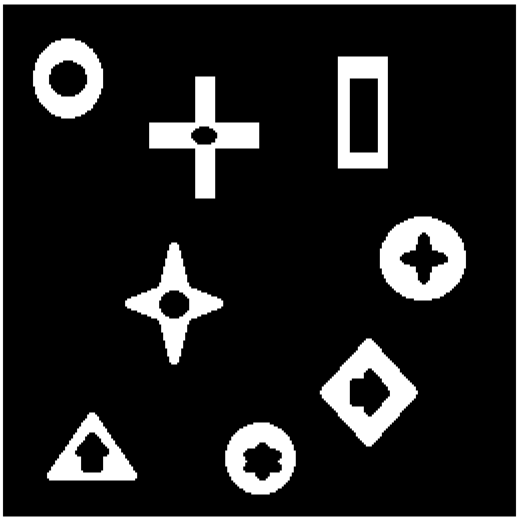
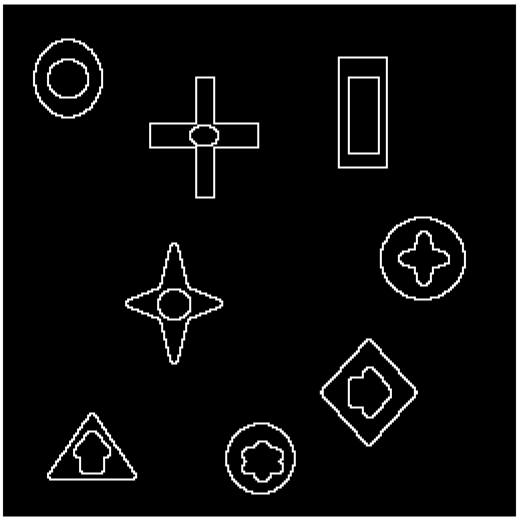
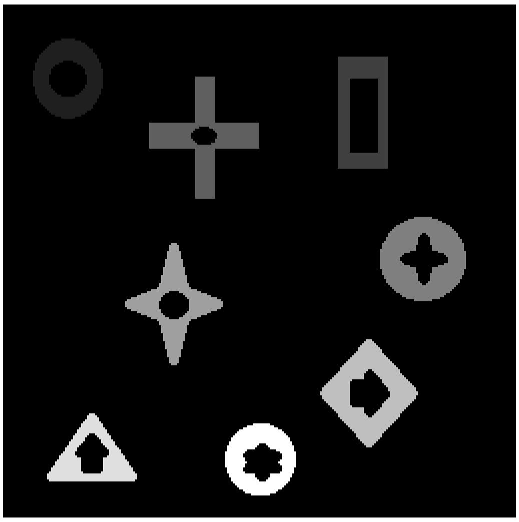
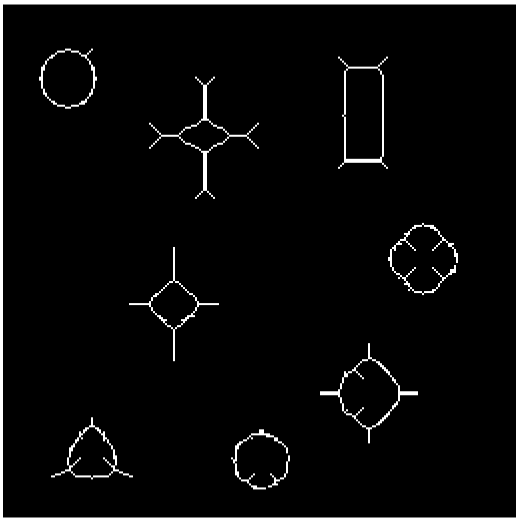
DIP HW3 Report

B03902082 資工四 江懿友 big2632@gmail.com

Problem 1: Morphological Processing

Input image	Boundary extraction (B)
	
Connected component labeling	Skeletonizing (S)
	

(a) Boundary extraction

I use morphological erosion to extract object boundaries. Because every object in input image has one hole, so every object have two boundaries, the inner and outer boundaries.

(b) Count object

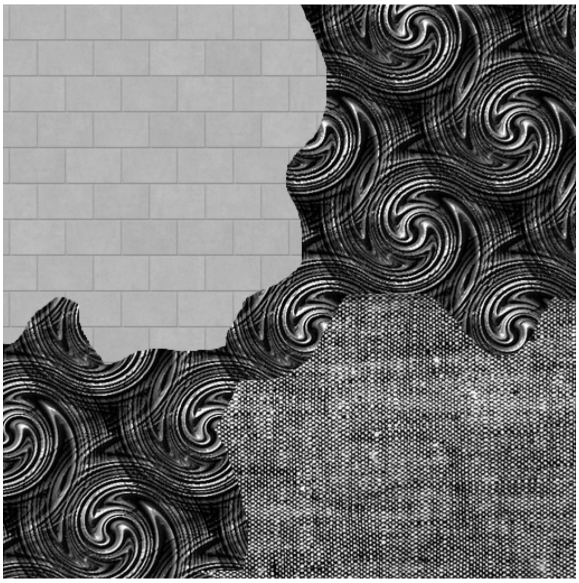
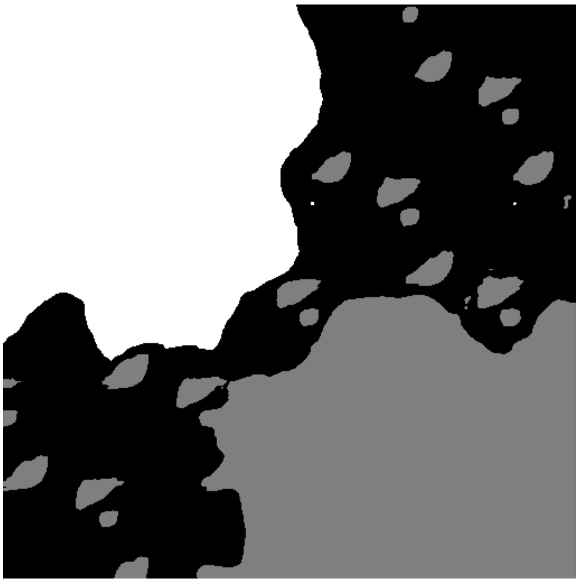
I use morphological dilation to label and count the connected components (object) in input image.
For each unlabeled pixel, if the pixel value is 1 then I assign it a new label and perform dilation on the pixel to also label it's connected neighbors. The dilation is to be repeatedly performed until all neighboring pixels are labeled. It works exactly like BFS. By using this algorithm, every object is labeled and the number of objects is 8.

(c) Skeletonizing

I implemented the 2-stage morphological hit-or-miss processing. I had to lookup the complete LUT from the textbook as the table provided on lecture slides is not complete. After the skeletonizing procedure converges, there may be some "holes" on the resulting image, because the skeletonizing procedure would remove too much pixels. In order to restore connectivity, an extra "bridge gap" filling pass need to be performed after skeletonization completes.

Problem 2: Texture Analysis

(a) Laws' mask texture segmentation

Input	Segmented texture map
	

I used these five 1d-masks as the basis:

```
  double L5[5] = {1, 4, 6, 4, 1};
  double S5[5] = {1, 0, -2, 0, 1};
  double R5[5] = {1, -4, 6, -4, 1};
  double E5[5] = {-1, -2, 0, 2, 1};
  double W5[5] = {-1, 2, 0, -2, 1};
```

and by convolving these 1d-masks pairwise, I can generate 25 2d-masks.

Then I convolve the input image with these 25 masks and I can obtain a 25-channel feature map.

Then I compute the local energy of each pixel by aggregating neighbor's energy. The window size I choose is 13 pixels wide. I aggregate the energy of each channel by computing both the **root mean square** and **mean** of neighbor. So after energy computation I have a feature map of 50 channels.

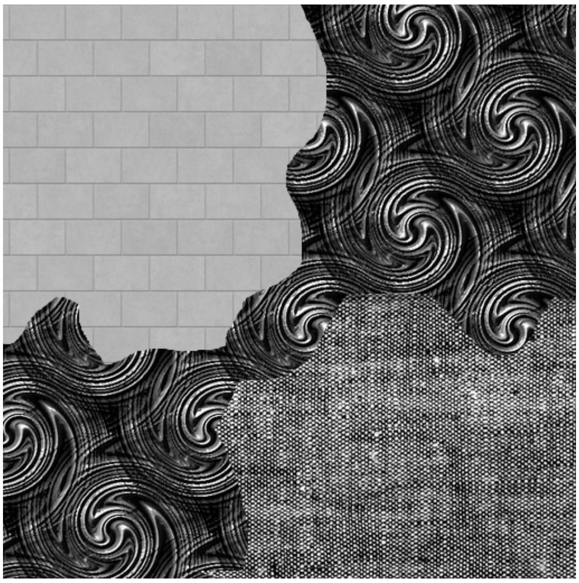
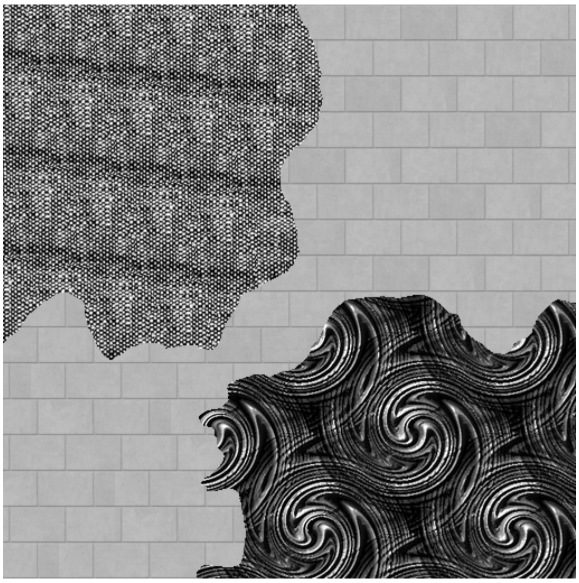
Next I normalize the features, so that each channel's mean equals zero and standard deviation equals one.

Then I perform k-means clustering with the feature vector and clustered the pixels into 3 groups.

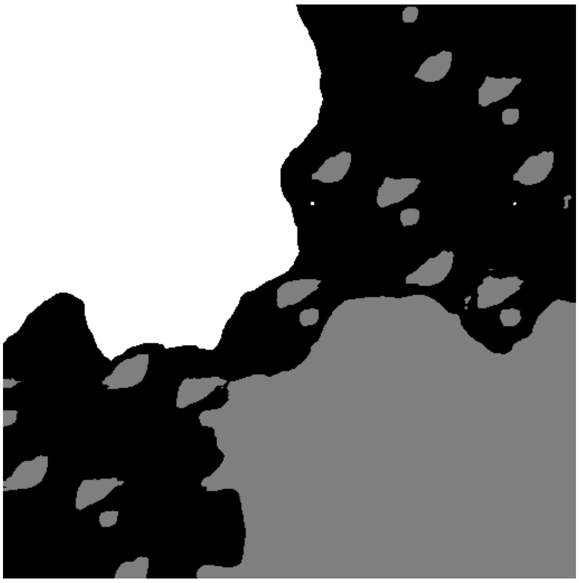
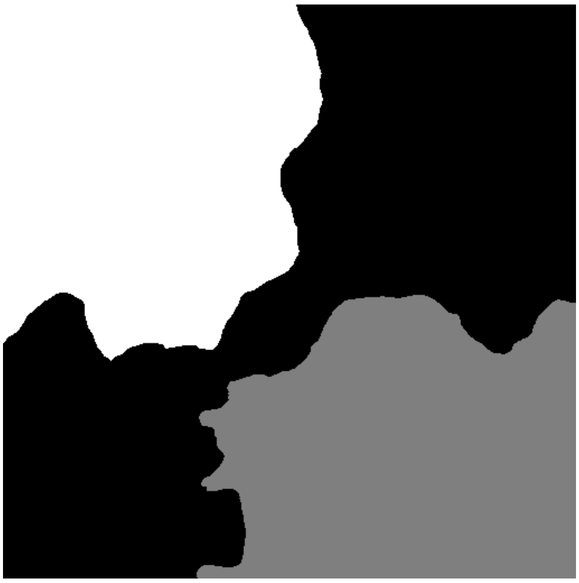
Finally, for each cluster I assigned it a different color and output the texture map.

The result texture map has a lot of "holes" in it. The "swirl" texture isn't clustered well. This is because Laws' masks can only capture low-level features, and high-level features such as "curve" wasn't extracted.
Part of the "swirl" pattern is clustered with "mesh" pattern instead. This is because both the "swirl" and "mesh" pattern have very high spatial frequency, thus they have similar low-level features.

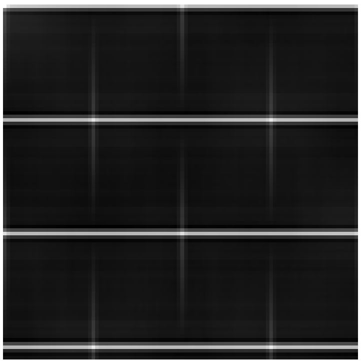
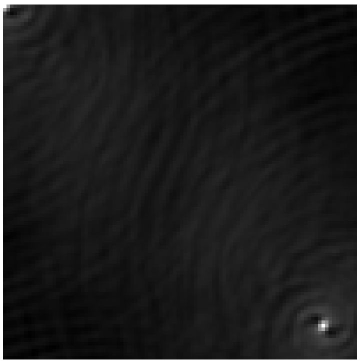
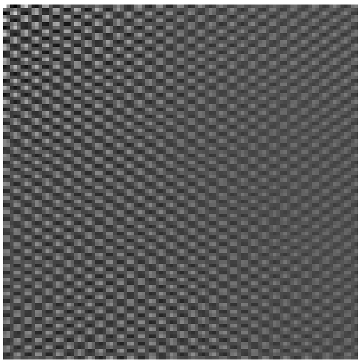
(b) Texture exchange

Input	Result
	

First I tried to fix the mis-classified "holes" in the texture map K. By labeling small holes (with connected component labeling algorithm) and filling them with neighbor's color, I generate another texture map K' that represents the textures better than K.

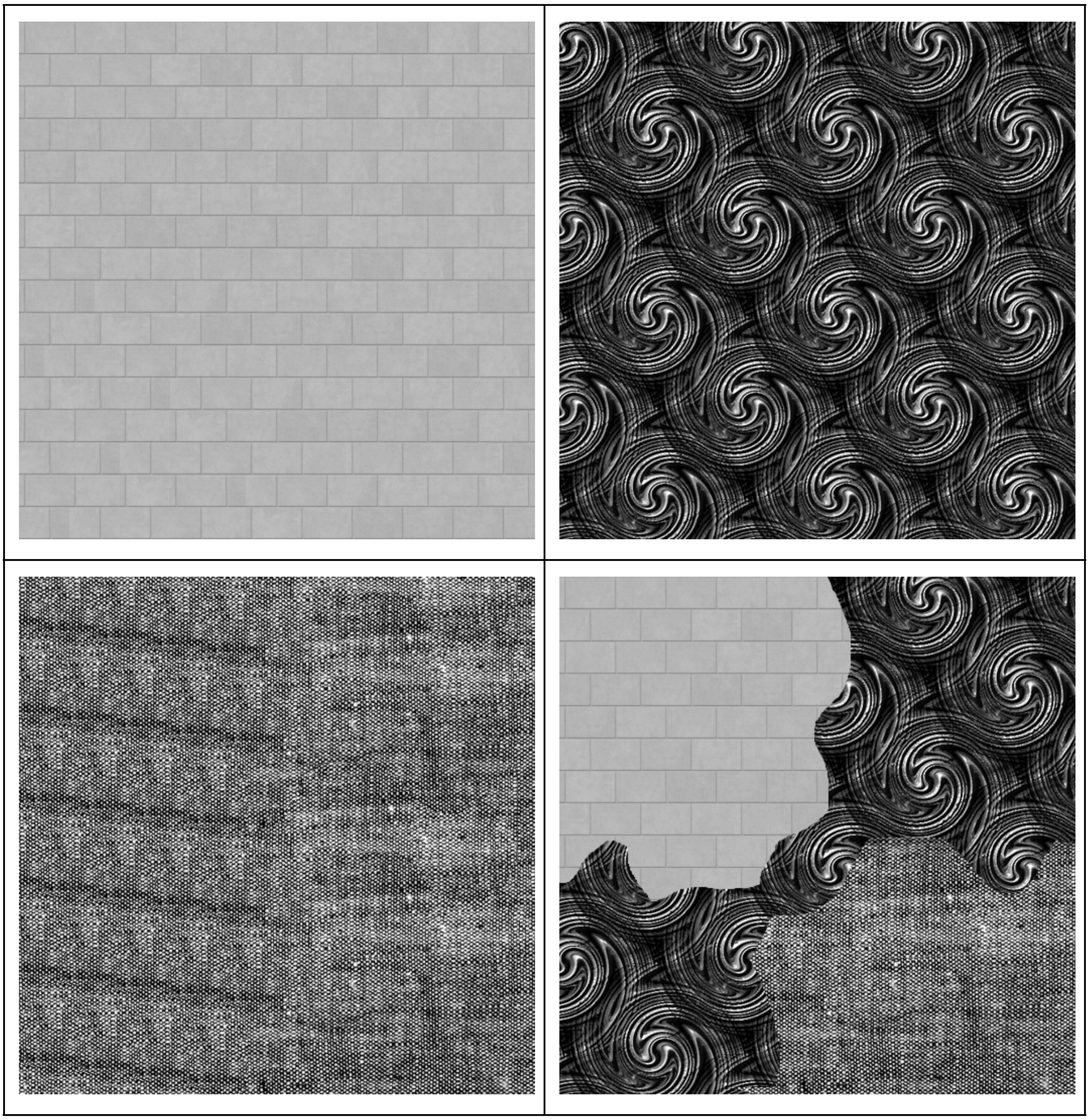
Texture map (K)	K'
	

Then I calculate the autocorrelation of each texture patch. Autocorrelation generates a "peak" and "valley" map, and the "peaks" indicate a high self-correlation. So by finding the local maximals of the autocorrelation I can determine the repeating cycle of a texture patch.
The autocorrelation of the three texture patches:

Top left (Brick)	Middle (Swirl)	Bottom right (Mesh)
		

As one can tell from the figure that the texture's "orientation" and "cycle length" can be easily determined by finding the local maxima!

Then by repeating the texture patch along the determined orientation and cycle, I can stitch together and synthesize a whole texture block.
Following is the three texture blocks that I synthesized, the bottom right corner is the input image for reference.



Note that the mesh texture has artifacts. This is due to the mesh texture being the most noisy, and the repeat-and-stitch process also repeats the noise periodically. Thus creating the artifact of false periodic pattern.

Finally, I use the texture map K' to reconstruct the image with each of the textures exchanged.