Digital Image Processing HW3 Report

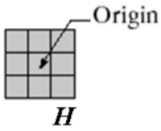
- ## Problem1

The original image(**sample1.png**) is the follows.

- (a) I performed boundary extraction via the following formula(in lecture slides).

$$\beta(F(j,k)) = F(j,k) - (F(j,k) \Theta H(j,k))$$

where H is the 3-by-3 mask.



The following is my result(**result1.png)**

The reuslt is nice.

- (b)

  My hole filling process consists of two steps.

  Step1: Fill the background outside all objects

  I perform the following method to fill the background(in lecture slides)

$$G_i(j,k) = (G_{i-1}(j,k) \oplus H(j,k)) \cap F^c(\mathrm{j},\mathrm{k}) \quad i = 1,2,3...$$
$$G(j,k) = G_i(j,k) \cup F(\mathrm{j},\mathrm{k})$$

  G_0 is image with only one pixel on the upper-left corner. By recurssion, G_i will gradually evolve to the image filling all holes with starting point indexed by G_0. Once we find there is no difference between G_i and G_{i-1}, we stop the recursion.

  The following is my result after the first step. The black boundary is caused due to the lack of processing on boundaries, which is just for convinience and can be easily done if needed.

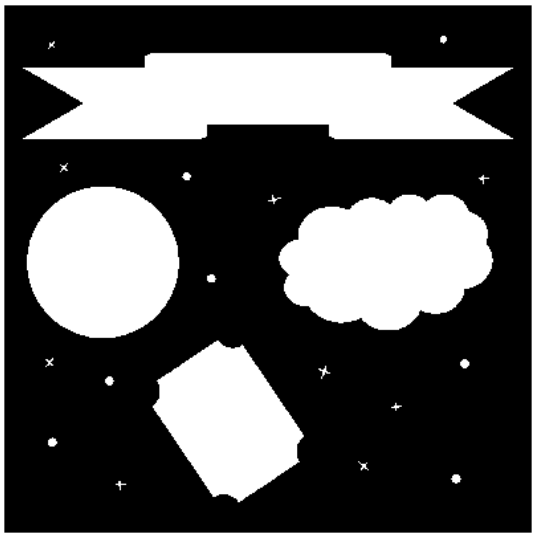
  Step2: Complement the background-filled image, and union it with the original image.

  As wee see, when we take the complement of the background-filled image, only holes inside objects will be bright. The below image of taking the complement shows this feature.

background-filled image

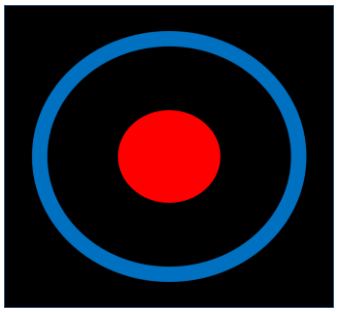
background-filled image (complement)

  One can see the complement is exactly all the holes in the image. Then by performing union with original image, we can fill up the holes.The following is my result(**result2.png**)



- (c)

  First we specify “what is an object”. Take the following figure as an example:



Whether the blue part and the red part are considered to be one same object is a question. Let’s say **definition 1** to be what if the red circle is combined with the blue ring to be one object, and **definition 2** is which they are independent 2 objects.

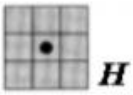
The main idea of my algorithm to count is stated as follows:

Step1: Find a while pixel(foreground) and do connected component labeling with it. If no white pixels, end the algorithm.

I use the following formula to connect all neighbors(in lecture slides).

$$G_i(j,k) = (G_{i-1}(j,k) \oplus H(j,k)) \cap F(j,k) \quad \mathrm{i} = 1,2,3,...$$

where H is the 3-by-3 mask(according to 8-connectivity)



Similarly as before, $\mathrm{G_i}$ gradually evolve to contain the whole connected object. Once we find there is no difference between $\mathrm{G_i}$ and $\mathrm{G_{i-1}}$, we stop the recurrsion. Say the result image is G.

Step2: Decrement the original image with G, increment 1 to the counting, and go back to Step1:

By decrementing, we eliminate the previous object we labled from the original image. And we can fo back to ***Step1*** to find the next object.

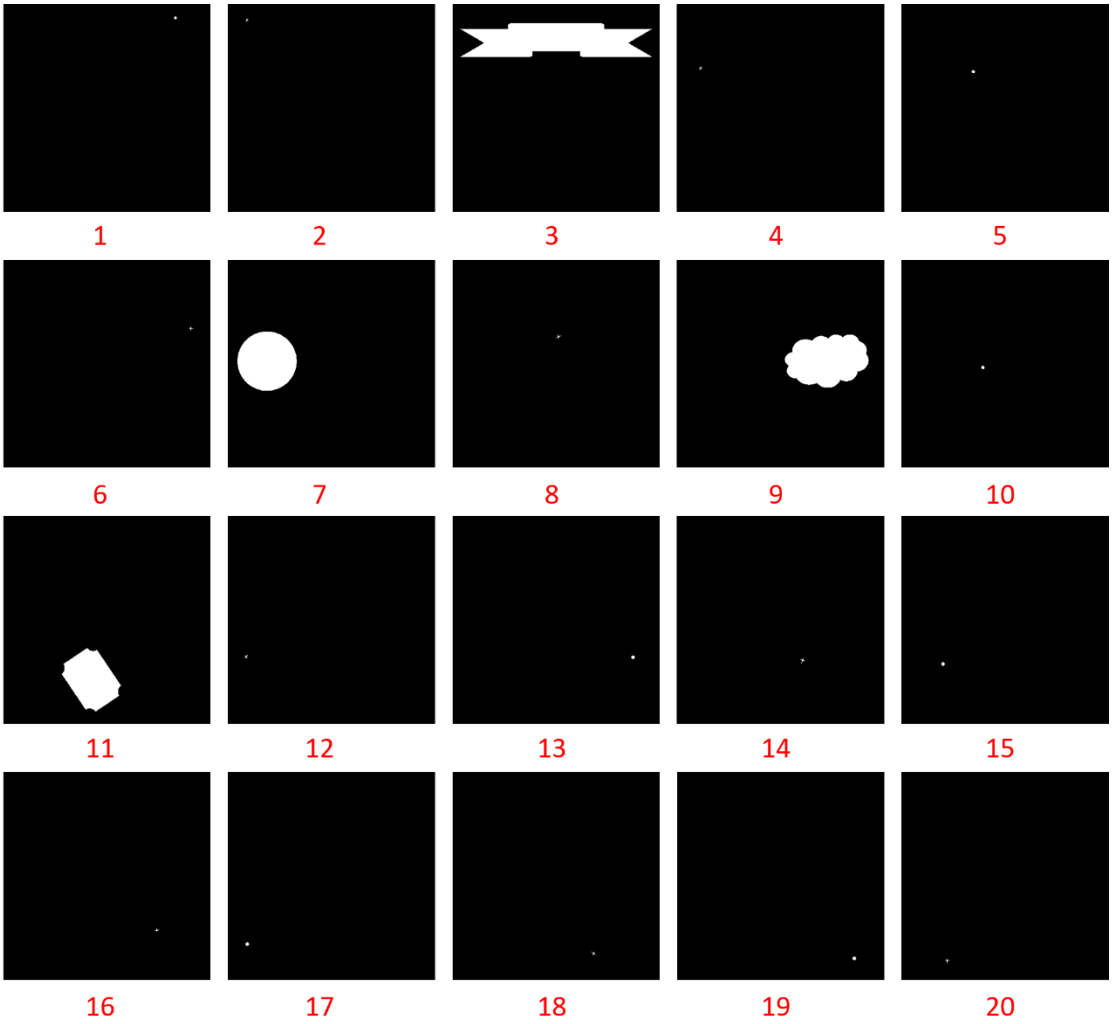
Once there is no white pixel in original image, all objects are eliminated and counted.

The difference of **definition 1** and **definition 2** decides what we use to be the input image. We use the hole-filled image(like **result2.png**) to be the input image for **definition 1**, while in **definition 2** we need to use **sample1.png**.

I show both results in below.

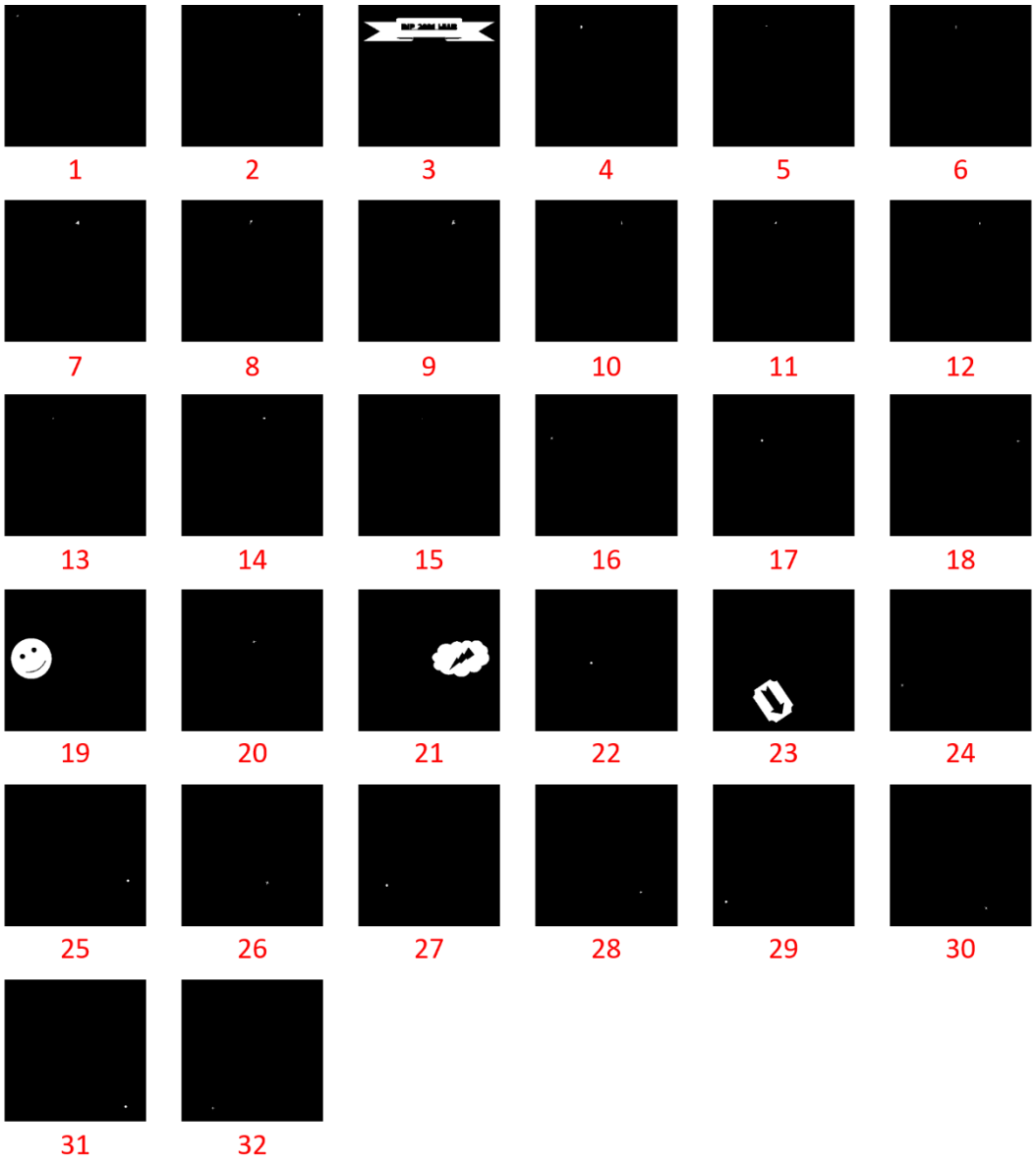
Definition 1: **Count 20**

The following are all the 20 objects.



***Definition 2*: Count 32**

The following are all the 32 objects.



One can compare with **Definition 1** to see the fact that most objects are included in **Defintion 2**. The main difference is the text in the upper flag. In **Definition 1**, the whole flag is considered to be an object, while in **Definition 2** several components in words are separated and independently considered.

- Probelm2

  Here is the original image(**sample2.png**).

- (a)

  I perform Laws' method by the following two steps:

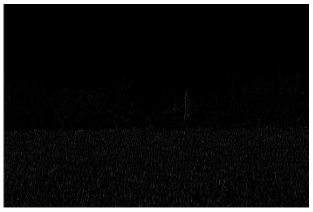
  Step1: Convolution

  Convolution original image with 9 masks(the **Laws-1** to **Laws-9** in lecture slides)

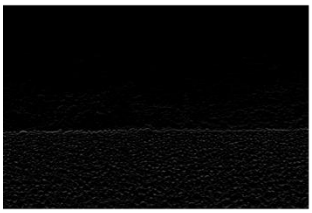
  The following are the 9 results, labeled with M_1 to M_9

M_1

M_2

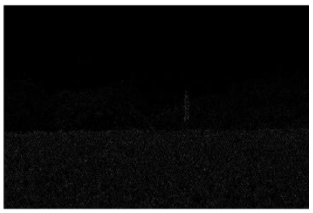


M_3

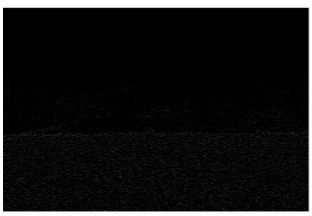


M_4

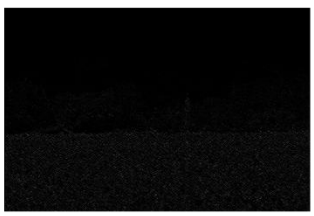
M_5



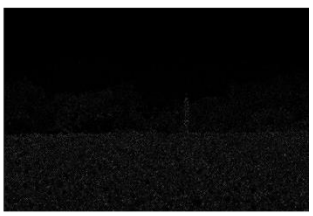
M_6



M_7



M_8



M_9

Each M is constituted with three 1-dimensional masks –L_3, E_3, S_3, which represents local averaging, edge detection and spot detector, respectively. Hence we can see some features in above image.

For example, M_1 is constituted by row-L_3 and column-L_3. As a result, it acts like going through a low-pass filter. M_4 is consists of a row-E_3 and column-L_3. We can see in horizontal direction, pixel intensity is averaged, while in vertical direction, the edge between flowers and trees are intensified. In M_9, all flowers are enhanced. This is because there exists a spike at every flower.

Step2: Energy Computation

Set 9 2-dimensional arrays T_1 to T_9 with size equal to the image. Their values are computed by sum of squares of M_1 to M_9:

$$T_i(j,k) = \sum_{m \in w} \sum_{n \in w} M_i(j+m, k+n)^2$$

where w implies the mask-size, in which I set $\mathrm{w} = \{-6, -5, \dots, 5, 6\}$ (mask-size = 13)

Since the values may go over 255, I save T in .npy-file.(**T.npy**)

- (b)

I use k-means algorithm with k=4, where I guess there are 4 textures –the mountains(background), the mountains(foreground), the trees, and the flowers . Define the **T-value** of a pixel to be a 1-dimensional, 9-lengthed vector, which has coordinates' values be "the corresponding pixel value of T_i, $i \in \{1,2, \dots ,9\}$".

First I set 4 **T-value**s as standards, say standard 1, 2, 3 and 4, and fill their values with random number,s ranging from zero to the max value of T_i.

Next for each pixel, I compute their distance with the 4 standards, where the distance is defined to be the sum of squares of the difference in each coordinates. Find the smallest one among the 4, and assume the smallest to be the distance with standard $\boldsymbol{s}$.

Then I add this pixel into a set, say set $\boldsymbol{s}$, and continue to compute the next pixel distance with standards and add it into some set. In this way, all pixels will be categorized to a set.

Now, upadte the **T-value**s of the 4 standards. Compute the average **T_value** of each set and set them to be the new standards. Then again, re-categorized all pixels with their distance with the 4 new-updated standards and add them to the "nearest" sets .

The iteration stops when the **T-value**s of the 4 standards are unchanged. At this time, the 4 sets should contain pixels of different textures. Then I color each pixel with different colors according to their corresponding set.

Using Color:

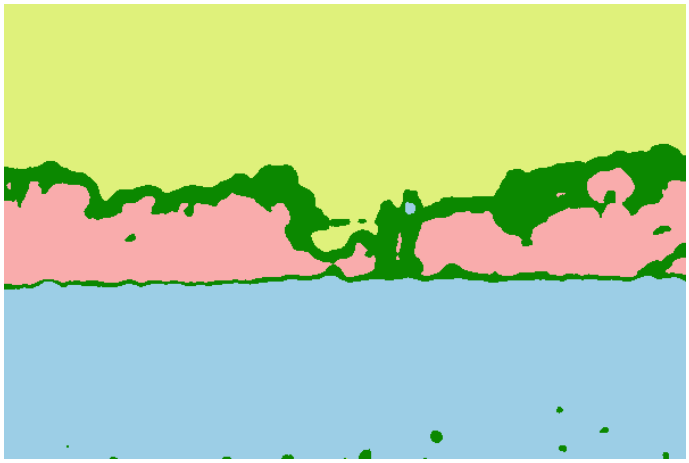

(B,G,R)=(230,206,156)

(B,G,R)=(123,241,223)


(B,G,R)=(123,241,223)


(B,G,R)=(172,172, 249)

The result is the following(**result3.png**).



- (c)

The textures classified in **result3.png** did not meet my expectations. Explicitly speaking, the mountains in background and foreground are not separated. Hence I decided to intentionally choose the initial standards; that is, instead of using random numbers to be initial **T-value**s, I selected 4 “initial pixels”, which are pixels with coordinates [25, 200], [120, 185], [190, 220], [350, 210]. These 4 pixels correspond to the mountains(background), the mountains(foreground), the trees, and the flowers in original image. I set their **T-value**s to be the initial standards. In this way I hope the textures classified in the result to be the textures of the 4 coordinates.

The following is my result(**result4.png**).

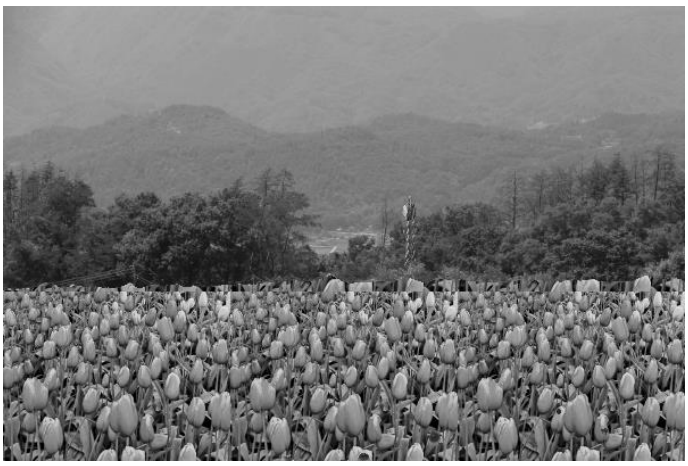
As I wanted, the mountains in background and foreground are separated.

- (d)

  I try the gray one. I simply try to replace all textures of the flowers(the pink area as the above **result4.png**) to the given image(**sample3.png**).

  For each pixel of flower-texture, I replace it with the corresponding pixel in the pattern image. The corresponding rule is as follows: Let the pattern-image size *H×W*, for pixel *(i, j)* in original image, use pixel *(i (mod H), j (mod W))* in pattern image to do the replacing.

  The result is as the follows.



  Since there is an obvious border in image, around the flower and the trees, this method is not good enough.