CSci 5561
Computer Vision

Programming assignment 3 report

Spring Semester 2016

Xiao Lu

ID：5079620

Overview Structure and Code Files

1. Algorithms and code files

   Main.m

   1.1 Histogram the Intensity Values of Each Image

      Preprocessing.m

      Smooth.m

      Histogram.m

      OTSUThreshold.m

   1.2 Connected Region Extraction

      RasterScan.m

   1.3 Computation of Blob Statistics

      BlobStatistics.m

      Perimeter.m

      FindHole.m

2. Test images

   2.1 *reg3.jpg*

   2.2 *reg4.jpg*

3. Output and plot

   3.1 *Smoothed image plot*

   3.2 *Histogram plot*

   3.3 *Threshold*

   3.4 *Extracted image plot*

   3.5 *Individual region plot along with perimeter*

   3.6 *MBR and centroid plot*

   3.7 *Parameter output txt*

Part 1 Histogram the Intensity Values of Each Image

1.1 Basic concepts and implementation

This part consists of three basic algorithms: Smoothing, Histogram and Thresholding. All of them are fundamental algorithms of computer vision.

For Smoothing, I have implemented two methods: two neighbors and four neighbors. The basic idea of them are just replacing each element by the sum of itself plus its two or four neighbors. To get the reasonable threshold, I eventually chose the two neighbors smoothing method.

For Intensity Histogram, I just went through whole image and count the occurrence of the value between 0 and 255.

Based on previous programming assignment, the way I chose OTSU thresholding method. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal. Consequently, Otsu's method is roughly a one-dimensional, discrete analog of Fisher's Discriminant Analysis.

1.2 Results

For smoothing and histogram parts, I just show the results of reg3.jpg.

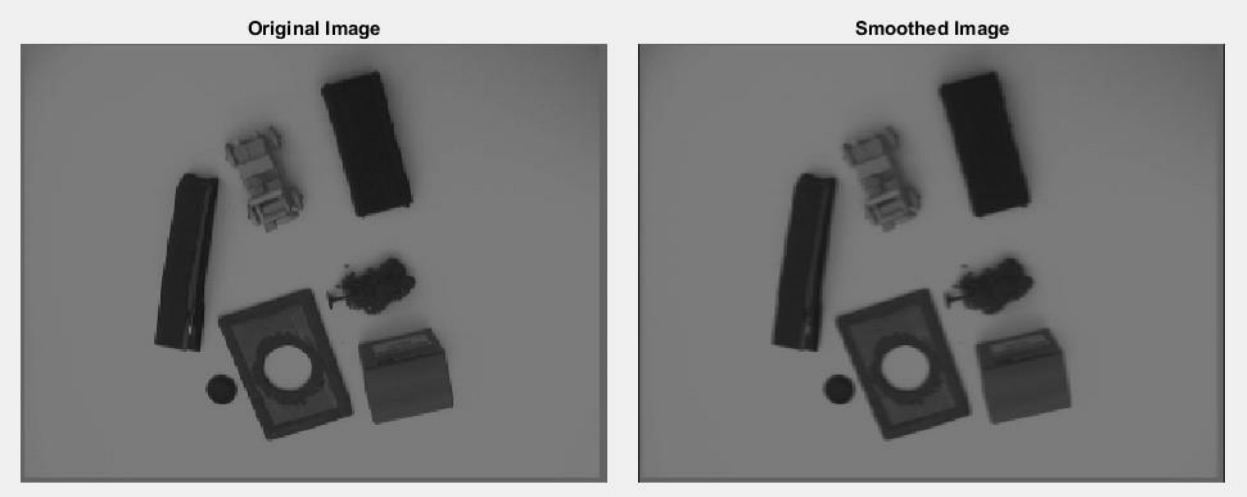


Figure 1. Comparison between original image and smoothed image

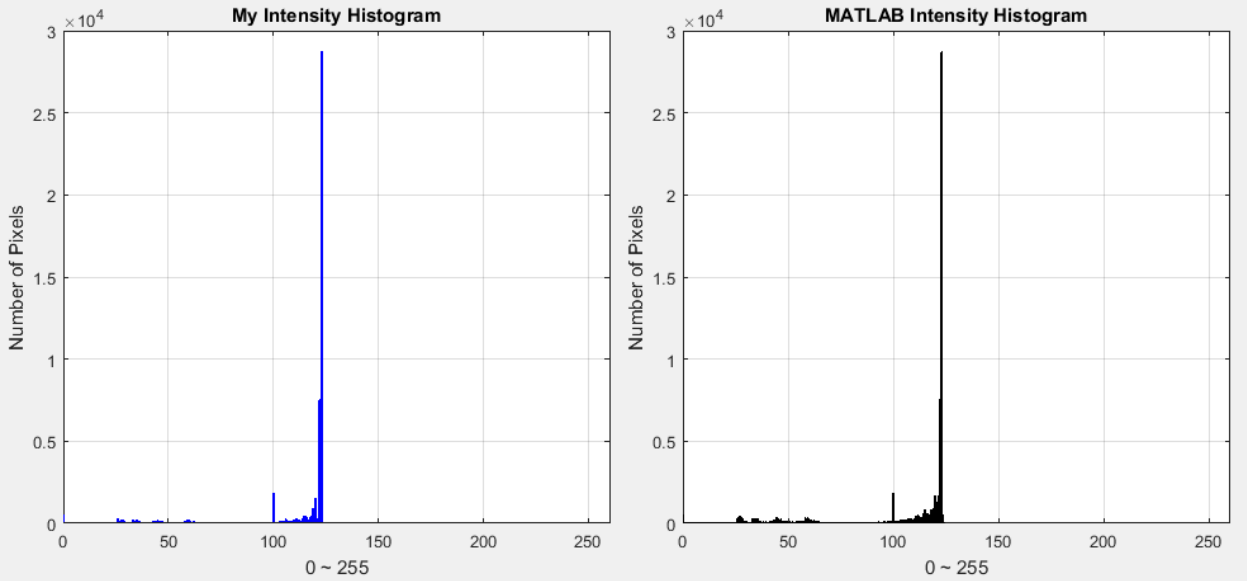


Figure 2. Histogram between my function and MATLAB function

Through the OTSU algorithm, the threshold for image reg3.jpg is 79, while reg4.jpg's threshold is 78.

Part 2 Connected Region Extraction

2.1 Basic concepts and implementation

Connected region extraction is an algorithmic application of computer vison and graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. It is used to detect connected regions in binary digital images, although color images and data with higher dimensionality can also be processed.

In my implementation, I adopted simple raster scan with two pass, the algorithm is shown as following.

On the first pass:

1. Iterate through each element of the data by column, then by row (Raster Scanning)
2. If the element is not the background
    1. Get the neighboring elements of the current element
    2. If there are no neighbors, uniquely label the current element and continue
    3. Otherwise, find the neighbor with the smallest label and assign it to the current element
    4. Store the equivalence between neighboring labels

On the second pass:

1. Iterate through each element of the data by column, then by row
2. If the element is not the background
    1. Relabel the element with the lowest equivalent label

Here, the background is a classification, specific to the data, used to distinguish salient elements from the foreground. If the background variable is omitted, then the two-pass algorithm will treat the background as another region.

Since my implementation is based on MATLAB, I cannot find a good way to deal with linked labels. My implementation is relatively slow, the entire raster scan part will run around 1 mins. I would like to improve its speed in the future.

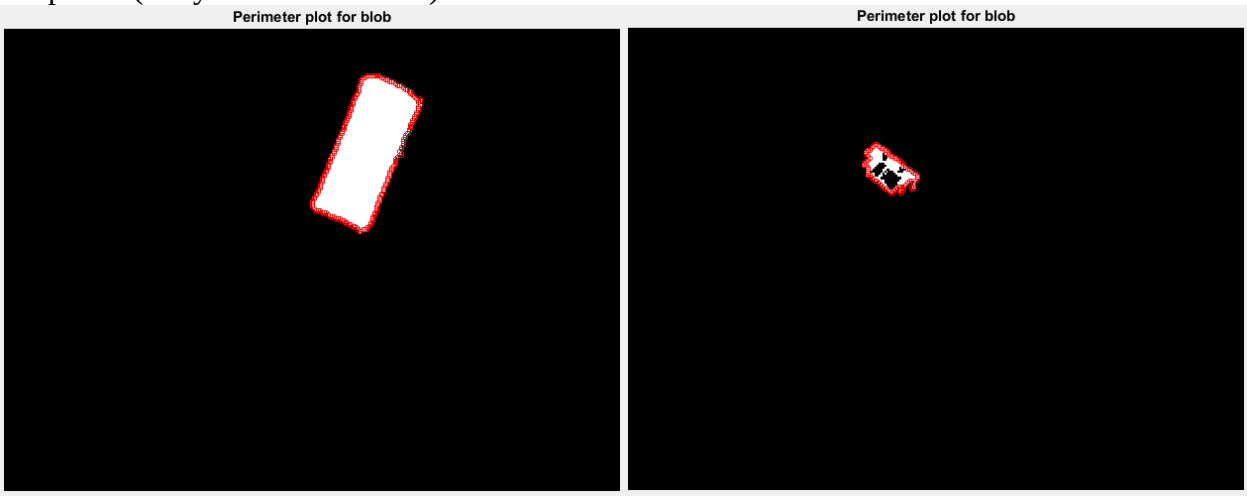
2.2 Results

The extracted images are shown as below.

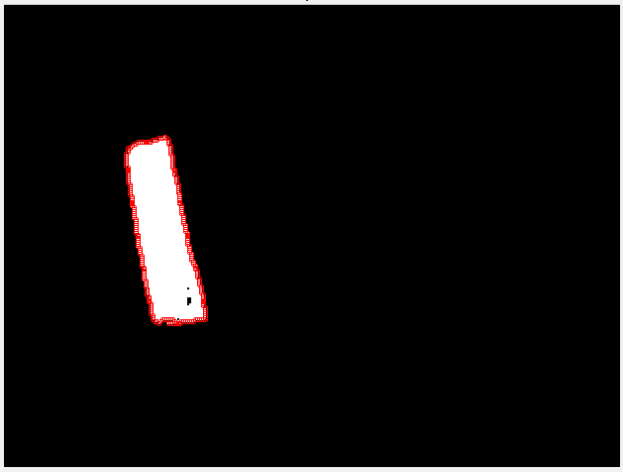
Figure 3. Comparison between original and extracted image for reg3.jpg

Figure 4. Comparison between original and extracted image for reg4.jpg

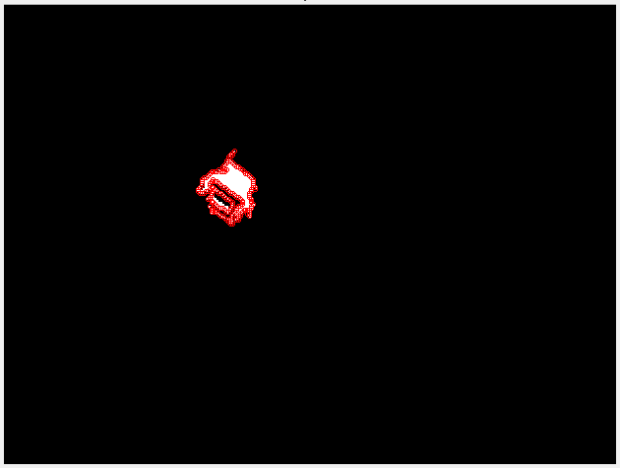
The following images are presented the individual resulting region along with the perimeter pixels (They are shown in red).



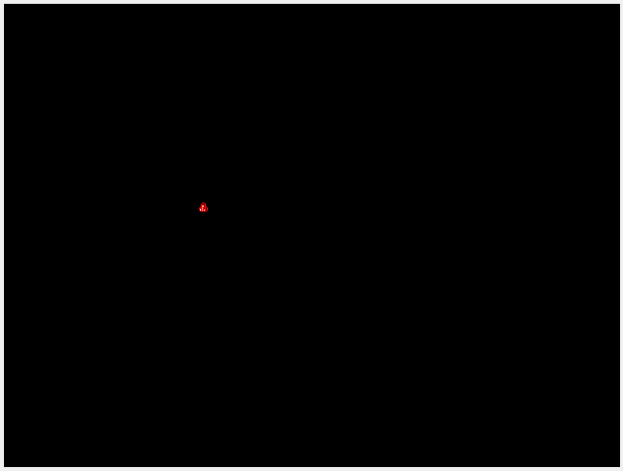
Perimeter plot for blob



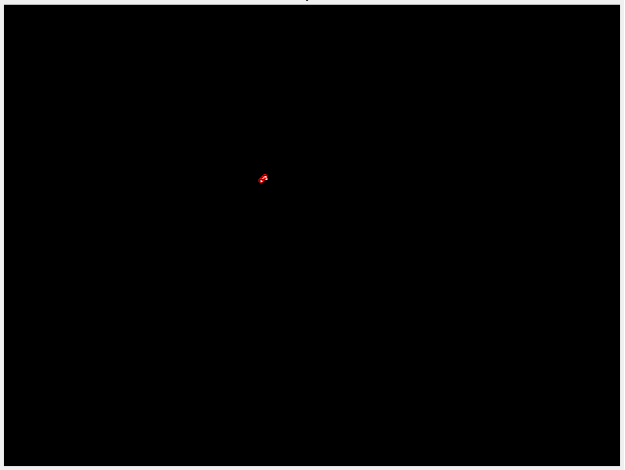
Perimeter plot for blob



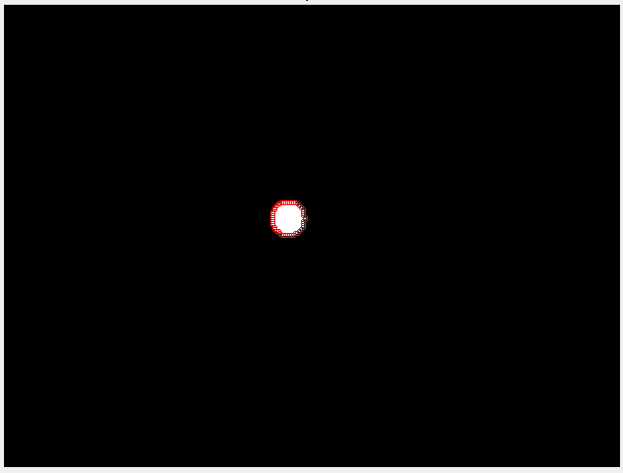
Perimeter plot for blob



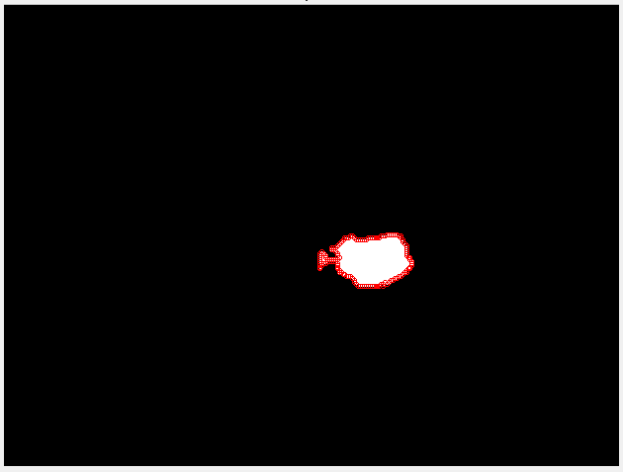
Perimeter plot for blob



Perimeter plot for blob



Perimeter plot for blob



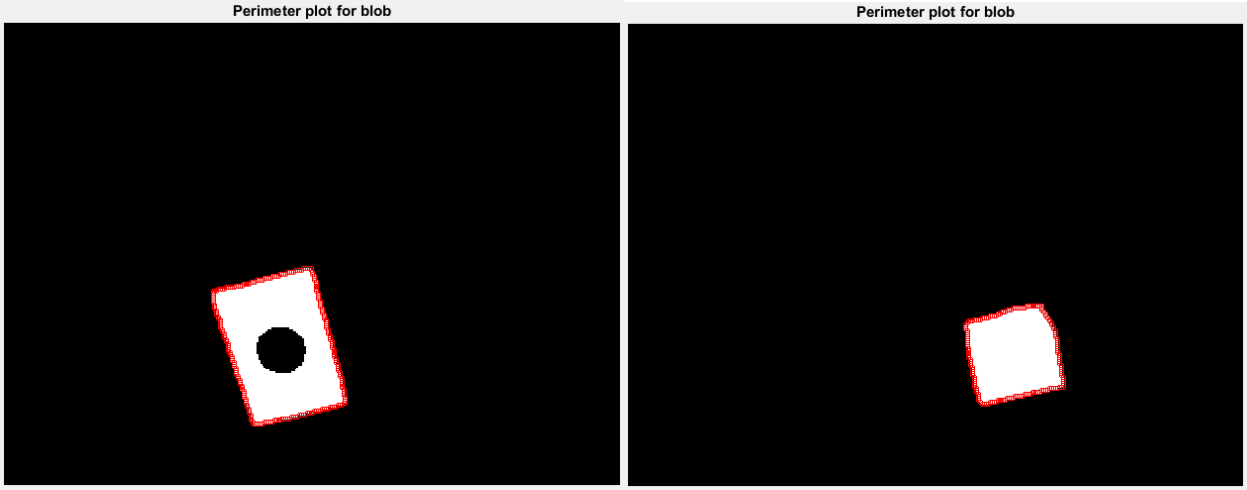


Figure 5. Individual region plot along with perimeter

In my implementation, I found that the model car is divided into several parts since its color is quite similar to background. At the same time, I also noticed that the three right-bottom objectives sometimes merge into one. I have tried to fix both of two problem by adjusting threshold manually, but I just can fix either one. This means that a better threshold contribute to a better region extraction results. We should notice that, the previous results is based on adjusted OTSU algorithm.

Part 3 Computation of Blob Statistics

3.1 Basic concepts and implementation

To obtain the statistical information of a blob part, such as MBR, centroid, number of holes, area, area of holes, perimeter, and elongation, the most difficult parts are to count number of holes and to calculate perimeter.
To count number of holes, I used raster scan algorithm again. To calculate perimeter of blob, I implemented the border tracking algorithm.

3.2 Results

The following images show the MBR and centroid plot, the yellow rectangles are MBR of each region, and the blue asterisks are the centroid of each region.

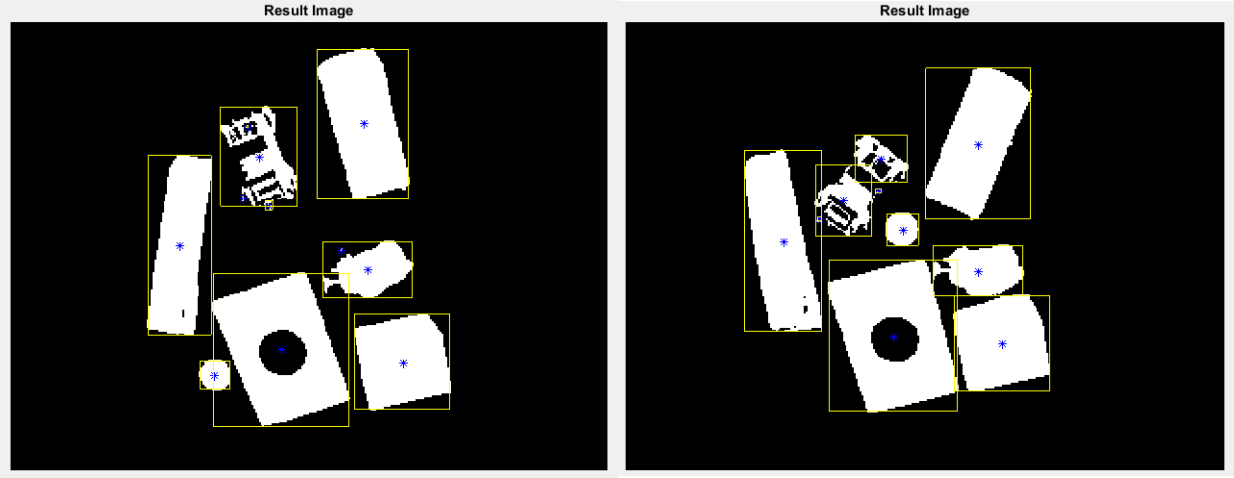


Figure 4. MBR and centroid plot for reg3.jpg and reg4.jpg

The following two table are the whole parameter of reg3.jpg and reg4.jpg

Max_X	Min_X	Max_Y	Min_Y	CT_X	CT_Y	Area	Prmt	N_Hole	A_Hole	Elongation
95	15	214	165	55	190	2562	203	0	0	16.0846994535519
99	46	154	113	73	134	825	246	5	135	73.3527272727273
57	56	128	127	57	128	2	3	0	0	4.5
168	72	108	74	120	91	2254	229	1	4	23.2657497781721
96	94	125	125	95	125	3	5	0	0	8.33333333333333
101	96	141	137	99	139	10	15	0	0	22.5
148	118	216	168	133	192	863	129	0	0	19.2827346465817
123	122	178	178	123	178	2	3	0	0	4.5
217	135	182	109	176	146	3330	229	1	486	15.748048048048
208	157	236	185	183	211	2022	167	0	0	13.7927794263106
197	182	118	102	190	110	211	46	0	0	10.0284360189573

Table 1 Parameters of reg3.jpg

Max_X	Min_X	Max_Y	Min_Y	CT_X	CT_Y	Area	Prmt	N_Hole	A_Hole	Elongation
106	25	217	161	66	189	2498	197	0	0	15.5360288230584
86	61	151	123	74	137	301	81	3	114	21.797342192691
166	69	105	64	118	85	2516	237	3	9	22.3247217806041
115	77	132	102	96	117	484	186	0	0	71.4793388429752
92	90	137	134	91	136	5	5	0	0	5
120	103	157	140	112	149	262	51	0	0	9.92748091603053
107	105	105	103	106	104	5	7	0	0	9.8
147	120	213	165	134	189	911	135	0	0	20.0054884742042
209	128	178	109	169	144	3416	239	1	496	16.7216042154567
198	147	227	176	173	202	2059	166	0	0	13.3831957260806

Table 2 Parameters of reg4.jpg

Part 4 Discussion and Conclusion

3.1 Discussion and Conclusion

I already made some conclusions in previous part, there are two more conclusions which are declared here.

To obtain the relatively reasonable connected components extraction, an important factor is how to choose a suitable threshold.

To get a well- separable results, it is nice to try smoothing algorithm, but in my implementation, it did not work well, thus I chose to use original image directly.

3.2 Future work

My future work is to speed up my code, especially the raster scan part.

References:

[1] Nikos Papanikolopoulos. “CSci 5561Computer Vision” *Slide1-3.*

[2] Wikipedia. “Conneted component labeling” *https://en.wikipedia.org/wiki/ Connected-component_labeling*