Table of Contents

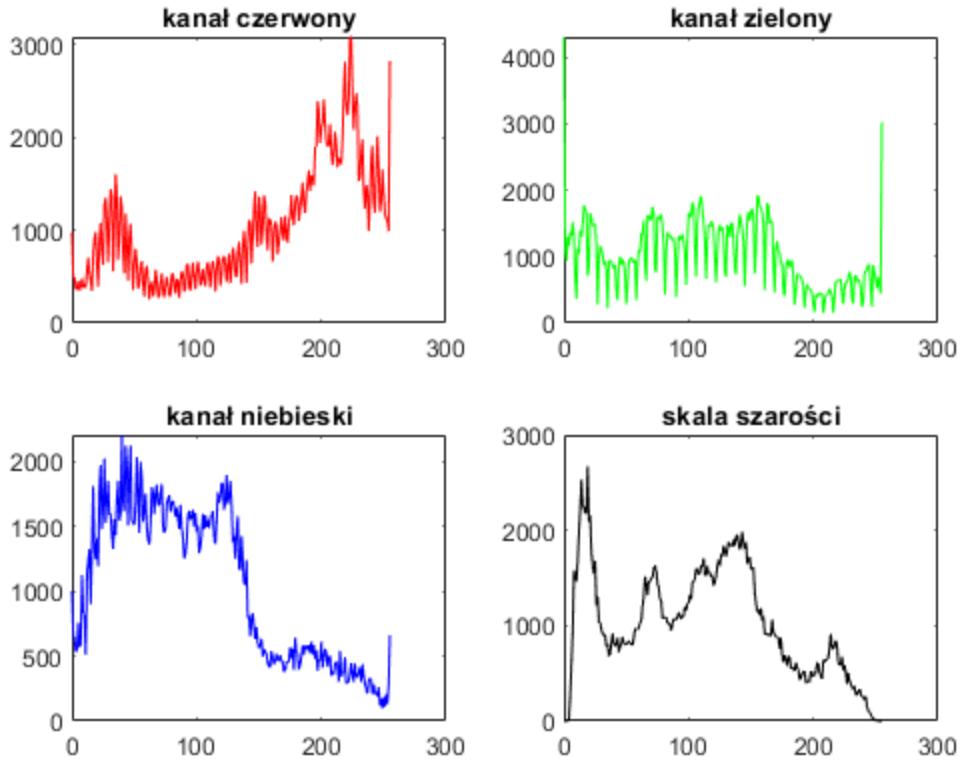
Sprawozdanie laboratorium 6-7

```
% imi# i nazwisko: Daniel Siutkowski
% numer albumu: 47035
% kierunek: teleinformatyka
% semestr: 4
% przedmiot: Przetwarzanie Obrazów
```

Zadanie 1

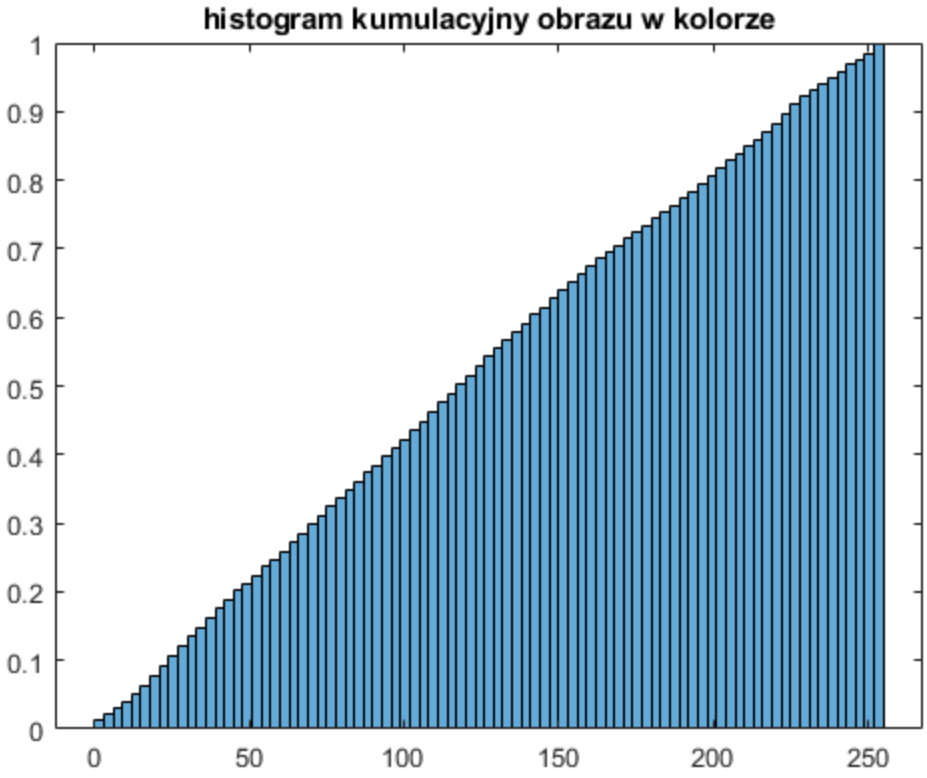
```
clear all, close all, clc;
imC = imread('PictureColor.bmp'); % kolorowy obraz
imG = imread('PictureBW.bmp'); % obraz w skali szaro#ci

% wydzielenie warto#ci z poszczególnych kana#ów
Red = imC(:,:,1);
Green = imC(:,:,2);
Blue = imC(:,:,3);

% stworzenie histogramów
[yRed, x] = imhist(Red);
[yGreen, x] = imhist(Green);
[yBlue, x] = imhist(Blue);
[yGray, xGray] = imhist(imG);

% wy#wietlenie obrazów
subplot(1,2,1),imshow(imC);title('Wczytany obraz w kolorze');
subplot(1,2,2),imshow(imG);title('Wczytany obraz w odcieniach
 szaro#ci');
figure;
```

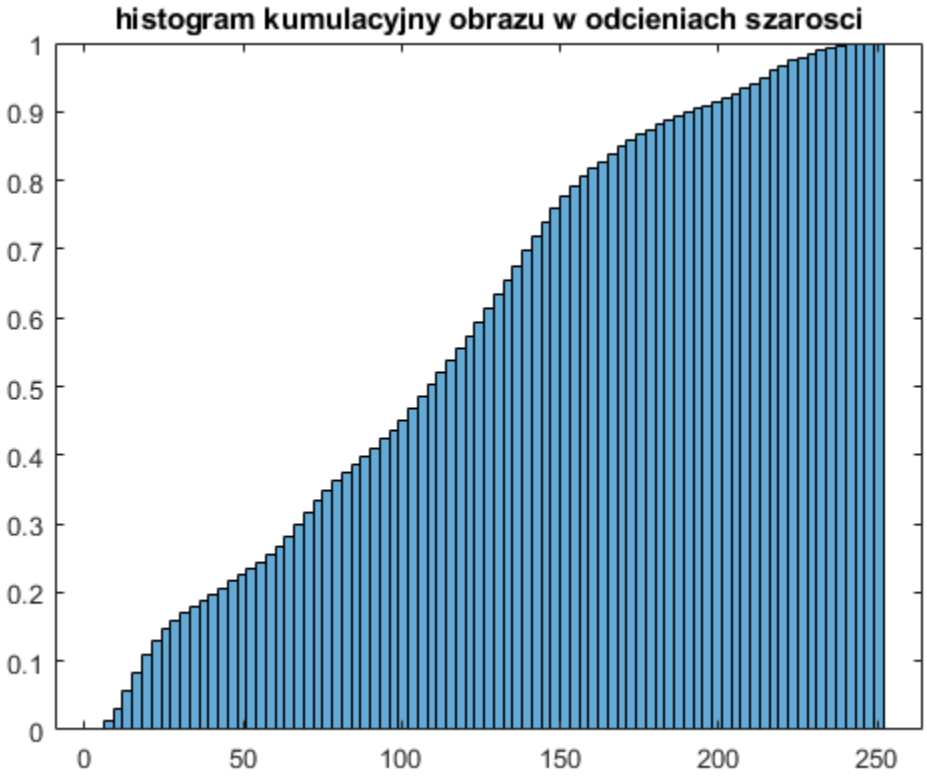
```
% wy#wietlenie histogramów
subplot(2,2,1), plot(x, yRed, 'Red');title('kana# czerwony');
subplot(2,2,2), plot(x, yGreen, 'Green');title('kana# zielony');
subplot(2,2,3), plot(x, yBlue, 'Blue');title('kana# niebieski');
subplot(2,2,4), plot(xGray, yGray, 'Black');title('skala szaro#ci');
pause(0.5);
```



Zadanie 2

```
clear all, close all, clc;
imC = imread('PictureColor.bmp'); % kolorowy obraz
imG = imread('PictureBW.bmp'); % obraz w skali szaro#ci

histogram(imC,'Normalization','cdf'),title('histogram kumulacyjny
 obrazu w kolorze');figure;
histogram(imG,'Normalization','cdf'),title('histogram kumulacyjny
 obrazu w odcieniach szarosci');
pause(0.5);
```


histogram kumulacyjny obrazu w kolorze
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200
250


histogram kumulacyjny obrazu w odcieniach szarosci
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200
250

Zadanie 3

```
clear all, close all, clc;
imC = imread('PictureColor.bmp'); % kolorowy obraz
imG = imread('PictureBW.bmp'); % obraz w skali szaro#ci

imBW = im2bw(imC, 0.2);
imBW2 = im2bw(imC, 0.4);
imBW3 = im2bw(imC, 0.6);
%BW = imbinarize(imG,'adaptive', 'Sensitivity',0.7);

subplot(1,3,1),imshow(imBW);title('prog 0.2');
subplot(1,3,2),imshow(imBW2);title('prog 0.4');
subplot(1,3,3),imshow(imBW3);title('prog 0.6');
pause(0.5);
```

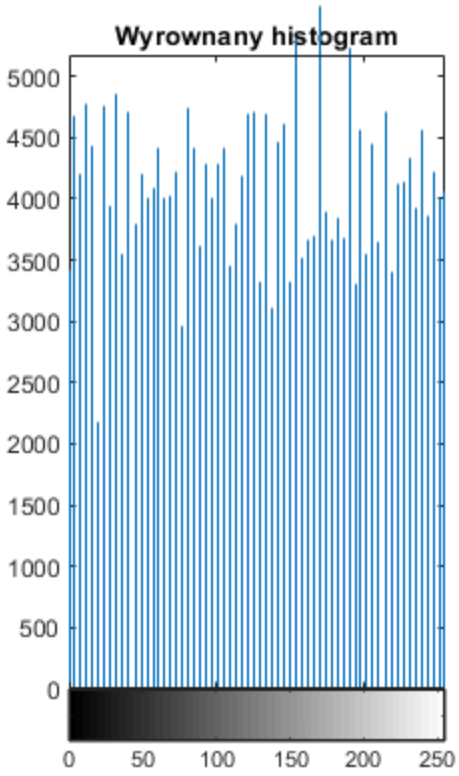
Zadanie 4

```
clear all, close all, clc;
imC = imread('PictureColor.bmp'); % kolorowy obraz
imG = imread('PictureBW.bmp'); % obraz w skali szaro#ci

imshow(otsu(imG));
pause(0.5);
```

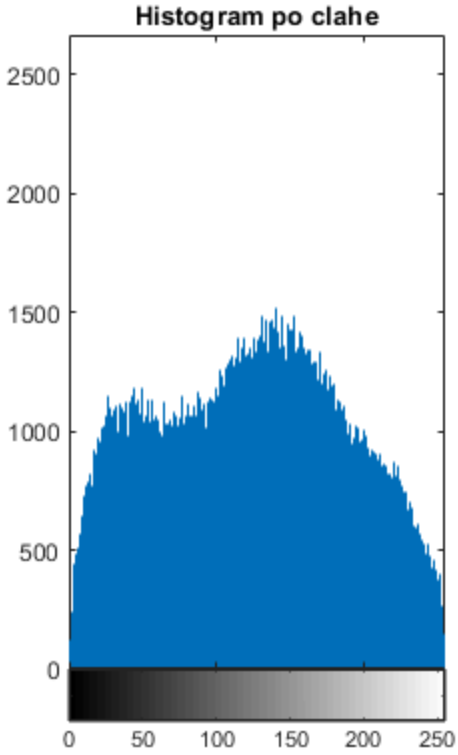
Zadanie 5

```
clear all, close all, clc;
imG = imread('PictureBW.bmp');

imH = histeq(imG);
subplot(1,2,1), imshow(imH),title('Obraz po wyrownaniu histogramu');
subplot(1,2,2), imhist(imH),title('Wyrownany histogram');
figure;

imA = adapthisteq(imG);
subplot(1,2,1), imshow(imA), title('Obraz po clahe');
subplot(1,2,2), imhist(imA),title('Histogram po clahe'); % na
 subplocie histogram jest bardzo #ci#ni#ty
figure;

subplot(1,2,1), imshow(imH);
subplot(1,2,2), imshow(imA);
sgtitle('Porównanie wyników');
figure;
```

```
level = graythresh(imH); % binaryzacja przed clahe
imB = imbinarize(imH,level);
level = graythresh(imA); % binaryzacja po clahe
imB2 = imbinarize(imA,level);

subplot(1,2,1), imshow(imB);
subplot(1,2,2), imshow(imB2);
sgtitle('Porównanie binaryzacji przed i po clahe');
pause(0.5);
```



Obraz po clahe



Porównanie wyników

Porównanie binaryzacji przed i po clahe

Zadanie 6

```
clear all, close all, clc;
imG = imread('PictureBW.bmp');
[row,col] = size(imG);
tablica = {128,64,32,16,8};
zerowanko = double(tablica{3}); %rezultat wykonywanych operacji jest
 #ci#le uzale#niony od warto#ci pobieranej z tablicy (1-5)
result = zeros(row,col);

for i = 1:row
   for j = 1:col
        imGtoBIN = imG(i,j);
        result(i,j) = bitand(imGtoBIN, zerowanko);
   end
end

result = uint8(result);
imshow(result);
pause(0.5);
```

Zadanie 7

```
clear all, close all, clc;
im = imread('lew.jpeg');

Red = im(:,:,1); %5 bitow
Green = im(:,:,2); %7 bitow
Blue = im(:,:,3); %4 bity

[row,col] = size(Red);
tablica = {248,254,240};

resRed = zeros(row,col);
resGreen = zeros(row,col);
resBlue = zeros(row,col);

for i = 1:row
   for j = 1:col
        andForRed = Red(i,j);
        andForGreen = Green(i,j);
        andForBlue = Blue(i,j);
```

```
            resRed(i,j) = bitand(andForRed, tablica{1});
            resGreen(i,j) = bitand(andForGreen, tablica{2});
            resBlue(i,j) = bitand(andForBlue, tablica{3});
        end
    end

    allTogether = cat(3, resRed, resGreen, resBlue);
    allTogether = uint8(allTogether);

    subplot(1,2,1),imshow(im),title('Obraz bazowy');
    subplot(1,2,2),imshow(allTogether),title('Obraz wynikowy');
    pause(0.5);
```

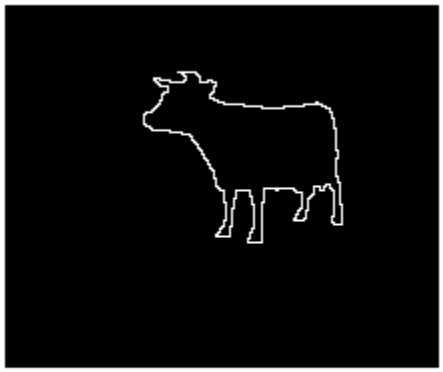
Zadanie 8

```
clear all, close all, clc;
im = imread('A04.bmp');

%erozja
erozja = erozjaLubDylatacja(im,1,0);

%dylatacja
dylatacja = erozjaLubDylatacja(im,0,1);

%otwarcie: erozja + dylatacja
```

```
otwarcie = erozjaLubDylatacja(erozja,0,1);

%zamkni#cie: dylatacja + erozja
zamkniecie = erozjaLubDylatacja(dylatacja,1,0);

%Funkcje wbudowane
dylatacjaFunWbud = bwmorph(im, 'dilate');
erozjaFunWbud = bwmorph(im, 'erode');
otwarcieFunWbud = bwmorph(im, 'open');
zamkniecieFunWbud = bwmorph(im, 'close');

%Porownanie funkcji

subplot(2,4,1), imshow(erozja),
title('erozja f wlasna');

subplot(2,4,2), imshow(erozjaFunWbud),
title('erozja f wbudowana');

subplot(2,4,3), imshow(dylatacja),
title('dylatacja f wlasna');

subplot(2,4,4), imshow(dylatacjaFunWbud),
title('dylatacja f wbudowana');

subplot(2,4,5), imshow(otwarcie),
title('otwarcie f wlasna');

subplot(2,4,6), imshow(otwarcieFunWbud),
title('otwarcie f wbudowana');

subplot(2,4,7), imshow(zamkniecie),
title('zamkniecie f wlasna');

subplot(2,4,8), imshow(zamkniecieFunWbud),
title('zamkniecie f wbudowana');
pause(0.5);
```

Zadanie 9

```
clear all, close all, clc;
im = imread('A04.bmp');
res = detekcjaKrawedzi(im,1,0);
imshow(res);
pause(0.5);
```



Zadanie 10

```
clear all, close all, clc;
im = imread('A04.bmp');
im_sp = imnoise(double(im), 'salt & pepper', 0.005);

subplot(1,2,1);imshow(im_sp);title('Obraz bazowy');
res = zad10funkcja(im_sp);

subplot(1,2,2);imshow(res);title('Obraz wynikowy');
pause(0.5);
```

Zad 8 - funkcja

```
function res = erozjaLubDylatacja(im,fp1,fp2)

    [rows,cols] = size(im);
    area = 3;
    counter = 0;

    [k,l] = deal(0,0);
    for i = ceil(area/2) : rows - floor(area/2)
        k=k+1;
        for j = ceil(area/2) : cols - floor(area/2)
            l=l+1;
            valsFromArea = reshape(im(i-(floor(area/2)) : i
+(floor(area/2)), j-(floor(area/2)) : j+(floor(area/2))),1,[]);
            for x = 1 : length(valsFromArea)
                if valsFromArea(x) == fp1
                    counter = counter + 1;
                end
            end
            if counter == 9
                im(k,l) = fp1;
            else
                im(k,l) = fp2;
            end
            counter = 0;
        end
        l=0;
    end

    res = im;
end
```

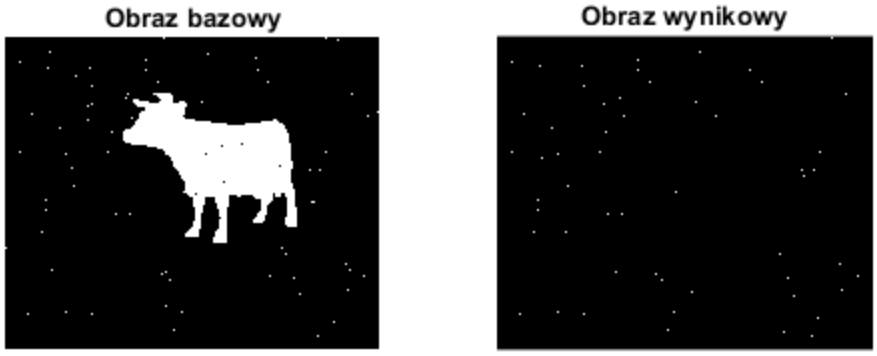
Zad 9 - funkcja

```
function res = detekcjaKrawedzi(im,v1,v2)

    [rows,cols] = size(im);
    area = 3;
    counter = 0;
```

```
    obrazRoboczy = im;

    for i = ceil(area/2) : rows - floor(area/2)
        for j = ceil(area/2) : cols - floor(area/2)
            valsFromArea = reshape(obrazRoboczy(i-(floor(area/2)) : i
+(floor(area/2)), j-(floor(area/2)) : j+(floor(area/2))),1,[]);

            for x = 1 : length(valsFromArea)
                if valsFromArea(x) == v1
                    counter = counter + 1;
                elseif valsFromArea((ceil(length(valsFromArea)/2))) ==
 v2
                    counter = counter + 1;
                end
            end
            if counter == 9
                im(i,j) = v2;
            else
                im(i,j) = im(i,j);
            end
            counter = 0;
        end
    end

    res = im;
end
```

Zad 10 - funkcja

```
function res = zad10funkcja(im)

    [rows,cols] = size(im);
    area = 3;
    counter = 0;
    newIm = zeros(rows,cols);

    for i = ceil(area/2) : rows - floor(area/2)
        for j = ceil(area/2) : cols - floor(area/2)
            valsFromArea = reshape(im(i-(floor(area/2)) : i
+(floor(area/2)), j-(floor(area/2)) : j+(floor(area/2))),1,[]);

            for x = 1 : length(valsFromArea)
                if valsFromArea(x) == 0
                    counter = counter + 1;
                end
            end
            if counter == 8
                if valsFromArea((ceil(length(valsFromArea)/2))) == 1
                    newIm(i,j) = 1;
                end
            else
                newIm(i,j) = newIm(i,j);
            end
            counter = 0;
        end
    end

    res = newIm;
end
```



Zad 4 - funkcja otsu

```
function [k]=otsu(a)
    b=a;
    [c,d]=size(b);
    b=reshape(b,[],1);
    [m,n]=size(b);
    weightb=zeros(3,256);
    weightf=zeros(3,256);
    r=1;
    l=1;
    for T=0:255
        [wb,wf,mb,mf,vrb,vrf]=cal(T,b,m);
        weightb(1,r)=wb;
        weightb(2,r)=mb;
        weightb(3,r)=vrb;
        r=r+1;
        weightf(1,l)=wf;
        weightf(2,l)=mf;
        weightf(3,l)=vrf;
        l=l+1;
    end
    %Within class variance
    wcv=zeros(1,256);
    for g=1:256
```

```
 wcv(1,g)=((weightb(1,g)*weightb(3,g))+((weightf(1,g)*weightf(3,g))));
    end
    % min(wcv)
    [threshold_value,val]=min(wcv);
     tval=(val-1)/256;
    %  b=imresize(b,[c d])
     a=im2bw(a,tval);
    k= medfilt2(a,[25 25]);
end
```

Zad 4 - funkcja cal (potrzebna do wykonania f otsu)

```
function [wb,wf,mb,mf,vrb,vrf]=cal(i,b,m)
      % weight
      wb=0;
      wf=0;
      mb=0;
      mf=0;
      b=double(b);
      vb=0;vf=0;
      vrb=0;
      vrf=0;
      for s=1:m
          if(b(s,1)<(i))
              wb=wb+1;
              mb=mb+b(s,1);
                 else
              wf=wf+1;
              mf=mf+b(s,1);
           end
      end
      %mean
      if(wb==0)
          mb=0;
       mf=mf/wf;
      elseif(wf==0)
      wf=0;
      mb=mb/wb;
      else
      mb=mb/wb;
      mf=mf/wf;
      end
      %weight
      wb=wb/m;
      wf=wf/m;
      %variance
      for t=1:m
          if(b(t,1)<(i))
      vrb=vrb+((b(t,1)-mb)^2);
      vb=vb+1;
```

```
            else
        vrf=vrf+((b(t,1)-mf)^2);
        vf=vf+1;
            end
        end
        if(vb==0)
            vrb=0;
            vrf=vrf/vf;
        elseif(vf==0)
            vrf=0;
            vrb=vrb/vb;
        else
        vrb=vrb/vb;
        vrf=vrf/vf;
        end
end
```

Published with MATLAB® R2021a