Университет ИТМО

Цифровая обработка сигналов
Лабораторная работа №6

Выполнила: Калугина Марина

Группа: P3402

г. Санкт-Петербург

2020 г.

Задание

Задание № 1:

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

Таблица 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	$z = a * e^{b*N(0,1)}$	1	0.25
Реллея	$z = \sqrt{b * ln * [1 - U(0,1)]}$	0	1
Экспоненциальный	$z = -\frac{1}{a} * \ln[1 - U(0,1)]$	1	-
Эрланга	$z = E_1 + E_2 + \ldots + E_b$, где $E_n = -\frac{1}{a} * \ln\left[1 - U(0,1)\right]$	2	5

N(0, 1) – нормальная Гауссова величина со средним 0 и дисперсией 1 (аналогична функции *randn*). *U(0, 1)* – равномерная случайная величина из интервала (0, 1) (аналогична функции *rand*).

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.

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.

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.

В таблице 3.2 *n и m* – разрешение окна фильтрации. В качестве окна фильтрации используйте окно размером 3x3. *G* – исходное зашумленное изображение. *S* и *t* – координаты середины окна просмотра.

Имя фильтра	Уравнение
Арифметическое среднее	$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$
Геометрическое среднее	$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$
Гармоническое среднее	$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$
Контрагармонический среднее	$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$
Медиана	$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}} \{g(s,t)\}$
Максимум	$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$
Минимум	$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$
Срединная точка	$\hat{f}(x,y) = \frac{1}{2} \left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\} \right]$
α-усеченное среднее	$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$

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.

Результат

Равномерный шум

```
function noise = uniformNoise(M,N)
    noise = rand(M,N);
end
```

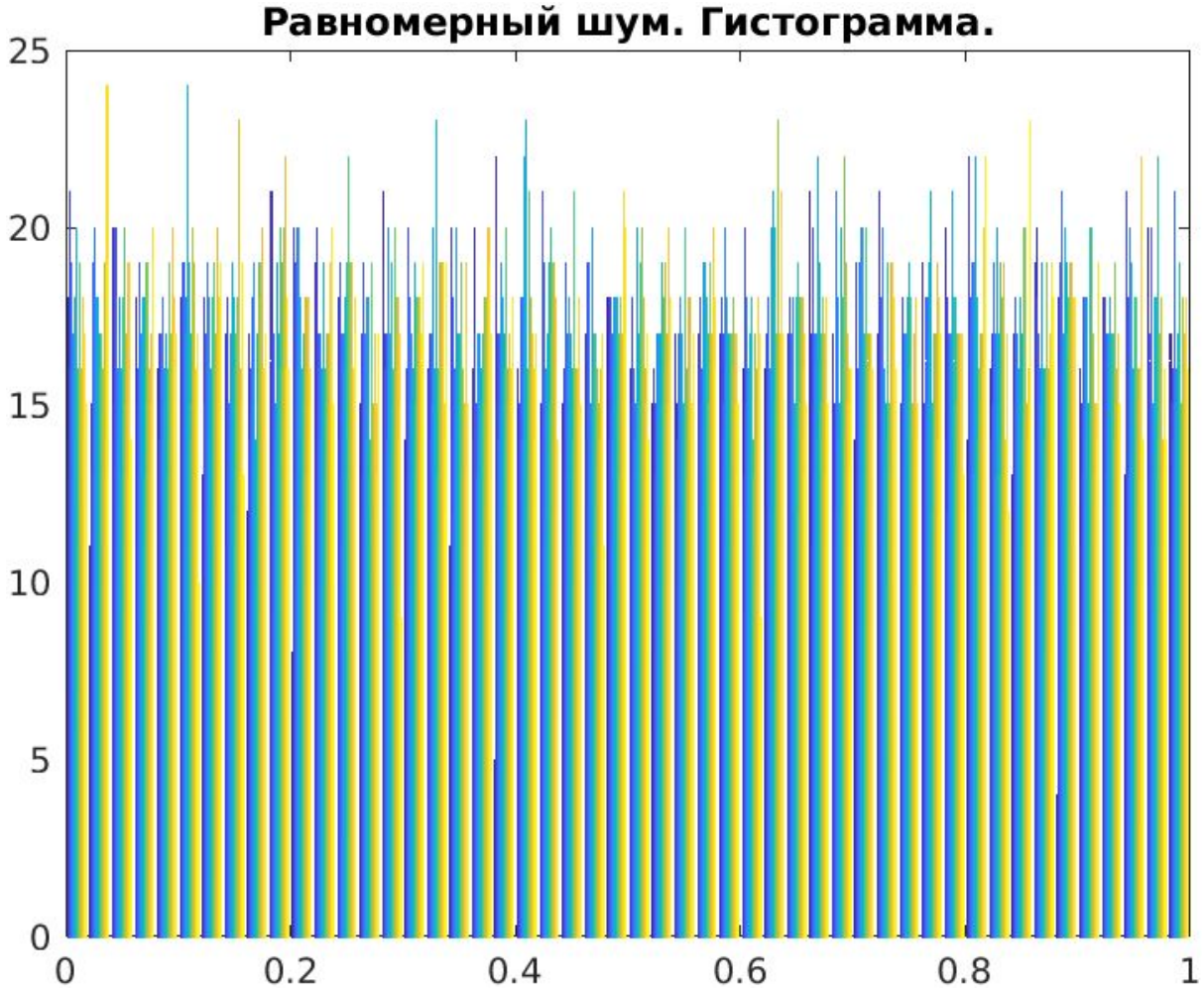


Рисунок 1 - гистограмма равномерного шума

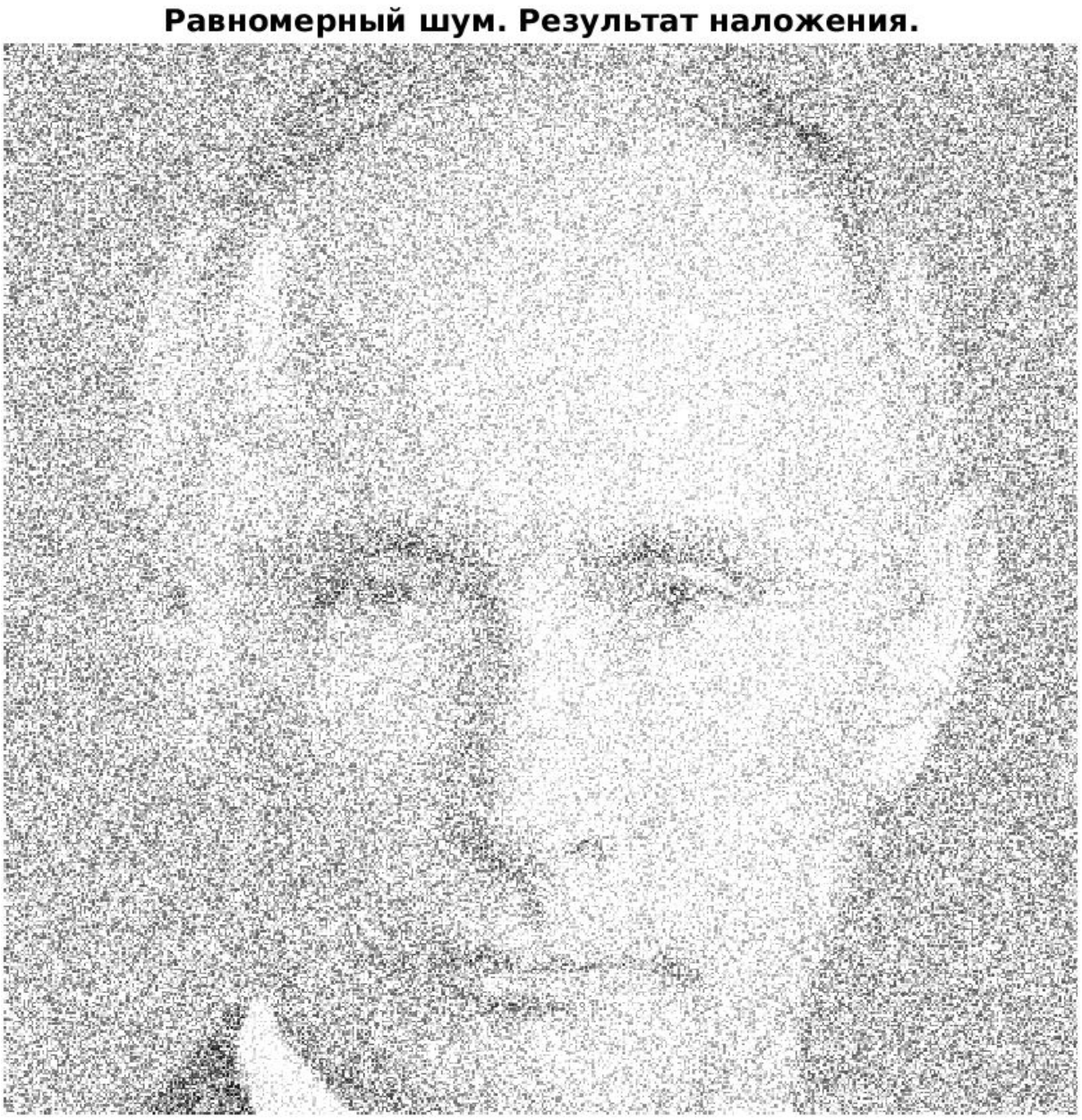


Рисунок 2 - результат наложения равномерного шума

Гауссов шум

```
function noise = gaussianNoise(M,N, a, b)
    noise = a + b*randn(M,N);
end
```

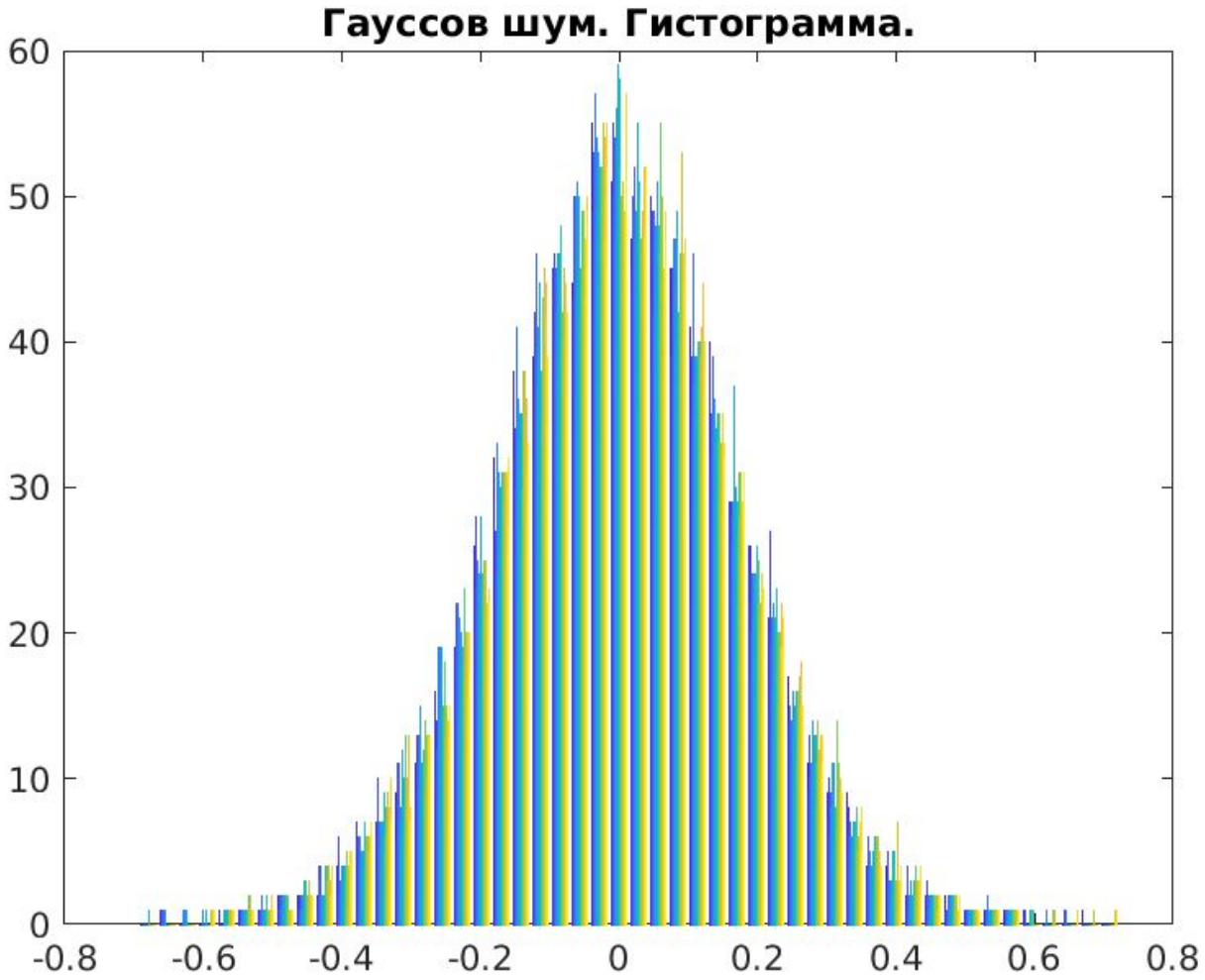


Рисунок 3 - гистограмма гауссового шума

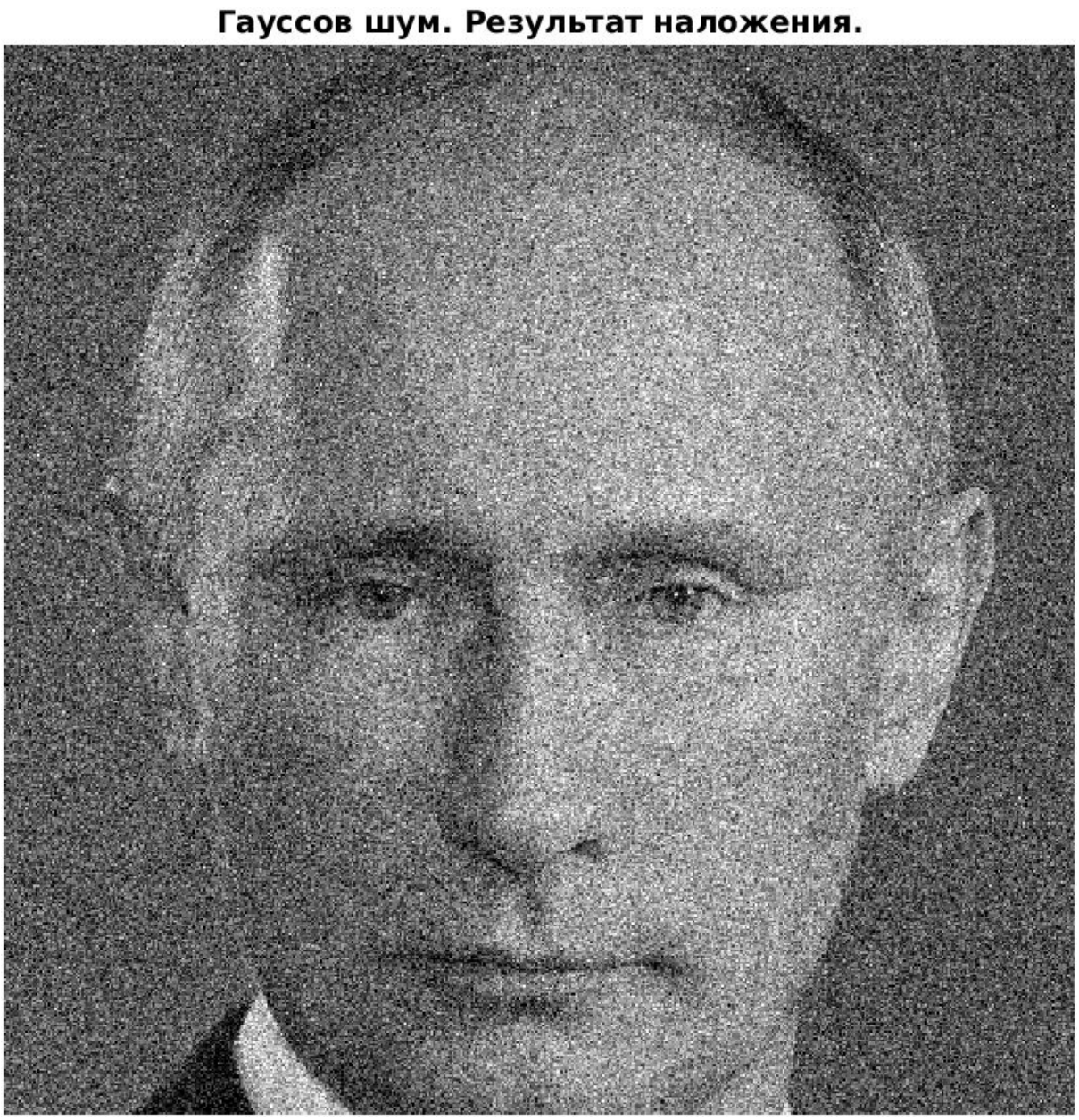


Рисунок 4 - результат наложения гауссова шума

Логарифмически нормальный шум

```
function noise = logarifmicNoise(M, N, a, b)
    noise = a*exp(b*randn(M, N));
end
```

Рисунок 5 - гистограмма логарифмически нормального шума

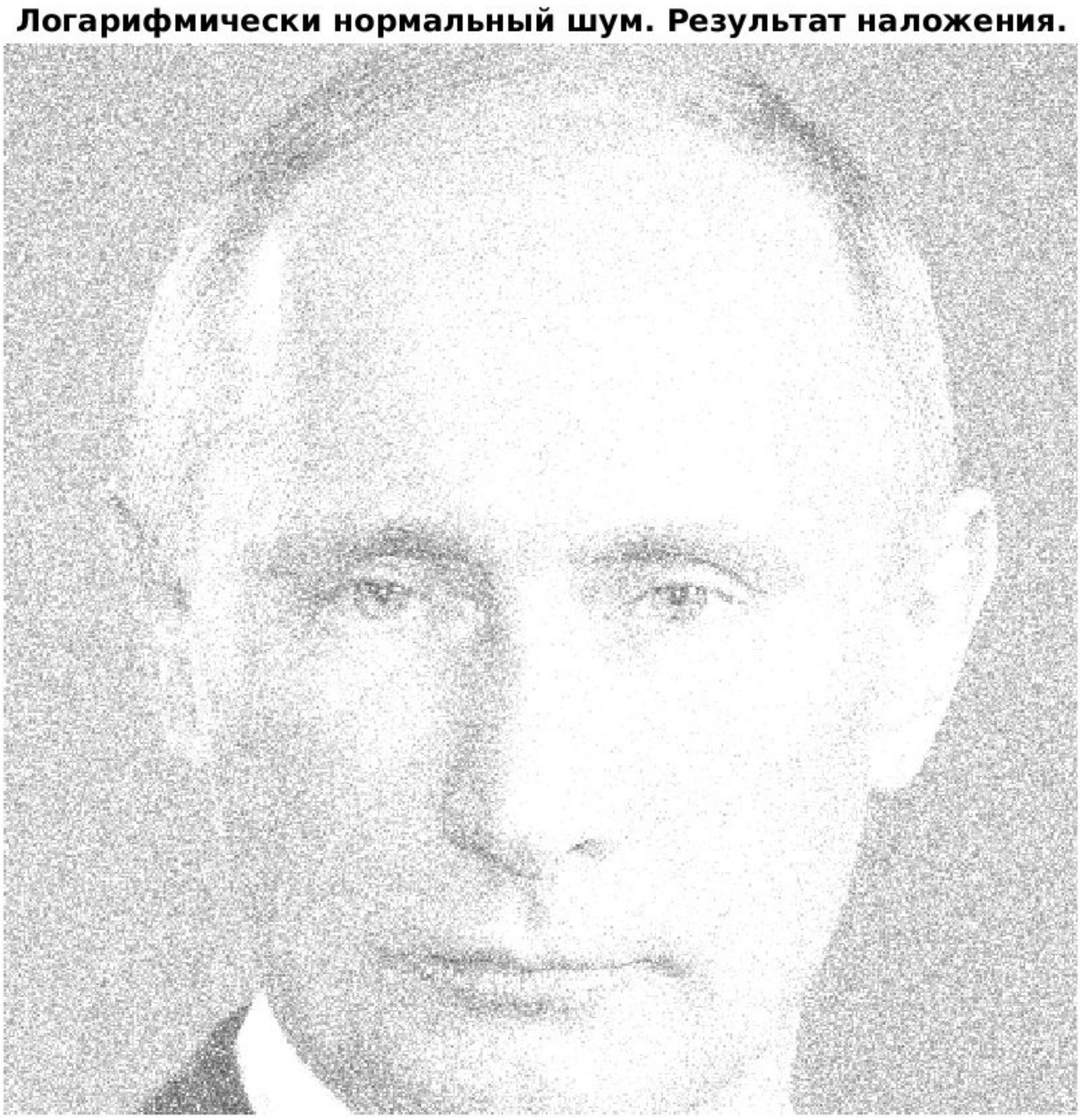


Рисунок 6 - результат наложения логарифмически нормального шума

При уменьшении параметра ‘a’ результат наложения шума получается менее ярким.

Шум Релея

```
function noise = relleyNoise(M, N, a, b)
    noise=a+(-b*log(1-rand(M,N))).^0.5;
end
```

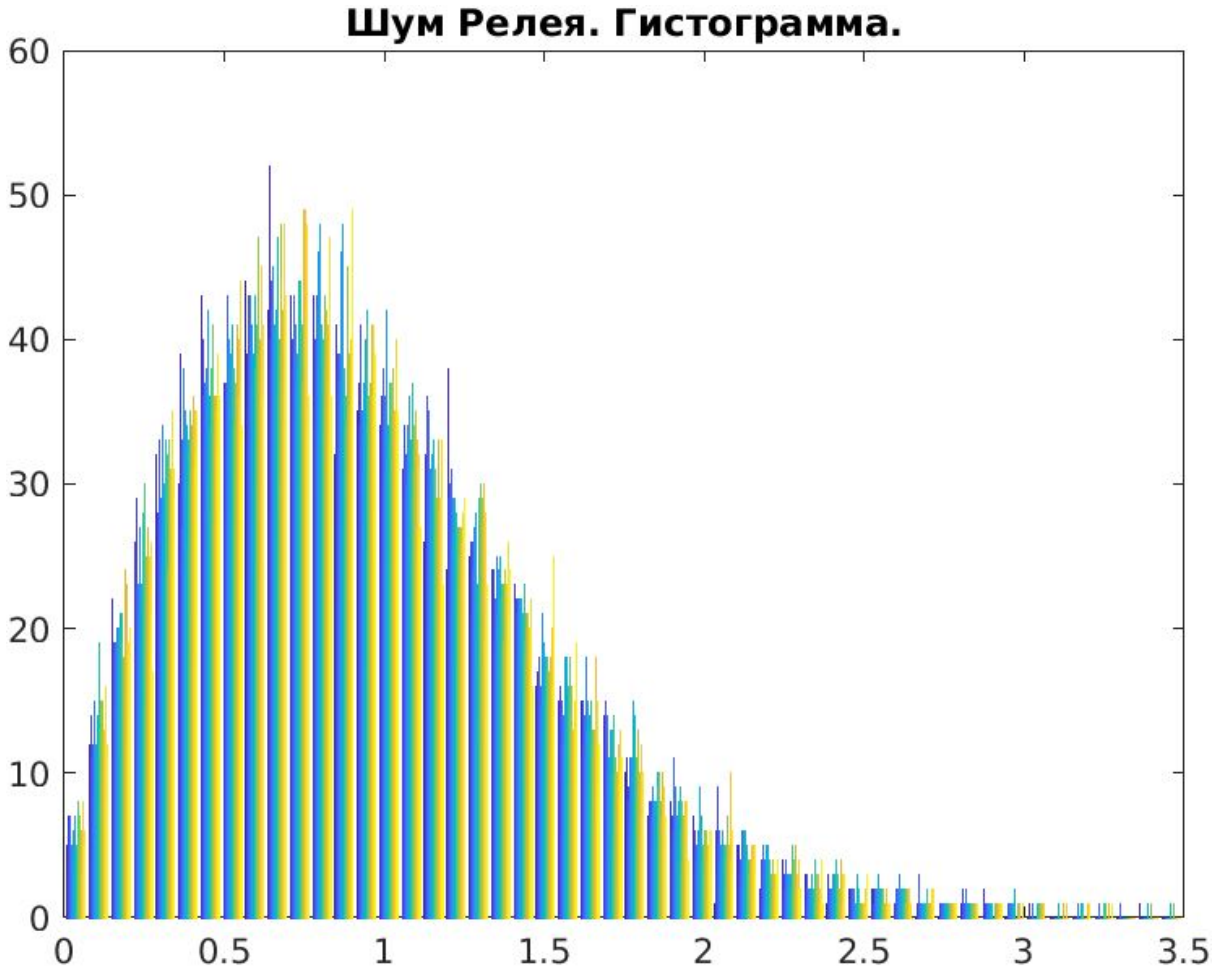


Рисунок 7 - гистограмма шума Релея

Шум Релея. Результат наложения.

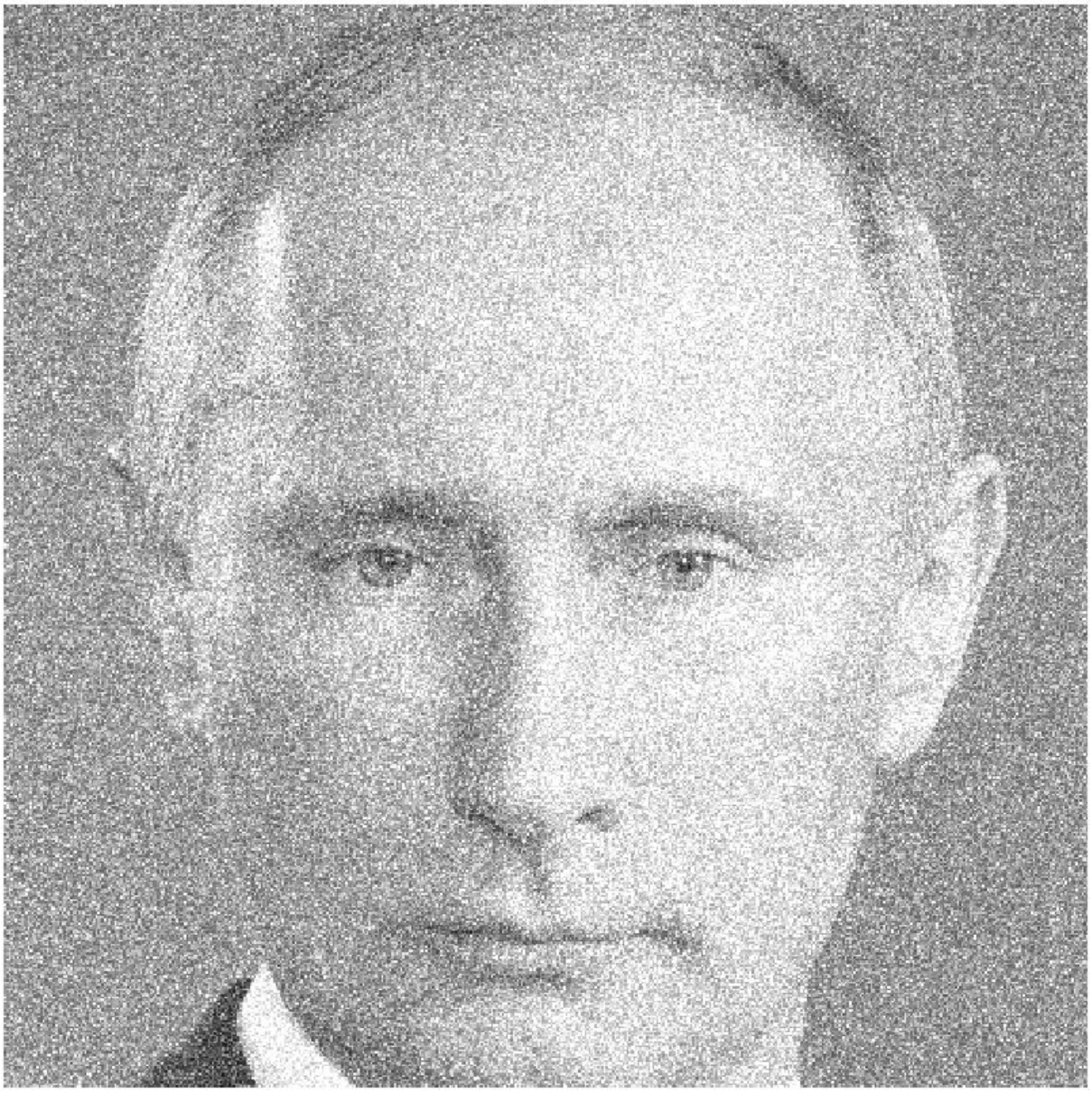


Рисунок 8 - результат наложения шума Релея

Экспоненциальный шум

```
function noise = exponentialNoise(M, N, a)
    noise=-1/a*log(1-rand(M,N));
end
```

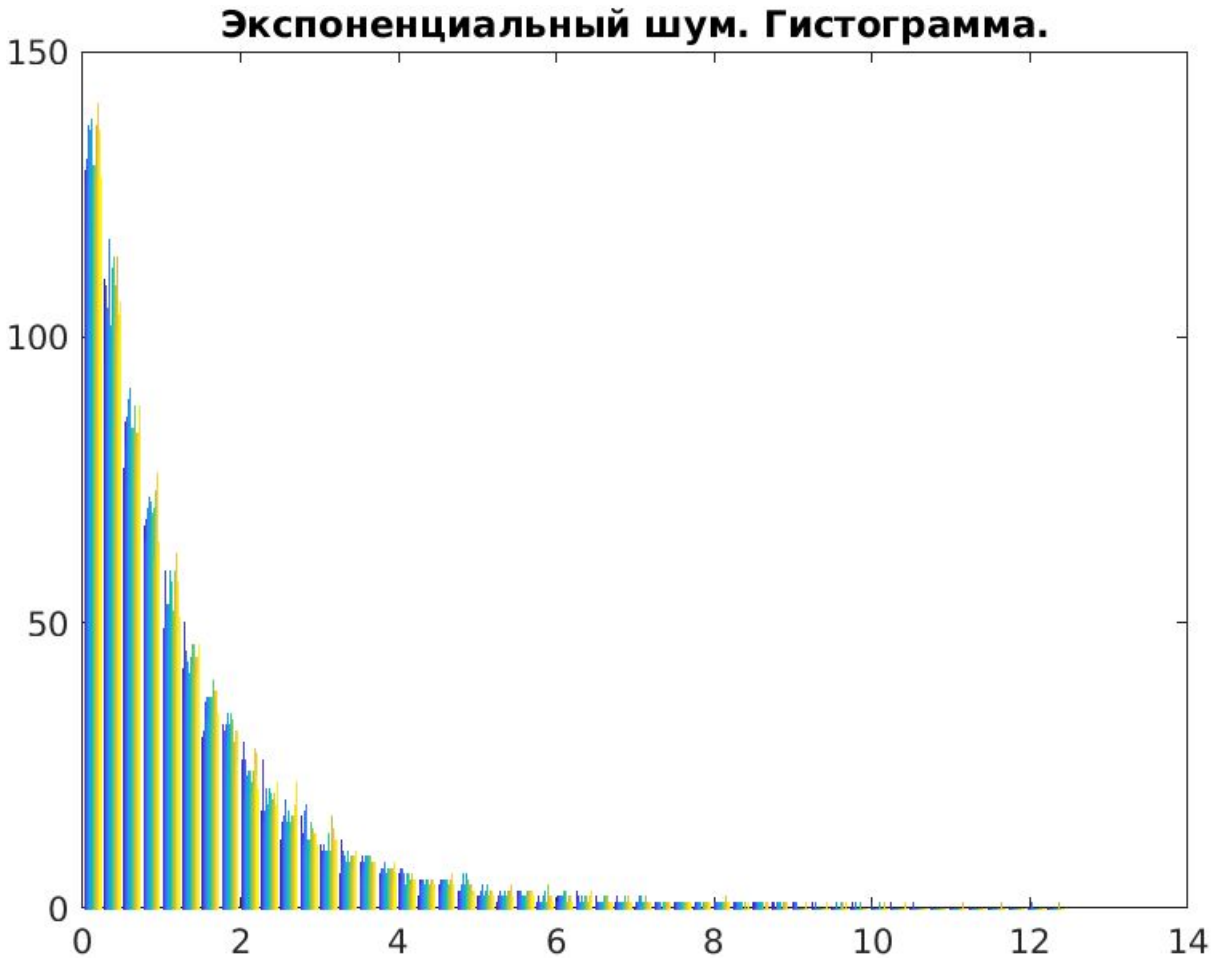


Рисунок 9 - гистограмма экспоненциального шума

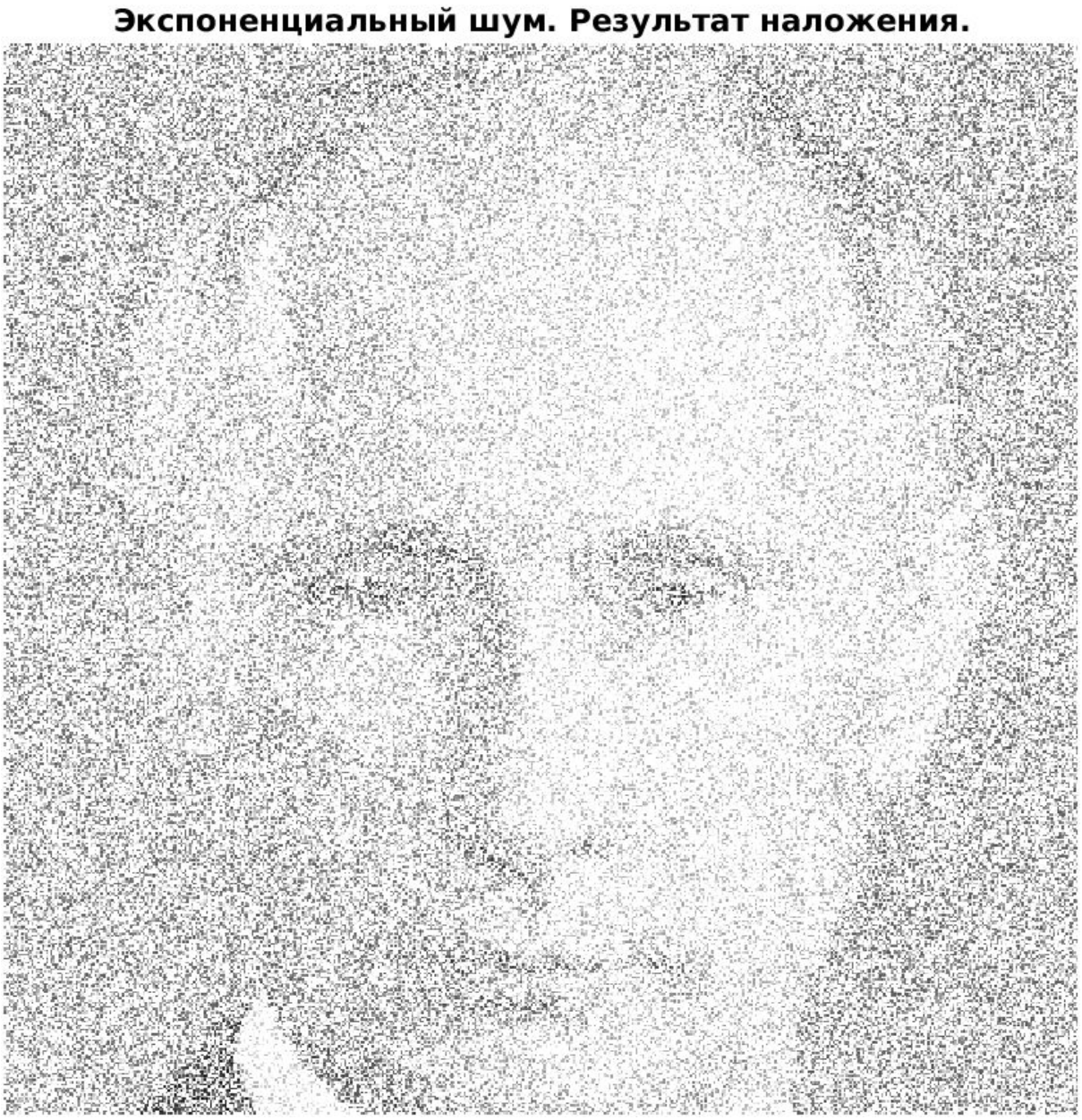


Рисунок 10 - результат наложения экспоненциального шума

Шум эрланга

```
function noise = erlangNoise(M, N, a, b)
    noise=zeros(M,N);
    for i=1:b
        noise = noise + (-1/a)*log(1-rand(M,N));
    end
end
```

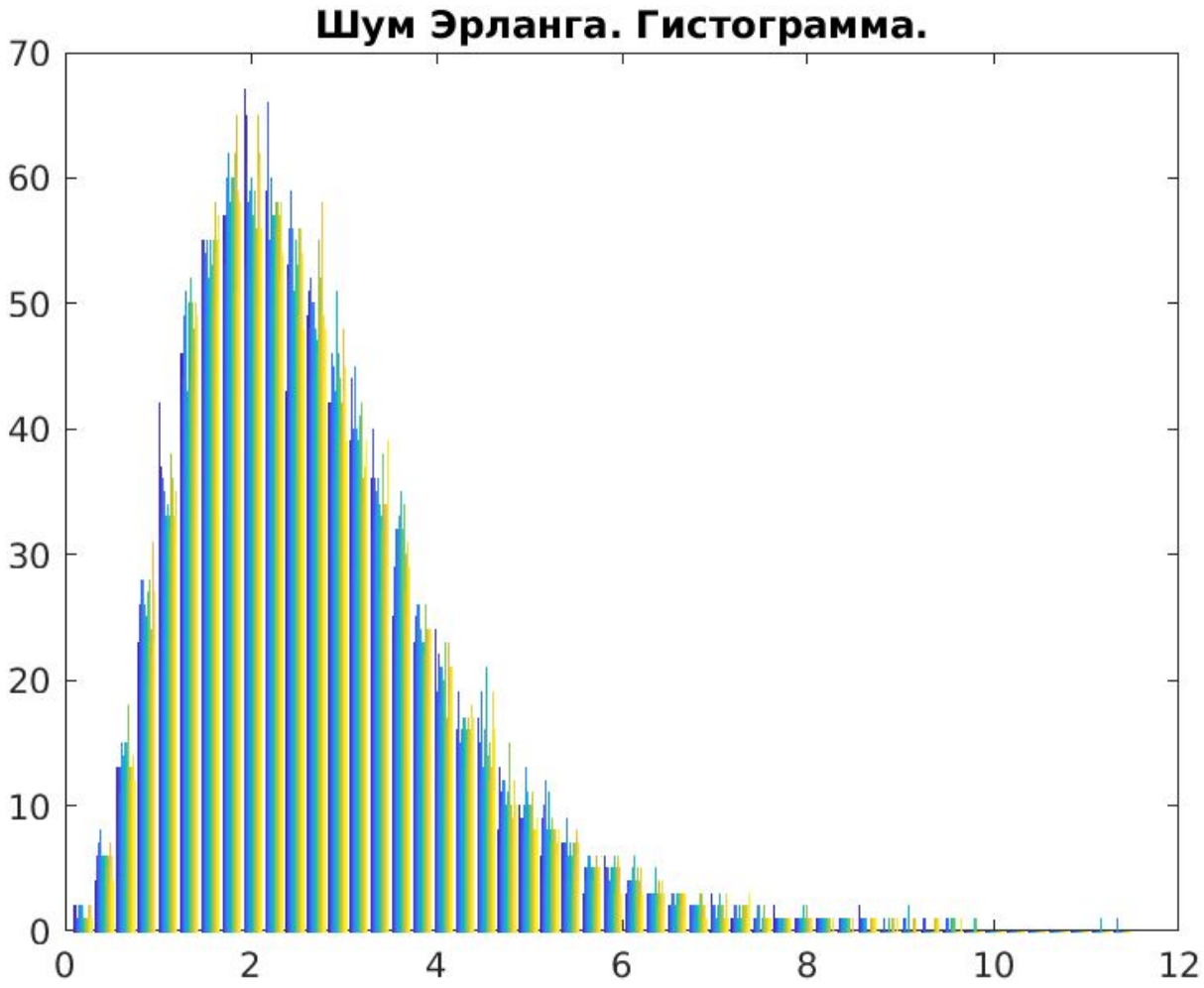


Рисунок 11 - гистограмма шума Эрланга

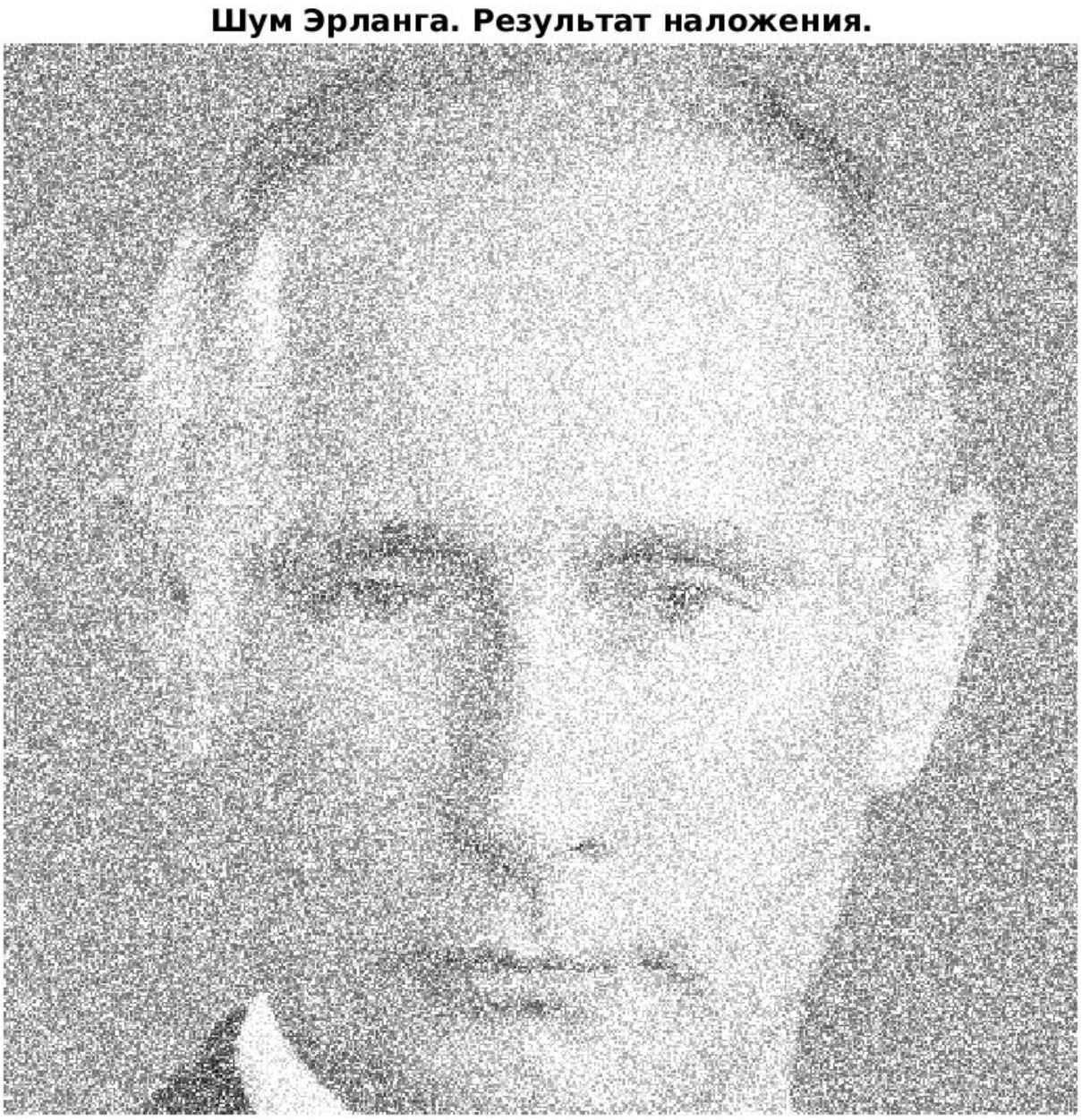


Рисунок 12 - результат наложения шума Эрланга

Арифметическое среднее

```
function f = arifmeticMeanFilter(g, m, n)
    g = im2double(g);
    [rows, columns] = size(g);
    f = zeros(rows, columns);
    halfWindowSize = fix(m/2);
    for i=halfWindowSize + 1:(rows - halfWindowSize)
        for j=halfWindowSize + 1:(columns - halfWindowSize)
            wind = g((i-halfWindowSize) : (i + halfWindowSize),
(j-halfWindowSize) : (j + halfWindowSize));
            f(i,j) = sum(wind,'all')/(m*n);
        end
    end
end
```

Геометрическое среднее

```
function f = geometricMeanFilter(g, m, n)
    g = im2double(g);
    f = real(exp(imfilter(log(g), ones(m, n), 'replicate')).^(1 /
m / n));
end
```

Гармоническое среднее

```
function f = harmonicMeanFilter(g, m, n)
    f = m * n ./ imfilter(1./(g + eps), ones(m, n), 'replicate');
end
```

Контргармоническое среднее

```
function f = contrHarmonicMeanFilter(g, m, n)
    q=1.5;
    f = imfilter(g.^(q+1), ones(m, n), 'replicate');
    f = real(f ./ (imfilter(g.^q, ones(m, n) , 'replicate') +
eps));
end
```

Медиана

```
function f = medianFilter(g, m, n)
    g = im2double(g);
    [rows, columns] = size(g);
    f = zeros(rows, columns);
    halfWindowSize = fix(m/2);
    for i=halfWindowSize + 1:(rows - halfWindowSize)
        for j=halfWindowSize + 1:(columns - halfWindowSize)
            wind = g((i-halfWindowSize) : (i + halfWindowSize),
(j-halfWindowSize) : (j + halfWindowSize));
```

```
            f(i,j) = median(wind,'all');
        end
    end
end
```

Максимум

```
function f = maxFilter(g, m, n)
    g = im2double(g);
    [rows, columns] = size(g);
    f = zeros(rows, columns);
    halfWindowSize = fix(m/2);
    for i=halfWindowSize + 1:(rows - halfWindowSize)
        for j=halfWindowSize + 1:(columns - halfWindowSize)
            wind = g((i-halfWindowSize) : (i + halfWindowSize),
(j-halfWindowSize) : (j + halfWindowSize));
            res = reshape(wind, 1, m*n);
            f(i,j) = max(res);
        end
    end
end
```

Минимум

```
function f = minFilter(g, m, n)
    g = im2double(g);
    [rows, columns] = size(g);
    f = zeros(rows, columns);
    halfWindowSize = fix(m/2);
    for i=halfWindowSize + 1:(rows - halfWindowSize)
        for j=halfWindowSize + 1:(columns - halfWindowSize)
            wind = g((i-halfWindowSize) : (i + halfWindowSize),
(j-halfWindowSize) : (j + halfWindowSize));
            res = reshape(wind, 1, m*n);
            f(i,j) = min(res);
        end
    end
end
```

Срединная точка

```
function f = midPointFilter(g, m, n)
    g = im2double(g);
    [rows, columns] = size(g);
    f = zeros(rows, columns);
    halfWindowSize = fix(m/2);
    for i=halfWindowSize + 1:(rows - halfWindowSize)
        for j=halfWindowSize + 1:(columns - halfWindowSize)
            wind = g((i-halfWindowSize) : (i + halfWindowSize),
(j-halfWindowSize) : (j + halfWindowSize));
            res = reshape(wind, 1, m*n);
```

```
            f(i,j) = (max(res)+min(res))/2;
        end
    end
end
```

Альфа-усечённое среднее

```
function f = alfaTrimFilter(g, m, n)
    d = 2;
    g = im2double(g);
    f = imfilter(g, ones(m, n), 'symmetric');
    for k = 1:d/2
        f = imsubtract(f, ordfilt2(g, k, ones(m, n),
'symmetric'));
    end
    for k = (m*n - (d/2) + 1):m*n
        f = imsubtract(f, ordfilt2(g, k, ones(m, n),
'symmetric'));
    end
    f = f / (m*n - d);
end
```

Корреляция Пирсона после применения фильтров.

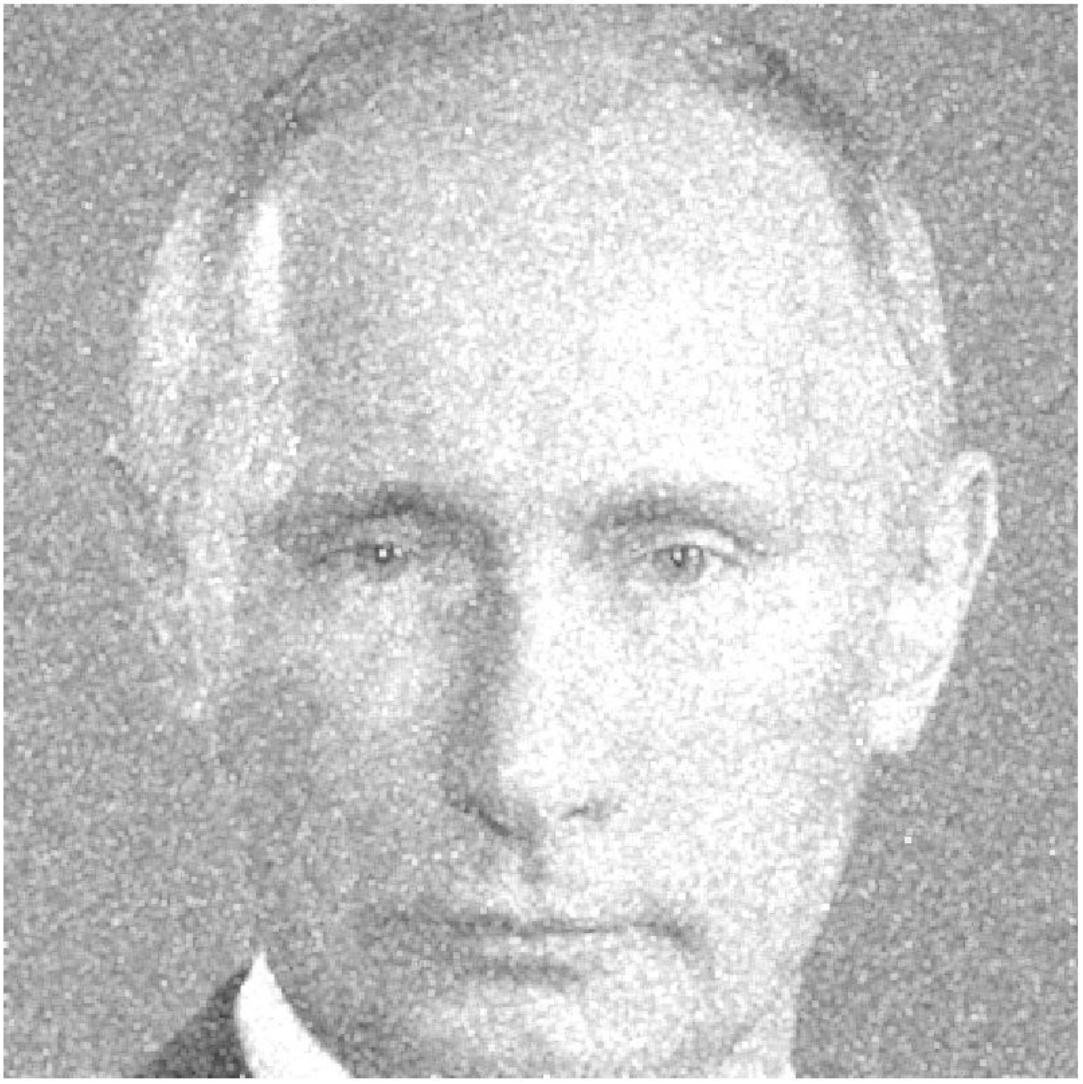
	Равномерный	Гауссов	Логарифмическ и нормальный	Реллея	Экспоненциальный	Эрланга
Арифметическое среднее	0.7367	0.8999	0.8294	0.8643	0.3646	0.7541
Геометрическое среднее	0.8021	0.9087	0.9370	0.9279	0.5132	0.8419
Гармоническое среднее	**0.8087**	0.0181	**0.9391**	0.9271	0.6293	**0.8666**
Контргармоническое среднее	0.7735	0.8894	0.9153	**0.9304**	0.0858	0.5730
Медиана	0.6243	0.8755	0.8201	0.8912	0.3601	0.7341
Максимумы	0.6847	0.7793	0.6741	0.7030	0.1164	0.3972
Минимумы	0.7822	0.8063	0.8398	0.8807	**0.7316**	0.8580
Срединная точка	0.7896	0.8615	0.8398	0.8308	0.2151	0.6085
α-усеченное среднее	0.7534	**0.9166**	0.9331	0.9181	0.3852	0.8040

Равномерный шум. Гармоническое среднее. Результат фильтрации.

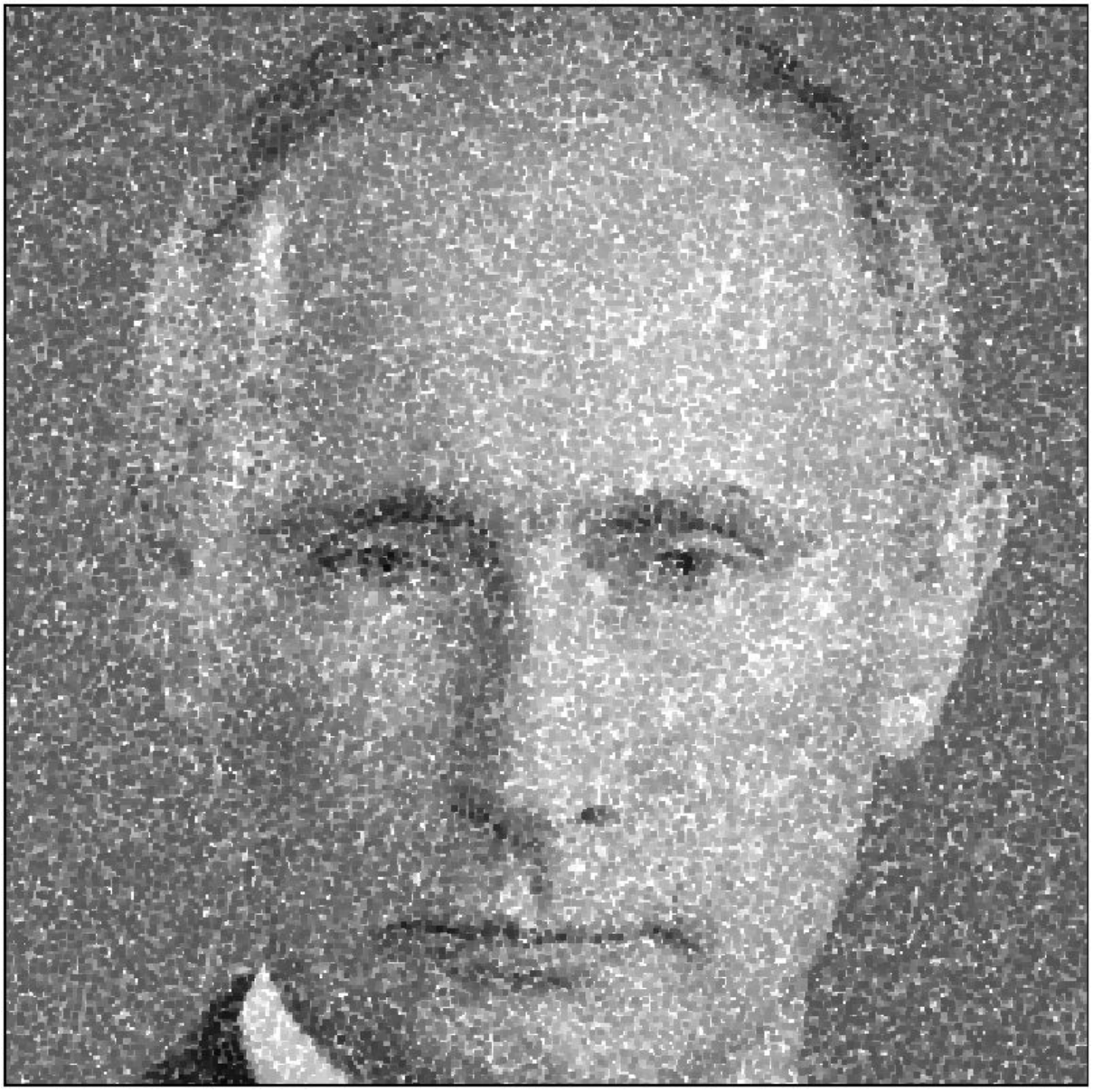
Гауссов шум. Альфа-усечённое среднее. Результат фильтрации.

Логарифмически нормальный шум. Геометрическое среднее. Результат фильтрации.

Шум Релея. Контргармоническое среднее. Результат фильтрации.

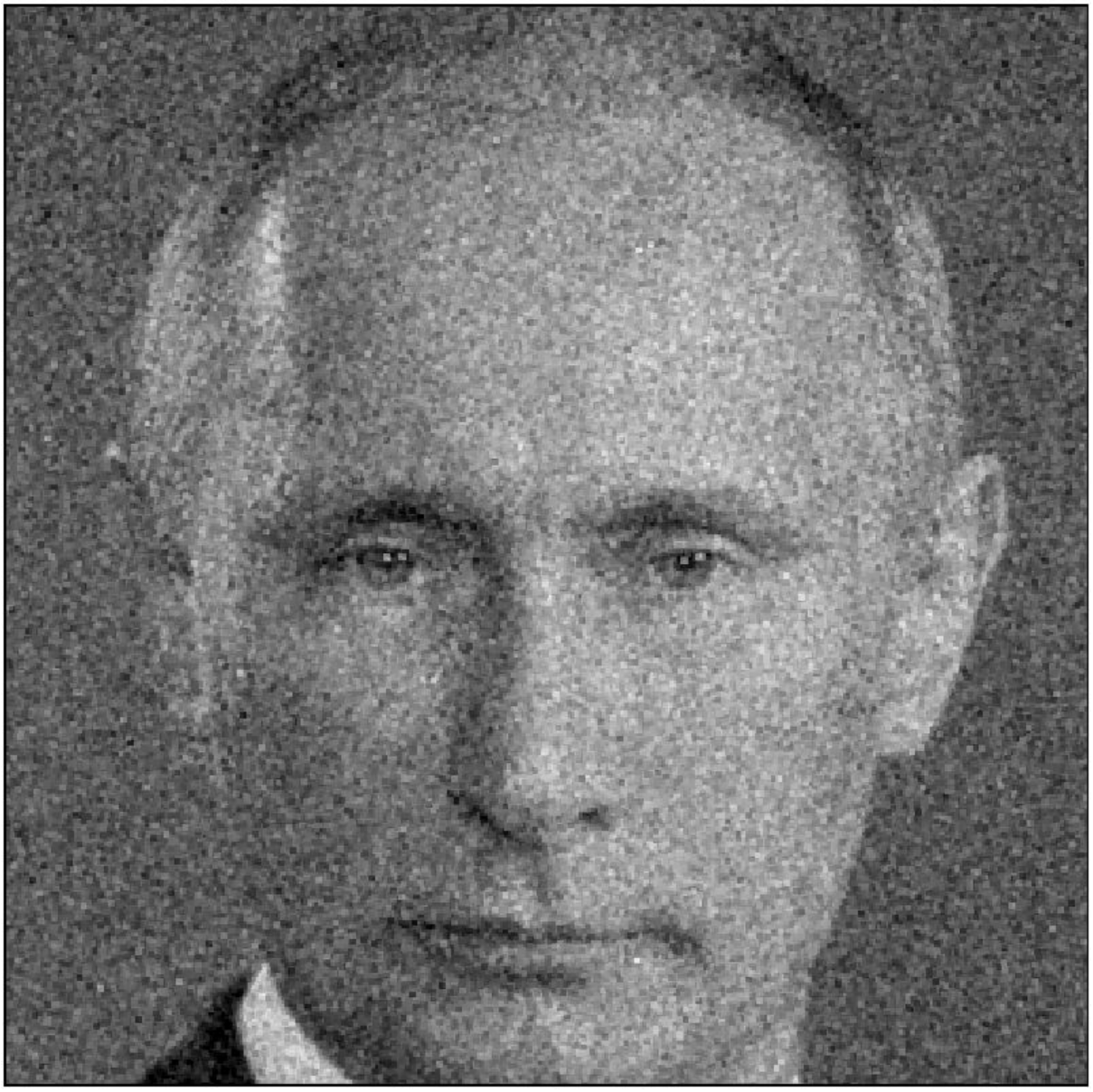


Экспоненциальный шум. Минимум. Результат фильтрации.



Шум Эрланга. Медиана. Результат фильтрации.

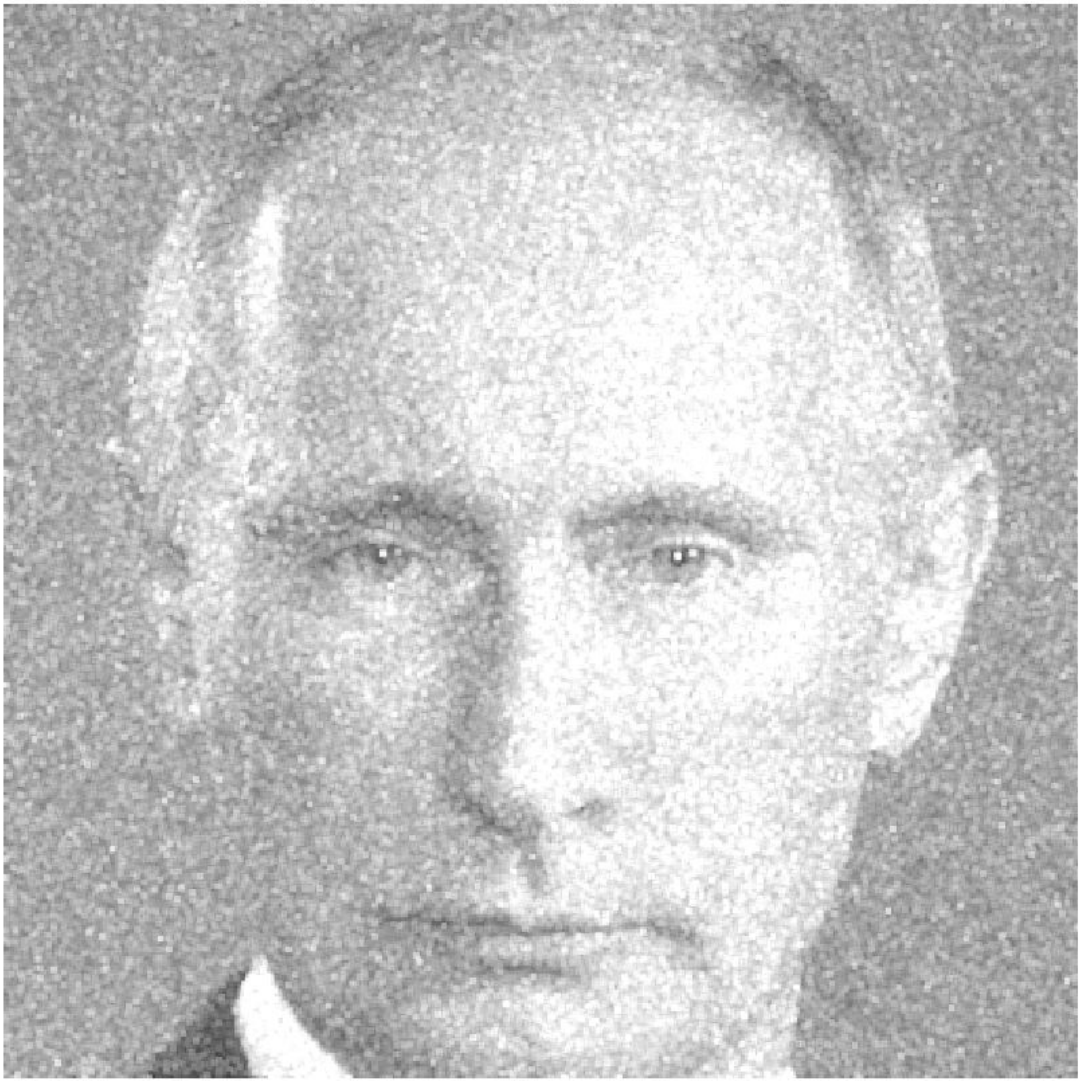
Гауссов шум. Срединная точка. Результат фильтрации.



Гауссов шум. Арифметическое среднее. Результат фильтрации.

Шум Релея. Максимум. Результат фильтрации.

Шум Релея. Контргармоническое среднее. Результат фильтрации.



Вывод

В ходе данной работы были реализованы генераторы шумов 6 различных видов, а также набор фильтров, позволяющих восстановить зашумлённое изображение. В ходе работы бы определено, что наиболее подходящим фильтром для Равномерного, Логарифмически нормального шумов и шума Эрланга является фильтр по гармоническому среднему. Для гауссова шума - фильтр альфа-усечённого среднего, Релея - контргармонического среднего, для экспоненцицального шума - фильтр минимумов.