Комментарий: краткое описание каждого алгоритма приведено в теории. Если посмотреть на картинку ниже, то можно увидеть, что в целом все фильтры дают общее очертание мечети и её минаретов, причём наилучший результат даже не у алгоритма Кэнни, а у фильтра Лапласа, поскольку после применения этого фильтра виден вход в мечеть, в то время как Кэнни хоть и выделил площадь перед мечетью, но вход в здание трудно различить.

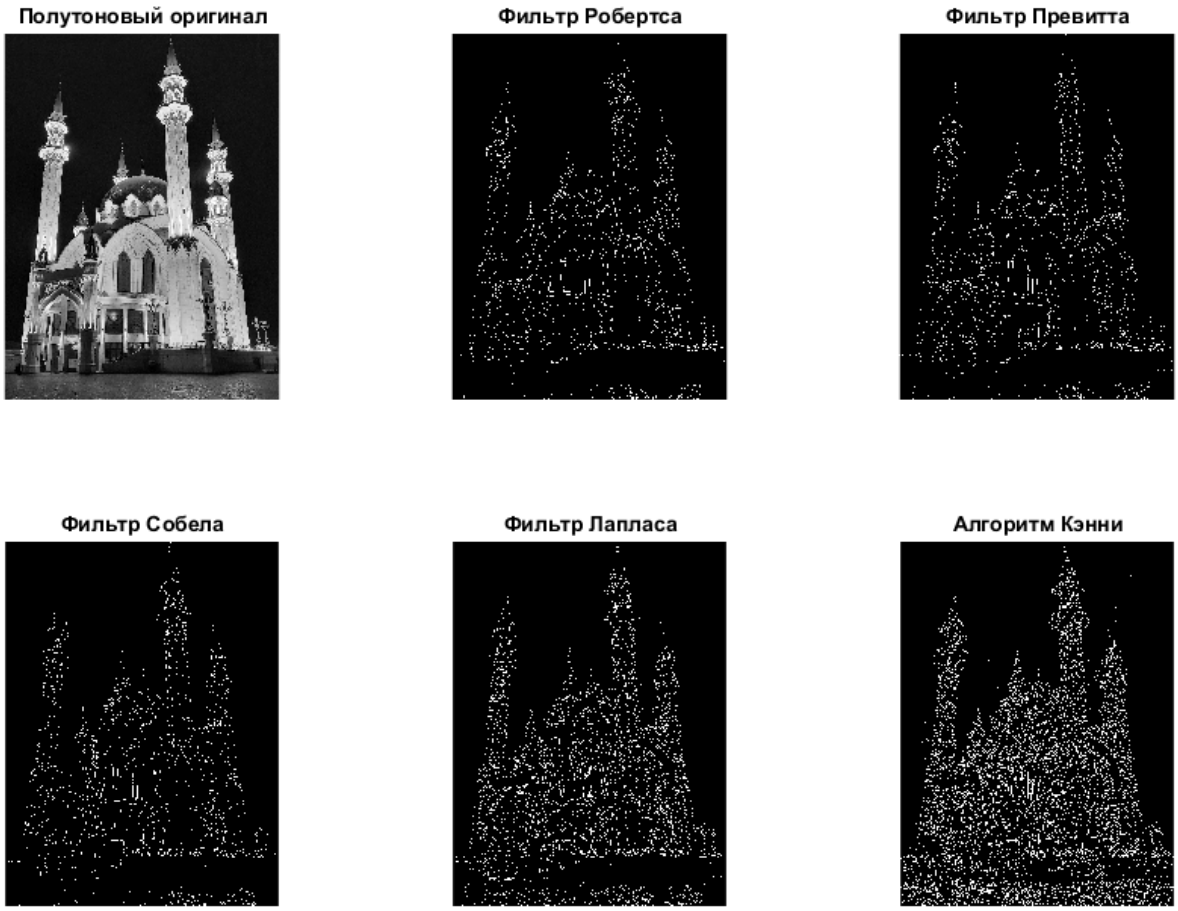

Рисунок 9. Использование высокочастотной фильтрации

Листинг 8. *Инвертированный вариант высокочастотной фильтрации*

```
figure;
subplot(2,3,1)
imshow(I)
title('Полутоновый оригинал')

I_roberst = edge(I,'roberts'); % работает с маской 2 x 2, 2 штуки
subplot(2,3,2)
imshow(~I_roberst)
title('Фильтр Робертса')

I_prewitt = edge(I,'prewitt'); % 2 ортогональные маски размерами 3 x 3
subplot(2,3,3)
imshow(~I_prewitt)
title('Фильтр Превитта')
```

```
I_sobel = edge(I,'sobel'); % отличается от Превитта наличием весов
subplot(2,3,4)
imshow(~I_sobel)
title('Фильтр Собела')
I_laplas = edge(I,'log'); % использует аппроксимацию вторых производных
subplot(2,3,5)
imshow(~I_laplas)
title('Фильтр Лапласа')
I_canny = edge(I,'canny');
subplot(2,3,6)
imshow(~I_canny)
title('Алгоритм Кэнни')
```

Комментарий: на инвертированном варианте хуже видны результаты работы циклов за исключением цикла Кэнни. Здесь нагляднее видно основание мечети и вход.

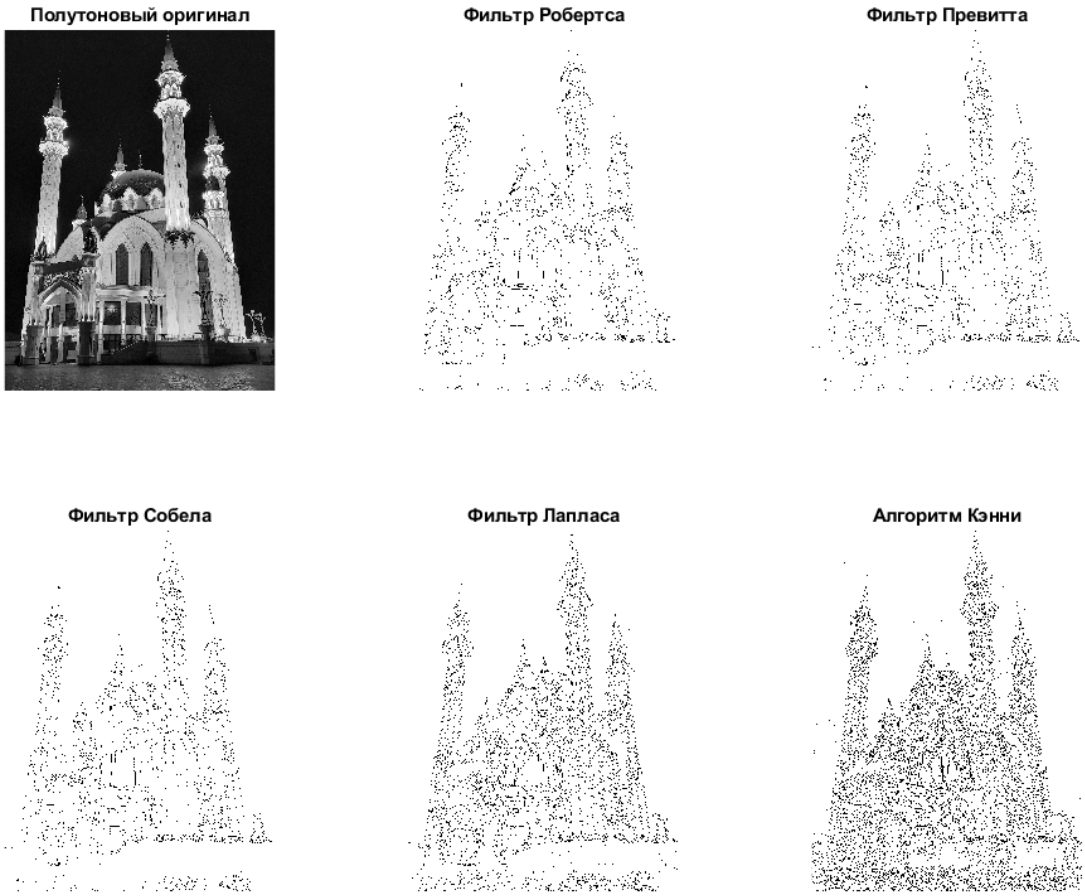

Рисунок 10. Инвертированный вариант высокочастотной фильтрации

Листинг 9. *Адаптивная медианная фильтрация*

```
function I_rez = AMF(I,s_max)

I = im2double(I);
s = 3;

[nrows, ncols] = size(I);

I_rez = I;
mask_base = padarray(I,[floor(s_max / 2) floor(s_max / 2)], 'symmetric');

for i = 1 : nrows
   for j = 1 : ncols
      mask = mask_base(i : i + s - 1, j : j + s - 1);
      z_min = min(mask(:));
      z_max = max(mask(:));
      z_med = median(mask(:));
      while s <= s_max
         A1 = z_med - z_min;
         A2 = z_med - z_max;
         if A1 > 0 && A2 < 0  % 1.a == True
            B1 = I(i,j) - z_min;
            B2 = I(i,j) - z_max;
            if B1 > 0 && B2 < 0
               I_rez(i,j) = I(i,j);
            else
               I_rez(i,j) = z_med;
            end
            break;
         else % 1.a == False
            s = s + 2;
            if s <= s_max
               continue
            else
               I_rez(i,j) = I(i,j);
            end
         end
      end
```

```
        s = 3;

    end
end
end


I = imread("church.jpg");
I = rgb2gray(I);  % перевод в полутоновое
I = imrotate(I,-90);
I = im2double(I);
I_imp_noise = imnoise(I, 'salt & pepper');
I_speckle = imnoise(I,'speckle');
I_gauss = imnoise(I,"gaussian");
I_poisson = imnoise(I, 'poisson');

output_imp = AMF(I_imp_noise, 11);
output_sp = AMF(I_speckle, 11);
output_ga = AMF(I_gauss, 11);
output_po = AMF(I_poisson, 11);

figure
subplot(3,3,1)
imshow(I)
title('Полутоновый оригинал')
subplot(3,3,2)
imshow(I_imp_noise)
title('Импульсный шум')
subplot(3,3,3)
imshow(I_speckle)
title('Мультипликативный')
subplot(3,3,4)
imshow(I_gauss)
title('Гауссов шум')
subplot(3,3,5)
imshow(I_poisson)
title('Шум квантования')
subplot(3,3,6)
imshow(output_imp)
```

```
title('Импульсный шум после фильтра')
subplot(3,3,7)
imshow(output_sp)
title('Мультипликативный шум после фильтра')
subplot(3,3,8)
imshow(output_ga)
title('Гауссов шум после фильтра')
subplot(3,3,9)
imshow(output_po)
title('Шум квантования после фильтра')
```

Комментарий: код также реализовывался через функцию. Само собой, функции должны идти в конце файла, но для наглядности сначала идёт функция. Последовательно идём по алгоритму и увеличиваем размер окна до тех пор (если не выполнены предыдущие условия), пока не достигнем предела. Тогда заканчиваем фильтрацию.

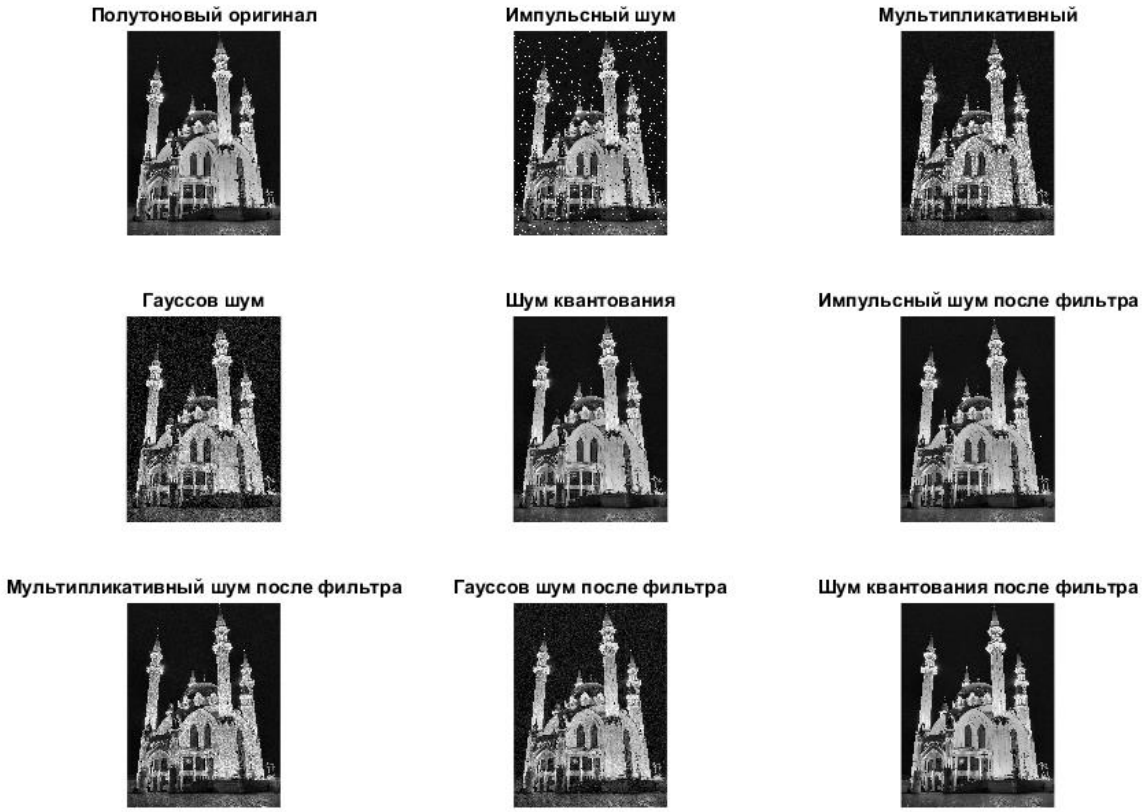

Рисунок 11. Применение адаптивного медианного фильтра

Комментарий: на рисунке видно, что написанный фильтр очень хорошо справился с импульсным шумом – буквально несколько белых и чёрных точек осталось, уменьшил количества Гауссова шума и шума квантования, а вот с мультипликативным можно сказать, что и не справился – да, шума стало меньше, но совсем немного.

Выводы. В ходе лабораторной работы были выполнены следующие шаги:

- На исходное изображение были наложены 4 типа шумов: импульсивный, мультипликативный, Гауссов и шум квантования;
- С помощью низкочастотной фильтрации изображение с каждым типом шума было отфильтровано с помощью усредняющего, Гауссова и контргармонического фильтра при разных параметрах Q. В целом, шум фильтровался, но не до конца;
- Был реализован контргармонический фильтр при разных значениях параметра Q – при одних он убирает «соль», при других «перец», но в целом остальные типы шума убирает неплохо;
- С помощью методов нелинейной фильтрации (медианная, ранговая, минимальная, максимальная, Винеровская) изображения с шумами также подверглись обработке. Наилучший средний результат показала медианная фильтрация и адаптивная медианная фильтрация;
- Была реализована функция для адаптивной медианной фильтрации, которая показала себя достаточно хорошо с импульсивными шумами, шумом квантования и средненько с Гауссовым шумом, однако с мультипликативным шумом функция справилась плохо.
- Также была проведена высокочастотная фильтрация для поиска контуров на изображении. Использовались фильтр Робертса, Превитта, Собела, Лапласа и алгоритм Кэнни. В целом, сама мечеть и её минареты различимы неплохо, но то ли картинка сложная, то ли ещё какие-то трудности, но ни один алгоритм не смог отчётливо обозначить детали.

Ответы на вопросы к лабораторной работе:

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

   Ответ: пожалуй, основные сложности заключаются в настройке размеров окна и вычислительных затратах, поскольку заранее сложно предсказать, какой размер нужен, а чтобы проверить гипотезу, придётся подождать расчётов. Также подбор весовых коэффициентов может не всегда давать желаемые результаты. Или, например Винеровская фильтрация базируется на статистических данных, а потому если вдруг какой-то шум, например, импульсный, попадёт в окно несколько раз, то результат фильтрации будет неудачным.

2. При каких значениях параметра Q контргармонический фильтр будет работать как арифметический, а при каких – как гармонический?

   Ответ: данный фильтр работает как арифметический при $Q = 0$, а как гармонический при $Q = -1$. При отрицательном значении параметра подавляет шумы «соль», а при положительном – «перец».

3. Какими операторами можно выделить границы изображений?

   Ответ: границы изображения можно выделить оператором Робертса (2 маски 2 х 2, вычисляется первая производная, градиент), оператор Превитта (2 ортогональные маски 3 х 3, вычисляется первая производная, градиент), оператор Собела (2 маски 3 х 3 с весами, вычисляется первая производная и градиент), оператор Лапласа (аппроксимирует вторую производную), алгоритм Кэнни (устраняет шум, расчёт градиента, поиск краевых пикселей).

4. Для чего на первом шаге выделения контуров, как правило, выполняется низкочастотная фильтрация?

   Ответ: поскольку данная фильтрация позволяет убрать высокочастотные компоненты и снизить уровень шума, это позволяет получить сглаженное или размытое изображение. Поскольку в процессе поиска контуров происходит расчёт градиентов, резкие скачки в интенсивности могут помешать корректному расчёту, что приведёт к некорректному отображению контуров.