

Gliwice, 22.11.2022

# Przetwarzanie Obrazów Cyfrowych

## CECHY NIEZMIENNICZE I KLASYFIKACJA OBIEKTÓW



**Politechnika  
Śląska**

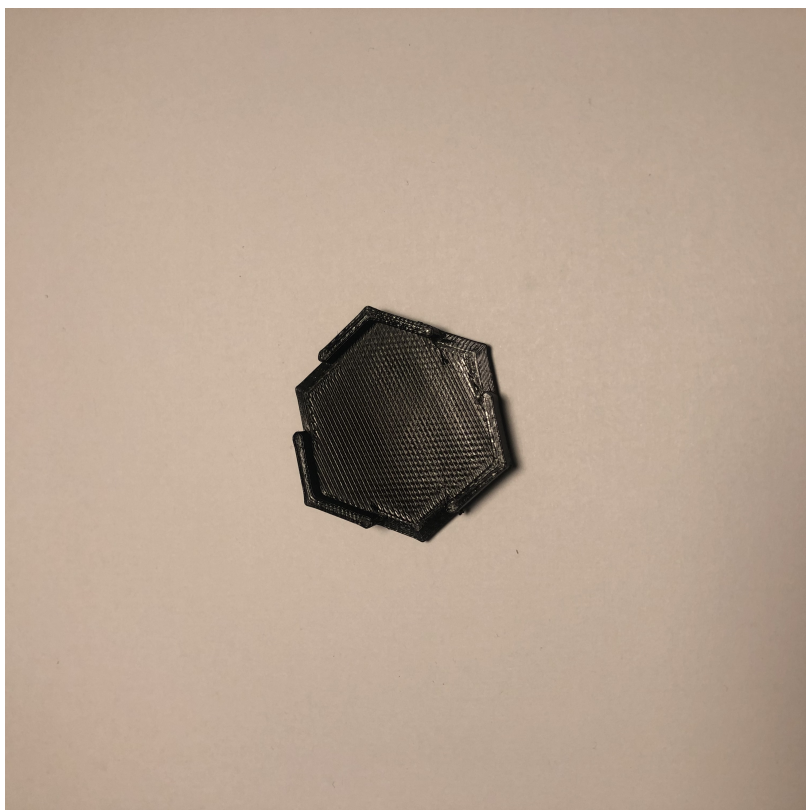
Jakub Zeifert

# 1 Przygotowanie obrazów testowych

(Zaliczone podczas zajęć) Do laboratorium zostały przygotowane 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:

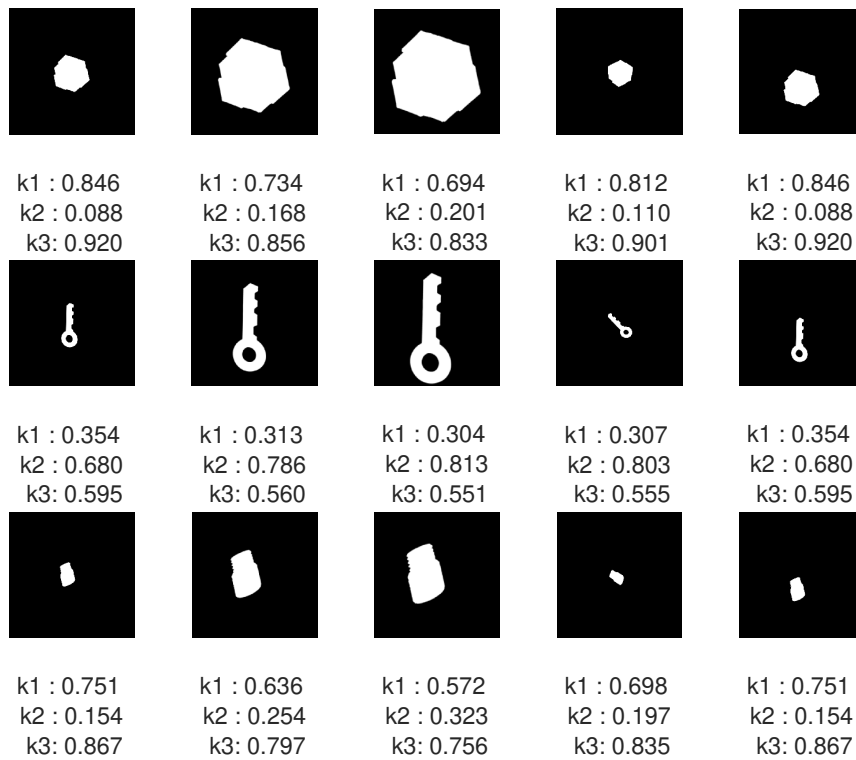
```

1 function [k1 , k2] = count_factors(img, type)
2     stats = regionprops('table', img, "Area", "Perimeter");
3     area = double(stats.Area(255));
4     perimeter = double(stats.Perimeter(255));
5     switch type
6         case "compact"
7             k1 = (area*4*pi)/(perimeter*perimeter);
8         case "circular"
9             k1 = 2 * sqrt(area/pi);
10            k2 = perimeter/pi;
11
12         case "malinowska"
13             k1 = perimeter/(2*sqrt(pi * area)) - 1;
14         case "mz"
15             k1 = (2*sqrt(pi*area))/perimeter;
16         otherwise
17             disp("ERROR!!! WRONG TYPE INPUT");
18     end
19 end

```

count\_factors.m

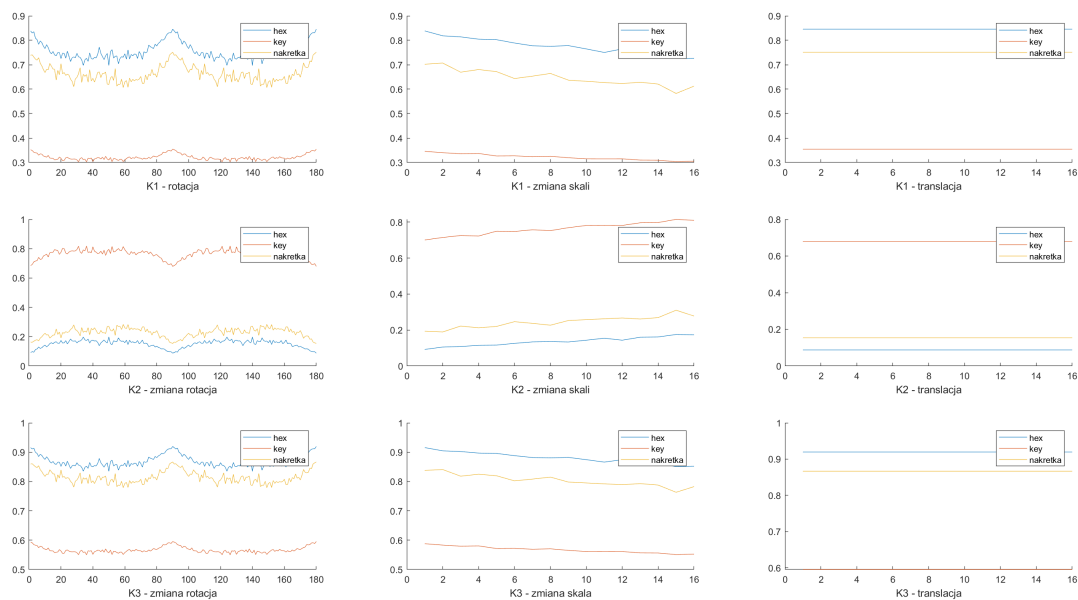
- k1 - Kompaktowy ( $K = \frac{4\pi \cdot S}{L}$ )
- k2 - Malionwskiej ( $M = \frac{L}{2 \cdot \sqrt{\pi \cdot S}} - 1$ )
- k3 - Mz ( $K = \frac{2 \cdot \sqrt{\pi \cdot S}}{L}$ )



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 wskaźniki polegają na wielkości obiektu, który zmienia się wraz z obrotem jak i również skalowaniem. W przypadku translacji wskaźniki wskazują tę samą wartość. Wynika to z faktu, że obiekt nie zmienia żadnego parametru, który może go opisać.

Wykorzystany kod:

```
1 clc
2 clear
```

```

3 %%
4 [file ,path]=uigetfile('*.','Select an image');
5 img_1 = imread([path, file]);
6
7 [~,~,DIM] = size(img_1);
8 if DIM > 1
9     img_1 = rgb2gray(img_1);
10 end
11
12
13 [file ,path]=uigetfile('*.','Select an image');
14 img_2 = imread([path, file]);
15 [~,~,DIM] = size(img_2);
16 if DIM > 1
17     img_2 = rgb2gray(img_2);
18 end
19
20 [file ,path]=uigetfile('*.','Select an image');
21 img_3 = imread([path, file]);
22 [~,~,DIM] = size(img_3);
23 if DIM > 1
24     img_3 = rgb2gray(img_3);
25 end
26
27 % zdj_1 = "circle";
28 % zdj_2 = "triangle";
29 % zdj_3 = "rectangle";
30 zdj_1 = "hex";
31 zdj_2 = "key";
32 zdj_3 = "nakretka";
33
34
35 figure(1);
36 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","compact");
37 subplot(3,3,1);
38 hold on;
39 plot(X,k1);
40 plot(X,k2);
41 plot(X,k3);
42 xlabel("K1 - rotacja");
43 legend(zdj_1,zdj_2,zdj_3);
44
45 subplot(3,3,2);
46 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","compact");
47 hold on;
48 plot(X,k1);
49 plot(X,k2);
50 plot(X,k3);
51 xlabel("K1 - zmiana skali");
52 legend(zdj_1,zdj_2,zdj_3);
53
54 subplot(3,3,3);
55 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","compact");
56 hold on;
57 plot(X,k1);
58 plot(X,k2);
59 plot(X,k3);
60 xlabel("K1 - translacja");
61 legend(zdj_1,zdj_2,zdj_3);
62
63 subplot(3,3,4);
64 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","malinowska");
65 hold on;
66 plot(X,k1);
67 plot(X,k2);

```

```

68 plot(X,k3);
69 xlabel("K2 - zmiana rotacja");
70 legend(zdj_1,zdj_2,zdj_3);
71
72 subplot(3,3,5);
73 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","malinowska");
74 hold on;
75 plot(X,k1);
76 plot(X,k2);
77 plot(X,k3);
78 xlabel("K2 - zmiana skali");
79 legend(zdj_1,zdj_2,zdj_3);
80
81 subplot(3,3,6);
82 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","malinowska");
83 hold on;
84 plot(X,k1);
85 plot(X,k2);
86 plot(X,k3);
87 xlabel("K2 - translacja");
88 legend(zdj_1,zdj_2,zdj_3);
89
90 subplot(3,3,7);
91 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"rotate","mz");
92 hold on;
93 plot(X,k1);
94 plot(X,k2);
95 plot(X,k3);
96 xlabel("K3 - zmiana rotacja");
97 legend(zdj_1,zdj_2,zdj_3);
98
99 subplot(3,3,8);
100 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"scale","mz");
101 hold on;
102 plot(X,k1);
103 plot(X,k2);
104 plot(X,k3);
105 xlabel("K3 - zmiana skala");
106 legend(zdj_1,zdj_2,zdj_3);
107
108 subplot(3,3,9);
109 [k1,k2,k3,X] = plotter(img_1,img_2,img_3,"translate","mz");
110 hold on;
111 plot(X,k1);
112 plot(X,k2);
113 plot(X,k3);
114 xlabel("K3 - translacja");
115 legend(zdj_1,zdj_2,zdj_3);
116
117
118 exportgraphics(figure(1),"zad3.eps");
119 %% Functions
120
121
122
123 function [factor_1,factor_2,factor_3,mode_iterator] = plotter(img_1,img_2,img_3,mode,
    factor)
124     factor_1 = [];
125     factor_2 = [];
126     factor_3 = [];
127     switch mode
128         case "rotate"
129             mode_iterator = 1:180;
130             X = 180;
131             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

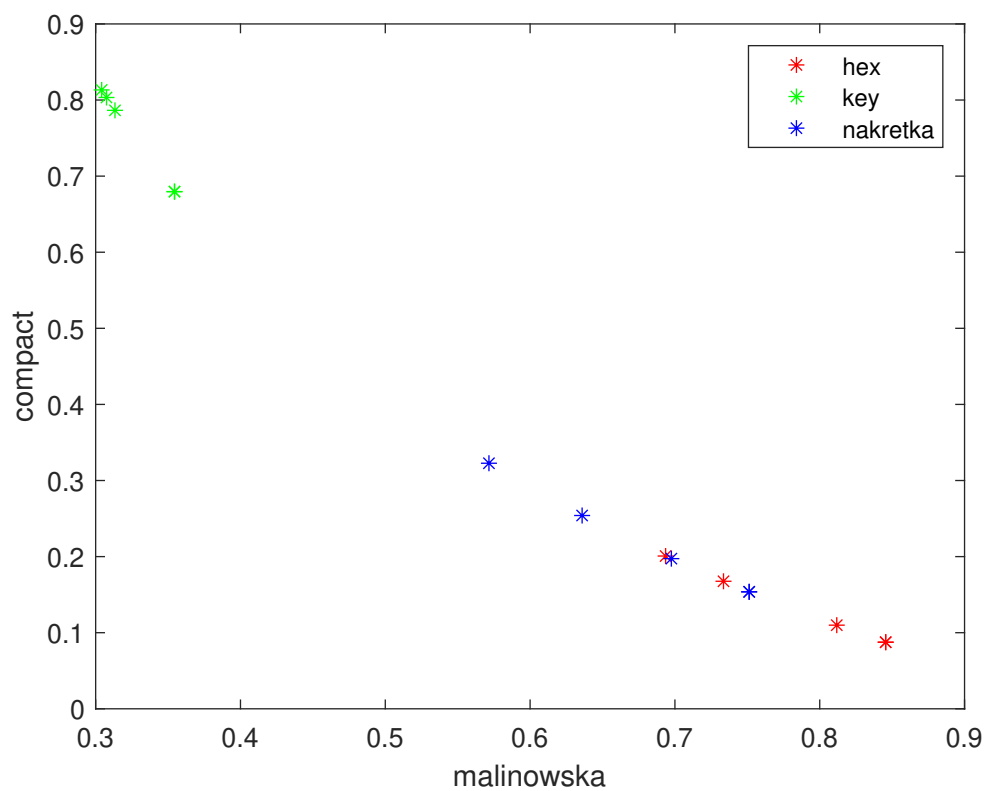
```

zad2.m

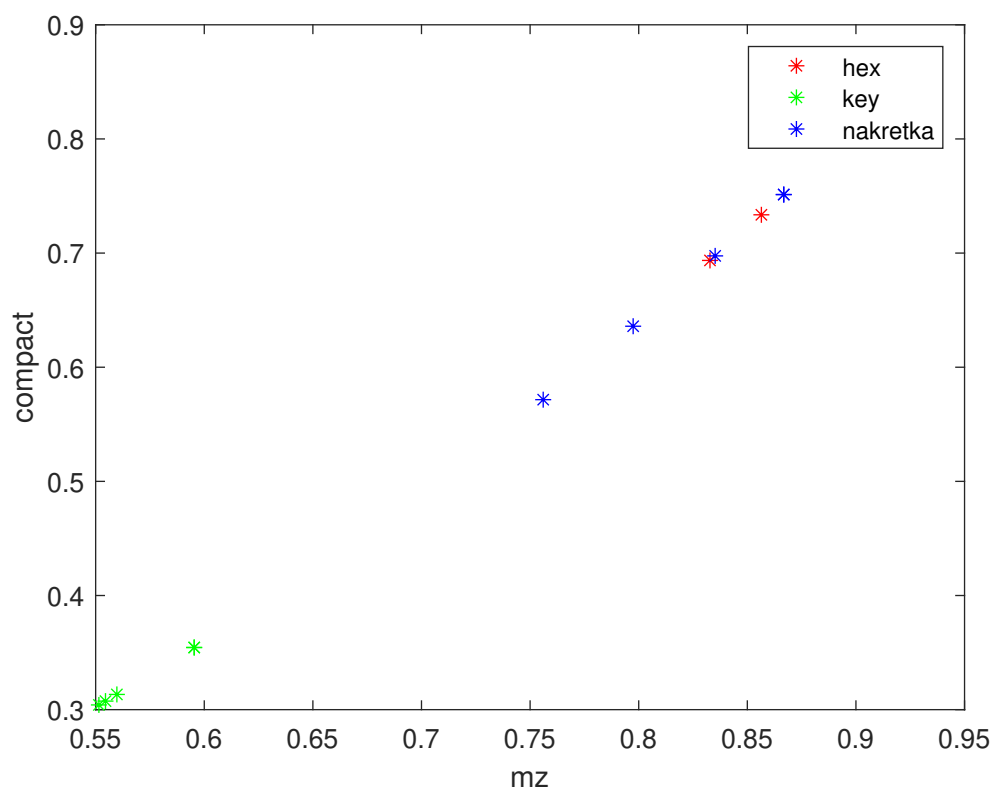
## 4 Wizualizacja obiektów

(Zaliczone podczas zajęć)

### 4.1 Na płaszczyźnie cech 2D

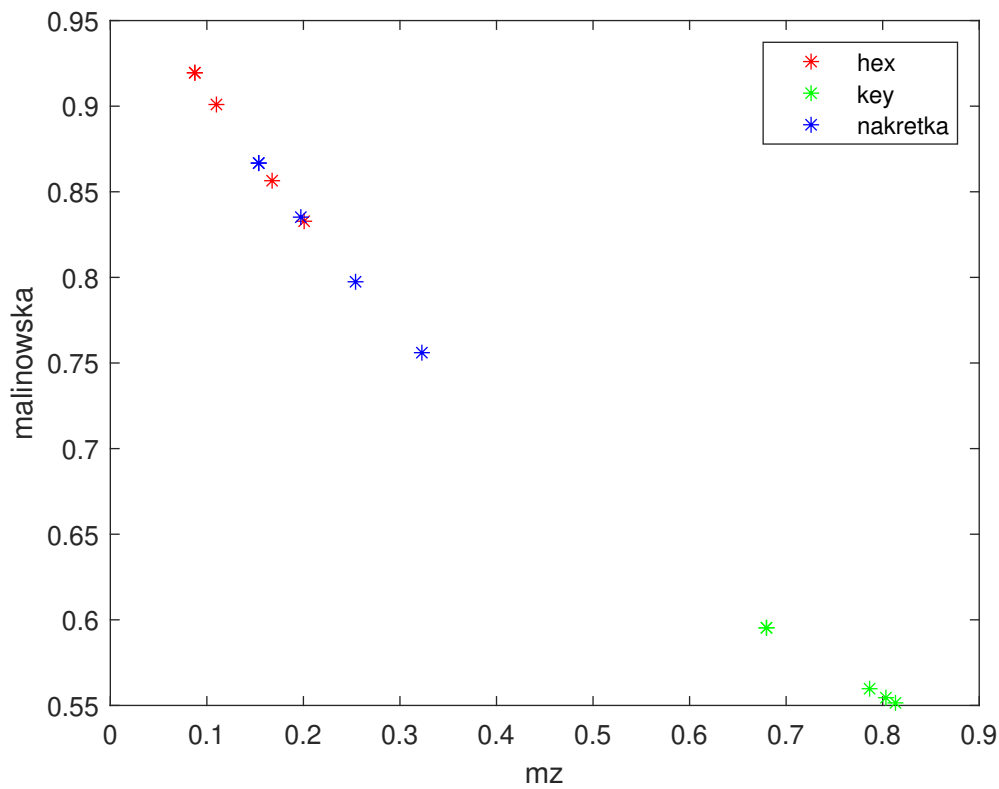


Rysunek 6



Rysunek 7

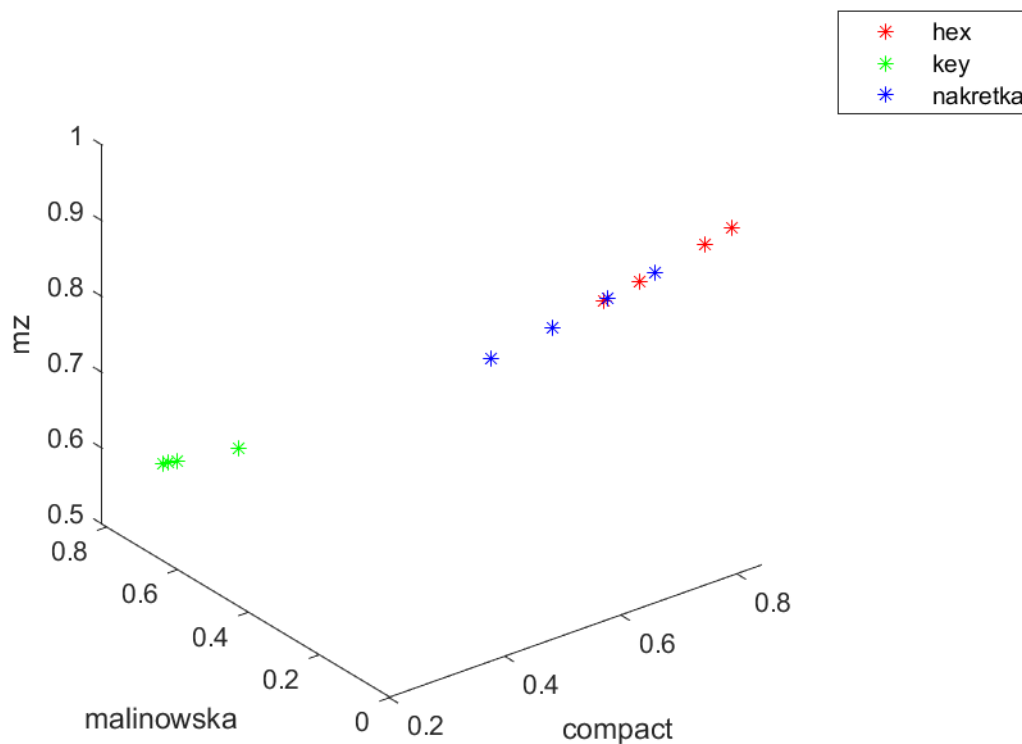




Rysunek 8

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:

```
1 clc
2 clear
3 %%
4 % [file,path]=uigetfile('*.','Select an image');
5 % img = imread([path, file]);
6
7
8 folders_name = ["hex","key","nakretka"];
9 plotting(folders_name)
10
11 %% Functions
12
13 function plotting(folder_name)
14     figure(1);
15     factor_1_c = [];
16     factor_1_r = [];
17     factor_1_t = [];
18     factor_2_c = [];
19     factor_2_r = [];
20     factor_2_t = [];
21     factor_3_c = [];
22     factor_3_r = [];
23     factor_3_t = [];
24     for i=1:1:3
25         for j=1:1:5
26             file_name = append(folder_name(i), '/', '1_', sprintf("%d",j-1), '.png');
27             image_data = imread(file_name);
28             [~,~,DIM] = size(image_data);
29             if DIM > 1
30                 image_data = rgb2gray(image_data);
31             end
32             switch i
33                 case 1
34                     k1 = count_factors(image_data,"compact");
35                     factor_1_c = [factor_1_c k1];
36                     k2 = count_factors(image_data,"malinowska");
37                     factor_2_c = [factor_2_c k2];
38                     k3 = count_factors(image_data,"mz");
39                     factor_3_c = [factor_3_c k3];
40                 case 2
41                     k1 = count_factors(image_data,"compact");
42                     factor_1_r = [factor_1_r k1];
43                     k2 = count_factors(image_data,"malinowska");
44                     factor_2_r = [factor_2_r k2];
45                     k3 = count_factors(image_data,"mz");
46                     factor_3_r = [factor_3_r k3];
47
48                 case 3
49                     k1 = count_factors(image_data,"compact");
50                     factor_1_t = [factor_1_t k1];
51                     k2 = count_factors(image_data,"malinowska");
52                     factor_2_t = [factor_2_t k2];
53                     k3 = count_factors(image_data,"mz");
54                     factor_3_t = [factor_3_t k3];
55             end
56         end
57     end
58     figure(1);
59     plot(factor_1_c, factor_2_c, "r*");
60
```

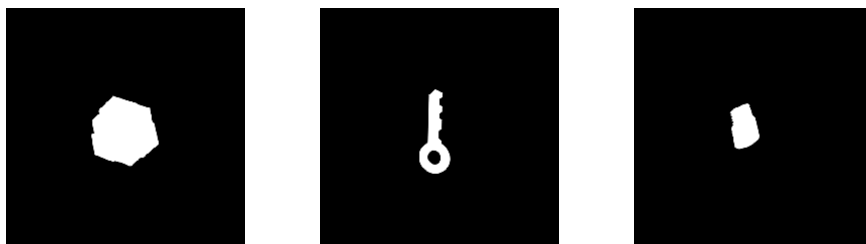
```

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

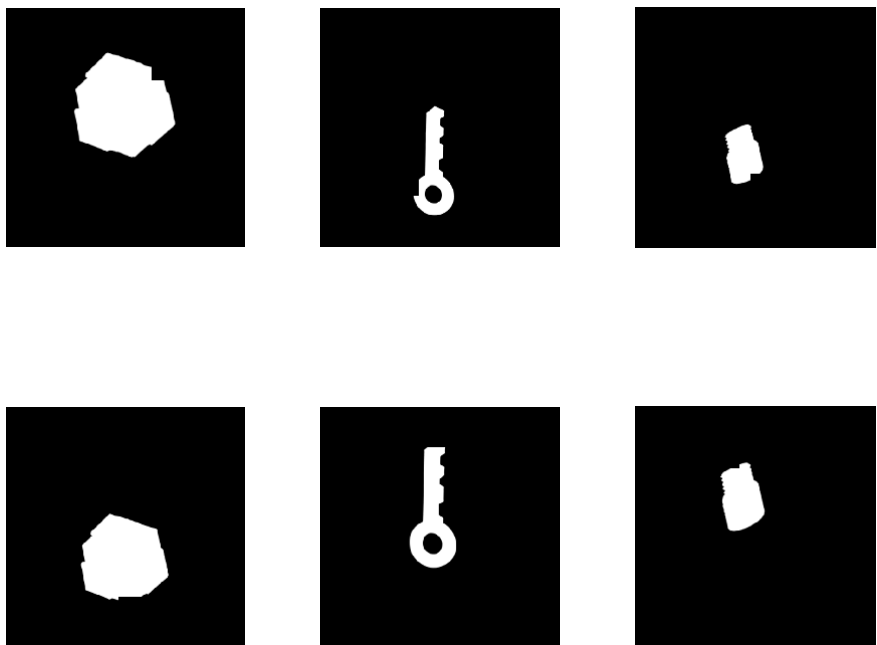
```

zad3.m

## 5 Klasyfikator k-NN

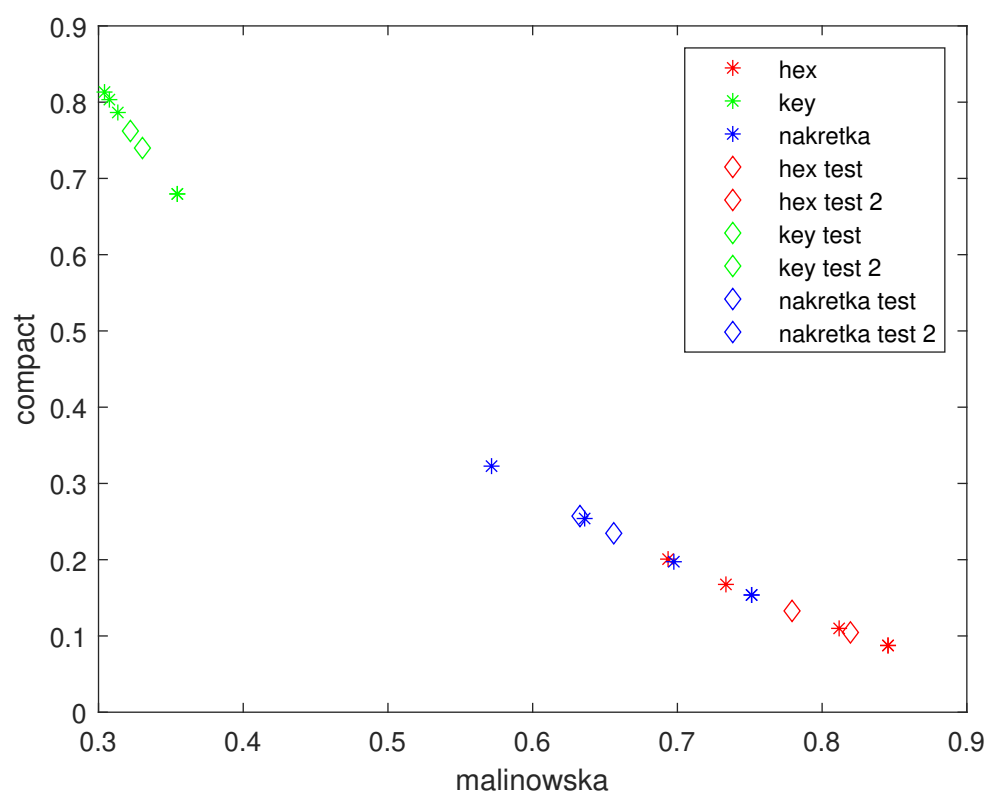


Rysunek 10: Zdjęcia oryginalne

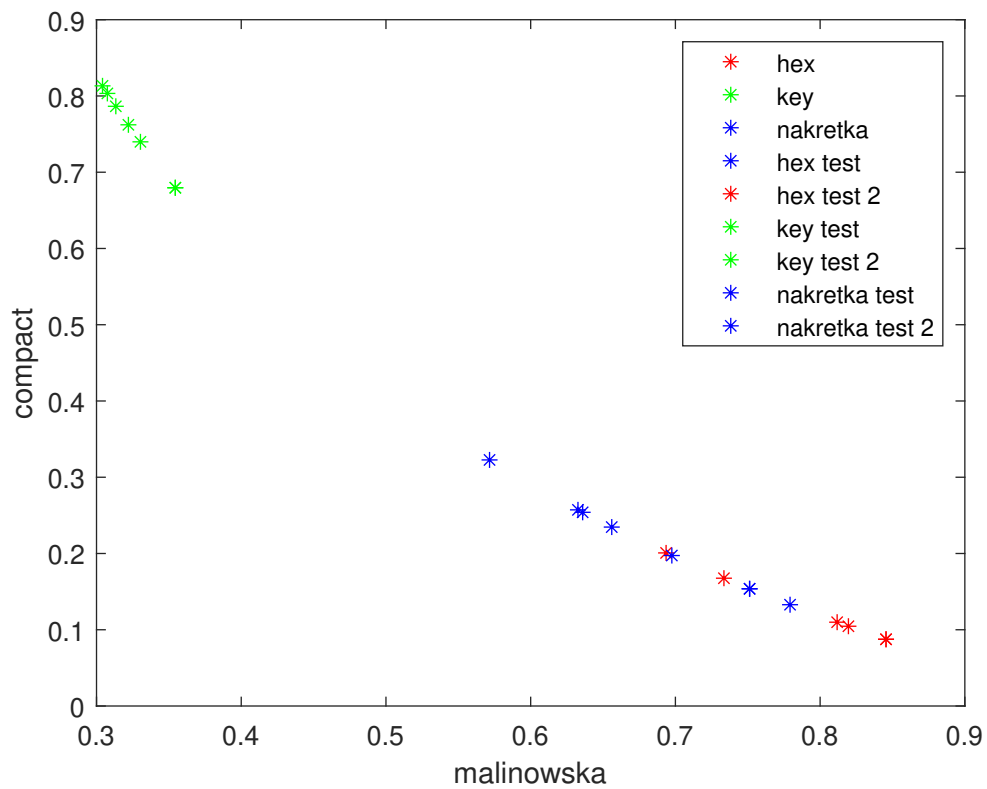


Rysunek 11: Zdjęcia testowe

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

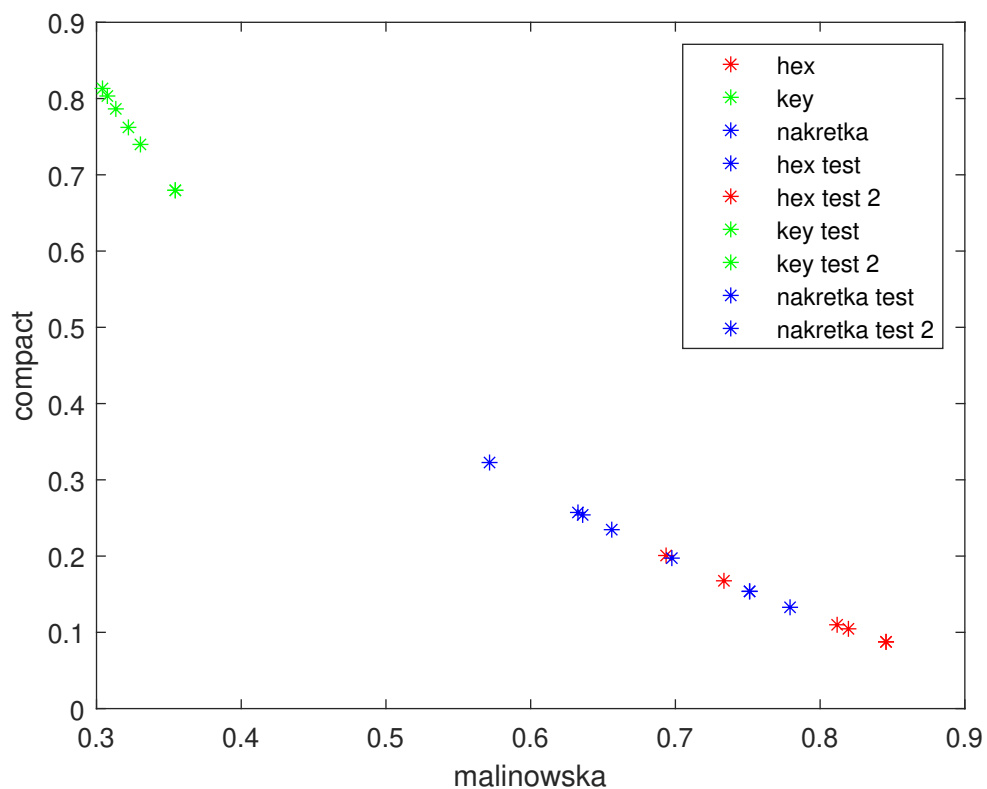


Rysunek 12: Rozkład przed klasyfikacją



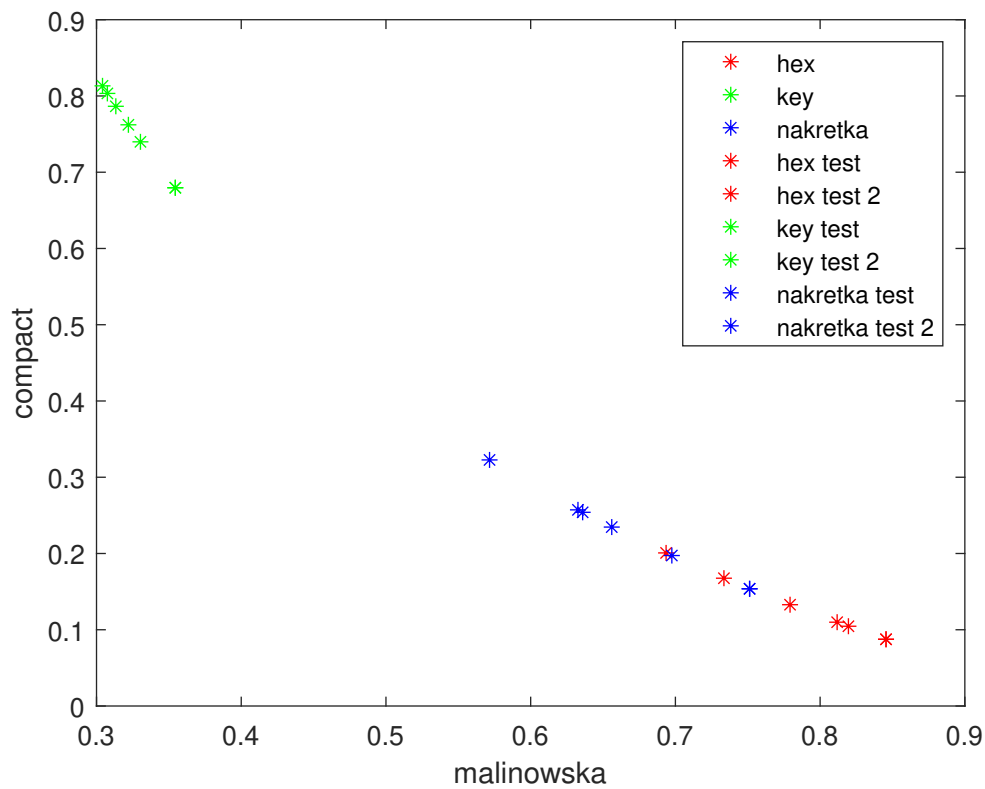
Rysunek 13: Klasyfikator k-NN dla  $k = 1$

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

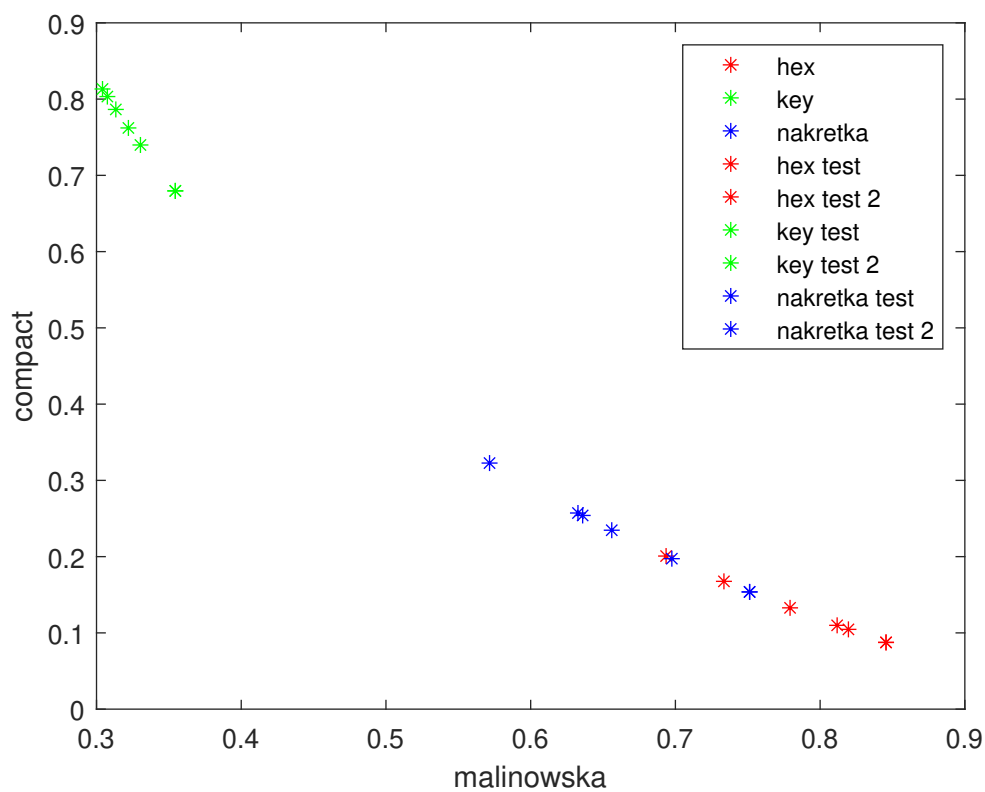


Rysunek 14: Klasyfikator k-NN dla  $k = 3$

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)
```

```

3   folder_name = ["hex","key","nakretka"];
4   factor_1_c = [];
5   factor_1_r = [];
6   factor_1_t = [];
7   factor_2_c = [];
8   factor_2_r = [];
9   factor_2_t = [];
10  factor_3_c = [];
11  factor_3_r = [];
12  factor_3_t = [];
13  for i=1:1:3
14      for j=1:1:5
15          file_name = append(folder_name(i), '/', '1_', sprintf("%d",j-1), '.png');
16          image_data = imread(file_name);
17          [~,~,DIM] = size(image_data);
18          if DIM > 1
19              image_data = rgb2gray(image_data);
20          end
21          switch i
22              case 1 %circle
23                  k1 = count_factors(image_data,"compact");
24                  factor_1_c = [factor_1_c k1];
25                  k2 = count_factors(image_data,"malinowska");
26                  factor_2_c = [factor_2_c k2];
27                  k3 = count_factors(image_data,"mz");
28                  factor_3_c = [factor_3_c k3];
29              case 2 %rectangle
30                  k1 = count_factors(image_data,"compact");
31                  factor_1_r = [factor_1_r k1];
32                  k2 = count_factors(image_data,"malinowska");
33                  factor_2_r = [factor_2_r k2];
34                  k3 = count_factors(image_data,"mz");
35                  factor_3_r = [factor_3_r k3];
36
37              case 3 %triangle
38                  k1 = count_factors(image_data,"compact");
39                  factor_1_t = [factor_1_t k1];
40                  k2 = count_factors(image_data,"malinowska");
41                  factor_2_t = [factor_2_t k2];
42                  k3 = count_factors(image_data,"mz");
43                  factor_3_t = [factor_3_t k3];
44          end
45      end
46  end
47  [circle,rectangle,triangle] = knn_test_factors();
48  test_points = [circle{1}{1} circle{1}{2}; circle{2}{1} circle{2}{2}; ...
49                rectangle{2}{1} rectangle{2}{2};rectangle{1}{1} rectangle{1}{2};triangle
50                {1}{1} triangle{1}{2} ; ...
51                triangle{2}{1} triangle{2}{2}];
52  factor_points = zeros(15,2);
53  factor_points(:, 1) = [factor_1_c factor_1_r factor_1_t];
54  factor_points(:,2) = [factor_2_c factor_2_r factor_2_t];
55  factor_points(:,3) = [ones(5,1); ones(5,1)+1; ones(5,1)+2];
56  final_result = [];
57  for i=1:1:6
58      factor_points(:,4) = pdist2(factor_points(:,1:2),[test_points(i,1)
59          test_points(i,2)]);
60      [class, sort_indx]= sort(factor_points,1);
61      results = factor_points(:,3);
62      class_result = results(sort_indx(:,4));
63      result = mode(class_result(1:k));
64      final_result = [final_result result];
65  end
end

```

Pozostałe użyte funkcje:

```

1
2 function [factor_1, factor_2, factor_3, mode_iterator] = plotter(img_1, img_2, img_3, mode,
3     factor)
4     factor_1 = [];
5     factor_2 = [];
6     factor_3 = [];
7     switch mode
8         case "rotate"
9             mode_iterator = 1:180;
10            X = 180;
11        case "translate"
12            mode_iterator = 1:16;
13            X = 16;
14        case "scale"
15            mode_iterator = 1:16;
16            X = 16;
17    end
18    for i=1:1:X
19        disp(i);
20        switch mode
21            case "rotate"
22                changed_img_1 = imrotate(img_1, i);
23                changed_img_2 = imrotate(img_2, i);
24                changed_img_3 = imrotate(img_3, i);
25            case "translate"
26                changed_img_1 = imtranslate(img_1, [i, i], 'FillValues', 0);
27                changed_img_2 = imtranslate(img_2, [i, i], 'FillValues', 0);
28                changed_img_3 = imtranslate(img_3, [i, i], 'FillValues', 0);
29            case "scale"
30                changed_img_1 = imresize(img_1, 0.1*i);
31                changed_img_2 = imresize(img_2, 0.1*i);
32                changed_img_3 = imresize(img_3, 0.1*i);
33        end
34        k1 = count_factors(changed_img_1, factor);
35        k2 = count_factors(changed_img_2, factor);
36        k3 = count_factors(changed_img_3, factor);
37        factor_1 = [factor_1 k1];
38        factor_2 = [factor_2 k2];
39        factor_3 = [factor_3 k3];
40    end
41    disp("done");
42 end

```

plotter.m

```

1 function [circle, rectangle, triangle] = knn_test_factors()
2
3     circle_1 = change_dim(imread("knn/hex_1.png"));
4     circle_2 = change_dim(imread("knn/hex_2.png"));
5     rectangle_1 = change_dim(imread("knn/key_1.png"));
6     rectangle_2 = change_dim(imread("knn/key_2.png"));
7     triangle_1 = change_dim(imread("knn/nakretka_1.png"));
8     triangle_2 = change_dim(imread("knn/nakretka_2.png"));
9     img_data = {circle_1 circle_2 rectangle_1 rectangle_2 triangle_1 triangle_2};
10
11 % K1 - compact k2 - Malinowska k3 - mz
12 circle{1}{1} = count_factors(img_data{1}, "compact");
13 circle{1}{2} = count_factors(img_data{1}, "malinowska");
14 circle{1}{3} = count_factors(img_data{1}, "mz");
15
16 circle{2}{1} = count_factors(img_data{2}, "compact");

```



```

17 circle{2}{2} = count_factors(img_data{2},"malinowska");
18 circle{2}{3} = count_factors(img_data{2},"mz");
19
20 rectangle{1}{1} = count_factors(img_data{3},"compact");
21 rectangle{1}{2} = count_factors(img_data{3},"malinowska");
22 rectangle{1}{3} = count_factors(img_data{3},"mz");
23
24 rectangle{2}{1} = count_factors(img_data{4},"compact");
25 rectangle{2}{2} = count_factors(img_data{4},"malinowska");
26 rectangle{2}{3} = count_factors(img_data{4},"mz");
27
28 triangle{1}{1} = count_factors(img_data{5},"compact");
29 triangle{1}{2} = count_factors(img_data{5},"malinowska");
30 triangle{1}{3} = count_factors(img_data{5},"mz");
31
32 triangle{2}{1} = count_factors(img_data{6},"compact");
33 triangle{2}{2} = count_factors(img_data{6},"malinowska");
34 triangle{2}{3} = count_factors(img_data{6},"mz");
35 end

```

knn\_test\_factors.m

```

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

```

crop\_img.m

```

1 function img = change_dim(img)
2     [~,~,DIM] = size(img);
3     if DIM > 1
4         img = rgb2gray(img);
5     end
6 end

```

change\_dim.m