Digital Image Processing (2021 spring)

HOMEWORK ASSIGNMENT #3 Morphological Processing, Texture Analysis

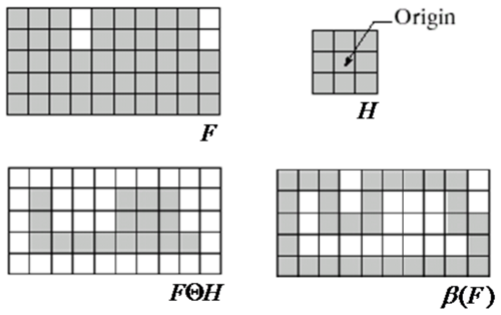
environment: C++, opencv4

Law Po Ying b06902091

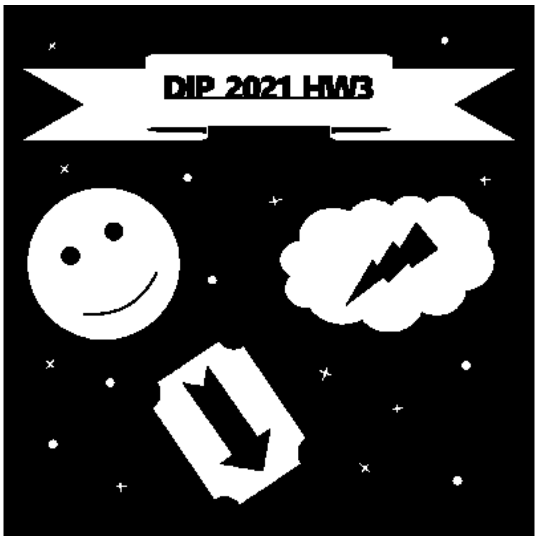
Problem 1: EDGE DETECTION

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

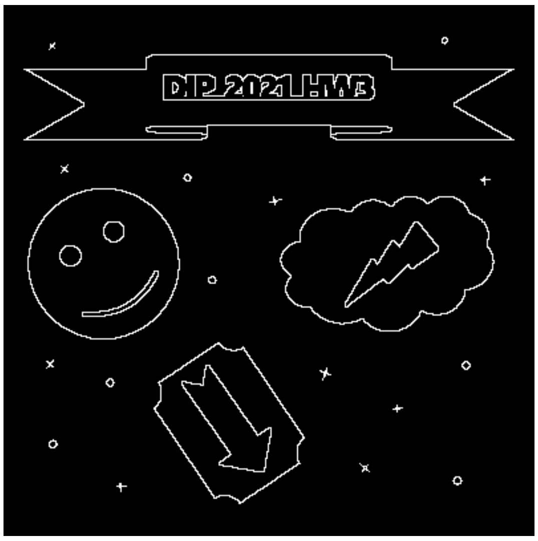
$$\beta(F(j,k)) = F(j,k) - (F(j,k)\Theta H(j,k))$$



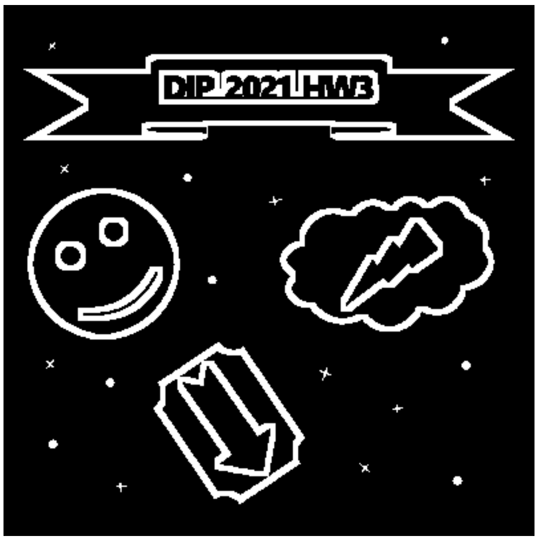
Using a 3x3 kernel to extract the boundaries by erosion. If we use a larger size kernel, the width of the boundaries of the resultant image will be thicker.



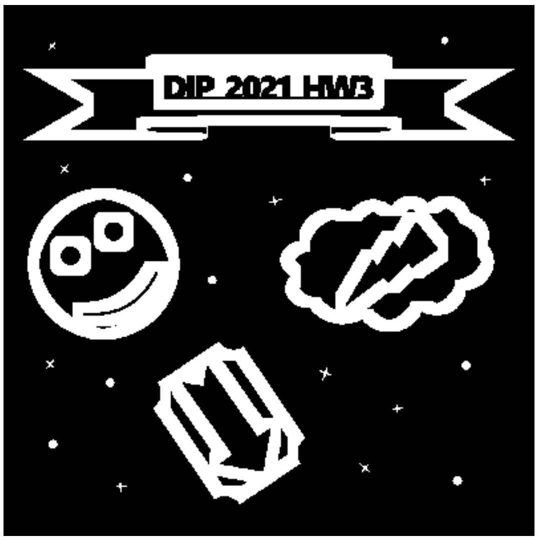
sample1.png



result1.png (3x3 kernel)



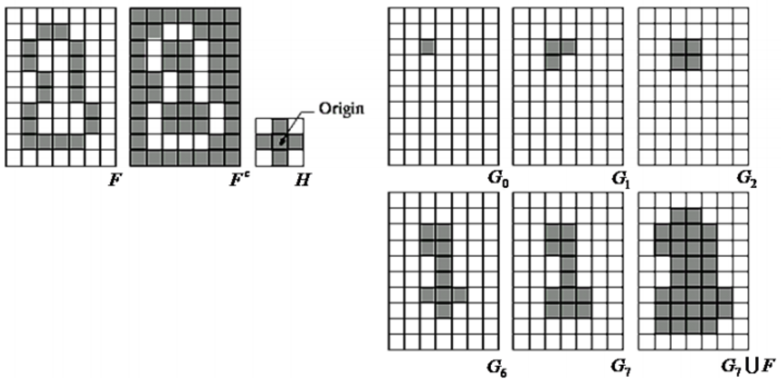
result1.png (9x9 kernel)



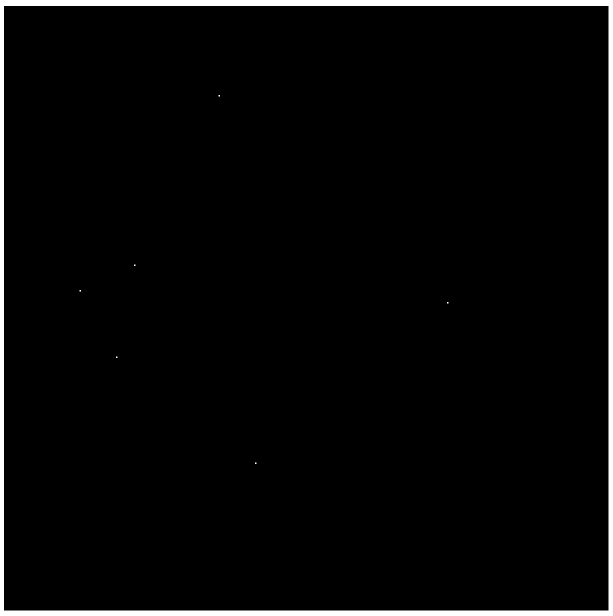
result1.png (17x17 kernel)

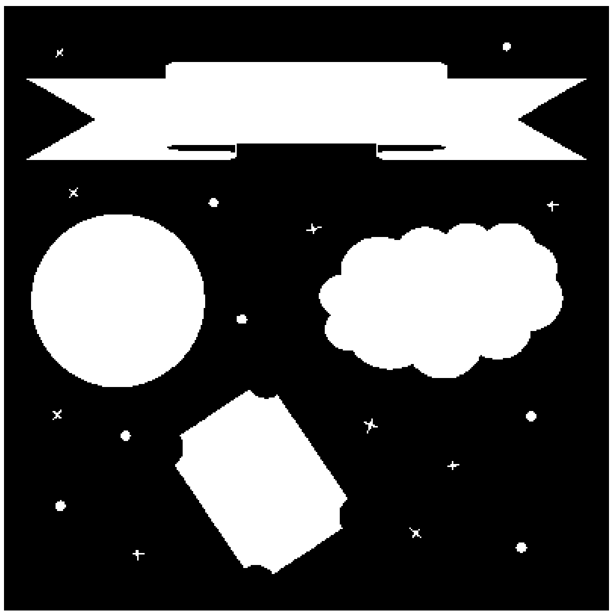
(b) Perform hole filling on sample1.png and output the result as result2.png.

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



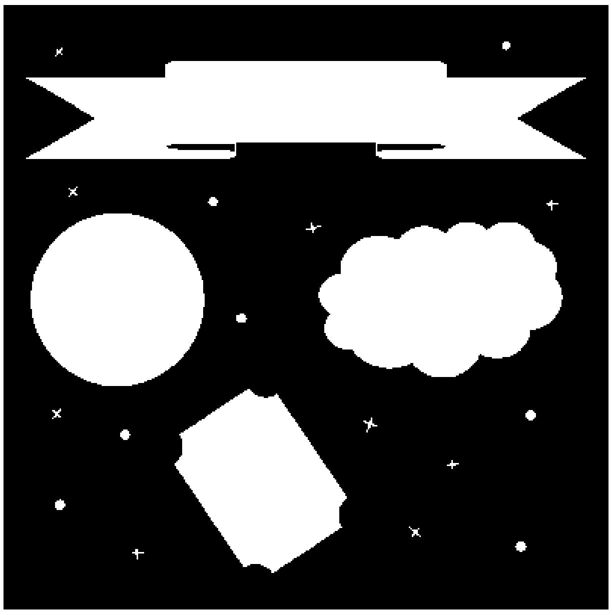
1. Mark points to the regions that need to be filled.
2. Starting from these points and implement dilation until it is totally filled. If we touch the boundaries of the object, skip it and implement dilation on the next pixel.

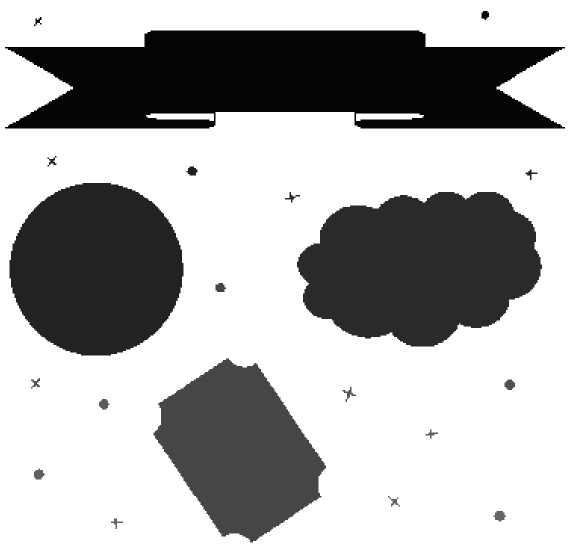

Mark point

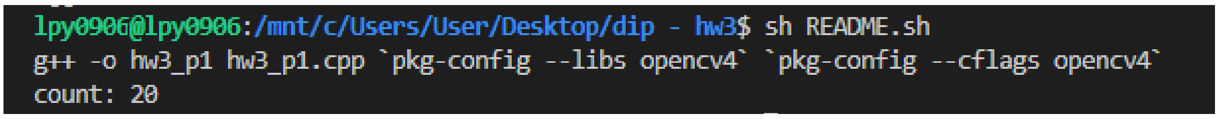

result2.png (3x3 kernel)

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

1. Hole filling
2. Connected component labeling
   a. labels the pixel with a new tag if all of its neighbors have not labeled it.
   b. If its neighbor pixel has been labeled, use the label of neighbor pixel to be the label of the current pixel.
   c. If the current pixel is connected by different labels, record them in the equivalence table and use the smaller label.
   d. Labeling repeatedly.
   e. Apply the equivalence table to the image.
3. Count the final number of labels used.


Hole filling


objects with different grayscales (colors)


```
lpy0906@lpy0906:/mnt/c/Users/User/Desktop/dip - hw3$ sh README.sh
g++ -o hw3_p1 hw3_p1.cpp `pkg-config --libs opencv4` `pkg-config --cflags opencv4`
count: 20
```

Problem 2: TEXTURE ANALYSIS

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

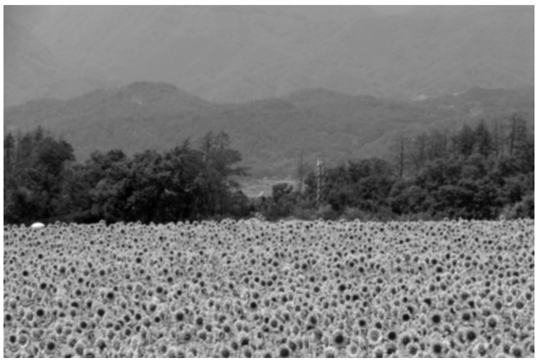
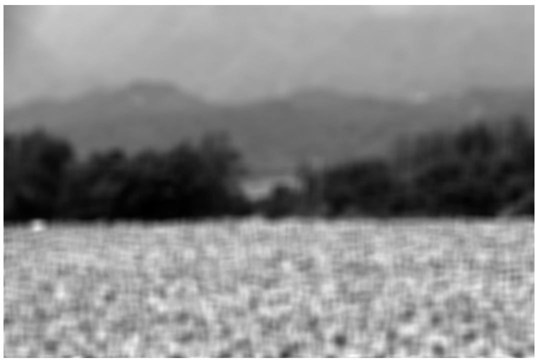
Convolution

$$M_i(j,k) = F(j,k) \otimes H_i(j,k)$$

Energy computation

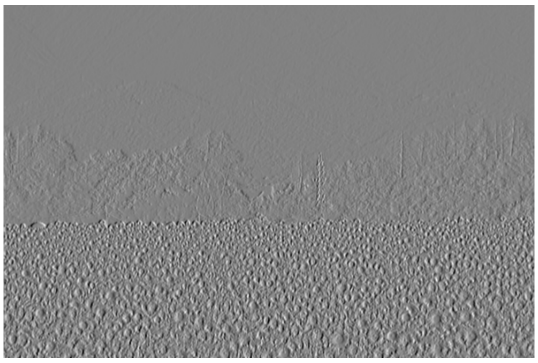
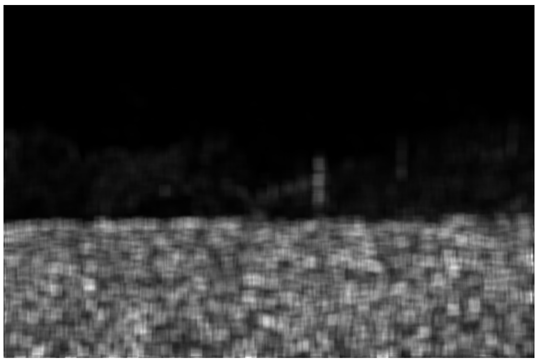
$$T_i(j,k) = \sum\sum \left| M_i(j+m,k+n) \right|^2$$

(window size = 31x31)

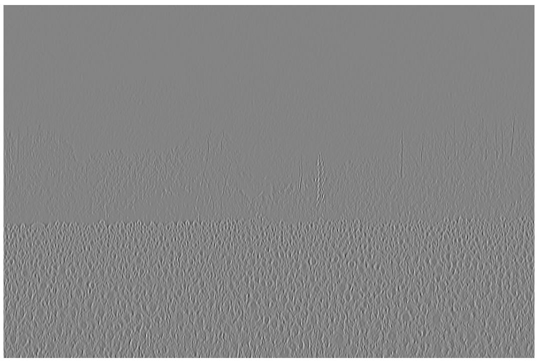
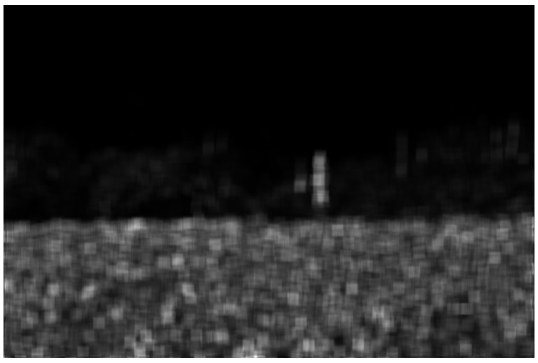
$$\frac{1}{36}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Laws 1

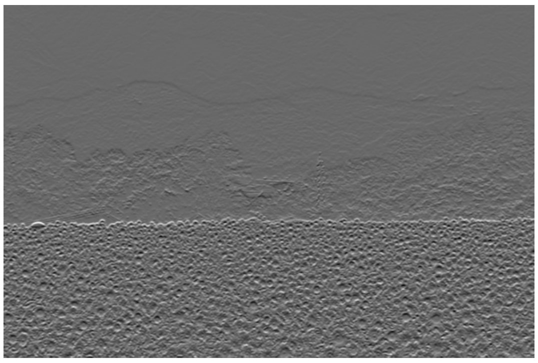
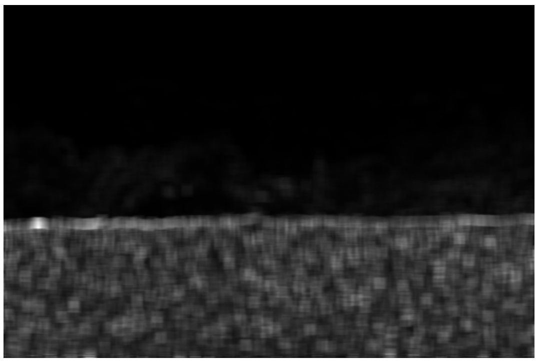
$$\frac{1}{12}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Laws 2

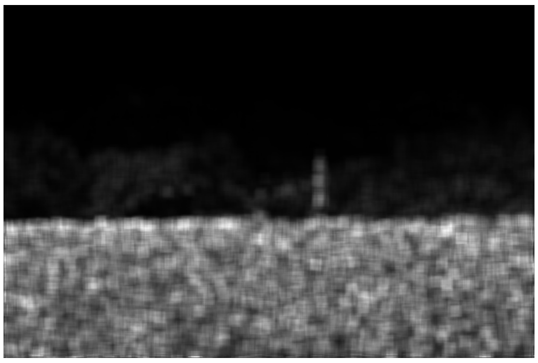
$$\frac{1}{12}\begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

Laws 3

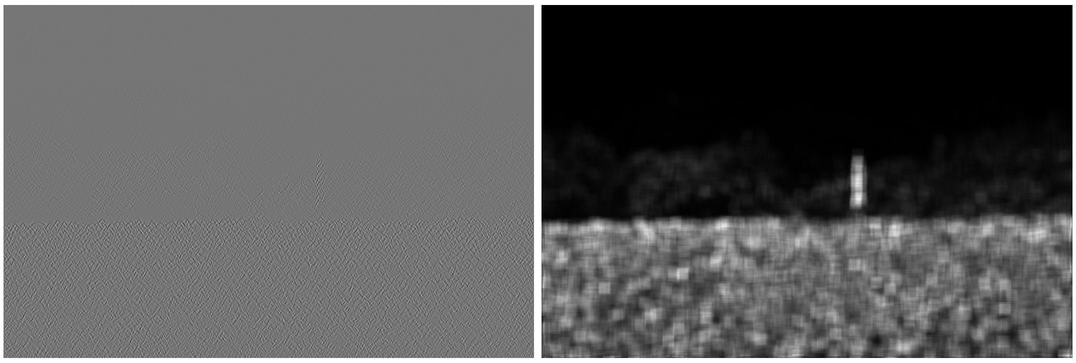
$$\frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Laws 4



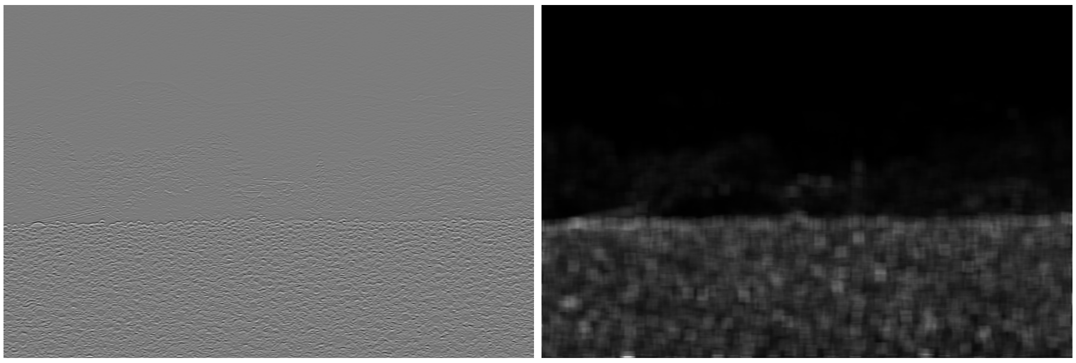
$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Laws 5



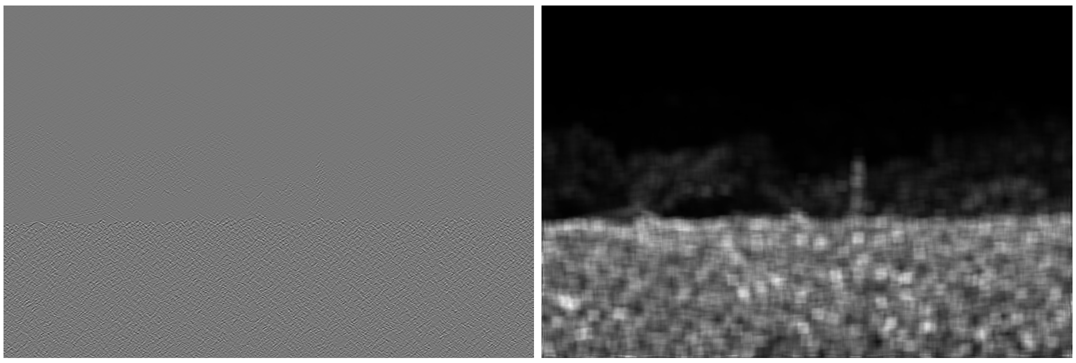
$$\frac{1}{4}\begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

Laws 6



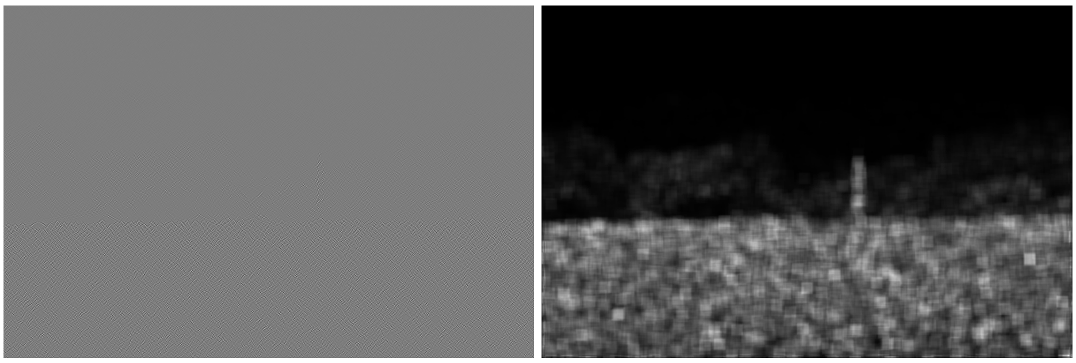
$$\frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix}$$

Laws 7



$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix}$$

Laws 8



$$\frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

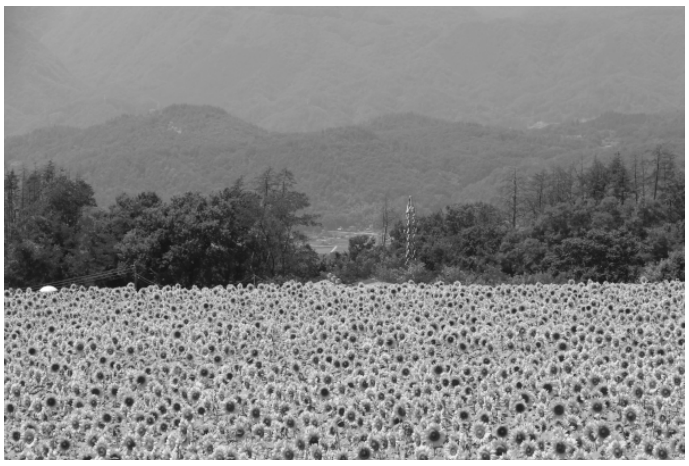
Laws 9

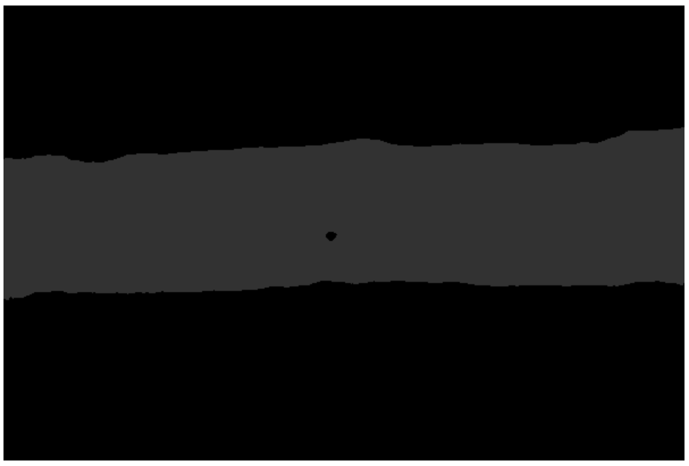
Different laws' masks give different feature representation.

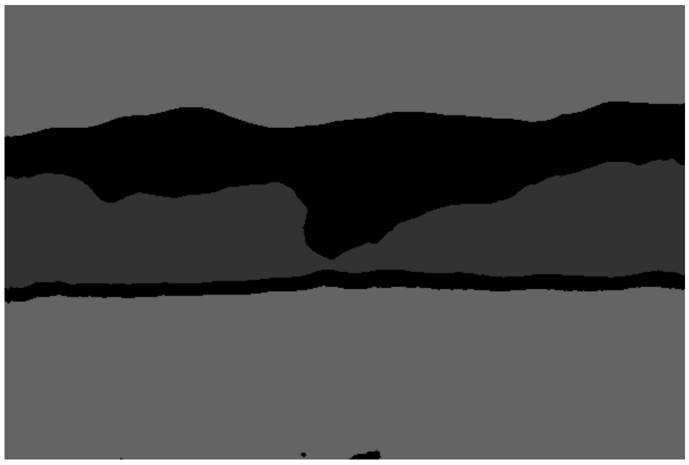
- Laws 1,3,7,9: gives a center-weighted local average
- Laws 2,4,6,8: responds to row or col step edges
- Laws 5: detects spots

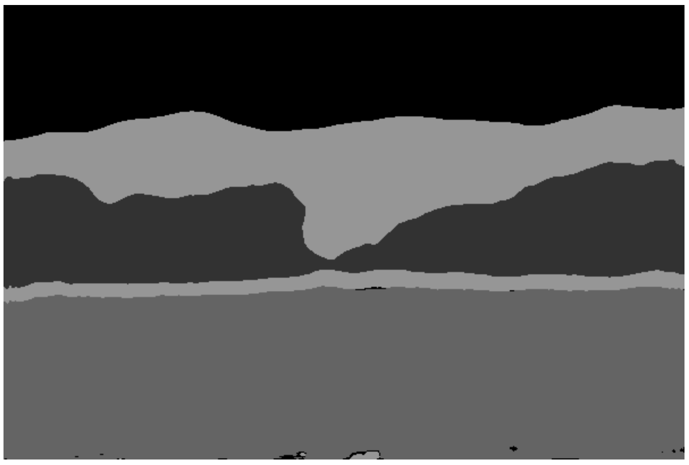
(b) Use k-means algorithm to classify each pixel with the feature vectors you obtained from (a). Label the same kind of texture with the same color and output it as result3.png.

DIfferent value of k

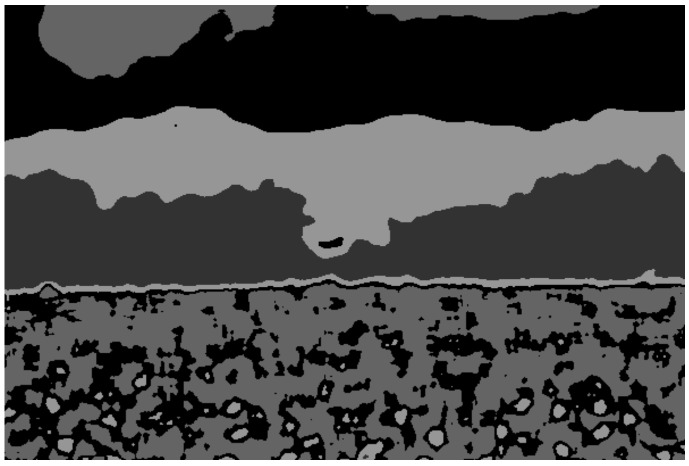

Sample2.png

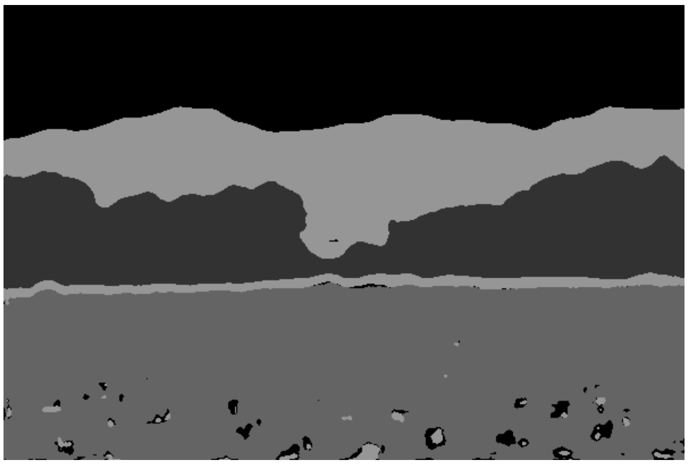

k = 2

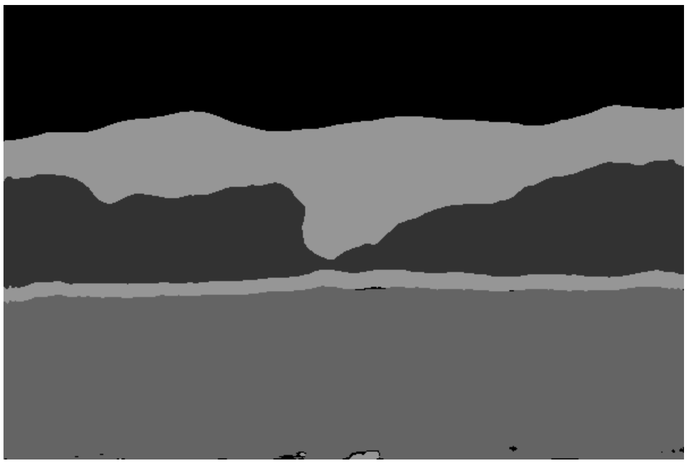

k = 3

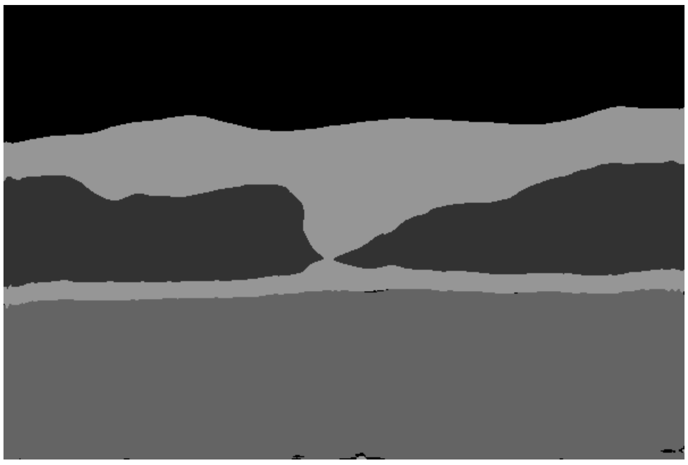

k = 4

DIfferent value of window sizes of energy computation


k = 4, window size = 13x13


k = 4, window size = 19x19

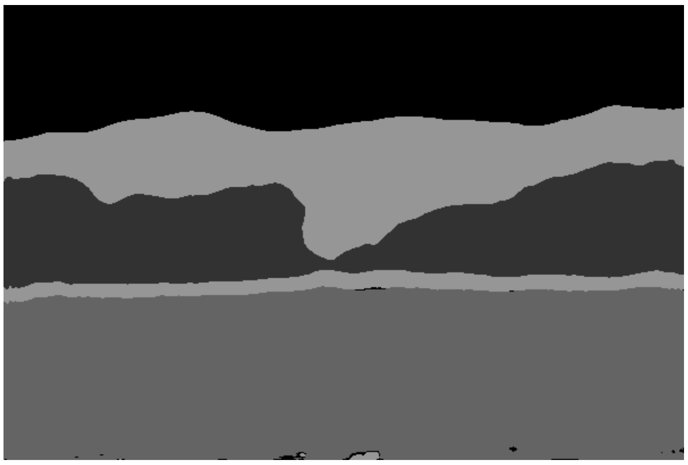

k = 4, window size = 31x31

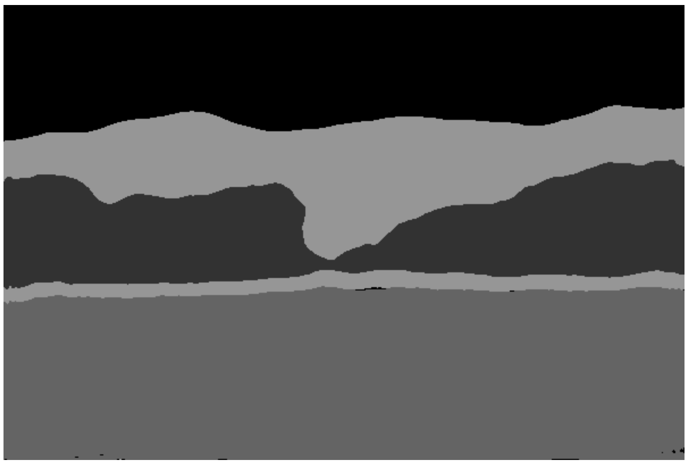

k = 4, window size = 41x41

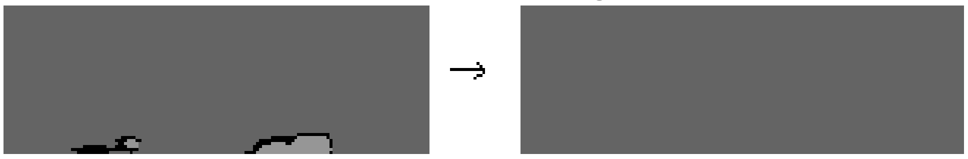
In the K-means algorithm, we can observe that there are many holes if the window size of energy computation is too small. If we use a larger window, the region of texture segmentation may be disconnected. We found that the result with window size 31x31 is the most suitable one. It is completed and has fewer holes.

(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.

There are three key points to improve the classification result. The first one is to use a suitable k-mean to better classify the structure. Then use a suitable (larger) window size of energy computation. Finally, perform hole filling on the resultant image. I have used flood filling to improve the classification result.


Before hole filling

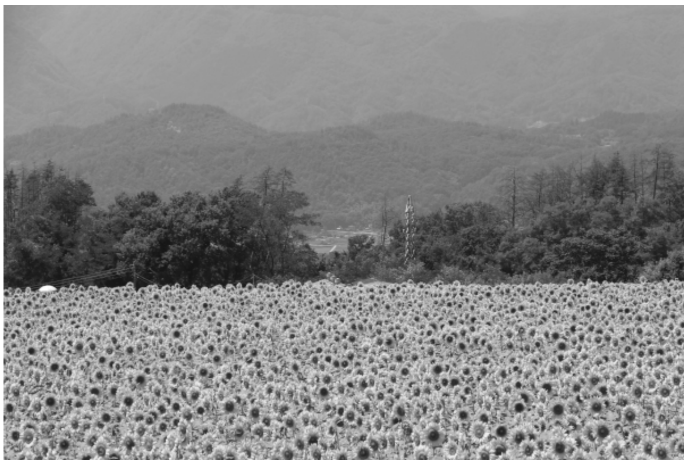
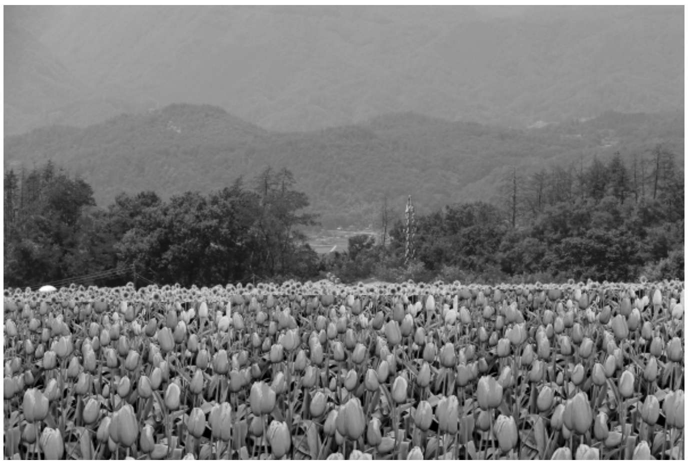
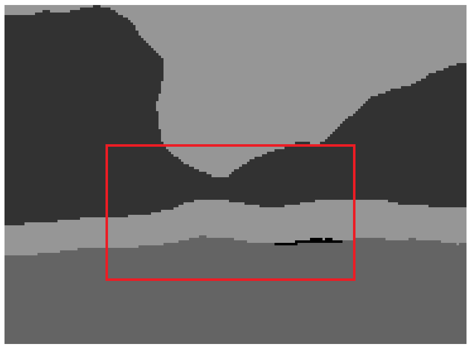
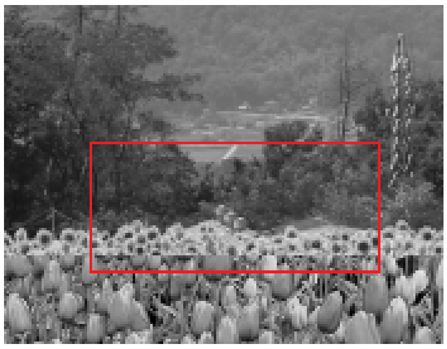

Afterhole filling



(d) (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. It's allowed to utilize external libraries to help you accomplish it, but you should specify the implementation detail and functions you used in the report.

Implementation detail
Scan the resultant image after applying k-mean classification. If the current pixel belongs to the region of the flowers and it has never been changed before, replace the region with sample3.png. Then record the pixels to avoid unnecessary replacement.

The result is well but there is a flaw that there is a gap between the trees and flowers.

Reference / supplement:

1. https://www.youtube.com/watch?v=f5jQ0d06_kQ&ab_channel=SankalpMohanty
2. https://jason-chen-1992.weebly.com/home/-region-filling
3. https://www.tutorialspoint.com/cplusplus-program-to-find-the-connected-components-of-an-undirected-graph
4. https://blog.csdn.net/chengyq116/article/details/104562114
5. https://zh.wikipedia.org/wiki/Flood_fill
6. https://www.researchgate.net/figure/Five-1-D-Laws-vectors-By-convoluting-any-vertical-one-dimensional-vector-with-a_fig1_224317670
7. https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect12.pdf