Gliwice, 22.11.2022

Przetwarzanie Obrazów Cyfrowych

CECHY NIEZMIENNICZE I KLASYFIKACJA OBIEKTÓW

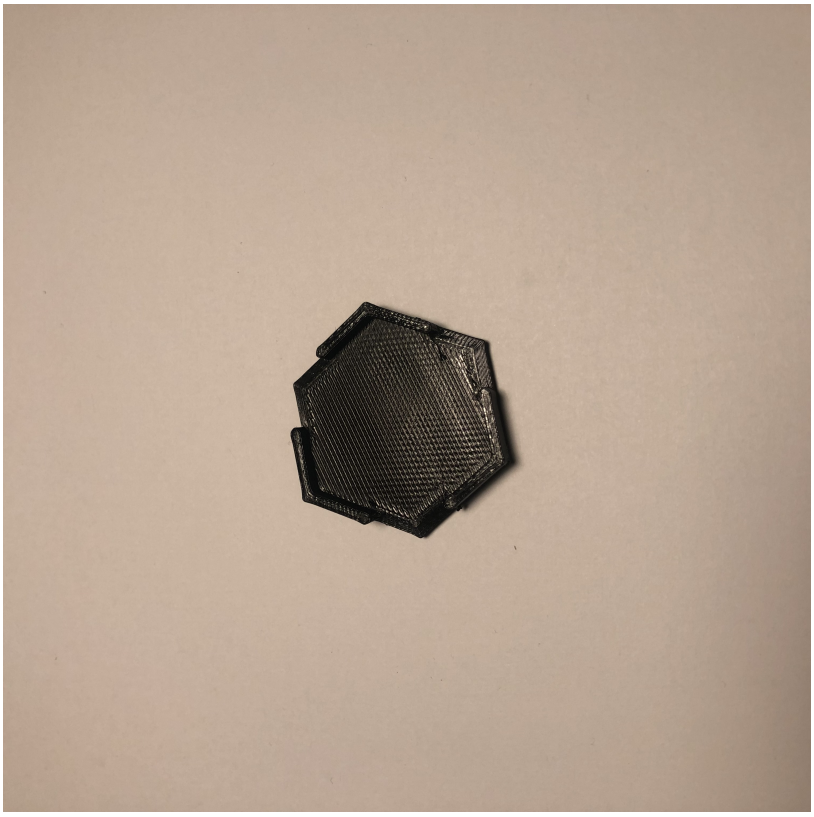
Politechnika Śląska

Jakub Zeifert

1 Przygotowanie obrazów testowych

(Zaliczone podczas zajęć) Do laboratorium zostały przygotwane obrazy wykonane ręcznie. Jako obiekt do zdjęć zostały wykorzystane druki 3D wykonane z czarnego PLA.

Rysunek 1: klucz



Rysunek 2: hex

Rysunek 3: nakrętka

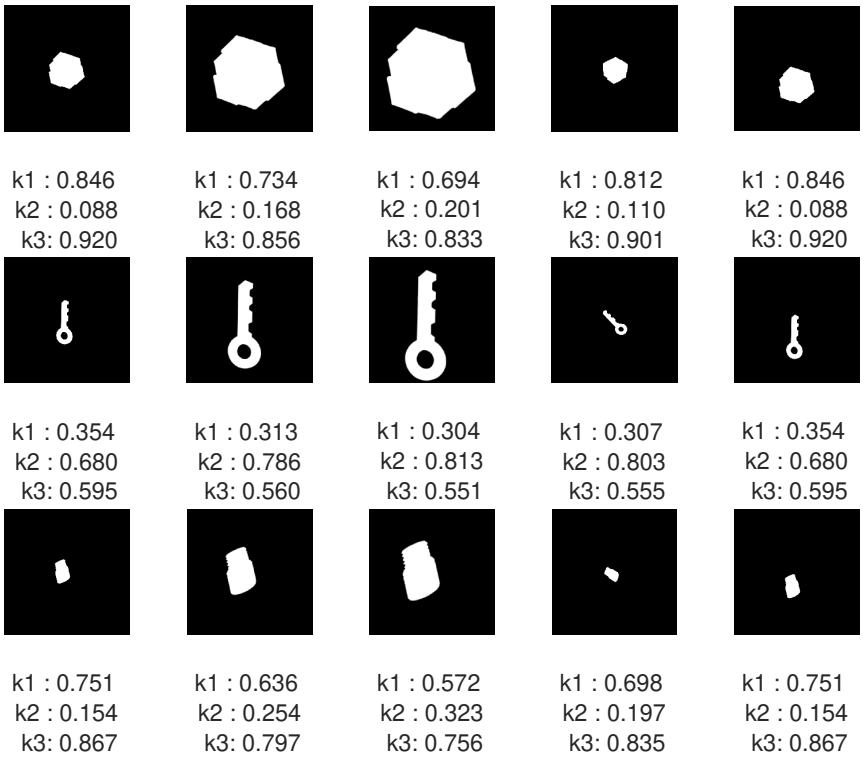
2 Implementacja wskaźników (min.3) niezmienniczych

(Zaliczone podczas zajęć) Zaimplementowane zostało 5 wskaźników, a wykorzystano 3 wybrane:

```
function [k1 , k2] = count_factors(img, type)
    stats = regionprops('table', img, "Area", "Perimeter");
    area = double(stats.Area(255));
    perimeter = double(stats.Perimeter(255));
    switch type
        case "compact"
            k1 = (area*4*pi)/(perimeter*perimeter);
        case "circular"
            k1 = 2 * sqrt(area/pi);
            k2 = perimeter/pi;

        case "malinowska"
            k1 = perimeter/(2*sqrt(pi * area)) - 1;
        case "mz"
            k1 = (2*sqrt(pi*area))/perimeter;
        otherwise
            disp("ERROR!!! WRONG TYPE INPUT");
    end
end
```

count_factors.m

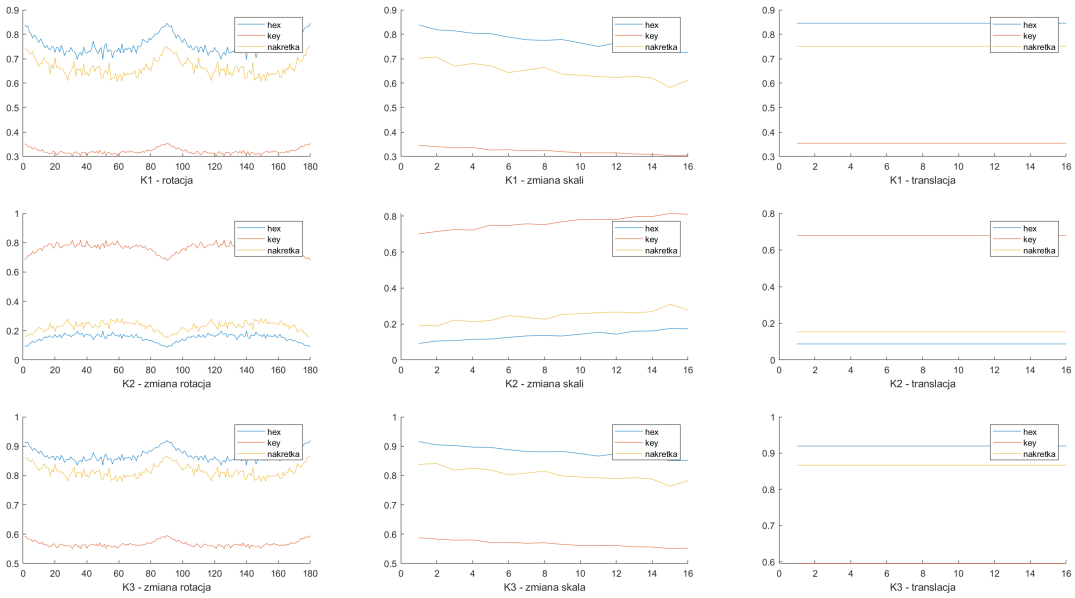
- k1 - Kompaktowy ($K = \frac{4 \cdot \pi \cdot S}{L}$)
- k2 - Malionwskiej ($M = \frac{L}{2 \cdot \sqrt{\pi \cdot S}} - 1$)
- k3 - Mz ($K = \frac{2 \cdot \sqrt{\pi \cdot S}}{L}$)



Rysunek 4

3 Ocena właściwości niezmienniczych wybranych wskaźników

(Zaliczone podczas zajęć)



Rysunek 5

Jak widać na wykresach, niestety żaden z wybranych wskaźników nie okazał się niezmienny. Wynika to z tego, że wszystkie wskazniki polegają na wielkości obiektu, który zmienia się wraz z obrotem jak i również skalowaniem. W przypadku translacji wskaźniki wskazują te samą wartość. Wynika to z faktu, że objekt nie zmienia żadnego parametru, który może go opisać.
Wykorzystany kod:

```
clc
clear
```

```
%%
[file,path]=uigetfile('*.*', 'Select an image');
img_1 = imread([path, file]);

[~,~,DIM] = size(img_1);
if DIM > 1
    img_1 = rgb2gray(img_1);
end


[file,path]=uigetfile('*.*', 'Select an image');
img_2 = imread([path, file]);
[~,~,DIM] = size(img_2);
if DIM > 1
    img_2 = rgb2gray(img_2);
end

[file,path]=uigetfile('*.*', 'Select an image');
img_3 = imread([path, file]);
[~,~,DIM] = size(img_3);
if DIM > 1
    img_3 = rgb2gray(img_3);
end

% zdj_1 = "circle";
% zdj_2 = "triangle";
% zdj_3 = "rectangle";
zdj_1 = "hex";
zdj_2 = "key";
zdj_3 = "nakretka";


figure(1);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","compact");
subplot(3,3,1);
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K1 - rotacja");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,2);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","compact");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K1 - zmiana skali");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,3);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","compact");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K1 - translacja");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,4);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","malinowska");
hold on;
plot(X,k1);
plot(X,k2);
```

```
plot(X,k3);
xlabel("K2 - zmiana rotacja");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,5);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","malinowska");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K2 - zmiana skali");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,6);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","malinowska");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K2 - translacja");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,7);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","mz");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K3 - zmiana rotacja");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,8);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","mz");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K3 - zmiana skala");
legend(zdj_1,zdj_2,zdj_3);

subplot(3,3,9);
[k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","mz");
hold on;
plot(X,k1);
plot(X,k2);
plot(X,k3);
xlabel("K3 - translacja");
legend(zdj_1,zdj_2,zdj_3);


exportgraphics(figure(1),"zad3.eps");
%% Functions



function [factor_1,factor_2,factor_3,mode_iterator] = plotter(img_1,img_2,img_3,mode,
    factor)
    factor_1 = [];
    factor_2 = [];
    factor_3 = [];
    switch mode
        case "rotate"
            mode_iterator = 1:180;
            X = 180;
        case "translate"
```

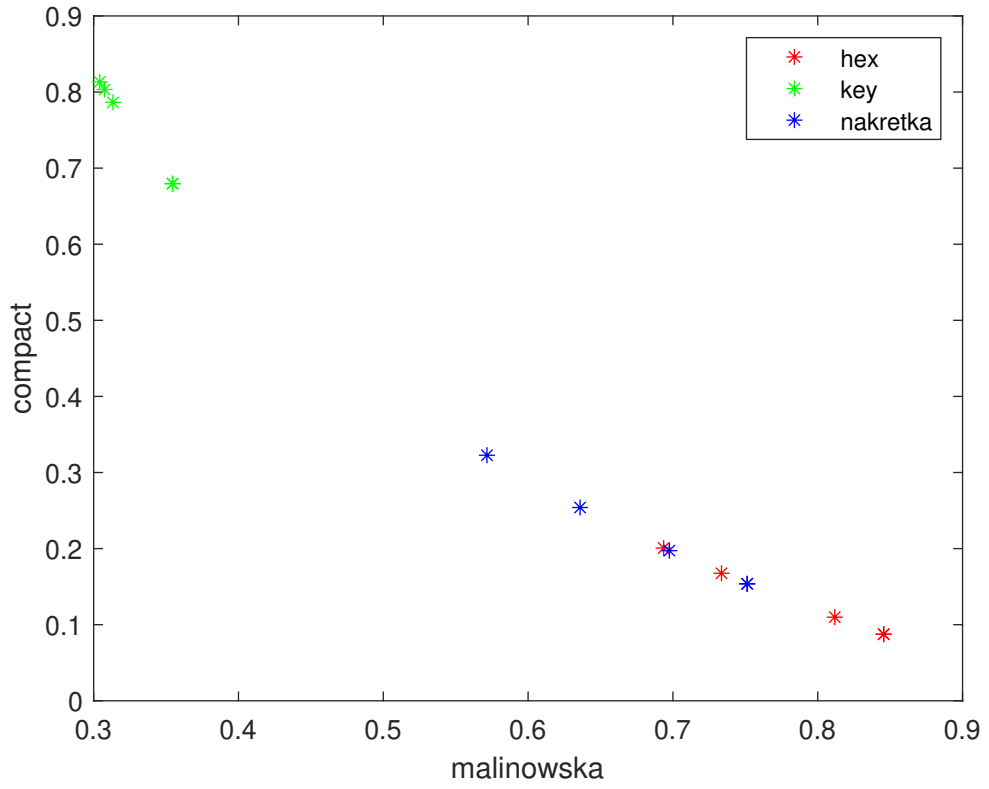
```
132             mode_iterator = 1:16;
133             X = 16;
134         case "scale"
135             mode_iterator = 1:16;
136             X = 16;
137     end
138     for i=1:1:X
139         disp(i);
140         switch mode
141             case "rotate"
142                 changed_img_1 = imrotate(img_1,i);
143                 changed_img_2 = imrotate(img_2,i);
144                 changed_img_3 = imrotate(img_3,i);
145             case "translate"
146                 changed_img_1 = imtranslate(img_1,[i, i],'FillValues',0);
147                 changed_img_2 = imtranslate(img_2,[i, i],'FillValues',0);
148                 changed_img_3 = imtranslate(img_3,[i, i],'FillValues',0);
149             case "scale"
150                 size_factor = 1+i*0.1;
151                 changed_img_1 = crop_img(img_1,size_factor);
152                 changed_img_2 = crop_img(img_2,size_factor);
153                 changed_img_3 = crop_img(img_3,size_factor);
154         end
155         k1 = count_factors(changed_img_1,factor);
156         k2 = count_factors(changed_img_2,factor);
157         k3 = count_factors(changed_img_3,factor);
158         factor_1 = [factor_1 k1];
159         factor_2 = [factor_2 k2];
160         factor_3 = [factor_3 k3];
161     end
162     disp("done");
163 end
```

zad2.m

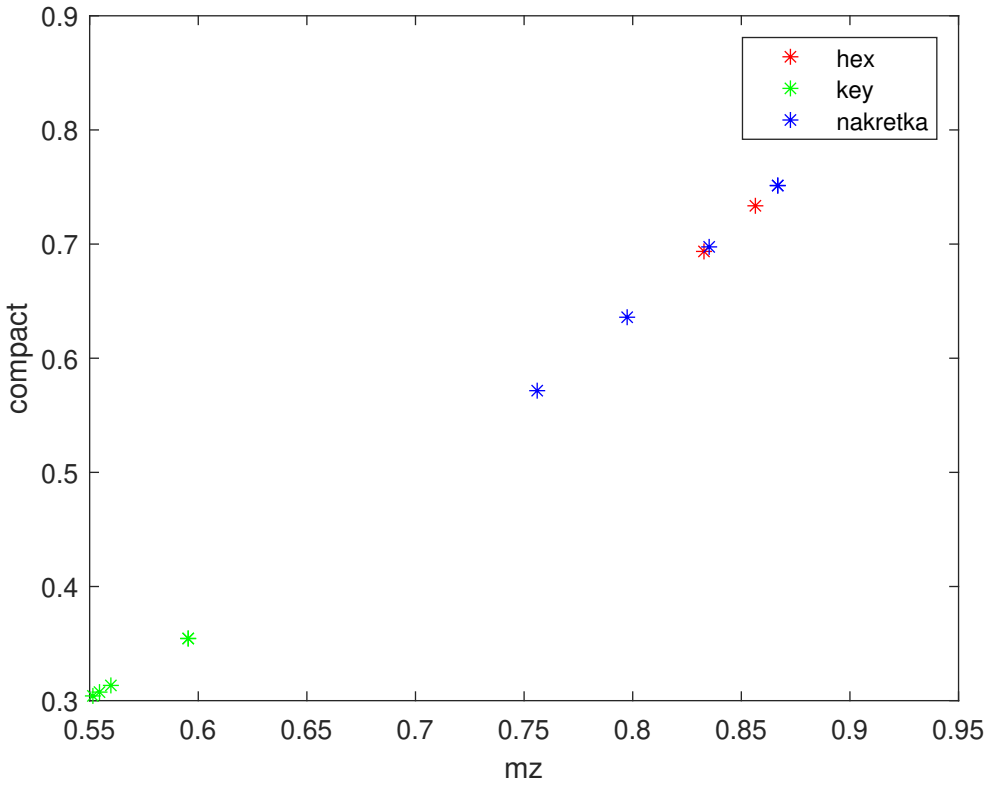
4 Wizualizacja obiektów

(Zaliczone podczas zajęć)

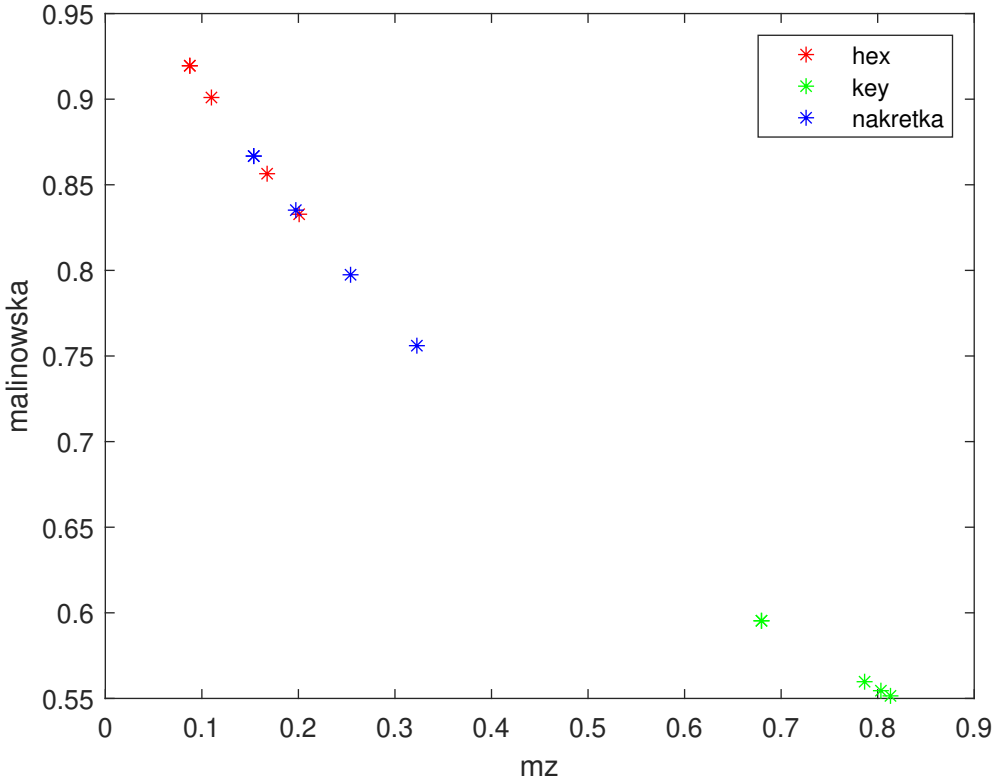
4.1 Na płaszczyźnie cech 2D



Rysunek 6



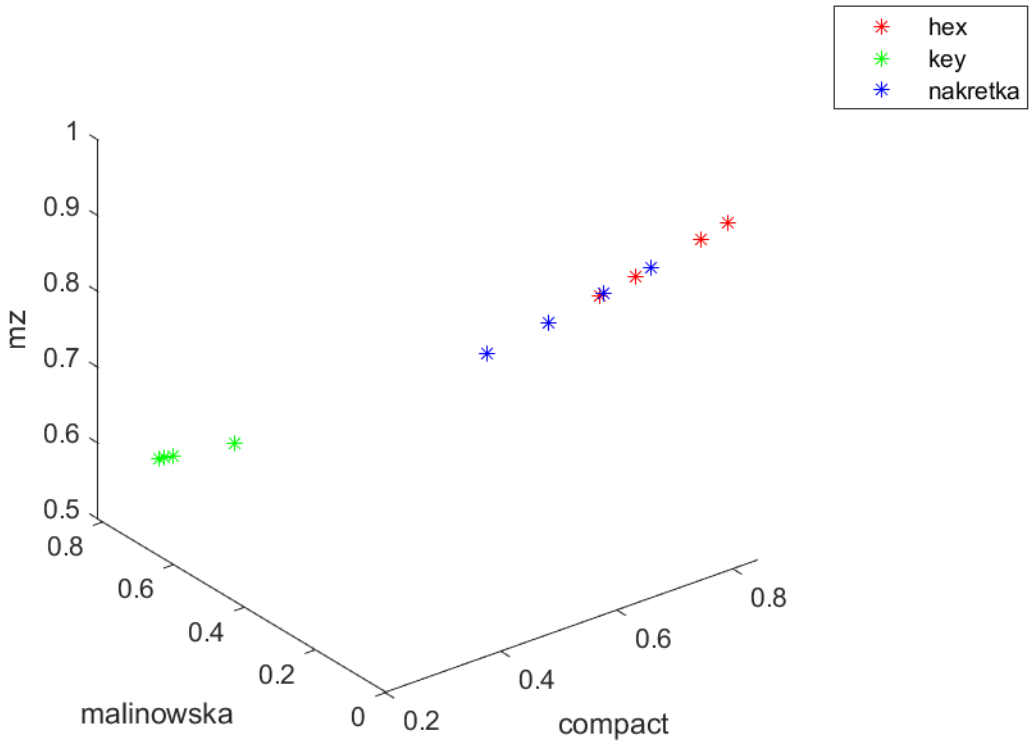
Rysunek 7



Rysunek 8

Widzimy, że klucz jest zdecydowanie oddziela się wartościami od pozostałych klas. W przypadku nakrętki oraz hexa, potrafią się znajdować blisko siebie ze względu na podobne kształty i rozmiar. W celu uzyskania lepszej klasyfikacji najlepiej byłoby użycie innych wskaźników, aby pomóc rozróżnić owe obiekty od siebie.

4.2 W przestrzeni cech 3D



Rysunek 9

W przypadku wykresu w 3D wnioski są takie same. Widzimy zdecydowanie oddalenie się klasy klucza od pozostałych dwóch klas, które lekko się pokrywają.

Wykorzystany kod:

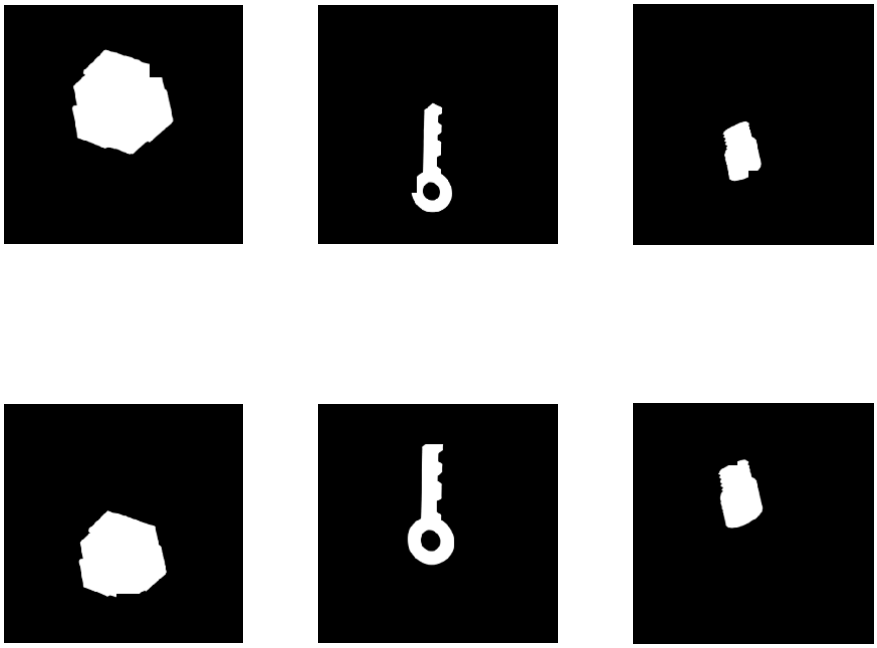
```
clc
clear
%%
% [file,path]=uigetfile('*.*', 'Select an image');
% img = imread([path, file]);


folders_name = ["hex","key","nakretka"];
plotting(folders_name)

%% Functions

function plotting(folder_name)
    figure(1);
    factor_1_c = [];
    factor_1_r = [];
    factor_1_t = [];
    factor_2_c = [];
    factor_2_r = [];
    factor_2_t = [];
    factor_3_c = [];
    factor_3_r = [];
    factor_3_t = [];
    for i=1:1:3
          for j=1:1:5
              file_name = append(folder_name(i),'/','1_',sprintf("%d",j-1),'.png');
              image_data = imread(file_name);
              [~,~,DIM] = size(image_data);
              if DIM > 1
                  image_data = rgb2gray(image_data);
              end
              switch i
                  case 1
                      k1 = count_factors(image_data,"compact");
                      factor_1_c = [factor_1_c k1];
                      k2 = count_factors(image_data,"malinowska");
                      factor_2_c = [factor_2_c k2];
                      k3 = count_factors(image_data,"mz");
                      factor_3_c = [factor_3_c k3];
                  case 2
                      k1 = count_factors(image_data,"compact");
                      factor_1_r = [factor_1_r k1];
                      k2 = count_factors(image_data,"malinowska");
                      factor_2_r = [factor_2_r k2];
                      k3 = count_factors(image_data,"mz");
                      factor_3_r = [factor_3_r k3];

                  case 3
                      k1 = count_factors(image_data,"compact");
                      factor_1_t = [factor_1_t k1];
                      k2 = count_factors(image_data,"malinowska");
                      factor_2_t = [factor_2_t k2];
                      k3 = count_factors(image_data,"mz");
                      factor_3_t = [factor_3_t k3];
              end
          end

    end
     figure(1);
     plot(factor_1_c,factor_2_c,"r*");
```

```
61      hold on
62      plot(factor_1_r,factor_2_r,"g*");
63      plot(factor_1_t,factor_2_t,"b*");
64      ylabel("compact");
65      xlabel("malinowska");
66      legend("hex","key","nakretka");
67
68      figure(2)
69      plot(factor_3_c,factor_1_c,"r*");
70      hold on
71      plot(factor_3_r,factor_1_r,"g*");
72      plot(factor_3_t,factor_1_t,"b*");
73
74      legend("hex","key","nakretka");
75      xlabel("mz");
76      ylabel("compact");
77
78      figure(3);
79      plot(factor_2_c,factor_3_c,"r*");
80      hold on
81      plot(factor_2_r,factor_3_r,"g*");
82      plot(factor_2_t,factor_3_t,"b*");
83      ylabel("malinowska");
84      xlabel("mz");
85      legend("hex","key","nakretka");
86
87      figure(4);
88      plot3(factor_1_c,factor_2_c,factor_3_c,"r*");
89      hold on
90      plot3(factor_1_r,factor_2_r,factor_3_r,"g*");
91      plot3(factor_1_t,factor_2_t,factor_3_t,"b*");
92      ylabel("malinowska");
93      xlabel("compact");
94      zlabel("mz");
95      legend("hex","key","nakretka");
96      exportgraphics(figure(1),"2D_1.eps");
97      exportgraphics(figure(2),"2D_2.eps");
98      exportgraphics(figure(3),"2D_3.eps");
99      exportgraphics(figure(4),"3D.eps");
100 end
```

zad3.m

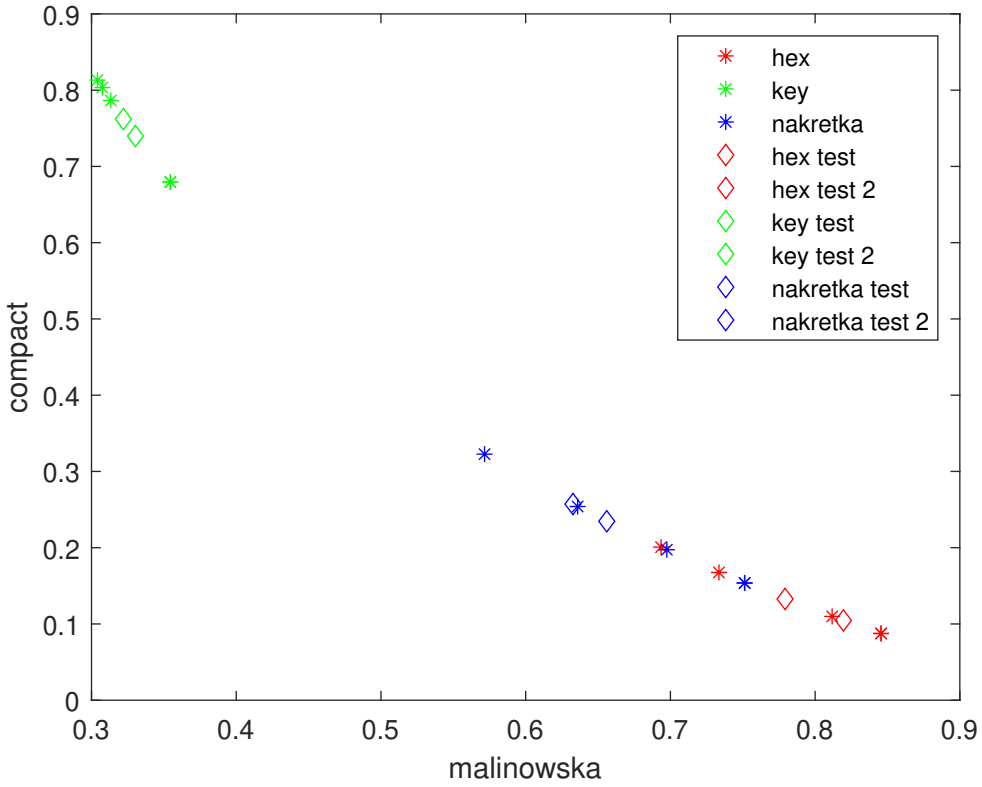
5 Klasyfikator k-NN

Rysunek 10: Zdjęcia oryginalne

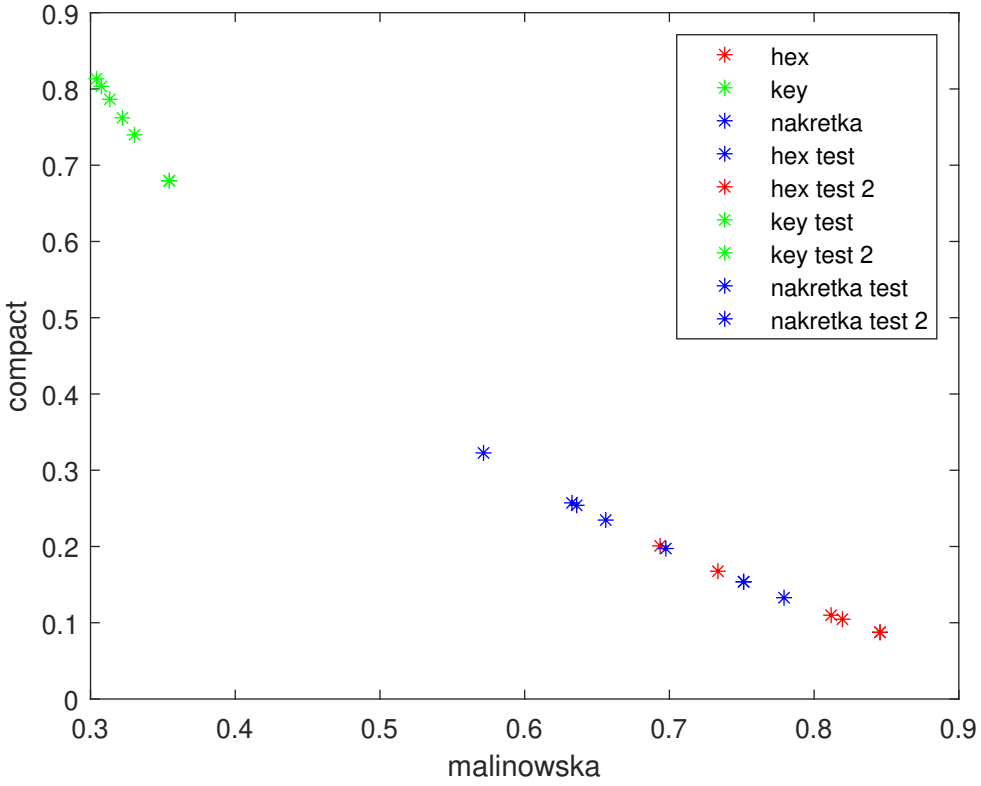


Rysunek 11: Zdjęcia testowe

Zdjęcia zostały specjalnie zmodyfikowane aby różniły się od pozostałych ze swojej klasy.

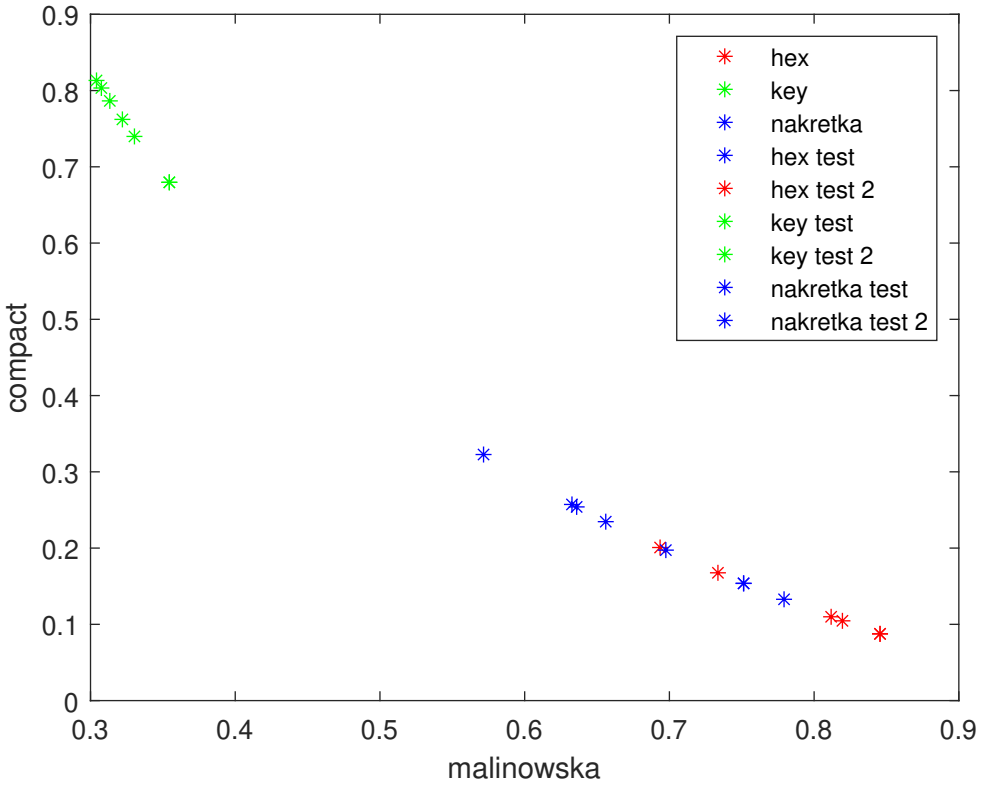


Rysunek 12: Rozkład przed klasyfikacją



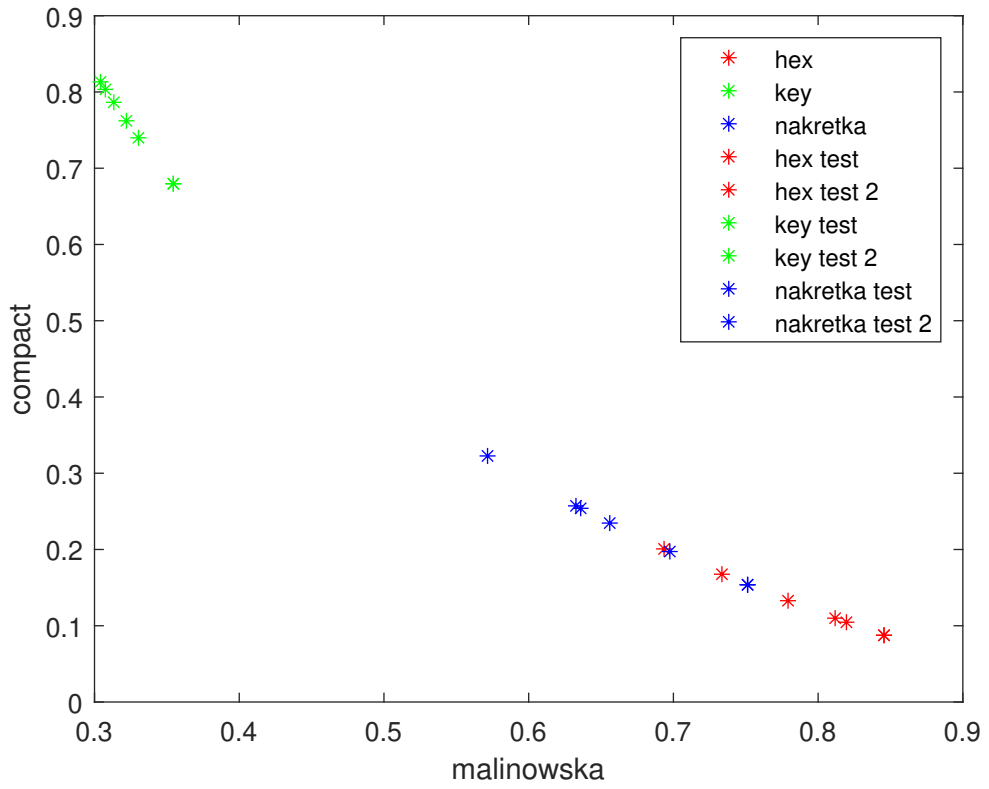
Rysunek 13: Klasyfikator k-NN dla k =1

Dla k=1 widzimy, że hex został źle dopisany do klasy.

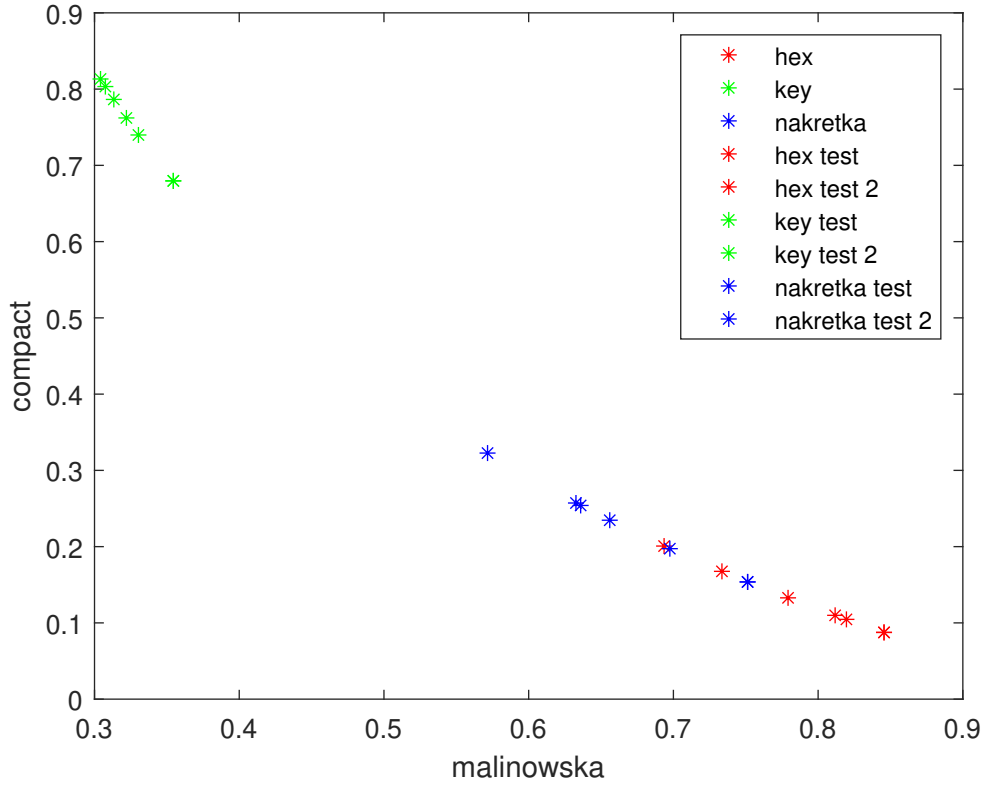


Rysunek 14: Klasyfikator k-NN dla k =3

W przypadku k = 3 sytuacja się nie zmieniła i nadal klasyfikacja przebiegła pomyślnie poza jednym przypadkiem.



Rysunek 15: Klasyfikator k-NN dla k =5



Rysunek 16: Klasyfikator k-NN dla k =7

Dla k = 5 oraz k=7 klasyfikacja przebiega w pełni pomyślnie. Zdecydowanie widać polepszenia się jakości klasyfikacji wraz ze wzrostem wartości k, która oznacza ilość punktów z otoczenia, których większość będzie determinować przypisanie.

Kod zaimplementowany w języku Matlab.

```
1
2 function final_result =  knn(k)
```

```
    folder_name = ["hex","key","nakretka"];
    factor_1_c = [];
    factor_1_r = [];
    factor_1_t = [];
    factor_2_c = [];
    factor_2_r = [];
    factor_2_t = [];
    factor_3_c = [];
    factor_3_r = [];
    factor_3_t = [];
    for i=1:1:3
        for j=1:1:5
            file_name = append(folder_name(i),'/','1_',sprintf("%d",j-1),'.png');
            image_data = imread(file_name);
            [~,~,DIM] = size(image_data);
            if DIM > 1
                image_data = rgb2gray(image_data);
            end
            switch i
                case 1 %circle
                    k1 = count_factors(image_data,"compact");
                    factor_1_c = [factor_1_c k1];
                    k2 = count_factors(image_data,"malinowska");
                    factor_2_c = [factor_2_c k2];
                    k3 = count_factors(image_data,"mz");
                    factor_3_c = [factor_3_c k3];
                case 2 %rectangle
                    k1 = count_factors(image_data,"compact");
                    factor_1_r = [factor_1_r k1];
                    k2 = count_factors(image_data,"malinowska");
                    factor_2_r = [factor_2_r k2];
                    k3 = count_factors(image_data,"mz");
                    factor_3_r = [factor_3_r k3];

                case 3 %triangle
                    k1 = count_factors(image_data,"compact");
                    factor_1_t = [factor_1_t k1];
                    k2 = count_factors(image_data,"malinowska");
                    factor_2_t = [factor_2_t k2];
                    k3 = count_factors(image_data,"mz");
                    factor_3_t = [factor_3_t k3];
            end
        end
    end
        [circle,rectangle,triangle] = knn_test_factors();
        test_points = [circle{1}{1} circle{1}{2}; circle{2}{1} circle{2}{2}; ...
            rectangle{2}{1} rectangle{2}{2};rectangle{1}{1} rectangle{1}{2};triangle
                {1}{1} triangle{1}{2} ; ...
            triangle{2}{1} triangle{2}{2}];
        factor_points = zeros(15,2);
        factor_points(:, 1) = [factor_1_c factor_1_r factor_1_t];
        factor_points(:,2) = [factor_2_c factor_2_r factor_2_t];
        factor_points(:,3) = [ones(5,1); ones(5,1)+1; ones(5,1)+2];
        final_result = [];
        for i=1:1:6
            factor_points (:,4) = pdist2(factor_points(:,1:2),[test_points(i,1)
                test_points(i,2)]);
            [class, sort_indx]= sort(factor_points,1);
            results = factor_points(:,3);
            class_result = results(sort_indx(:,4));
            result = mode(class_result(1:k));
            final_result = [final_result result];
        end

end
```

knn.m

Pozostałe użyte funkcje:

```
function [factor_1,factor_2,factor_3,mode_iterator] = plotter(img_1,img_2,img_3,mode,
    factor)
    factor_1 = [];
    factor_2 = [];
    factor_3 = [];
    switch mode
        case "rotate"
            mode_iterator = 1:180;
            X = 180;
        case "translate"
            mode_iterator = 1:16;
            X = 16;
        case "scale"
            mode_iterator = 1:16;
            X = 16;
    end
    for i=1:1:X
        disp(i);
        switch mode
            case "rotate"
                changed_img_1 = imrotate(img_1,i);
                changed_img_2 = imrotate(img_2,i);
                changed_img_3 = imrotate(img_3,i);
            case "translate"
                changed_img_1 = imtranslate(img_1,[i, i],'FillValues',0);
                changed_img_2 = imtranslate(img_2,[i, i],'FillValues',0);
                changed_img_3 = imtranslate(img_3,[i, i],'FillValues',0);
            case "scale"
                changed_img_1 = imresize(img_1,0.1*i);
                changed_img_2 = imresize(img_2,0.1*i);
                changed_img_3 = imresize(img_3,0.1*i);
        end
        k1 = count_factors(changed_img_1,factor);
        k2 = count_factors(changed_img_2,factor);
        k3 = count_factors(changed_img_3,factor);
        factor_1 = [factor_1 k1];
        factor_2 = [factor_2 k2];
        factor_3 = [factor_3 k3];
    end
    disp("done");
end
```

plotter.m

```
function [circle,rectangle,triangle] = knn_test_factors()

    circle_1 = change_dim(imread("knn/hex_1.png"));
    circle_2 = change_dim(imread("knn/hex_2.png"));
    rectangle_1 = change_dim(imread("knn/key_1.png"));
    rectangle_2 = change_dim(imread("knn/key_2.png"));
    triangle_1 = change_dim(imread("knn/nakretka_1.png"));
    triangle_2 = change_dim(imread("knn/nakretka_2.png"));
    img_data = {circle_1 circle_2 rectangle_1 rectangle_2 triangle_1 triangle_2};

%     K1 - compact k2 - Malinowska k3 - mz
    circle{1}{1} = count_factors(img_data{1},"compact");
    circle{1}{2} = count_factors(img_data{1},"malinowska");
    circle{1}{3} = count_factors(img_data{1},"mz");

    circle{2}{1} = count_factors(img_data{2},"compact");
```

```
    circle{2}{2} = count_factors(img_data{2},"malinowska");
    circle{2}{3} = count_factors(img_data{2},"mz");

    rectangle{1}{1} = count_factors(img_data{3},"compact");
    rectangle{1}{2} = count_factors(img_data{3},"malinowska");
    rectangle{1}{3} = count_factors(img_data{3},"mz");

    rectangle{2}{1} = count_factors(img_data{4},"compact");
    rectangle{2}{2} = count_factors(img_data{4},"malinowska");
    rectangle{2}{3} = count_factors(img_data{4},"mz");

    triangle{1}{1} = count_factors(img_data{5},"compact");
    triangle{1}{2} = count_factors(img_data{5},"malinowska");
    triangle{1}{3} = count_factors(img_data{5},"mz");

    triangle{2}{1} = count_factors(img_data{6},"compact");
    triangle{2}{2} = count_factors(img_data{6},"malinowska");
    triangle{2}{3} = count_factors(img_data{6},"mz");
end
```

knn_test_factors.m

```
function result = crop_img(img,scale)
    dif_size = 256;
    scaled_img = imresize(img,scale,"bicubic");
    [X,Y] = size(scaled_img);
    result = scaled_img((X/2)-dif_size:(X/2)+dif_size,(Y/2)-dif_size:(Y/2)+dif_size);
end
```

crop_img.m

```
function img = change_dim(img)
    [~,~,DIM] = size(img);
    if DIM > 1
        img = rgb2gray(img);
    end
end
```

change_dim.m