Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Цифровая обработка сигналов

**Лабораторная работа №6**

**Студент:**

Черезов Игорь Юрьевич

**Группа:**

P3400

**Преподаватель:**

Тропченко Андрей Александрович

Санкт-Петербург

2020

# Цель работы

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

# Задание

**Задание № 1:**

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

Таблица 3.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип шума | Генератор шума (z) | Параметры для данной работы | |
| a | b |
| Равномерный | функция rand | - | - |
| Гауссов | функция a + b\*randn | 0 | 0.15 |
| Логарифмически нормальный |  | 1 | 0.25 |
| Реллея |  | 0 | 1 |
| Экспоненциальный |  | 1 | - |
| Эрланга | z = E1 + E2 + … + Eb,  где En = | 2 | 5 |

function g = even\_dist(rows, columns)

g = rand(rows, columns);

endfunction

function g = gauss\_dist(rows, columns)

g = 0.15\*randn(rows, columns);

endfunction

function z = log\_normal\_dist(rows, columns)

z = exp(0.25\*randn(rows, columns));

endfunction

function z = rayleigh\_dist(rows, columns)

z = sqrt(-log(1 - rand(rows, columns)));

endfunction

function g = even\_dist(rows, columns)

g = rand(rows, columns);

endfunction

function z = gamma\_dist(rows, columns)

order = 5; % b

a = 2;

z = -1/a\*log(1 - rand(rows, columns));

for i = 2:order

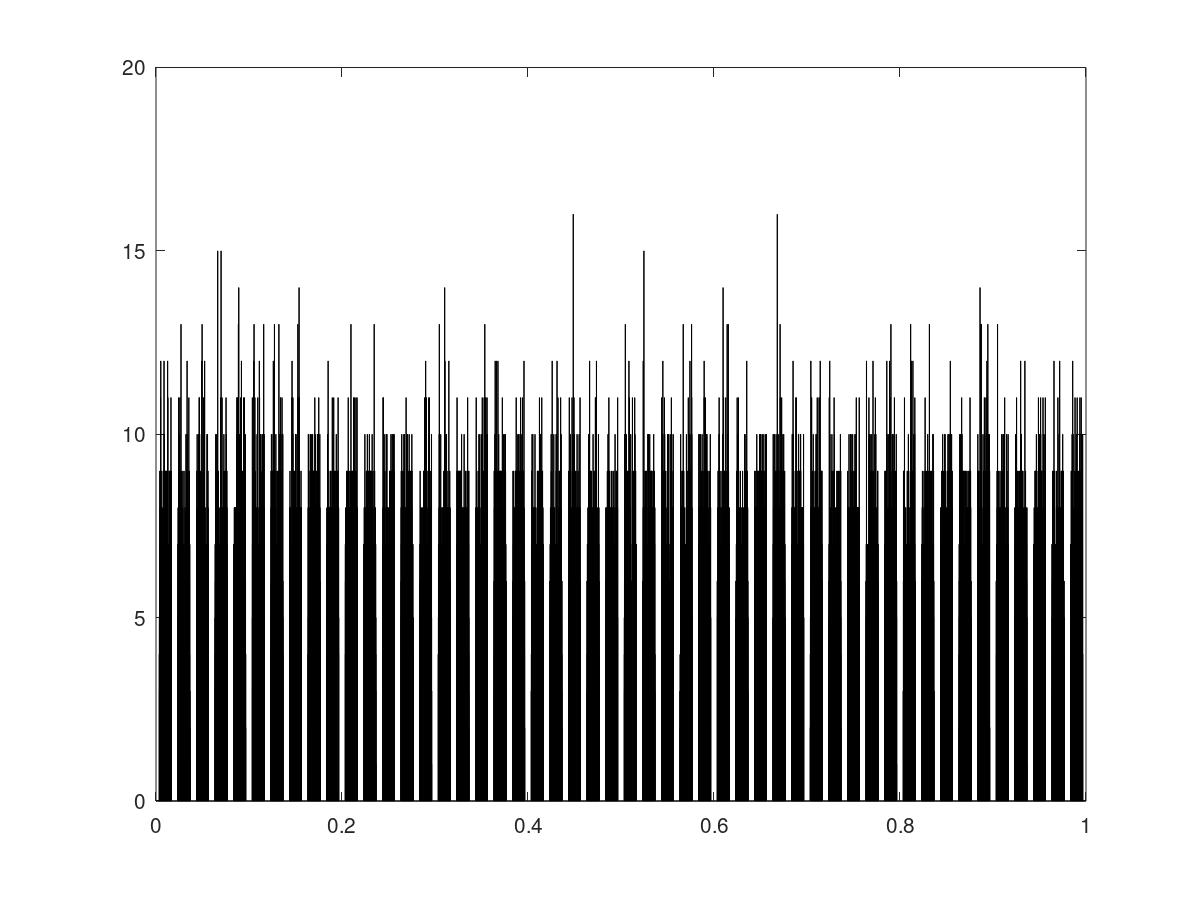
z += -1/a\*log(1 - rand(rows, columns));

endfor

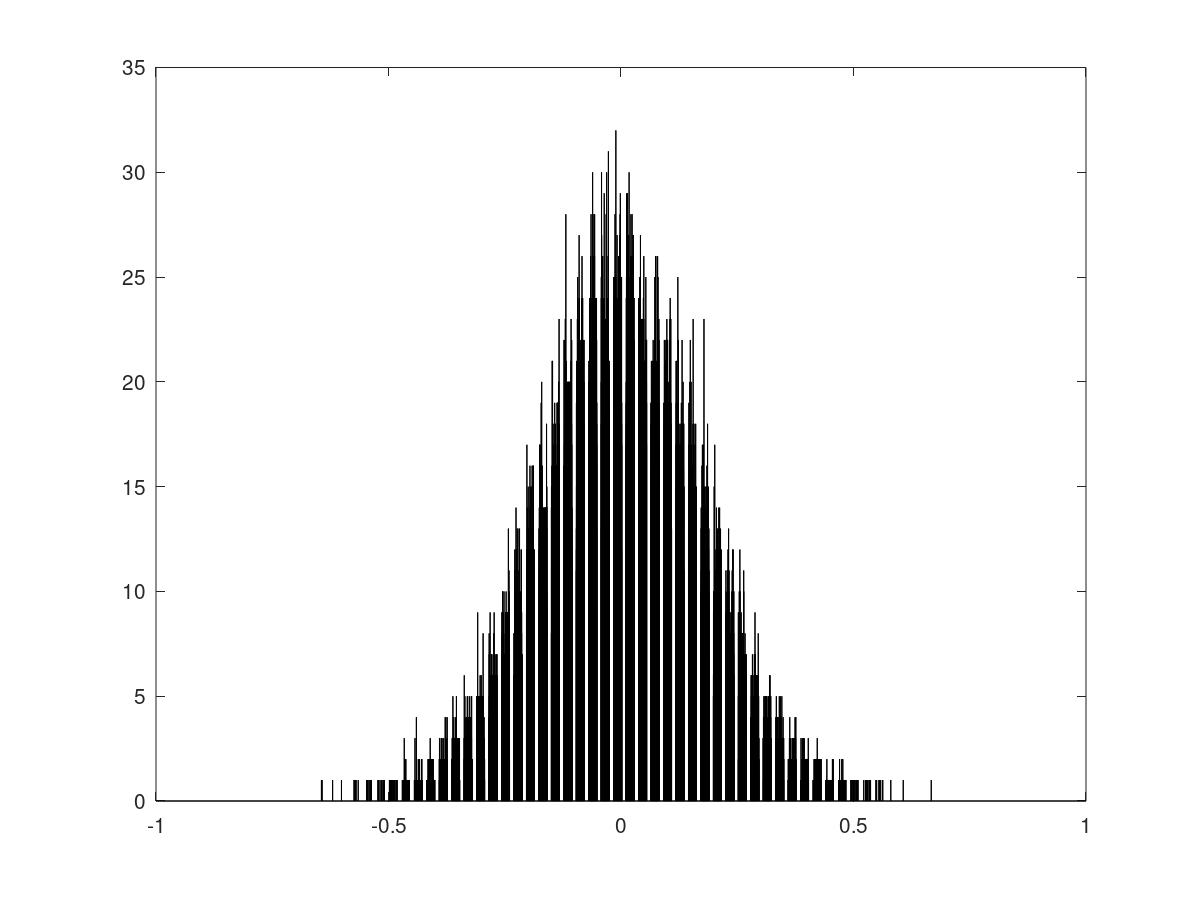
endfunction

**Задание № 2:**

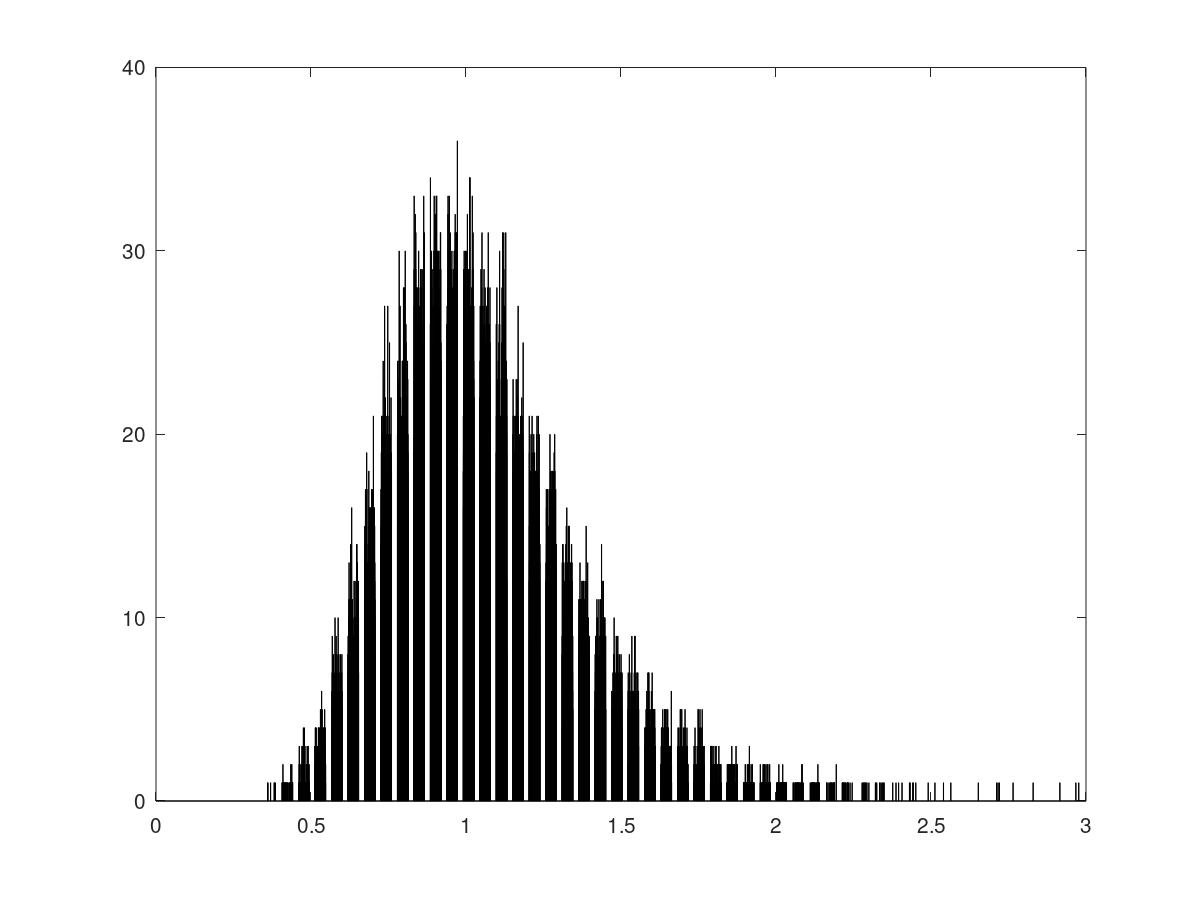
Осуществите генерацию каждого типа шума размером 256x256 при помощи созданных функций. Для всех шумов, кроме шума Соль и перец создайте гистограмму и поместите их в отчет. Количество корзин гистограммы должно иметь значение, равное 50.



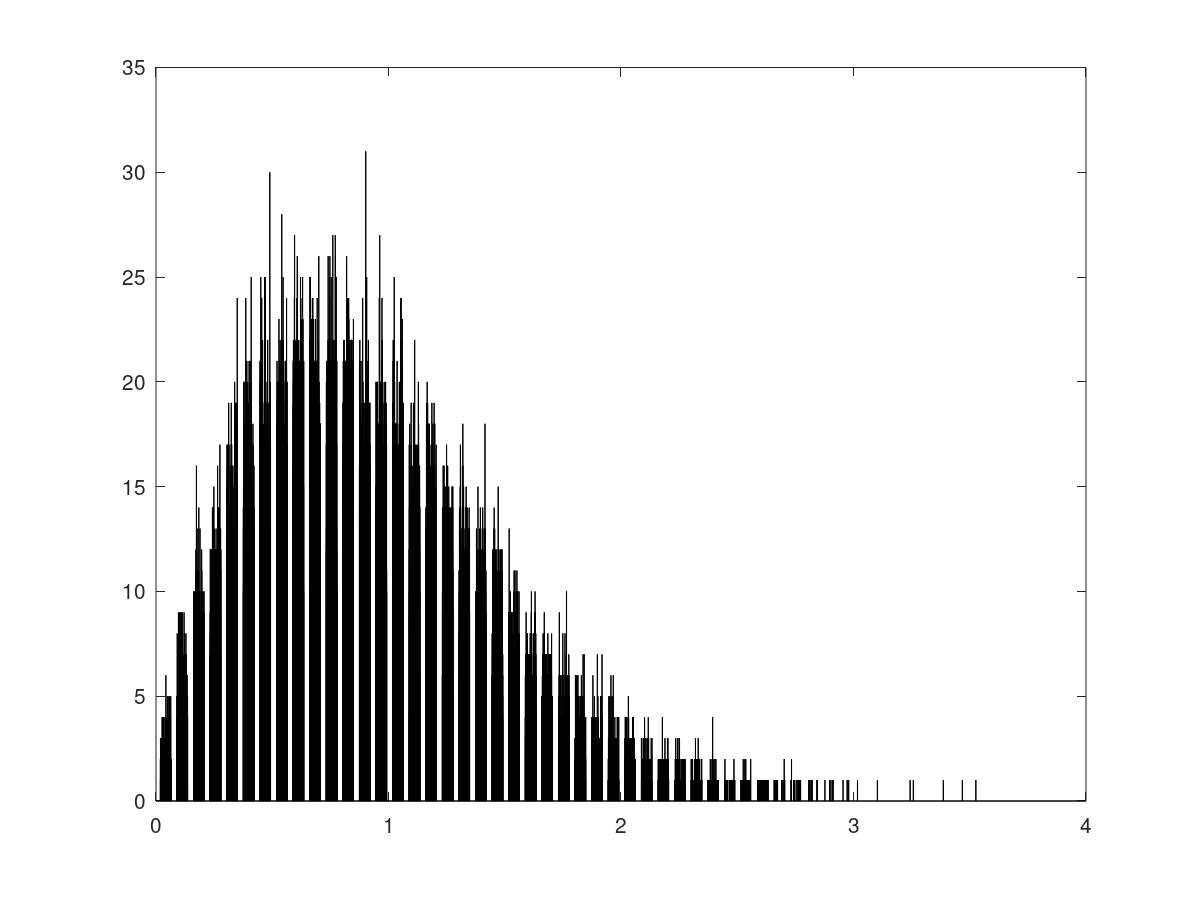
Изображение 1 Распределения раномерное



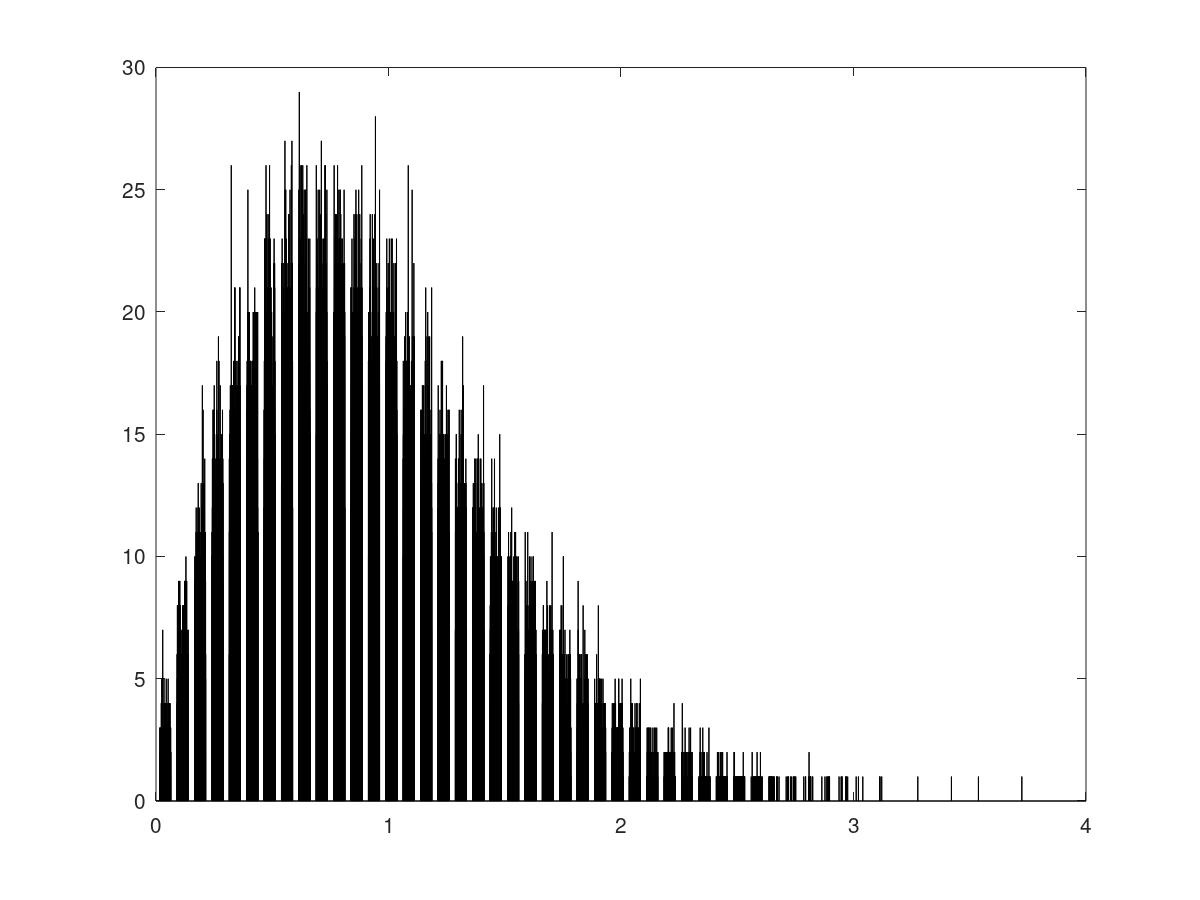
Изображение 2 Распределения Гаусса



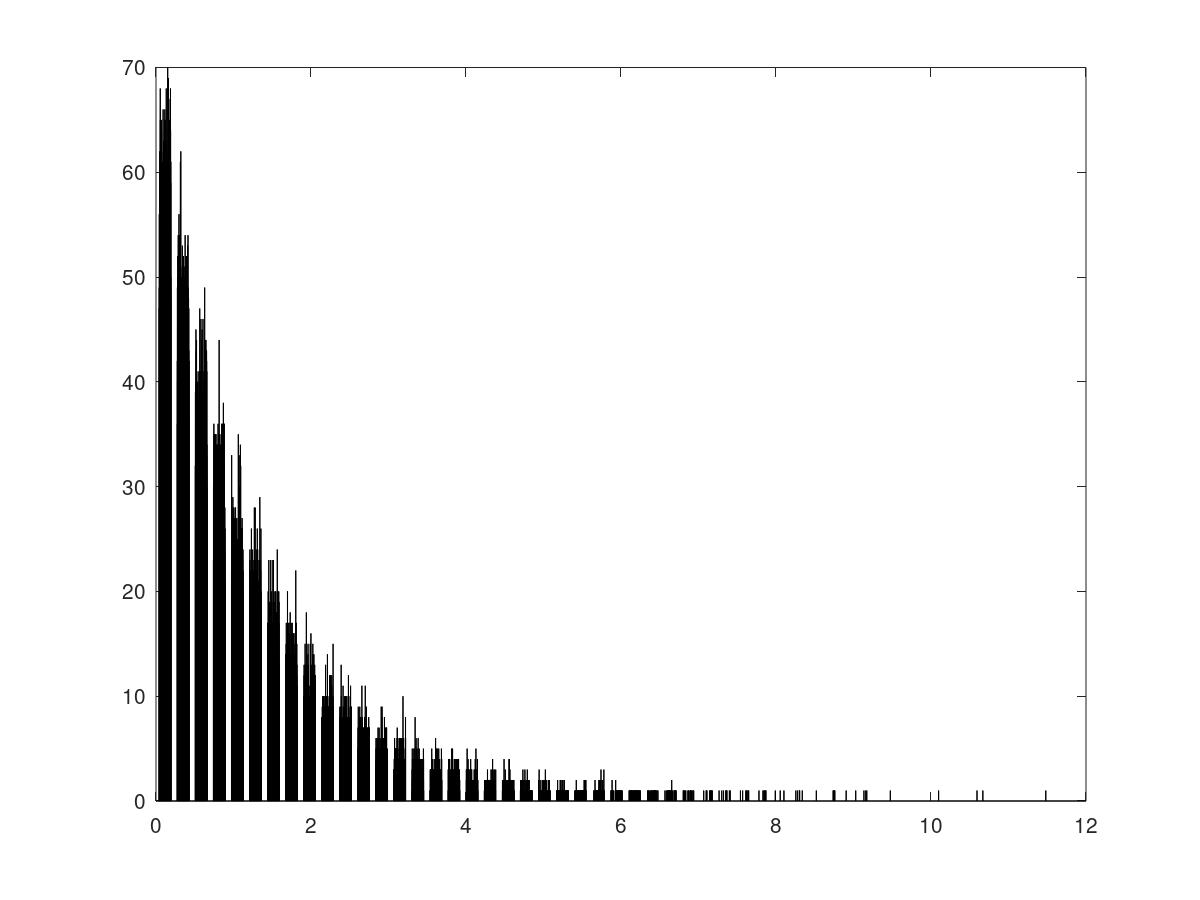
Изображение 3 Логарифмическое нормальное распределение



Изображение 4 Распределение Рэлея



Изображение 5 Распределение Эрланга



Изображение 6 Экспоненциальное распределение

**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.



Изображение 7 Равномерное зашумление



Изображение 8 Зашумление Гаусса



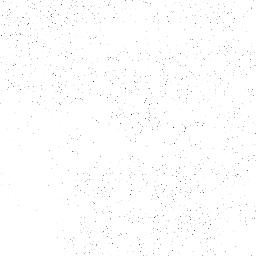
Изображение 9 Логарифмическое нормальное зашумление



Изображение 10 Зашумление Релея



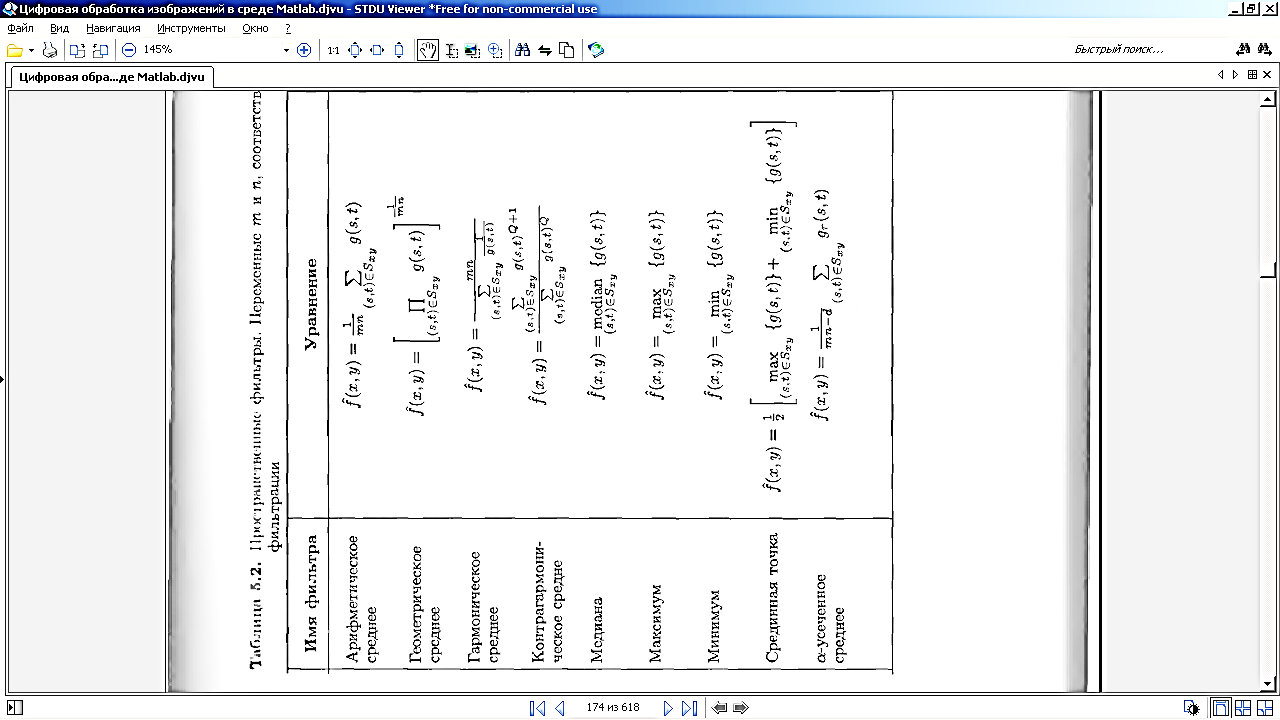
Изображение 11 Экспоненциальное зашумление



Изображение 12 Зашумление Эрланга

**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.



function g = filter\_ar\_avg(window)

m = size(window)(1,1);

n = size(window)(1,2);

g = 1/(m\*n)\*sum(sum(window));

endfunction

function g = filter\_geo\_avg(window)

m = size(window)(1,1);

n = size(window)(1,2);

g = prod(prod(window))^(1/(m\*n));

endfunction

function g = filter\_harm\_avg(window)

m = size(window)(1,1);

n = size(window)(1,2);

g = (m\*n)/sum(sum(1./window));

endfunction

function g = filter\_counterharm\_avg(window)

m = size(window)(1,1);

n = size(window)(1,2);

Q = m\*n;

g = sum(sum(window.^(Q+1)))/(sum(sum(window.^Q)));

endfunction

function g = filter\_median(window)

g = median(window(:));

endfunction

function g = filter\_max(window)

g = max(max(window));

endfunction

function g = filter\_min(window)

g = max(window(:));

endfunction

function g = filter\_mid\_point(window)

g = 0.5\*(max(window(:)) + min(window(:)));

endfunction

function g = filter\_alpha\_avg(window)

alpha = 1;

m = size(window)(1,1);

n = size(window)(1,2);

g = 1/(m\*n-alpha)\*sum(window(:));

endfunction

**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 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, отобразите 10 восстановленных изображений с указанием: какой тип фильтра изначально был использован.

Для автоматизации выполнения данной задачи была написана функция, которая скользит по изображению окном 3x3, применяя фильтр, переданный ей. Основная функция итерируется по массиву зашумлённых изображений и массиву указателей на фильтры, номера которых соответствуют их порядку в заполняемой таблице.

function g = iter(mtx, fun)

g = mtx;

rows = size(mtx)(1, 1);

columns = size(mtx)(1, 2);

window = zeros(3,3);

for i=2:rows-1

for j = 2:columns-1

for m=1:3

for n = 1:3

window(m,n) = mtx(i-2+m, j-2+n);

endfor

endfor

g(i,j) = fun(window);

endfor

endfor

endfunction

function cor = cor\_table()

img = imread("batman256.png");

img = rgb2gray(img);

img = im2double(img);

cor = zeros(9, 6);

noised\_imgs = ["n\_even\_bat.jpg",

"n\_gauss\_bat.jpg",

"n\_log\_bat.jpg",

"n\_ray\_bat.jpg",

"n\_exp\_bat.jpg",

"n\_gamma\_bat.jpg"];

noised\_imgs = cellstr(noised\_imgs);

filters = ["filter\_ar\_avg",

"filter\_geo\_avg",

"filter\_harm\_avg",

"filter\_counterharm\_avg",

"filter\_median",

"filter\_max",

"filter\_min",

"filter\_mid\_point",

"filter\_alpha\_avg"];

filters = cellstr(filters);

for i=1:size(noised\_imgs)(1,1)

n\_img = imread(char(noised\_imgs(i,1)));

n\_img = im2double(n\_img);

for j=1:size(filters)(1,1)

dn\_img = iter(n\_img, str2func(char(filters(j,1))));

cor(j, i) = corr2(img, dn\_img);

disp(ctime(time));

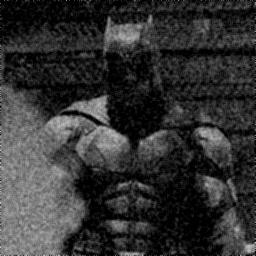
disp(cor);

endfor

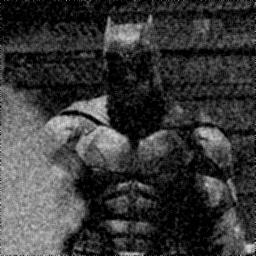
endfor

endfunction

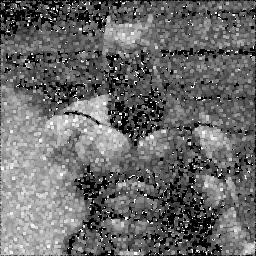
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Равномерный | Гауссов | Логарифмически нормальный | Релея | Экспоненциальный | Эрланга |
| Арифметическое среднее | 0.798234 | 0.94286 | 0.710199 | 0.674684 | 0.639038 | 0.352319 |
| Геометрическое среднее | 0.786904 | 0.884261 | 0.705552 | 0.679134 | 0.670155 | 0.32958 |
| Гармоническое среднее | 0.810027 | 0.871996 | 0.698618 | 0.675701 | 0.727976 | 0.295864 |
| Контргармоническое среднее | 0.375916 | 0.833507 | 0.583868 | 0.150309 | 0.05735 | 0.374149 |
| Медиана | 0.709495 | 0.928827 | 0.478941 | 0.466054 | 0.430552 | 0.202199 |
| Максимумы | 0.332261 | 0.850242 | 0.051409 | 0.056083 | 0.059056 | 0.019873 |
| Минимумы | 0.332261 | 0.850242 | 0.051409 | 0.056083 | 0.059056 | 0.019873 |
| Срединная точка | 0.807002 | 0.900038 | 0.730066 | 0.700077 | 0.747273 | 0.307798 |
| α-усеченное среднее | 0.800176 | **0.943584** | 0.680966 | 0.675716 | 0.641459 | 0.278511 |



Изображение 13 Зашумление Гаусса, арифметический усредняющий фильтр



Изображение 14 Зашумление Гаусса, альфа-усечённый усредняющий фильтр



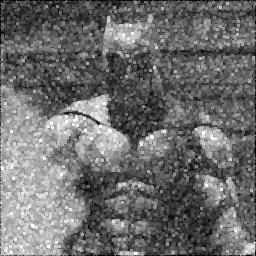
Изображение 15 Зашумление Гаусса, контргармонический усредняющий фильтр



Изображение 16 Зашумление Гаусса, геометрический усредняющий фильтр



Изображение 17 Зашумление Гаусса, гармонический усредняющий фильтр



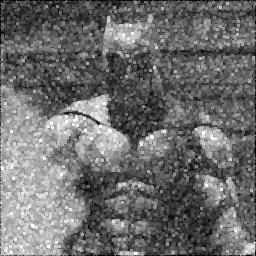
Изображение 18 Зашумление Гаусса, максимальный фильтр



Изображение 19 Зашумление Гаусса, Медианный фильтр



Изображение 20 Зашумление Гаусса, фильтр срединной точки



Изображение 21 Зашумление Гаусса, минимальный фильтр



Изображение 22 Равномерное зашумление, фильтр срединной точки

# Выводы

В ходе выполнения данной лабораторной работы были изучены и реализованы различные виды зашумления. Также были изучены и реализованы различные методы фильтрации зашумлённых изображений. Наиболее высокий показатель корреляции Пирсона с исходным изображением был получен при зашумлении Гаусса и альфа-усечённым усредняющем фильтре.