HOMEWORK ASSIGNMENT #3
Morphological Processing, Texture Analysis

Poy Lu 呂栢頤
D09944015
網媒所博一
ariapoy@gmail.com

April 21, 2021

1 Problem 1: MORPHOLOGICAL PROCESSING

A binary image, **sample1.png**, is given in Figure 1. Please implement several morphological operations and provide discussions about the results. (Note that the white pixels represent foreground objects and the black pixels are background.)

Given an image, **sample1.jpg**.

Original image **sample1.jpg** for question Problem 1: MORPHOLOGICAL PROCESSING.

1.1 (a)

Perform boundary extraction on **sample1.png** to extract the objects'boundaries and output the result as **result1.png**.

Motivation Extract the boundary (or outline) of an object. Recall of generalized **dilation** and **erosion** in *Lec 4 page 32.*

- dilation: $G(j,k) = F(j,k) \oplus H(j,k)$
- erosion: $G(j,k) = F(j,k) \ominus H(j,k)$

where $H(j,k)$ is **structure element**.

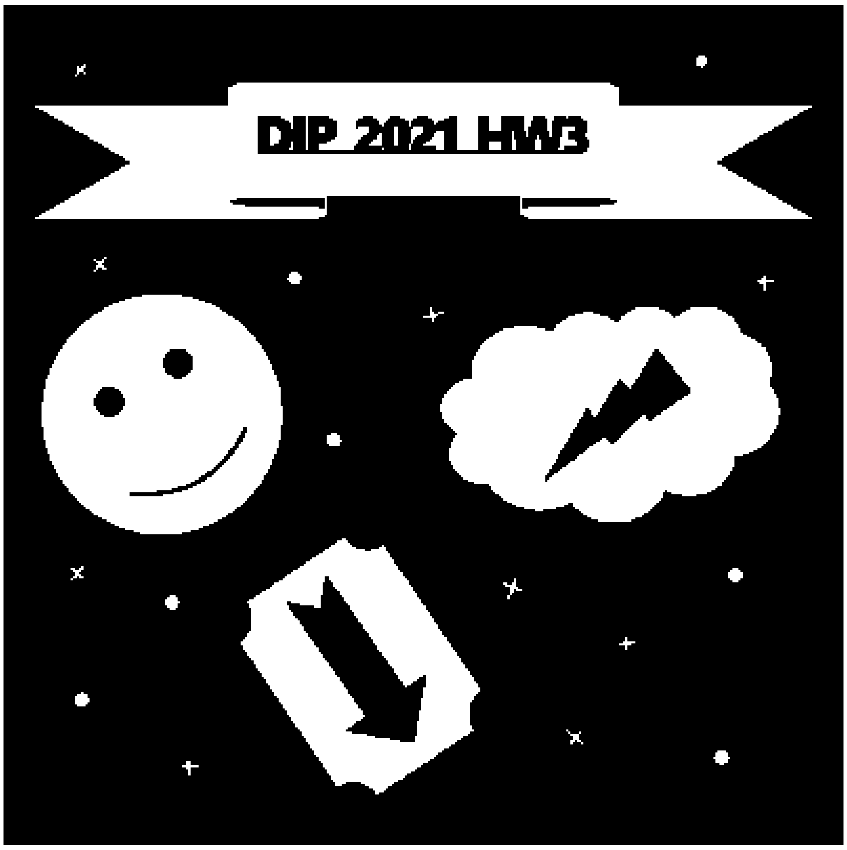

Figure 1: **sample1.jpg**

Approach We could follow the formula in *Lec 4 page 39*. The formula of **boundary extraction** is

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

It is the original image **minus** its erosion.

I start from generalized **dilation** and **erosion**. This is my **core function** for (a), (b) & (c).

1. Convolution with **structure element** $H(j,k)$ to get $T_{r,c}\{F(j,k)\}$.
2. Conduct **erosion** or **dilation**:
   - **erosion**: $G(j,k) = \cap_{(r,c)\in H} T_{r,c}\{F(j,k)\}$
     As we **intersection** of all $T_{r,c}$, we could check the **sum of matching** $T_{r,c}$ is **equal** of # 1 in $H_{j,k}$.
   - **dilation**: $G(j,k) = \cup_{(r,c)\in H} T_{r,c}\{F(j,k)\}$
     As we **union** of all $T_{r,c}$, we could check the **sum of matching** $T_{r,c}$ is **greater than** 0.

This approach is to transform **intersection** & **union** as **convolution with condition sum**.

Then conduct the **boundary extraction** by formula.

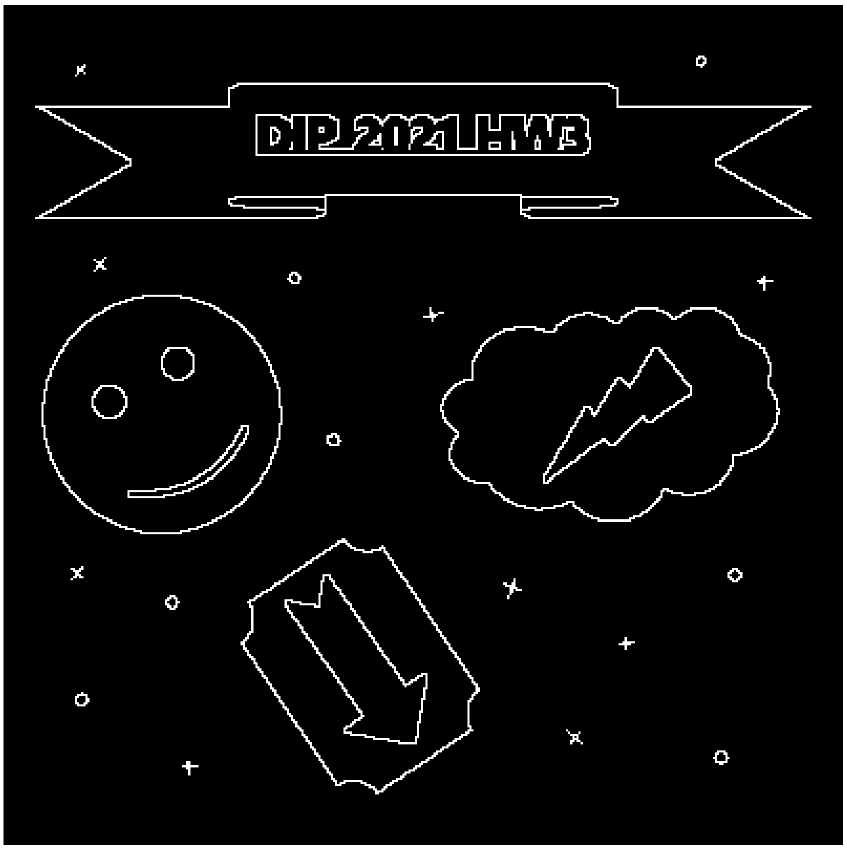

Figure 2: **result1.jpg** Boundary extraction

1. Design the **structure element** $H(j,k)$ as well as **origin points**.
2. Calculate **erosion** $F(j,k) \ominus H(j,k)$
3. Obtain **boundary extraction**.

Performance of results In the end, I choose same **structure element** in *Lec 4 page 39*, that is

$$H(j,k) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

with **center point as origin**.

Result of problem 1(a): **result1.jpg** Boundary extraction.

Discussion I try some different **structure element**.

Different of structure element:

$$H(j,k) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

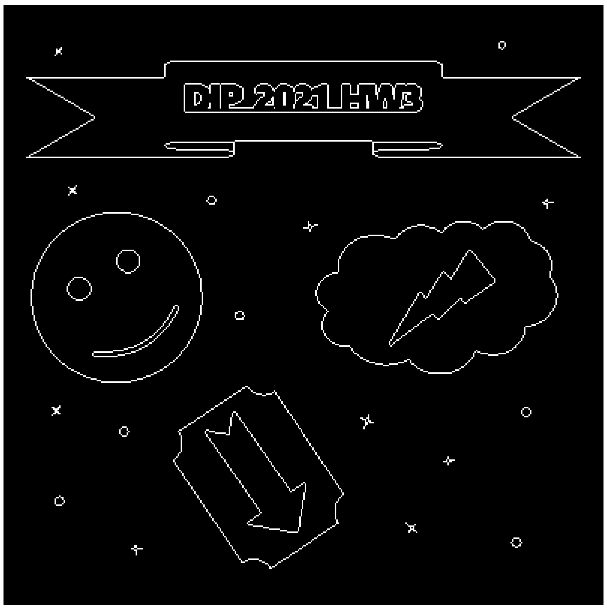

Figure 3: **result1.jpg** Boundary extraction with **cross** structure

The boundary of **result1.jpg** Boundary extraction with **cross** structure is thinner than **result1.jpg** Boundary extraction.

Different size of structure element:

$$H(j,k) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The boundary of **result1.jpg** Boundary extraction with $\mathbf{1}_{5\times5}$ structure is thicker than **result1.jpg** Boundary extraction.

The above results is obvious as we try the larger size filled with 1. The number of 1 result the thick/thin boundary of objects.

1.2 (b)

Perform hole filling on **sample1.png** and output the result as **result2.png**.

Motivation Given a pixel inside a boundary, hole filling attempts to fill that boundary with object pixels.

First, this operation uses **dilation**($\oplus$). We could reuse the core function designed in (a). But I find out the result is not good as my expectation.

Second, as we could treat it as *Fill tool in Painter*, we could use flood fill algorithm for this problem.

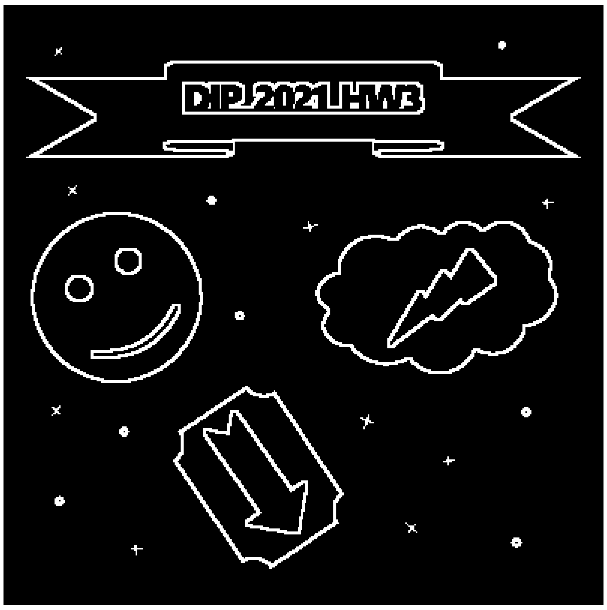

Figure 4: **result1.jpg** Boundary extraction with $\mathbf{1}_{5\times 5}$ structure

Approach My idea is start from **breadth-first search (BFS)**. The idea comes from LeedCode discussion.

1. Initialize empty queue and add the **start point** to fill.
2. Scan the color of position (i, j)
   - if the current is not visited &
   - the color of current is the same as queue of the start point.

   We fill the new color of this pixels.

I use the trick that start from the **top-left corner pixel** with **black**. And fill up all **background** as **tag** (2). Then I keep it as background and replace other pixels with **white color** (255).

Performance of results In the end, I choose the **flood fill** algorithm.

Result of problem 1(b): **result2.jpg** Hole filling with flood fill.

Discussion We could follow the formula in *Lec 4 page 41*. The formula of **hole filling** is

$$G(j,k) = ((G_{i-1}(j,k) \oplus H(j,k)) \cap F^c(j,k)) \cup F(j,k)$$

where $i = 1, 2, \ldots,$ $F^c(j,k)$ means the **complementary** of original image. It is the **interior** of original image **union** with **boundary**.

My approach is

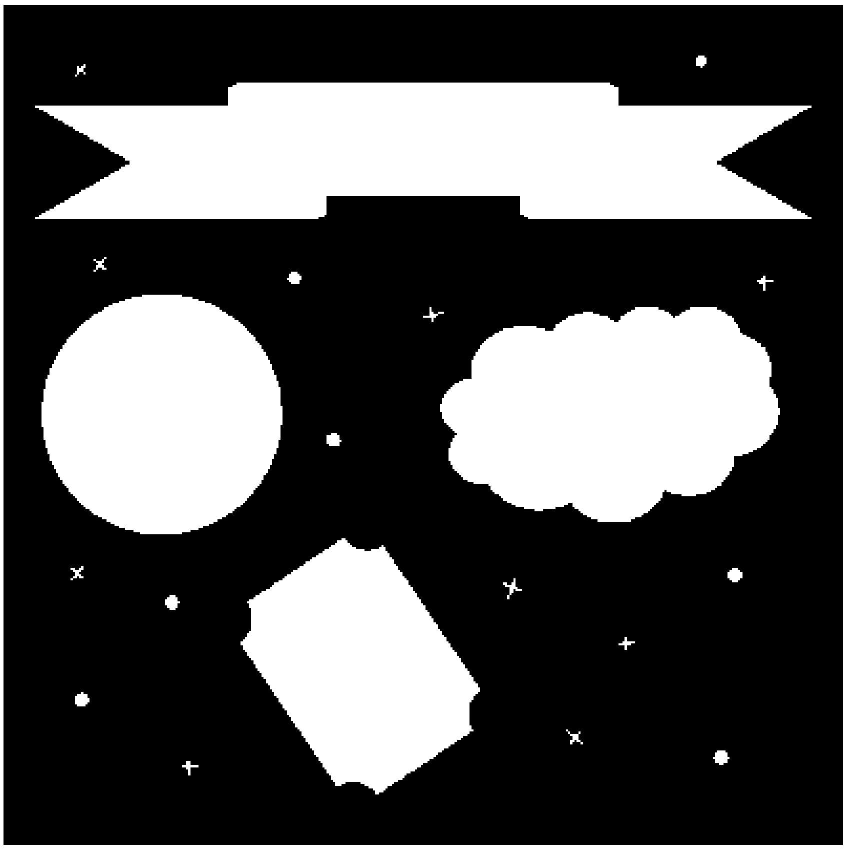

Figure 5: **result2.jpg** Hole filling with flood fill

1. Calculate complementary $F^c(j,k)$
2. Dilation then **intersection** with complementary $(F(j,k) \oplus H(j,k)) * F^c(j,k)$
   As we just get $\{0,1\}$ of dilation as well as complementary results, we could times $(*)$ them as **intersection**.

Moreover, I add parameters of # **repeated** of hole filling operations.

Give the same **structure element**

$$H(j,k) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

with **center point as origin**. And repeat hole filling with **12** times.

Here is the result: **result2_lec4.jpg** Hole filling in Lec 4 page 41

Although **result2.jpg** Hole filling with flood fill shows we successfully fill all holes, it expands original objects and connect to each other. Why does it occur? I consider that my approach is not as good as *Lec 4 page 41*. As I don't choose the **interior points** in the objects. So all objects become larger and larger when we expand on the **boundary of objects** then intersection with background.

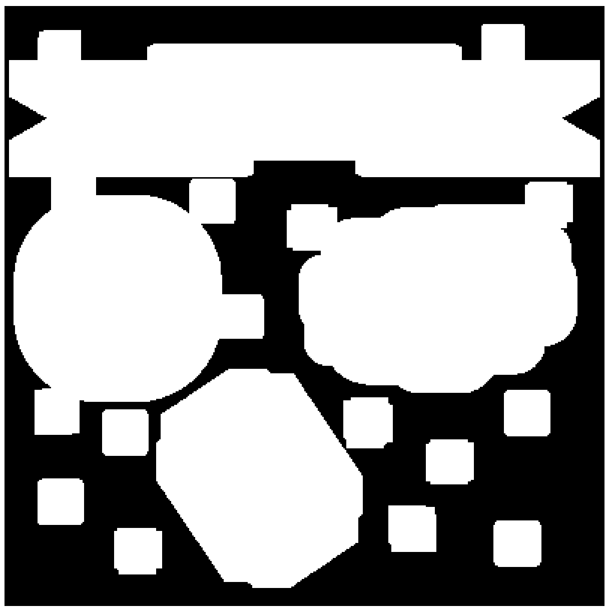

Figure 6: **result2_lec4.jpg** Hole filling in Lec 4 page 41

1.3 (c)

Please design an algorithm to count the number of objects in Figure 1. Describe the steps in detail and specify the corresponding parameters.

Motivation Scan an image and groups its pixels into components based on pixel connectivity.

First, this operation uses **dilation**($\oplus$) again. We could reuse the core function designed in (a). However, I can't count the number of objects as I don't calculate it **iteratively**.

Second, I use **two-stage scan** for couting the number of objects. The detail steps is shown below.

Approach My approach is

1. Initialize the object with tag (-1).
2. Scan the 8-neighbors of object $F(j,k) \neq 0$ as `G_subarray`.
    - if there are more than two tags in `G_subarray`, we store these tags as **connected cluster**.
    - else substitute the new cluster tag for -1 or propogate existed tag in `G_subarray`.
3. Get `label_object_adj_mat` from **connected cluster**.
4. Check the connectivity of label object adjacency matrix by $\text{sgn}(A^k)$.
   where k is number of clusters (candidate label objects) and $\text{sgn}(\bullet)$ is **sign function**.

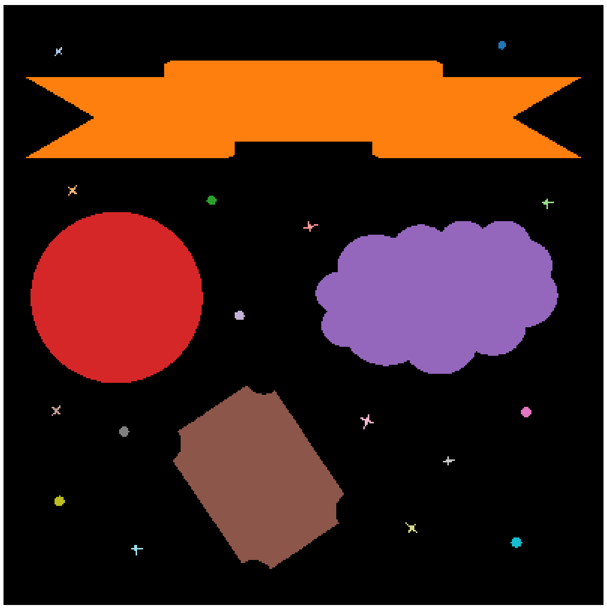

Figure 7: # 20 Objects with color

It works based on the properties of **adjacency matrix**. The A^k means the **steps** from position row i to column j. And 1 means (i, j) are connective after walks k steps.

5. Merge the connective clusters as final label objects. And count it!

Performance of results I use the **result2.jpg** Hole filling with flood fill than count it!

Result of problem 1(c): # 20 Objects with color with 20 objects.

Discussion We could also conduct it for **result2_lec4.jpg** Hole filling in Lec 4 page 41. Result of problem 1(c): # 20 Objects with color with 11 objects.

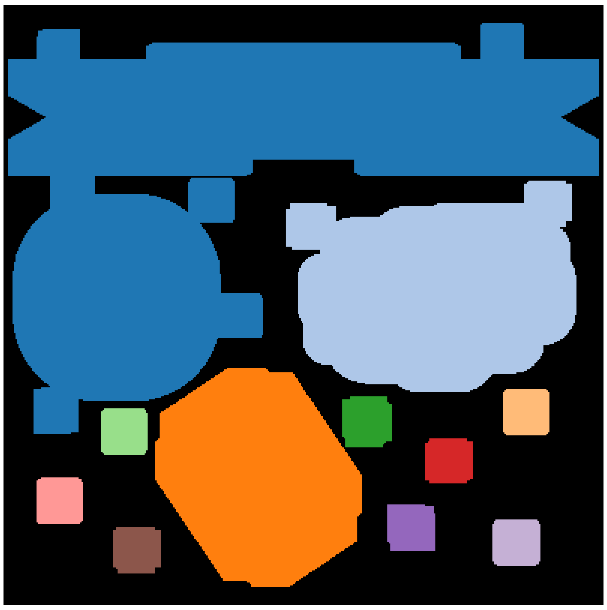

Figure 8: # 11 Objects with color (Lec 4)

2 Problem 2: TEXTURE ANALYSIS

In this problem, there is an image **sample2.png** composed of several different textures.

Given an image, **sample2.jpg**.

Original image **sample2.jpg** for question Problem 2: TEXTURE ANALYSIS.

2.1 (a)

Perform Law's method on sample2.png to obtain the feature vector of each pixel and discuss the feature vectors in your report.

Motivation Micro-structure (Multi-channel) method. It emphasize the microstructure of the texture with 9-types of **filtering** (low-pass/local average, edge detector/1$^{\text{st}}$-order, spot detector/2$^{\text{st}}$-order) then compute **energy distribution** of these subbands.

Approach We could follow the two steps of Law's method in *Lec 5 page 16–25*.

1. Convolution $M_i(j,k) = F(j,k) \otimes H_i(j,k)$.
   where $H_i(j,k)$ is **micro-structure impulse** response arrays.
2. Energy computation with $T_i(j,k) = \sum_{(m,n)\in w} |M_i(j+m, k+n)|^2$.
   where w is **window size**.

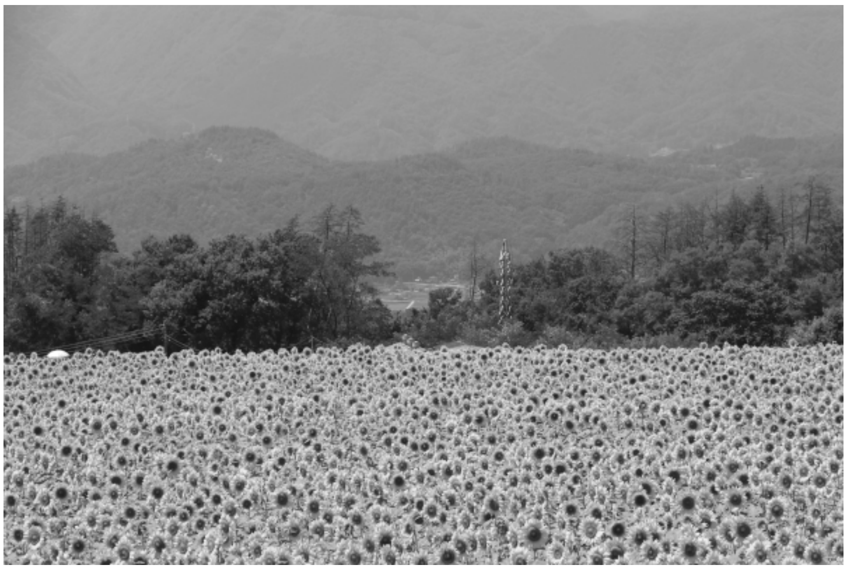

Figure 9: **sample2.jpg**

	feature 1	feature 2	feature 3	feature 4	feature 5	feature 6	feature 7	feature 8	feature 9
mean	632578.51	12606.37	10740.66	12496.21	14566.39	17710.59	10324.98	18076.89	30689.82
std	255219.15	14841.65	12589.02	14507.87	16546.63	19929.37	11479.35	19661.54	33759.59
min	24443.18	5.72	10.08	5.4	14.94	28.06	10.03	24.63	46.38
25%	503566.26	111.81	114.33	222.96	189.88	227.88	178.12	264.81	391.8
50%	741495.28	3197.15	3338.21	3865.16	4423.5	6215.19	4084.01	7322.06	12601.59
75%	818331.18	26522.01	21761.25	25030.34	30454.5	36466.03	20445.08	36738.09	61693.36
max	1168527.54	65892.08	73024.13	122812.59	72651.31	89798	69072.48	91929.81	145748.75

Table 1: Prob 2(a) Summary statistics

Performance of results In the end, I choose the window size $w = 13$.

Result of problem 2(a):

Shape: $(400, 600, 9)$ We could see the Law's method generate 9 features for each pixel.

Statistics: I export summary statistics to the `tmp/prob2a_stats.csv`.

The Prob 2(a) Summary statistics tell us:

- We could see the **feature 1/ LP×LP** is relative larger than other features. With mean 632578.
- The **feature 9/ HP×HP** is second large features. And its **standard deviation** is largest one and only one that larger than its mean. With std 33759.

Then we could observe the **scatters** Scatter matrix to get the correlation between features.

- Most of features are **positive skew**. Only **feature 1** is relative uniform.

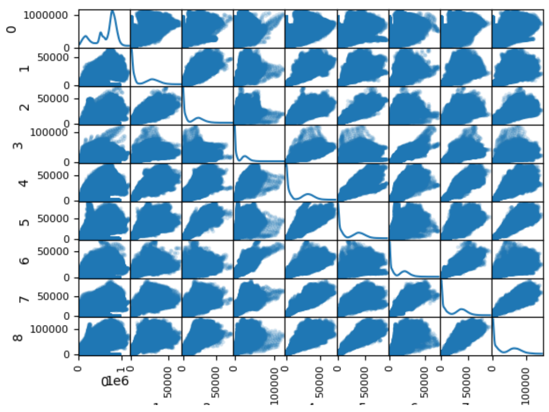
Figure 10: Scatter matrix

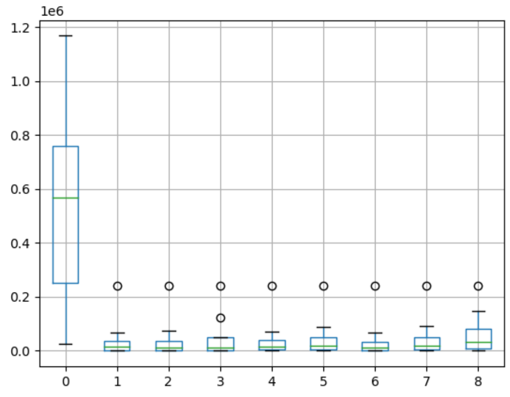
Figure 11: Box plot

- It seems **feature 9** are positive correlative with **feature 4, 5, 7**.

Then, we observe the **outlier** Box plot of each features. We could see the **feature 3/ LP×HP** have 2 **larger outliers** than others.

Finally, we observe the **probability–probability plot** Probability plot to evaluate the skewness of a distribution. We could check if we **remove/drop** 0 values, most of our features are not skew. The special features are

- **feature 1** which is original uniform than others.
- **feature 4/ BP×LP** is skew up.

For this result, we could design **row features** to enhance our performance when classifying.

Discussion I'm not sure the scale/normalization of $F_{j,k} \in [0, 255]$ or $F_{j,k} \in [0, 1]$. My intuition tells me the difference occurs at **energy computation**. Unfortunately, I keep

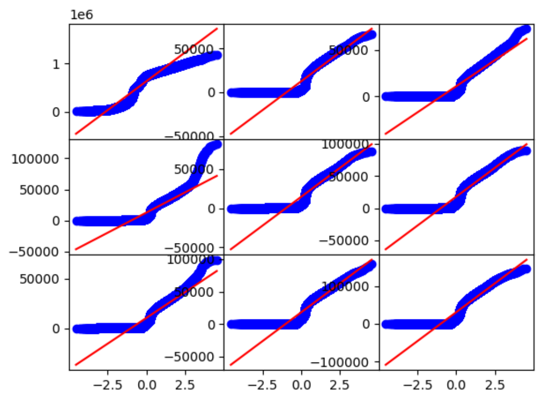
Figure 12: Probability plot

this issue in future.

2.2 (b)

Use k-means algorithm to classify each pixel with the feature vectors you obtained from (a). Label same kind of texture with the same color and output it as **result3.png**.

Motivation Un-supervised texture classification. The good classification is **large inter-clustering** as well as **small intra-clustering**. Follow the characteristics of this image, we could choose $k = 4$ or $k = 4$ as our parameter.

Approach Follow this gist,

1. Initialize k vectors as the initial centroids. This approach select the **existing points** in training data.
2. With `maxIters` iterative times:
   - Form k clusters using nearest neighbor rule.
   - re-compute the centroid of each cluster.

After we get the result of clusters, I use `matplotlib.plt.cmap` to assign the colors for each cluster.

Performance of results In the end, I choose the $k = 4$.

Result of problem 2(b): **result3.png** K-means of Law's method. The top *sky* is similar to *flowers*. It seem not a good approach.

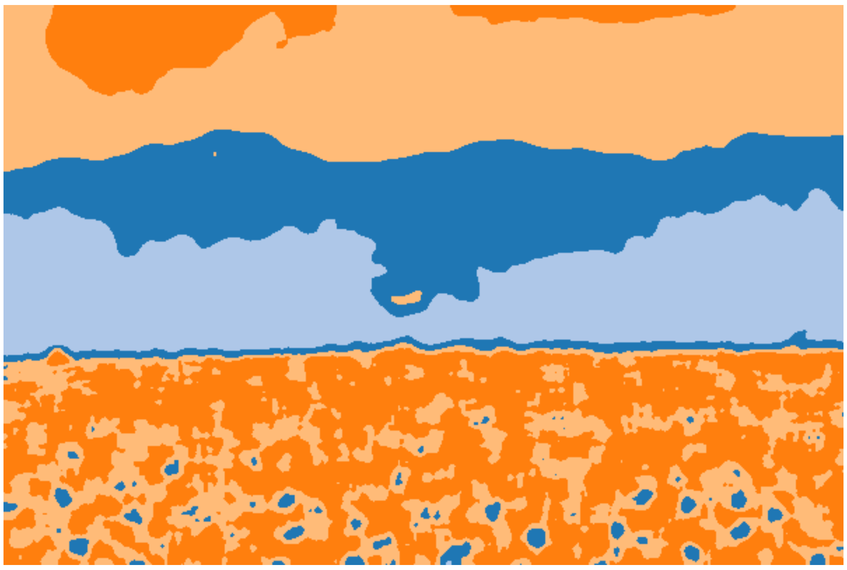

Figure 13: **result3.png** K-means of Law's method

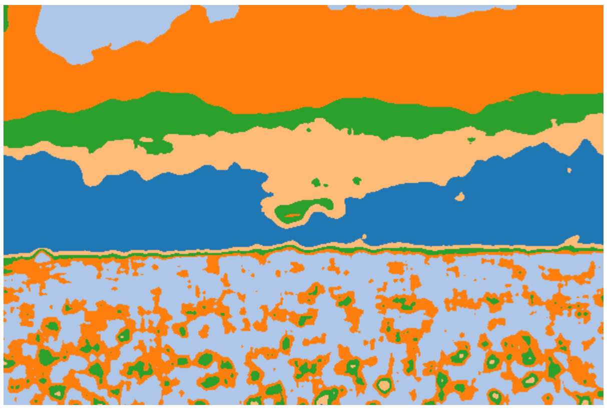

Figure 14: **result3.png** K-means with $k = 5$

Discussion For $k = 5$, we have **result3.png** K-means with $k = 5$ We could see the separate between top *sky & tree* and down *flowers* is better. But it has some space to improve.

Also, I find out it is **unstable** of my result. I consider that the **initial centroid points** in the beginning of **K-means** cause this problem.

2.3 (c)

Based on **result3.png**, design a method to improve the classification result and output the updated result as **result4.png**. Describe the modifications in detail and explain the reason why.

Motivation Carefully observe this image, most of textures are **horizontal**. We could add the features to enhance this information!

Approach My approach is

1. Add **row position/index** as new feature on each pixel.
2. Make this feature scale more closer to other features. $X_{10}^{\text{new}} = \delta X_{10}$
   $\delta = \frac{\bar{\bar{X}}}{200}$,
   where $\bar{\bar{X}}$ is mean of average of features.
3. Conduct **k-means** algorithm.

Performance of results I follow the settings of **result3.png** K-means of Law's method than add the above feature!

Result of problem 2(c): **result4.png** K-means add row-enhance feature

Discussion After we add the **row enhancement feature**, it seems better than **result3.png** K-means of Law's method. I consider that **hand-craft features** is important of imporvement. But I don't take use the advanced feature engineering and classifiers, maybe there are better approach to remove some **spots in large region**.

Given an image, **sample3.jpg**.

Original image **sample3.jpg** for question (Bonus).

2.4 (Bonus)

Try to replace the flowers in color or gray-scale **sample2.png** with **sample3.png** or other texture you prefer by using the result from (c), and output it as **result5.png**.

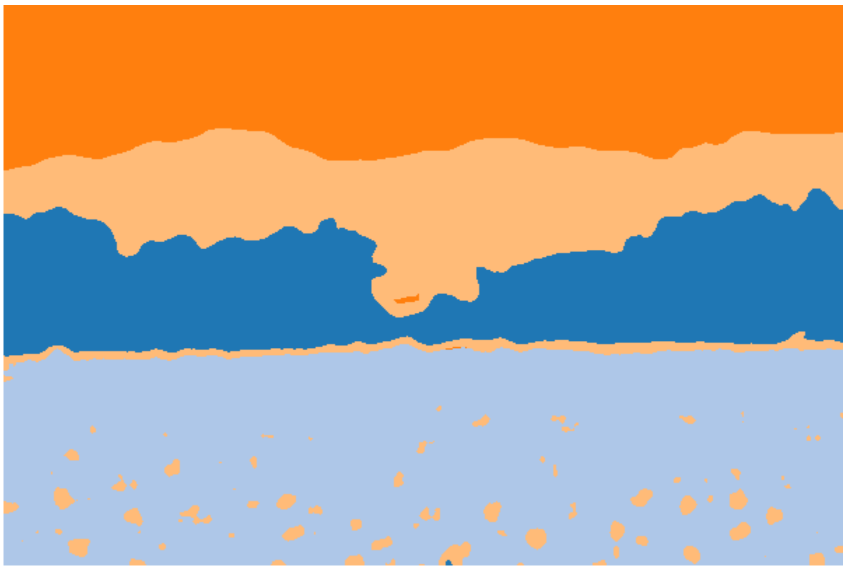

Figure 15: **result4.png** K-means add row-enhance feature

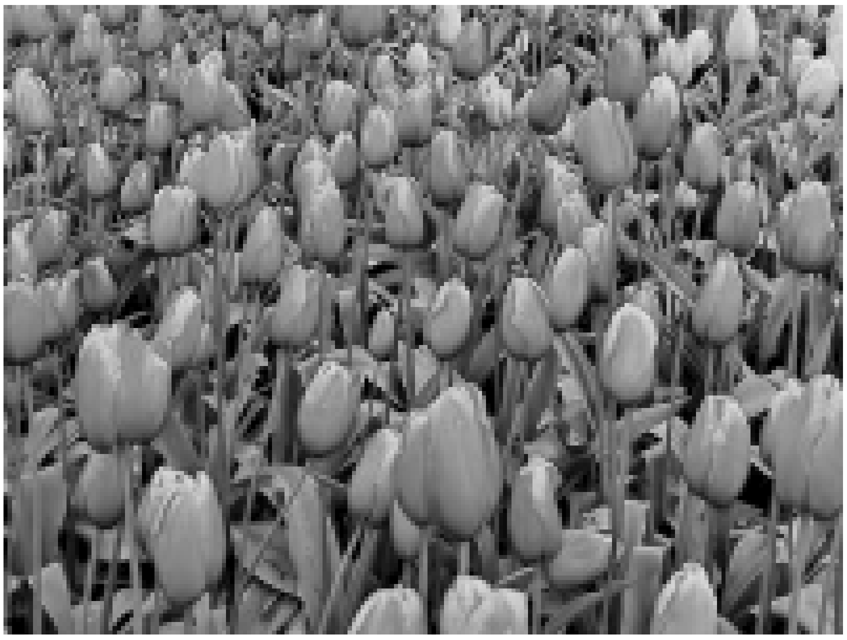

Figure 16: **sample3.jpg**

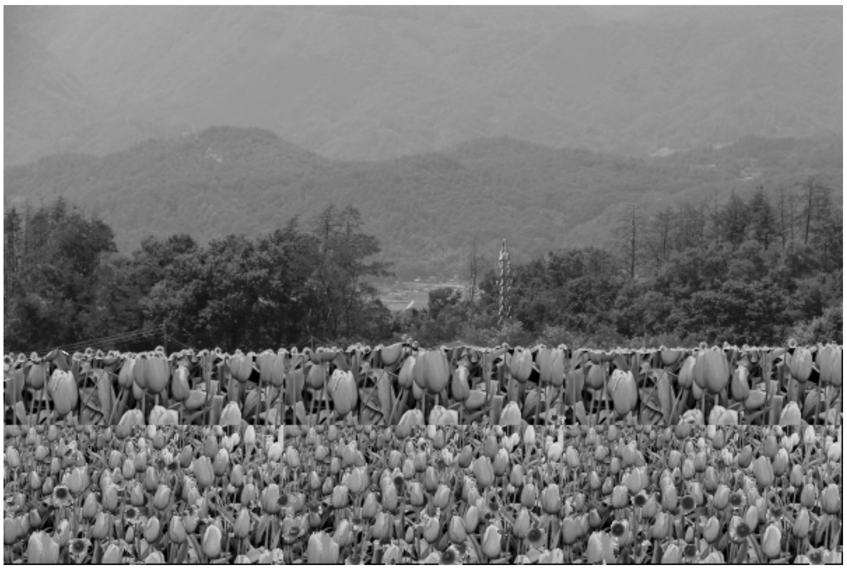

Figure 17: **result5.png** Texture replacement

Approach My approach is

1. Use the intermediates of cluster results, which means the **tags of each pixel**.
2. Expand the **sample3.jpg** to $(450, 600)$ that larger than **sample2.jpg**.
3. Replace the **tags of flowers** with **sample3.jpg**.
4. Keep other tags as original **sample2.jpg**.

Performance of results Result of problem 2(Bonus): **result5.png** Texture replacement

Discussion Thanks to TA's advice. I try to expand the **sample3.jpg** that could be assigned to new figures. But it still has some **strange boundaries** at **horizontal view**. Maybe I could try better in future.