Федеральное агентство связи

Ордена трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе №5

по дисциплине «Введение в профессию»

Выполнил: студент группы БВТ1905

Виноградов Георгий Владимирович

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019

**Синтезирование изображений в MatLab.**

**Цель работы:**

Изучить возможности по синтезированию цифровых изображений, генерации шума различной природы возникновения. Использование базовых арифметических и геометрических операций над цифровым изображением.

**Индивидуальное задание:**

****

**Пример кода и вывода данных:**

clear, clc;

%1, 2)

gray\_image = imnoise2('uniform', 800, 800);

imagesc(gray\_image);

colormap(gray);

imshow(gray\_image);

%3)

h1 = figure('Name', 'gist 1');

hold on

[h, cx] = imhist(gray\_image);

plot(cx, h);

hold off;

saveas(h1, 'gist1.jpg');

%4)

obj1\_image = zeros(400, 400);

obj1\_image(170:230,50:350) = 1;

obj1\_image(50:350,170:230) = 1;

imagesc(obj1\_image);

imwrite(obj1\_image, 'obj1.bmp','bmp');

%5)

obj1\_img\_bilinear\_s = imresize(obj1\_image, 0.5, 'bilinear');

obj1\_img\_bilinear\_b = imresize(obj1\_image, 2, 'bilinear');

obj1\_img\_nearest\_s = imresize(obj1\_image, 0.5, 'nearest');

obj1\_img\_nearest\_b = imresize(obj1\_image, 2, 'nearest');

imwrite(obj1\_img\_bilinear\_s, 'obj1\_bilinear\_05.bmp', 'bmp');

imwrite(obj1\_img\_bilinear\_b, 'obj1\_bilinear\_2.bmp', 'bmp');

imwrite(obj1\_img\_nearest\_s, 'obj1\_nearest\_05.bmp', 'bmp');

imwrite(obj1\_img\_nearest\_b, 'obj1\_nearest\_2.bmp', 'bmp');

%6)

obj2\_3\_img = imnoise2('uniform', 600, 600);

orig\_obj2\_3\_img = obj2\_3\_img;

disk\_m = strel('disk', 101, 8);

obj2\_3\_img(50:250, 50:250) = 1.\*(disk\_m.Neighborhood > 0) +...

orig\_obj2\_3\_img(50:250, 50:250).\*(disk\_m.Neighborhood == 0);

disk\_m = strel('disk', 91, 8);

obj2\_3\_img(60:240, 60:240) = orig\_obj2\_3\_img(60:240, 60:240).\*(disk\_m.Neighborhood > 0) +...

obj2\_3\_img(60:240, 60:240).\*(disk\_m.Neighborhood == 0);

disk\_m = strel('disk', 51, 8);

obj2\_3\_img(100:200, 100:200) = 1.\*(disk\_m.Neighborhood > 0) +...

orig\_obj2\_3\_img(100:200, 100:200).\*(disk\_m.Neighborhood == 0);

disk\_m = strel('disk', 41, 8);

obj2\_3\_img(110:190, 110:190) = orig\_obj2\_3\_img(110:190, 110:190).\*(disk\_m.Neighborhood > 0) +...

obj2\_3\_img(110:190, 110:190).\*(disk\_m.Neighborhood == 0);

dmd\_m = strel('diamond', 100);

obj2\_3\_img(390:510, 350:550) = 1.\*(dmd\_m.Neighborhood(40:160, 1:201) > 0) +...

orig\_obj2\_3\_img(390:510, 350:550).\*(dmd\_m.Neighborhood(40:160, 1:201) == 0);

dmd\_m = strel('diamond', 90);

obj2\_3\_img(400:500, 360:540) = orig\_obj2\_3\_img(400:500, 360:540).\*(dmd\_m.Neighborhood(40:140, 1:181) > 0) +...

obj2\_3\_img(400:500, 360:540).\*(dmd\_m.Neighborhood(40:140, 1:181) == 0);

%7)

obj2\_3\_img\_h = obj2\_3\_img(1:end, end:-1:1);

imwrite(obj2\_3\_img\_h, 'obj2\_3\_hor.bmp', 'bmp');

%8)

obj2\_3\_img\_v = obj2\_3\_img(end:-1:1, 1:end);

imwrite(obj2\_3\_img\_v, 'obj2\_3\_vert.bmp', 'bmp');

%9)

rt\_angle = 45\*pi/180;

T = [cos(rt\_angle) sin(rt\_angle) 0; -sin(rt\_angle) cos(rt\_angle) 0; 0 0 1];

obj2\_3\_img\_rot = imtransform(obj2\_3\_img , maketform('affine', T));

imwrite(obj2\_3\_img\_rot, 'obj2\_3\_rot45.bmp', 'bmp');

%10)

rt\_angle = -45\*pi/180;

T = [cos(rt\_angle) sin(rt\_angle) 0; -sin(rt\_angle) cos(rt\_angle) 0; 0 0 1];

obj2\_3\_img\_rot = imtransform(obj2\_3\_img , maketform('affine', T));

imwrite(obj2\_3\_img\_rot, 'obj2\_3\_rot-45.bmp', 'bmp');

imshow(obj2\_3\_img\_rot);

function R = imnoise2(type,M,N,a,b)

% set default values.

if nargin == 1

a = 0; b = 1;

M = 1; N = 1;

elseif nargin == 3

a = 0; b = 1;

end

%as we need only small letters as the type so...

switch lower(type)

case 'uniform'

R = a + (b-a)\*rand(M,N);

case 'gaussian'

R = a + b\*randn(M,N);

case 'salt & pepper'

if nargin <= 3

a = 0.05; b = 0.05;

end

% check to make sure that Pa + Pb is not > 1.

if (a + b) > 1

error('The sum of the Pa and Pb cannot exeed 1.')

end

R(1:M,1:N) = 0.5;

% Generate an M by N array of uniformly distributed random numbers in the

% range (0,1). Then, Pa\*(M\*N) of them will have values <= a. The

% coordinates of these points we call 0 (pepper noise). Similarly, Pb\*(M\*N)

% points will have values in the range > a & <= (a+b). These we call

% (salt noise).

X = rand(M,N);

c = find(X<=a);

R(c) = 1;

u = a + b;

c = find(X > a & X <= u);

R(c) = 1;

case 'lognormal'

if nargin<=3

a = 1; b = 0.25;

end

R = a\*esp(b\*randn(M,N));

case 'Rayleigh'

R = a + (-b\*log(1-rand(M,N)))^0.5;

case 'exponential'

if nargin <= 3

a = 1;

end

if a <= 0

error('the value of a must b positive for exponential operation')

end

k = -1/a

R = k\*log(1 - rand(M,N));

case 'erlang'

if nargin <= 3

a = 2; b = 5;

end

if (b ~= round(b)| b <= 0)

error('Parameter b should b a negative value for erlang')

end

k = -1/a;

R = zeros(M,N);

for j = 1:b

R = R + k\*log(1 - rand(M,N));

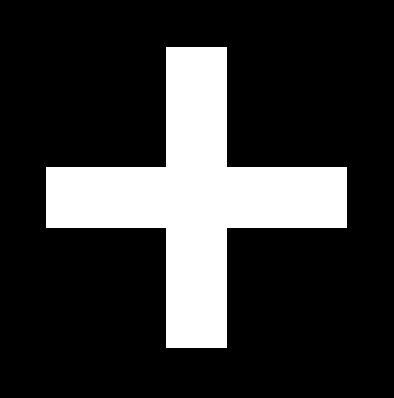
end

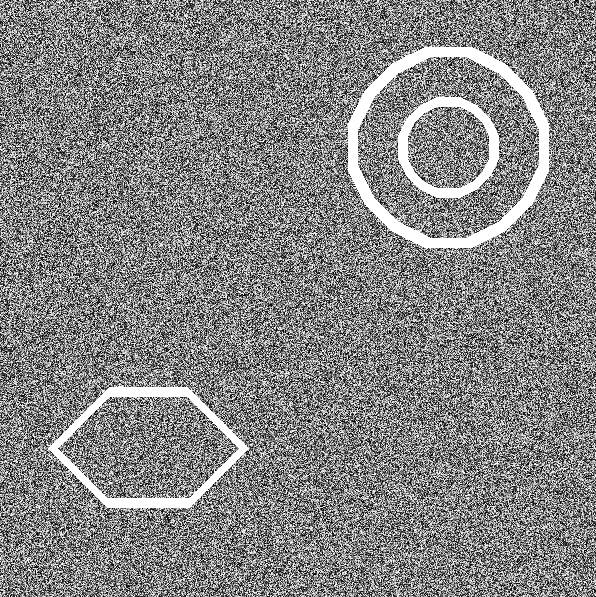
otherwise

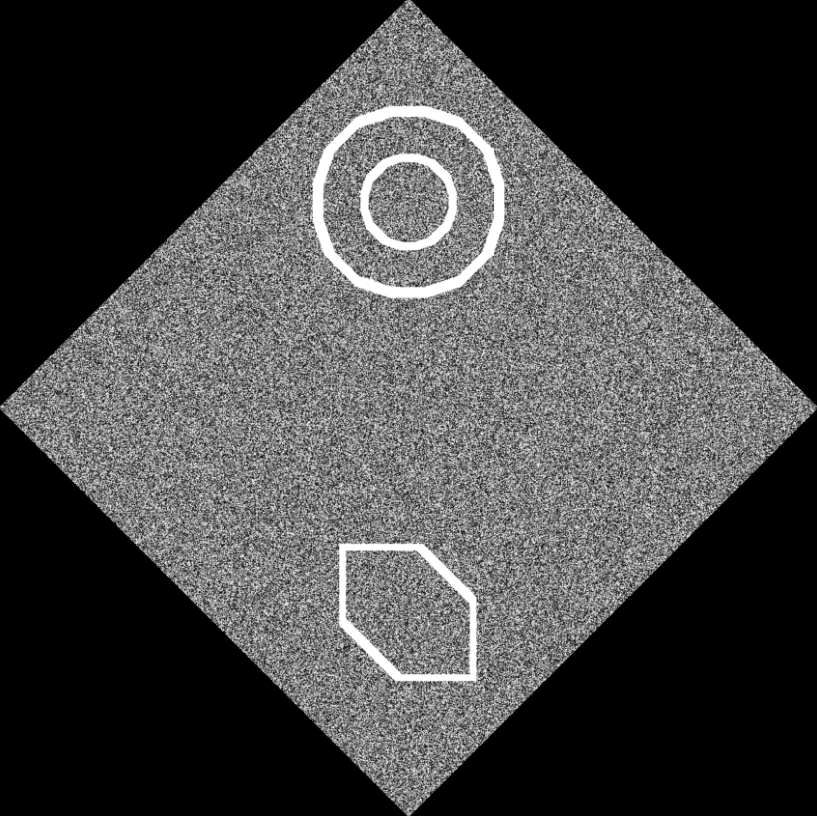
error('Unknown distribution type.')

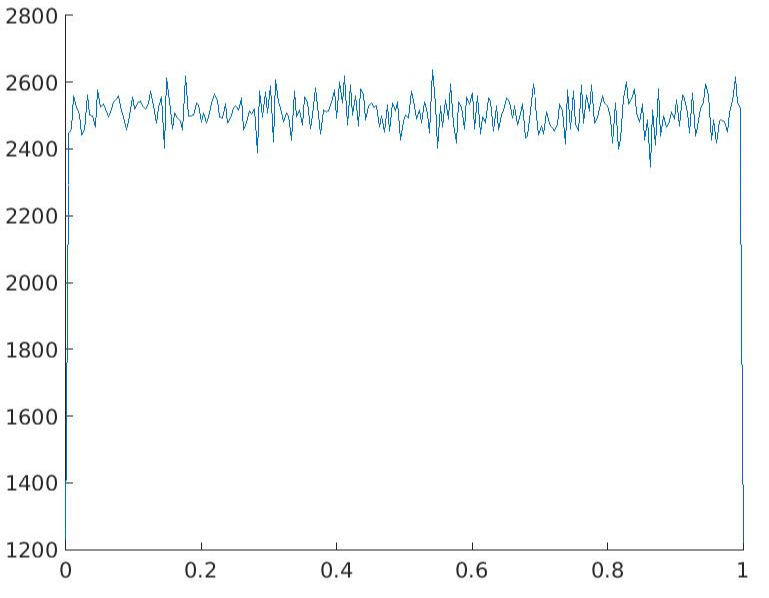
end

end









**Вывод:**

В этой лабораторной работе я научился синтезировать цифровое изображение, генерировать шум различной природы возникновения и применять базовые арифметические и геометрические операции над цифровым изображением.