|  |
| --- |
| УНИВЕРСИТЕТ ИТМО |
| Лабораторная работа №6 по дисциплине «Цифровая обработка сигналов» |
| Восстановление изображений в системе Matlab |
| Группа Р3402 |
| **Выполнила: Орлова Кристина Александровна** |
| **Преподаватель: Тропченко Андрей Александрович** |

|  |
| --- |
| *23.01.20* |

**Цель**

* Получение навыков формирования аддитивных шумов в системе Matlab
* Обучение восстановлению изображения после шумового воздействия

**Задание №1**

Используя данные таблицы 1, создайте функции, реализующие каждый из представленных видов шумов. Программный код функций с комментариями поместите в отчет.

Исходный код

**\*uniformNoise.m\***

function f = uniformNoise(m, n)

f = rand(m, n);

end

**\*gaussianNoise.m\***

function f = gaussianNoise(m, n)

a = 0;

b = 0.15;

f = a + b \* randn(m, n);

end

**\*logNormalNoise.m\***

function f = logNormalNoise(m, n)

a = 1;

b = 0.25;

f = a \* exp(b \* randn(m, n));

end

**\*relayNoise.m\***

function f = relayNoise(m , n)

b = 1;

f = sqrt(-b \* log(1 - rand(m, n)));

end

**\*exponentialNoise.m\***

function f = exponentialNoise(m, n)

a = 1;

f = -log(1 - rand(m, n)) / a;

end

**\*erlangNoise.m\***

function f = erlangNoise(m, n)

a = 2;

b = 5;

f = 0;

for i = 1:b

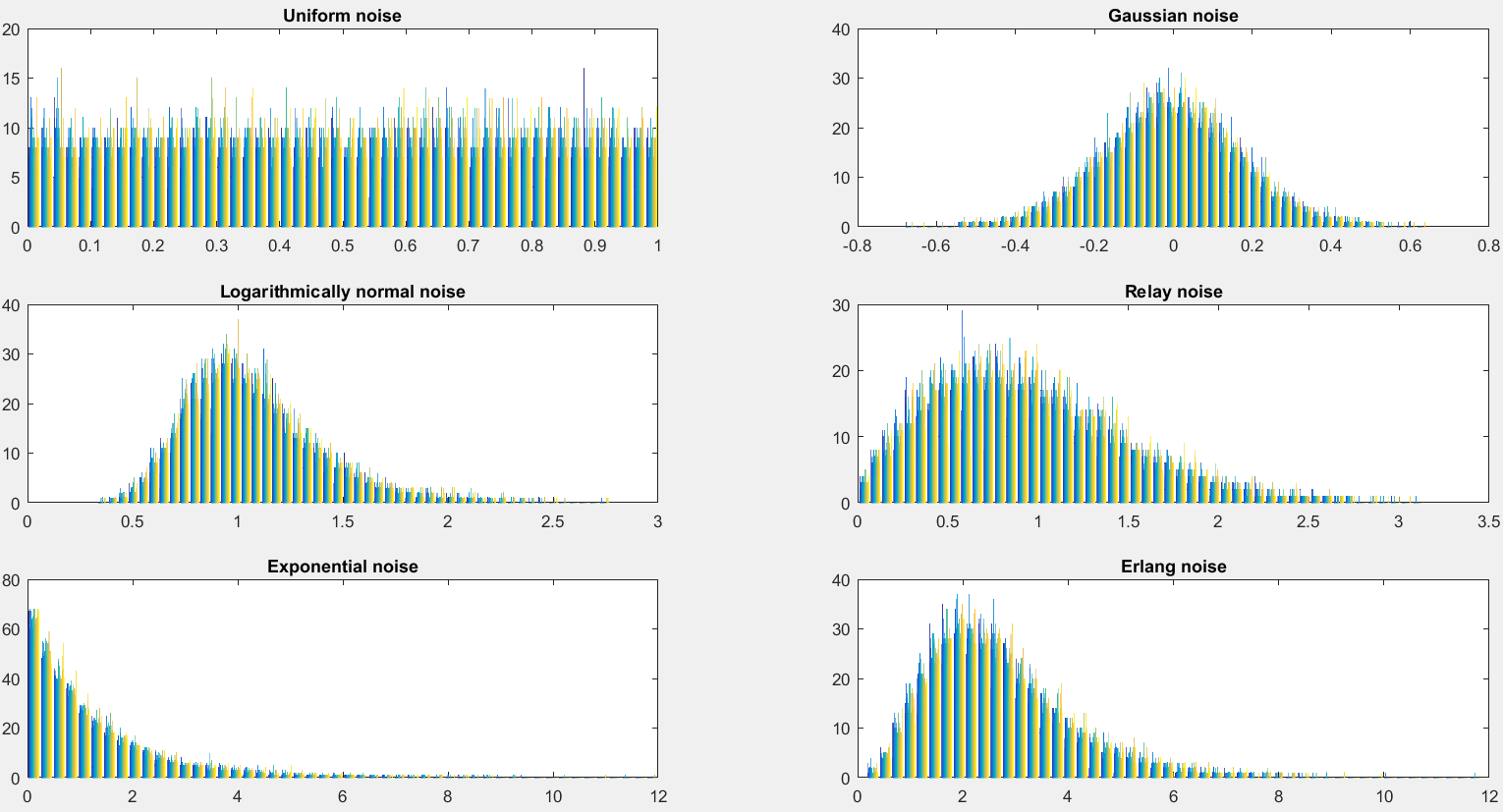
f = f - log(1 - rand(m, n)) / a;

end

end

**Задание №2**

Осуществите генерацию каждого типа шума размером 256×256 при помощи созданных функций. Для всех шумов, кроме шума Соль и перец, создайте гистограмму и поместите их в отчет. Количество корзин гистограммы должно иметь значение, равное 50.



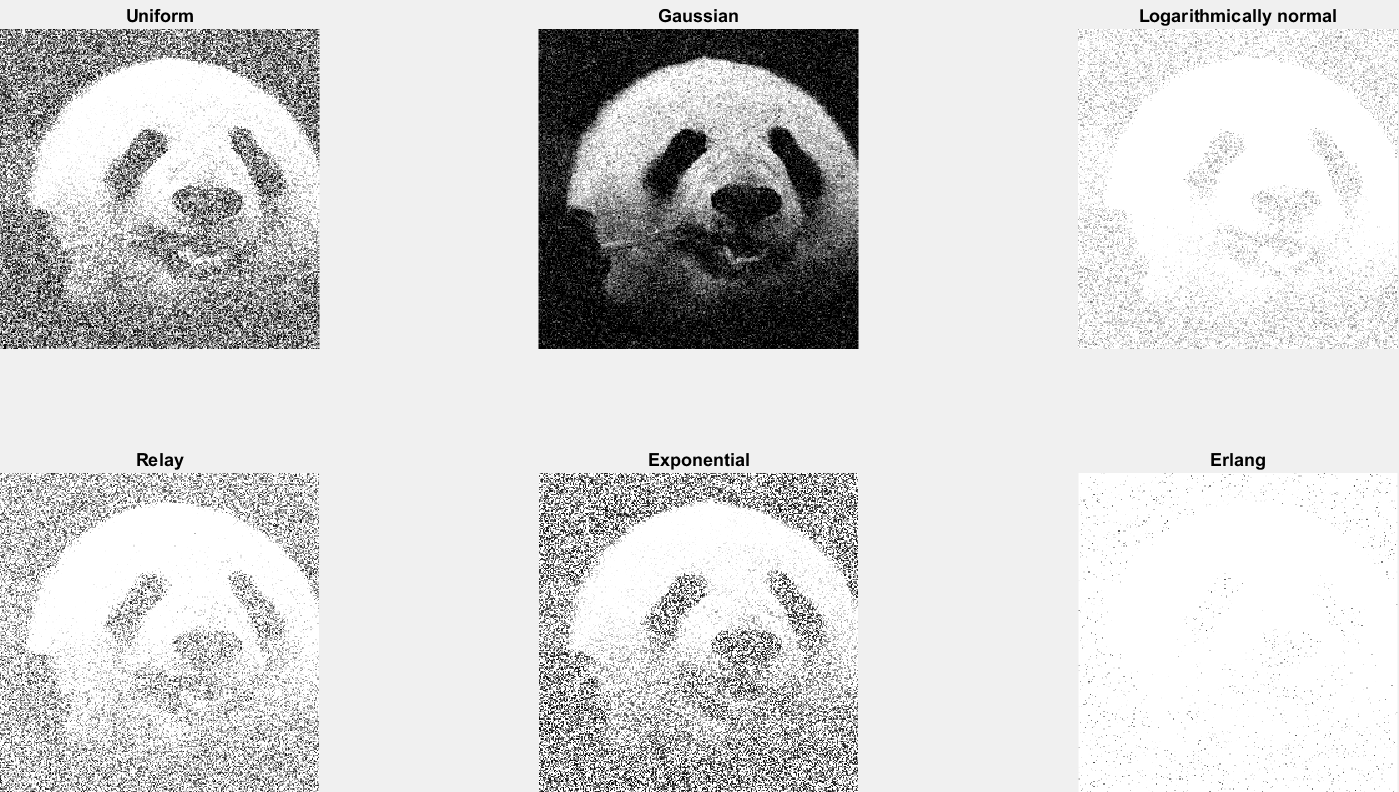
*Рисунок 1 - Гистограммы пространственных шумов*

**Задание №3**

Используя созданные функции и полученное для данной лабораторной работы изображение, осуществите зашумление полученного изображения при помощи всех видов шумов, представленных в таблице 1. Каждый полученный файл сохраните в виде отдельного изображения. Полученные изображения для каждого вида шума поместите в отчет, указав при этом вид шума, которому изображение соответствует.



*Рисунок 2 - Исходное изображение*



*Рисунок 3 - Зашумленные изображения*

**Задание №4**

Создайте функции, реализующие фильтрацию изображения каждым фильтром, формула которого представлена в таблице 2. При реализации конкретного фильтра используйте только его формулу. Код программы с комментариями поместите в отчет.

Исходный код

**\*arithmeticFilter.m\***

function f = arithmeticFilter(block)

f = sum(sum(block)) / numel(block);

end

**\*geometricFilter.m\***

function f = geometricFilter(block)

f = nthroot(prod(prod(block)), numel(block));

end

**\*harmonicFilter.m\***

function f = harmonicFilter(block)

f = numel(block) / sum(sum(1 ./ block));

end

**\*contravarianceFilter.m\***

function f = contravarianceFilter(block)

q = 1;

f = sum(sum(block .^ (q+1))) / sum(sum(block .^ q));

end

**\*medianFilter.m\***

function f = medianFilter(block)

f = median(median(block));

end

**\*maximumFilter.m\***

function f = maximumFilter(block)

f = max(max(block));

end

**\*minimumFilter.m\***

function f = minimumFilter(block)

f = min(min(block));

end

**\*middlePointFilter.m\***

function f = middlePointFilter(block)

f = (max(max(block)) + min(min(block))) / 2;

end

**\*alphaTruncatedFilter.m\***

function f = alphaTruncatedFilter(block)

d = 3;

f = sum(sum(block)) / (numel(block) - d);

end

**Задание №5**

* Каждое изображение, полученное в задании 3, подвергните фильтрации всеми фильтрами, используя созданные в задании № 4 функции. Используя функцию корреляции Пирсона corr2, определите качество каждого восстановленного изображения, сравнив его с оригиналом, выданным для выполнения данной работы. Код программы с комментариями поместите в отчет.
* Выполняя данное задание, заполните таблицу 3.4 и поместите ее в отчет. В каждую ячейку, находящуюся на пересечении строки с типом шума и столбца с типом фильтра необходимо вписать значение корреляции между оригиналом и восстановленным изображением. Жирным выделить значение наилучшего показателя корреляции Пирсона.
* После заполнения таблицы 3.4 представьте десять вариантов восстановленных изображений и укажите, какой тип фильтра был при этом использован.

Исходный код

**\*main.m\***

image = imread('panda.pgm');

figure, imshow(image);

image = im2double(image);

[uniform, gaussian, logNormal, relay, exponential, erlang] = getNoises(256, 256);

uniformNoised = image + uniform;

gaussianNoised = image + gaussian;

logNormalNoised = image + logNormal;

relayNoised = image + relay;

exponentialNoised = image + exponential;

erlangNoised = image + erlang;

noisedImages = {uniformNoised; gaussianNoised; logNormalNoised;

relayNoised; exponentialNoised; erlangNoised};

filters = getFilters();

filtersCount = length(filters);

imagesCount = length(noisedImages);

correlationCoefficients = zeros(filtersCount, imagesCount);

for j = 1:imagesCount

figure

for i = 1:filtersCount

finalImage = nlfilter(noisedImages{j}, [3 3], filters{i});

correlationCoefficients(i, j) = corr2(image, finalImage);

subplot(3,3,i), imshow(finalImage), title(filters{i, 2});

end

end

**\*getNoises.m\***

function [uniform, gaussian, logNormal, relay, exponential, erlang] = getNoises(m, n)

uniform = uniformNoise(m, n);

gaussian = gaussianNoise(m, n);

logNormal = logNormalNoise(m, n);

relay = relayNoise(m, n);

exponential = exponentialNoise(m, n);

erlang = erlangNoise(m, n);

end

**\*getFilters.m\***

function f = getFilters()

f = {

@arithmeticFilter, 'Arithmetic'

@geometricFilter, 'Geometric'

@harmonicFilter, 'Harmonic'

@contravarianceFilter, 'Contravariance'

@medianFilter, 'Median'

@maximumFilter, 'Maximum'

@minimumFilter, 'Minimum'

@middlePointFilter, 'Middle point'

@alphaTruncatedFilter, 'Alpha-truncated'

};

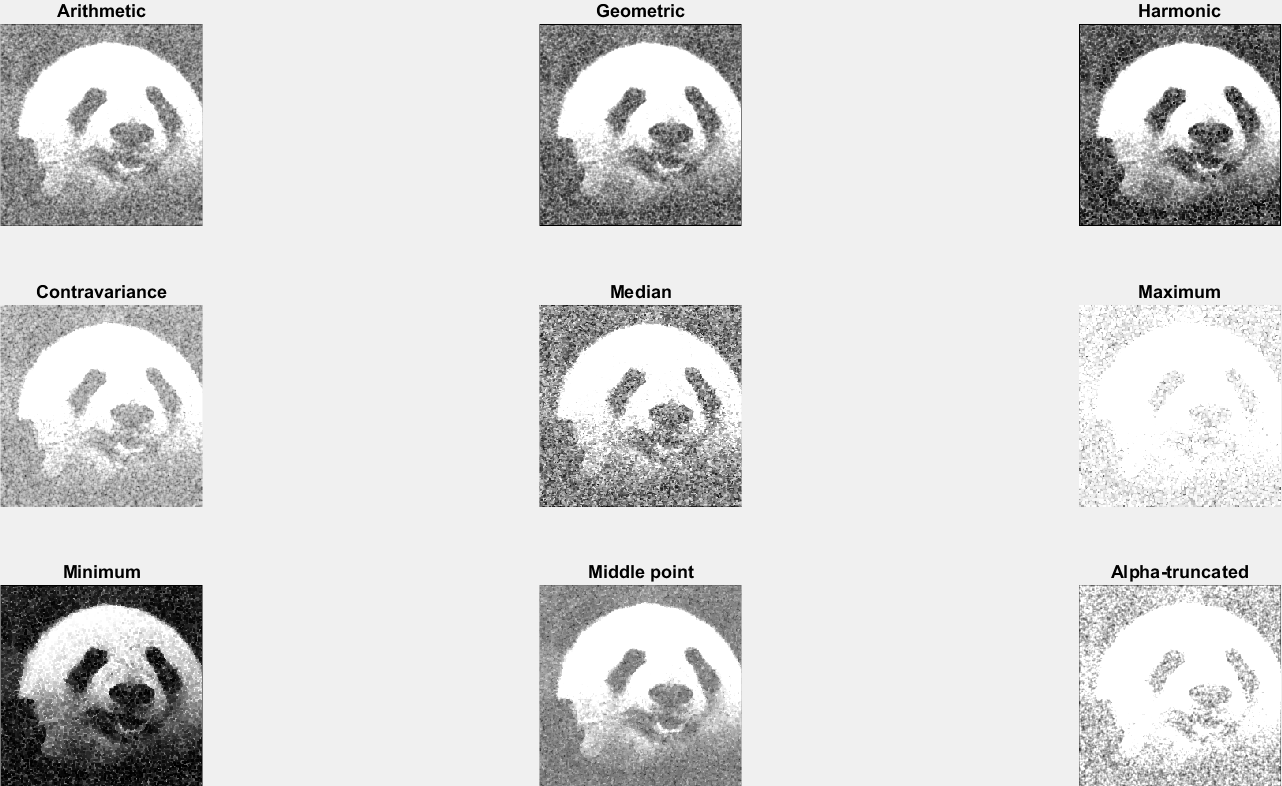
end

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Равномерный | Гауссов | Логарифмически нормальный | Реллея | Экспоненциальный | Эрланга |
| Арифметическое среднее | 0.9555 | **0.9863** | **0.9572** | **0.8994** | 0.707 | 0.656 |
| Геометрическое среднее | 0.9442 | 0.9353 | 0.9073 | 0.8862 | 0.8372 | 0.6346 |
| Гармоническое среднее | 0.9446 | 0.0159 | 0.9124 | 0.8942 | 0.8941 | **0.6653** |
| Контргармоническое среднее | 0.9528 | 0.0073 | 0.9527 | 0.8408 | 0.2897 | 0.5094 |
| Медиана | 0.8943 | 0.9777 | 0.9433 | 0.8363 | 0.6561 | 0.5580 |
| Максимумы | 0.9565 | 0.9593 | 0.824 | 0.6865 | 0.2852 | 0.3214 |
| Минимумы | 0.953 | 0.9573 | 0.9179 | 0.894 | **0.9374** | 0.6455 |
| Срединная точка | **0.9739** | 0.9767 | 0.921 | 0.853 | 0.4834 | 0.5188 |
| α-усеченное среднее | 0.9555 | **0.9863** | **0.9572** | **0.8994** | 0.707 | 0.656 |

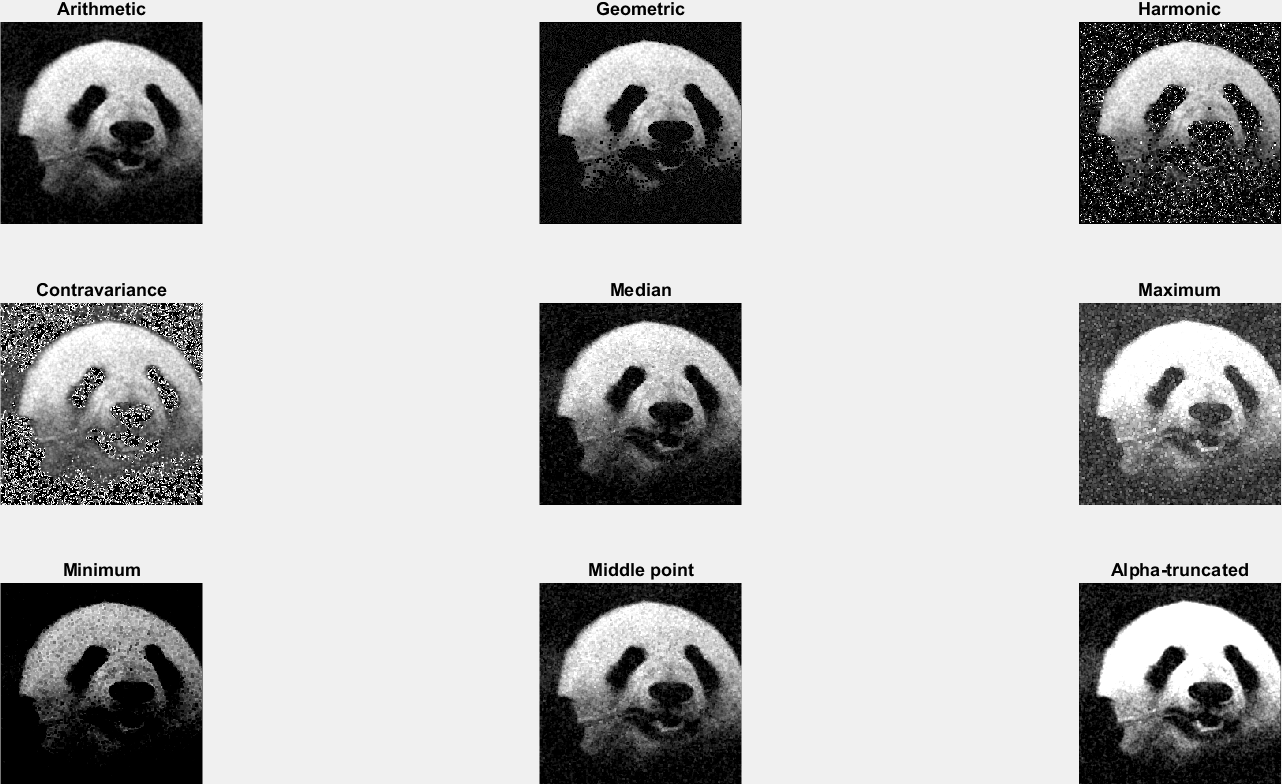
*Таблица 1 - Результаты фильтрации*

Результаты

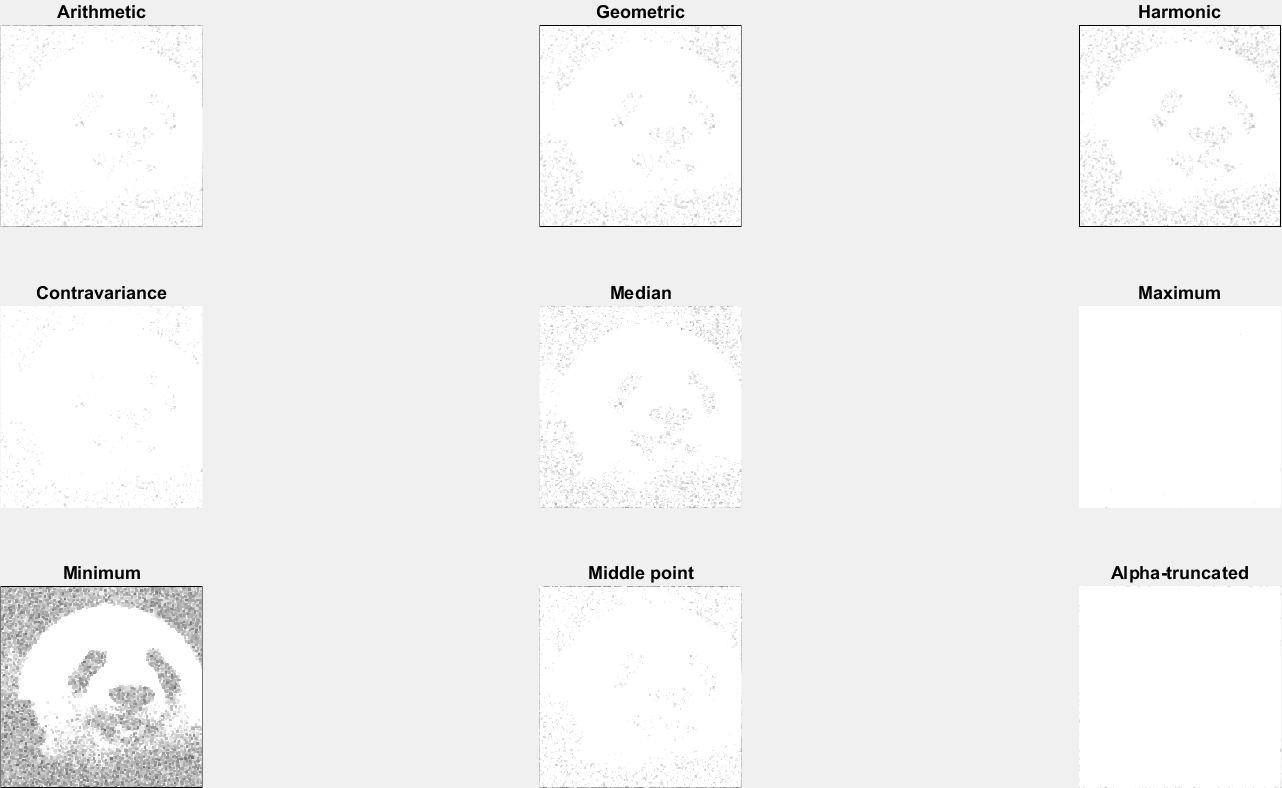
На Рисунках 4 - 9 представлены отфильтрованные изображения после воздействия на них каждым типом шума. Исходные изображения можно видеть на Рисунке 3.



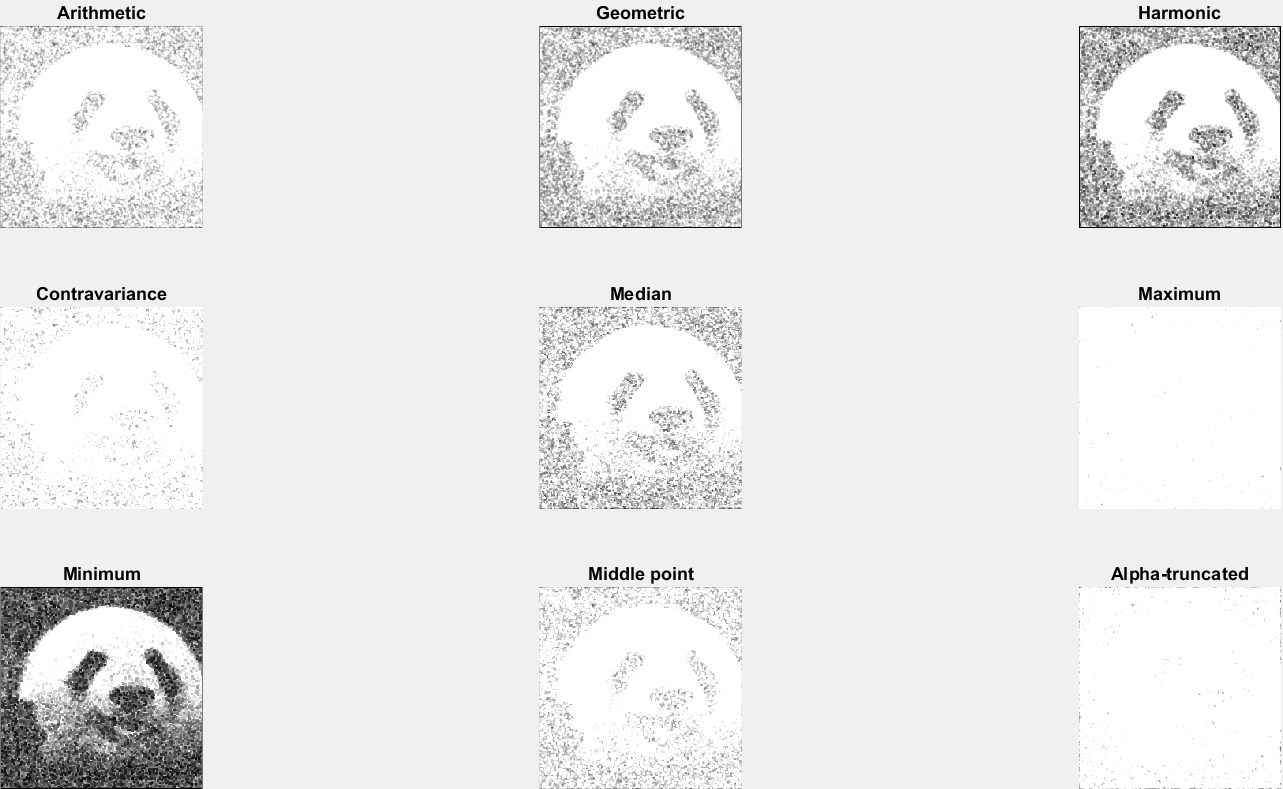
*Рисунок 4 - Фильтрация равномерного шума*



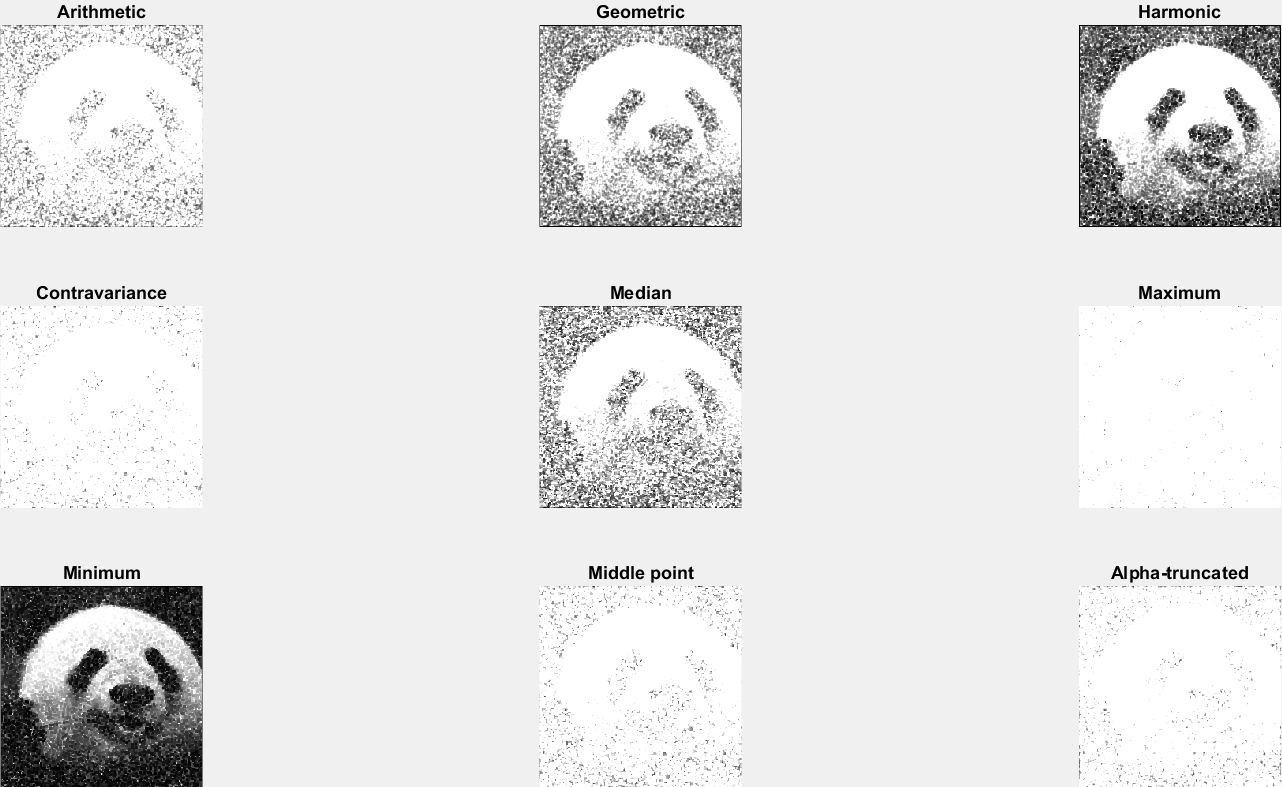
*Рисунок 5 - Фильтрация Гауссова шума*



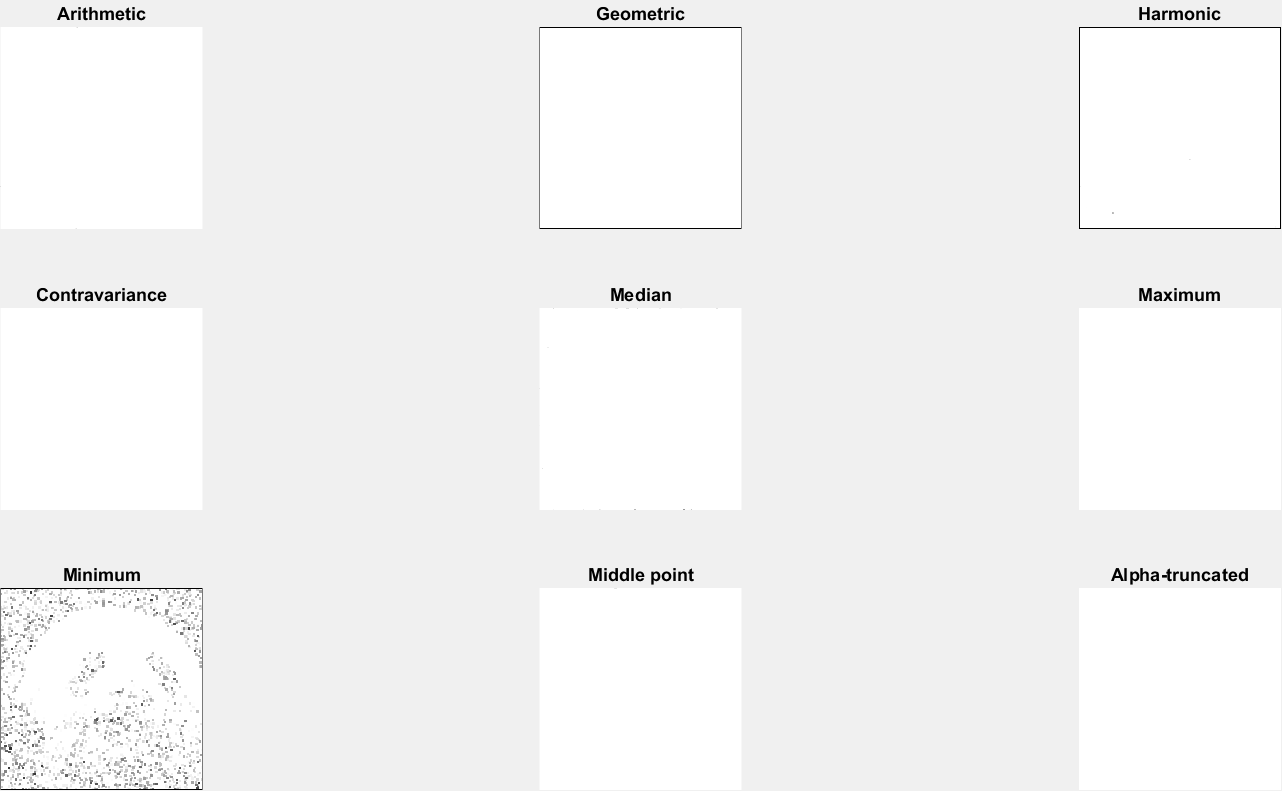
*Рисунок 6 - Фильтрация логарифмически нормального шума*



*Рисунок 7 - Фильтрация шума Реллея*



*Рисунок 8 - Фильтрация экспоненциального шума*



*Рисунок 9 - Фильтрация шума Эрланга*

**Выводы**

* Результаты фильтрации, приведенные в Таблице 1, показали, что наилучшим фильтром против равномерного шума стала *срединная точка*, которая удалила наибольшее количество шумов с фотографии*.* Однако, если посмотреть на Рисунок 4, визуально может показаться, что фильтры *минимум* и *гармоническое среднее* довели картинку практически до оригинала, приблизив цвета конечного изображения к цветам оригинального.
* Результаты фильтрации, приведенные в Таблице 1, показали, что лучшего всего убрали Гауссовы шумы фильтры *арифметическое среднее* и *α-усеченное среднее*. Однако, если обратиться к Рисунку 5, можно заметить, что практически все фильтры отлично справились с удалением Гауссовых шумов, что может быть связано со слабым эффектом, который они придают фотографии при наложении.
* Если положиться на результаты фильтрации, приведенные в Таблице 1, то выходит, что для логарифмически нормального шума лучшими фильтрами также могут послужить *арифметическое среднее* и *α-усеченное среднее*. Однако из Рисунка 6 можно видеть, что фильтры *максимум* и *α-усеченное среднее* имеют одинаковый эффект, чего не скажешь про их совершенно разные значения коэффициента корреляции из Таблицы 1. Визуально лучшим фильтром кажется опять-таки *минимум*.
* Из результатов фильтрации по Таблице 1 для борьбы с шумами Реллея лучше всего использовать многофункциональные фильтры *арифметическое среднее* и *α-усеченное среднее*. Однако визуально по Рисунку 7 фильтр *минимум* снова привел изображение к практически оригинальному виду.
* С экспоненциальными шумами визуально по Рисунку 8 справилось уже большее количество фильтров, чем с шумами Реллея. Уже можно сказать, что фильтры *минимум* и *гармоническое среднее* привели картинку к практически оригинальному виду, однако по Таблице 1 лучшим фильтром для борьбы с шумами послужил только *минимум*.
* Шумы Эрланга оказались наносящими фотографии наибольший ущерб. Из результатов Таблицы 1 лучшего всего удалил с фотографии шумы фильтр *гармоническое среднее*. Однако, если обратиться к Рисунку 9, можно заметить только, что после фильтрации всеми фильтрами, кроме *минимума*, вид фотографии приблизился к белому квадрату, не несущему никакой графической информации.

Таким образом, наибольший ущерб фотографии нанесли шумы Эрланга и логарифмически нормальные.

Многофункциональными фильтрами, удалившими наибольшее количество шумов, стали *арифметическое среднее* и *α-усеченное среднее*.

Фильтр, который практически во всех случаях вернул фотографии очертания и цвета, максимально близкие к оригинальным, стал *минимум*.