CSE 585/EE 555: Digital Image Processing II
Computer Project #2
Homotopic Skeletonization and Shape Analysis

Pavan Gurudath, Ming-Ju Li

February 2018

---

1 Objectives

This project aims to familiarize certain concepts of digital image processing such as skeletonization, granulometry, pattern spectrum (pecstrum), shape complexity amongst other filters that we have learnt in class. The main objectives of this project include the following:

- To perform the function of thinning to determine the skeleton of the image
- To implement shape analysis for two sets of images

2 Methods

This project can be subdivided into two portions namely, skeletonization and shape analysis. Therefore there are three functions that have to be run in order to obtain the results of this project. The first one is called $main\_skele.m$ which is for the first question while the other two are $main\_shape2a.m$ and $main\_shape2b.m$. Section 2.1 details the methodology used to obtain the conclusion of homotopisc skeletonization while section 2.2 refers to the methodology of Shape Analysis.

2.1 Homotopic Skeletonization

In order to perform this objectives, two input images named $penn256.gif$ and $bear.gif$ have been utilized. When the $main_skele.m$ function is executed, it calls the functions $erosion.m$ which performs the function of eroding the input image using the structuring element that have been passed.
NOTE: Please run main_skele.m to obtain the results
For homotopic skeletonization, the following algorithm has been implemented.

- The two images are read from the working directory.

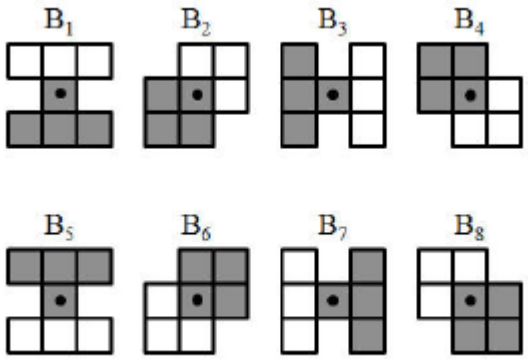


Figure 1: Structuring elements B_i

- Figure 1 refers to the set of structuring elements that have been utilized in order to perform the necessary functions.
  Here, for each structuring element, the center pixel (2,2) has been used as the origin where every shaded pixel is defined to be 0 black and the unshaded pixels are defined as 1 white on MATLAB. The undefined pixels are set as NaN.
  Therefore B_1 is equivalent to

1	1	1
NaN	0	NaN
0	0	0

- The morphological skeleton algorithm $sk(X)$ as given in L6-1 is implemented

$$sk(X) = \bigcup_{0<=n<=N} S_n(X) \tag{1}$$

  where $S_n(X) = (X \ominus nB) - (X \ominus nB)_B$.

- To obtain this on MATLAB, we perform *hit or miss* transform as shown in Equation 2 and pass that result to compute the difference (Equation 3). This would result in one layer being stripped off from the object.

$$X \circledast B_i = (X - B_i^f) \cap (X - B_i^b) \tag{2}$$

  where B_i^f, B_i^b is a set of two structuring elements and $X \circledast B_i$ the hit-or-miss transform.

$$X \bigcirc B_i = X - X \circledast B_i \tag{3}$$

- The above step is continuously executed until no more layers can be stripped off. This is achieved by counting the number of white pixels after every iteration. We stop the iteration when the number of white pixels remain the same between consecutive layers.
- The results are then obtained and are as shown in Section 3.

2.2 Shape Analysis

In order to perform this objective, four input images named $match1.gif$, $match3.gif$, $shadow1.gif$ and $shadow1rotated.gif$ have been utilised. When the $main\_shape2a.m$ and $main\_shape2b$ function are executed, they call the functions $erosion\_amg.m$, $dilation\_amg.m$, $area\_func.m$, $complexity\_func.m$, $sym\_matrix.m$, $pecstrum.m$ and $pecstral\_analysis$. These functions perform the tasks:

$erosion\_amg.m$	Function of eroding input image with given structural element
$dilation\_amg.m$	Function of dilating input image with given structural element
$area\_func.m$	Calculates the area of the input image number of white pixels
$complexity\_func.m$	Calculates teh measure of shape complexity
$sym\_matrix.m$	Function for outputting a symmetric matrix
$pecstrum.m$	Function for calculating the pattern spectrum of of shape X
$pecstral\_analysis$	Function of pattern recognition

The inbuilt function regionprops has been used to obtain the bounding box around every connected component.
NOTE: Please run main_shape2a.m and main_shape2b to obtain the results

For shape analysis, the following algorithm has been implemented.

- We isolate every connected label by drawing a bounding box around each of the connected component in the input figure. In the following methodology, the connected component image being analyzed will be referred to as X.
- Granulometry of X:
  We compute the size distribution of object X using equation 4.

$$U(r) = m(X_{rB}), \quad r \epsilon \Re \tag{4}$$

  where X_{rB} is the opening of X by rB and $m(X_{rB})$ is the area of X_{rB} This is implemented in the MATLAB function $area\_func.m$ and the main function.
- Pattern Spectrum (pecstrum) of X:
  We compute pattern spectrum using the equation 5

$$f(r) = \frac{-\frac{du(r)}{dr}}{m(X)} \quad \forall r > 0 \tag{5}$$

  This is implemented in the MATLAB function $pecstrum.m$

- Complexity of X:
  The complexity of the shape is calculated using equation 6. This is implemented in the MATLAB function *complexity_func.m*

$$H(X|B) = -\left[\Sigma_{i=0}^{N} f(i) log f(i)\right] \tag{6}$$

  where $f(i)$ is the pattern spectrum of X. This is also known as the entropy and higher the entropy, higher is the complexity of the image.

- Pattern Matching:
  The distance between the object that is being tested and all the reference objects are calculated. The least distance amongst them is relates the mapping of test and reference object. The distance is calculated using the equation 7.

$$d_i = \left[\Sigma_{i=0}^{N} C_n (f(n) - f_{R_i}(n))^2\right]^{1/2} \tag{7}$$

  where C_n are the weights to emphasise various components. These weights are chosen randomly and by hit and trial they're improved. Therefore as C_n (weights) change, the matching results may change as well. The reference object that minimises the distance is the best match for that test object.

The shape analyis algorithm is used for two cases.
In the first case, the two figures match1.gif and match3.gif have been used in order to obtain the matching of the spade between the two images. There are four objects in each of these images. First, the objects in match1.gif are taken using the bounding box method as discussed above and all the necessary calculations are done. The steps of steps of computing the granulometry, pecstrum and complexity is carried out. Next, the same procedure is followed for the four objects of match3.gif. Using the results obtained in these procedures, the matching is done by calculating the distances. The results are shown in section 3.2.1.
In the second case, the two figures shadow1.gif and shadow1rotated.gif are used. The same set of procedures are followed as in the first case except that there are 8 objects in this case. The pattern matching of the four shadowed cases are executed. The results are shown in section 3.2.2.

The following flowchart summarises the methodology that has been used. Flowchart 2 refers to the question 1 of homotopic skeletonization, while flowchart 3(a) and 3(b) refer to the question 2a and question 2b respectively from the problem statement.

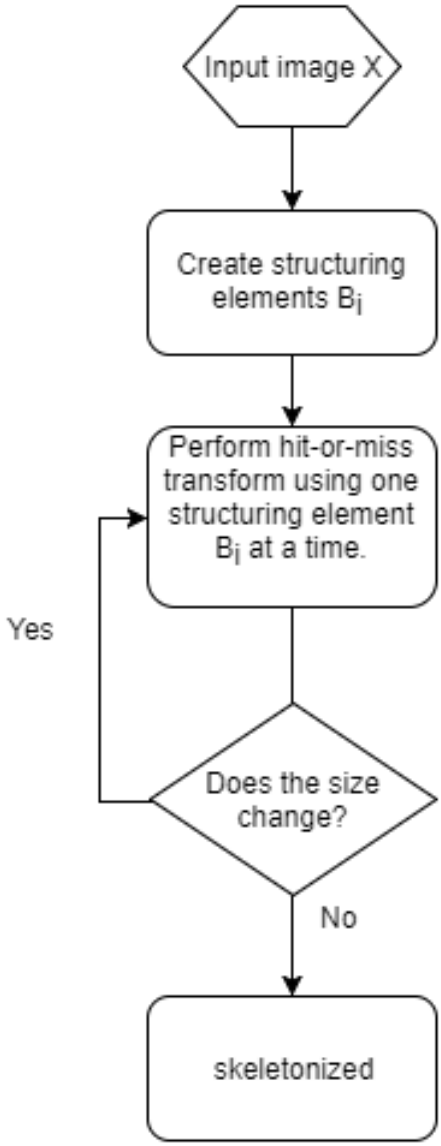
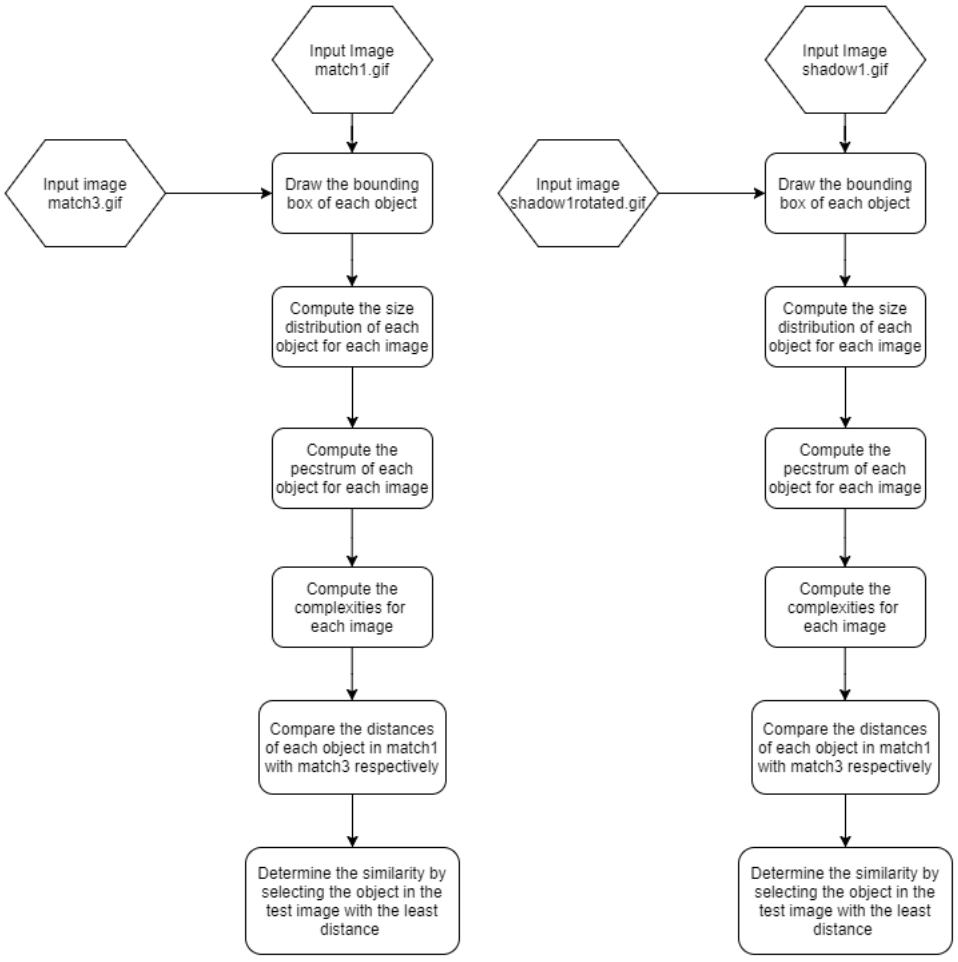


Figure 2: Homotopic Skeletonization



(a) Match1 and Match3 (b) Shadow1 and Shadow1Rotated

Figure 3: Shape Analysis

3 Results

3.1 Homotopic Skeletonization

To implement homotopic skeletonization, two input images penn256.gif and bear.gif are considered. These are shown in Figure 4 and Figure 5. The program prompts the user to input 1 if you'd like to work with the penn256.gif image and press 2 if bear.gif. The image that is selected undergoes a series of hit and miss transform using the structuring elements as shown in Figure 1. The penn256.gif image takes 8 iterations in order to obtain the skeleton of the image, while bear.gif takes 22 iterations. This is justified since the after every iteration only one layer is stripped off and the bear.gif contains a more dense image than penn256.gif and therefore it takes more iterations and thereby takes more time. The image in figure 4 takes 11.8593 seconds to obtain the skeleton as shown in 9. This time does not include the loading of the structuring element and writing of images onto the working directory. Similarly, the image in figure 5 takes 34.622 seconds to obtain the skeleton as shown in 15.

Figure 4: penn256.gif input image

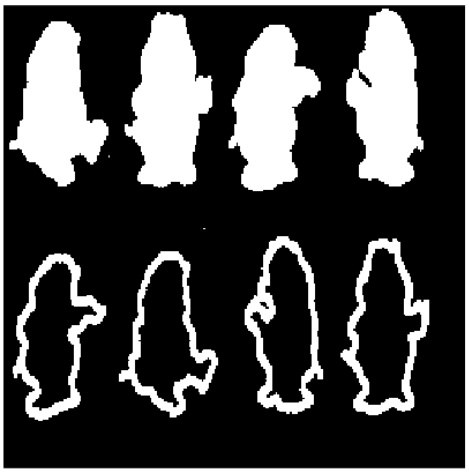


Figure 5: bear.gif input image

The program saves the image for every iteration on the working directory. This report consists of the images for only a few iterations. Section **??** contains the results that are obtained during the iterations and the final image while section **??** contains the results associated with iterations 2, 5, 10, 20 and the final image. Their respective superimposed image are also reported.

3.1.1 Penn256

Figure 6 through Figure 8 showcases the images during the iteration 2, 4 and 6 which are respectively X_2, X_4 and X_6. While the question asks for X_10, our program terminates at the eighth iteration.

Figure 6: X_2 of penn256 (Iteration 2)

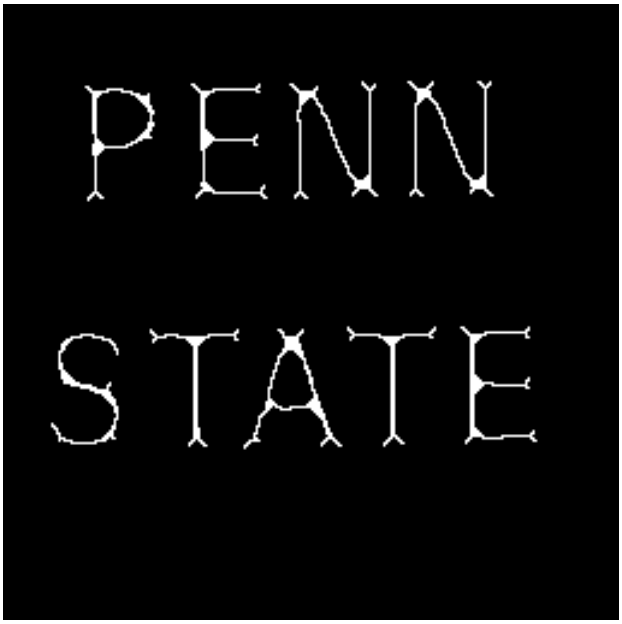


Figure 7: X_4 of penn256 (Iteration 4)

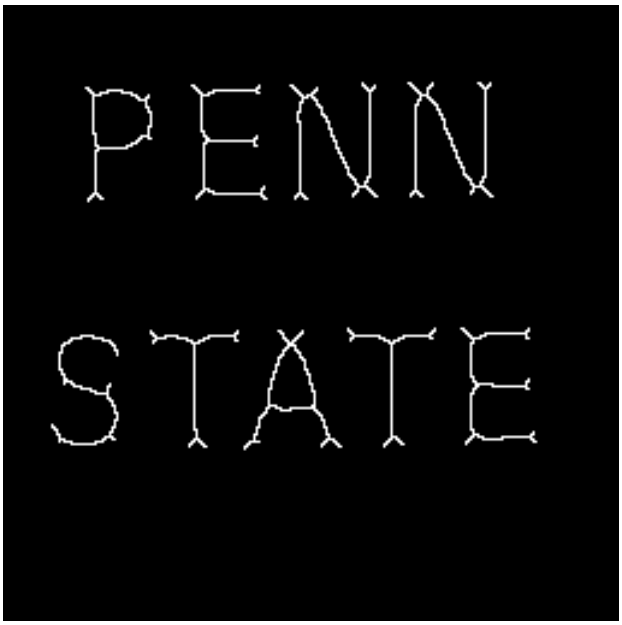


Figure 8: X_6 of penn256 (Iteration 6)

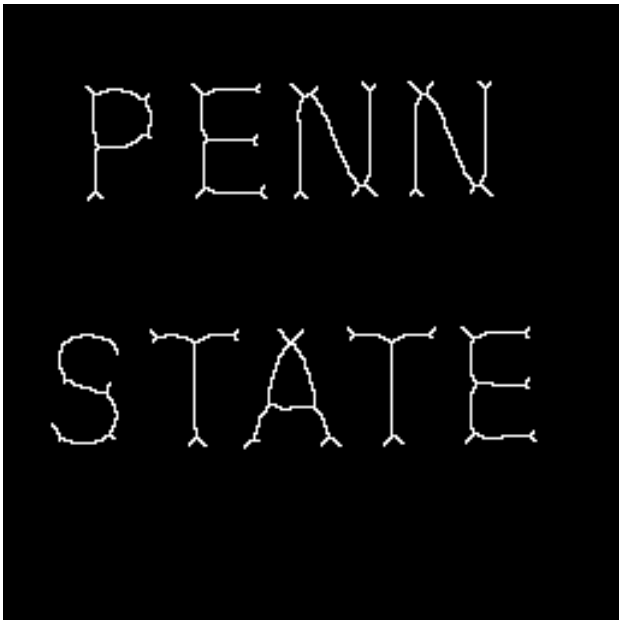


Figure 9: Final skeleton of penn256 (X_8)

The final thinned skeleton image and the superimposed images are as shown in Figure 9 and Figure 10. It can be seen from Figure 6 through Figure 9 that the skeleton becomes thinner and thinner.

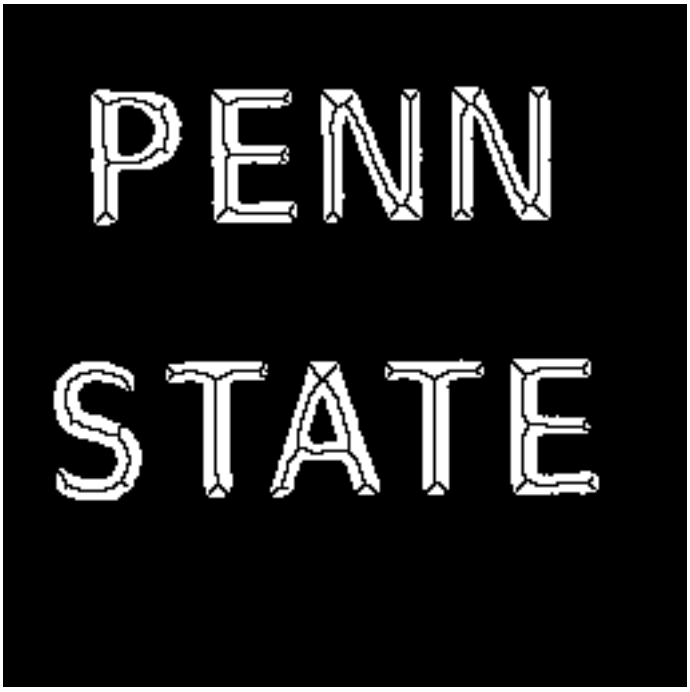


Figure 10: Homotopic skeleton superimposed on original image penn256.gif

3.1.2 Bear

Figure 10 through Figure 13 showcases the images during the iteration 2, 5, 10 and 20 which are respectively X_2, X_5, X_10 and X_20.

The final thinned skeleton image and the superimposed images are as shown in Figure 9 and Figure 10. It can be seen from Figure 10 through Figure 15 that the skeleton becomes thinner and thinner.

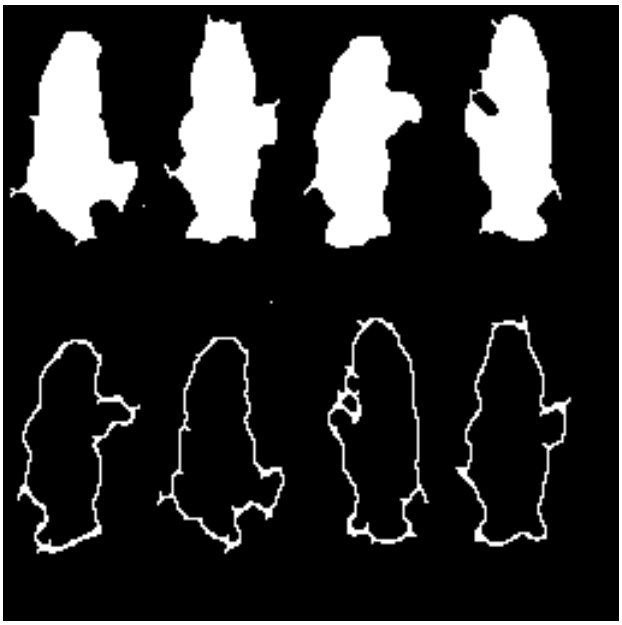


Figure 11: X_2 of bear (Iteration 2)

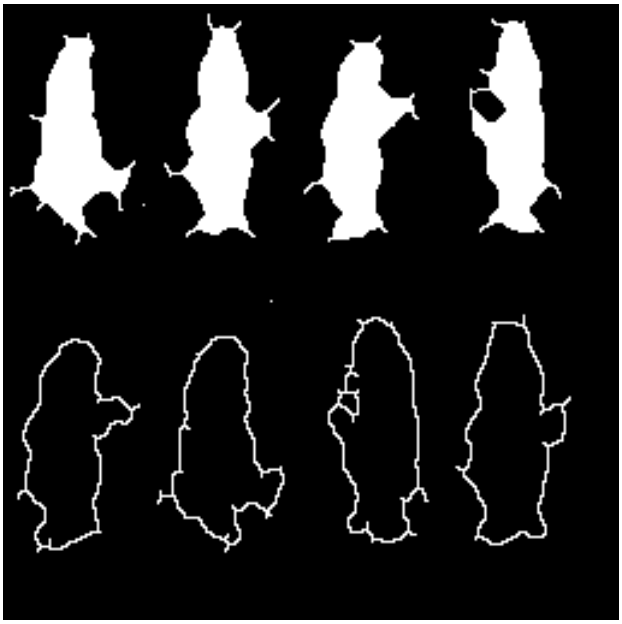


Figure 12: X_5 of bear (Iteration 5)

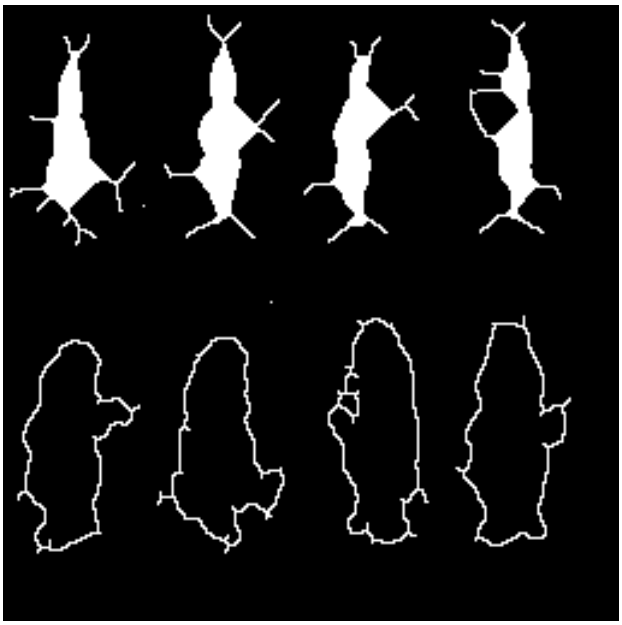


Figure 13: $X_1 0$ of bear (Iteration 10)

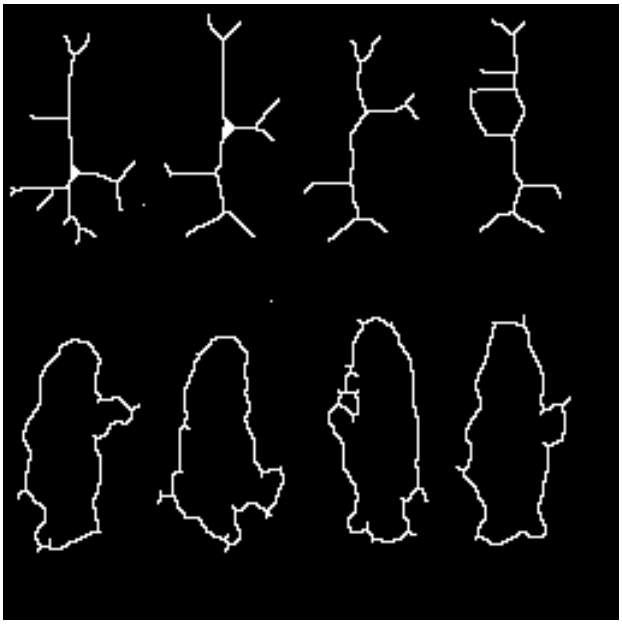


Figure 14: $X_2 0$ of bear (Iteration 20)

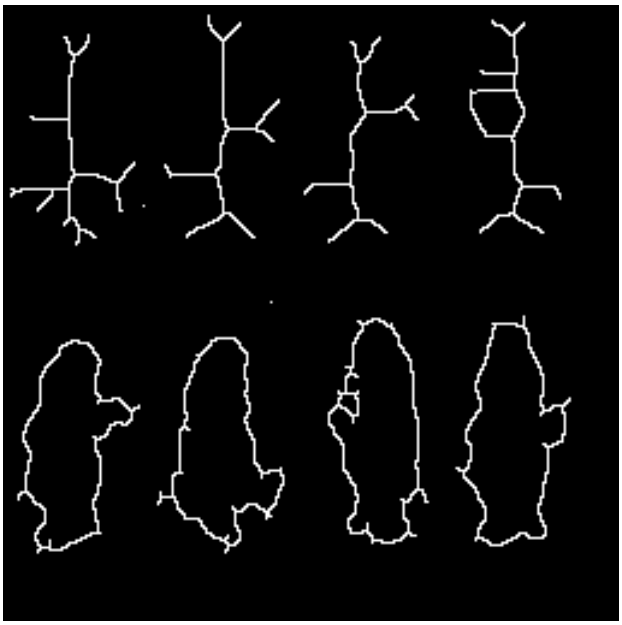


Figure 15: Final skeleton of penn256 ($X_2 2$)

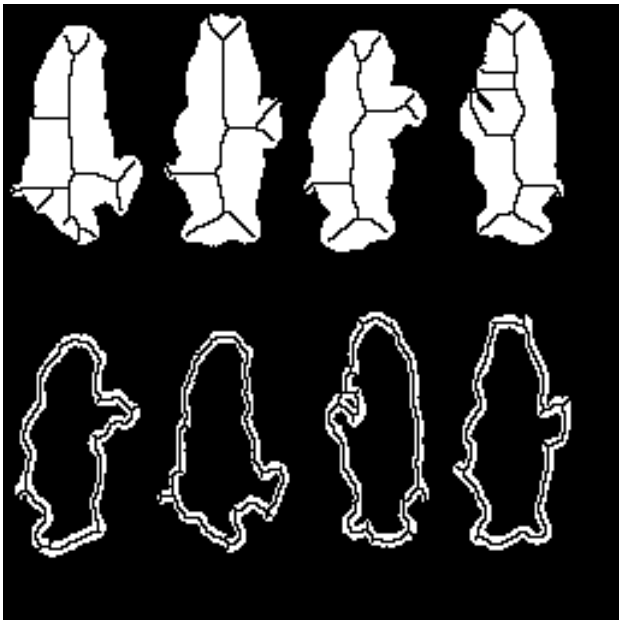


Figure 16: Homotopic skeleton superimposed on original image bear.gif

3.2 Shape Analysis

3.2.1 Match1 and Match3

These results are obtained after running the file *main_shape2a*. The input images are as shown in Fig 17 and 18. It can be seen that Fig 17 and Fig 18 are the same images in terms of the objects present however the objects contained in Fig 18 have been rotated and their sizes have been changed (shrunk/stretched). Figures 19(a) through 19(d) shows the bounding box on each object namely flower, bull, aeroplane and spade respectively that have been applied and these bounded objects have been calculate the necessary functions. Similarly figure 20 represents the bounding box around the objects in figure 18

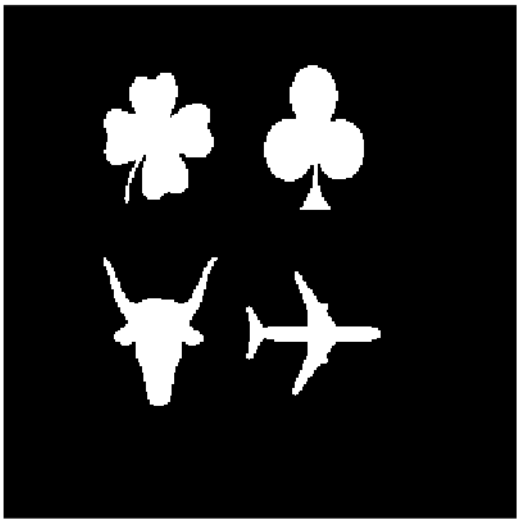


Figure 17: Match1.gif input image

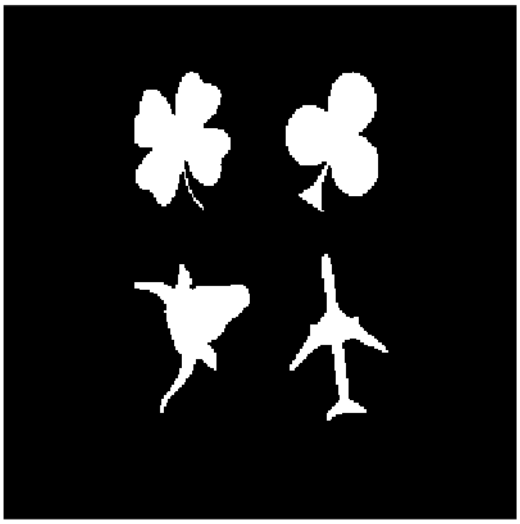
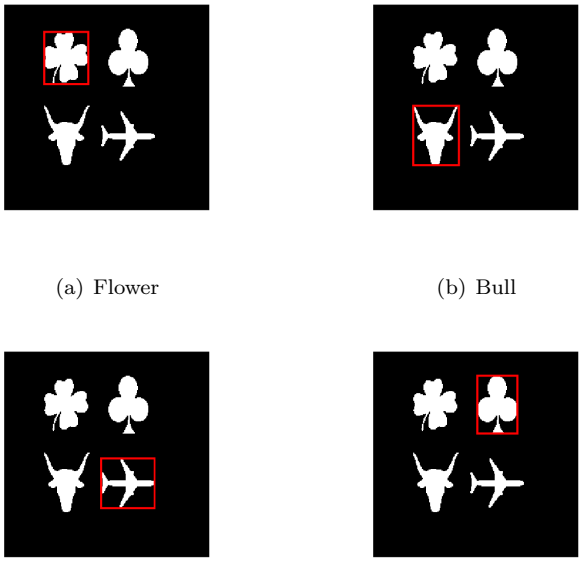


Figure 18: Match3.gif input image

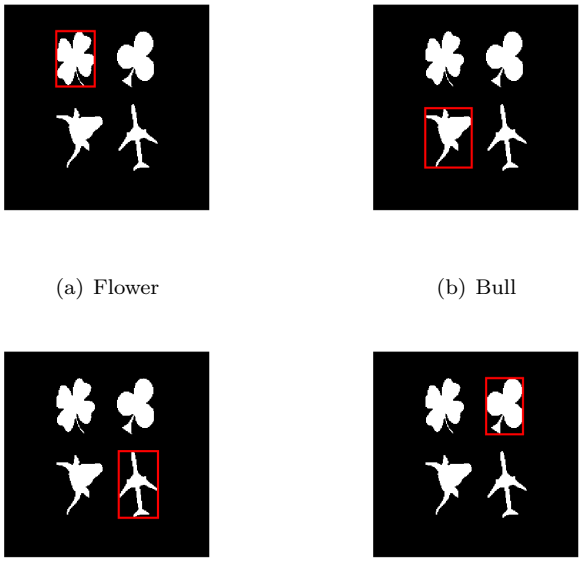
After extracting these individual objects, the size distribution and pattern spectrum is calculated. The following figures showcase the size distribution along with their pattern spectrum. Figure 36 through Figure **??** shows the results for flower, bull, airplane and spade respectively. The complexity of each object is calculated and the maximum and minimum complexity has been printed on the MATLAB command window. A screen shot of it has been shown in Figure 25.



(a) Flower (b) Bull

(c) Airplane (d) Spade

Figure 19: Bounding box on individual objects



(a) Flower (b) Bull

(c) Airplane (d) Spade

Figure 20: Bounding box on individual objects

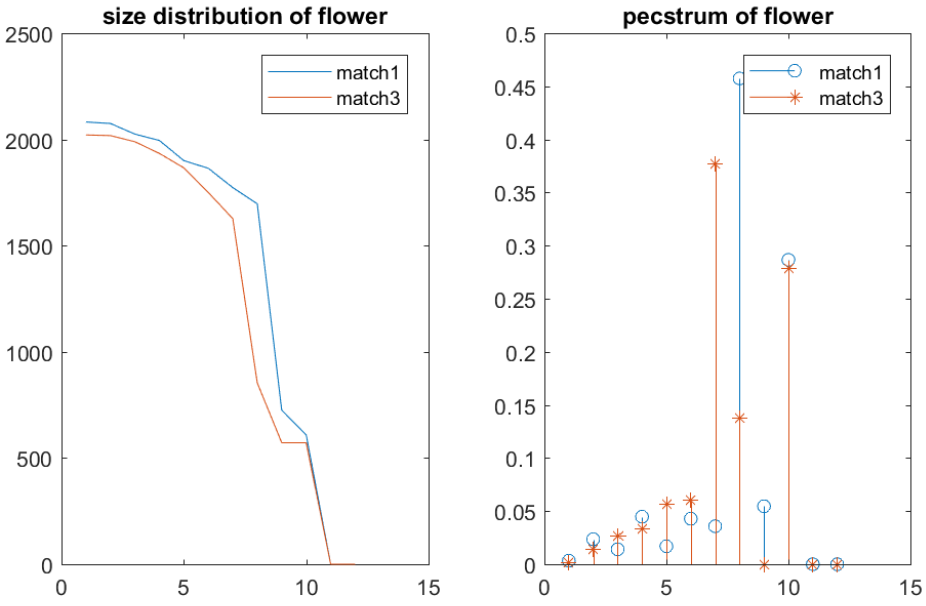


Figure 21: Granulometry and pecstrum of Flower

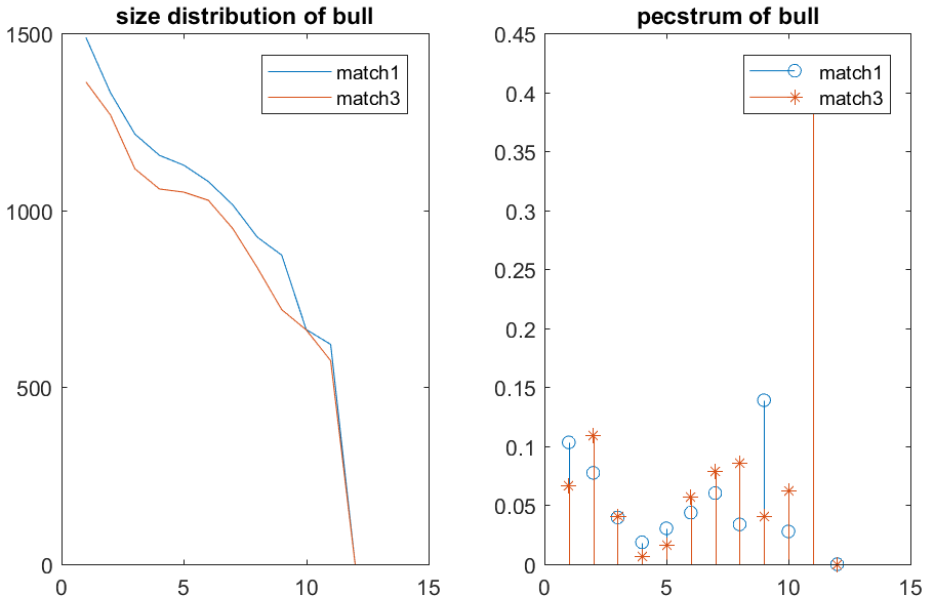


Figure 22: Granulometry and pecstrum of Bull

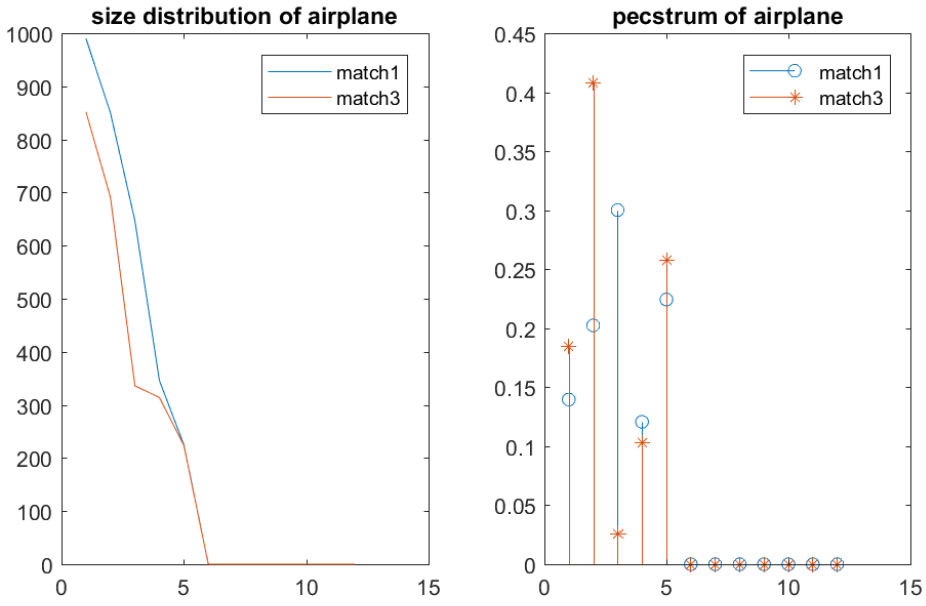


Figure 23: Granulometry and pecstrum of Airplane

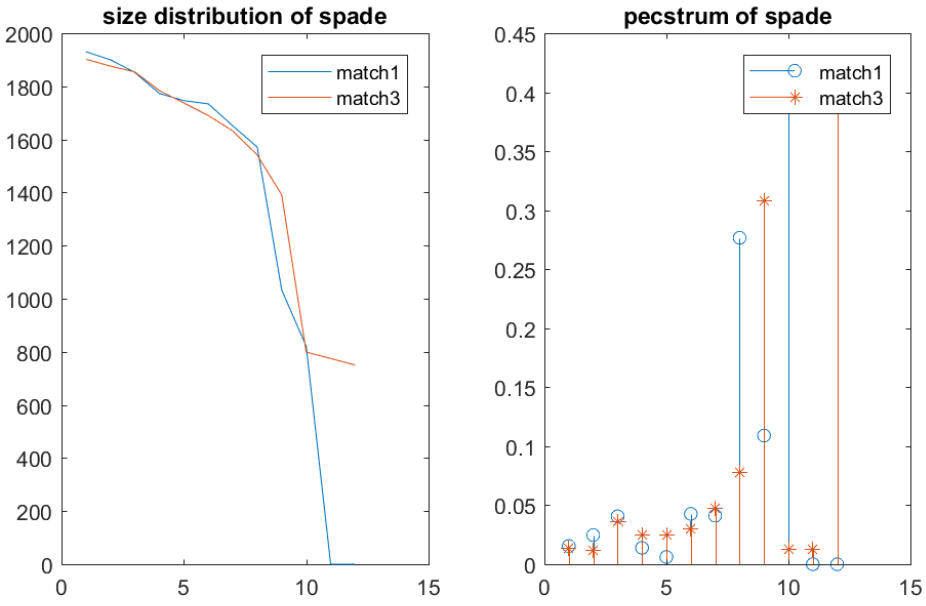


Figure 24: Granulometry and pecstrum of Spade

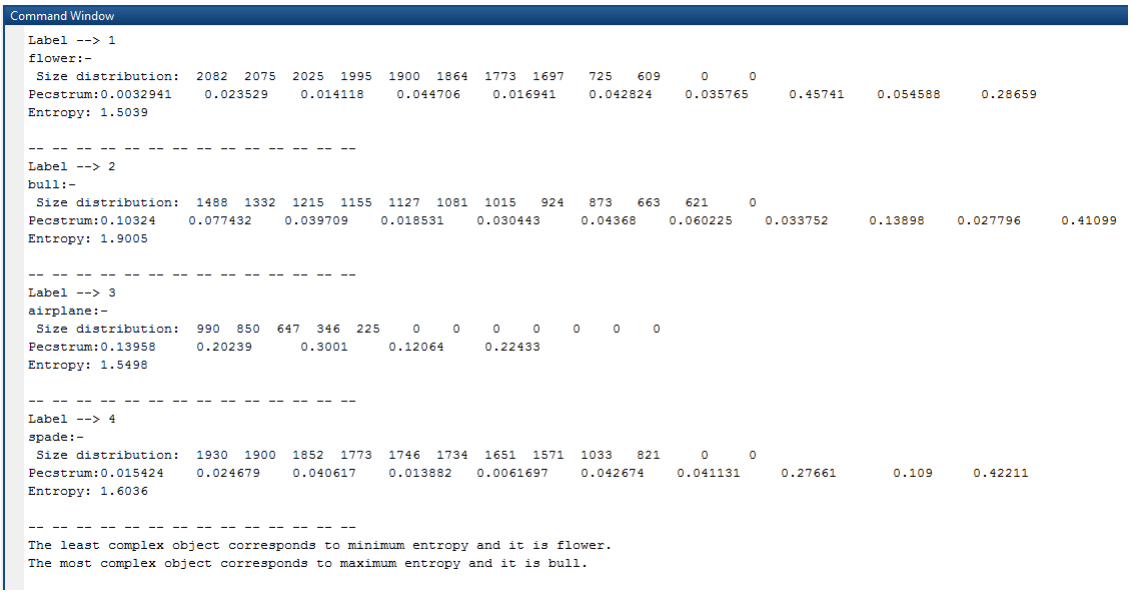
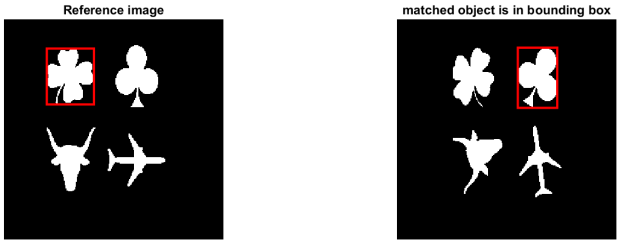


Figure 25: Granulometry and pecstrum on command window

While the objective of the question was to match spade from Figure 17 to the spade from Figure 18, our algorithm can match 3 of the objects with their respective images and have been as shown in figure 30 through 33. The only object that doesnt match accordingly is the flower. This makes sense since the flower looks very similar to the spade. However, if the weights C_n were to be changed, then a combination could be so achieved that would match all the images. One of the future work could possibly be to train this using a neural network to obtain the ideal C_n.
The complexities of match1 and match3 are shown in table 1.

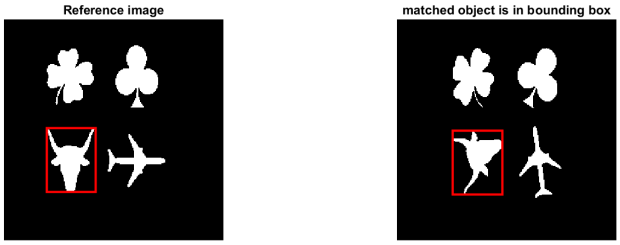
Objects	Entropy_Match1	Entropy_Match3
Flower	1.5039	1.6071
Bull	1.9005	1.8949
Airplane	1.5498	1.3546
Spade	1.6036	1.7021

Table 1: Complexity of objects from Match1 and Match3



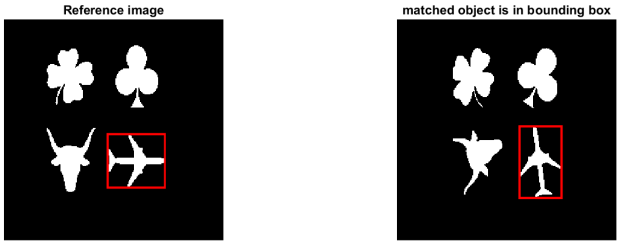
(a) Reference Image (b) Matched Image

Figure 26: Pattern Matching : Flower



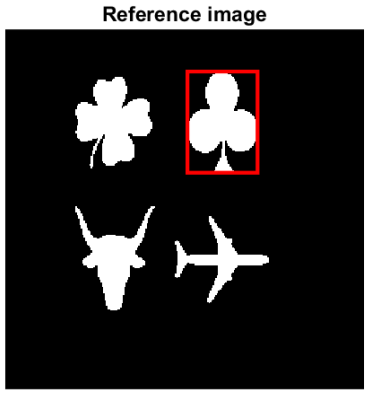
(a) Reference Image (b) Matched Image

Figure 27: Pattern Matching : Bull

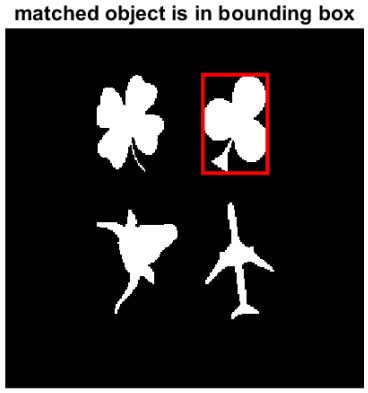


(a) Reference Image (b) Matched Image

Figure 28: Pattern Matching : Airplane



(a) Reference Image



(b) Matched Image

Figure 29: Pattern Matching : Spade

3.2.2 Shadow1 and Shadow1rotated

The same set of functions have been carried out as in Section 3.2.1. The figures have in shadow1 from left-to-right and top-to-bottom are labelled as outline_1, outline_2, outline_3, outline_4, shaded_1, shaded_2, shaded_3, shaded_4. The objective of this question was to map the contents of shaded_1, shaded_2, shaded_3 and shaded_4 from Figure 30 to 31 where Figure 30 refers to the shadow1.gif and Figure 31 refers to shadow1rotated.gif. The entropy of the objects from both the images are as shown in Table 2. Intuitively, the complex objects would be shaded_2 for both the input images since the others can be interpreted as a thick vertical line with a few irreuglarities while this object has a certain curvature throughout and seems to be more complex to draw using our hand. The results are backed up by the Table 2.

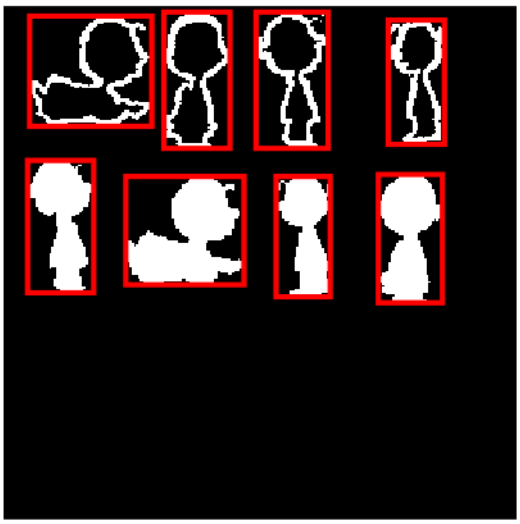


Figure 30: Shadow1.gif input image

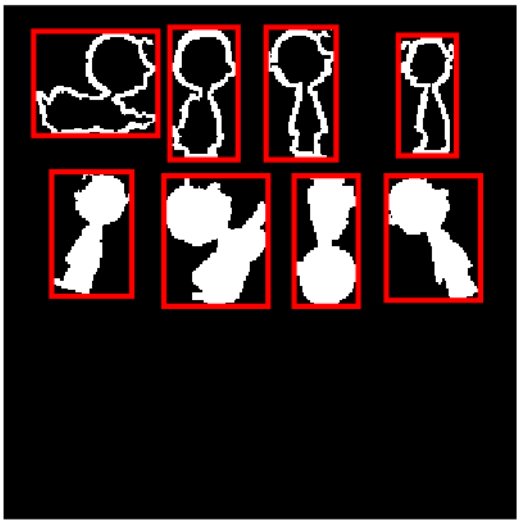


Figure 31: Shadow2.gif input image

Objects	Entropy_Shadow1	Entropy_Shadow1rotated
outline_1	0.2777	0.3459
outline_2	0.4203	0.4623
outline_3	0.1698	0.2069
outline_4	0.4460	0.3844
shaded_1	1.8511	1.6002
shaded_2	1.9982	2.0798
shaded_3	1.8322	1.9752
shaded_4	1.9962	1.8032

Table 2: Caption

The size distribution and pecstrum of the lables shaded_1 to shaded_4 are as shown in figures 32 through 35. Figure **??** through Figure **??** show the matching of individual objects. We got a matching of 75%. By changing the weights, the percentage can be made 100.

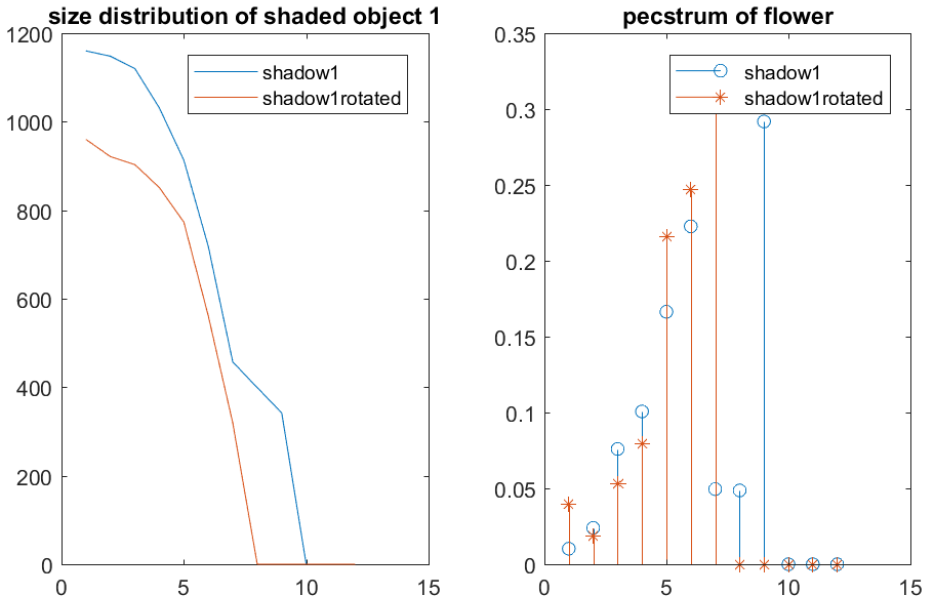


Figure 32: Granulometry and pecstrum of shaded_1

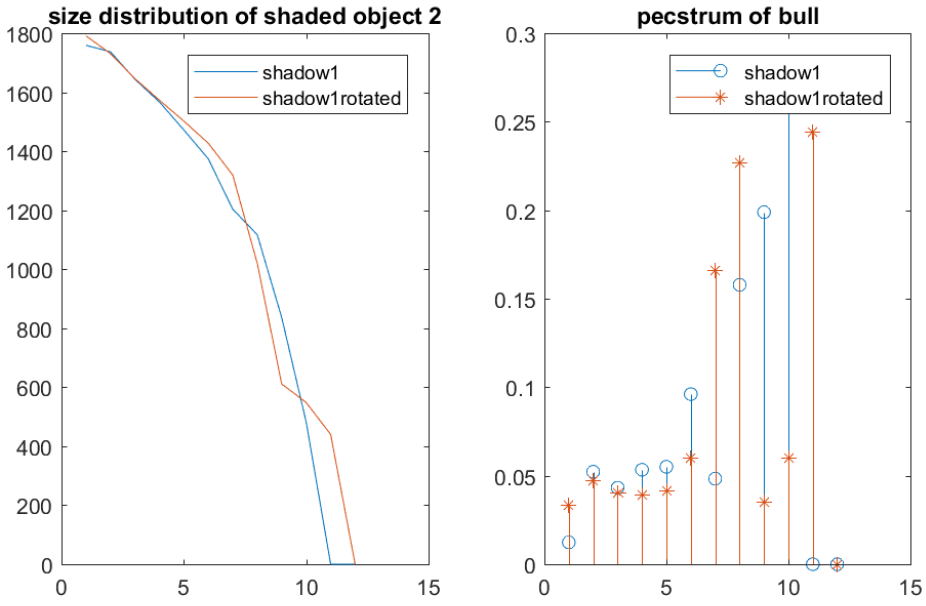


Figure 33: Granulometry and pecstrum of shaded_2

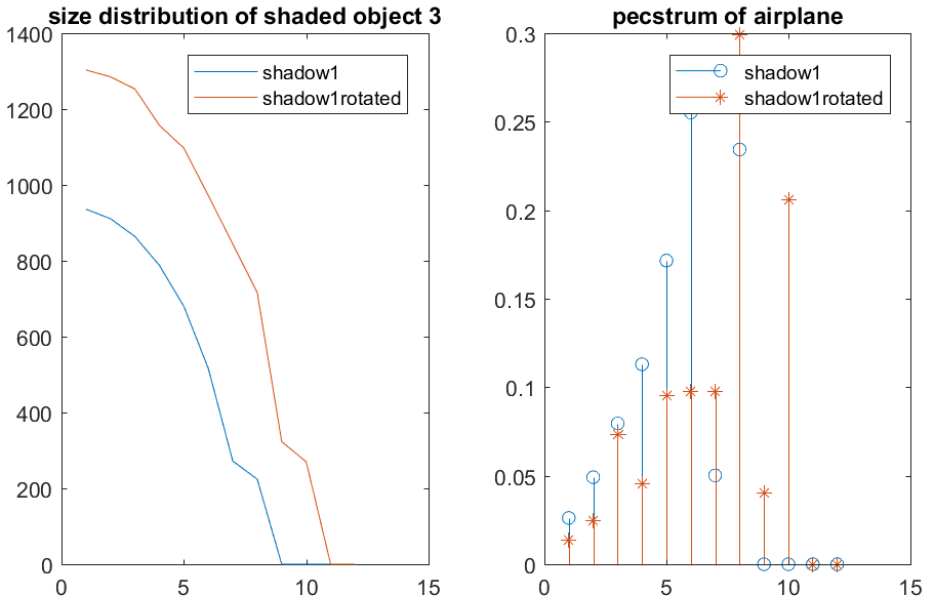


Figure 34: Granulometry and pecstrum of shaded_3

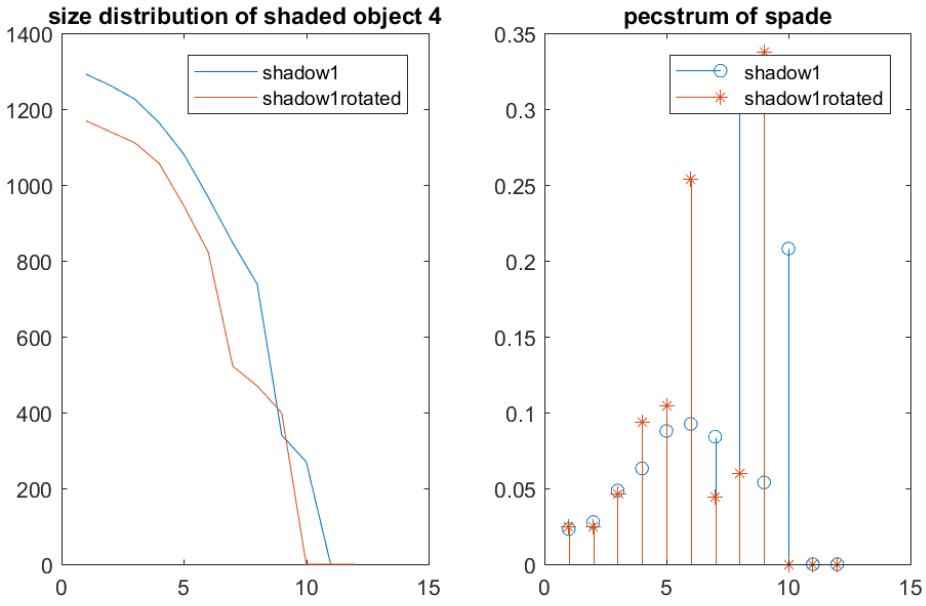
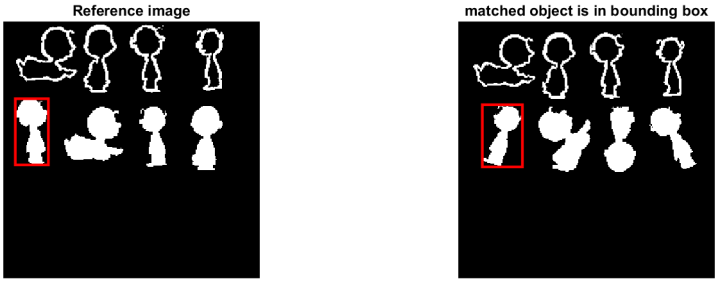
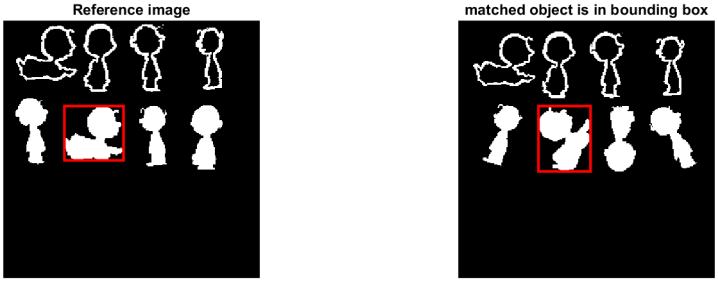


Figure 35: Granulometry and pecstrum of shaded_4



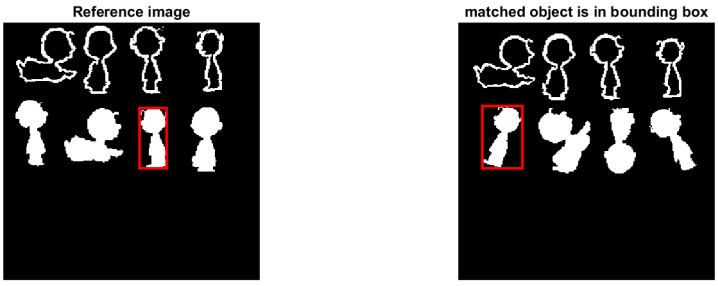
(a) Reference Image (b) Matched Image

Figure 36: Pattern Matching : Shaded_1



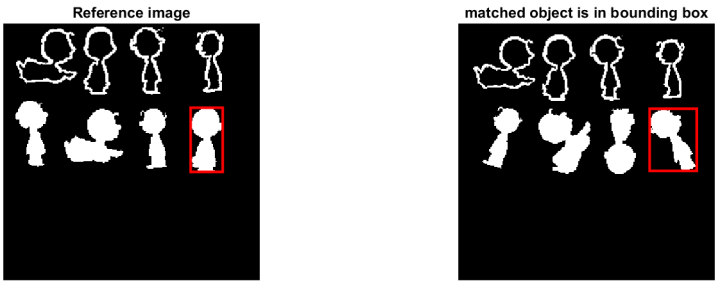
(a) Reference Image (b) Matched Image

Figure 37: Pattern Matching : Shaded_2



(a) Reference Image (b) Matched Image

Figure 38: Pattern Matching : Shaded_3



(a) Reference Image (b) Matched Image

Figure 39: Pattern Matching : Shaded_4

4 Conclusion

After completing the necessary tasks to meet the objectives of this project, we can come to the following set of conclusions:

- Using different structuring elements, an object can have different skeletons. If we used only a few structuring elements of B_i that were mentioned, then we would get a different skeleton.

- Using the skeleton, several structures can be identified and thus pattern recognition and classification could be done.
- Shape analysis is an effective method for matching similar objects. Moreover, while computing the distances between test objects and reference objects, weights can be introduced in order to emphasize various components in the objects. Therefore, better matching results can be obtained.