

Name: Yu Shixin

Matric. No.: A0195017E







Date: 2019/4/20

Part I. Noise cleaning

In order to clean up noise in the image, I use binary graphcuts from GCO-v3.0 toolbox to help me to do this experiment.

In this experiment, I separate the RGB value of image, firstly, and choose $[0,0,255]$ as foreground color, as well as $[245,210,110]$ as background color. Use kmeans function to divide original noise image's pixel label. Then apply the GCO toolbox to compute. I try different m_lambda value, and the results are shown as below:

Table.1 reconstruction picture with different m_lambda

lambda	Reconstruction picture	lambda	Reconstruction picture
1		20	
50		100	
150		200	

Compare with the original image Fig.1, even $m_lambda=1$, we can find the noise decrease a lot, from big spots to small spots. When $m_lambda=50$, the noise spots is almost invisible. We can get good result, while $m_lambda=100$. However, the bigger is not better for the value of m_lambda . When $m_lambda=150$, the blue part, which should be reserved, is removed.

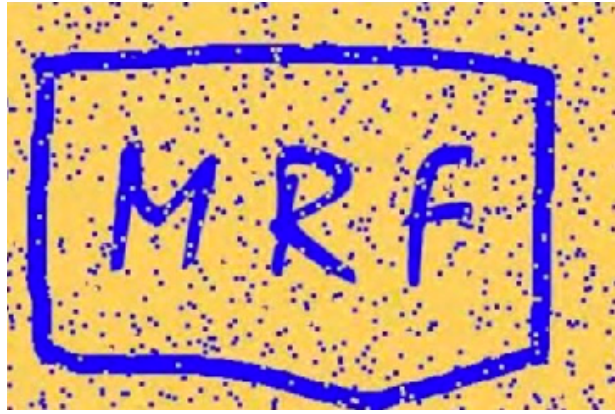


Figure.1 noise image

Part II. color segmentation

The second part is multi-classification, whose algorithm is very similar to the part 1. But in this part, I do not need to set center color by myself, just classify different color part not denoising. I use this picture shown in Fig.2.











Figure.2 part II experiment image

Thus, in this experiment, my aim is enter different K value (the number of the segmented colors) to observe the results.

According to the table.2, we can find that with the increase of K, the outline of every part of this picture is more and more clearly. The more K, the more color. When K=25 & 30, we can get a good cut-result. If we always increase the K, finally, we can get a quite goof result with the K big enough.

Table.2 Graphcut picture with different K

K	Graphcut picture	K	Graphcut picture
2		3	
4		5	
6		7	
25		30	

Drawback:

1. Graph cut suffers from the time complexity. It has a very high time complexity. When $K=30$, it spends 286.579113 seconds to run.
2. The minimum cut criteria occasionally supports cutting isolated nodes in the graph due to the small values achieved by partitioning such nodes.
3. Graphcut has huge memory consumption. To obtain high-resolution output, graph cuts usually build massive multidimensional grid-like graphs containing billions of nodes and even more edges. These graphs do not fit in memory.

Improvement:

We can regards 4 neighbor pixel as a unit, calculate their RGB value as their common value (the size of picture decrease quarter), and then use this graphcut algorithm to compute and expand the size to the original. Although we will loss some pixel information, the memory consumption decreases quarter.