CSE585/EE555: Digital Image Processing II
Computer Project # 2
Homotopic Skeletonization and Shape Analysis

Mudit Garg, Mayank Murali, Niranjan Thirusangu

Date: 02/21/2020

A. Objectives

1. Implement an algorithm for homotopic skeletonizations by using thinning operation, which in itself is an application of hit-or-miss transform.
2. Shape analysis - isolate distinct objects and find the similar objects in another image. Following needs to be computed in order to achieve this objective:
   a. Size distribution, pecstrum and shape complexity.
   b. Perform the pecstral analysis.

B. Methods

The process flow followed for the project implementation (both parts) is as follows:

Part A

1. Read images "$bear.gif$" and "$penn256.gif$", create 8 structuring element B, each, for both foreground and background.
2. Call skeletonization function to perform homotopic skeletonization.
3. Skeletonization will perform thinning operation on the given images using the structuring elements by applying hit-or-miss transformation followed by intersection. As per P&V section 6.10, equation 6.10.1 illustrates the thinning operation. It is given as
$$X \bigcirc B_i = X - X \circledast B_i,$$
where, $B_i = \{B_i^f, B_i^b\}$ is a set of two structuring elements and $X \circledast B_i$ is the hit-or-miss transform given by $X \circledast B_i = (X \ominus B_i^f) \cap (X^C \ominus B_i^b)$
4. Image obtained from Step 3 will be fed as input to the thinning operation again. This step will be continued until the new image obtained is same as the one that is fed as input to the thinning operation.
5. The final obtained image will be skeleton of the image.
6. Following is the pseudo code of the algorithm. This algorithm works similar to the grass-fire mechanism mentioned in P&V 6.8.

```
i=0;
Xi = X;
do
    i = i+1;
    Xi = Xi-1;
    for j = 1,2,…,8 do
        Xi = Xi◯ Bj
    end for
while Xi ≠ Xi-1
```

7. Following 8 structuring elements are used in the above-mentioned algorithm

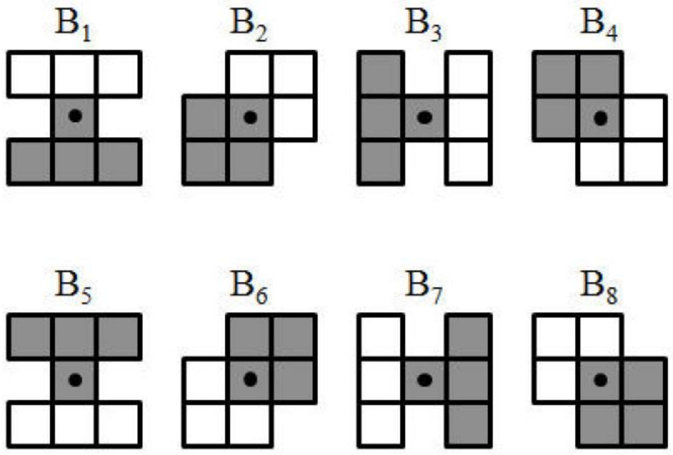


Figure 1: Given structuring elements B_i, where the 4 shaded pixels represents the 1's in the structuring element B_{if} and the 3 white pixels represents 1's of the structuring element B_{ib}.

8. Each structuring element pair B_i strips/thins off a layer of pixels in a particular direction; i.e., B_1 strips off top (north) pixels, B_2 strips off top-right (northeast) pixels, B_3 strips off right (east) pixels, etc. Thus, each thinning iteration through the complete set of 8 B_j isotropically "burns off" a layer around X. Thus, the algorithm effectively implements the "grass fire" transform, while preserving the homotopy of the original image.

Part B

Finding Minimum Bounding Rectangle (MBR)

1. Obtain the Minimum bounding rectangle for all the objects in image "$match1.gif$" using MATLAB connected component labeling function. This is performed by calling $mbr()$ function.
2. In order to do so, we first got a labeled image by applying $bwlabel$ function on given image.
3. This function labels the separate regions. We found all the pixels grouped by each label/object.
4. This step is performed for each label. The minimum value of the rows in the obtained pixels gives the y_min and the maximum value gives the y_max. Similarly, the minimum value of the column gives the x_min and the maximum value gives the x_max.
5. The obtained 4 values give the bounding box of each object.
6. A rectangle is formed on the given image to highlight the MBR.
7. All the objects are also saved separately so that the operations can be performed easefully.

Calculating Size Distribution

1. According to P&V 6.11, size distribution is given by:
   $$u(r) = m(X_{rB})$$
   where, $X \rightarrow$ the image,
   $rB \rightarrow$ the structuring element with which X is opened
   $m(.) \rightarrow$ defines the measurement of the area.

   From a physical point of view, the size distribution of X gives the area of X which can be covered by a disk rB of radius r, when this disk moves inside the object X.
2. Size distribution is obtained by calling function $size\_distribution()$.
3. The structuring element which is given is of size 3×3. This will be used as a structuring element with radius 1 unit.
4. In order to get nB, we can perform the dilation operation $n - 1$ times on the same element B with itself as the structuring element. This will give us another structuring element of size $3 + (2 \times (n - 1))$.
5. Each object obtained was individually opened using rB.
6. We performed opening of the image with radii $\in \{1:12\}$.

7. The radii range 1:12 was only considered because the area obtained with the structuring element $12B$ was 0. This implies that with any structuring element of radius > 12 will yield size distribution of 0.
8. The area was obtained by counting the number of pixels in the opened image.
9. Finally, size distribution was plotted on the graph.

Calculating Pecstrum

1. Equation 6.11.8 in P&V 6.11 illustrates the formula for pecstrum, which is given by
$$f(n) = \frac{m(X_{nB}) - m\big(X_{(n+1)B}\big)}{m(X)}$$
2. In the above equation, $m(X)$ means the total area of the original object, $m(X_{nB})$ and $m(X_{(n+1)B})$ are the size distribution with structuring element of radius *n* and *n+1*.
3. Pecstrum is calculated by calling function $pecstrum()$. It takes the size distribution and the total area of the object as the arguments. Thus, computing the pecstrum with the formula mentioned in point 1.
4. Once obtained, pecstrum was plotted in the graph, which is shown in the results.

Calculating Shape Complexity

1. Shape complexity is similar to the entropy of an object. As mentioned in L6-15 (Maragos-Schafer, eq-40), which is given by
$$H(X|B) = -\sum_{i=0}^{N-1} f(i)\log\big(f(i)\big)$$
where, $f(i)$ is the pecstrum with radius i.
2. It is implemented in function $complexity()$, which takes an array with pecstrum values as the argument.
3. Thus, it can be computed by calling $complexity(pecstrum)$ in our code.

Object Matching

1. As discussed in the lectures and P&V 6.11, spectrum can also be used in pattern matching.
2. Using equation 6.11.10 of P&V, if $f_R(n)$ is a reference pecstrum and $f(n)$ is the pecstrum of new object X, the distance d is given by,

$$d = \left[\sum_{n=0}^{N-1} c_n \big(f(n) - f_R(n)\big)^2 \right]^{1/2}$$

3. It can be used to decide if the new object coincides with a reference pattern.
4. Smaller the value of d, more similar are the objects. Objects with $d = 0$ are considered to be same.
5. The c_n in the equation corresponds to weights. They are chosen to emphasize certain components of pecstrum which has significance. We have obtained the weights in our algorithm by monotonically decreasing the weights with increase in the radius of the structuring element. This approach is employed as it's intuitive that the shape information will be lost monotonically with the increase of the radius of the structuring element.
6. This distance is implemented in function $dist\_calculation()$. It takes the pecstrum values of an object, pecstrum value of referenced object and the weights as arguments.
7. It will calculate the distance and returns a grid containing the distance between all the objects in the test image and referenced image.

Flowchart Diagram (Illustration of MATLAB code)

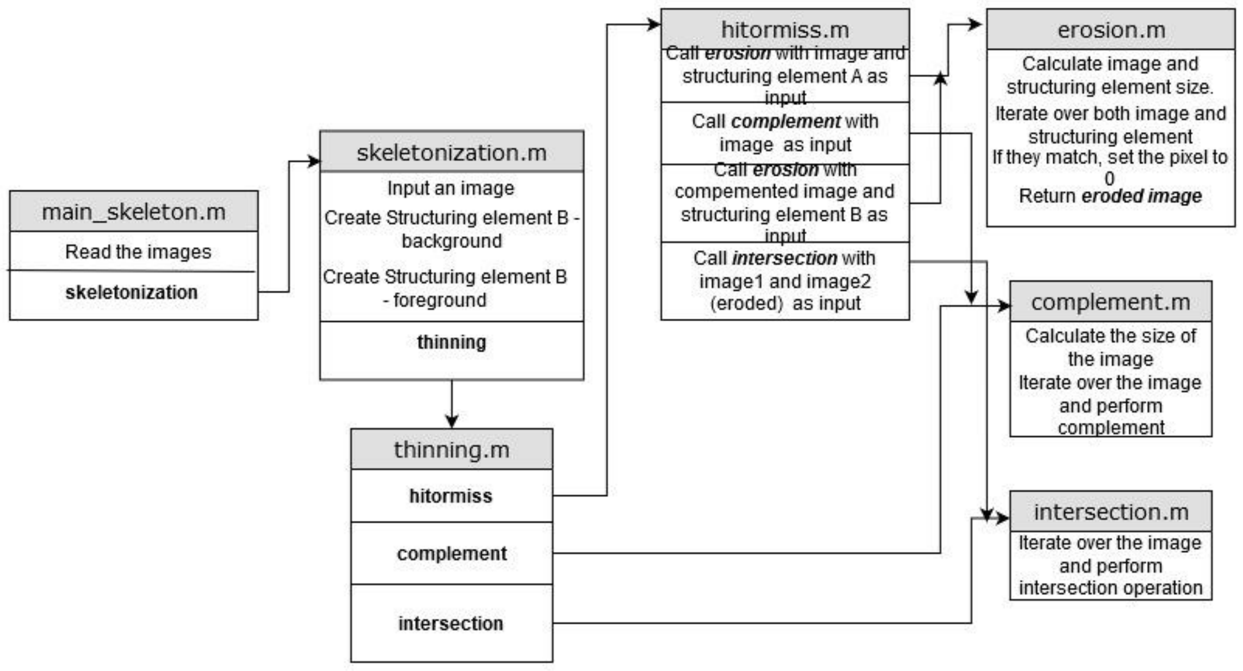

main_skeleton.m
Read the images
skeletonization
skeletonization.m
Input an image
Create Structuring element B - background
Create Structuring element B - foreground
thinning
thinning.m
hitormiss
complement
intersection
hitormiss.m
Call erosion with image and structuring element A as input
Call complement with image as input
Call erosion with compemented image and structuring element B as input
Call intersection with image1 and image2 (eroded) as input
erosion.m
Calculate image and structuring element size.
Iterate over both image and structuring element
If they match, set the pixel to 0
Return eroded image
complement.m
Calculate the size of the image
Iterate over the image and perform complement
intersection.m
Iterate over the image and perform intersection operation

Figure 2: Flow-chart showing the code execution of main_skeleton.m. It's the Problem 1 of the project.

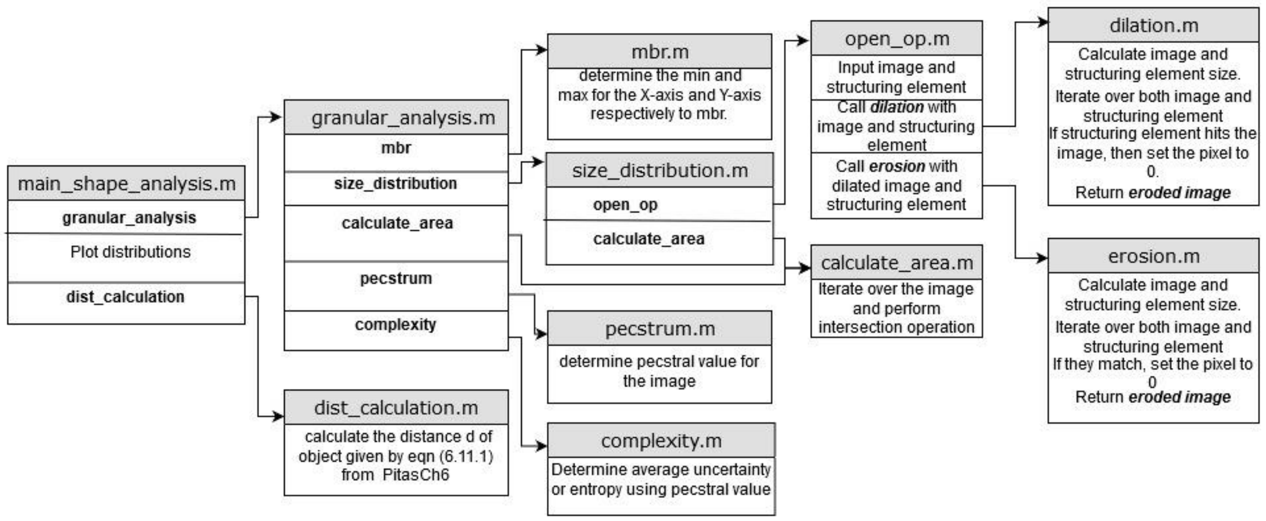

main_shape_analysis.m
granular_analysis
Plot distributions
dist_calculation
granular_analysis.m
mbr
size_distribution
calculate_area
pecstrum
complexity
dist_calculation.m
calculate the distance d of object given by eqn (6.11.1) from PitasCh6
mbr.m
determine the min and max for the X-axis and Y-axis respectively to mbr.
size_distribution.m
open_op
calculate_area
pecstrum.m
determine pecstral value for the image
complexity.m
Determine average uncertainty or entropy using pecstral value
open_op.m
Input image and structuring element
Call dilation with image and structuring element
Call erosion with dilated image and structuring element
calculate_area.m
Iterate over the image and perform intersection operation
dilation.m
Calculate image and structuring element size.
Iterate over both image and structuring element
If structuring element hits the image, then set the pixel to 0.
Return eroded image
erosion.m
Calculate image and structuring element size.
Iterate over both image and structuring element
If they match, set the pixel to 0
Return eroded image

Figure 3: Flow-chart showing code execution of main_shape_analysis.m. It's the first part of Problem 2 in the project.

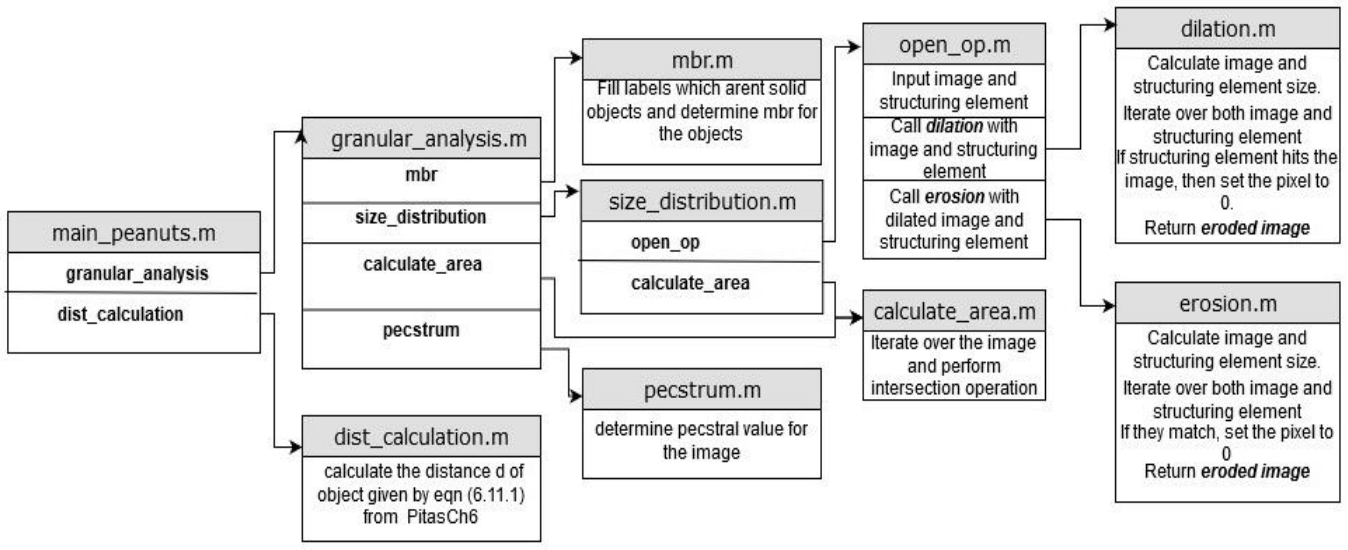

main_peanuts.m
granular_analysis
dist_calculation
granular_analysis.m
mbr
size_distribution
calculate_area
pecstrum
dist_calculation.m
calculate the distance d of object given by eqn (6.11.1) from PitasCh6
mbr.m
Fill labels which arent solid objects and determine mbr for the objects
size_distribution.m
open_op
calculate_area
pecstrum.m
determine pecstral value for the image
open_op.m
Input image and structuring element
Call dilation with image and structuring element
Call erosion with dilated image and structuring element
calculate_area.m
Iterate over the image and perform intersection operation
dilation.m
Calculate image and structuring element size.
Iterate over both image and structuring element
If structuring element hits the image, then set the pixel to 0.
Return eroded image
erosion.m
Calculate image and structuring element size.
Iterate over both image and structuring element
If they match, set the pixel to 0
Return eroded image

Figure 4: Flow-chart showing the code flow of main_peanuts.m. It's the second part of Problem 2 in this project.

Running MATLAB code:

In order to obtain the results as shown in next section, just run $main\_skeleton.m$ to obtain the homotopic skeleton.

For shape analysis, run file $main\_shape\_analysis.m$. It will give the size distribution, pecstrum and shape complexity of 'match1.gif'. Also, it will give the distances of objects between 'match1.gif' and 'match3.gif'.

For comparing the solid objects in images 'shadow1.gif' and 'shadow1rotated.gif', just run file $main\_peanuts.m$. It will display the distances between objects, which can help in finding the similar objects between the images.

C. Results

Part A:

We are given images 'bear.gif' and 'penn256.gif' which are shown in figure 5 and figure 6 respectively. The task is to perform the homotopic skeletonization of the given images and give the skeletonized image superimposed with original image.

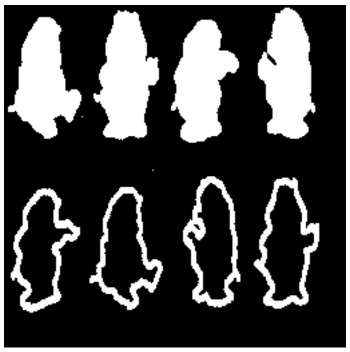


Figure 5: Bear Image before homotopic skeletonization

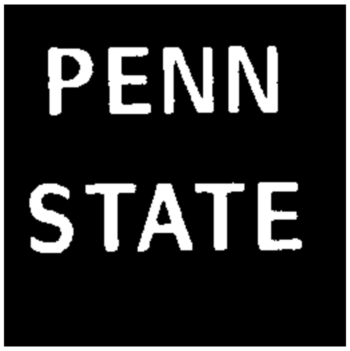


Figure 6: Penn256 Image before homotopic skeletonization

We used background and foreground structuring elements, each of count 8 as given in Figure 1. We also identified that structuring elements B_5, B_6, B_7 and B_8 are symmetric to B_1, B_2, B_3 and B_4 respectively. So, in order to obtain these elements, we rotated B_1, B_2, B_3 and B_4 by 180°.

Next, we made a copy of the original images and executed the algorithm listed in the methods section.

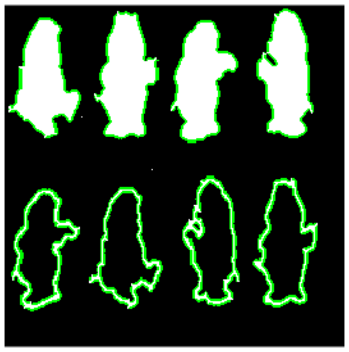


Figure 7: Image fused with original image at 2nd step of algorithm

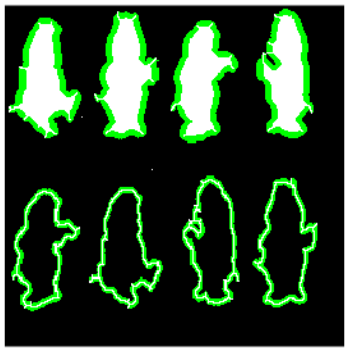


Figure 8: Image fused with original image at 5th step of algorithm

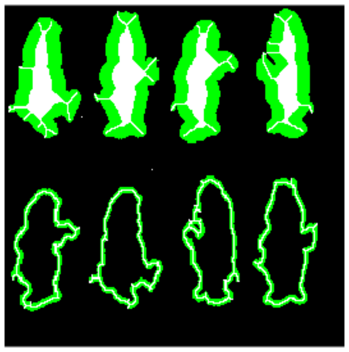


Figure 9: Image fused with original image at 10th step of algorithm

The above three images show the working of skeletonization algorithm. These images are fused with the original image (shown in Figure 5). These images are obtained at the step 2, 5 and 10 of the skeletonization algorithm respectively. It shows that at every step, image is getting reduced from all directions. This is very much similar to grass-fire transform. It keeps burning until a single shed like skeleton is obtained. Same happened with our image. Final skeletonized image (fused with original image) is shown below:

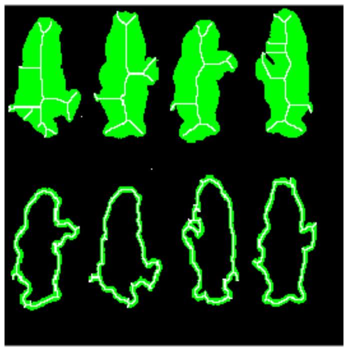


Figure 10: Final Skeletonized image of bear fused with original image

Entire image is burned off. Only the skeleton is left. Figure 10 also shows that original image can be reconstructed from this skeleton if we also have the radius with which it shrinked off at the points in skeleton.
The same algorithm was also executed on $penn256.gif$ image. Following are the images of penn256 with the given algorithm.

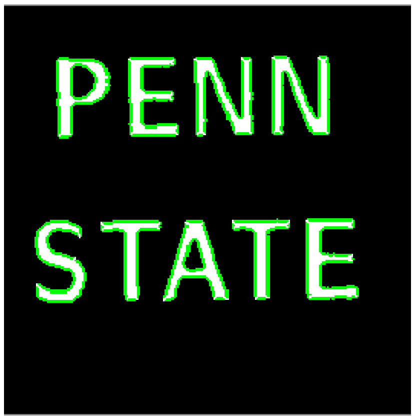


Figure 11: Image fused with original image at 2nd step of skeletonization algorithm

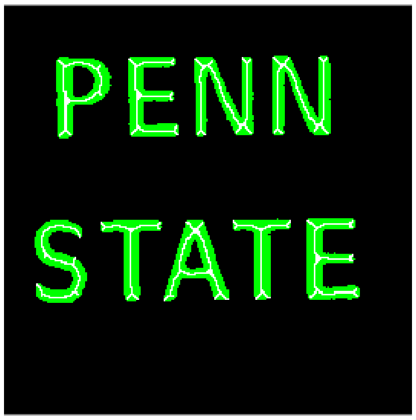


Figure 12: Image fused with original image at 5th step of skeletonization algorithm

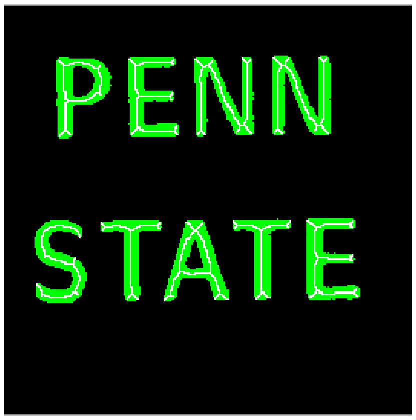


Figure 13: Final Skeletonized image fused with original image

The algorithm with $penn256$ image was completed in 8 steps only. It means that final skeleton was obtained in the 8th step only. Since the width of the objects in this image was less, the algorithm stopped very quickly. The execution of homotopic skeletonization algorithm on the *bear* and $penn256$ images shows that this algorithm will be expensive if we have broad width objects in the image.

Part 2

Problem (a)

We are given image $match1.gif$ and are supposed to found out the side distribution, pecstrum and shape complexity of all the 4 objects present in it. The objects are (1) Clover, (2) Steer, (3) Airplane, and (4) Spade.

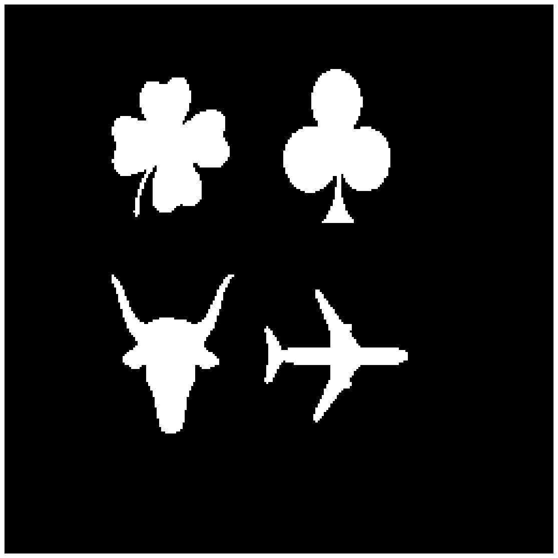


Figure14: Original Image (Match1.gif)

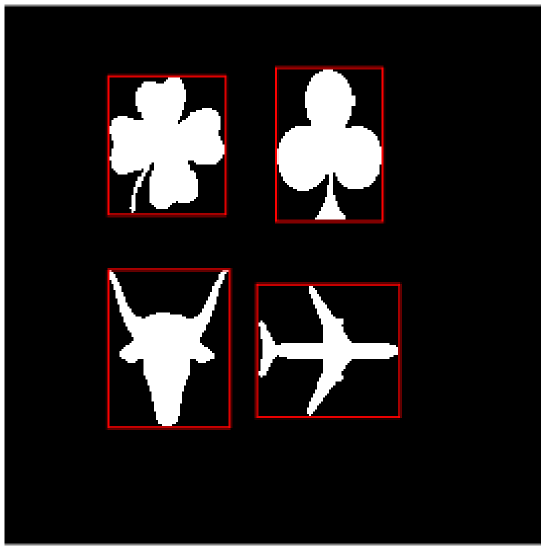


Figure 15: Image with MBR around objects

Firstly, minimum bounding rectangle was formed on $match1.gif$. This was necessary to obtain the coordinates of the bounding boxes so that all the four objects can be extracted. Figure 15 shows the $match1.gif$ image with MBR.

After obtaining MBR, the objects were extracted using the coordinates. In order to extract the objects, we filled the three objects with 1 so that we are left with just 1 object in the image. We did this for all the objects. Following shows the images displaying only one object.

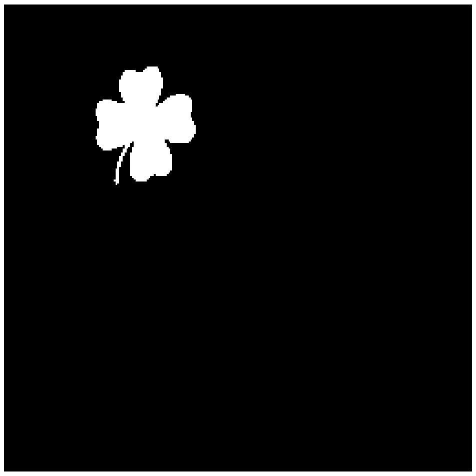


Figure 16: Clover Object

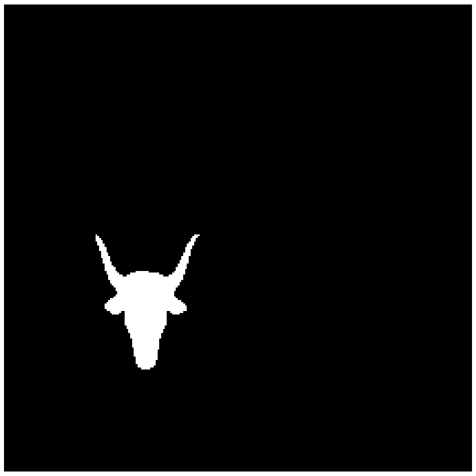


Figure 17: Steer Object

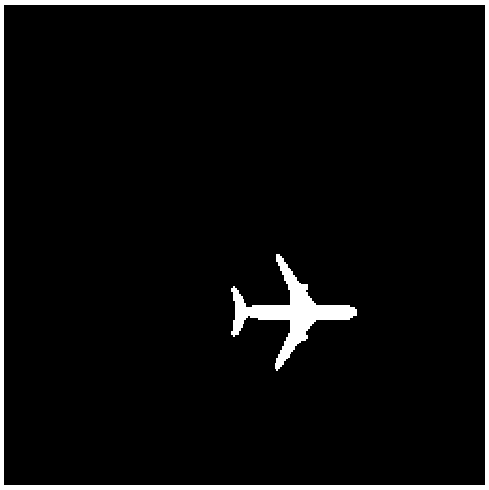


Figure 18: Airplane

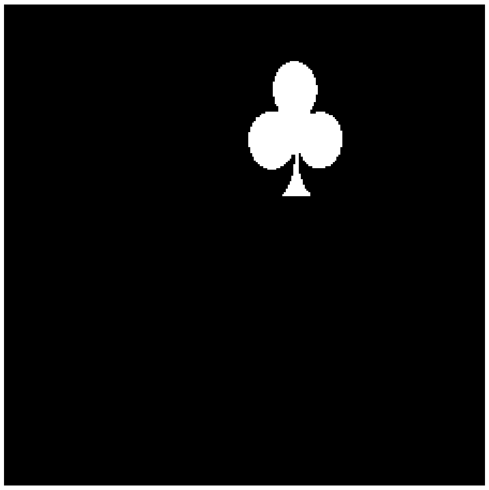


Figure 19: Spade

Images of the objects extracted from the match1.gif

After successfully extracting the objects, we performed the opened operation on them using structuring element of size $3, 5, 7, \dots, ((2 \times r) + 1)$. The area was calculated after every open operation by counting the number of pixels present in the opened image. This area is the size distribution of the objects. The detailed results of the size distribution of all the 4 objects are shown below.

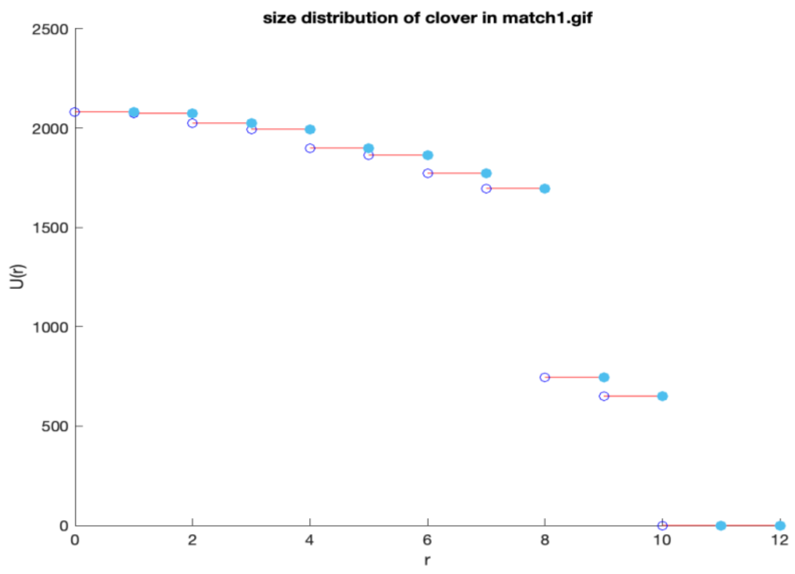
Size Distribution of objects in the image match1.gif

Radius	Objects			
	Clover	Steer	Airplane	Spade
1	2082	1488	990	1930
2	2075	1332	850	1900
3	2025	1215	647	1852
4	1995	1155	346	1773
5	1900	1127	225	1746
6	1864	1081	0	1734
7	1773	1015	0	1667
8	1697	924	0	1590
9	746	873	0	1072
10	651	663	0	821
11	0	621	0	0
12	0	0	0	0

Table 1: Size distribution of all 4 objects in match1.gif wrt different radius of structuring element.

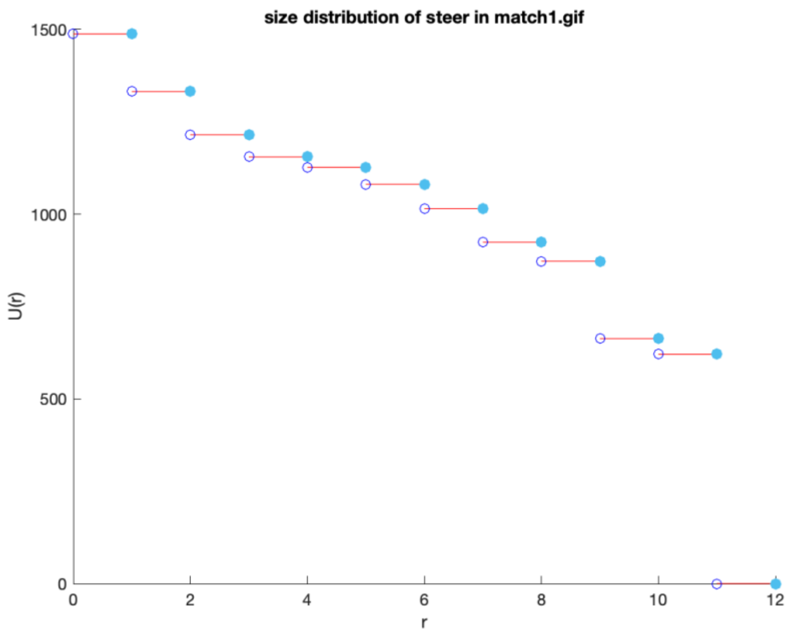
Table 1 shows that as the radius increases, the area covered by the opened image decreases. Thus, it proves that size distribution $U(r)$ is a strictly monotone function.

The plots of the size distribution of all 4 objects are shown on the graphs 1, 2 3 and 4. Plots also clearly show that area decreases with increase in radius *r*.

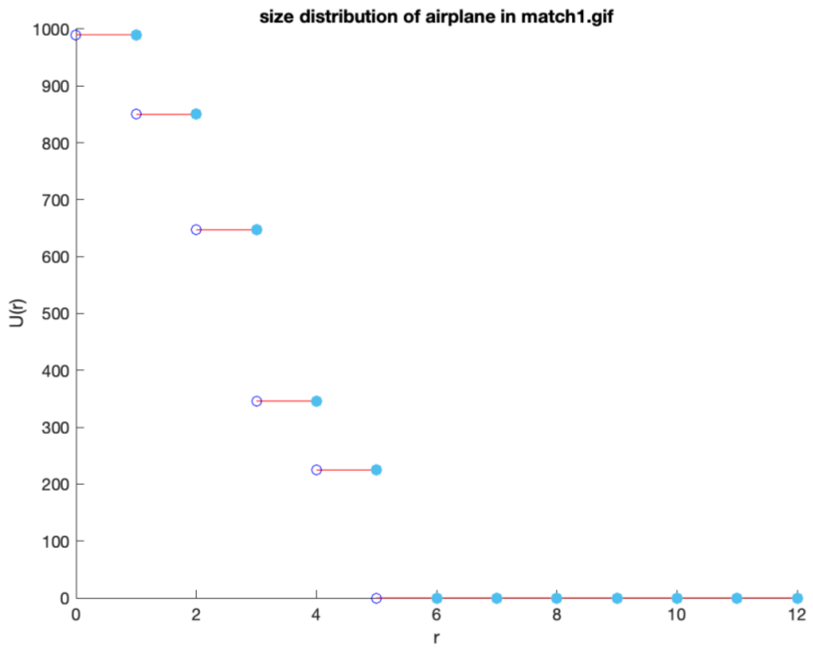


Graph 1: size distribution of clover object in image match1. gif

There is a sudden drop in size distribution of clover when radius of structuring element reaches 8. This is because the structuring element $(8B)$ is cutting off the leaves of clover object.

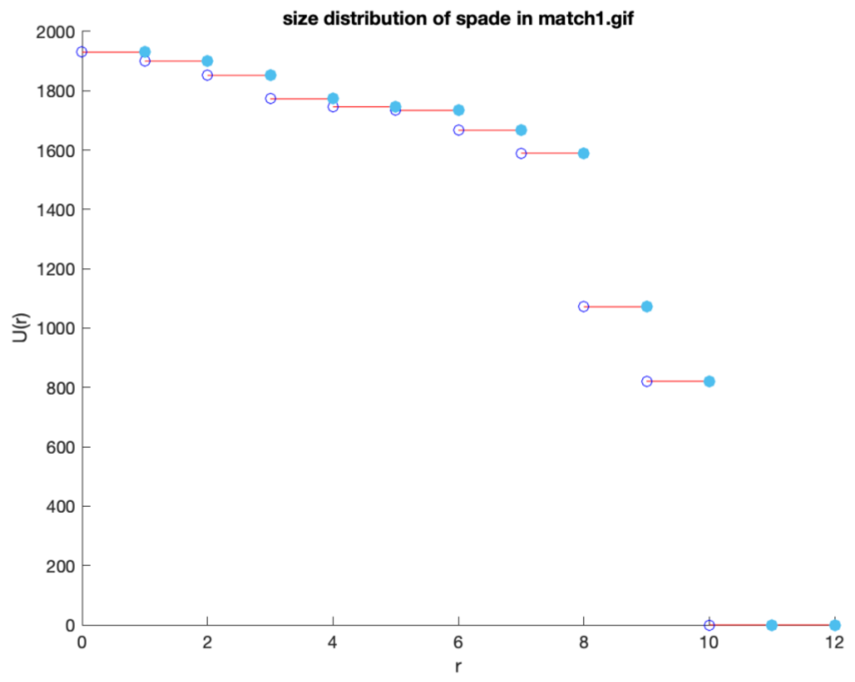


Graph 2: size distribution of steer object in match1.gif



Graph 3: size distribution of airplane in match1.gif

It is clearly evident in the graph 3, the airplane object has small protrusions, i.e., wings and tail are being slowly chopped off with the increase of the radius of the structuring element.



Graph 4: size distribution of spade in match1.gif

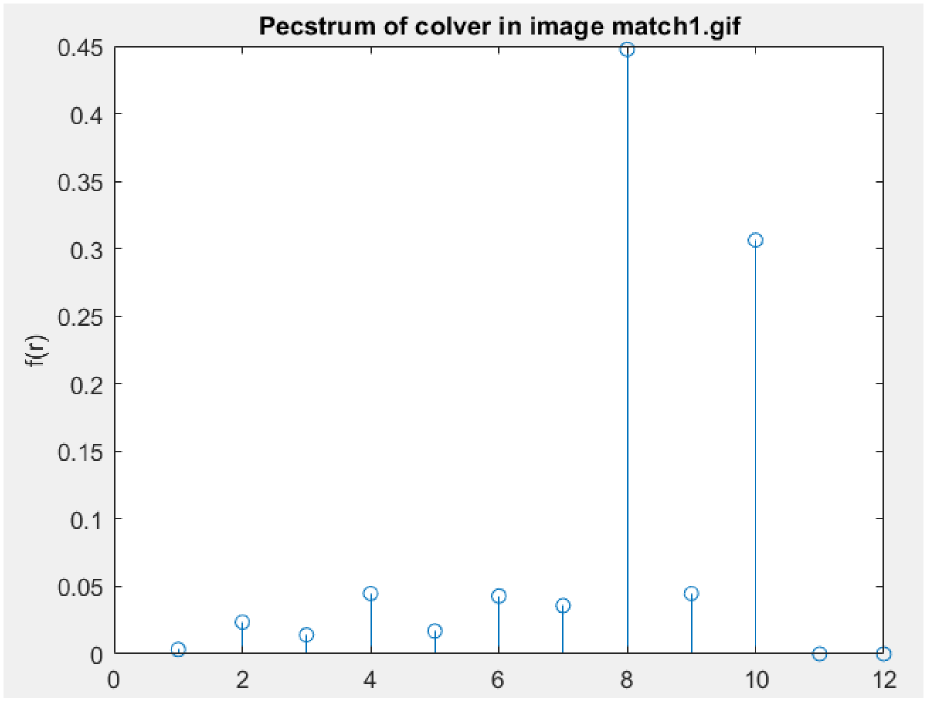
Similar to the clover object, the leaves are getting chopped when the radisu of structuring element reached 8. But, as it has only three leaves comparing to the four leaves of clover, the drop is low.

Pecstrum of objects in the image match1.gif

	Objects			
Radius	**Clover**	**Steer**	**Airplane**	**Spade**
1	0.0033	0.1032	0.1396	0.0154
2	0.0235	0.0774	0.2024	0.0247
3	0.0141	0.0397	0.3001	0.0406
4	0.0447	0.0185	0.1206	0.0139
5	0.0169	0.0304	0.2243	0.0062
6	0.0428	0.0437	0	0.0344
7	0.0358	0.0602	0	0.0396
8	0.4475	0.0338	0	0.2663
9	0.0447	0.139	0	0.129
10	0.3064	0.0278	0	0.4221
11	0	0.411	0	0
12	0	0	0	0

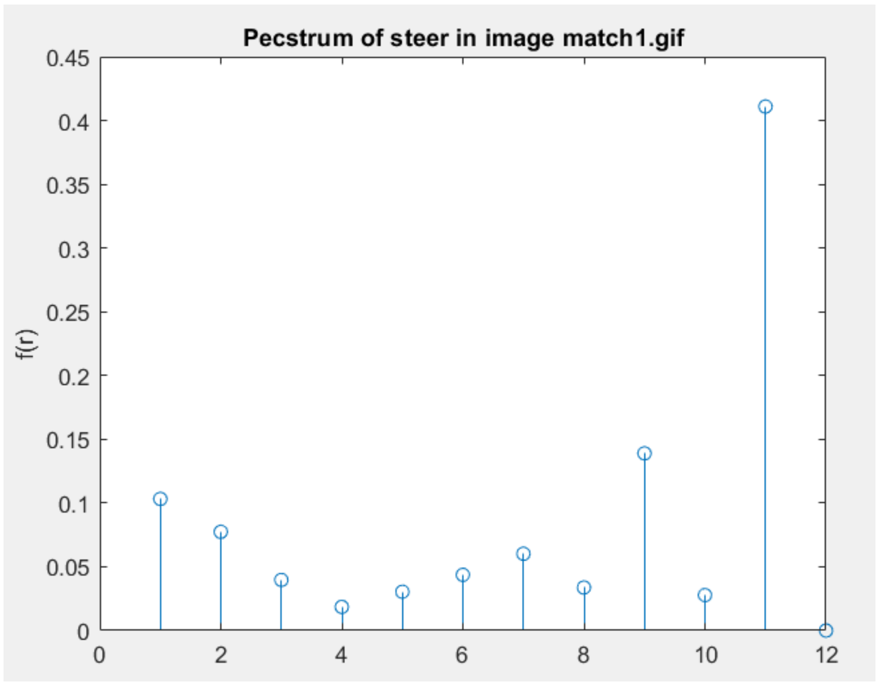
Table 2: Pecstrum values of all 4 objects of match1.gif with respect to different radius of structuring element

The above table gives the pecstrum of the 4 objects in $match1.gif$. It is calculated using eq 6.11.8 of P&V 6.11. The pecstrum of all the objects are plotted below.

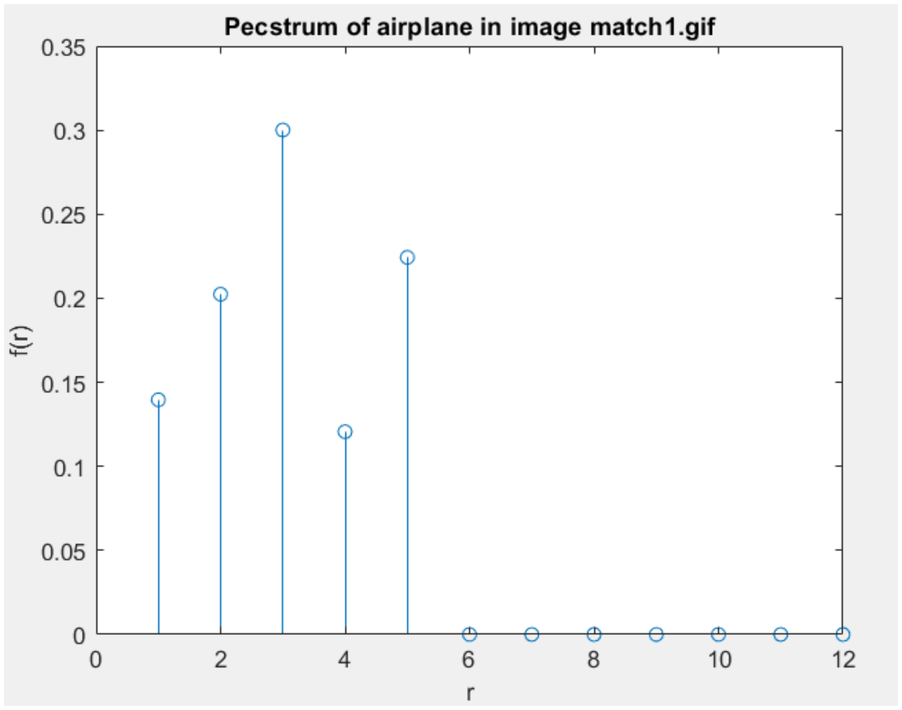


Graph 5: Pecstrum of clover in match1.gif

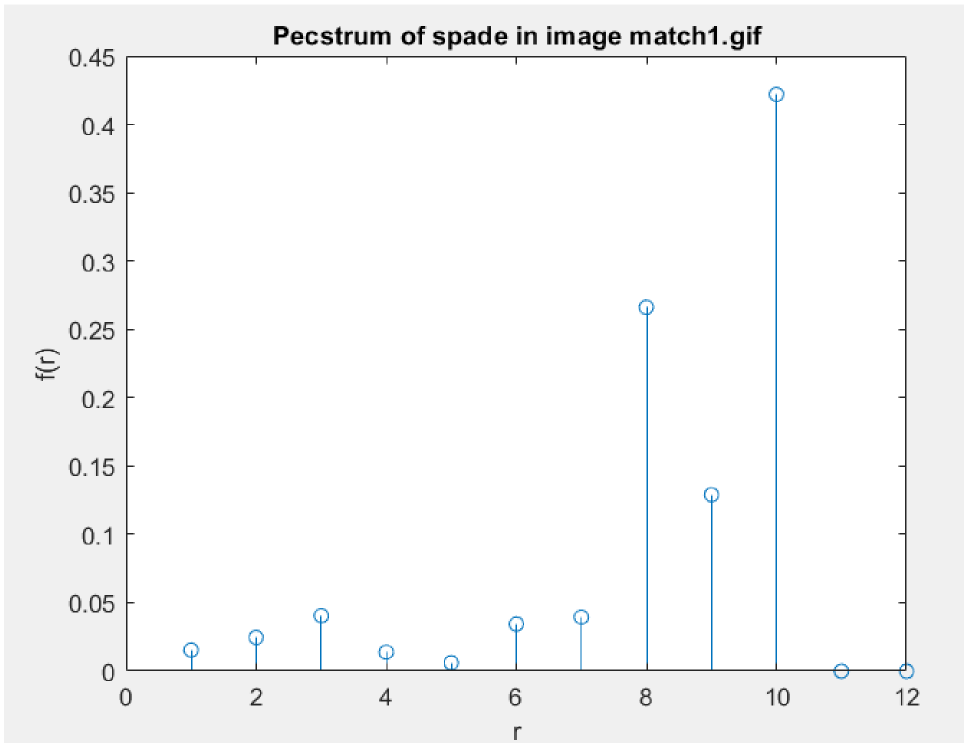
The sudden drop in size distribution, which we have already discussed, is getting reflected as the sudden spike in the pecstrum plot of clover (Graph 5). The reasoning behind the sudden fluctuation is same as in the size distribution case.



Graph 6: Pecstrum of steer in match1.gif



Graph 7: Pecstrum of airplane in match1.gif



Graph 8: Pecstrum of spade in match1.gif

The sudden drop in the size distribution due to removal of tails and wings in airplane is also pretty evident in the pecstrum plot (Graph 7) when the radius is 5. Graph 8 shows the pecstrum of spade object, which is very much similar to clover, and shows the same trend.

Shape Complexity

Objects	Shape-Complexity
Clover	2.1502
Steer	2.7418
Airplane	2.2359
Spade	2.3099

Table 3: Shape complexity values of all 4 objects of match1.gif

The above table shows the shape complexity of all the 4 objects. Shape complexity is similar to entropy and is given by equation discussed in Methods section. More the value, greater the complexity of the object. From the above values, it is clear that the most complex object is Steer, whereas the least complex is clover.

Comparison with $Match3.gif$

Another task was to compare the four objects of $match1.gif$ with the four objects of $match3.gif$. As discussed in the methods sections, we need to calculate the distance between the objects using pecstrum. We already have the pecstrum values of match1.gif. Now, we need to find the MBR of match3.gif and extract its objects.

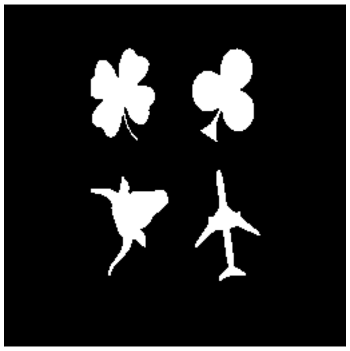


Figure 20: Original match3.gif image

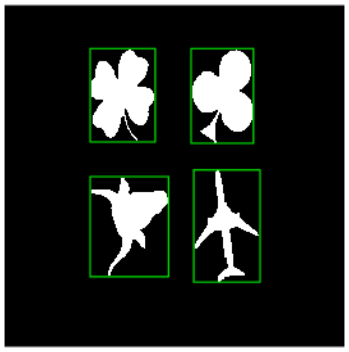


Figure 21: match3 Image with MBR

After finding the coordinates of MBR, we extracted the objects similar to the way they were extracted for $match1.gif$. The extracted objects are shown below. Clearly, these objects are the rotated versions of the objects in the $match1$ image. Since we need to identify these objects, let us call them U_1, U_2, U_3 *and* U_4. The caption of the images below describes the label of these unidentified objects.

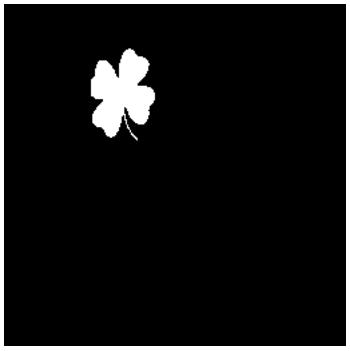


Figure 22: Object U_1

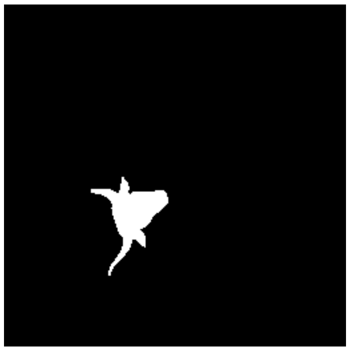


Figure 23: Object U_2

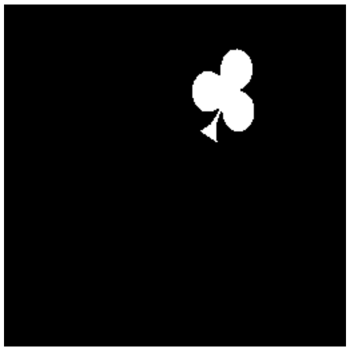


Figure 24: Object U_3

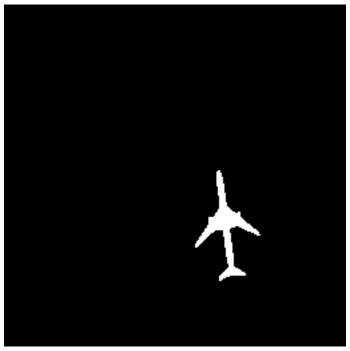


Figure 25: Object U_4

After extracting the objects, we calculated distance using the equation 6.11.10 of P&V, keeping $match1$ image objects as the reference ones. Following table gives the distance calculations between all the objects.

	$\boldsymbol{U_1}$	$\boldsymbol{U_2}$	$\boldsymbol{U_3}$	$\boldsymbol{U_4}$
Clover	**0.0004**	0.0135	0.0012	0.184
Steer	0.0147	**0.0025**	0.0124	0.1231
Airplane	0.1369	0.0944	0.1311	**0.012**
Spade	0.0009	0.0097	**0.0003**	0.1836

Table 4: Distance calculations between objects of match1.gif (reference image) and match3 image (test image)

The above table shows the distances between the known objects and unidentified objects. The distances in bold shows the minimum distance, meaning those objects are the closest match with each other. The above results are highly dependent on weight vector, which signifies the certain components of pecstrum. The weight vector which we have used is [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]. We chose these specific values of weight because from the size distribution, it is pretty evident that the objects are losing its shape information by losing its size, which is compensated by the reduction of the weights monotonically.

Known Object	Matched Object	Comments
Clover	U_1	Clover closely matches with object U_1. On seeing manually also, U_1 is a clover object
Steer	U_2	Steer matches with U_2, which is totally as expected.
Airplane	U_4	Airplane matches with U_4, which again is a correct match.
Spade	U_3	Spade matches with U_3. It is also a correct match.

Table 5: Table showing the matched objects of match3.gif with the objects of match1.gif

Table 5 proves that the algorithm works as expected.

Problem (b)

In this problem, we were given two images – $shadow1.gif$ and $shadow1rotated.gif$. Both the images are shown in figure 26 and figure 27 respectively. Our task is to find the matching solid objects in both the images.

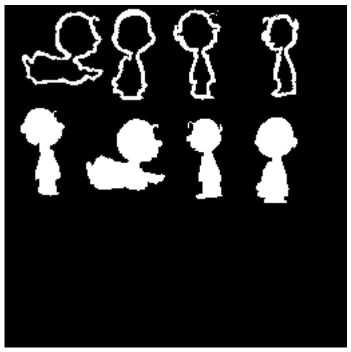


Figure 26: Original Image Shadow1.gif

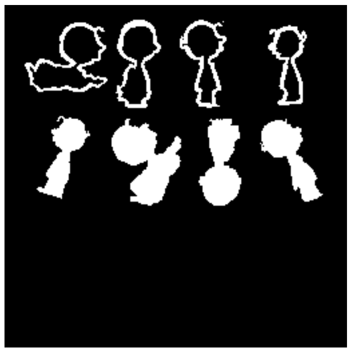


Figure 27: Original Image Shadow1rotated.gif

This task seems achievable using the object matching approach mentioned in the Methods section. But in these images, there are objects other than the solid ones. So, first we need to fill in the non-solid objects. We did this using the approach similar to the one used for finding MBR. We found the labels of these objects, retrieved the pixels for these labels and filled those pixels with 0 values. After filling the non-solid objects, we found the MBR of the solid objects using the approach used in problem (a). Images with MBR are shown in Figure 28 and Figure 29.

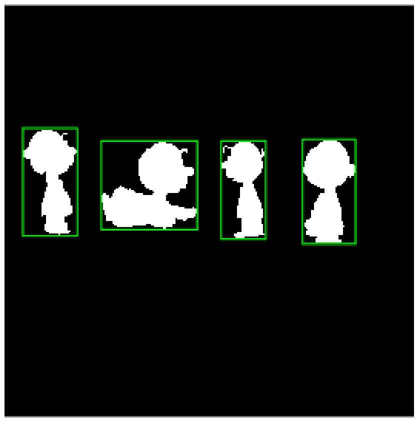


Figure 28: MBR of shadow1.gif Image

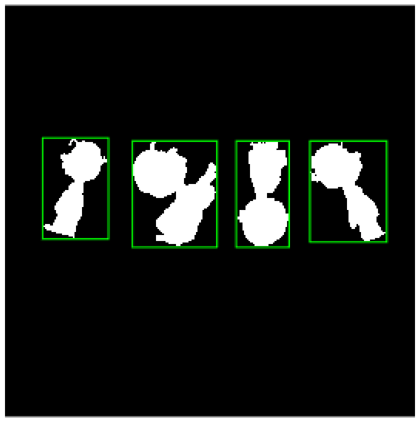


Figure 29: MBR of shadow1rotated.gif

After finding the MBR, we extracted the objects from both the images. Let us call the objects from $shadow1.gif$ be $Peannut_1, Peanut_2, Peanut_3$ and $Peanut_4$. These objects are shown below. We also found the pecstrum of these objects.

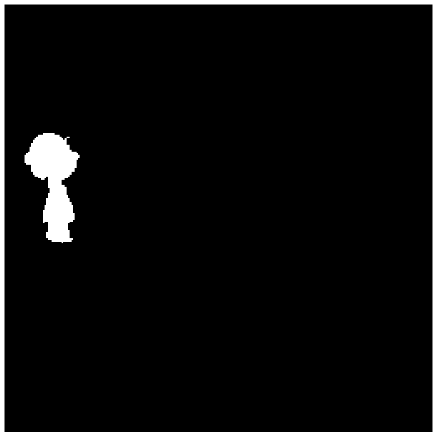


Figure 30: $Peanut_1$ Object

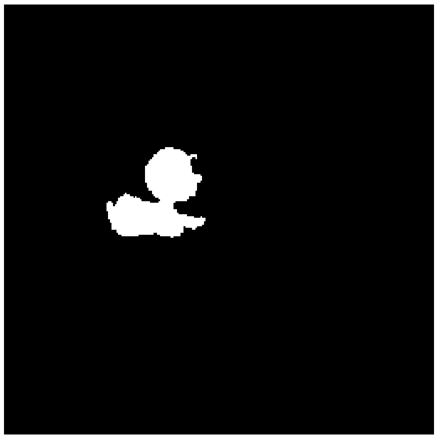


Figure 31: $Peanut_2$ Object

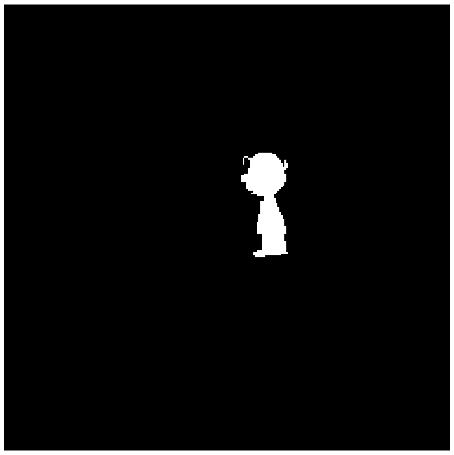


Figure 32: $Peanut_3$ *Object*

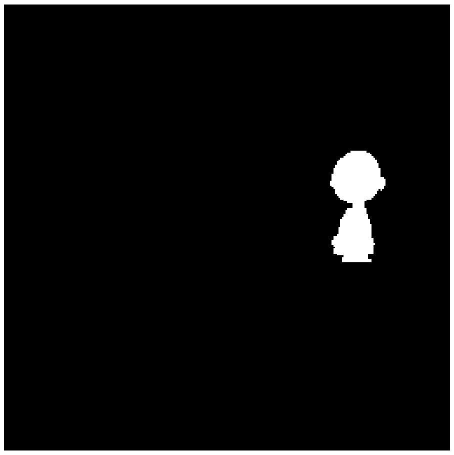


Figure 33: $Peanut_4$ *Object*

Similarly, after extracting the objects from $shadow1rotated.gif$, we found their pecstrum as well. Let us call these objects as U_1, U_2, U_3 and U_4. These extracted objects are shown below.

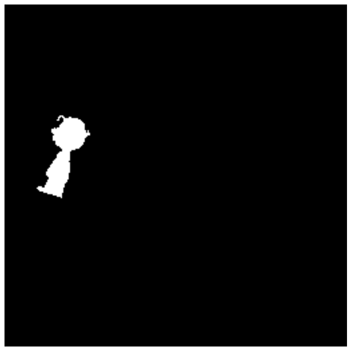


Figure 34: U_1

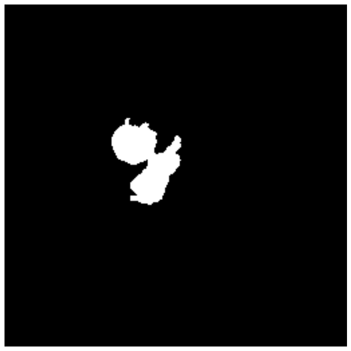


Figure 35: U_2

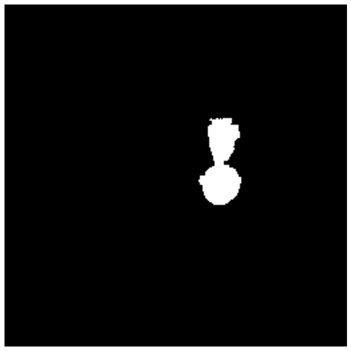


Figure 36: U_3

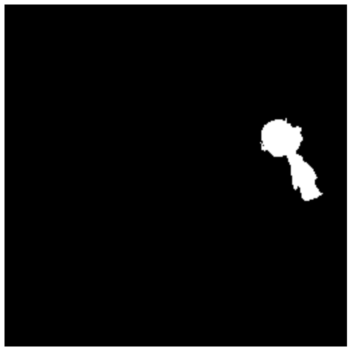


Figure 37: U_4

After founding the pecstrum for all the 8 objects, we calculated the distance using the equation used in part (a) of second problem. The objects from $shadow1.gif$ are used as reference to the objects of $shadow1rotated.gif$. Following table gives the distance calculations between these objects.

	$\boldsymbol{U_1}$	$\boldsymbol{U_2}$	$\boldsymbol{U_3}$	$\boldsymbol{U_4}$
$\boldsymbol{Peanut_1}$	0.0016	0.0066	0.0044	**0.0013**
$\boldsymbol{Peanut_2}$	0.006	**0.0006**	0.0012	0.0036
$\boldsymbol{Peanut_3}$	**0.0009**	0.0021	0.0018	0.0011
$\boldsymbol{Peanut_4}$	0.0042	0.0014	**0.0002**	0.0025

Table 6: Distance calculations between the objects of shadow1 and shadow1rotated. Shadow1 is used as a reference image.

The weights which are used to get the above results are [1, 1, 0.8, 0.7, 0.1, 0, 0, 0, 0, 0, 0, 0]. The distances shown in bold in Table 6 are the minimum distance between the two objects. Thus, making them the closest match.

Known Object	Matched Object
$\boldsymbol{Peanut_1}$	U_4
$\boldsymbol{Peanut_2}$	U_2
$\boldsymbol{Peanut_3}$	U_1
$\boldsymbol{Peanut_4}$	U_3

Table 7: Table showing the matched objects between shadow1.gif and shadow1rotated.gif

Table 7 shows the matched objects with the known ones. The matching of these objects is highly dependent on the weight vector. Choosing the weights inappropriately may lead to wrong results.

All the results that we have obtained are totally as per the expectations. Thus, this project is completed successfully.

D. Conclusion

This project was quite insightful in terms of knowledge gain. Firstly, the algorithm for implementing homotopic skeletonization is simple and easy to implement. It shows that at each step, a more sharped skeleton is getting obtained. It contains more and more finer details after every step, which can be very helpful in reconstructing the original image. The algorithm gives the visual evidence of grass fire transform. However, we also found out that the algorithm is expensive if the objects in images are much wider.

Secondly, pattern spectrum, or simply pecstrum is a very important function. It is very helpful in pattern recognition or finding similar objects. It is also helpful in describing the complexity of an object. We also learnt that choosing appropriate weights while calculating distance is an essential task to obtain quality results in pecstral analysis. This depends on the size distribution and pecstrum of the images that are needed to be analyzed.

This project also highlighted the importance of object extraction in images. It is highly complicated to analyze one object from a series of objects in an image. Thus, extracting the object is an appropriate choice and makes the task easier.