The Report of MP1

# **Basic Concepts**

* 1. Connected-Component Labeling

Connected-Component Labeling (CCL) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. It is used in computer vision to detect connected regions in binary digital images and to label the different region of component. In this task, I select the 4-connected method.

* 1. Algorithm

In this homework, I took the algorithm named ‘Two Pass’ to implement CCL. Apparently, The algorithm makes two passes over the image. The first pass to assign temporary labels and build up the equivalent\_table (E\_table), which is used to storage the equal value of the label, and the second pass to replace each temporary label by the smallest label of its equivalence class.

In the first pass, iterate through each element of the data by column, then by row.

If the element is not the background(0), get the neighboring elements(Lu & Ll) of the current element. If there are no neighbors, uniquely label the current element and go on. Otherwise, find the neighbor with the smallest label and assign it to the current element. Store the equivalence between neighboring labels in the established E\_table.

In the second pass, again iterate through each element of the data by column, then by row.

If the element is not the background(0), Relabel the element with the lowest equivalent label.

More importantly, how to build up the E\_table, which is shown as Table 1, is a key to the implementation of such algorithm. In my homework, I established a vector with the size of 1000, whose value is equal to its index. When there comes a new pair of equivalent labels, replace the value of E\_table(bigger one) by the smaller one. It will repeat again and again until all connected labels are replaced by the smaller one, the final vector is shown as Table 2. When relabeling in the second pass, use the smallest value to relabel the connected elements.

|  |  |
| --- | --- |
| Set ID | Equvalent labels |
| 1 | 1=3 |
| 2 | 2=4=5=9 |
| 3 | 6=7=8=10 |
| 4 | 11=1 |
| … | … |

Table E\_table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … |
| value | 1 | 2 | 1 | 2 | 2 | 6 | 6 | 6 | 2 |  |

Table The vector of E\_table after equalization

* 1. The Purpose of the Function

The MP1 requires us to implement the following function prototype:

function [labe\_img, num] = CCL(img)

where ‘img’ is input binary image, ‘label\_img’ contains the labels, and ‘num’ is the total number of different labels.

In addition, it would be better to select different color as different output of different label. And in the ‘gun.bmp’, it is better to use the size filter.

The matlab code of this function is shown in the file ‘CCL.m’.

# **Results and Analysis**

* 1. Test Results

MP1 is programmed and tested in the MATLAB\_R2016b, which is shown as Figure1 and Figure 2.

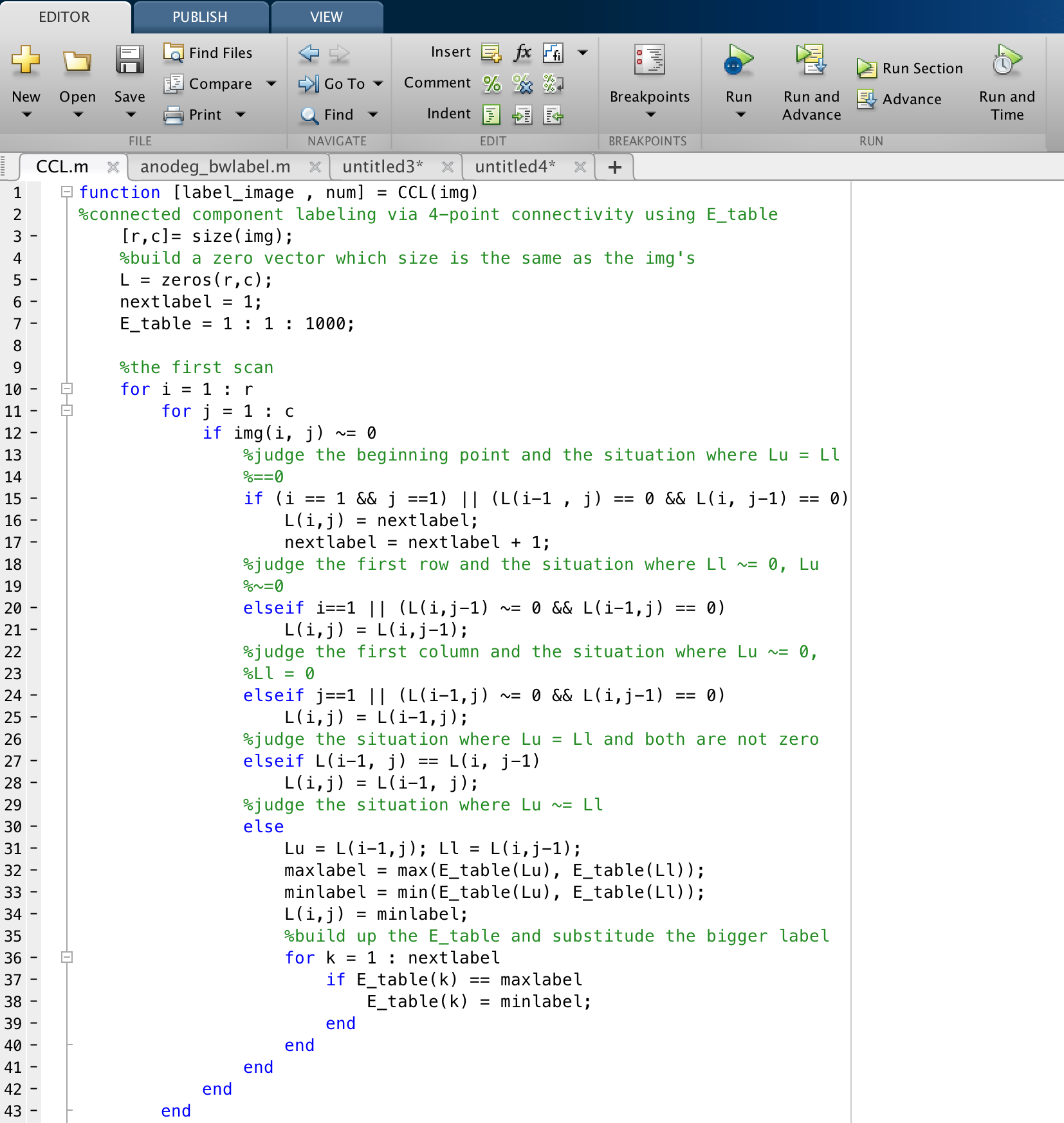
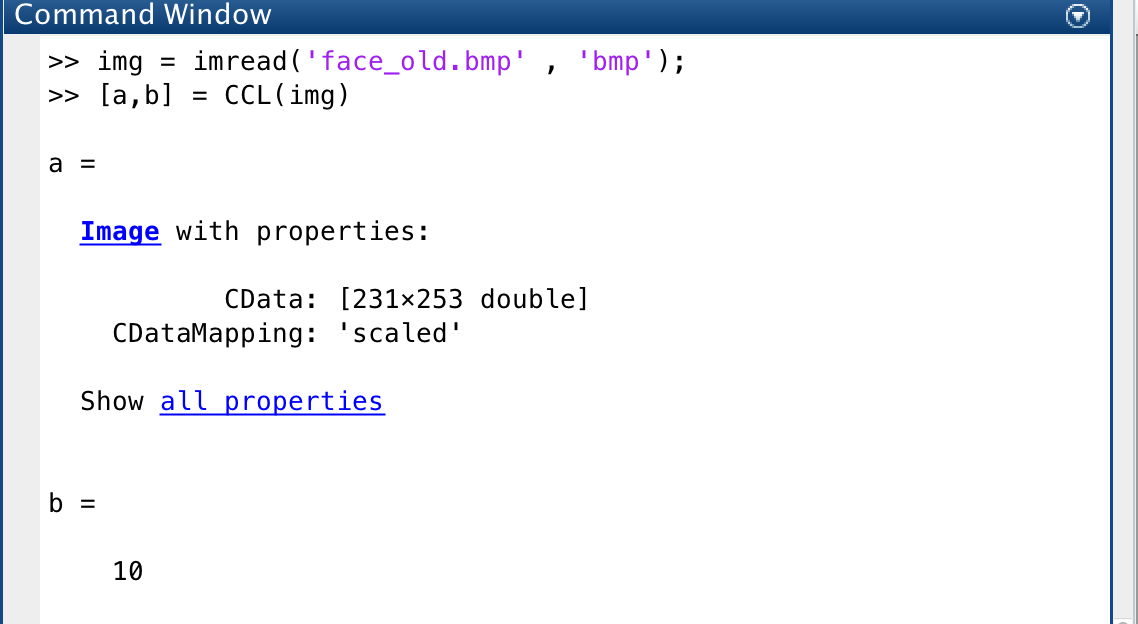
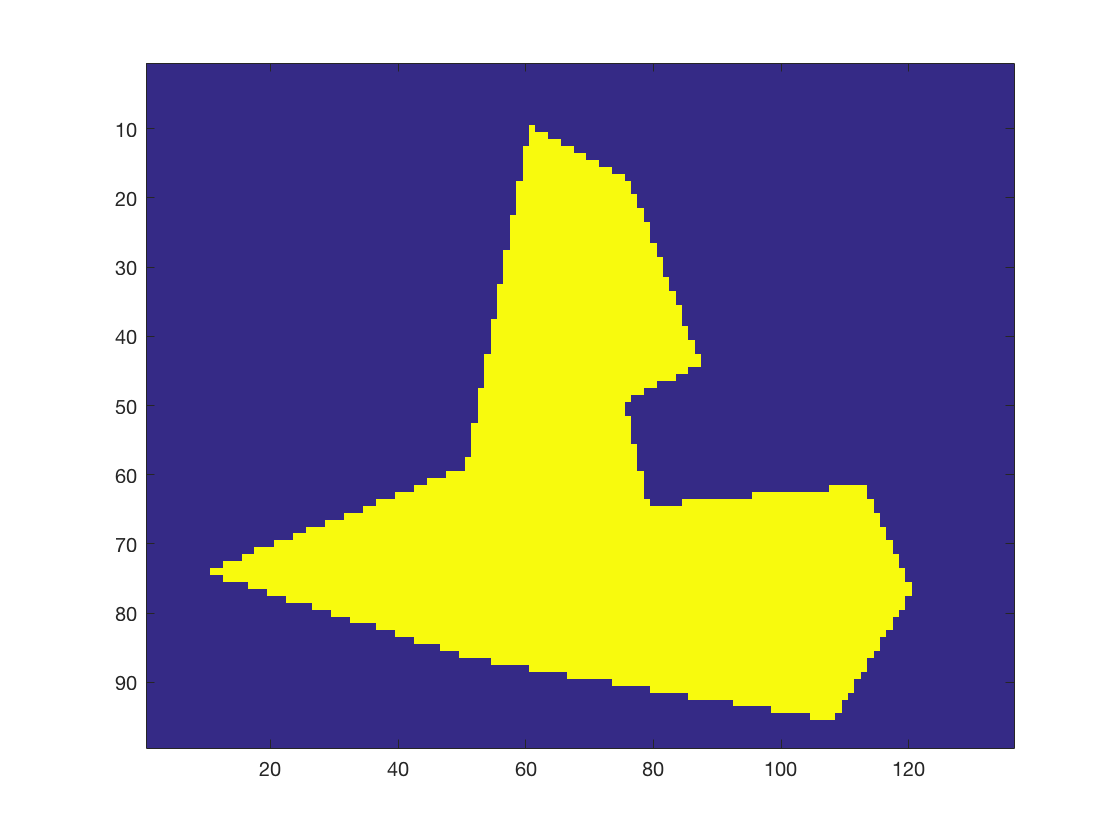
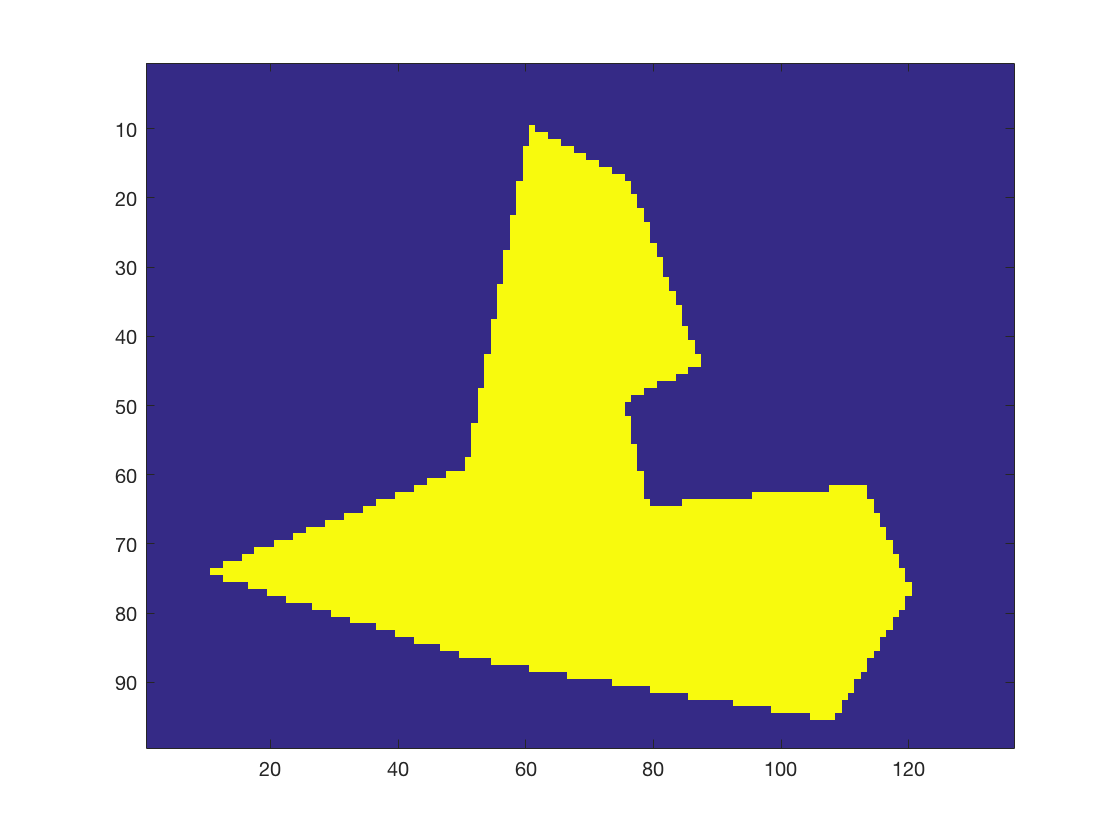
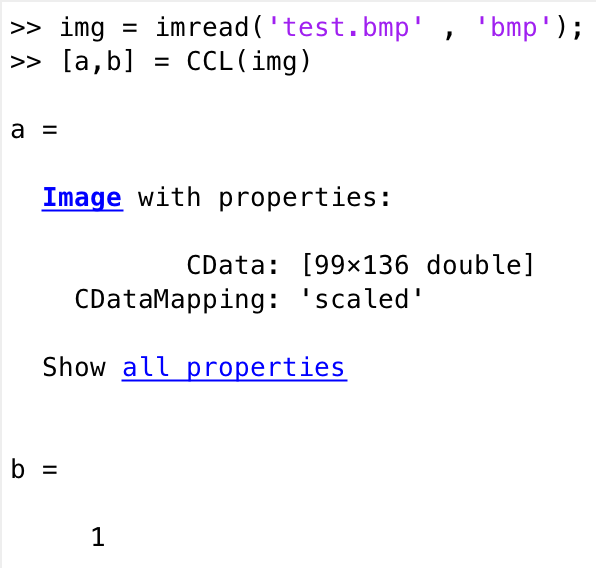
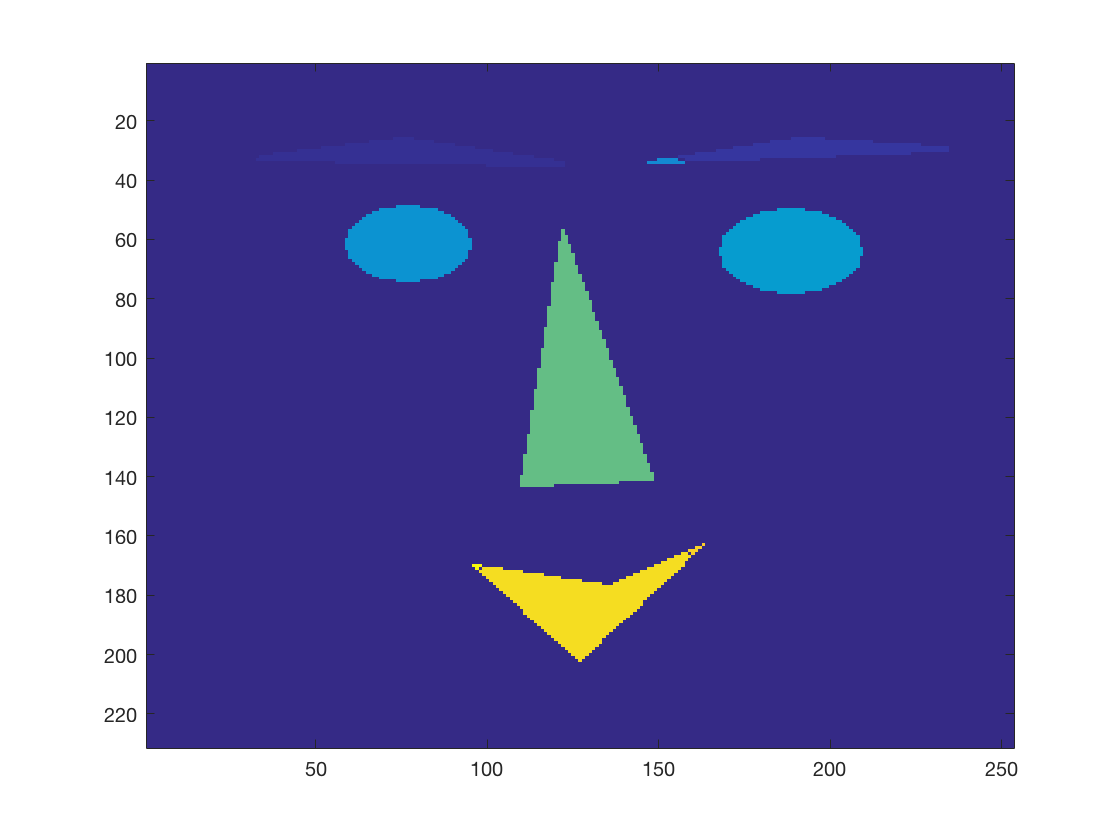


Figure 2 command window

Figure 1 matlab coding area

After running the codes and test the four different images, there come the results as followed.

1. The result of ‘test. bmp’ is shown in the Figure 3.  
   

 Figure 3 result of 'test.bmp'

1. The result of ‘’face\_old. bmp’ is shown in the Figure 4.

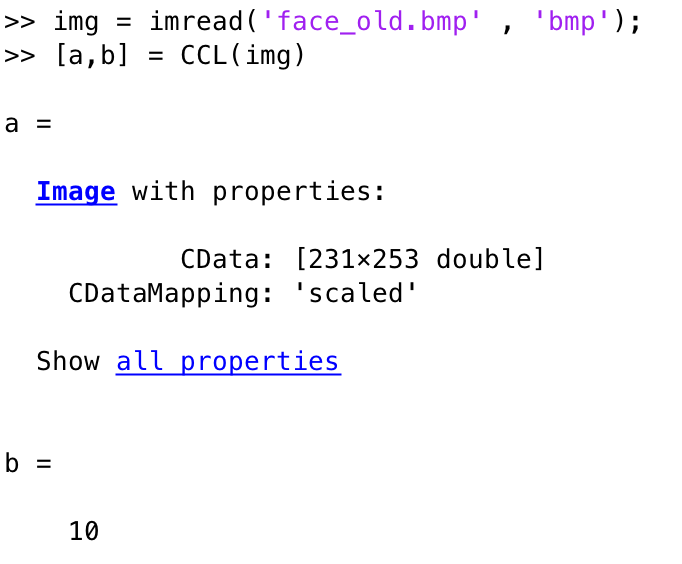


Figure 4 The Result of 'face\_old. bmp'

1. The result of ‘face. bmp’ is shown in the Figure 5.

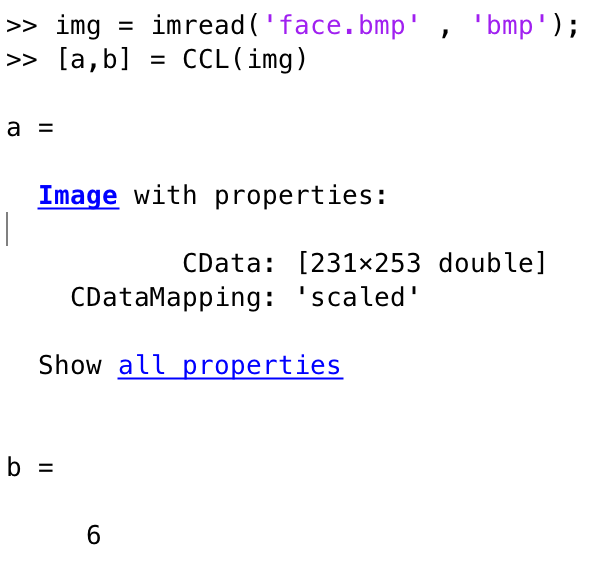
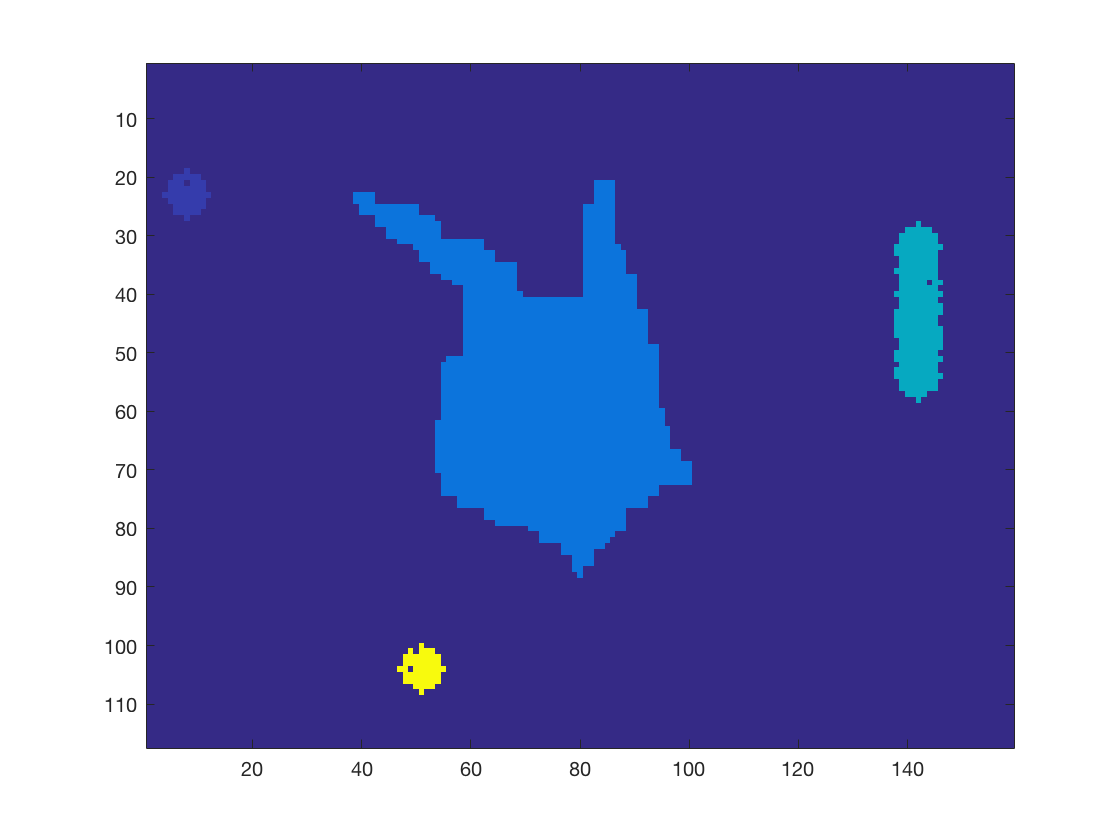


Figure 5 The Result of 'face.bmp'

1. .The result of ‘gun. bmp’ (without size filter) is shown in the Figure 6.

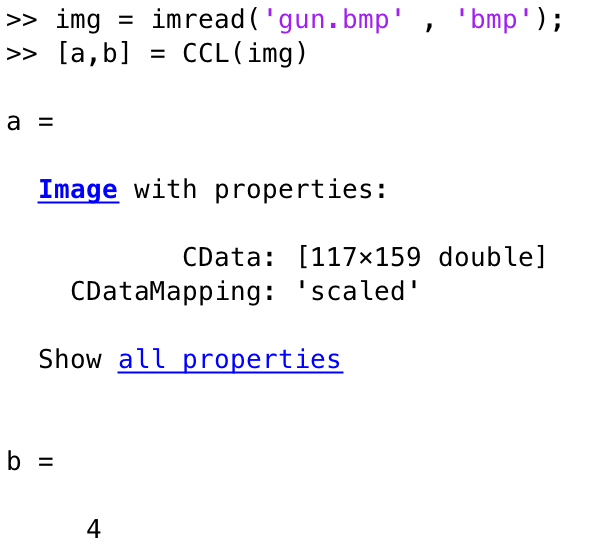


Figure 6 The Result of 'gun.bmp' (without size filter)

1. When adding the part of size filter, the result of ‘gun.bmp’ is shown in the Figure.7.

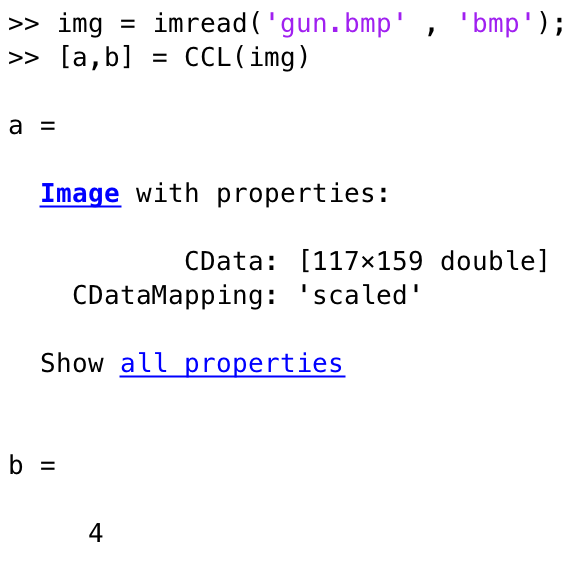
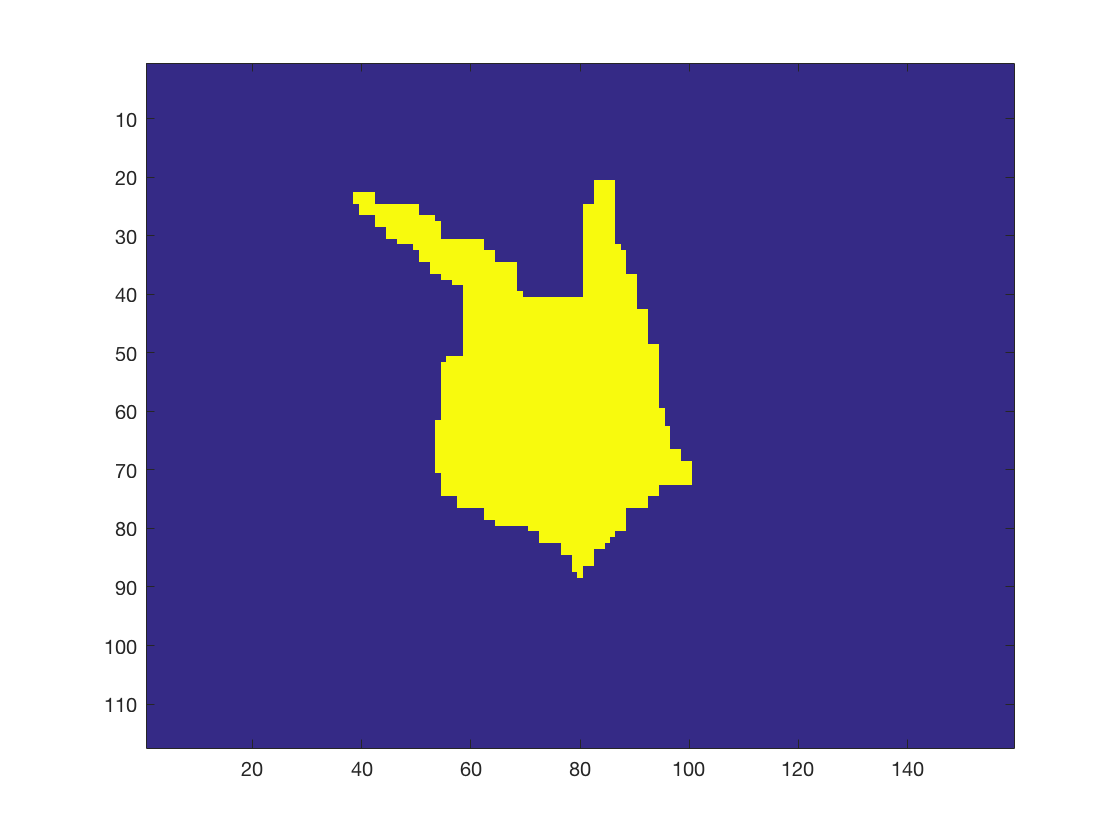
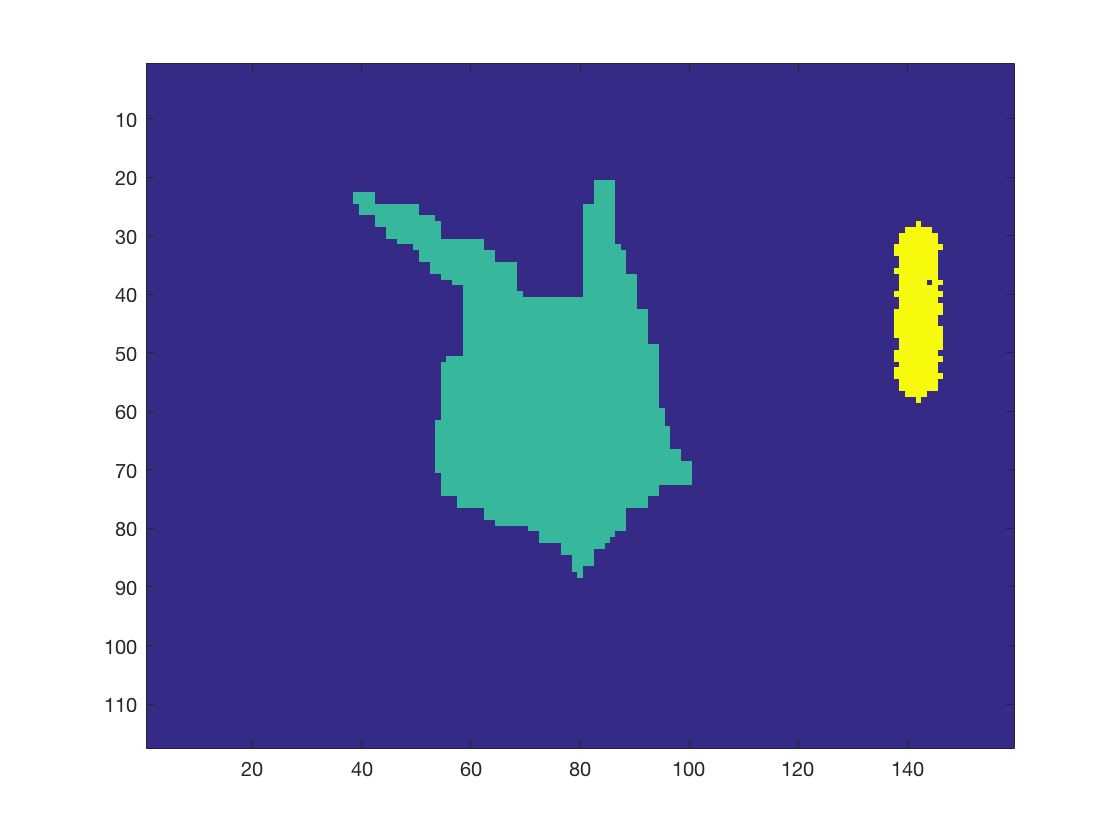


Figure 7 The Result of 'gun.bmp'

* 1. Results Analysis

For the first test, the result is there being just one connection region. The whole label is 1 and the num equals to 1.

For the second test, the result is there being 10 different regions in the ‘face\_old.bmp’, and the num equals to 10. Clearly, when it comes to the third test named ‘face.bmp’, the result turns to 6 parts, that is the num is equal to 6.

For the fourth test, as the Figure 7 shown, there are 4 different parts, and the num is equal to 4. When I add the size filter which threshold is 40, there is just one part on the picture, but when I set threshold as 15, there are two parts on the picture. That is to say, the size filter works out perfectly.

From this MP1, I get a better understanding of the core of the function of CCL as well as size filter. What’s more, it raises my interest in computer vision. I am looking forward to learning more interesting concepts of CV and machine learning.

* 1. 2.3 MATLAB Coding

1. function [label\_image , num] = CCL(img)
2. %connected component labeling via 4-point connectivity using E\_table
3. [r,c]= size(img);
4. %build a zero vector which size is the same as the img's
5. L = zeros(r,c);
6. nextlabel = 1;
7. %threshold = 40;
8. E\_table = 1 : 1 : 1000;
9. %Size\_filter = zeros(1000);
11. %the first scan
12. for i = 1 : r
13. for j = 1 : c
14. if img(i, j) ~= 0
15. %judge the beginning point and the situation where Lu = Ll
16. %==0
17. if (i == 1 && j ==1) || (L(i-1 , j) == 0 && L(i, j-1) == 0)
18. L(i,j) = nextlabel;
19. nextlabel = nextlabel + 1;
20. %judge the first row and the situation where Ll ~= 0, Lu
21. %~=0
22. elseif i==1 || (L(i,j-1) ~= 0 && L(i-1,j) == 0)
23. L(i,j) = L(i,j-1);
24. %judge the first column and the situation where Lu ~= 0,
25. %Ll = 0
26. elseif j==1 || (L(i-1,j) ~= 0 && L(i,j-1) == 0)
27. L(i,j) = L(i-1,j);
28. %judge the situation where Lu = Ll and both are not zero
29. elseif L(i-1, j) == L(i, j-1)
30. L(i,j) = L(i-1, j);
31. %judge the situation where Lu ~= Ll
32. else
33. Lu = L(i-1,j); Ll = L(i,j-1);
34. maxlabel = max(E\_table(Lu), E\_table(Ll));
35. minlabel = min(E\_table(Lu), E\_table(Ll));
36. L(i,j) = minlabel;
37. %build up the E\_table and substitude the bigger label
38. for k = 1 : nextlabel
39. if E\_table(k) == maxlabel
40. E\_table(k) = minlabel;
41. end
42. end
43. end
44. end
45. end
46. end
48. %the second scan
49. for i= 1:r
50. for j = 1:c
51. %update L(i,j) according to E\_table
52. if L(i,j) ~= 0 && E\_table(L(i,j)) ~= L(i,j)
53. L(i,j) = E\_table(L(i,j));
54. %count the number of the same labels
55. %Size\_filter(L(i,j)) = Size\_filter(L(i,j)) + 1;
56. end
57. end
58. end
60. %size filter
61. % for i= 1:r
62. % for j = 1:c
63. % if L(i,j) ~=0 && Size\_filter(L(i,j)) <threshold
64. %   L(i,j) = 0;
65. % end
66. %end
67. % end
69. %count the total number of different labels
70. totalnum = 0;
71. for i = 1 : nextlabel-1
72. if E\_table (i) == i
73. totalnum = totalnum + 1;
74. end
75. end
77. %label\_image = L;
78. label\_image = imagesc(L);
79. num = totalnum;
81. end