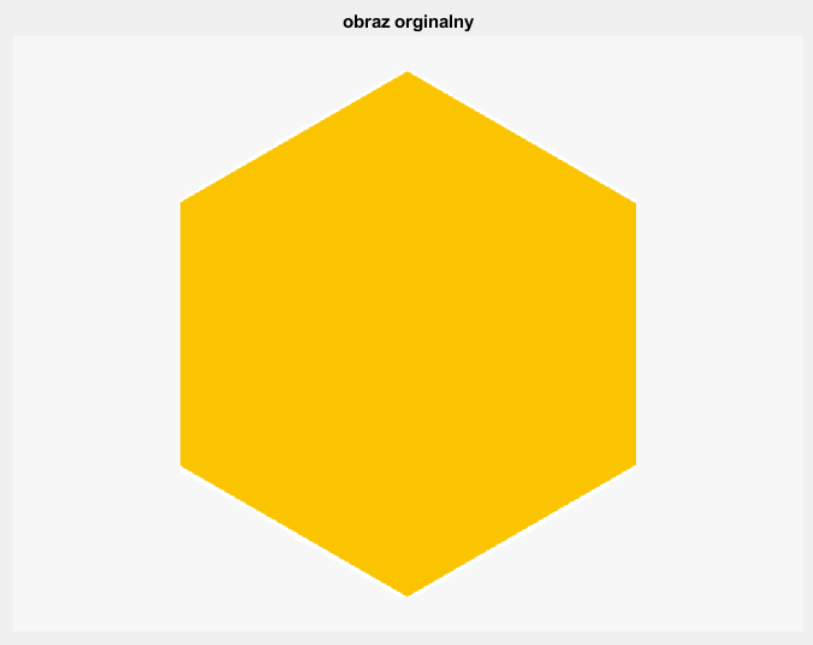
Sprawozdanie

Andrzej Żaba nr indeksu: **401490** Inżynieria Mechatroiczna

Zad 1.1

Obraz użyty w zadaniu:

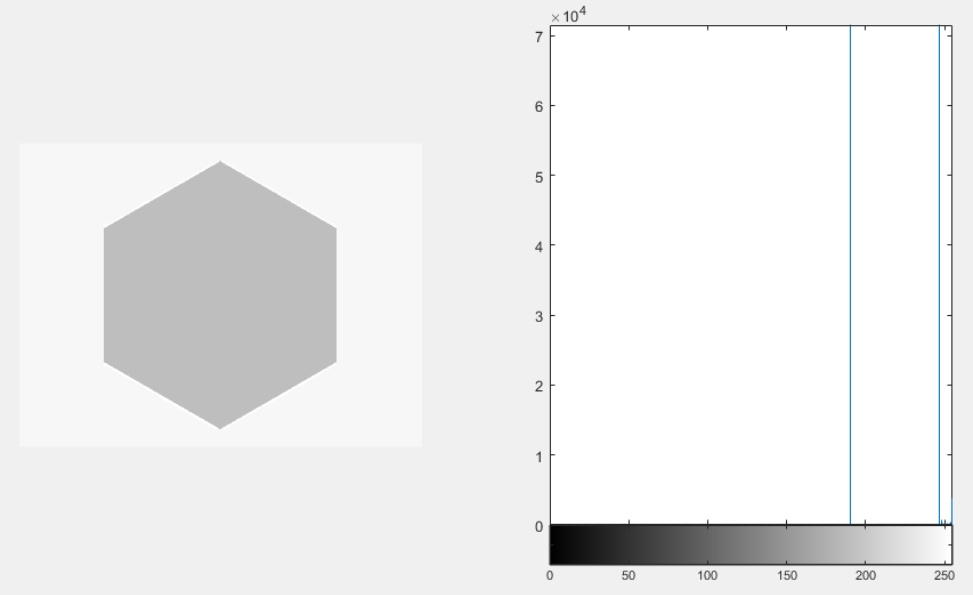


Całość kodu:

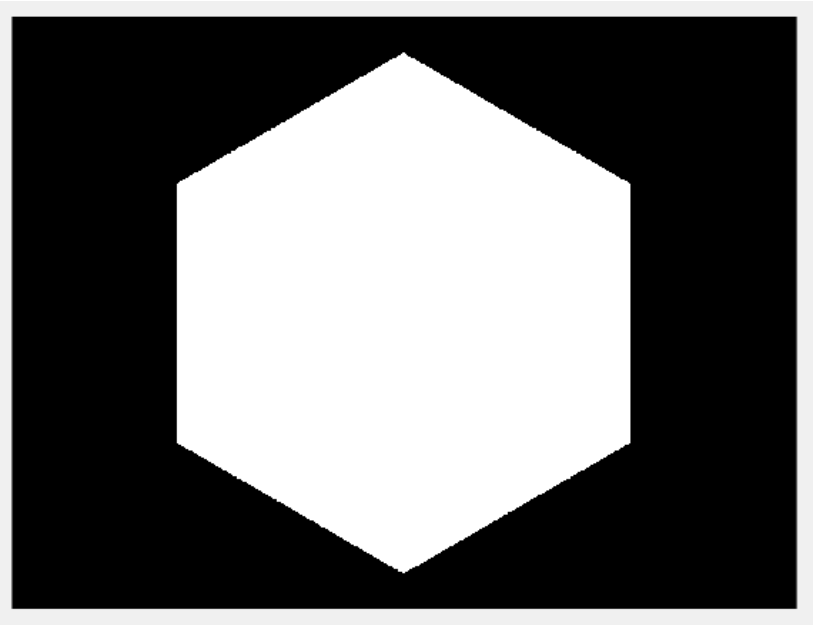
```
clc
close all
clear all
imfinfo('hex.png');
im = imread('hex.png');
figure(3)
imshow(im)
title('obraz orginalny')
img = im2gray(im);
figure(1)
subplot(1,2,1)
imshow(img)
subplot(1,2,2)
imhist(img)

%imbw = im2bw(img);
imbw = img < 200;
figure(2)
imshow(imbw)
%% filtracja -- aktualnie nie potrzebna

%% segmentacja
[L, num] = bwlabel(imbw,4);
figure(4)
imshow(imbw)
hold on
%% cechy
```

```
features = regionprops(L,'All')
%%
for i = 1:num
    rectangle('Position', features(i).BoundingBox, 'EdgeColor', 'red');
    centroids = cat(1, features.Centroid);
    plot(centroids(:,1), centroids(:,2), 'b*')
end

for k=1:length(features)
    x=features(k).Centroid(1,1);
    y=features(k).Centroid(1,2);
    plot(x,y,'*')
    a1 = tan((features(k).Orientation)*pi/180);
    b1 = y-a1*x;
    x1 = x-30;
    x2 = x+30;
    y1 = a1*x1+b1;
    y2 = a1*x2+b1;
    plot([x1 x2],[y1 y2], '-b')
end
hold off
```

Obraz w skali szarości oraz jego histogram:



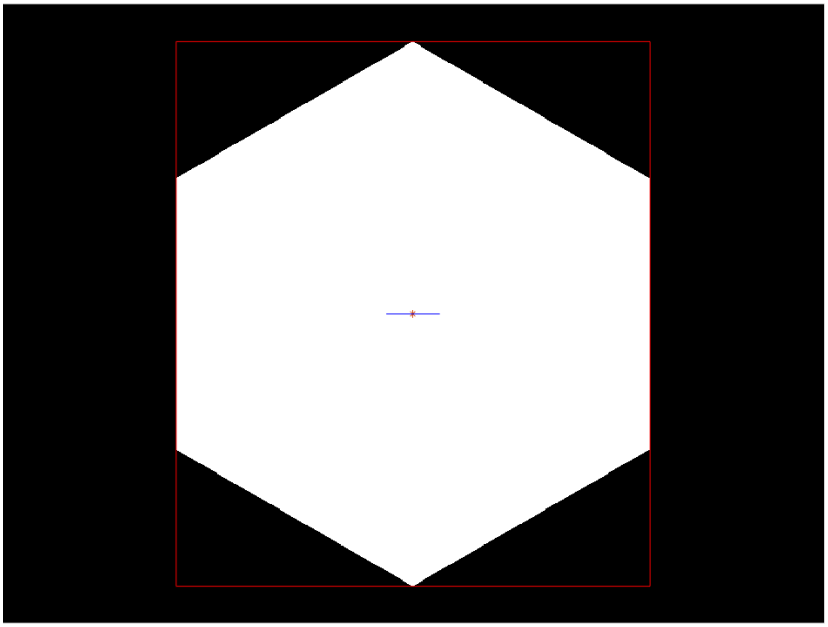
Obraz po binaryzacji:



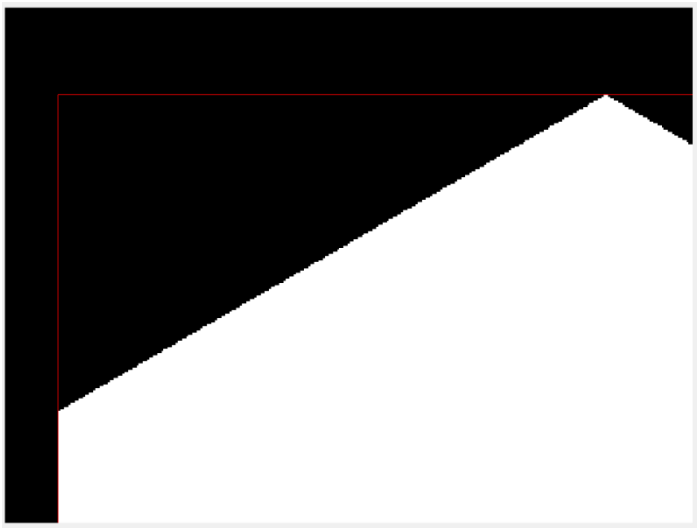
Wynik struktury **features**:

Field	Value
Area	242857
Centroid	[459.8328,347.9811]
BoundingBox	[194.5000,42.5000,531...
Subarrayldx	*1x2 cell*
MajorAxisLength	559.4143
MinorAxisLength	556.9896
Eccentricity	0.0930
Orientation	0.0248
ConvexHull	*639x2 double*
ConvexImage	*611x531 logical*
ConvexArea	243375
Circularity	0.9278
Image	*611x531 logical*
FilledImage	*611x531 logical*
FilledArea	242857
EulerNumber	1
Extrema	*8x2 double*
EquivDiameter	556.0712
Solidity	0.9979
Extent	0.7485
PixelldxList	*242857x1 double*
PixelList	*242857x2 double*
Perimeter	1.8136e+03
PerimeterOld	1.9215e+03
MaxFeretDiameter	612.3610
MaxFeretAngle	150.1275
MaxFeretCoordin...	[725.5000,195.5000;19...
MinFeretDiameter	530.1912
MinFeretAngle	59.9987
MinFeretCoordina...	[654.5000,541.5000;38...

Obraz po nałożeniu parametrów BoundingBox, Centroid, Orientation:



Wnioski: W tak prostym przypadku bardzo łatwo określić próg binaryzacji aby uzyskać kształt sześciokąta.



Parametry zobrazowane na zdjęciu zostały zaznaczone poprawnie – Bounding Box dokładnie otacza kształt a wyznaczony środek ciężkości znajduje się w środku figury.

Zad 1.2

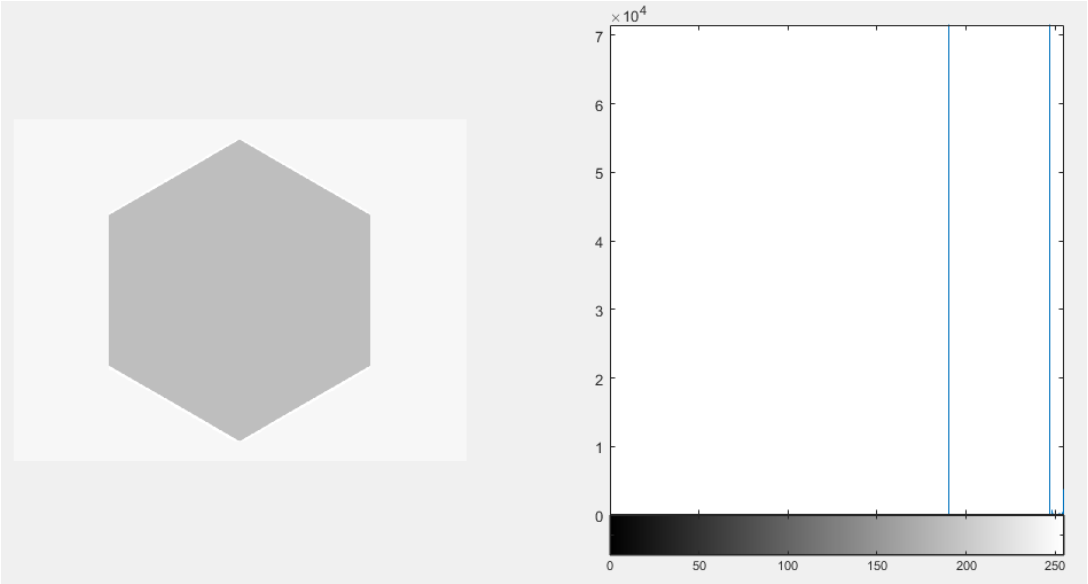
Obraz wykorzystany w zadaniu:

Całość kodu:

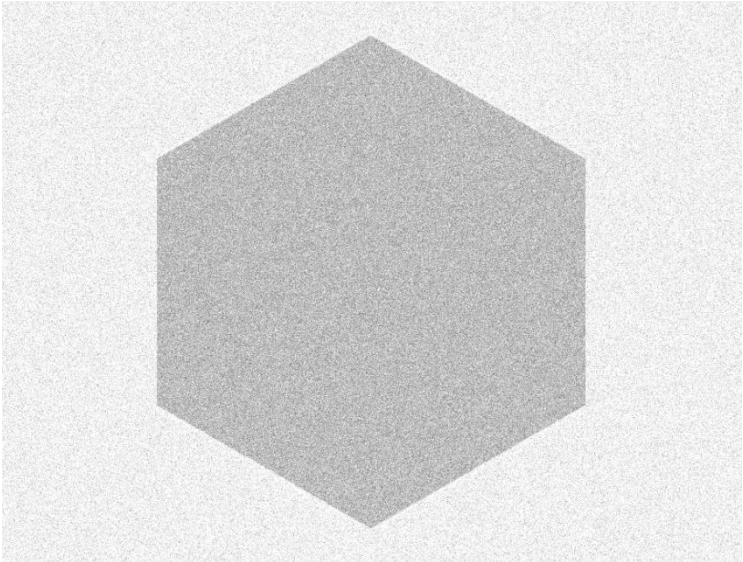
```
clc
close all
clear all
%% obraz
imfinfo('hex.png')
im = imread('hex.png');
figure(10)
imshow(im)
img = im2gray(im);
figure(1)
subplot(1,2,1)
imshow(img)
subplot(1,2,2)
imhist(img)

%% dodanie szumu
imgNoise = imnoise(img, 'gaussian', 0,0.01);  % im większy pierwszy
parametr tym jaśniejszy szum | drugi określa odchylenie standardowe  -
pierwszy przesuwa wykres Gaussa na osi X a drugi go rozszerza lub zweza
figure(2)
imshow(imgNoise)


%% Filtr
h2 = fspecial('gaussian',[9 9], 1000);
imF = imfilter(imgNoise,h2,'replicate');
figure(4)
subplot(1,2,1)
imshow(imF)
subplot(1,2,2)
imhist(imF)
%% Binaryzacja
%imbw = im2bw(I);
imba = imF < 215;
```

```
%% obraz
figure(5)
imshow(imba)
% open
se1 = strel('disk',5);
imOpen = imopen(imba,se1);


%% segmentacja
[L, num] = bwlabel(imOpen,4);
figure(7)
imshow(imOpen)
hold on
%% cechy
features = regionprops(L,'All');
%%
for i = 1:num
    rectangle('Position', features(i).BoundingBox, 'EdgeColor', 'red');
    centroids = cat(1, features.Centroid);
    plot(centroids(:,1), centroids(:,2), 'b*')
end

for k=1:length(features)
    x=features(k).Centroid(1,1);
    y=features(k).Centroid(1,2);
    plot(x,y,'*')
    a1 = tan((features(k).Orientation)*pi/180);
    b1 = y-a1*x;
    x1 = x-30;
    x2 = x+30;
    y1 = a1*x1+b1;
    y2 = a1*x2+b1;
    plot([x1 x2],[y1 y2], '-b')
end
hold off
```

Obraz w skali szarości wraz z histogramem:



Obraz z nałożonym szumem Gaussowskim:



Szum wygenerowano dzięki komendzie:

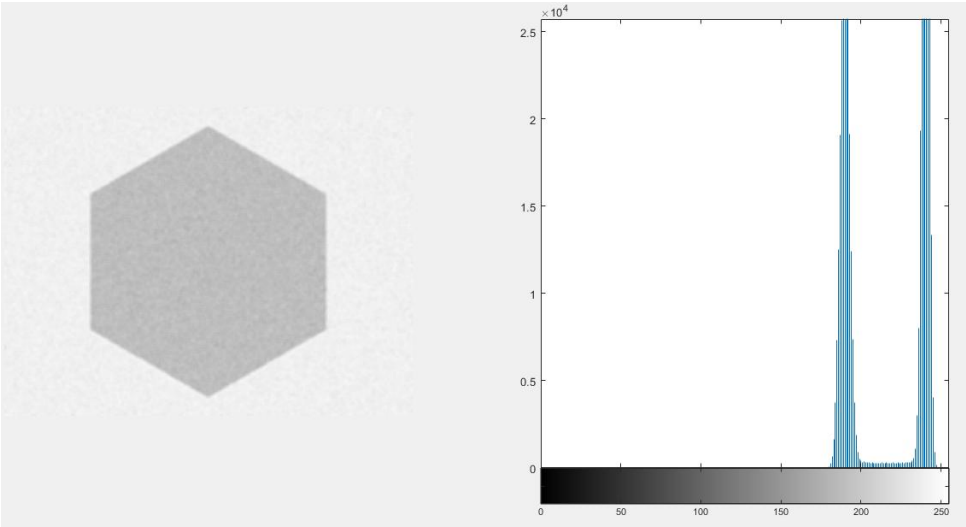
```
imgNoise = imnoise(img, 'gaussian', 0,0.01);
```

Używając parametrów: średnia wartość rozkładu = 0, wariancja = 0,01.

W celu filtracji obrazu możliwie redukującej szumy wykorzystano funkcje :

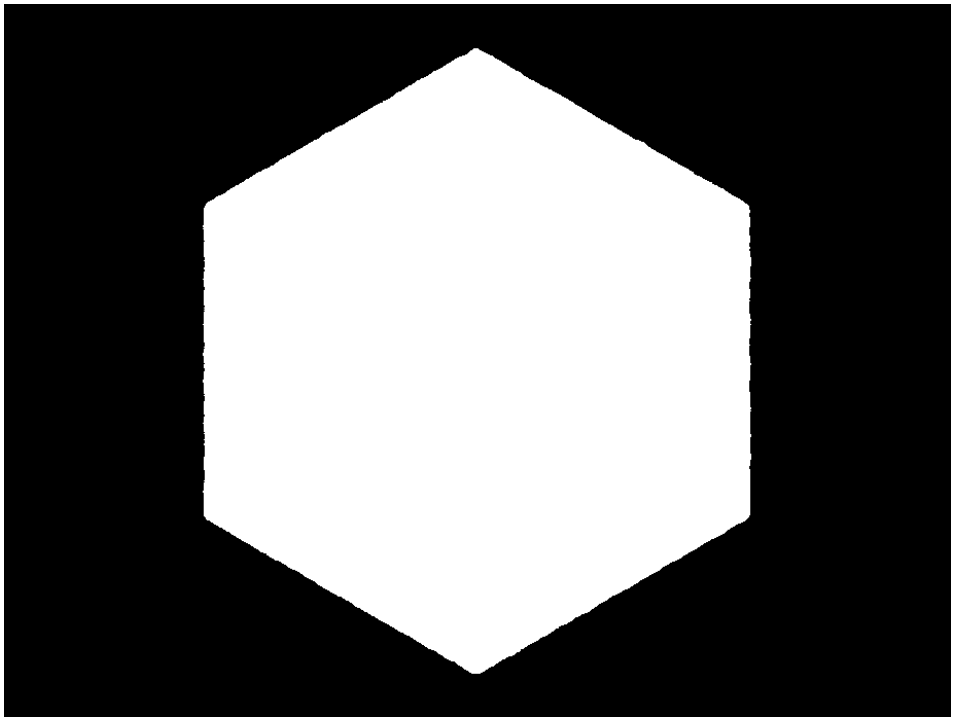
```
h2 = fspecial('gaussian',[9 9], 1000);
imF = imfilter(imgNoise,h2,'replicate');
```

Następnie w celu znalezienia progu binaryzacji pokazano histogram obrazu po zastosowaniu filtracji:



Do binaryzacji przyjęto próg 215.

Obraz po binaryzacji:



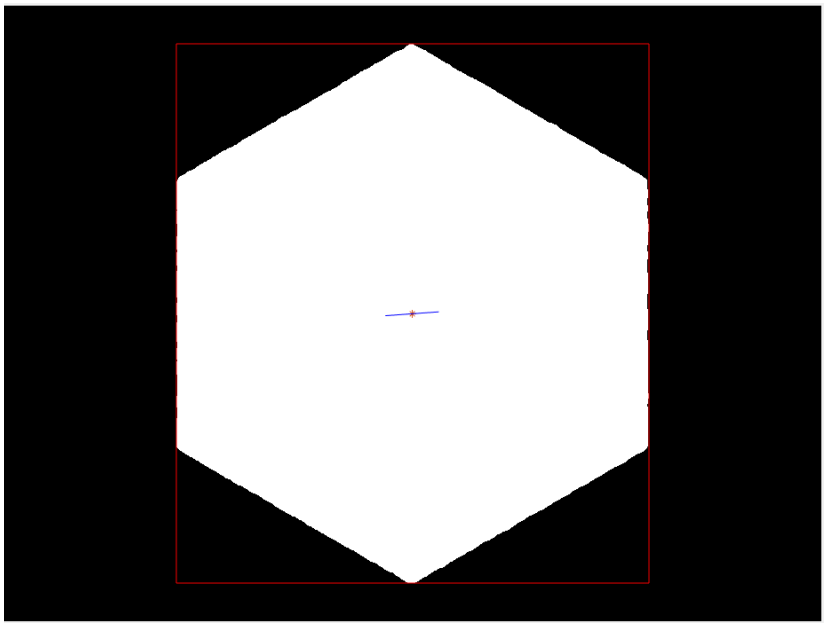
Następnie dzięki komendzie:

```
features = regionprops(L,'All');
```

otrzymano strukturę wypełnioną danymi o obrazie. Odczytano z niej wartości parametrów:

Field	Value
Area	242767
Centroid	[459.8210,347.8908]
BoundingBox	[194.5000,43.5000,532...
Subarrayldx	*1x2 cell*
MajorAxisLength	558.9649
MinorAxisLength	557.2078
Eccentricity	0.0792
Orientation	-4.1766
ConvexHull	*302x2 double*
ConvexImage	*608x532 logical*
ConvexArea	244365
Circularity	0.9112
Image	*608x532 logical*
FilledImage	*608x532 logical*
FilledArea	242767
EulerNumber	1
Extrema	*8x2 double*
EquivDiameter	555.9681
Solidity	0.9935
Extent	0.7505
PixelIdxList	*242767x1 double*
PixelList	*242767x2 double*
Perimeter	1.8298e+03
PerimeterOld	1.9307e+03
MaxFeretDiameter	610.0033
MaxFeretAngle	150.3250
MaxFeretCoordin...	[724.5000,196.5000;19...
MinFeretDiameter	531.9826
MinFeretAngle	-60.0285
MinFeretCoordina...	[622.5000,134.5000;35...

Następnie przedstawiono je na przefiltrowanym obrazie:

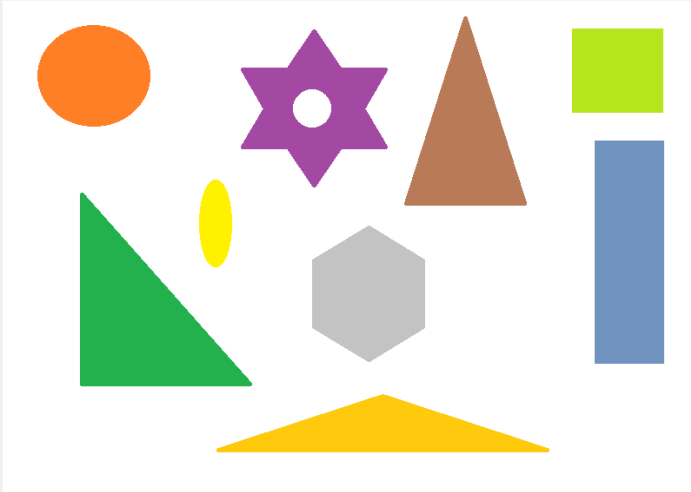


Wnioski: Dla tak przyjętych parametrów szumu filtracja dała zadowalające efekty. Odczytany kształt nie jest idealny ale w dużej mierze przypomina oryginał. Widoczne deformacje pojawiają się wyłącznie na krawędziach. Nastąpiło wyraźne ścięcie kątów badanego sześciokąta. Wykorzystanie zastosowanego filtra może przynieść zadowalające wyniki.

Cechy zaznaczone na obrazie wydają się być poprawne, jednak wyraźnie widać, że niebieska oś przedstawiająca parametr Orientation jest wyraźnie przekrzywiona. Z dużym prawdopodobieństwem można stwierdzić, że jest to skutek zniekształceń obiektu względem oryginał.

Zad 1.3

Obraz użyty w zadaniu:

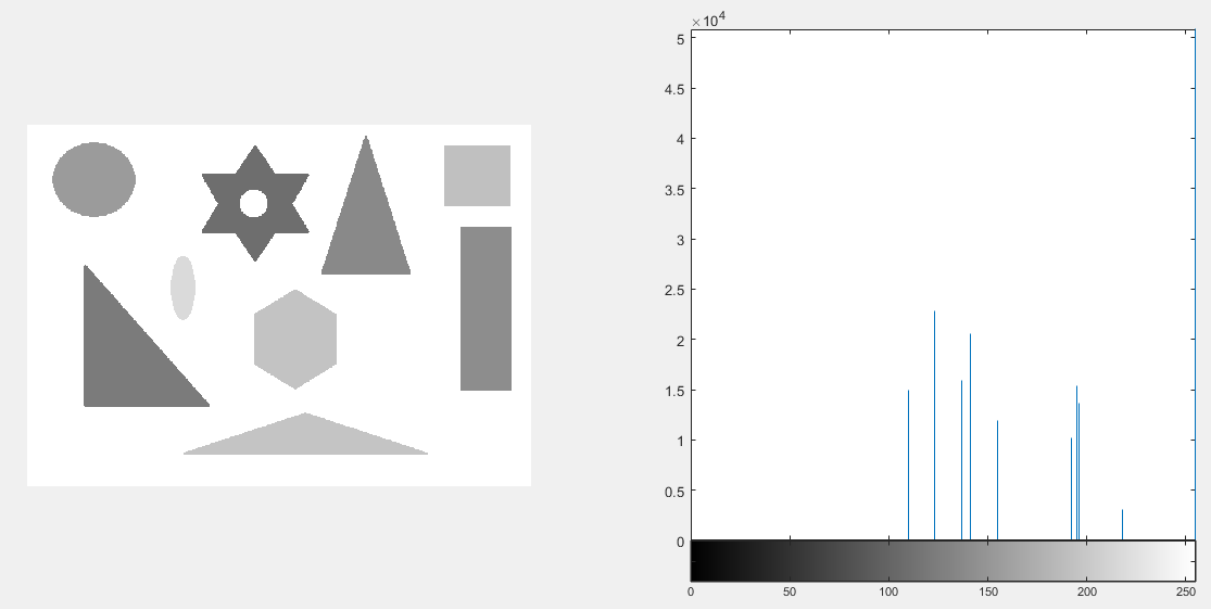


Całość kodu:

```
clc
close all
clear all
%% obraz
imfinfo('kszt.png')
im = imread('kszt.png');
figure(10)
imshow(im)
img = im2gray(im);
figure(1)
subplot(1,2,1)
imshow(img)
subplot(1,2,2)
imhist(img)

%% dodanie szumu
imgNoise = imnoise(img, 'gaussian', 0,0.01);
figure(2)
subplot(1,2,1)
imshow(imgNoise)
subplot(1,2,2)
imhist(imgNoise)
I = medfilt2(imgNoise);
figure(3)
imshow(I)
%% Filtr
h2 = fspecial('gaussian',[3 3], 1000);
imFilt = imfilter(img,h2,'replicate');
figure(4)
subplot(1,2,1)
imshow(imFilt)
subplot(1,2,2)
imhist(imFilt)
figure(5)
bw1 = imFilt < 195+0.01;
subplot(1,2,1)
```

```
imshow(bw1)
bw2 = imFilt > 195 - 0.01;
subplot(1,2,2)
imshow(bw2)

IM = and(bw1,bw2);
figure(6)
imshow(IM)
%% segmentacja
[L, num] = bwlabel(IM,4);
imshow(IM)
hold on
%% cechy
features = regionprops(L,'All')
```

Obraz w skali szarości:

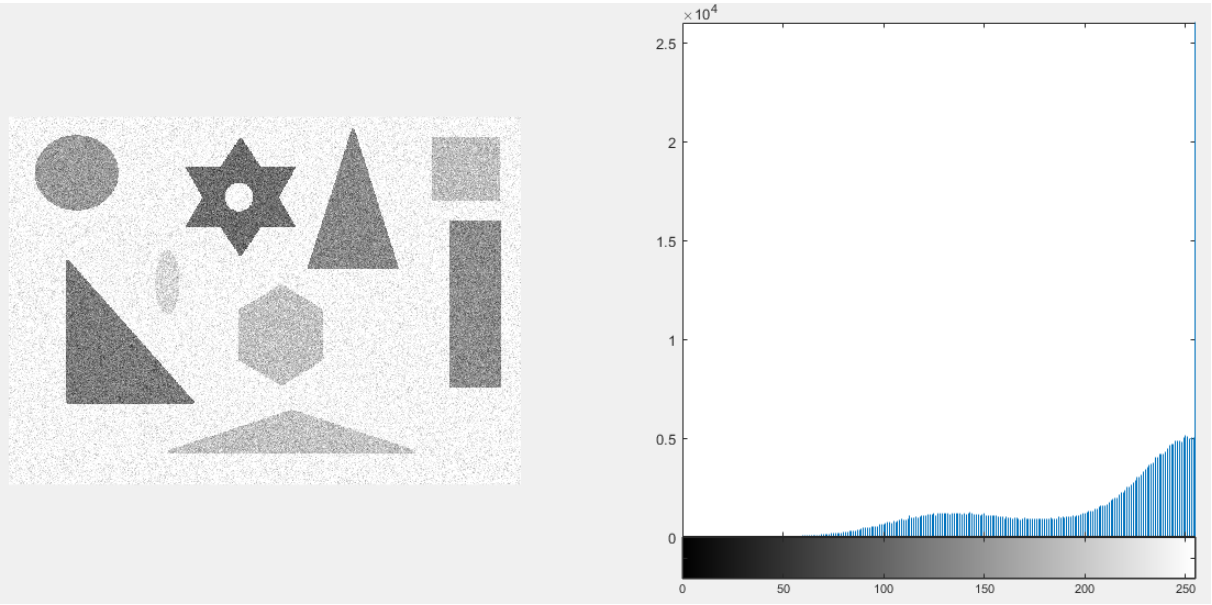


Następnie obraz poddano zaszumieniu z użyciem szumu Gaussowskiego:

```
imgNoise = imnoise(img, 'gaussian', 0,0.01);
```

Otrzymano :

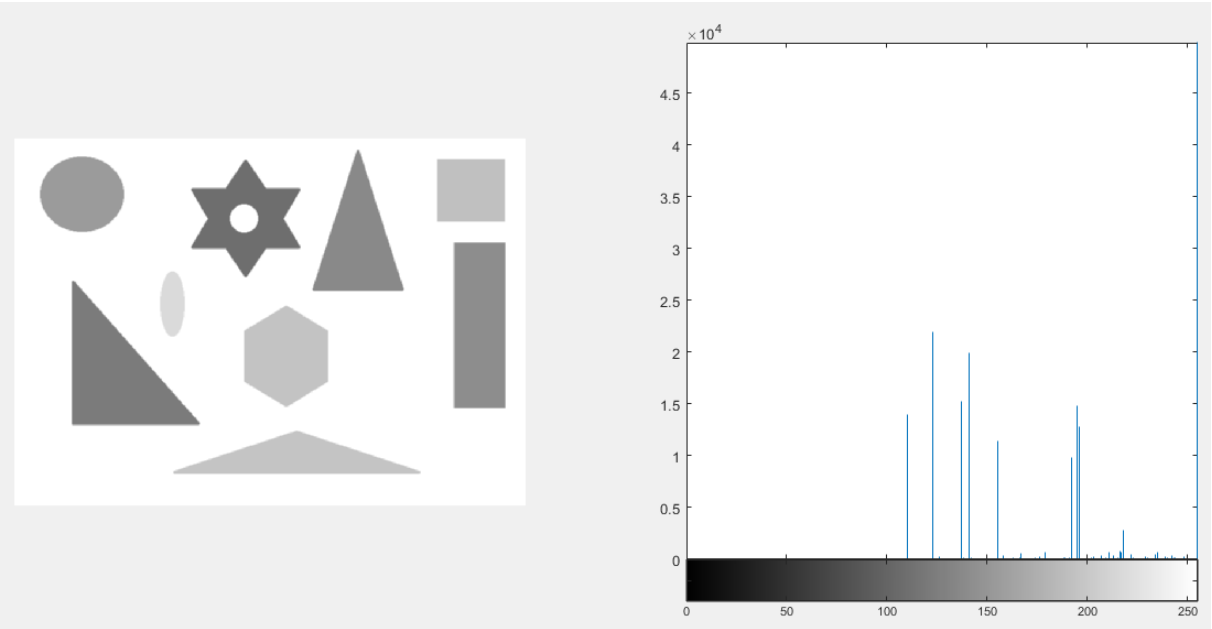
Zaszumiony obraz wraz z jego histogramem:



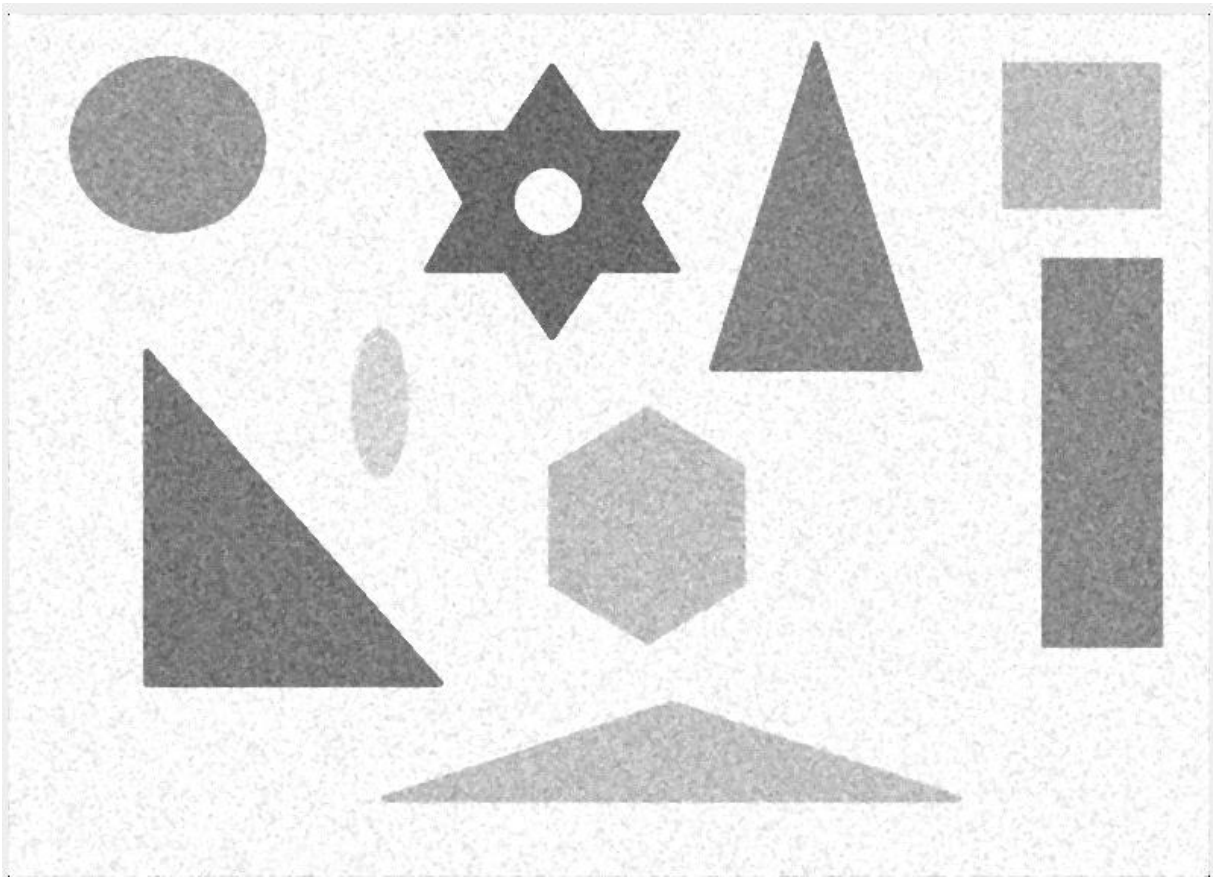
Stosując filtrację:

```
h2 = fspecial('gaussian',[3 3], 1000);
imFilt = imfilter(img,h2,'replicate');
```

Otrzymano dobrze przefiltrowany obraz:



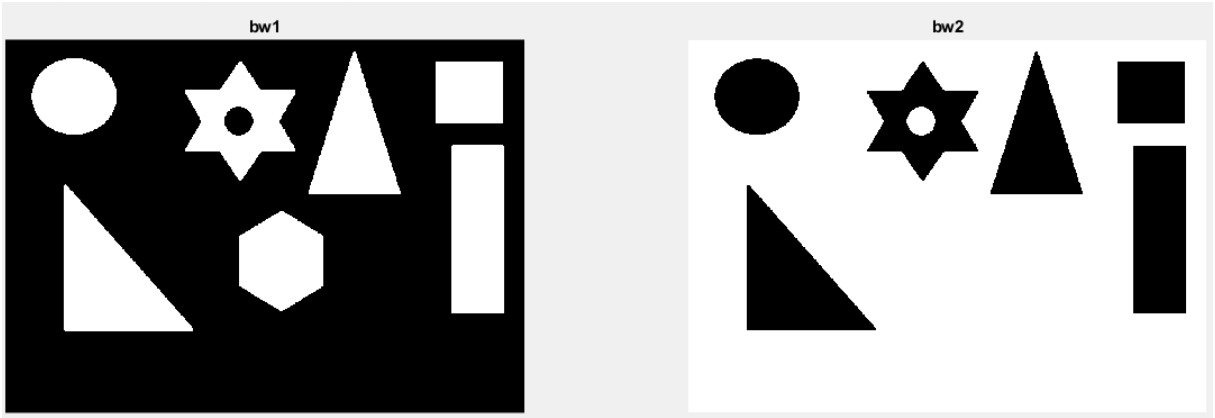
Dla porównania przeprowadzono również filtrację filtrem medianowym z użyciem komendy medfilt(), jednak jej wynik jest wyraźnie słabszy:



Mając odszumiony obraz przeprowadzono dwie binaryzacje w celu wydobycia konkretnego obiektu (sześciokąta). Zastosowano poniższe komendy:

```
bw1 = imFilt < 195+0.01;
bw2 = imFilt > 195 - 0.01;
```

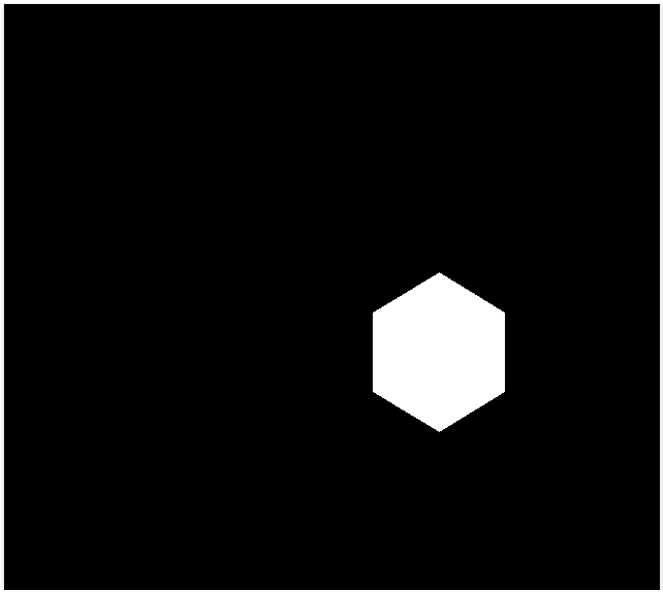
Dzięki nim uzyskano wyniki:



Kolejno łącząc je poleceniem and:

```
IM = and(bw1,bw2);
```

Otrzymano żądany kształt:



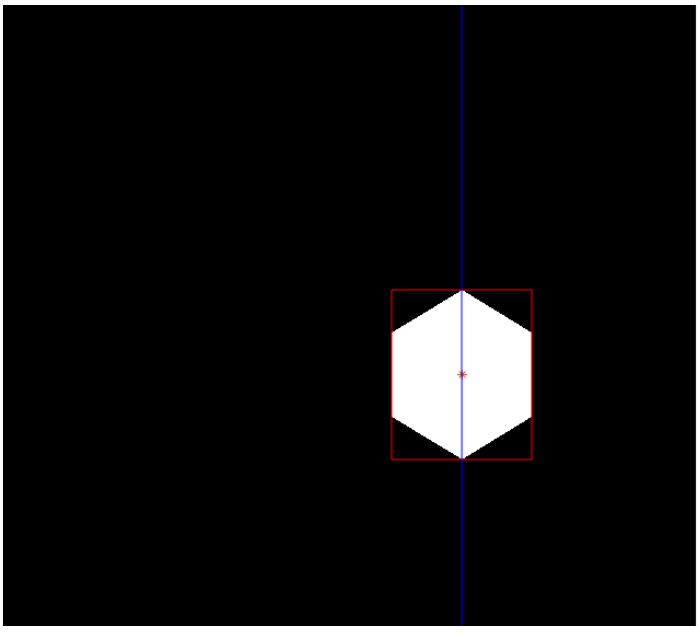
Cechy kształtu:

features

1x1 struct with 30 fields

Field	Value
Area	14814
Centroid	[421.5628,338.0305]
BoundingBox	[357.5000,260.5000,12...
Subarrayldx	*1x2 cell*
MajorAxisLength	141.0232
MinorAxisLength	134.7797
Eccentricity	0.2943
Orientation	89.9808

Sześciokąt po naniesieniu na niego odpowiednich cech.



Wnioski: filtr perfekcyjnie poradził sobie z naniesionym szumem Gaussa. Dzięki odpowiedniej binaryzacji udało się wydobyć konkretny kształt. Można stwierdzić poprawność naniesionych cech.

Zad 1.4

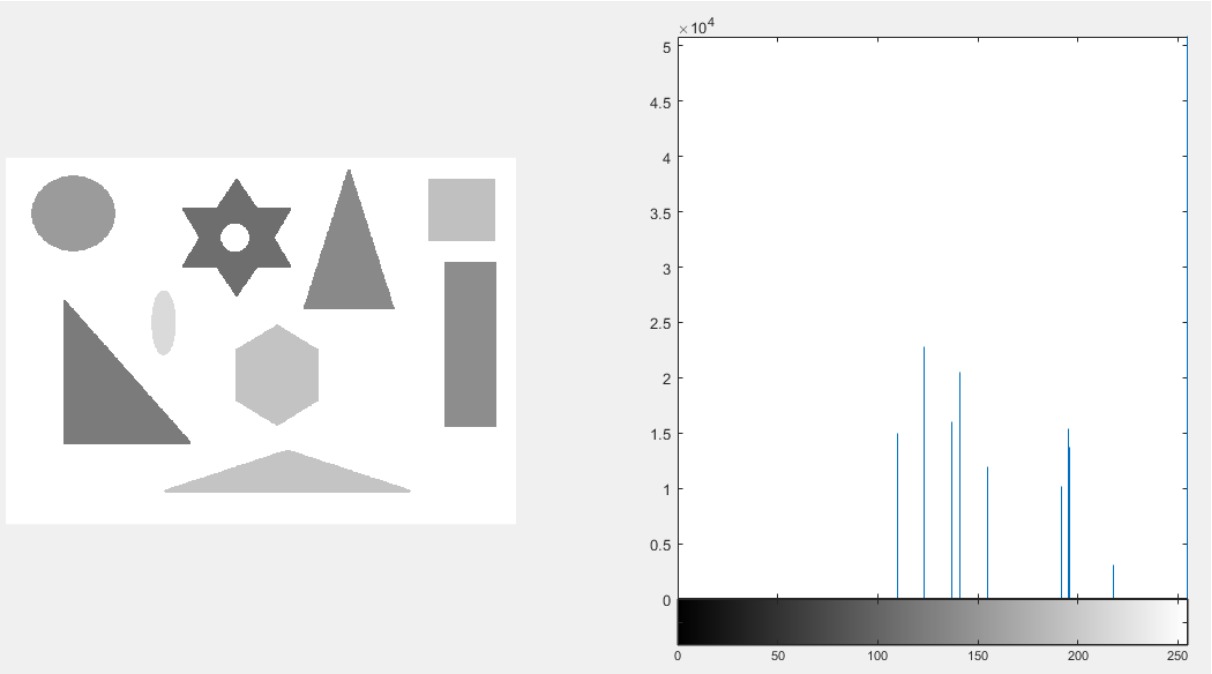
Obrazy użyte w zadaniu:

Całość kodu:

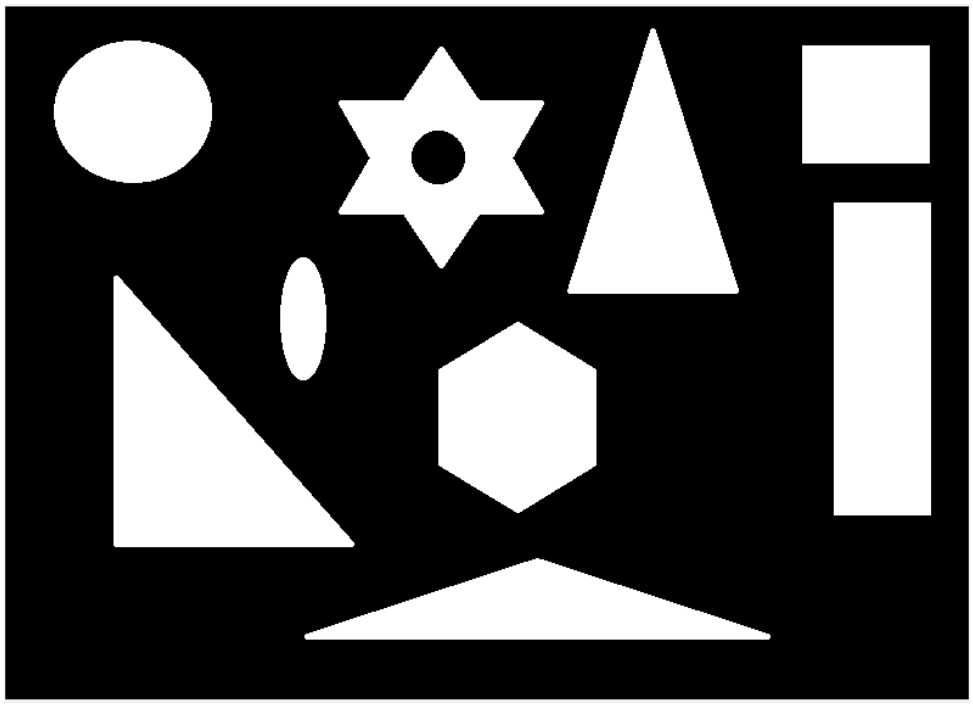
```
clc
close all
clear all
%% obraz
imfinfo('kszt.png')
im = imread('kszt.png');
%im = imrotate(im,90);
figure(10)
subplot(1,2,1)
imshow(im)
title('Obraz orginalny')
img = im2gray(im);
subplot(1,2,2)
imshow(img)
title('Obraz w sklai szarości')
figure(1)
subplot(1,2,1)
imshow(img)
subplot(1,2,2)
imhist(img)
%% binaryzacja
imbw = img < 218 + 0.01;
figure(2)
imshow(imbw)
%% segmentacja
[L, num] = bwlabel(imbw,4);
imshow(imbw)
hold on
%% cechy
feat = regionprops(L,'All')
%%
Areas = cat(1,feat.Area);
Perimeters = cat(1,feat.Perimeter);
MaxDiam = cat(1,feat.MajorAxisLength);
MinDiam = cat(1,feat.MinorAxisLength);
ConvexAreas = cat(1,feat.ConvexArea);
```

```
for k =1:num
feat(k).FormFactor= 4*pi*feat(k).Area/(feat(k).Perimeter^2);
feat(k).Roundness = 4*feat(k).Area/(pi*feat(k).MajorAxisLength.^2);
feat(k).AspectRatio = feat(k).MajorAxisLength / feat(k).MinorAxisLength;
feat(k).Solidity = feat(k).Area / feat(k).ConvexArea;
feat(k).Compactness = sqrt(4*feat(k).Area/pi)/feat(k).MajorAxisLength;
end
%%
figure(4)
imshow(imbw)
hold on
for i = 1:num
    rectangle('Position', feat(i).BoundingBox, 'EdgeColor', 'red');
    centroids = cat(1, feat.Centroid);
    plot(centroids(:,1), centroids(:,2), 'b*')
end

hold off
[x,y] = size(imbw);
X = 1:1:x;
Y = 1:1:y;
figure(5)
mesh(Y,X,L);
%% Sortowanie
%Macierz obiektów
L1 = L;
for i = 1:num
    if~(feat(i).Solidity >= 0.9 && feat(i).Eccentricity >= 0.8 &&
feat(i).Eccentricity<=1)

    for j=1:x
        for k=1:y
            if(L1(j,k) == i)
                L1(j,k)=0;
%objects = cat(2,objects,obj);

            end
        end

    end
    end
end
figure(7)
imshow(L1)
```

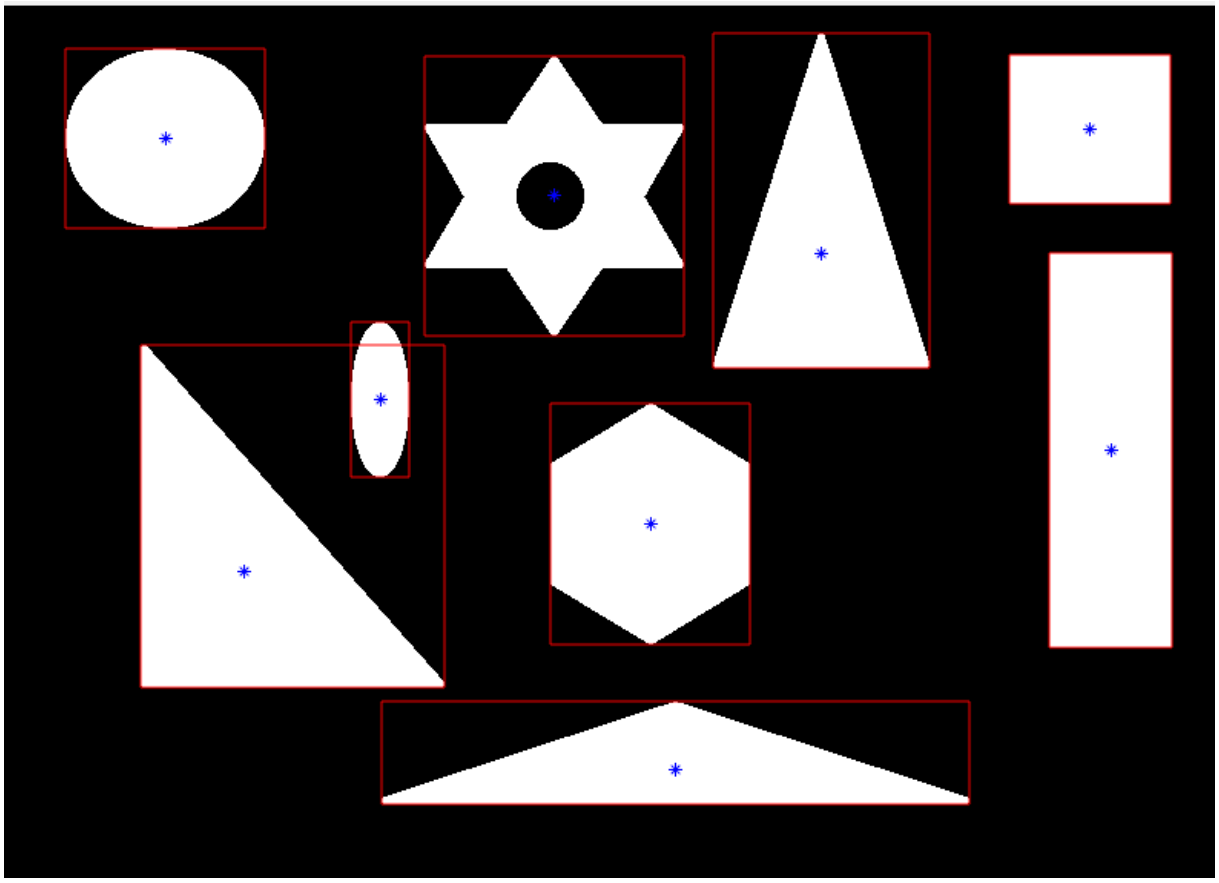
Obraz w odcieniach szarości razem z histogramem:



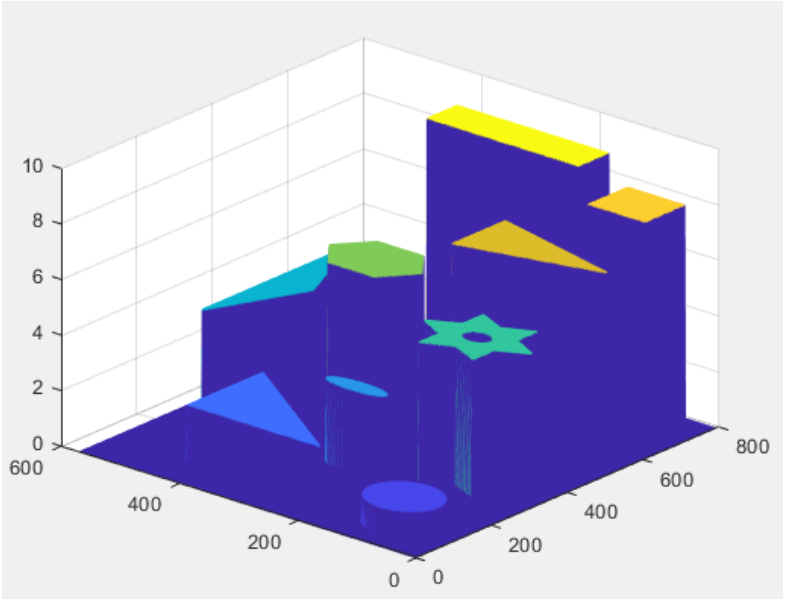
Obraz po binaryzacji:



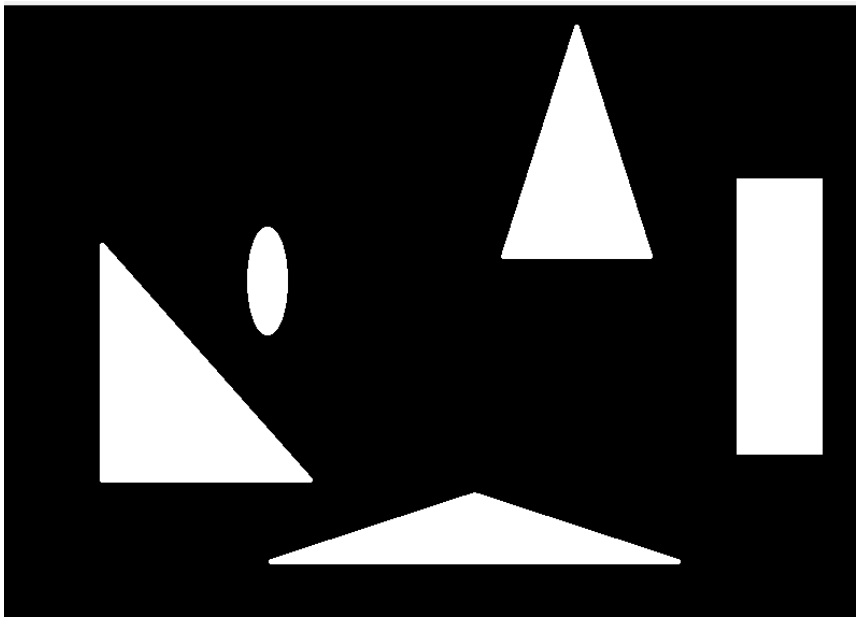
Zaznaczone środki ciężkości oraz Bounding boxy:



Określenie etykiety każdego z kształtów poleceniem mesh()



Wynik pierwszej segregacji:

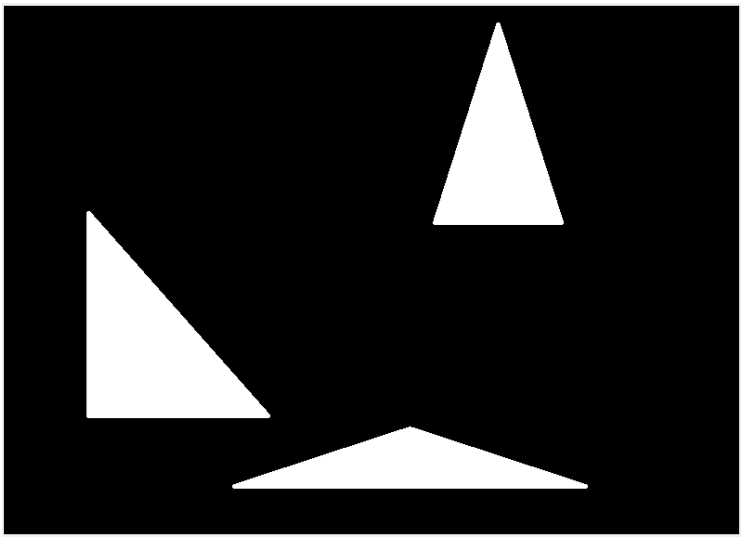


Celem była filtracja samych trójkątów bazując na obserwacji ich podobnych wartości dla cech: Solidity oraz Eccentricity. Nie udało się jednak poprawnie wydobyć wszystkich trzech trójkątów. Co więcej do wyniku wkradły się również prostokąt i elipsa.

Zmieniono instrukcję warunkową na następującą:

```
if~(feat(i).Solidity >= 0.9 && feat(i).Eccentricity
>= 0.8 && feat(i).Eccentricity<=1 && feat(i).Extent
>= 0.5 && feat(i).Extent <= 0.7)
```

uwzględniając parametr Extent, który dla wszystkich trójkątów zawierał się w przedziale 0.5 – 0.6. Zastosowano jednak wartość 0.7 ograniczającą przedział z góry.
Pozwoliło to uzyskać pożądany wynik – samych trójkątów, już bez dodatkowych elementów:

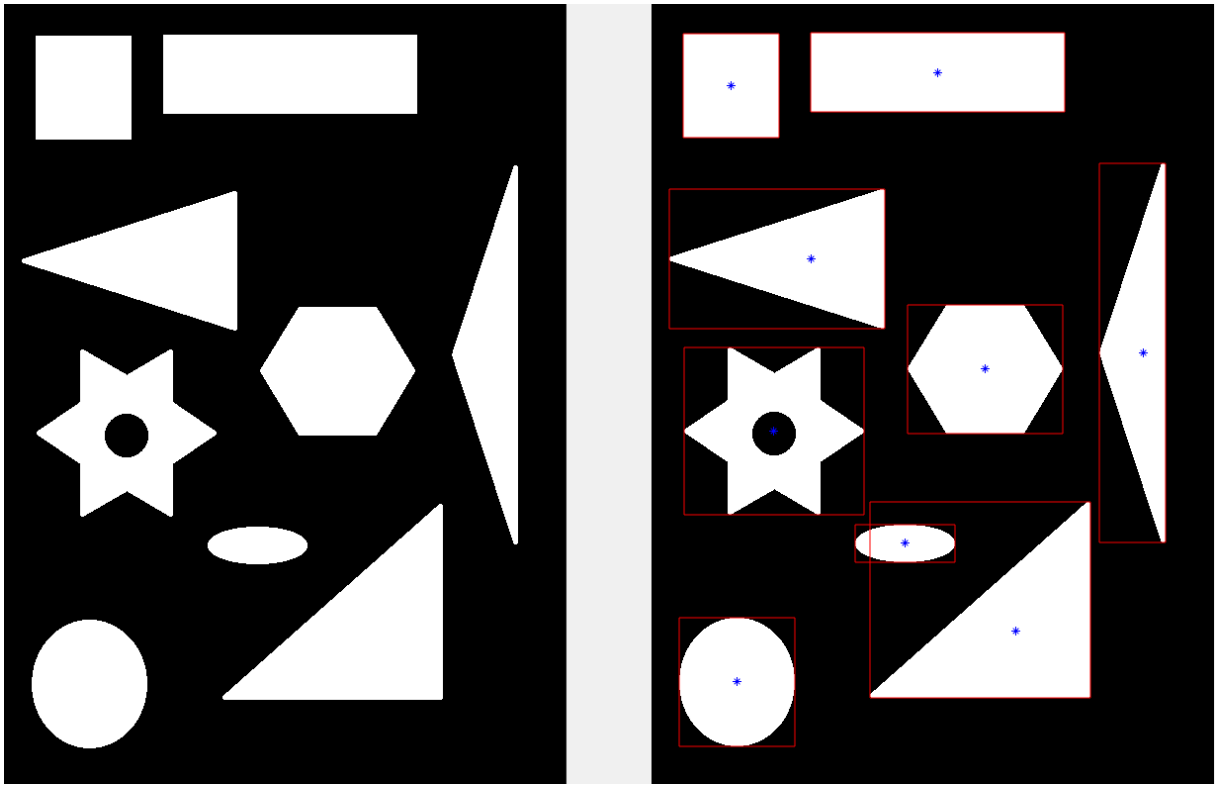


Dodano następującą linijkę zaraz po komendzie wczytującej obraz do Workspace:

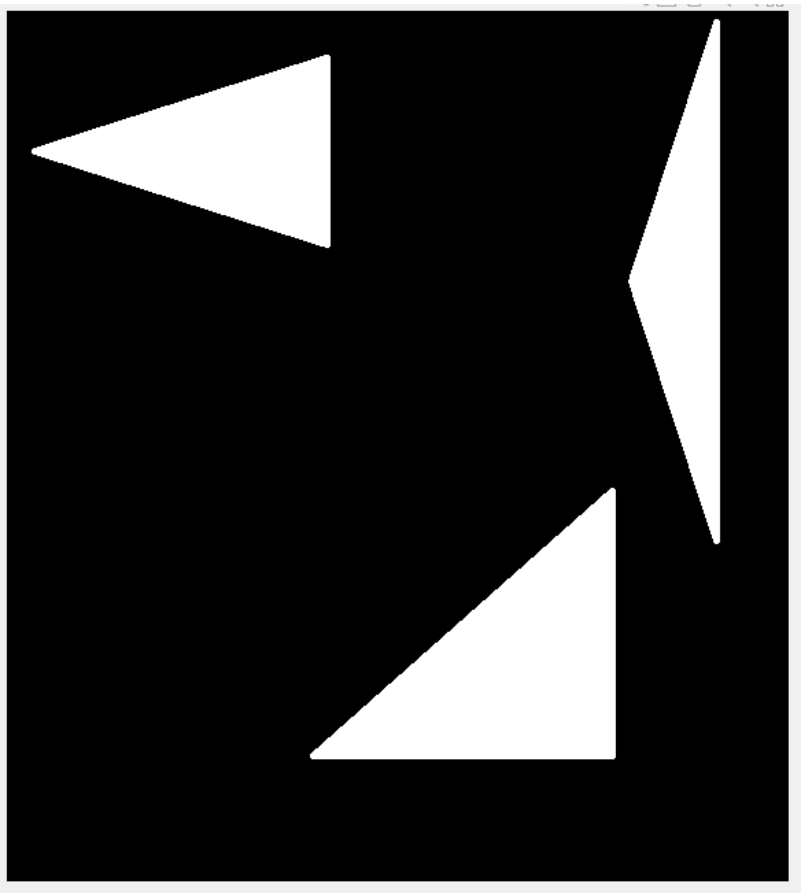
```
im = imrotate(im,90);
```

Powoduje ona obrót obrazu o 90 stopni.

Część wyników powstałych z kodu po zmianie:

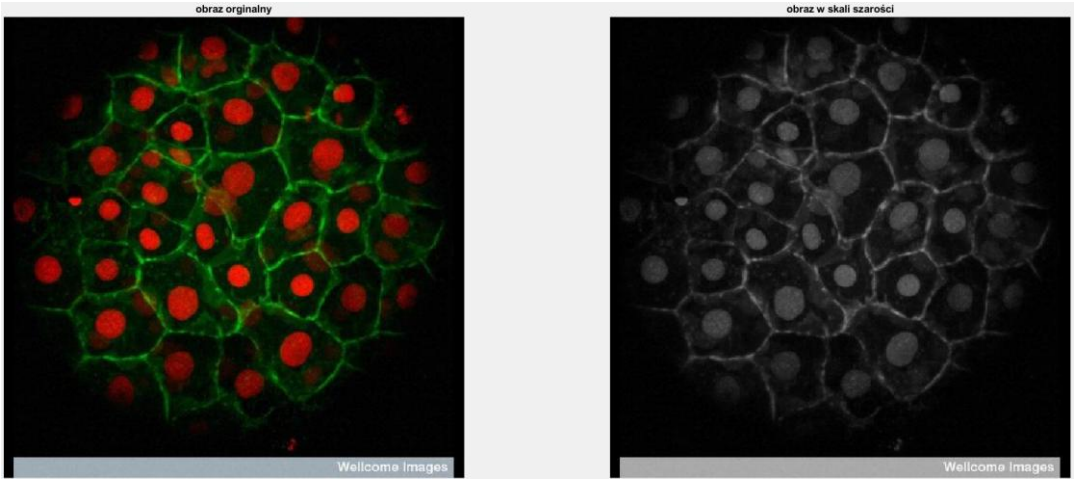


Wynik filtracji również okazał się poprawny:



Zad 1.5

Obraz użyty w zadaniu:

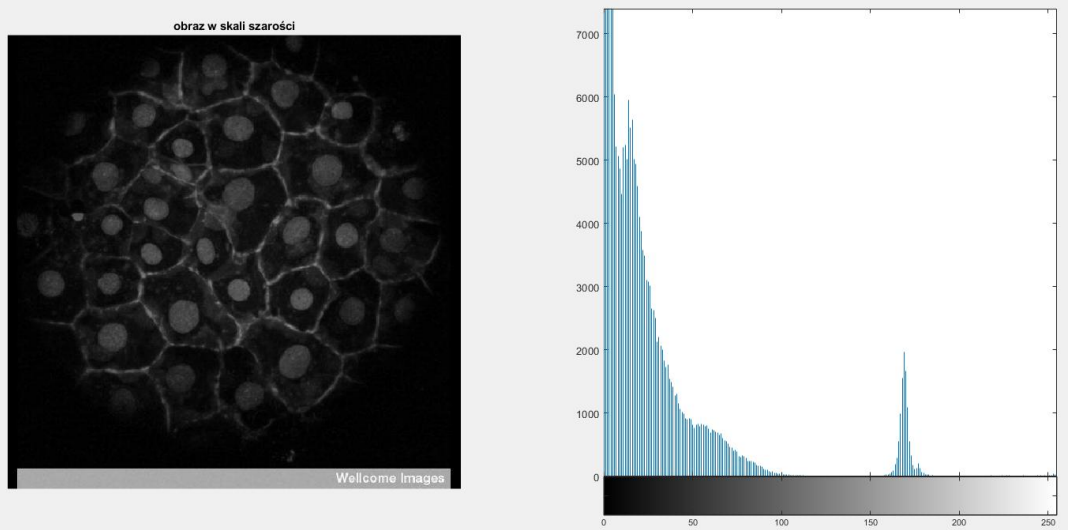


Całość kodu:

```
clc
close all
clear all

imfinfo('cell.png');
im = imread('cell4.jpg');
figure(1)
subplot(1,2,1)
imshow(im)
title('obraz orginalny')
img = rgb2gray(im);
subplot(1,2,2)
imshow(img)
title('obraz w skali szarości')

figure(2)
subplot(1,2,1)
imshow(img)
title('obraz w skali szarości')
subplot(1,2,2)
imhist(img)

%% binaryzacjia
bw1 = img > 60;
bw2 = img < 90;
figure(3)
subplot(2,2,1)
imshow(bw1)
subplot(2,2,2)
imshow(bw2)
bw = and(bw1, bw2);
subplot(2,2,3)
imshow(bw)
se1 = strel('disk',4);
se2 = strel('disk',5);
imOpen = imopen(bw,se1);
ime = imdilate(imOpen,se2);
```

```
subplot(2,2,4)
imshow(ime)

%% segmentacja
[L, num] = bwlabel(ime,4);
%% cechy
feat = regionprops(L,'All');
%% estymaty
I = ime;

Array = [feat.Perimeter];
Min = min(Array);
Max = max(Array);
Mean = mean(Array);
Std = std(Array);
figure(4)
hist(Array,20);
title('Histogram for Perimeter');

idx = find(Array < (Mean - Std));
L2 = ismember(L,idx);
L2 = I.*L2;
figure(5)
imshow(L2)
title('Objects below mean-std: Perimeter');

idx = find(Array > (Mean + Std));
L2 = ismember(L,idx);
L2 = I.*L2;
figure(6)
imshow(L2)
title('Objects above mean+std: Perimeter');

idx = find((Array < (Mean + Std)) & (Array > (Mean - Std)));
L2 = ismember(L,idx);
L2 = I.*L2;
figure(7)
imshow(L2)
title('Objects between mean-std and mean+std: Perimeter');
```

Pokazano histogram obrazu w skali szarości:



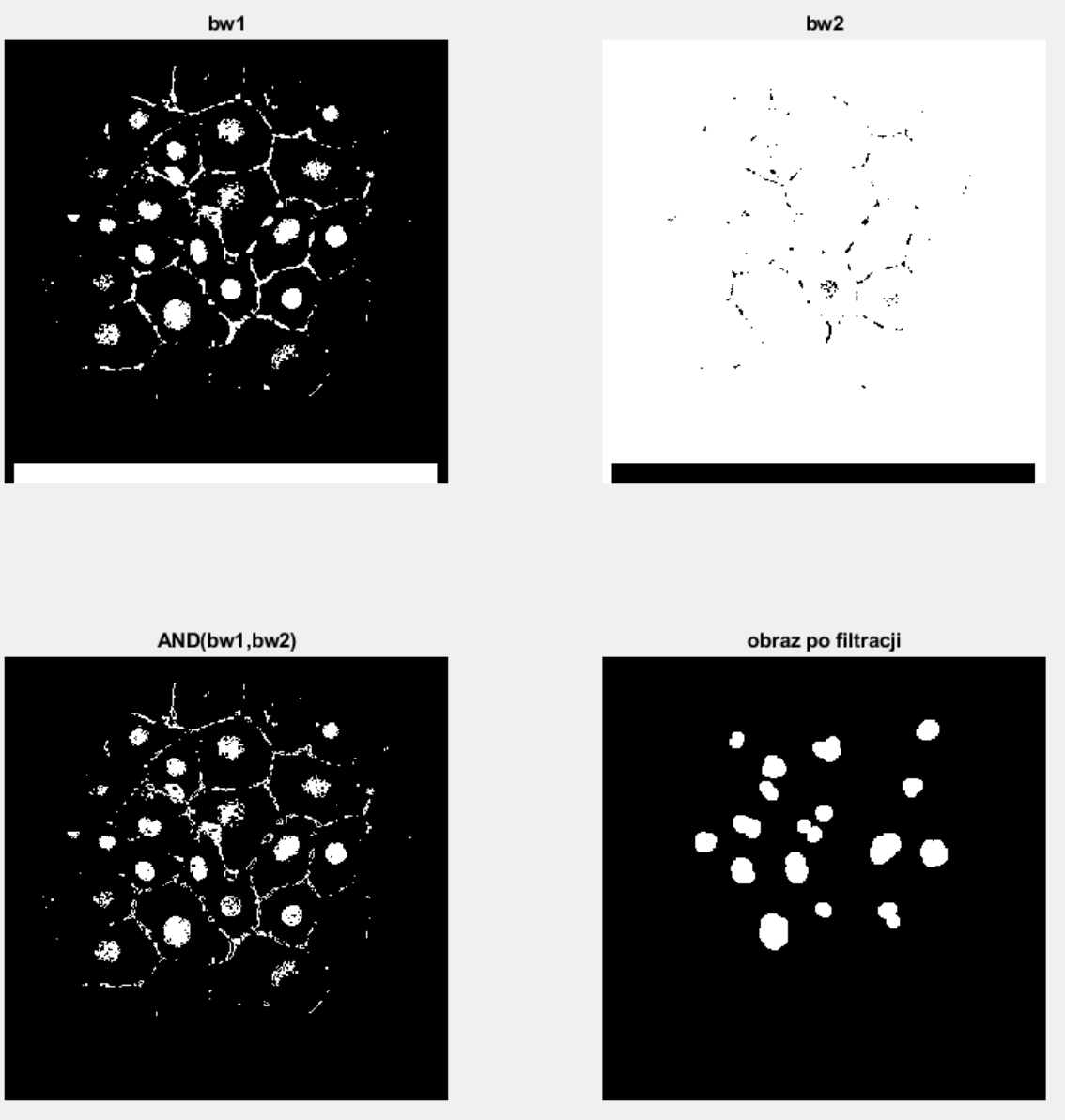
Następnie zastosowano binaryzację oraz filtrację w celu wydobycia zcerwonych elementów obrazu:

Binaryzacja:

```
bw1 = img > 60;
bw2 = img < 90;
bw = and(bw1, bw2);
```

Filtracja:

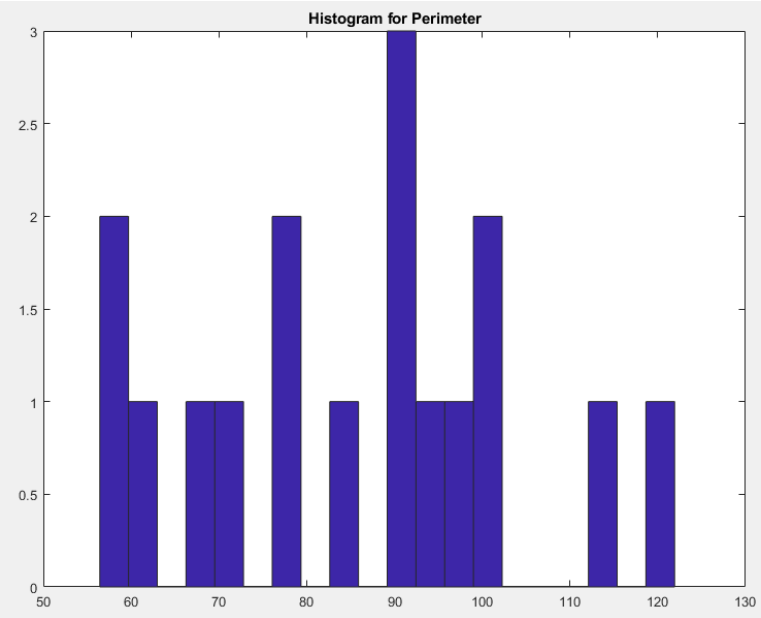
```
se1 = strel('disk',4);
se2 = strel('disk',5);
imOpen = imopen(bw,se1);
ime = imdilate(imOpen,se2);
```



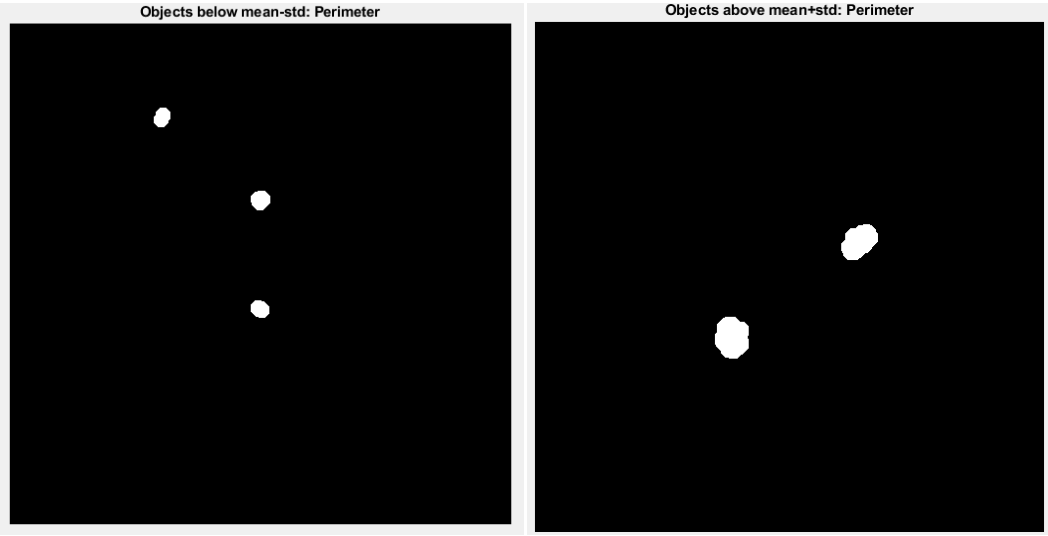
Następnie przeprowadzono segmentację i wydobyto cechy elementów:

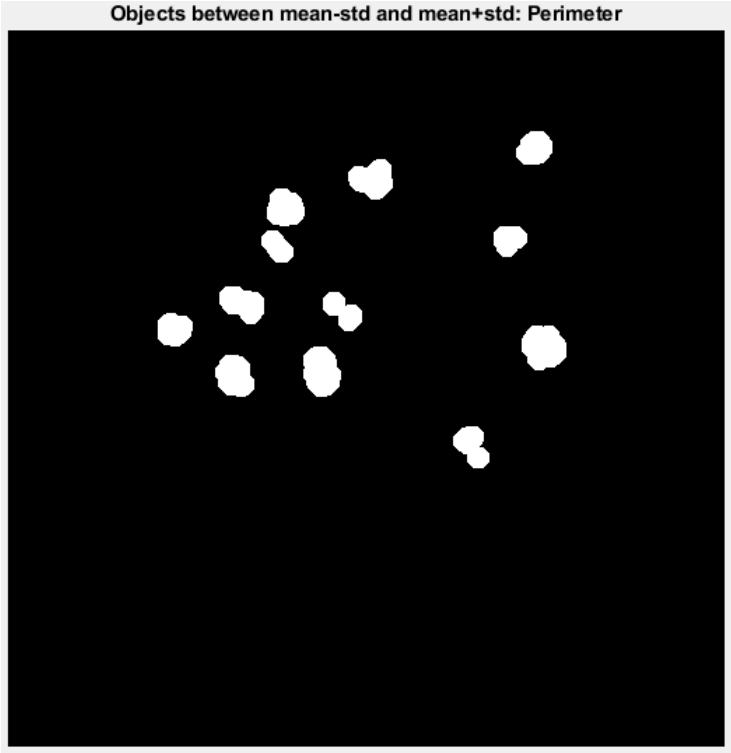
```
%% segmentacja
[L, num] = bwlabel(ime,4);
%% cechy
feat = regionprops(L,'All');
```

Dalej skupiono się na zadanym przez prowadzącego parametrze Permieter. Wykreślono histogram rozkładu elementów w zależności od ich obwodu:



Pokazano obiektu w zależności od ich obwodu:





Wnioski: Okazuje się możliwe wydobycie wielu podobnych elementów z dość skomplikowanego obrazu. Nie jest to wierne odwzorowanie , jednak daje ogólne informacje o rzeczywistych obiektach i ich cechach.