2.12.2020

Robot Vision

RV ex7 Shape factors and unsupervised object classification

Mateusz Warmuz

1. Propose an algorithm for calculating values of two shape factors: compactness k1 and circularity factor k2 (see APPENDIX A). Implement your algorithm in MATLAB.

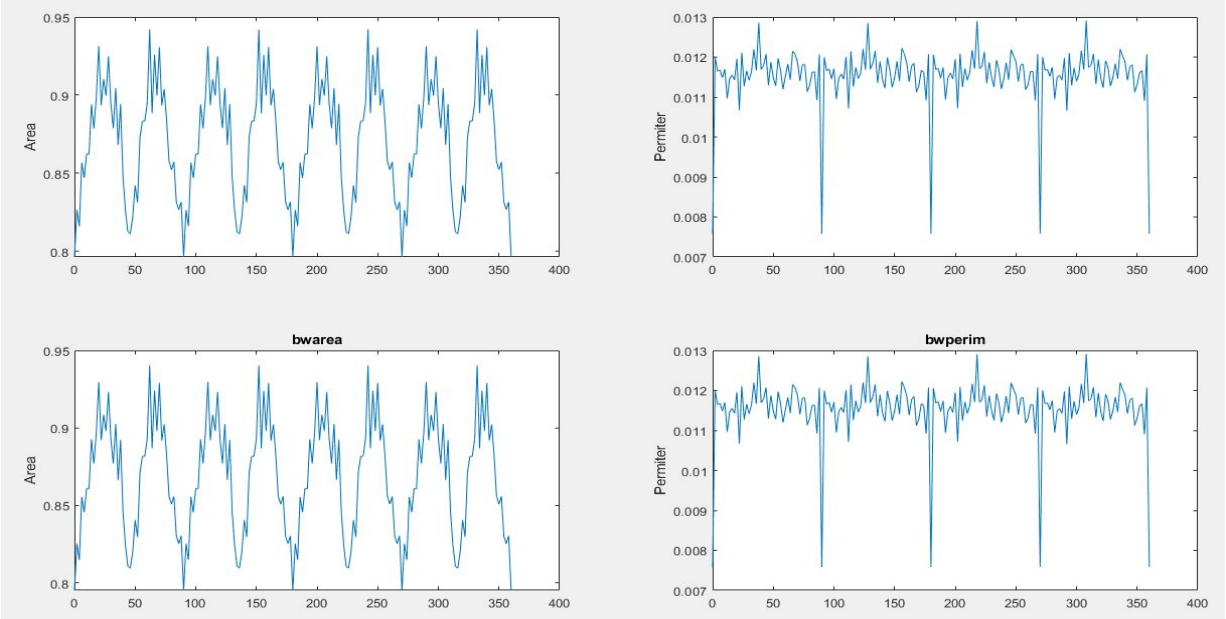
Code:

```
photo=imread('circle_1.bmp');
Area = area(photo);

se = strel('disk',1);
E = imerode(photo,se);

contour = (photo-E);
imshow(contour);

Perimeter = perimeter(contour);

circularity = Circularity(photo);
compactness = Perimeter^2/(4*pi*Area);

function Circular = Circularity(image)


% Centroid of object
props = regionprops(image, 'Centroid');
x = props.Centroid(1);
y = props.Centroid(2);

% Boundary
dim = size(image);
col = round(dim(2)/2)-90;
row = min(find(image(:,col)));
boundary = bwtraceboundary(image,[row, col], 'W');

distance = [];

%distance between centroid of object and boundary
for i=1:size(boundary,1)
      distance(i) = sqrt((x-boundary(i,1))^2 + (y-boundary(i,2))^2);
end

avrOfDistance = mean(distance);
stdOfDistance = std(distance);

Circular = (sqrt(3)*stdOfDistance)/avrOfDistance;
end


function count2 = perimeter(image)
   count2 = 0;
   for row = 1:size(image,1)
      for col = 1:size(image,2)
         if(image(row,col)>0)
            count2 = count2 + 1;
         end
      end
   end
end

function count1 = area(image)
   count1 = 0;
```

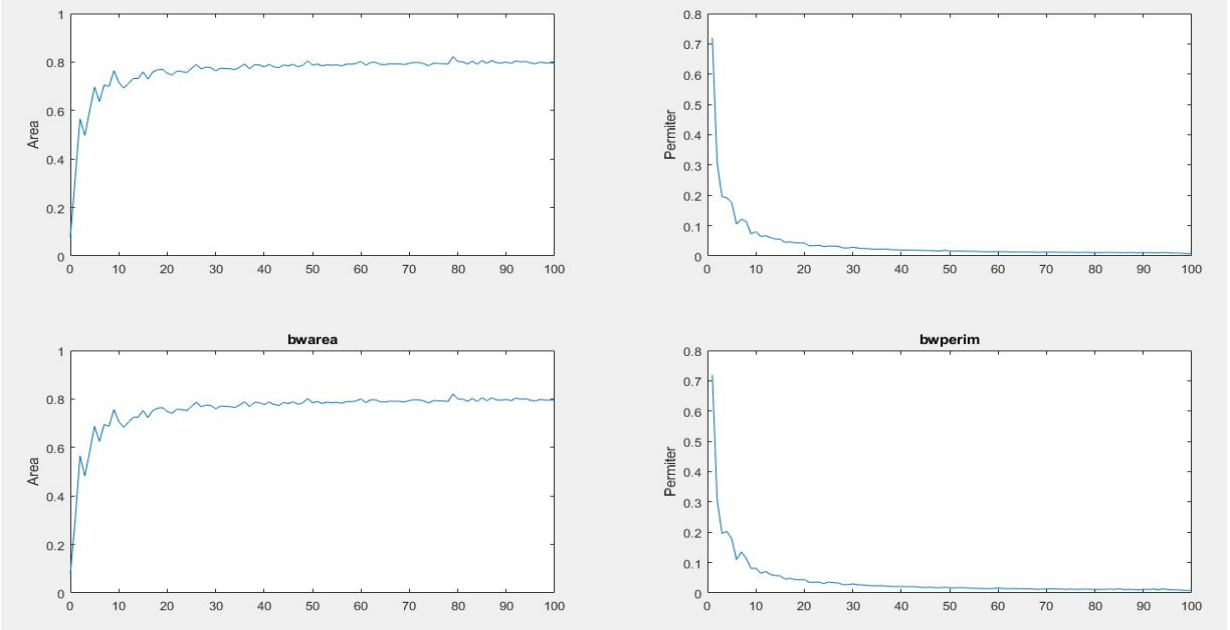
```
    for row = 1:size(image,1)
        for col = 1:size(image,2)
            if(image(row,col)>0)
                count1 = count1 + 1;
            end
        end
    end
end
```

2. Test if the compactness and circularity shape factors are RST-invariance (RST—Rotation, Scaling, Translation).

Rotation

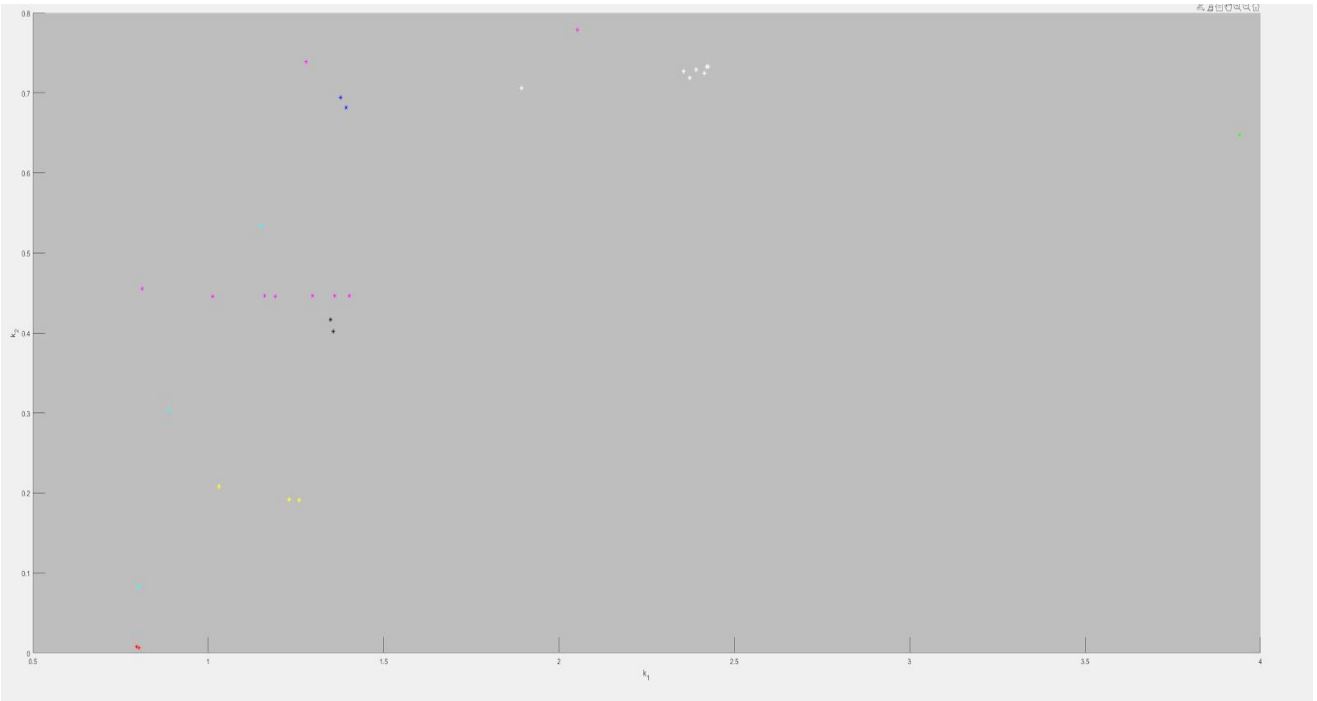


Scaling



We can see that compactness and circularity for rotation and scaling are not RSTinvariant.

3. Calculate compactness and circularity factors for objects in images from directory IMAGES_1&2. Use obtained values for creation of object feature plane (compactness/circularity), Find position of clusters which represent objects from this directory. Is it possible to recognize all objects in directories IMAGES_1&2 using above presented two shape factors? Find the maximal subset of objects that can be recognized with help of this factors? Have the number of holes in objects an influence on the value of factors?



L-white
trapeze - black
square - yellow
rectangle - magenta
ellipse - cyan
dumbell - blue
cross - green
circle - red

On the above feature plane we want to somehow classify/recognize objects by compactness and circularity factors. We can observe that some objects are really difficult to group, for example dumbbells are really hard to group, because they belong to other clusters (rectangles, squares and some how circle). But the easiest to cluster is the L figure due to that are really close to each other and does not have any outlier.

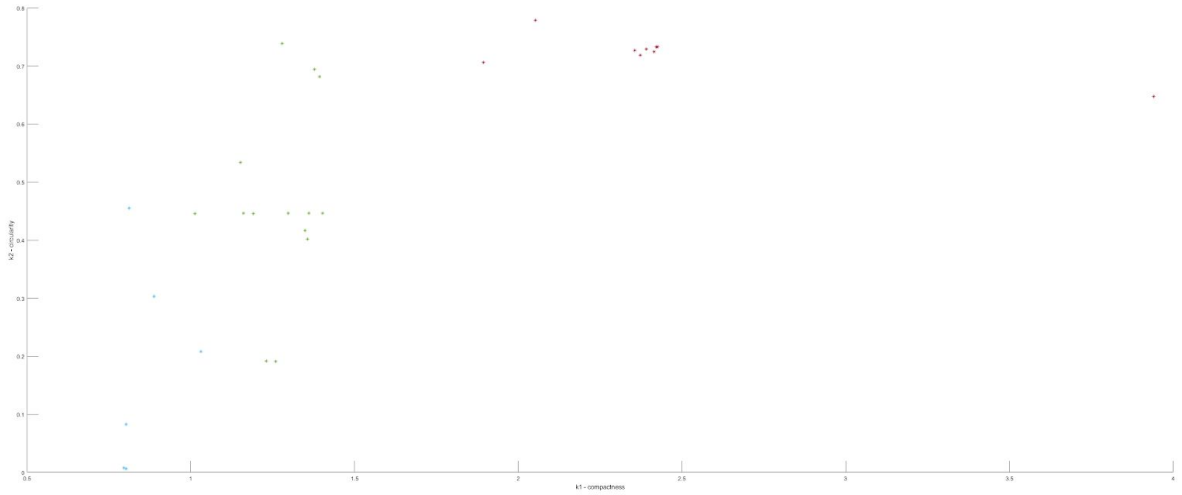
4. Propose and implement in MATLAB another shape factor that is also based on RST invariance.

I propose an aspect ratio. The algorithm is very simple, it measures the ratio of the object's height to its width.

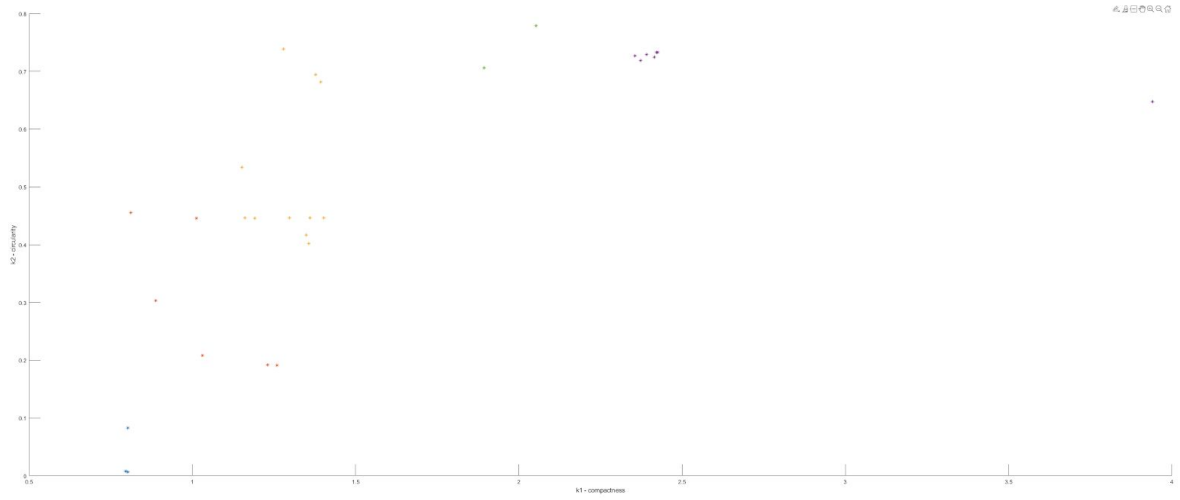
```
function aspect_ratio = aspectratio(countour)



    x_max = 0;
    x_min = size(countour,1);

    y_max = 0;
    y_min = size(countour,2);



    for row = 1:size(countour,1)
        for col = 1:size(countour,2)
            if(countour(row,col) == 1)
                if(row > y_max)
                    y_max = row;

                end
                if(row < y_min)
                    y_min = row;

                end
                if(col > x_max)
                    x_max = col;

                end
                if(col < x_min)
                    x_min = col;

                end
            end
        end
    end


    %gorizonatl and vertical of feret box
    out = [x_max-x_min, y_max-y_min];
    aspect_ratio = out(1)/out(2);
end
```

5. Implement k-means algorithm using MATLAB environment (see APPENDIX B). Is it possible to recognize all objects in IMAGES _1&2 using the k-means method of classification on the feature plane (compactness/circularity)? What are the advantages and disadvantages of the k-means clustering method?

for k = 3



for k = 5



Code:

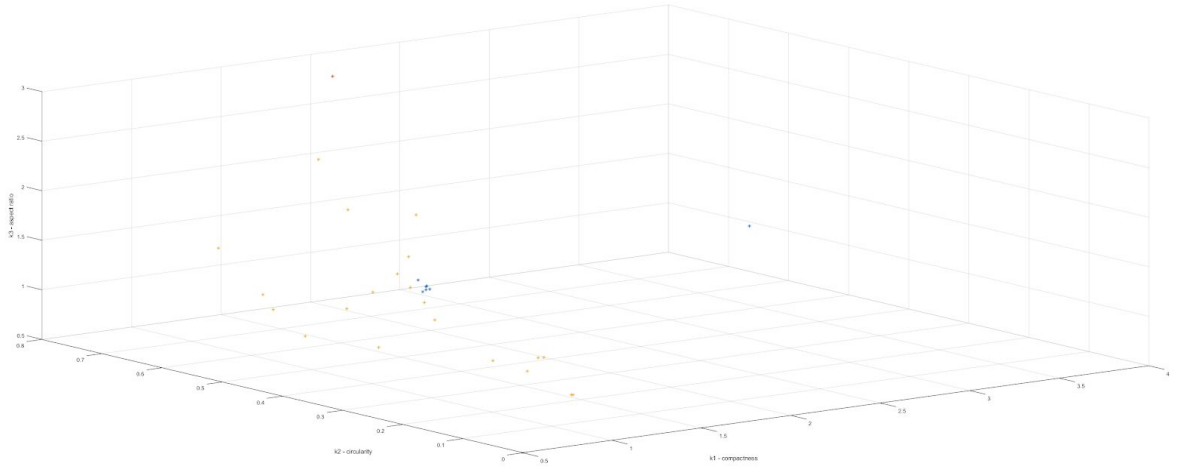
```
length = 1:29;

intervalk1 = 0:0.01:4;
intervalk2 = 0:0.01:0.5;

amount = 5;

for i=1:amount
    positionk1(i) = randsample(intervalk1, 1);
    positionk2(i) = randsample(intervalk2, 1);
    c(i,:) = [positionk1(i), positionk2(i)];
end
count = 0;
for it=1:200
```

```
    k = [k1(:), k2(:)];

    for j = 1:amount
        for i = 1:size(k,1)
            distance(i,j) = sqrt((c(j,1)-k(i,1))^2+(c(j,2)-k(i,2))^2);
        end
    end

    M = min(distance,[], 2);

    [a,b] = find(distance == min(distance,[], 2));
    pos = [a(:), b(:)];

    group = zeros(size(pos,2),size(pos,1));

    for j = 1:amount
        for i = 1:size(k,1)
            if(b(i)==j)
                group(j,i) = a(i);
            end
        end
    end

    transgroup = transpose(group);

    G = cell(1, amount);
    avg_k1k2 = [];

    for i = 1:amount
        temp = [];
        for j = 1:size(transgroup, 1)
            if(transgroup(j,i) ~= 0)

                temp(j,1:2) = k(transgroup(j,i),1:2);

            end
        end

        %Deleting zeros
        clean = ~all(temp, 2);
        temp(clean,:) = [];

        c(i,1:2) = mean(temp);


        G(i) = {temp};
    end
    count = count + 1;
end

hold on
for i = 1:amount
    if ~isempty(G{i})
        scatter(G{i}(:,1), G{i}(:,2), '*');
        set(gcf,'color','w');
    end
end
xlabel('k1 - compactness')
ylabel('k2 - circularity')
hold off
```

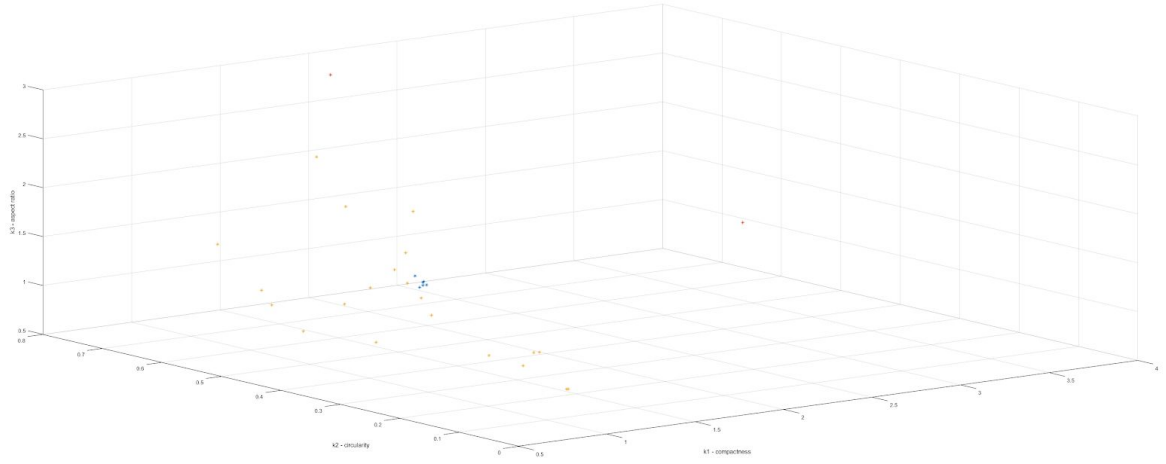
The advantage of this method is simplicity in implementation, which causes fast calculation. The drawback of this method is to take into account the outlier of the object to some group. To compensate for the drawback we can increase the number of k which causes more groups which is simpler to classify objects.

6. How will the results change if we use an additional shape factor from task 4? Compare the results with results from previous points.

for k=3



for k =5



Code:

```
length = 1:29;

intervalk1 = 0:0.01:4;
intervalk2 = 0:0.01:0.5;
intervalk3 = 0:0.01:1.5;
amount = 5;
for i=1:amount
   positionk1(i) = randsample(intervalk1, 1);
   positionk2(i) = randsample(intervalk2, 1);
   positionk3(i) = randsample(intervalk3, 1);
   c(i,:) = [positionk1(i), positionk2(i), positionk3(i)];
end
```

```
count = 0;
for it=1:200

    k = [k1(:), k2(:), k3(:)];

    for j = 1:amount
        for i = 1:size(k,1)
            distance(i,j) = sqrt((c(j,1)-k(i,1))^2+(c(j,2)-k(i,2))^2+(c(j,3)-k(i,3))^2);
        end
    end

    M = min(distance,[], 2);


    [a,b] = find(distance == min(distance,[], 2));
    pos = [a(:), b(:)];


    group = zeros(size(pos,2),size(pos,1));

    for j = 1:amount
        for i = 1:size(k,1)
            if(b(i)==j)
                group(j,i) = a(i);
            end
        end
    end

    transgroup = transpose(group);


    G = cell(1, amount);
    avg_k1k2 = [];

    for i = 1:amount
        temp = [];
        for j = 1:size(transgroup, 1)
            if(transgroup(j,i) ~= 0)

                temp(j,1:3) = k(transgroup(j,i),1:3);

            end
        end

        %Deleting zeros
        clean = ~all(temp, 2);
        temp(clean,:) = [];

        c(i,1:3) = mean(temp);


        G(i) = {temp};
    end
    count = count + 1;
end

view(3);
hold on
```

```
grid on
for i = 1:amount
   if ~isempty(G{i})
      scatter3(G{i}(:,1), G{i}(:,2), G{i}(:,3), '*');
      set(gcf,'color','w');
   end
end
hold off
xlabel('k1 - compactness');
ylabel('k2 - circularity');
zlabel('k3 - aspect ratio');
```

We can see that the results are slightly different than in two dimension metrics. The main difference is additional metric (aspect ratio) which gives more additional information about the object. Which causes better classification objects to groups.

7. Propose and implement an algorithm for classification objects with holes.

My source of knowledge:link

The idea is that we have three matices

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

These above matrices represent the edge of the picture. Comparing the image to the matrices and calculating how many times it appears we have sufficient information to calculate euler formula. This can be used to calculate how many holes are in the image. $euler = m1 - m2 + m3$.

Code:

```
photo=imread('square_2withholes.bmp');
count1 = 0;
count2 = 0;
count3 = 0;
m1 = [0 0; 0 1];
m2 = [0 1; 1 1];
m3 = [1 0; 0 1];

for i = 1:(size(photo,1)-1)
   for j = 1:(size(photo,2)-1)
      if (isequal(photo(i:(i+1), j:(j+1)), m1))
         count1 = count1 + 1;
      end
      if (isequal(photo(i:(i+1), j:(j+1)), m2))
         count2 = count2 + 1;
      end
      if (isequal(photo(i:(i+1), j:(j+1)), m3))
         count3 = count3 + 1;
      end
   end
end
euler = abs(count1-count2+count3);
```