CSE585/EE555: Digital Image Processing II

Computer Project # 2:

The Morphological Skeleton and Shape Analysis

Lindsey Schwartz, Ke Liang, Xilun Liu

Date: 02/22/2019

A. Objectives

The objectives of this project are divided into two parts. The first is understanding the morphological skeleton and the effect of using structuring elements of different shapes. The second is performing shape analysis on various binary images.

B. Methods

Part 1 (Morphological Skeleton)

1）Using the structuring elements Square, Rhombus and VEC045 to get the morphological skeletons on the “penn256.gif” and “bear.gif”.

2）Partially reconstructing the original binary threshold images from the skeleton subsets for structuring element radius levels 2, 3 and 4 for the structuring elements.

As for the first objective in Part 1, in order to create the skeleton images for both “ penn256.gif ” and “ bear.gif ” using the structuring elements (3x3 pixel square, rhombus and VEC045), we use the Algorithm given by P&V Sect.6.8, eq. (6.8.2).

Consider $E = \mathcal{Z}^2$ (2-D discrete space).

$sk(X) = \bigcup_{0 \le n \le N} S_n(X)$ where:

$S_n(X) = (X \ominus nB) - (X \ominus nB)_B$ n^{th} skeletal component

$nB = \underbrace{(B \oplus B \oplus B \oplus \ldots \oplus B)}_{n\ B's}$ structuring element of radius n

$n = 0, 1, 2, \ldots, N,$ $N = \max\{k : \ X \ominus kB \neq \emptyset\}$

assume $B = B^s$ NB = largest homothetic of B that fits inside X

In our situation, the " X " refers to the " penn 256.gif " and " bear.gif ", and the structuring elements, B (radius 1), used for the skeleton images are as follows:

Square:[1,1,1;1,1,1;1,1,1]*255

Rhombus:[0,1,0;1,1,1;0,1,0]*255

VEC045:[0,0,1;0,1,0;0,0,0]*255

The skeleton is the union of all the skeletal subsets. The skeletal subsets are the collection of points for each radius of the structuring element (nB). The maximum radius, N, is the number that is the largest number k, which make X⊖kB≠Ø.

As for the second objective in Part 1, in order to partially reconstruct the binary image by specifying the lower bound, k, for the reconstruction. The following equation shows the algorithm of this part.

$$X_{kB} = \bigcup_{k \le n \le N} S_n(X) \oplus nB, \quad k \ge 0$$

All the processes to get the skeleton are implemented in our code "MorphologicalSkeleton.m" using MATLAB. To run the code, we first use the imread () function to import the image, and then we build up the structuring element. After that, we use a loop to determined the maximum radius N and then calculate every Sn(X) which the radius n meets the requirement $X \ominus kB \neq \emptyset$. When we get every Sn(X), we add them together to get the final image (variable name: FinalImage) which is the skeleton of the image. In this process, we add a conditional statement about the value of n. If it meets the requirement of the lower bound, k, for the reconstruction, we add all of them to get X_{kB}. Finally, the figures of the skeleton and the partially reconstruction will appear after the code is executed. The following is the flow chart of our algorithm, shown in Figure 1.

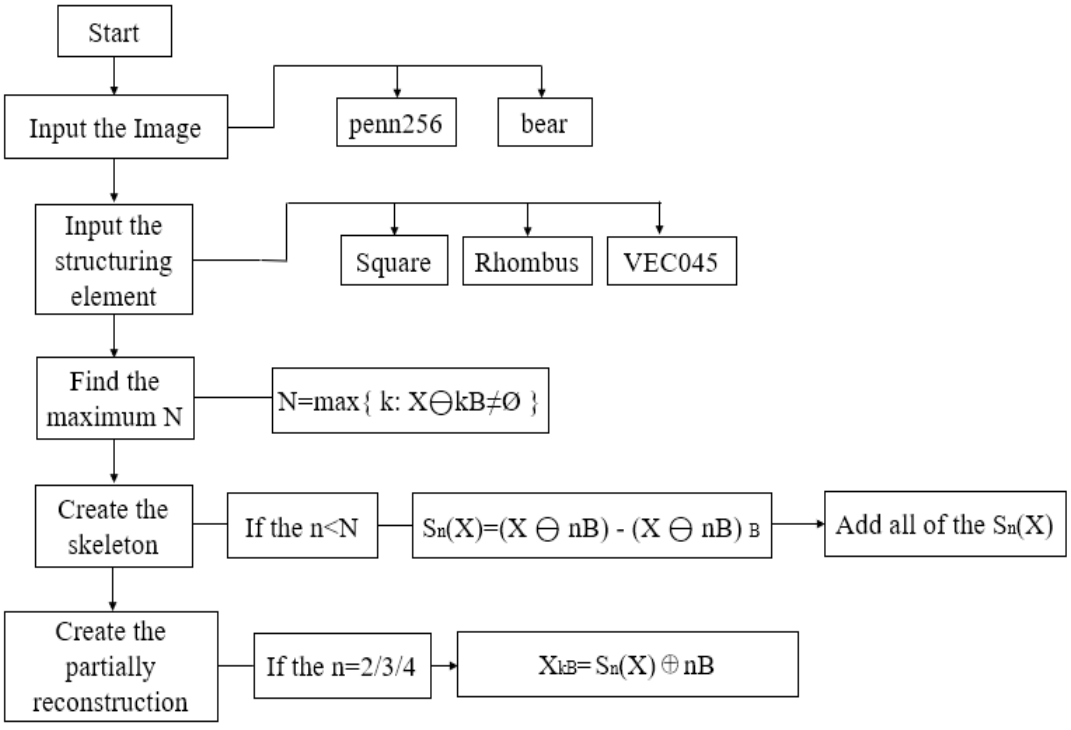


Figure 1. Algorithm flowchart of Part 1

Part 2 (Shape Analysis)

1) Isolating the four objects in “match1.gif” and computing their size distributions, pecstrums, and complexities.
2)Isolate the rotated versions of the “match1.gif” objects in “match3.gif” and use pecstral analysis to find the object in “match3.gif” that best matches the objects in “match1.gif”
3) Considering the image “shadow1”, which has four solid objects; the objects are characters from the “Peanuts” comic strip. Quantitatively (and automatically) matching them to the proper objects depicted in the complementary image “shadow1rotated”.

As for the third objective of the project, the goal is to first identify four objects in “match1.gif” by finding the minimum bounding box around each object and compute their size distributions, pecstrums, and complexities. This objective is completed by running “ShapeAnalysis.m”. The minimum bounding box is the smallest rectangle (in area) that contains the entire object to isolated. We use bwlabel() function in MATLAB to find it, and in order to be convenivent in the operation later, we use imcrop() function in MATLAB to get a little bit bigger bounding rectangle than the minimum one. The code of this operation is “Object = imcrop(X,[(minY-6) (minX-6) (maxY-minY+12) (maxX-minX+12)])”.
The figure will appear after running the code named “ShapeAnalysis.m”. With the objects isolated and the structuring element used (3x3 pixel square element), the size distribution of each object can be calculated using the equation presented below.

$$U(n) = m(X_{nB}) \quad , \quad n \in Z \geq 0$$

where XnB is the opening of X by nB and m(XnB) is the area of XnB besides nB is the dilated version of B by n times. After getting the size distributions, the pecstrum of the objects can be also computed by using the equation below.

$$f(n) = \frac{U(n) - U(n+1)}{U(0)} \quad , \qquad n \in Z > 0$$

With the normalizing constant of U(0), the sum of all f(n) values becomes unity and then we can calculate the complexity of each object by using the equation below.

$$H(X|B) = -\sum_{i=1}^{N} f(i) \log f(i)$$

where X is the object, B is the structuring element and f(i) is the pecstrum. All of the calculations are implemented in “ShapeAnalysis.m”. To run the code we just need to use the function imread() in MATLAB to import the image “match1.gif” into the program. There is the function bwlabel() to classify the image into 4 parts which contain the objects in the “match1”, and then we use the equations above to calculate the size distribution, pecstrum and complexity of the object which will appear when the code is running.

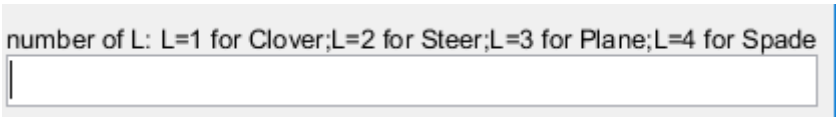
The fourth objective is to isolate the objects in “match3.gif” and use pecstral analysis to find the object in “match3.gif” that most closely matches the object in “match1.gif”.

In order to solve the matching problem we calculate the pecstrums for both “match3.gif” and “match1.gif” to calculate the distance between 2 objects in 2 images by using the following equation.

$$d_i = [\sum_{i=0}^{N-1} C_n(f(n) - f_{R_i}(n))^2]^{1/2}$$

where f(n) is the pecstrum of the test object, fRi(n) is the pecstrum of the reference object, and Cn is a weight.

As for the code for the matching problem, just input the object you want to find closely matching between “match1.gif” and “match3.gif”, as mentioned when you run the code “FindingObjectMatchBest1.m”. The indication of the program is shown below.



After you input the object you want to find in “match3.gif”, the pecstrum of the object in “match3.gif”, the distance of different values of L in “match 3” when the object in “match1” is certained, and the result of finding the object in “match3” will appear automatically based on the equation above.

In our algorithm, we used the Cn={1,0.8,0.6,0.4,0.2,0.1,0.1,0.1,0.1,0.1}, and we compare the distance of different object in “match3.gif” to the object we want in “match1.gif” to get the smallest distance, which means the distance corresponding object in “match3.gif” is the best match. The algorithm flow chart of this part is shown below.

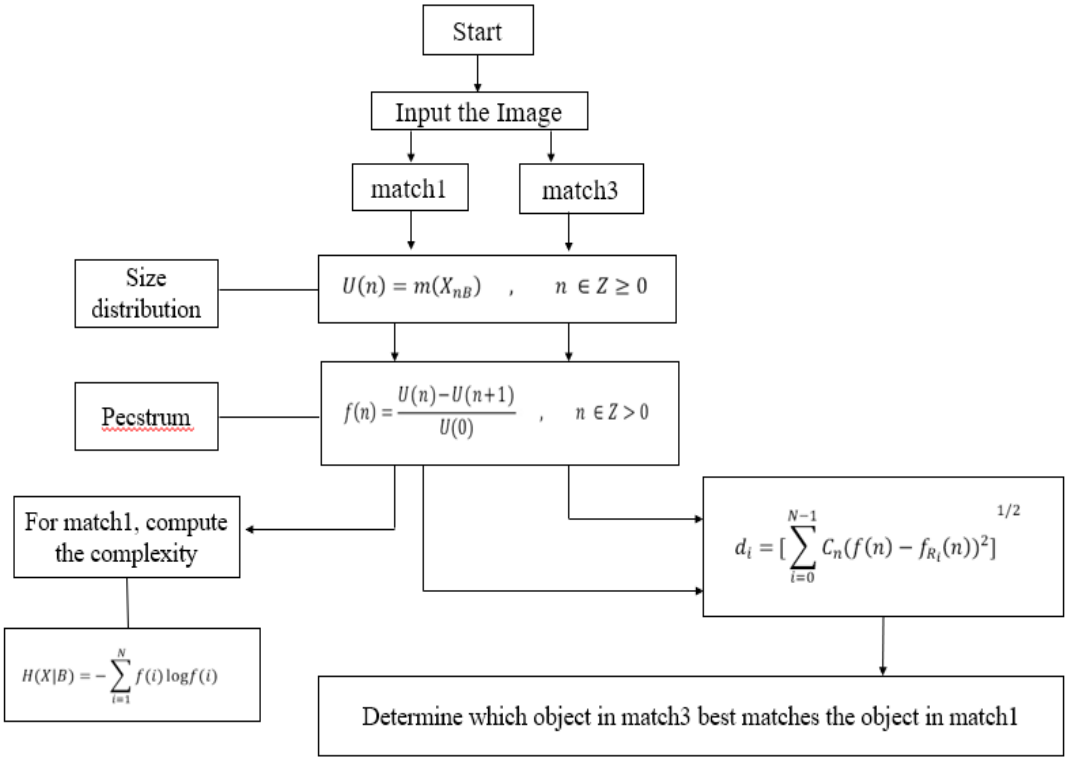


Figure 2. Algorithm flowchart of Part 2 (1)(2)

As for the fifth objective, isolate the objects in “shadow1.gif” and use pecstral analysis to find the object in “shadow1.gif” that most closely matches the object in “shadow1rotated.gif”.

We think the method of this part is almost same as how we get the best matching object between “match1.gif” and “match3.gif”, but this time there are 8 objects. Which is implemented in our code “FindingObjectMatchBest2.m”, we import 2 images in our program, and then we isolated 8 objects in “shadow1rotated.gif”. We use bwlabel() function in MATLAB to find it, and in order to be convenivent in the operation later, we use imcrop() function in MATLAB to get a little bit bigger bounding rectangle than the minimum one. The code of this operation is “Object = imcrop(X,[(minY-6) (minX-6) (maxY-minY+12) (maxX-minX+12)])”. After that we calculated the pecstrum of each object in “shadow1rotated.gif”, to do the loop for choosing the which is the best matching object in “shadow1.gif” by using the equation to

calculate the distance of the objects in these 2 images. Once, we get the smallest distance , we go back to the certain object in “shadow1.gif” as the same object. Finally, based on the same object in “shadow1.gif” we find the linking object which is the complementary image. The algorithm flow chart of this part is shown below.

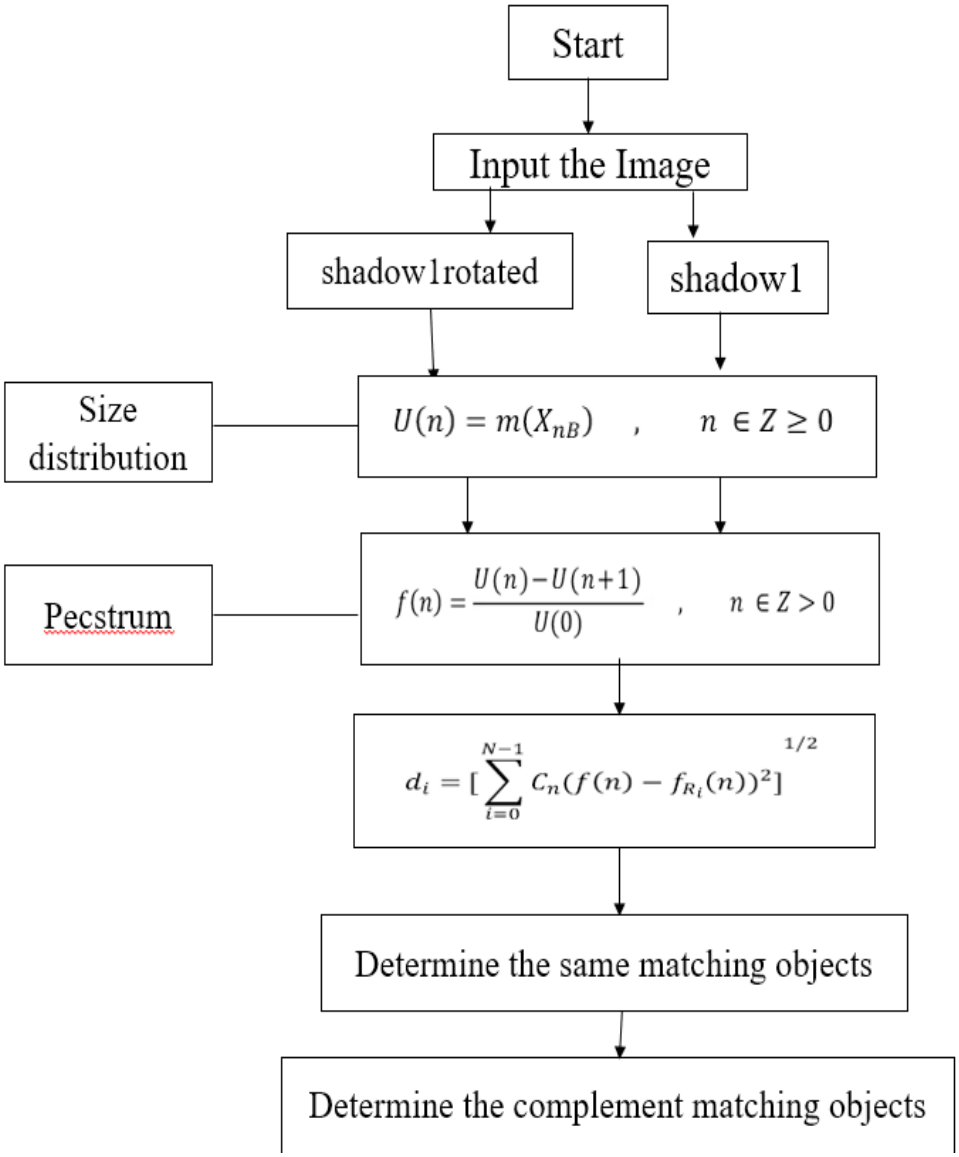
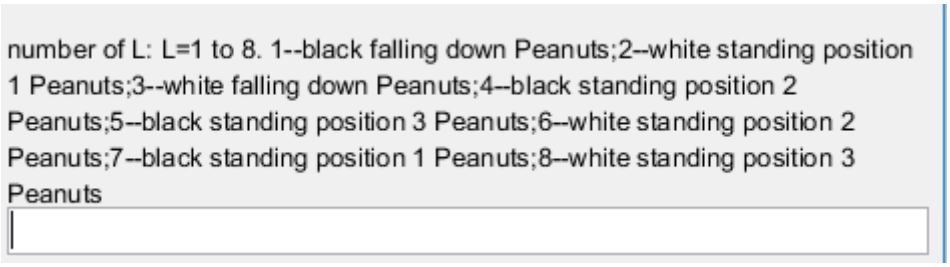


Figure 3. Algorithm flowchart of Part 2 (3)

As for the code for the matching problem, just input the object you want to find closely matching between “shadow1rotated.gif” and “shadow1.gif”, as mentioned when you run the code “FindingObjectMatchBest2.m”. The indication of the program is shown

below. In this question, you just need to enter L=2, 3, 6,8 four numbers to find the solid (black) Peanuts in “shawdow1”.In the following parts , the black means solid in the question.



After you choose the object you want to find, the program will give you the distance of different values of L in “shadow1” when the object in “shadow1rotated” is certained, and the result of finding the object in “shadow1” will appear automatically. In our algorithm, we used the Cn={1,0.8,0.6,0.4,0.2,0.1,0.1,0.1,0.1,0.1}.

C. Results

Part 1 (Morphological Skeleton)

Figure 4. ‘penn256’ original figure

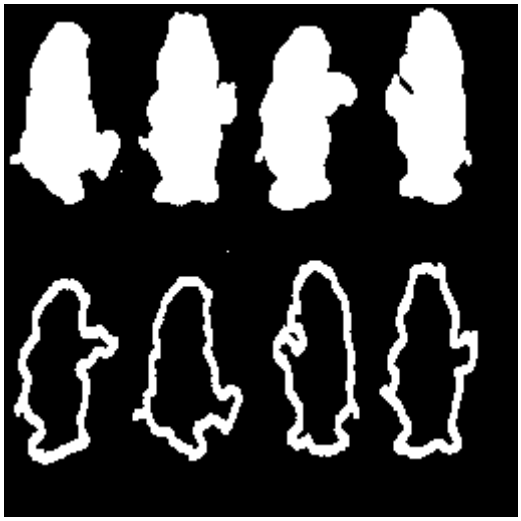


Figure 5. ‘bear’ original figure

Figure 4 and Figure 5 show the original figures used for morphological skeleton. The following figures 6 to 8 are the morphological skeletons of “penn256” using the Square, Rhombus, and VEC045 structuring elements.

Figure 6. Morphological skeleton with 3x3 square structuring element for penn256

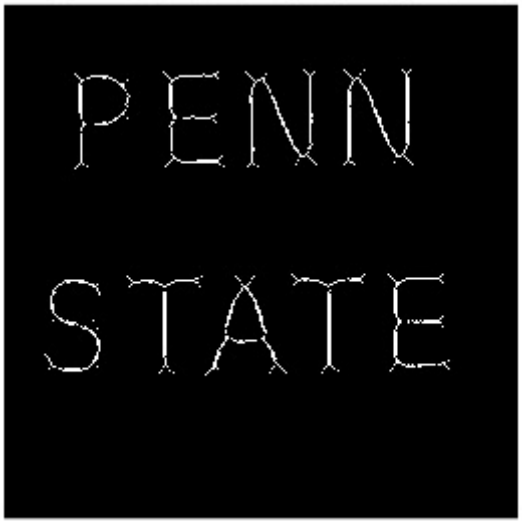


Figure 7. Morphological skeleton with RHOMBUS structuring element for penn256

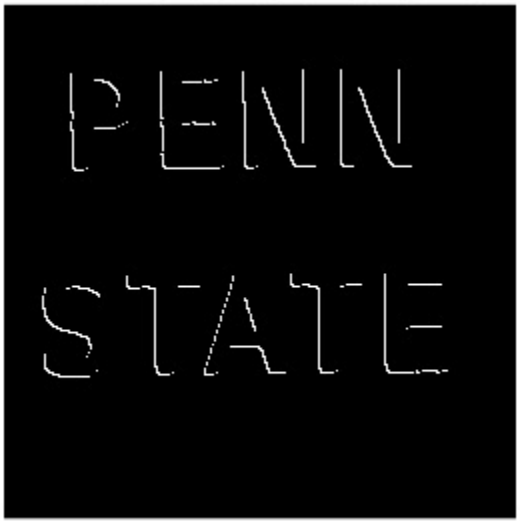


Figure 8. Morphological skeleton with VEC045 structuring element for penn256

The VEC045 structuring element has a cleaner morphological skeleton because the structuring element can have more precision while the radius increases. However, due to it not being symmetric; the morphological skeleton translates. The following figures 9 to 11 are the morphological skeletons of “bear” using the Square, Rhombus and VEC045 structuring elements.

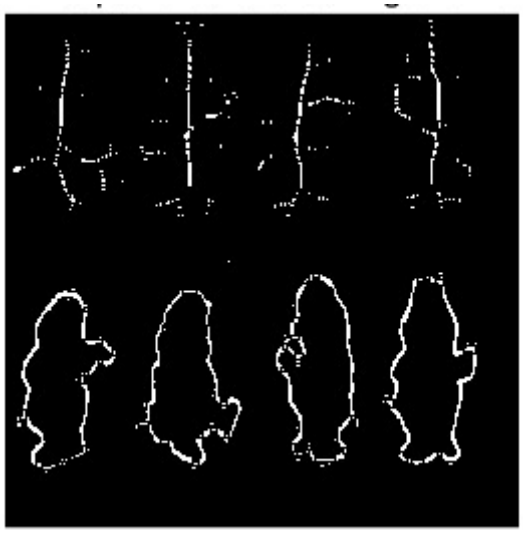


Figure 9. Morphological skeleton with3x3 square structuring element for ‘bear’

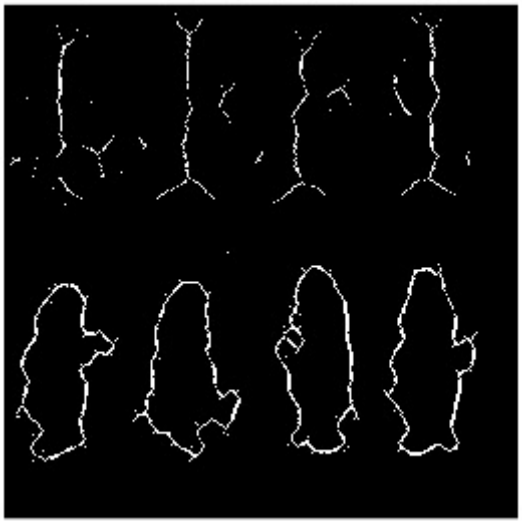


Figure 10. Morphological skeleton with RHOMBUS structuring element for ‘bear’

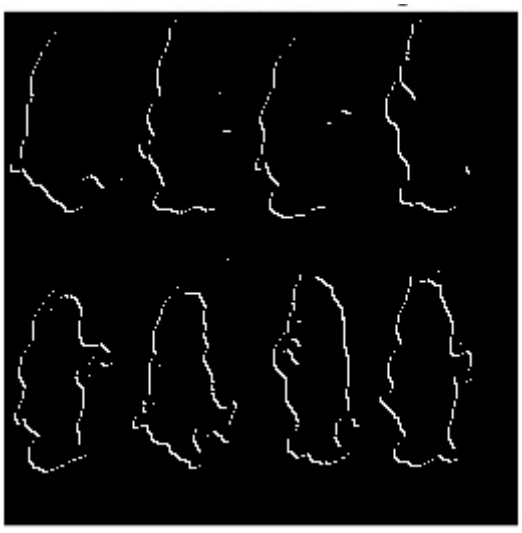


Figure 11. Morphological skeleton with VEC045 structuring element for ‘bear’

The VEC045 structuring element has a cleaner morphological skeleton because the structuring element can have more precision while the radius increases. However, due to it not being symmetric; the morphological skeleton translates. The following figures 12-14 show the partial reconstruction of the “penn256.gif ” using the square structuring element. The original skeletons are included in every partial reconstruction for penn256.gif.

Figure 12. Reconstruction with a 3x3 square as structuring element for penn256 of level 2

Figure 13. Reconstruction with a 3x3 square as structuring element for penn256 of level 3

Figure 14. Reconstruction with a 3x3 aquare as structuring element for penn256 of level 4

The following figures 15-17 show the partial reconstruction of the “penn256.gif” using the Rhombus structuring element. The original skeletons are included in every partial reconstruction for penn256.gif.

Figure 15 Reconstruction with Rhombus as structuring element for penn256 of level 2

Figure 16 Reconstruction with Rhombus as structuring element for penn256 of level 3

Figure 17. Reconstruction with Rhombus as structuring element for penn256 of level 4

The following figures 18-20 show the partial reconstructions of the penn256.gif image with the VEC045 structuring element.The original skeletons are included in every partial reconstruction for penn256.gif.

Figure 18. Reconstruction with VEC045 as structuring element for penn256 of level 2

Figure 19. Reconstruction with VEC045 as structuring element for penn256 of level 3

Figure 20. Reconstruction with VEC045 as structuring element for penn256 of level 4

The next set of figures will show the same partial reconstructions, but with the bear.gif image instead. Figures 21-23 shows the partial reconstruction with the square structuring element. The original skeletons are included in every partial reconstruction for bear.gif.

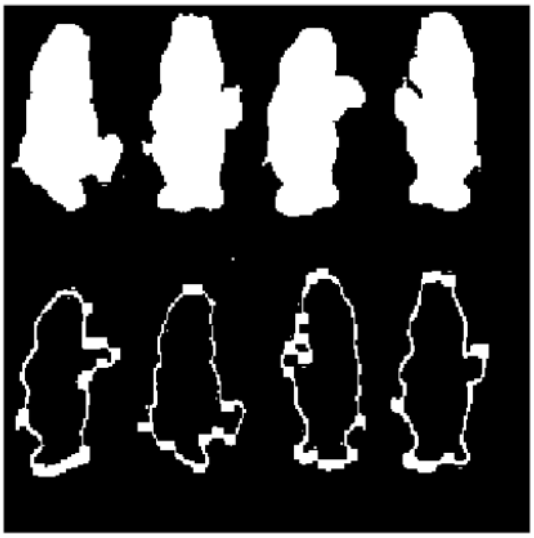


Figure 21. Reconstruction with 3x3 square as structuring element for bear of level 2

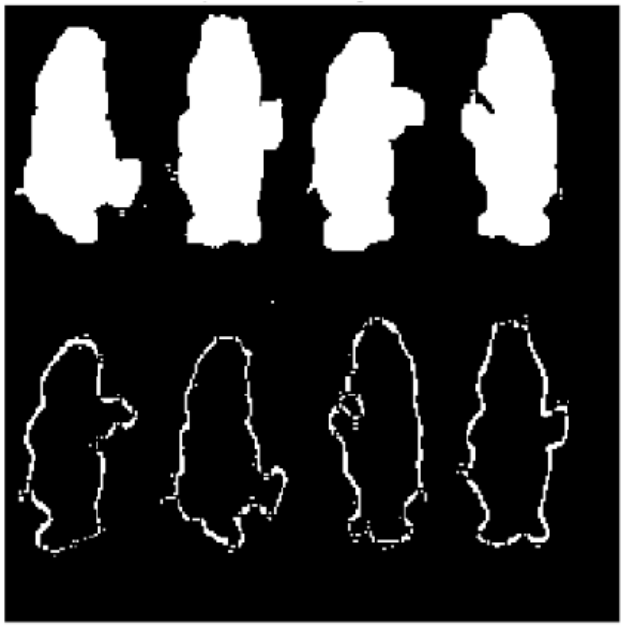


Figure 22. Reconstruction with 3x3 square as structuring element for bear of level 3

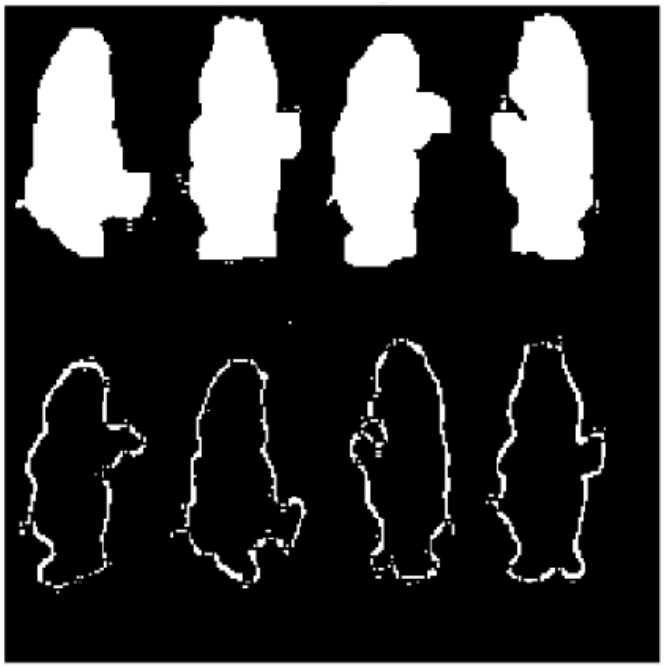


Figure 23. Reconstruction with 3x3 square as structuring element for bear of level 4

The following figures 24-26 show the partial reconstruction of the “bear.gif ” using the rhombus structuring element. The original skeletons are included in every partial reconstruction for bear.gif.

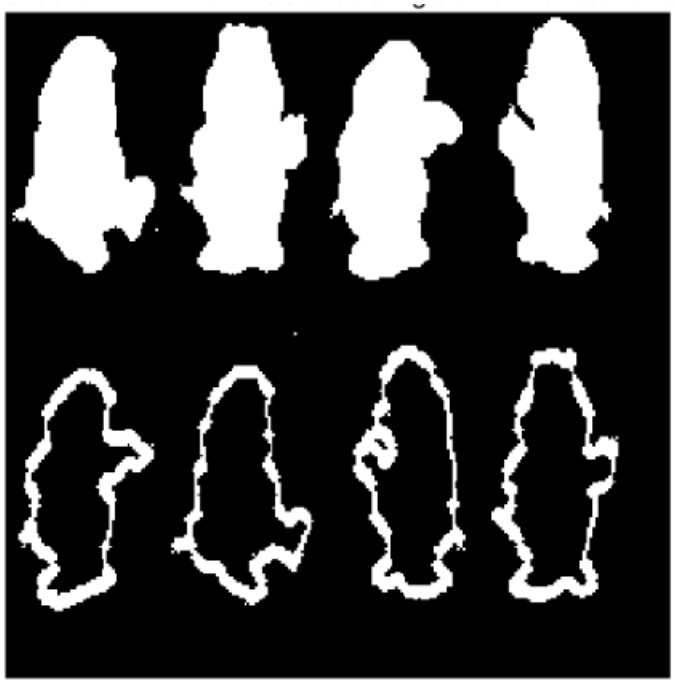


Figure 24. Reconstruction with rhombus as structuring element for bear of level 2

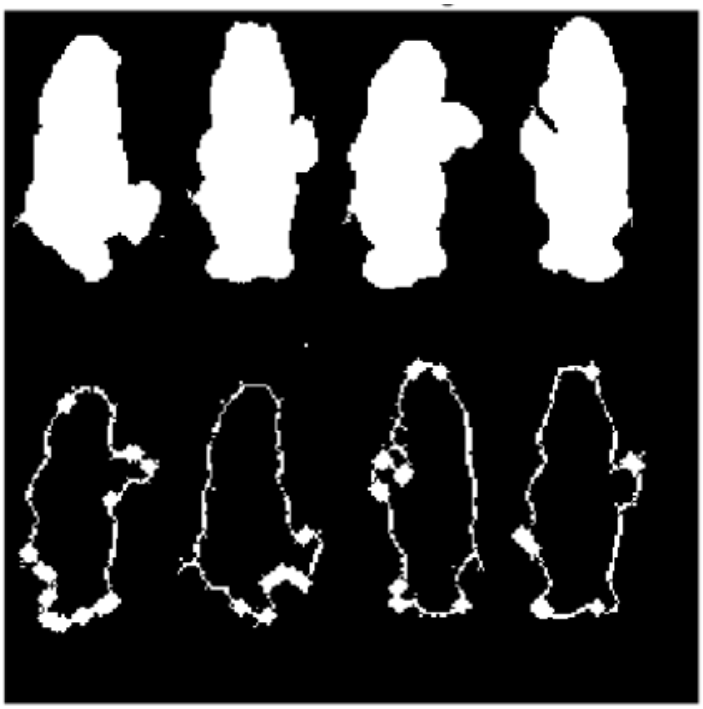


Figure 25. Reconstruction with rhombus as structuring element for bear of level 3

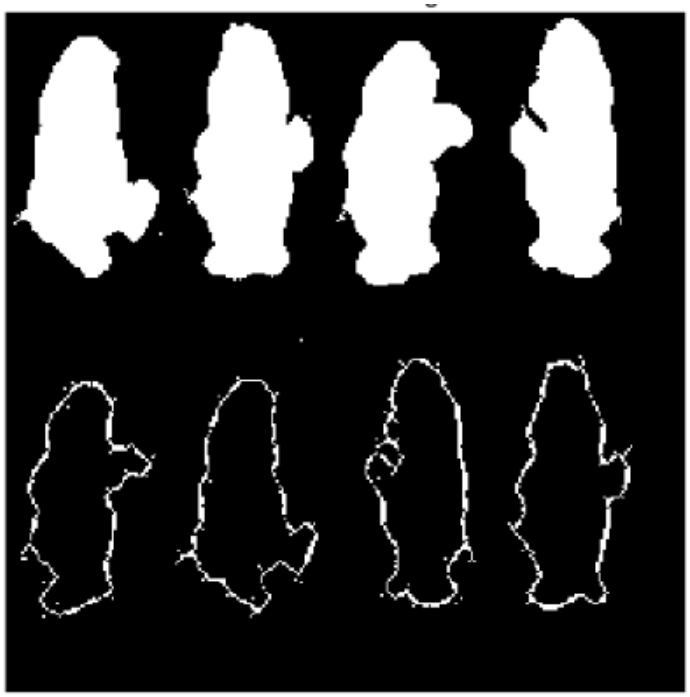


Figure 26. Reconstruction with rhombus as structuring element for bear of level 4

The following figures 27-29 show the partial reconstruction of the “bear.gif ” using the VEC045 structuring element. The original skeletons are included in every partial reconstruction for bear.gif.

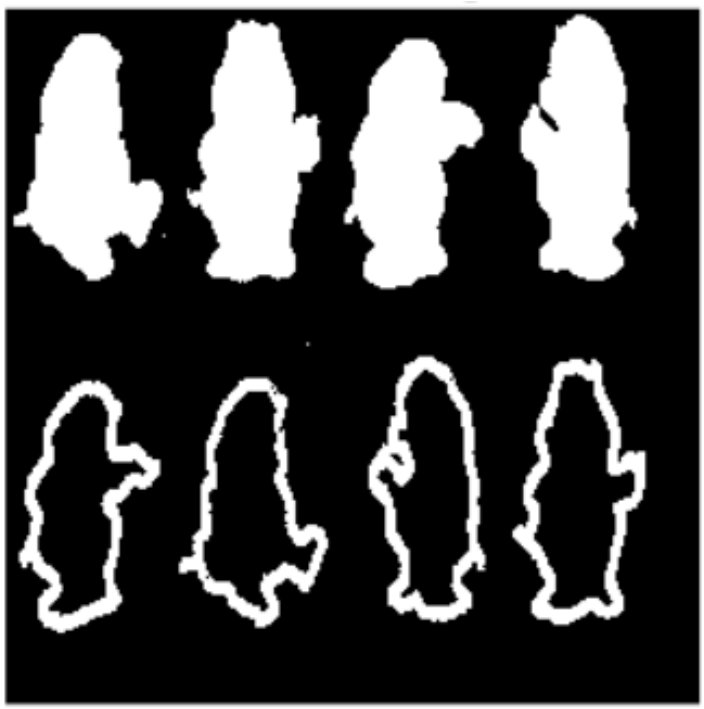


Figure 27. Reconstruction with VEC045 as structuring element for bear of level 2

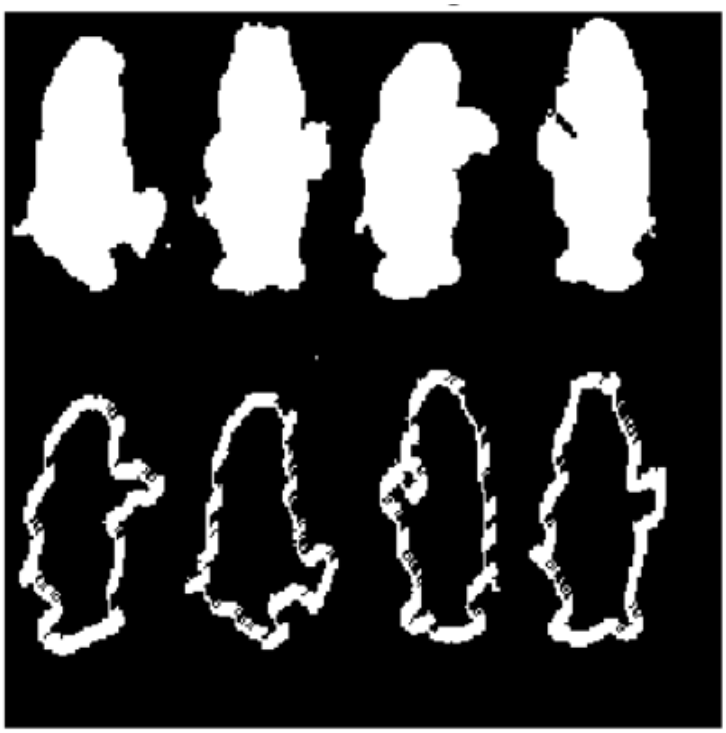


Figure 28. Reconstruction with VEC045 as structuring element for bear of level 3

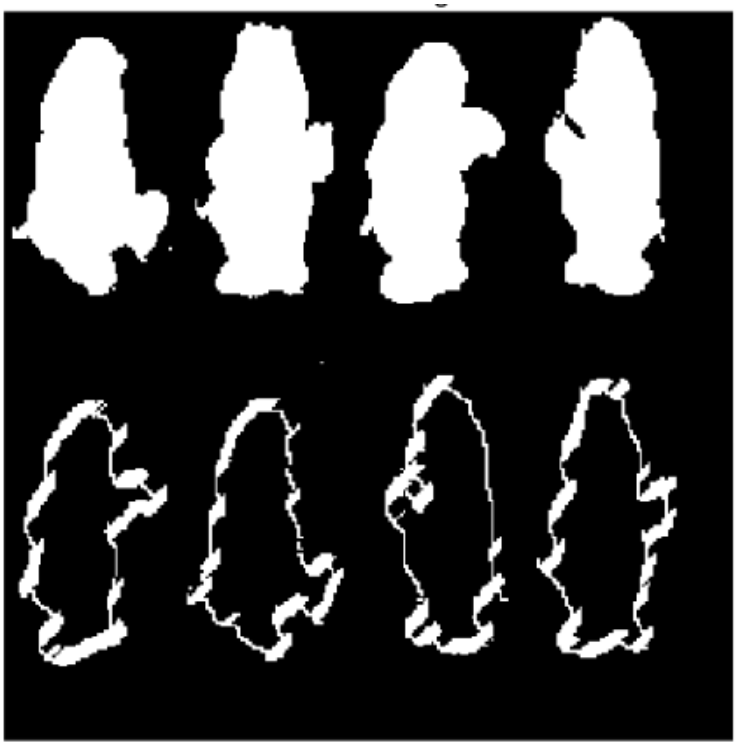


Figure 29. Reconstruction with VEC045 as structuring element for bear of level 4

The following figures 27-29 show the partial reconstructions of the penn256.gif image with the VEC045 structuring element. Due to the small size of the structuring element, the computational complexity is quite high. We can also see the side effect of the asymmetric structuring element. In this case, the skeleton is translated and partial reconstructed is translated the same amount again.

Part 2 (Shape Analysis)

The Figure 30 is the original “match1.gif” with the four objects (clover, spade, steer,and plane).

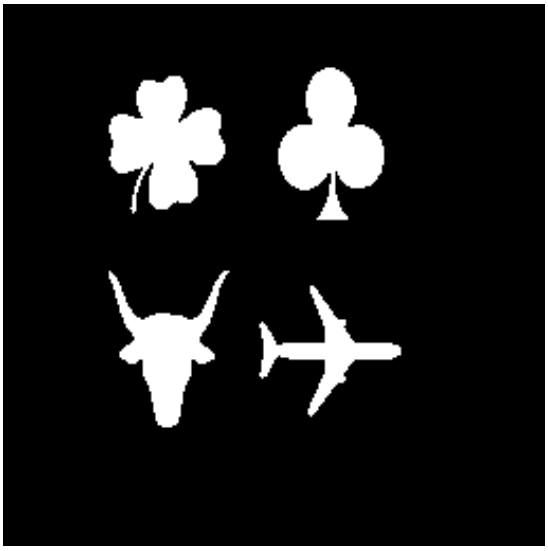


Figure 30. The original “match1.gif” with the four objects (clover, spade, steer,and plane)

The following figures 31-34 are the four isolated objects shown after the program executed.

Figure 31. Isolated Clover

Figure 32. Isolated Spade

Figure 33. Isolated Steer

Figure 34. Isolated Plane

The following figures 35-38 are the four size distribution of four isolated objects. In the figure of size distributions, the x-axis represents side length of Square SE and the y-axis is the area. Because the R is discrete, the figure is the a figure of points.

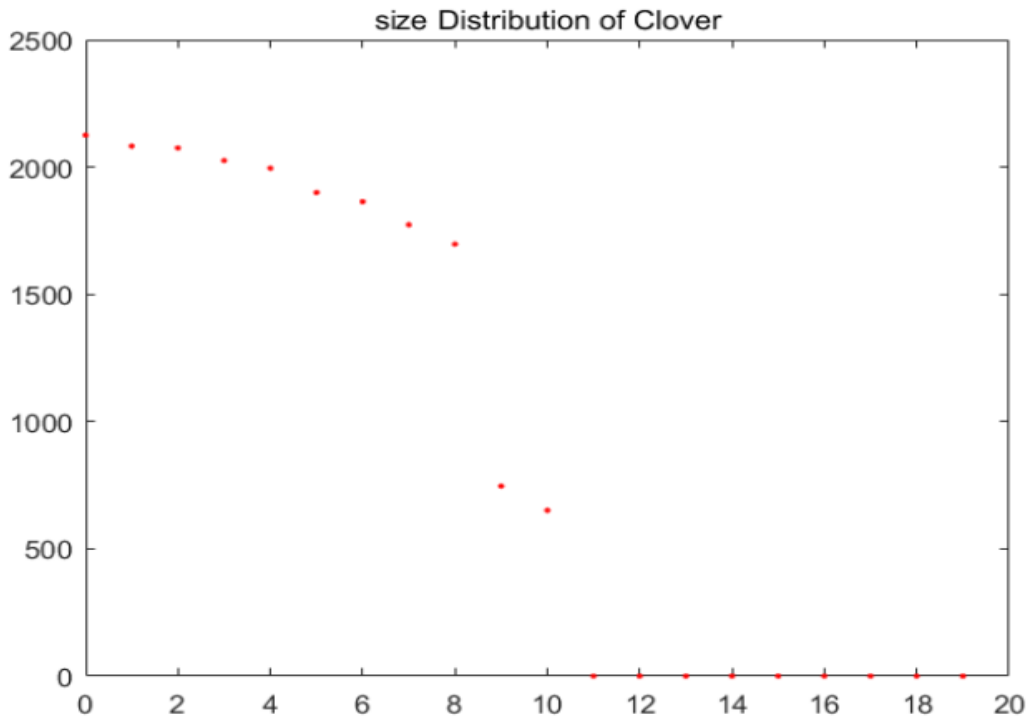


Figure 35. Size distribution of Clover

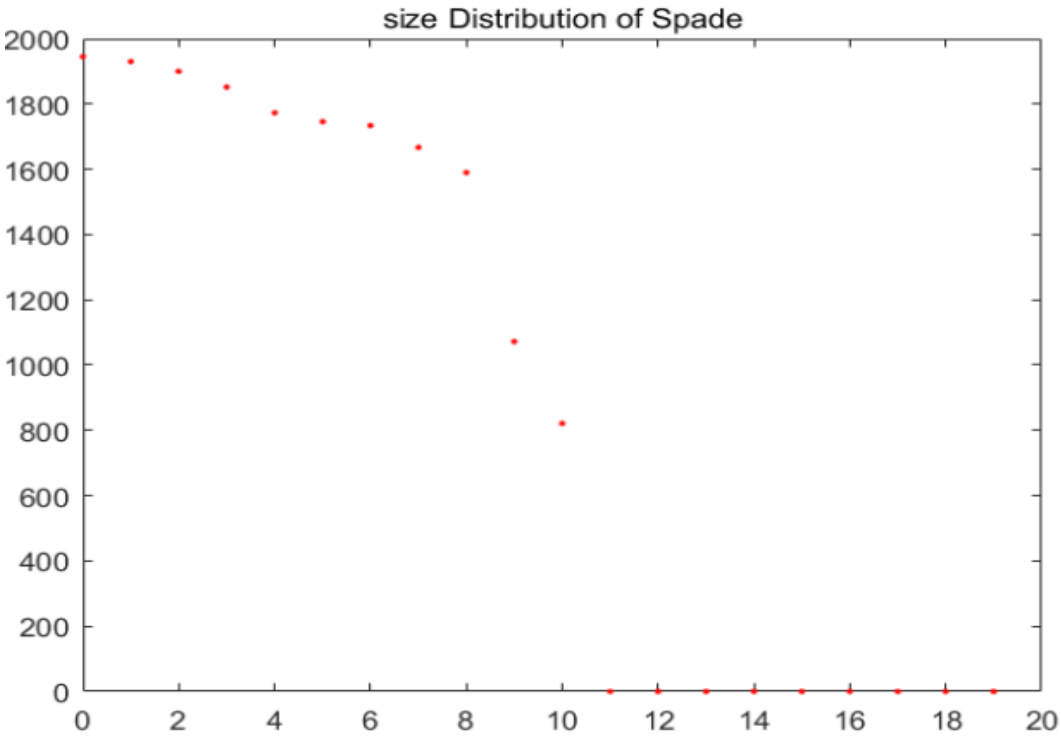


Figure 36. Size distribution of Spade

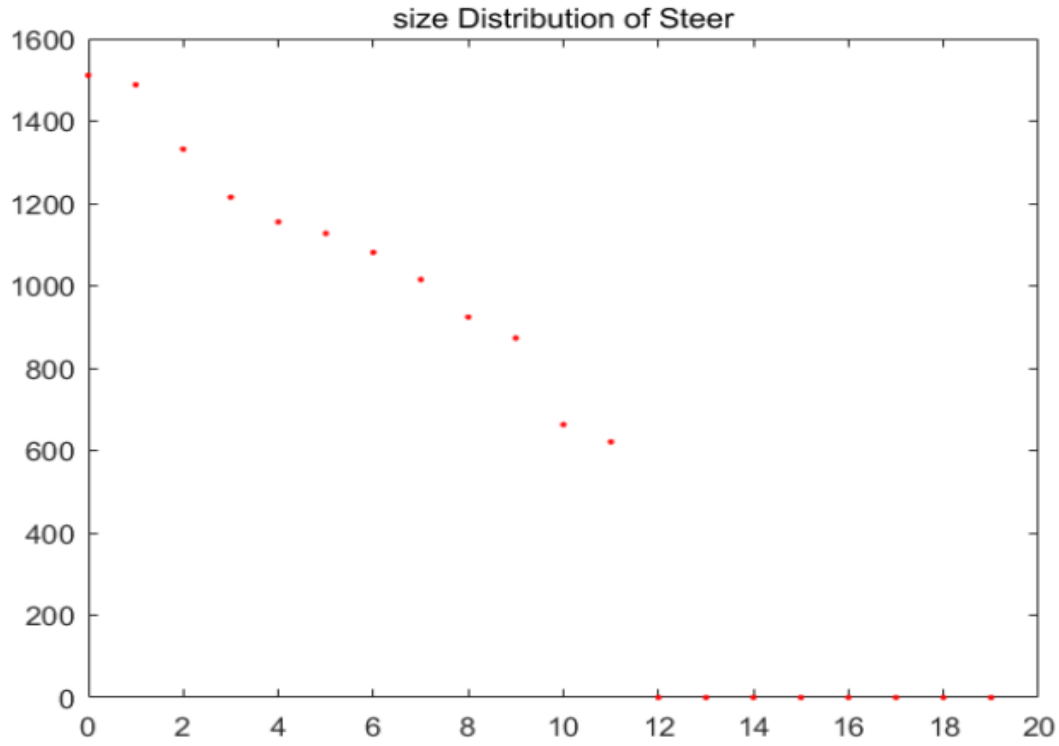


Figure 37. Size distribution of Steer

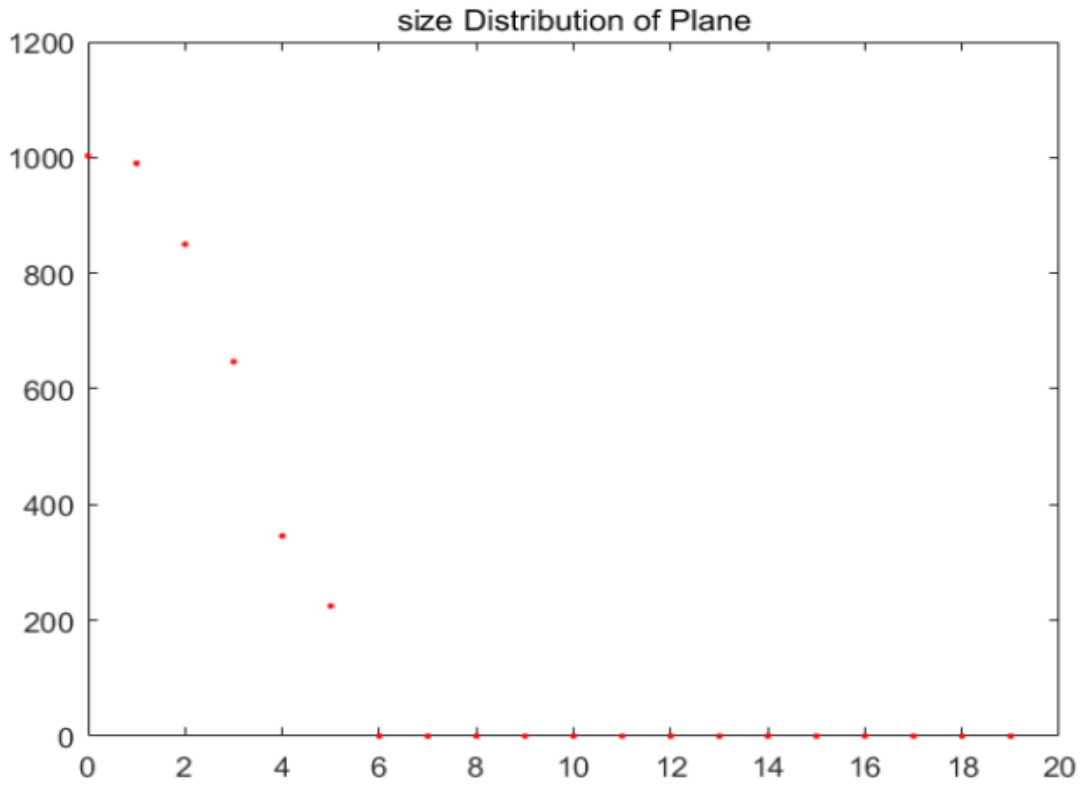


Figure 38. Size distribution of Plane

The following figures 39-42 are the four pectrums of the four isolated objects. In the figure of pectrum, x-axis is side length of Square SE and the y-axis is the Pectrum.

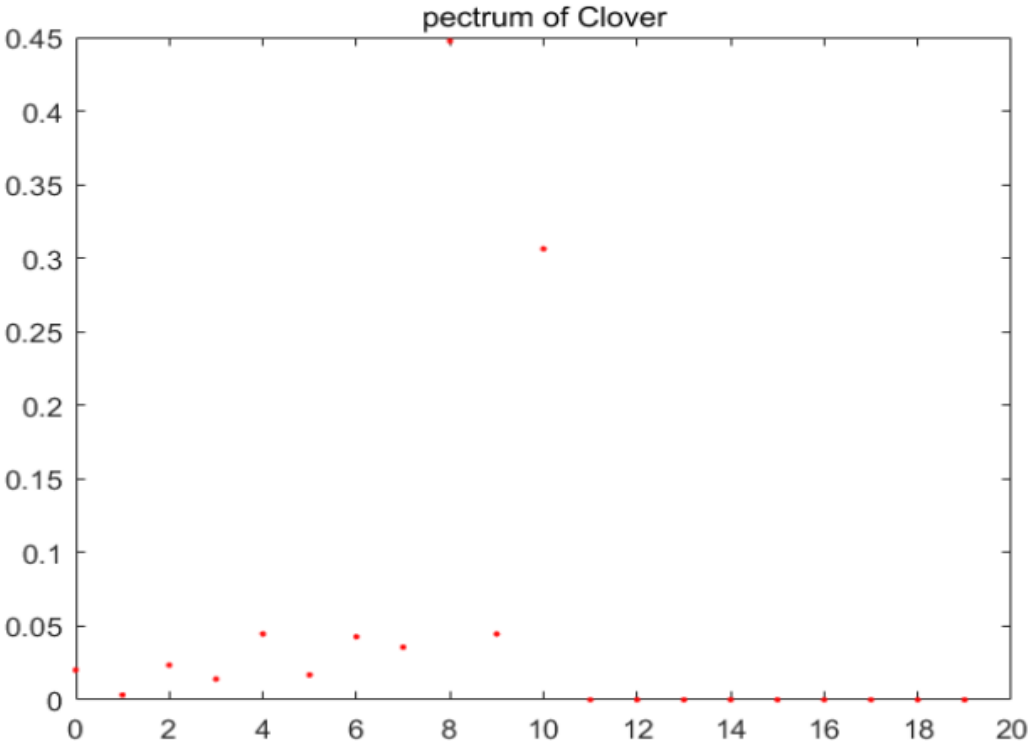


Figure 39. Pectrum of Clover

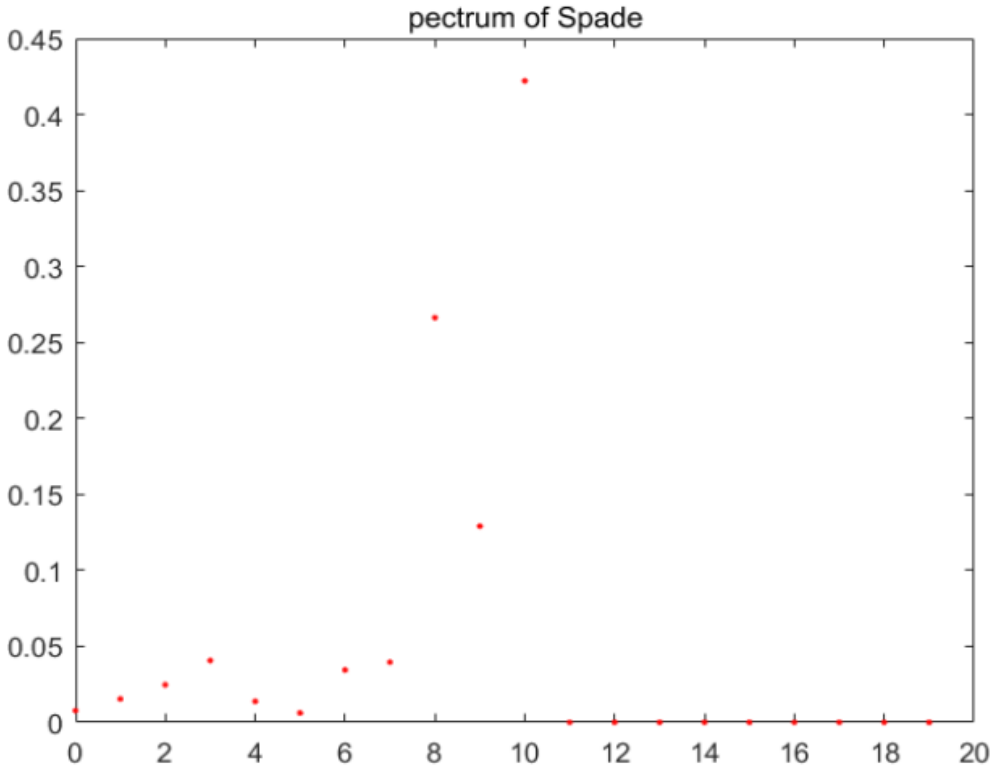


Figure 40. Pectrum of Spade

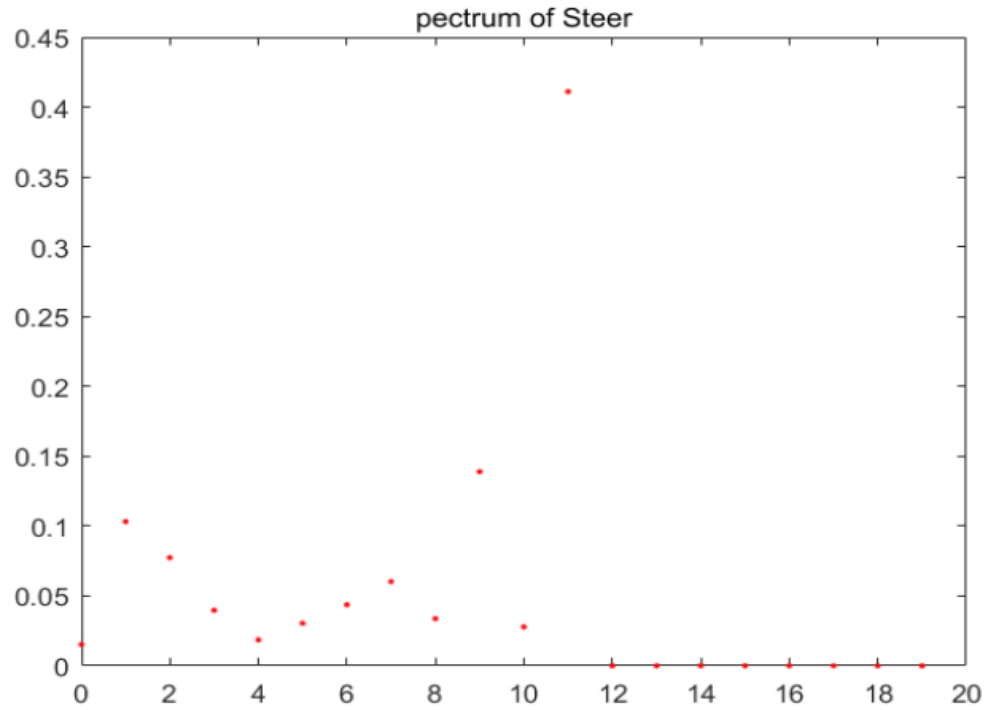


Figure 41. Pectrum of Steer

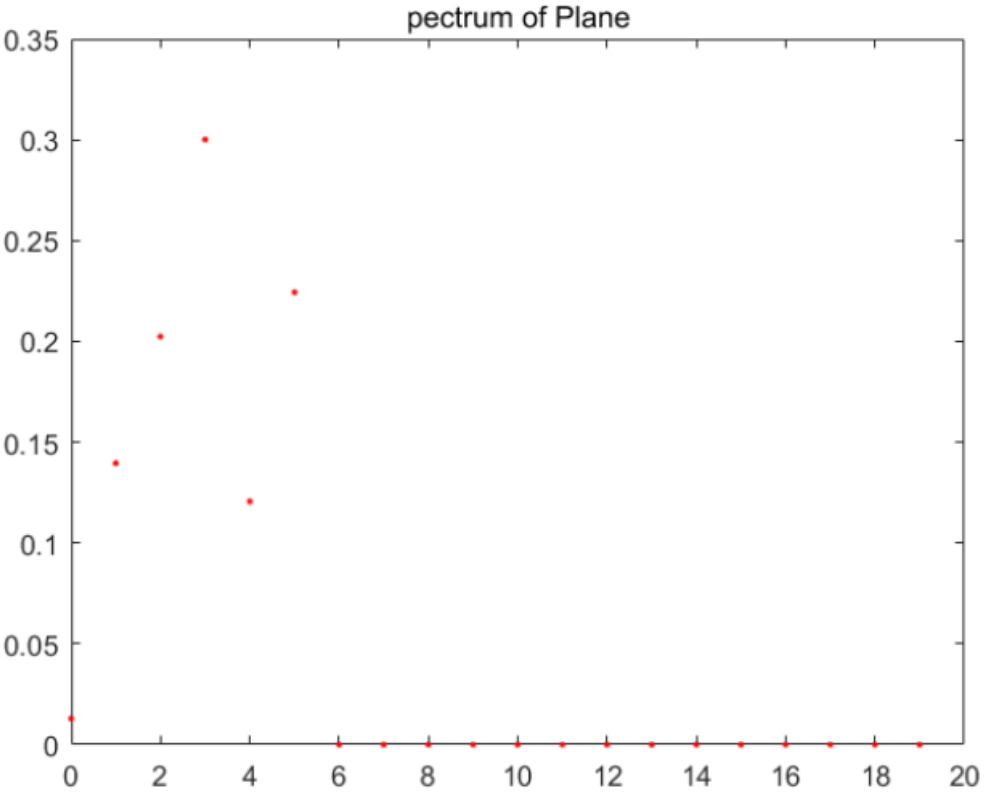


Figure 42. Pectrum of Plane

Figure 43 shows the complexity that was computed. A higher number means the object is more complex, so the steer was the most complex

object. This makes sense because the steer has a lot of small details, as well as larger components. The spade, clover, and plane were about the same complexity (in comparison to the steer).

Name	Complexity
'Clover'	0.68156
'Steer'	0.85304
'Plane'	0.69754
'Spade'	0.71165

Figure 43. The complexity of the clover, steer, plane, and spade in "match1.gif"

As for finding the best matching object in "match3.gif", here are some figures of the process. Figure 44 is The original "match3.gif" with the four objects (clover, spade, steer,and plane).

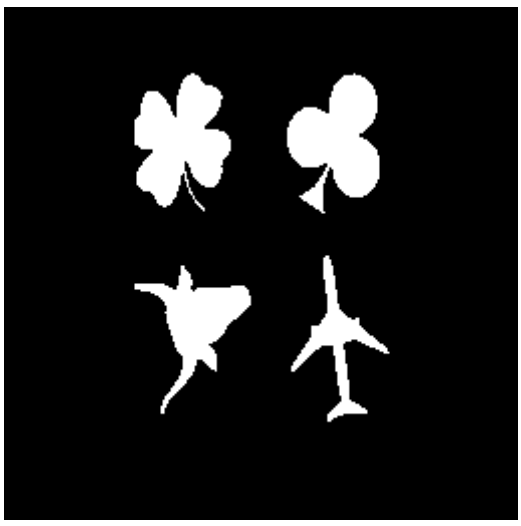


Figure 44. The original "match3.gif" with the four objects (clover, spade, steer,and plane)

The following figures 45-47 are the pectrums of object Clover in “match3”, the distance of different values of L in “match3” when the object in “match1” is Clover, and the result of finding Clover in “match 3”.

For the L=1 in “match1”, which means Clover in “match1”, we first do the loop for L=1 to L=4 in “match3” then, we can get the distance in Figure 46. Based on the figure 46, we can see that L=1 in “match3” has the smallest distance which mean the object corresponding to L=1 in the “match3” best matches Clover in “match1”. The program will automatically find the L=1 as the best matching object in “match3” so that figure 47 is presented, and the pectrum of the Clover in “match3” is also shown in figure 45. We can use this method to explain figure 48-56.

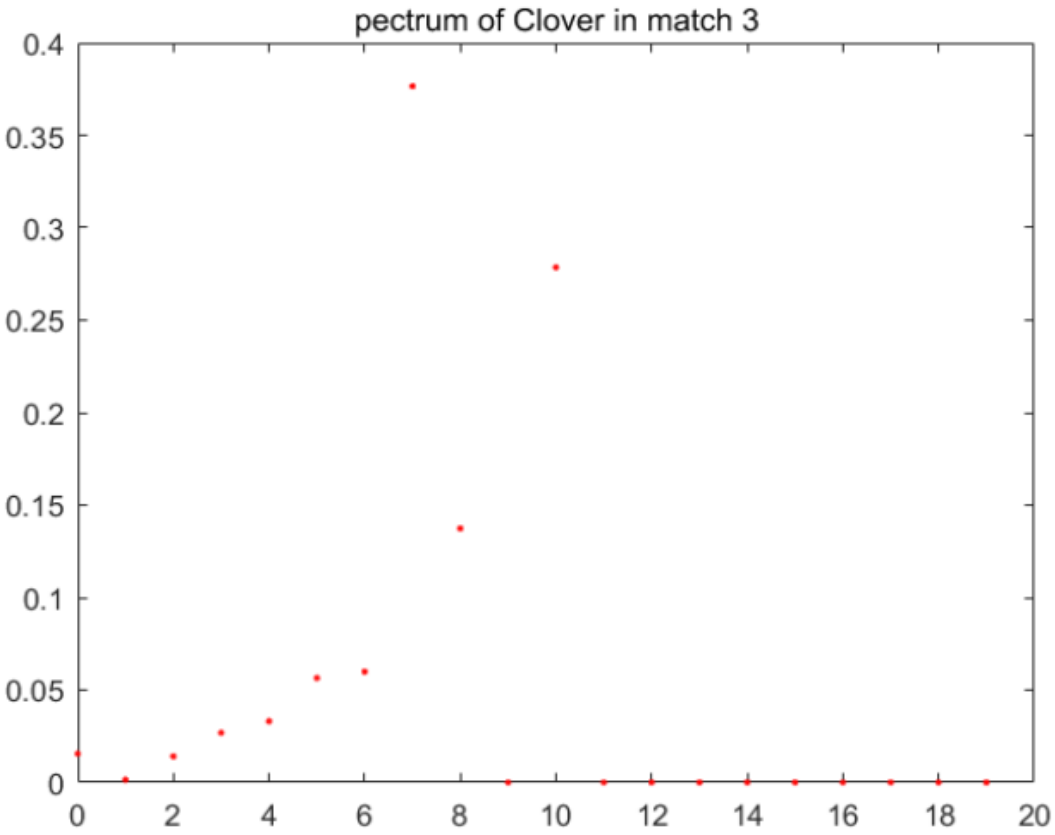


Figure 45. The pectrum of object Clover in “match3”

Name	distance
'L=1'	0.26466
'L=2'	0.29558
'L=3'	0.27802
'L=4'	0.75119

Figure 46. The distance of different value of L in “match3” when the object in “match1” is Clover

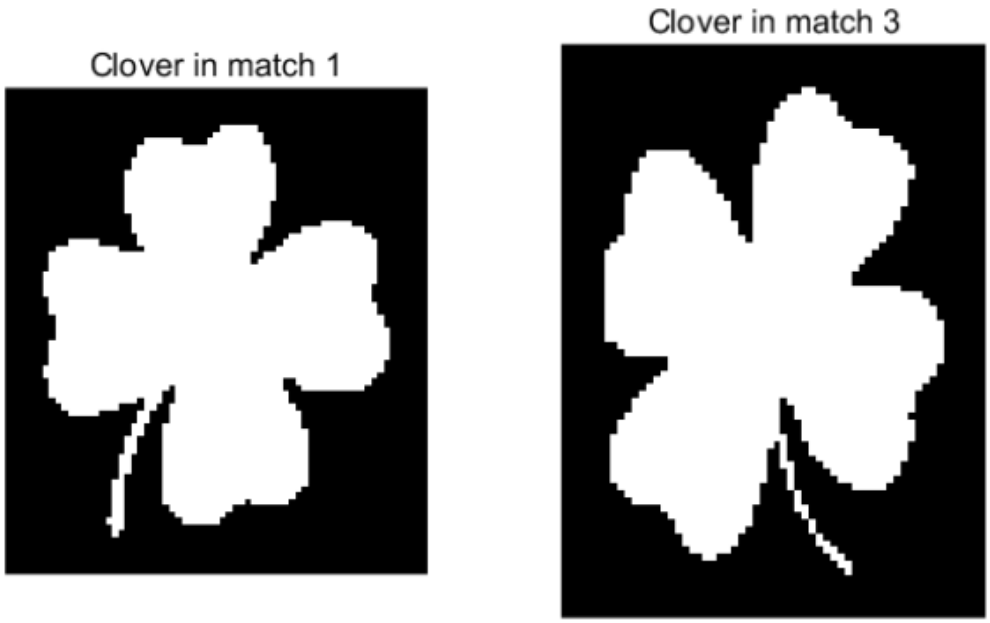


Figure 47. The result of finding Clover in “match3”

The following figures 48-50 are the pectrums of object Steer in “match3”, the distance of different value of L in “match 3” when the object in “match1” is Steer, and the result of finding Steer in “match3”.

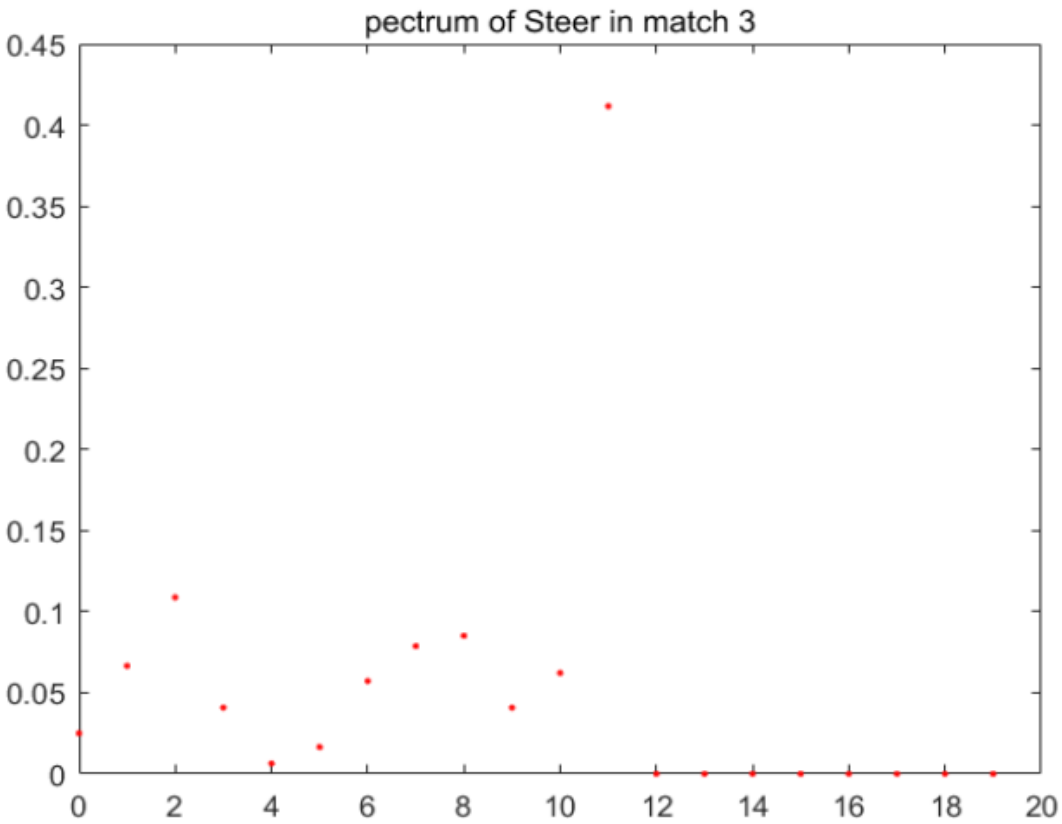


Figure 48. The pectrum of object Steer in “match3”

Name	distance
'L=1'	0.34524
'L=2'	0.13491
'L=3'	0.2277
'L=4'	0.54155

Figure 49. The distance of different values of L in “match3” when the object in “match1” is Steer

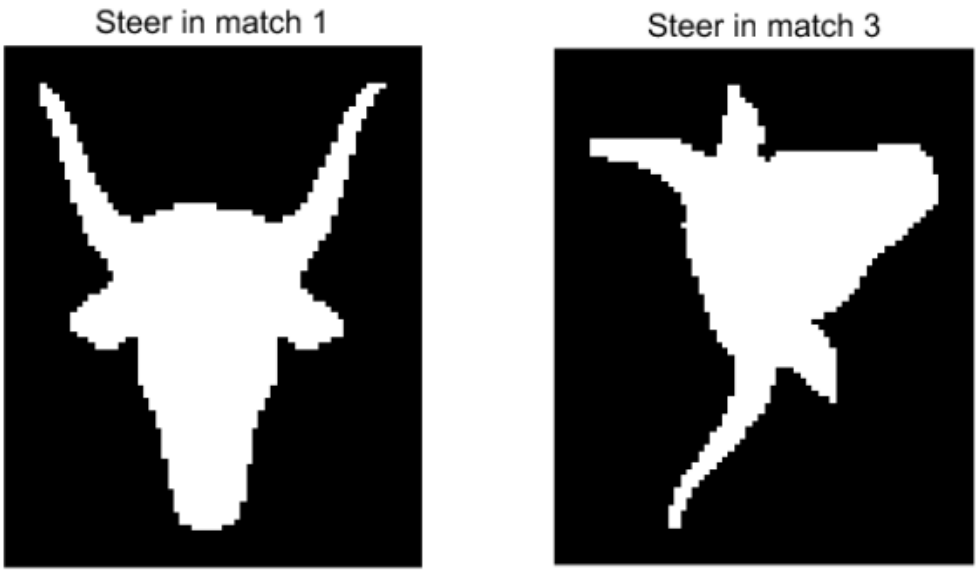


Figure 50. The result of finding Steer in “match3”

The following figures 51-53 are the pectrums of object Plane in “match3”, the distance of different value of L in “match3” when the object in “match1” is Plane and the result of finding Plane in “match3”.

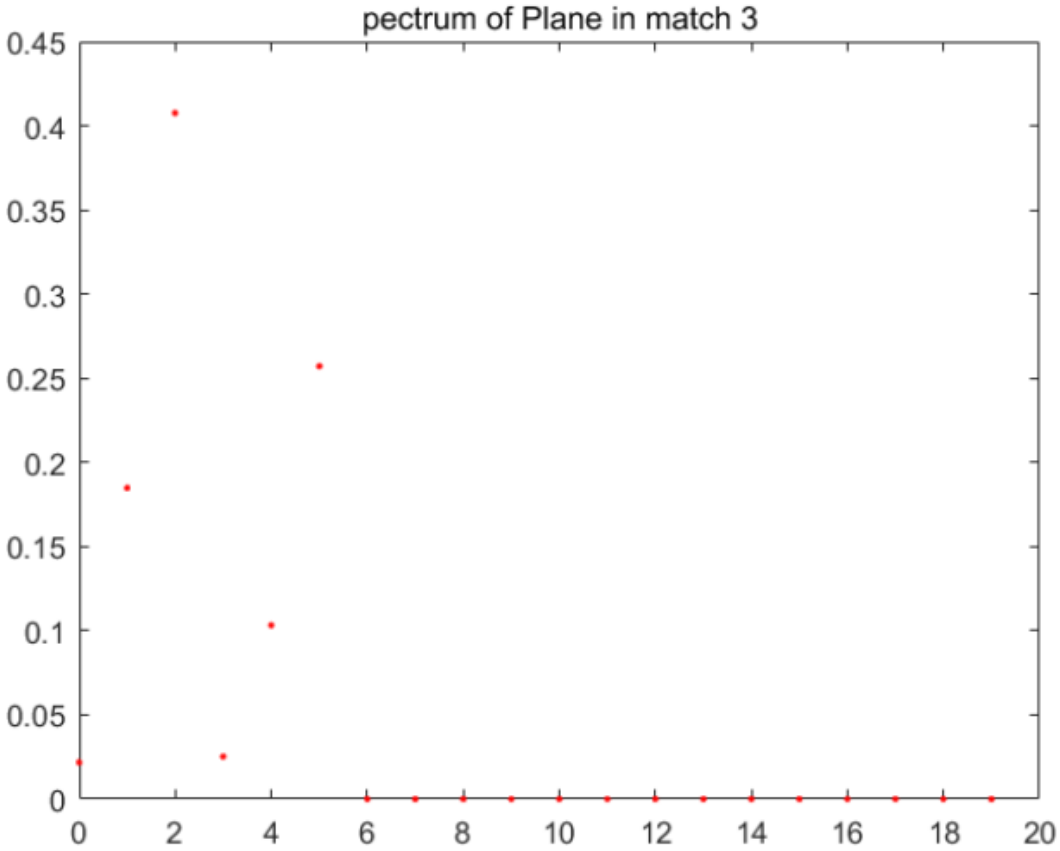


Figure 51. The pectrum of object Plane in “match3”

Name	distance
'L=1'	0.71851
'L=2'	0.51353
'L=3'	0.67973
'L=4'	0.40023

Figure 52. The distance of different value of L in “match3” when the object in “match1” is Plane

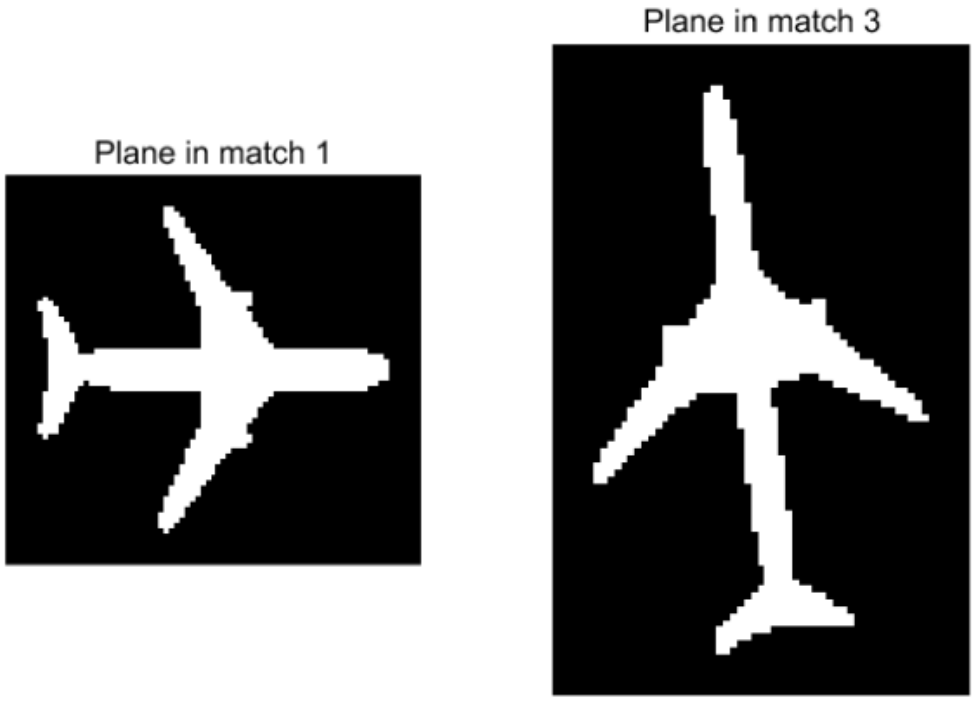


Figure 53. The result of finding Plane in “match3”

The following figures 54-56 are the pectrums of object Spade in “match3”, the distance of different value of L in “match3” when the object in “match1” is Spade and the result of finding Spade in “match3”.

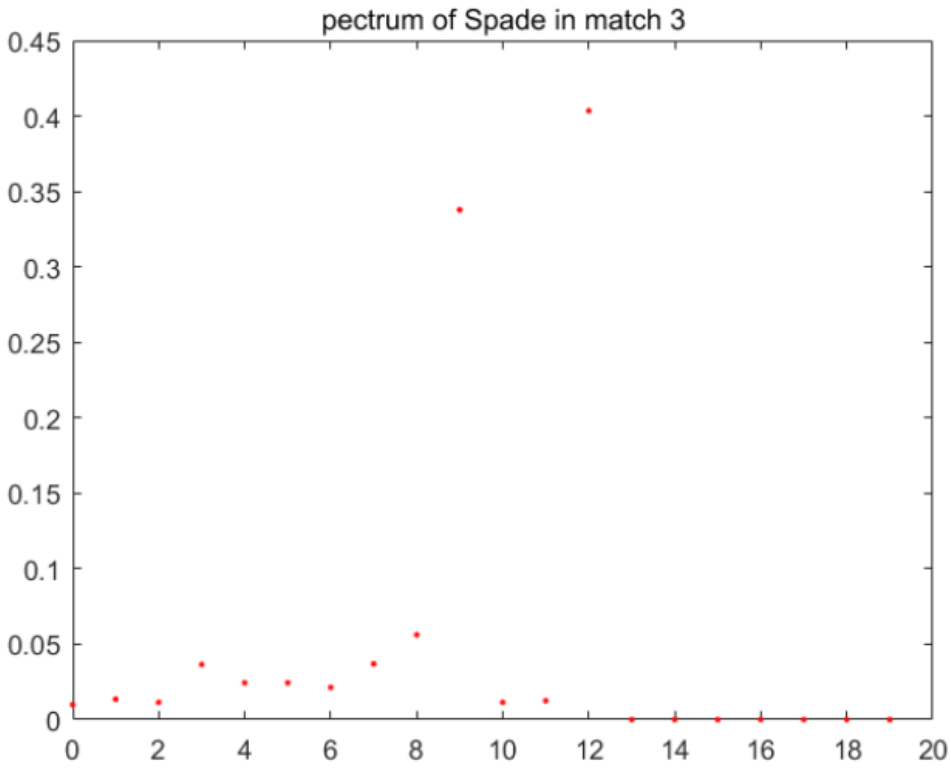


Figure 54. The pectrum of object Spade in “match3”

Name	distance
'L=1'	0.258
'L=2'	0.23974
'L=3'	0.16472
'L=4'	0.73976

Figure 55 The distance of different value of L in “match3” when the object in “match1” is Spade

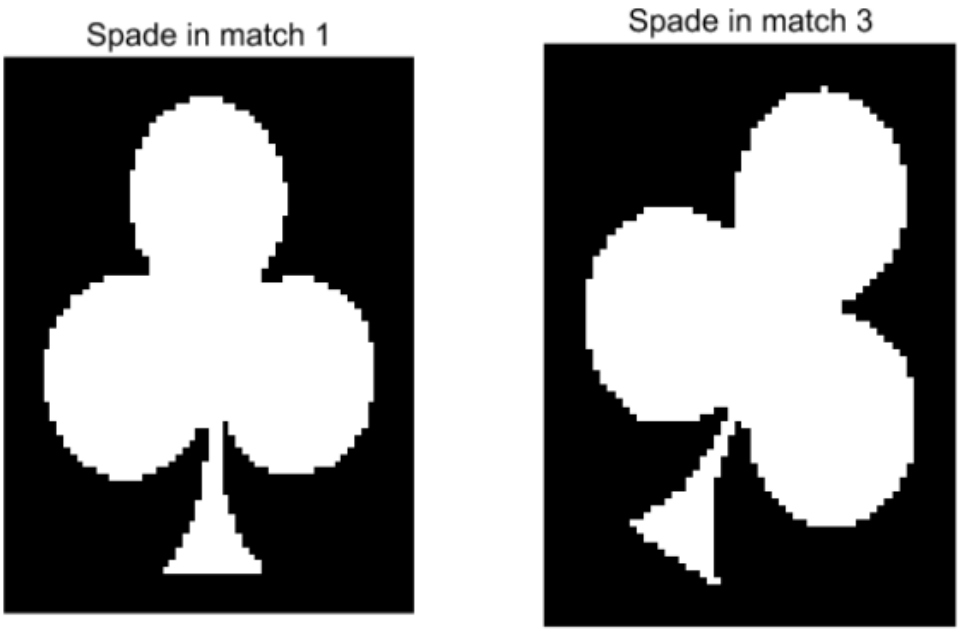


Figure 56. The result of finding Spade in “match3”

Based on figure 46,49,52,55 above, we can get the distance of different objects in “match1.gif” and “match3.gif”, as shown in Table 1. The smallest distance means the best matching object in this part.

Object Name	Distance
Clover	0.26466
Steer	0.13491
Plane	0.40023
Spade	0.16472

Table 1. The distance of different objects in “match1.gif” and “match3.gif” using a 3x3 square structuring element and a weight vector of Cn={1,0.8,0.6,0.4,0.2,0.1,0.1,0.1,0.1,0.1}

As for finding the best matching object in “shadow1.gif” that matches the object in “shadow1rotated.gif”, here are some figures of the process. Figure 57 and 58 are the original “shadow1.gif” and “shadow1rotated.gif”

with the 8 objects. In the following parts , the black means solid in the question.

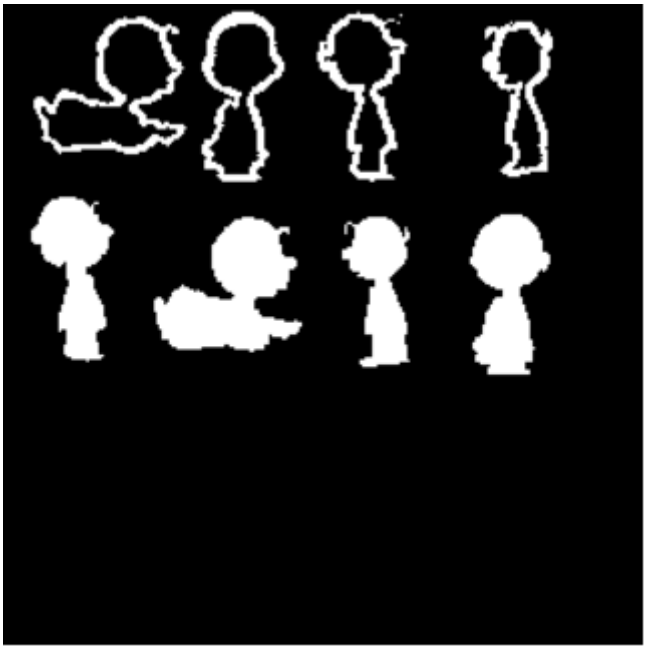


Figure 57. The original “shadow1.gif” with the 8 objects

Figure 58. The original “shadow1rotated.gif” with the 8 objects

In both figure 57 and figure 58, we named all the objects in it as followed. Figure 59 is black falling down Peanuts, and when you entered L=1 in the program, you can get it.

Figure 59. Black falling down Peanuts

Figure 60 is white standing position 1 Peanuts, and when you entered L=2 in the program, you can get it.

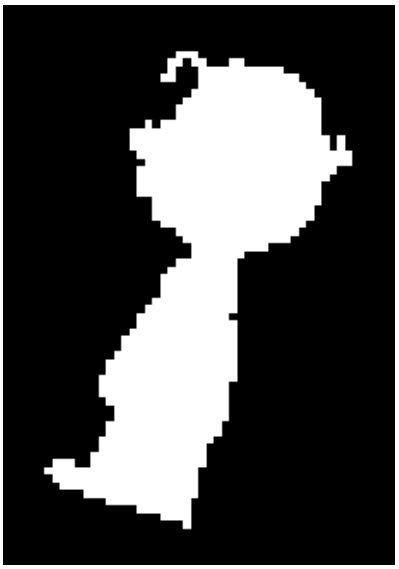


Figure 60. White standing position 1 Peanuts

Figure 61 is white falling down Peanuts, and when you entered L=3 in the program, you can get it.

Figure 61. White falling down Peanuts

Figure 62 is black standing position 2 Peanuts, and when you entered L=4 in the program, you can get it.

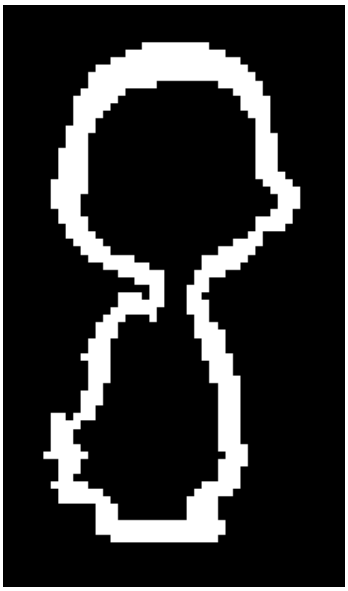


Figure 62. Black standing position 2 Peanuts

Figure 63 is black standing position 3 Peanuts, and when you entered L=5 in the program, you can get it.

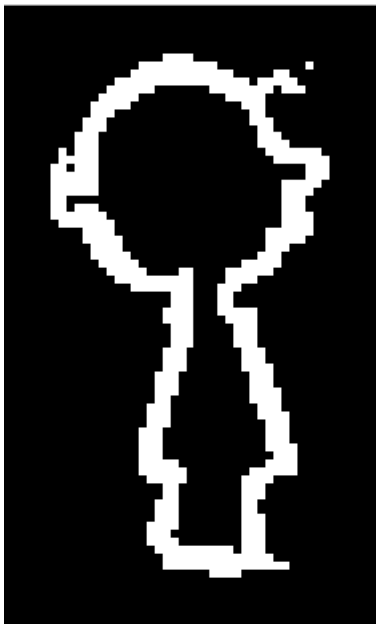


Figure 63. Black standing position 3 Peanuts

Figure 64 is white standing position 2 Peanuts, and when you entered L=6 in the program, you can get it.

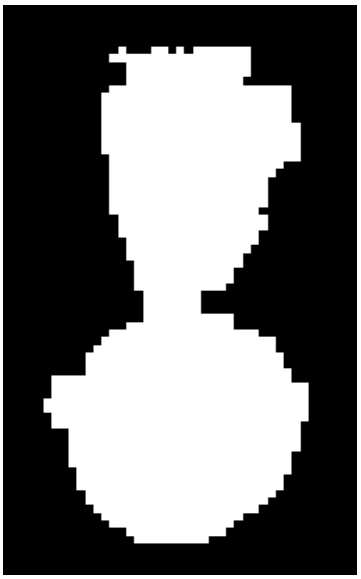


Figure 64. White standing position 2 Peanuts

Figure 65 is black standing position 1 Peanuts, and when you entered L=7 in the program, you can get it.

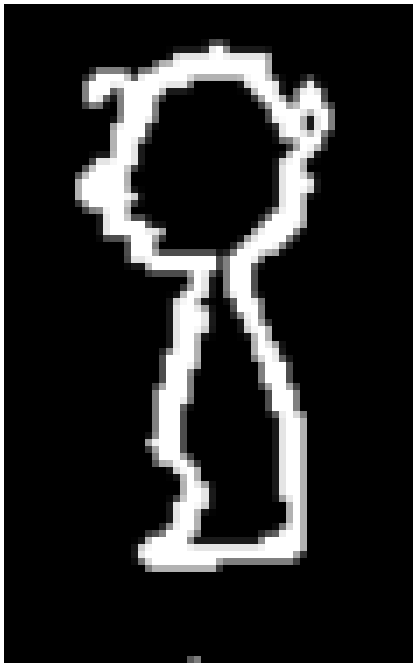


Figure 65. Black standing position 1 Peanuts

Figure 66 is white standing position 3 Peanuts, and when you entered L=8 in the program, you can get it.

Figure 66. White standing position 3 Peanuts

We can first find the links between each couple of L. Then we can find L=1 links to L=5; L=2 links to L=3; L=4 links to L=7;L=6 links to L=8. The following figures 67-70 are the pectrums of object black falling down Peanuts in “shadow1rotated.gif” and “shadow1.gif”, the distance of different values of L in “shadow1.gif” when the object in

“shadow1rotated.gif” is black falling down Peanuts, the result of finding black falling down Peanuts in “shadow1” and the linking complementary white falling down Peanuts in “shawdow1”.

For the L=1 in “shadow1rotated”, which means black falling down Peanuts in “shadow1rotated”, we first do the loop for L=1 to L=8 in “shadow1” then, we can get the distance in Figure 67. Based on the figure 67, we can see that L=2 in “shadow1” has the smallest distance which mean the object corresponding to L=2 in the “shadow1” best matches black falling down Peanuts in “shadow1rotated”. The program will automatically find the L=2 as the best matching object in “shadow1” so that figure 70 is presented, and the pectrum of the black falling down Peanuts in “shadow1rotated.gif” and “shadow1.gif” are also shown in figure 68 and 69. And once we find L=2, up to the links we mentioned, we can get the figure 71 as the final result of complementary object. We can use this method to explain figure

In the all the figure of pectrum below, x-axis is side length of Square SE and the y-axis is the Pectrum.

Name	distance
'L=1'	1.269
'L=2'	0.28748
'L=3'	1.1623
'L=4'	0.57777
'L=5'	0.58495
'L=6'	1.2326
'L=7'	0.91493
'L=8'	0.95161

Figure 67. When L=1 in shawdow1rotated the distance of the value of L in shawdow1

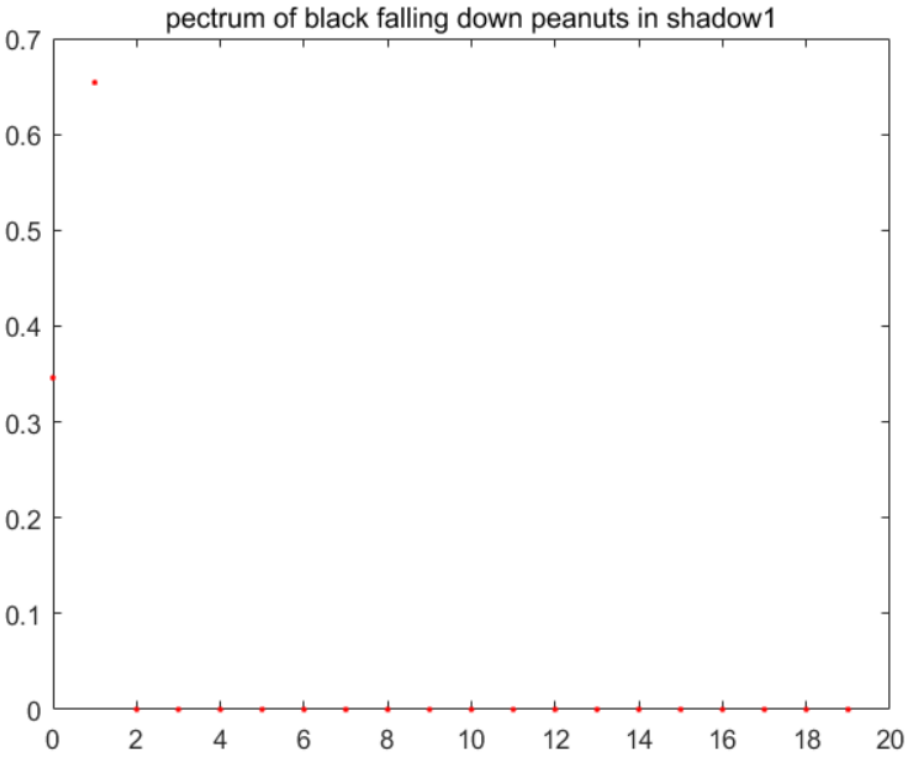


Figure 68. Pectrum of black falling down peanuts in shadow1

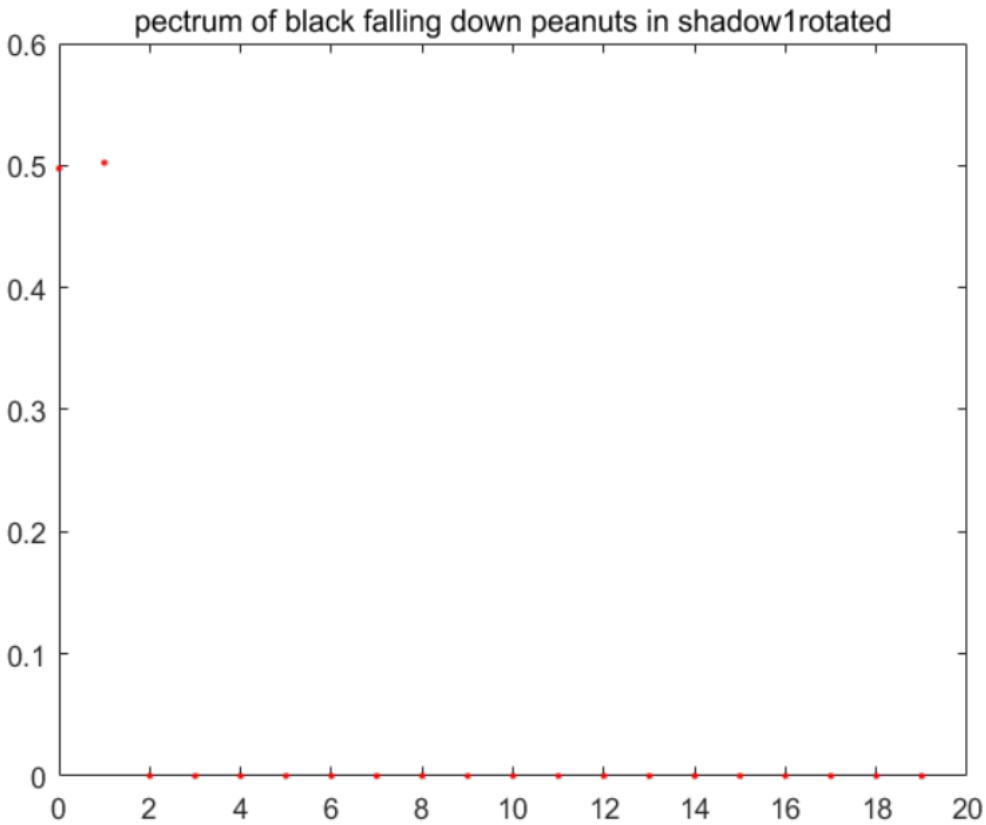


Figure 69. Pectrum of black falling down peanuts in shadow1rotated

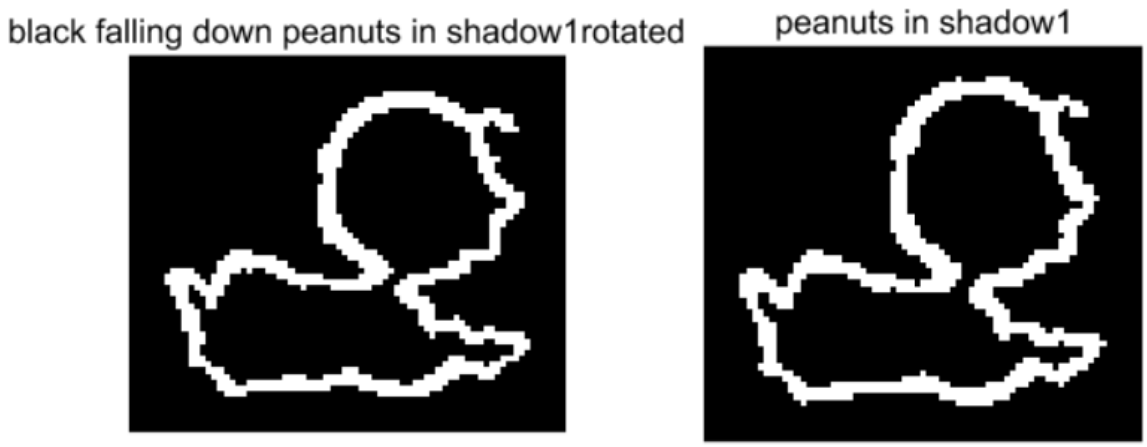


Figure 70. The result of finding black falling down Peanuts in “shawdow1”

Figure 71. The complementary result of finding black falling down Peanuts in “shawdow1”

The following figures 72-76 are the pectrums of object white standing position 1 Peanuts in “shadow1rotated.gif” and “shadow1.gif” , the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is white standing position 1 Peanuts, and the result and the complementary result of finding white standing position 1 Peanuts in “shadow1”.

Name	distance
'L=1'	0.3259
'L=2'	1.2035
'L=3'	0.43706
'L=4'	1.1494
'L=5'	1.1869
'L=6'	0.29736
'L=7'	1.1923
'L=8'	1.3437

Figure 72. When L=2 in shawdow1rotated the distance of the value of L in shawdow1

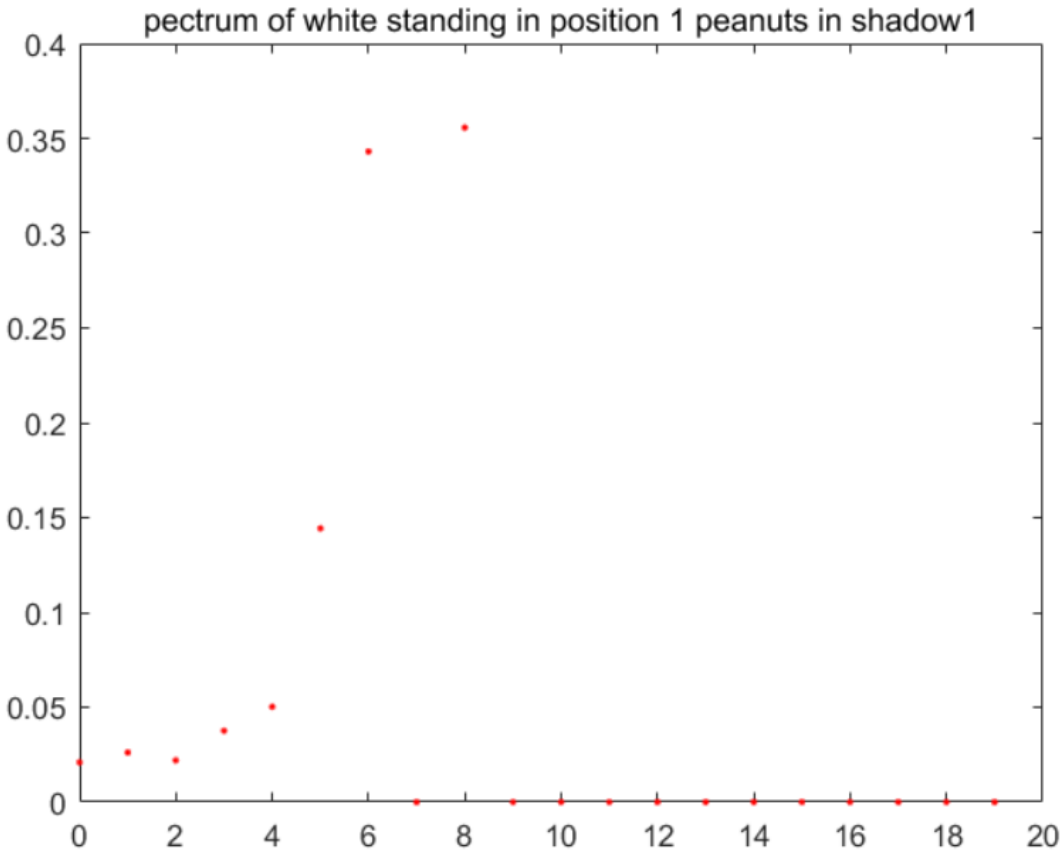


Figure 73. Pectrum of white standing position 1 peanuts in shadow1

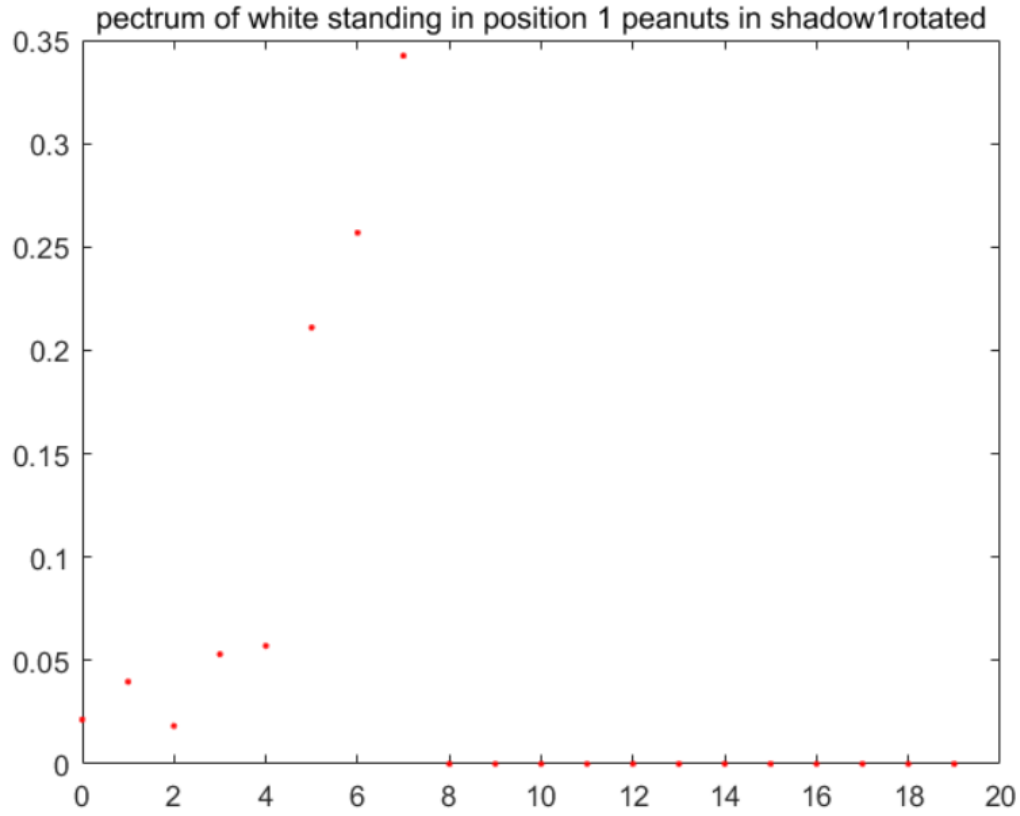


Figure 74. Pectrum of white standing position 1 peanuts in shadow1rotated

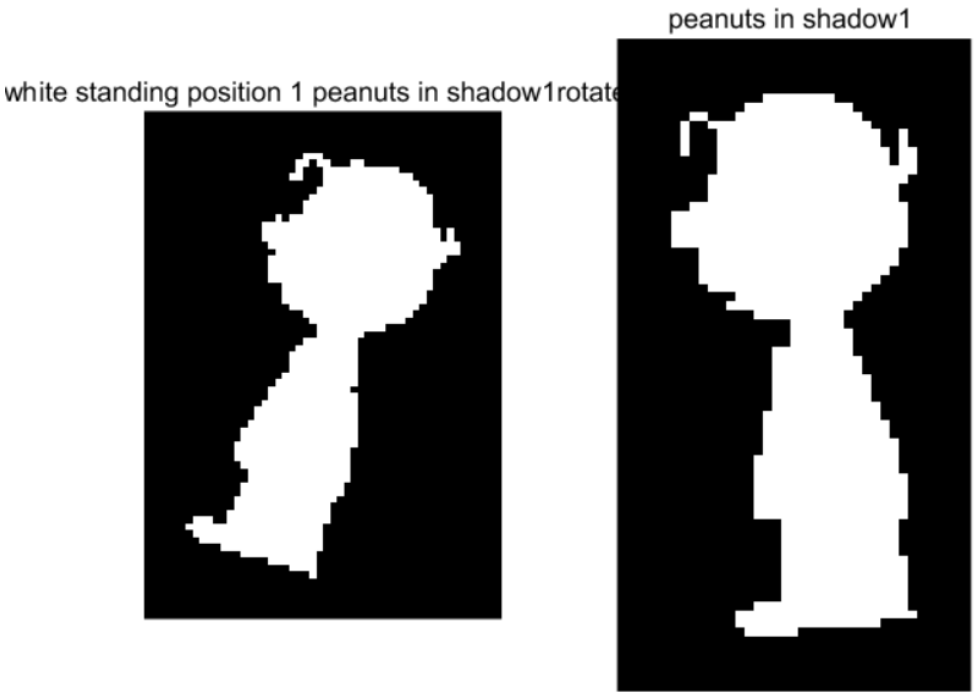


Figure 75. The result of finding white standing position 1 Peanuts in “shawdow1”

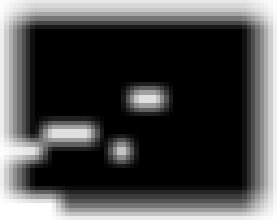


Figure 76. The complementary result of finding
white standing position 1 Peanuts in “shawdow1”

The following figures 77-81 are the pectrums of object white falling down Peanuts in “shadow1rotated.gif” and “shadow1.gif” , the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is white falling down Peanuts, and the result and the complementary result of finding white falling down Peanuts in “shadow1”.

Name	distance
'L=1'	0.4167
'L=2'	1.1098
'L=3'	0.20456
'L=4'	1.0317
'L=5'	1.0932
'L=6'	0.26764
'L=7'	1.063
'L=8'	1.2384

Figure 77. When L=3 in shawdow1rotated the distance of the value of L in shawdow1

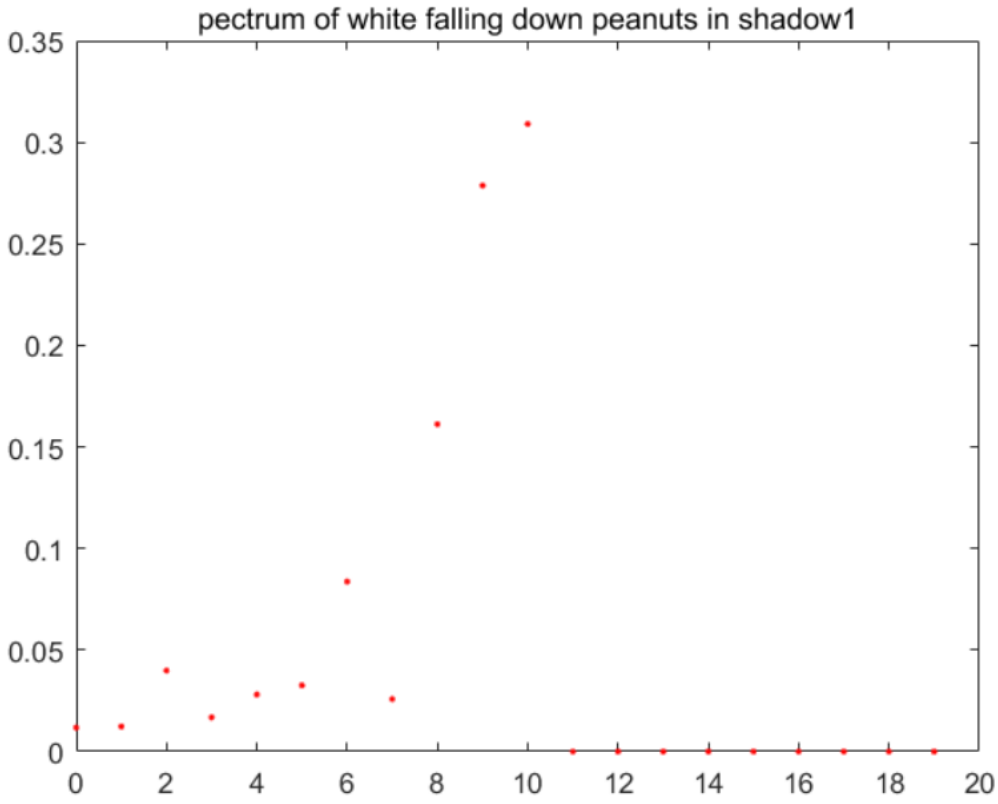


Figure 78. Pectrum of white falling down peanuts in shadow1

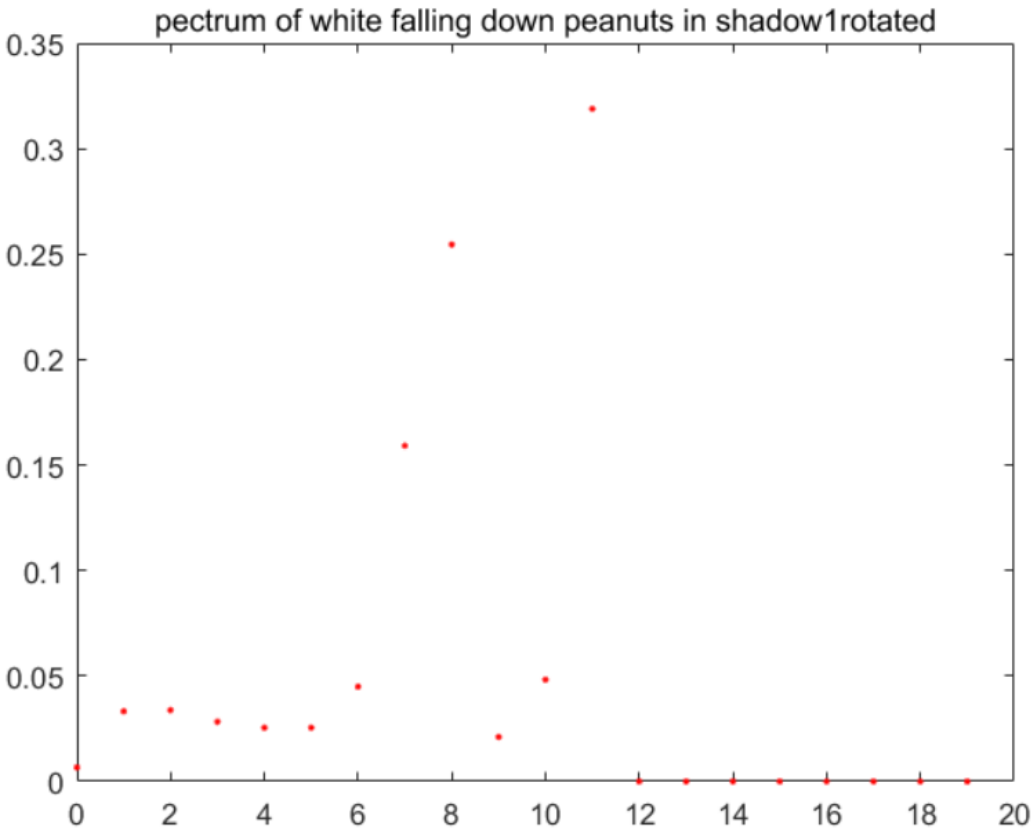


Figure 79. Pectrum of white falling down peanuts in shadow1rotated

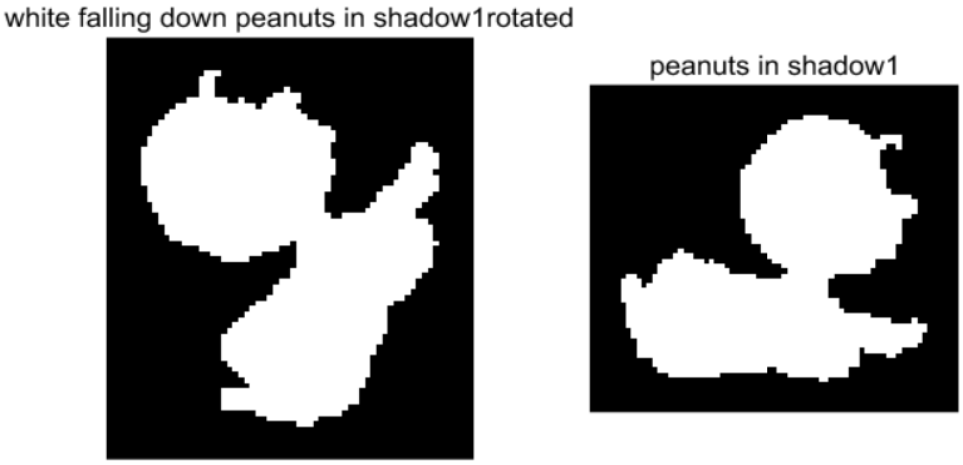


Figure 80. The result of finding white falling down Peanuts in “shawdow1”

Figure 81. The complementary result of finding
white white falling down Peanuts in “shawdow1”

The following figures 82-86 are the pectrums of object black standing position 2 Peanuts in “shadow1rotated.gif” and “shadow1.gif” , the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is black standing position 2 Peanuts, and the result and the complementary result of finding black standing position 2 Peanuts in “shadow1”.

Name	distance
'L=1'	1.2189
'L=2'	0.19396
'L=3'	1.0636
'L=4'	0.10253
'L=5'	0.22667
'L=6'	1.1616
'L=7'	1.1869
'L=8'	1.433

Figure 82. When L=4 in shawdow1rotated the distance of the value of L in shawdow1

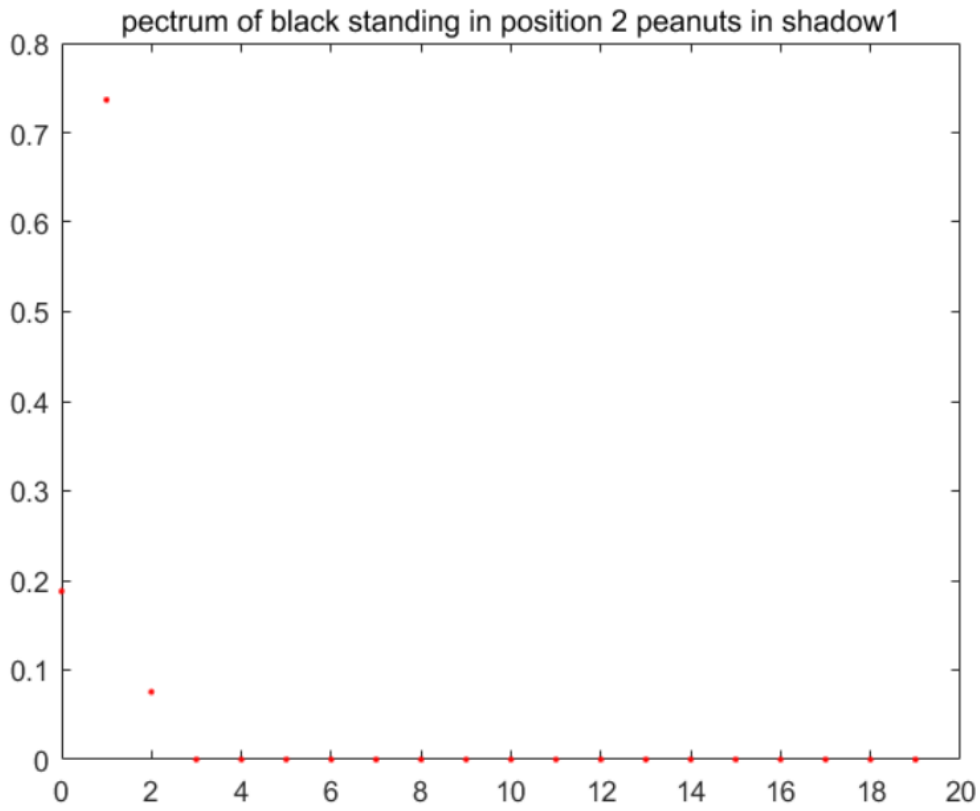


Figure 83. Pectrum of black standing position 2 peanuts in shadow1

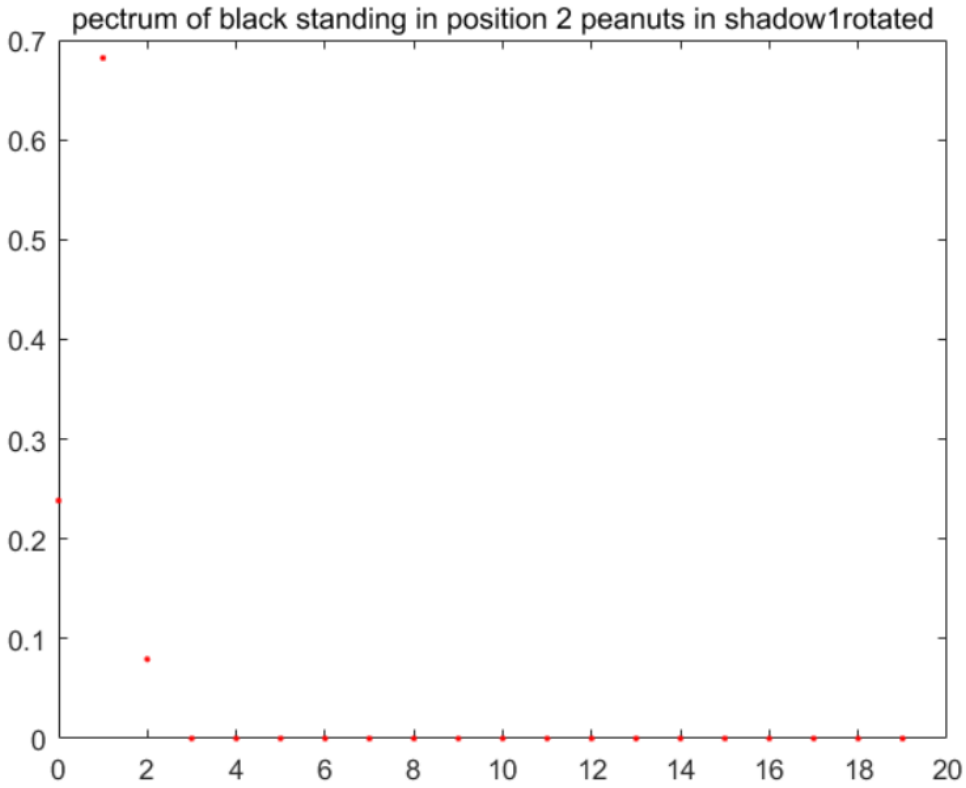


Figure 84. Pectrum of black standing position 2 peanuts in shadow1rotated

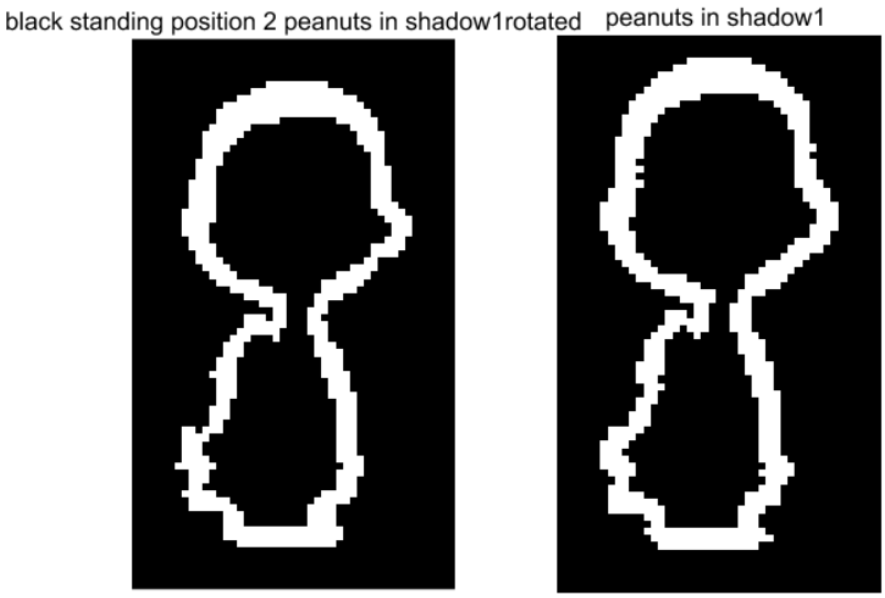


Figure 85. The result of finding black standing position 2 Peanuts in “shawdow1”

Figure 86. The complementary result of finding black standing position 2 Peanuts in “shawdow1”

The following figures 87-91 are the pectrums of object black standing position 3 Peanuts in “shadow1rotated.gif” and “shadow1.gif” , the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is black standing position 3 Peanuts, and the result and the complementary result and the complementary result of finding black standing position 3 Peanuts in “shadow1”.

Name	distance
'L=1'	1.2189
'L=2'	0.19396
'L=3'	1.0636
'L=4'	0.10253
'L=5'	0.22667
'L=6'	1.1616
'L=7'	1.1869
'L=8'	1.433

Figure 87. When L=5 in shawdow1rotated the distance of the value of L in shawdow1

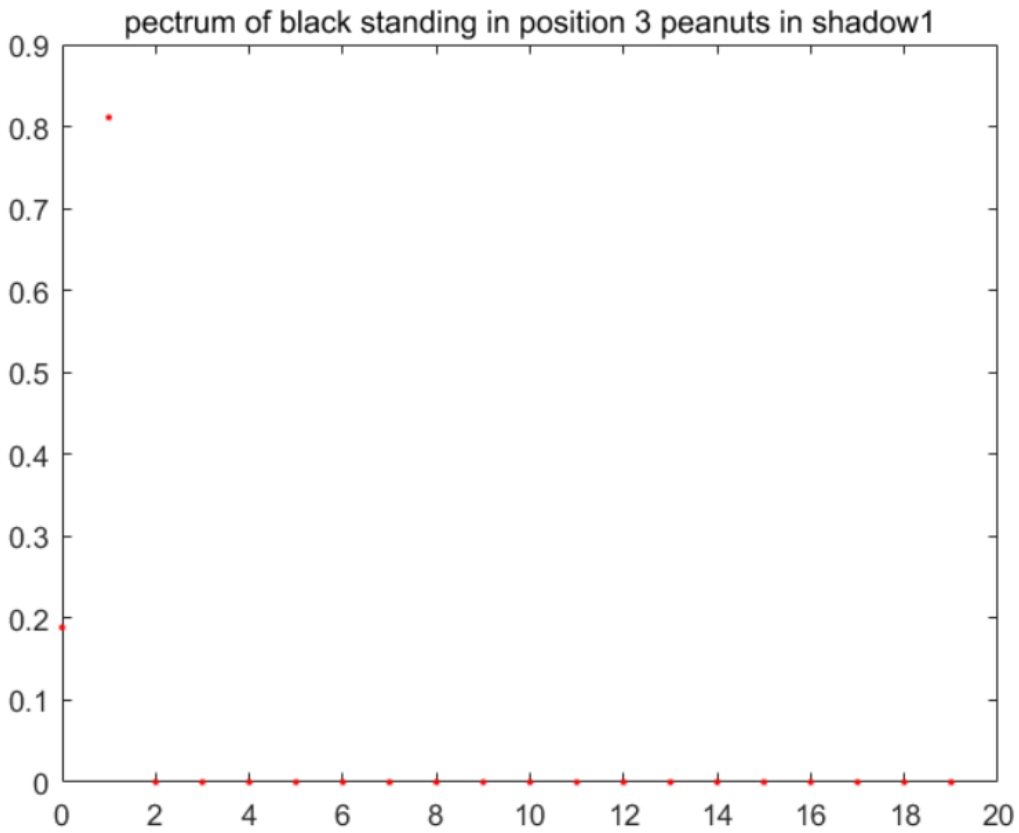


Figure 88. Pectrum of black standing position 3 peanuts in shadow1

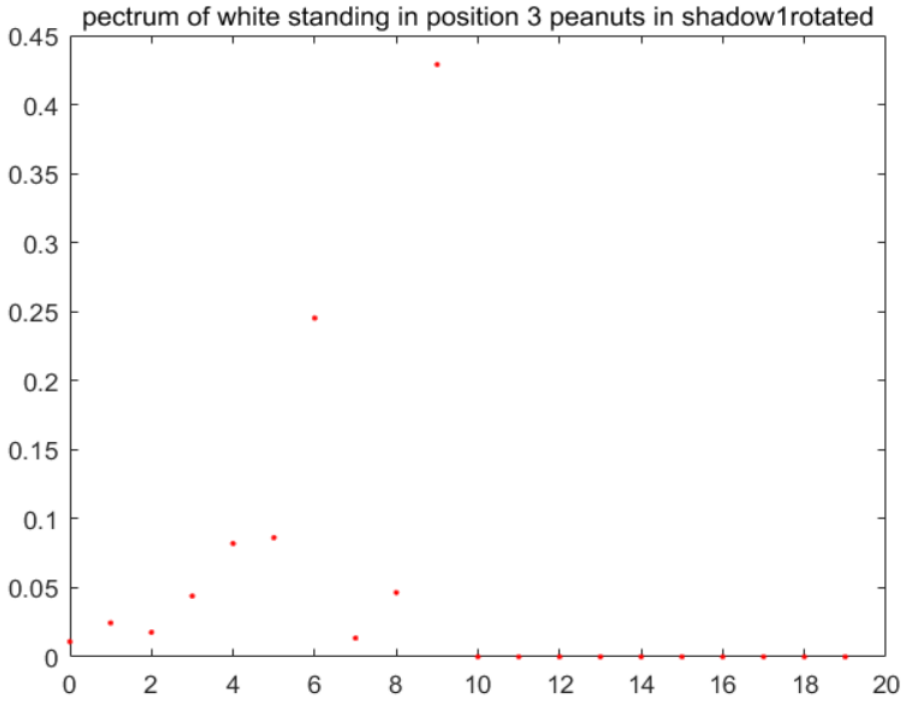


Figure 89. Pectrum of black standing position 3 peanuts in shadow1rotated

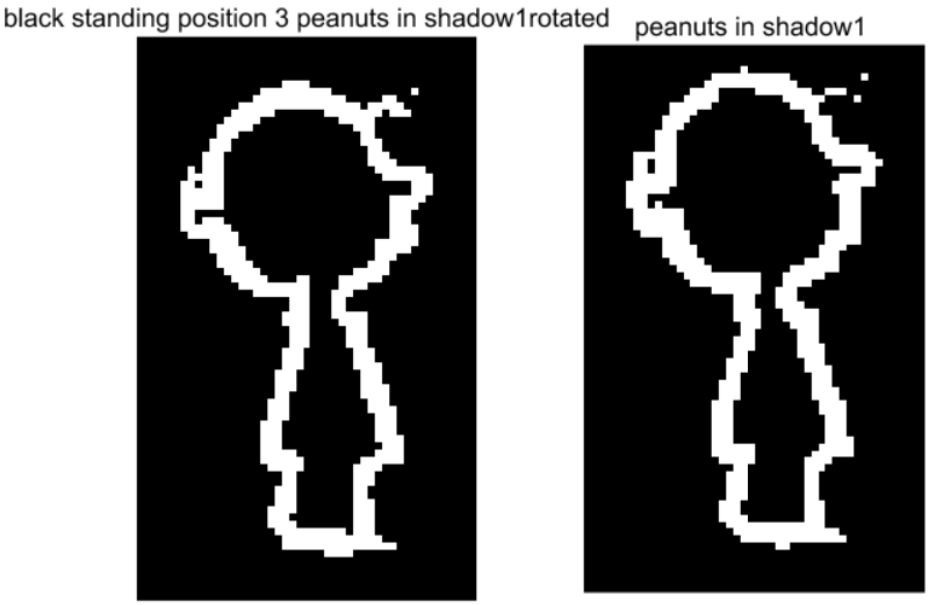


Figure 90. The result of finding black standing position 3 Peanuts in “shawdow1”

Figure 91. The complementary result of finding black standing position 3 Peanuts in “shawdow1”

The following figures 92-96 are the pectrums of object white standing position 2 Peanuts in “shadow1rotated.gif” and “shadow1.gif”, the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is white standing position 2 Peanuts, and the result and the complementary result of finding white standing position 2 Peanuts in “shadow1”.

Name	distance
'L=1'	0.34071
'L=2'	1.1361
'L=3'	0.19578
'L=4'	1.0915
'L=5'	1.1195
'L=6'	0.20472
'L=7'	1.0688
'L=8'	1.2106

Figure 92 When L=6 in shawdow1rotated the distance of the value of L in shawdow1

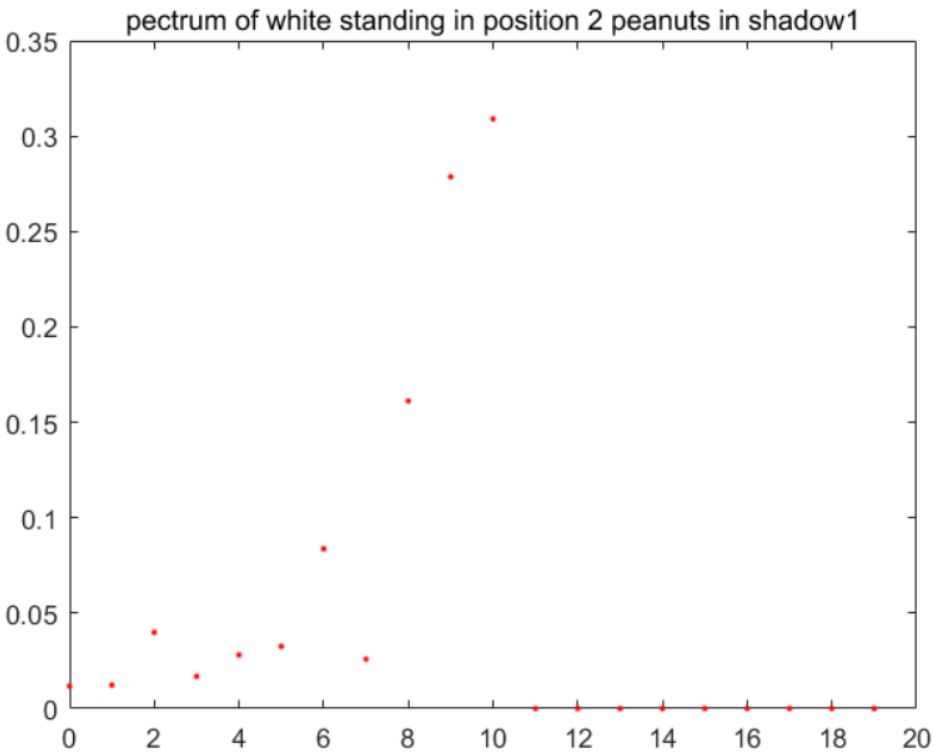


Figure 93 Pectrum of white standing position 2 peanuts in shadow1

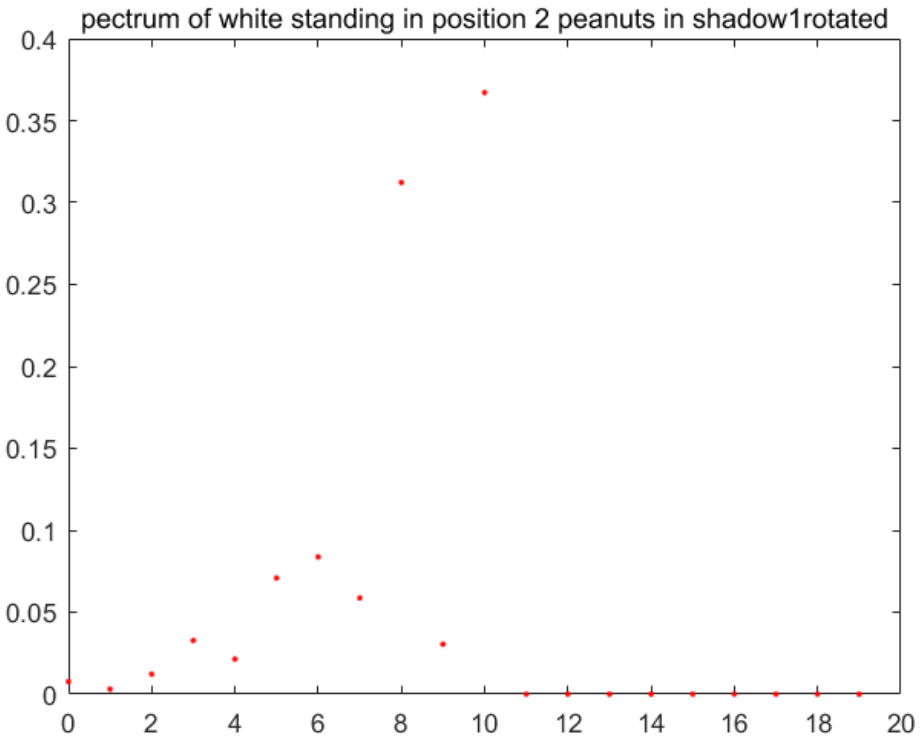


Figure 94. Pectrum of white standing position 2 peanuts in shadow1rotated

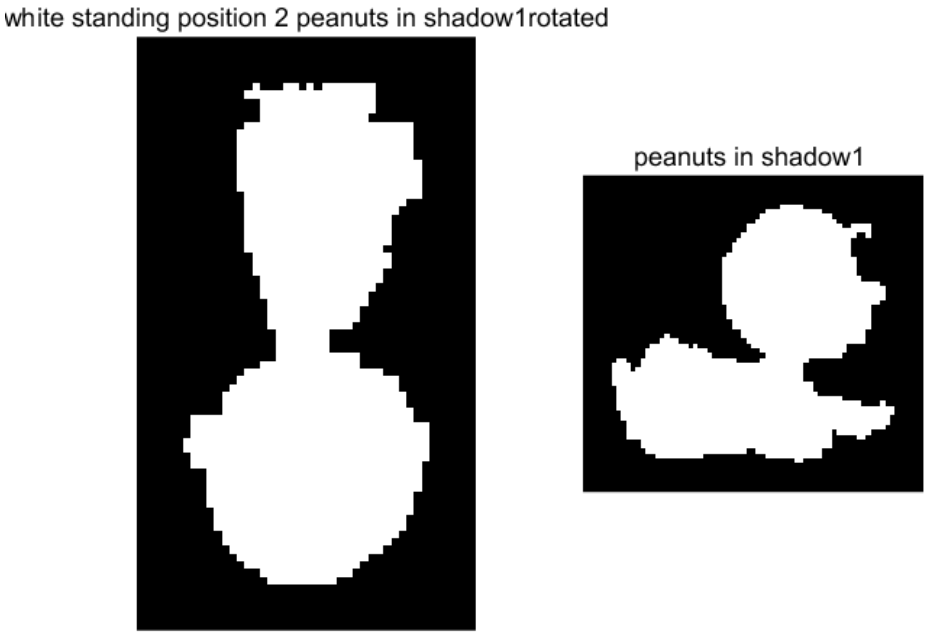


Figure 95. The result of finding white standing position 2
Peanuts in “shawdow1”

Figure 96. The complementary result of finding white standing position 2 Peanuts in “shawdow1”

The following figures 97-101 are the pectrums of object black standing position 1 Peanuts in “shadow1rotated.gif” and “shadow1.gif”, the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is black standing position 1 Peanuts, and the result and the complementary result of finding black standing position 1 Peanuts in “shadow1”.

Name	distance
'L=1'	1.3404
'L=2'	1.2391
'L=3'	1.2374
'L=4'	1.5294
'L=5'	1.5366
'L=6'	1.3324
'L=7'	0.96796
'L=8'	0

Figure 97 When L=7 in shawdow1rotated the distance of the value of L in shawdow1

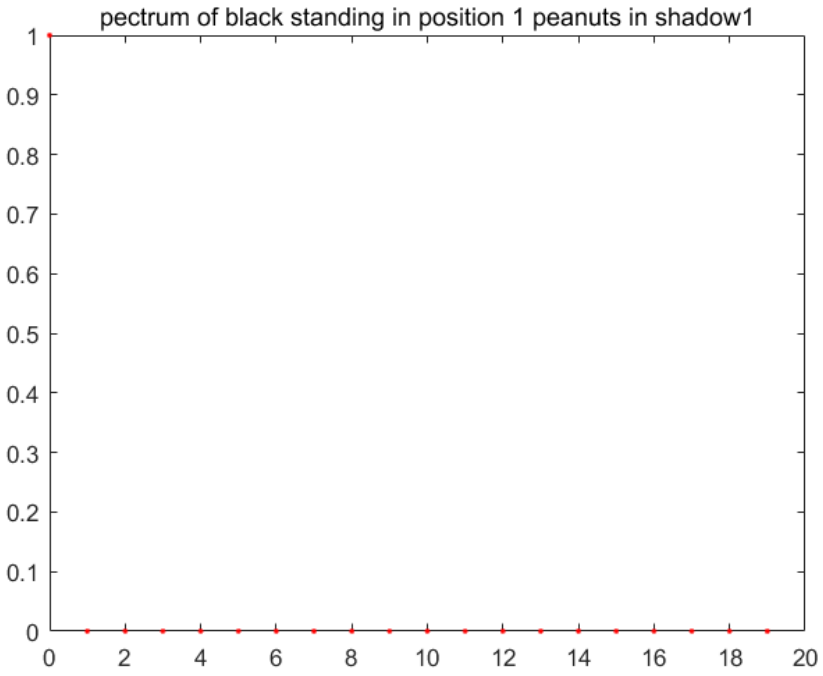


Figure 98 Pectrum of black standing position 1 peanuts in shadow1

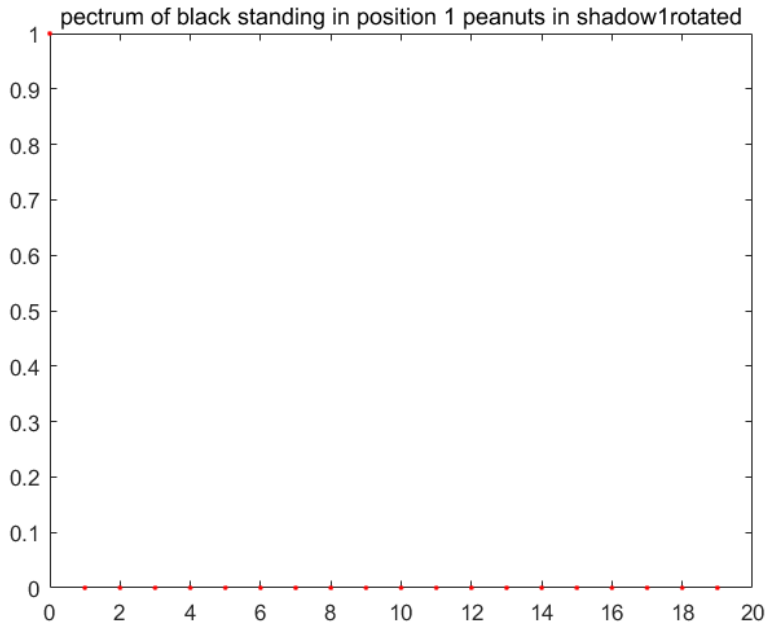


Figure 99. Pectrum of black standing position 1 peanuts in shadow1rotated

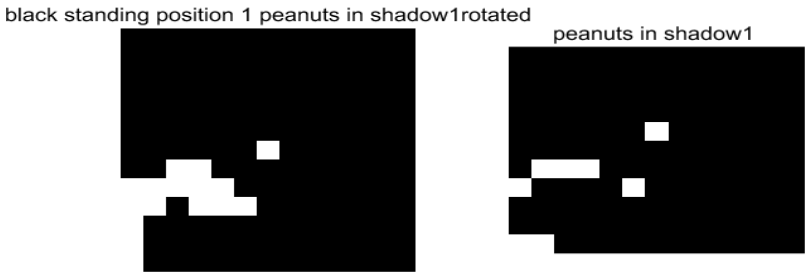


Figure 100. The result of finding black standing position 1
Peanuts in “shawdow1”

Figure 101. The complementary result of finding black standing position 1
Peanuts in “shawdow1”

The following figures 102-106 are the pectrums of object white standing position 3 Peanuts in “shadow1rotated.gif” and “shadow1.gif” , the distance of different values of L in “shadow1.gif” when the object in “shadow1rotated.gif” is white standing position 3 Peanuts, and the result and the complementary result of finding white standing position 3 Peanuts in “shadow1”.

Name	distance
'L=1'	0.091839
'L=2'	1.2358
'L=3'	0.22591
'L=4'	1.1825
'L=5'	1.2192
'L=6'	0.31982
'L=7'	1.1983
'L=8'	1.3487

Figure 102. When L=8 in shawdow1rotated the distance of the value of L in shawdow1

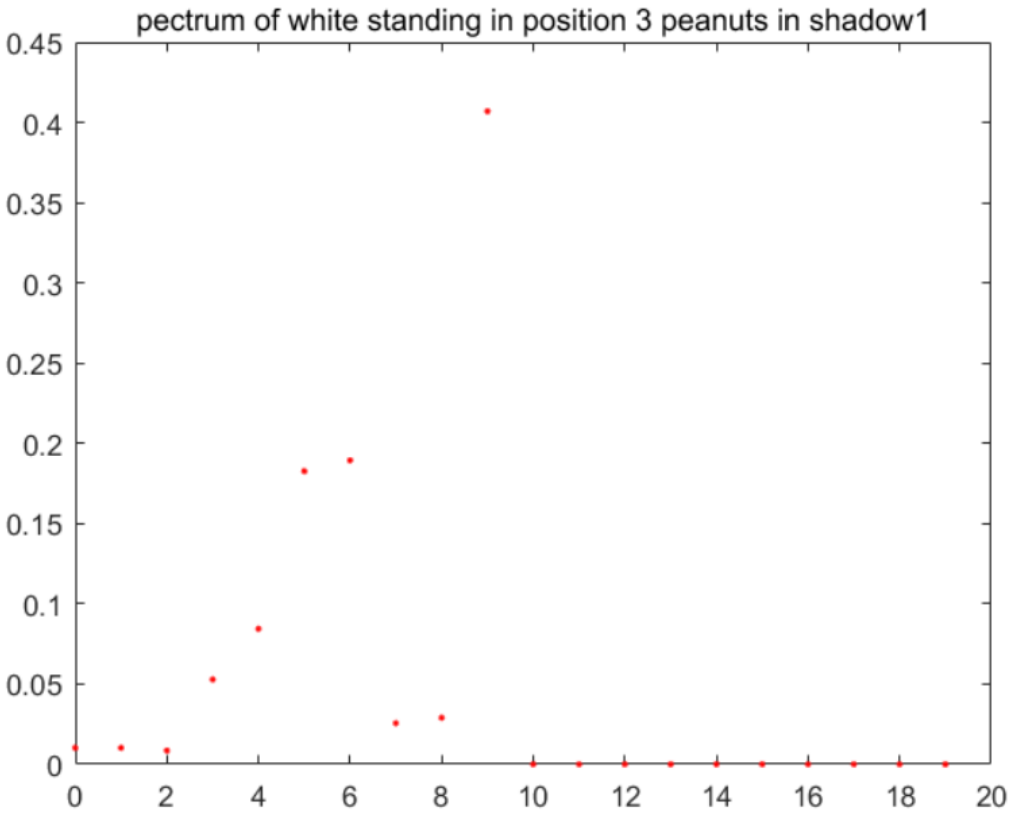


Figure 103. Pectrum of white standing position 3 peanuts in shadow1

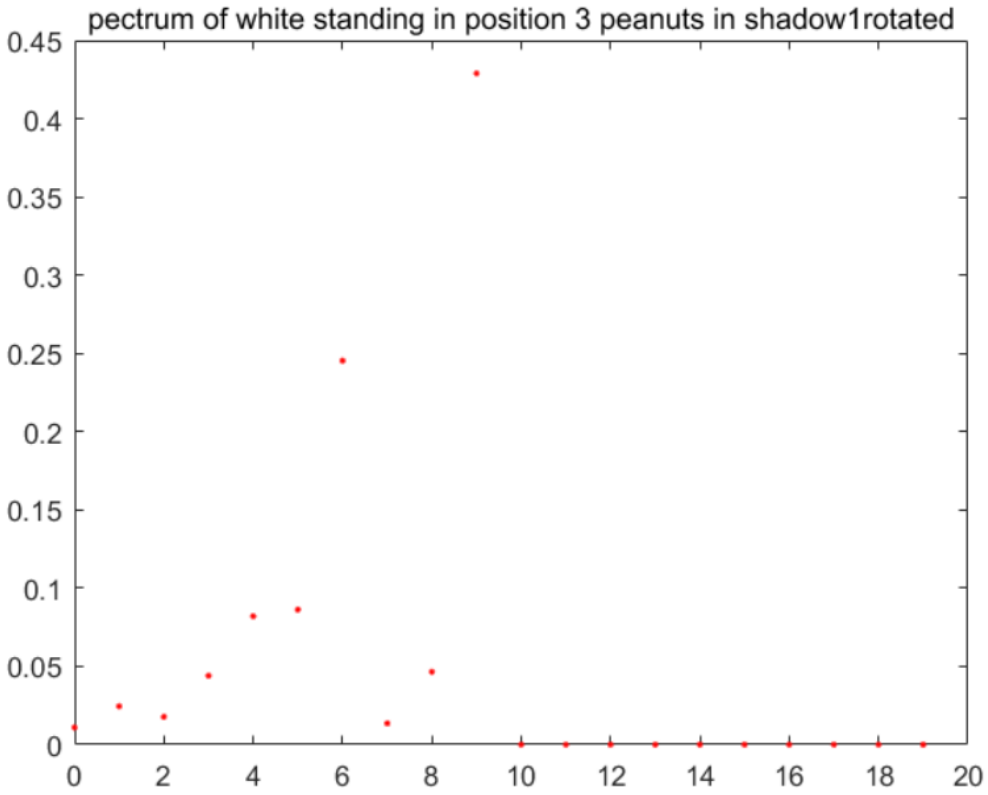


Figure 104. Pectrum of white standing position 3 peanuts in shadow1rotated

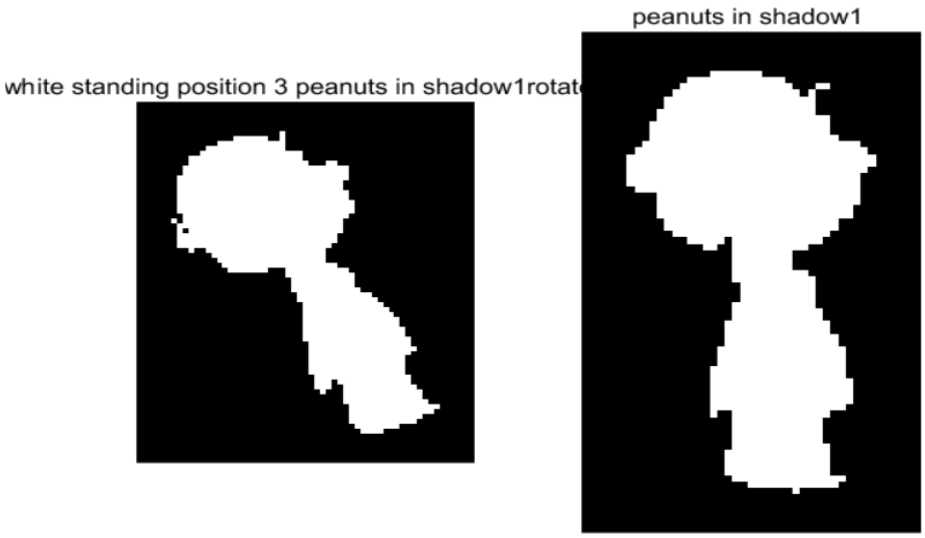


Figure 105. The result of finding white standing position 3 Peanuts in “shawdow1”

Figure 106. The complementary result of finding white standing position 3 Peanuts in “shawdow1”

As the result shown above, for this question we can use the algoritm to find the white falling down Peanuts and the white standing position 3 in “shadow1rotated” match the black(solid) falling down Peanuts and the black(solid) standing position 3 in “shadow1”

As for the bad results above shown, when we try to find result of white standing position 2 Peanuts and black standing position 1 Peanuts or try to find the complementary result of black standing position 2 Peanuts and white standing position 1 Peanuts, we cannot get the target object. But there are still some differences betweeh these situations.

For example When we enter L=6 to find white standing position 2 Peanuts ,we can still get the isolated objects in both “shawdow1rotated” and “shawdow1” , but when we enter L=7 to find black standing position 1 Peanuts ,we can’t get the isolated objects in either “shawdow1rotated” or “shawdow1” . In this result, we guess that the method bwlabel() function of isolating distinct objects and finding the minimum bounding rectangle(MBR) [also referred to as the bounding box] enclosing each distinct object in the algorithm we offer cannot recognize 8 objects in “shawdow1rotated” and “shawdow1” well. Then we try to export all the possible objects that the bwlabel() function can isolate, and it mentioned that black standing position 1 Peanuts can not isolated from “shawdow1rotated” and white standing position 2 Peanuts and black standing position 1 Peanuts can not isolated from “shawdow1”.

In our opinion, that the most part of the method is correct while the isolated object method needs to be improve.

D. Conclusions

This project demonstrated the method to morphological skeletons and partial reconstruction and it also provided a great example on computing size distributions, pecstrums, and complexities and how these change based on rotations of an object. What's more I think, our ability to write the report is also improved.

However, when we try to finish the project, there are lots of problems in understanding the definition and coding part solved which make us learn a lot. In the last question of the project, we only achieve finding 6 objects, while I think we find out the possible reason of it.