HOMEWORK #4

University of Southern California

Lecturer: C.-C.JayKuo | Issued: 3/4/2019 | Due:3/19/2019

Report

1 Problem 1: Texture Analysis

1.1 Texture Classification

1.1.1 Abstract and Motivation

Texture is a description of the spatial arrangement of color or intensities in an image or a selected region of an image. Laws developed a series of filters that could automatically detect the texture patterns. Laws' masks are well established as one of the best methods for texture analysis in image processing and are used in various applications, including medical image analysis. In this section, I use the five 1D laws filters to extract the features of 12 textures images and use kmeans method to classification.

1.1.2 Approach and Procedures

1. Preprocessing
   First of all, extend the image boundary with 2 pixels width for each edge by reflection. Then, normalize the input image pixel value by,

$$I(i,j) = \frac{I(i,j)}{255} \tag{1}$$

   Use the result image as the input for the following steps.

2. Set up the Laws filter
   The 2D Laws' filter is caculated by using 1D filter. The five 1D filters are as follows.

$$\begin{aligned} L5 &= [1,4,6,4,1] \\ E5 &= [-1,-2,0,2,1] \\ S5 &= [-1,0,2,0,-1] \\ W5 &= [-1,2,0,-2,1] \\ R5 &= [1,-4,6,-4,1] \end{aligned} \tag{2}$$

   This filter detect level, edge, spot, wave and ripple of the input image. The 2D Laws' filter is the outer product of the 1D laws filters. Thus, we can finally get 25 2D Laws' filters. For one Laws filter, it detect the different property in different direction. Take the E5S5 as an example, the E5S5 is

$$E5S5 = \begin{bmatrix} 1 & 0 & -2 & 0 & 1 \\ 2 & 0 & -4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 4 & 0 & -2 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix} \tag{3}$$

In the horizontal direction, it detects the edge characteristic of the image, while in the vertical direction, it detects the spot characteristic.

3. Convolution and Caculate the Energy
   Use the 2D filter obtained above, convolute the input image and get a filtered image. Caculate the square sum as the representation of the energy using the following equation,

$$Square\_sum = \sqrt{\frac{1}{MN}\sum_{i=1}^{i=M}\sum_{j=1}^{j=N} I(i,j)^2} \tag{4}$$

   Where, M is the height and N is the width of the image. The $Square\_sum$ is represent the texture property obtained by each filter. After filtered with 25 2-D laws filters, we can finally get a 1 by 25 dimension vector for each images. Since we have 12 texture images, we finally get a 12×25 two-dimension array.

4. Principal Component Analysis
   Suppose the input matix is a $N \times D$ matrix, where each row represent a training point while each column represent the feature of the data.

   First, normalize the feature of each data into zero mean and unit variance, i.e. subtract the mean from each column and divided its variance.

$$X(:,i) = \frac{X(:,i) - \mu}{var} \tag{5}$$

   Second, caculate the covariance matrix $\boldsymbol{C}$.

$$cov(\boldsymbol{C}) = \boldsymbol{X}^T\boldsymbol{X} \tag{6}$$

   Third, do the SVD decomposition of $\boldsymbol{C}$.

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{i=D} u_i\sigma_i v_i \tag{7}$$

   Where $\sigma_1, \sigma_2 ... \sigma_D$ is the sigular value sorted in decreasing order. Pick the first three largest sigular value and the corresponding left and right singular vectors to form the new low dimension dataset.

   Last, recover the low-dimension data by multiply the first three columns of the left-sigular-vector $\boldsymbol{U}(:, 1:3)$.

$$\hat{\boldsymbol{X}} = \boldsymbol{X} \times \boldsymbol{U}(:, 1:3) \tag{8}$$

5. K-means Clustering
   For a dataset $\boldsymbol{x}_1, ..., \boldsymbol{x}_N \in \mathbb{R}^D$, the K-means distortion objective function is,

$$J(\{\mu_k\}, \{r_{ik}\}) = \frac{1}{N}\sum_{i=i}^{N}\sum_{k=1}^{K} r_{ik} \parallel \mu_k - \boldsymbol{x}_i \parallel_2^2 \tag{9}$$

Where $\mu_1, ..., \mu_N$ are the centroids of K clusters and $r_{ik} \in \{0, 1\}$ represent whether example i belongs to cluster k. Our goal is to minimize J by update the centroid and clusters over iteration.

First, fixing the centroid and cluster the dataset. The cluster clasification is measured by Euclidean distance.

$$\hat{r_{ik}} = \begin{cases} 1 & k = argmin_{k'} \|\mu_{k'} - \boldsymbol{x}_i\|_2^2 \\ 0 & else \end{cases} \tag{10}$$

Second, fixing the cluster assignment and find the new centroid by minimize J.

$$\hat{\mu_k} = \frac{\sum_{i=1}^{N} r_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{N} r_{ik}} \tag{11}$$

By repeating this two steps, k-means finally cluster the input data into k cluster until the maximum iteration is reached.

1.1.3 Experimental Result

The feature obtained by laws filters are shown in Table 1.

Table 1: Laws' filter Results

	L5L5	L5E5	L5S5	L5W5	L5R5	E5L5	E5E5	E5S5	E5W5	E5R5	S5L5	S5E5	S5S5	S5W5	S5R5	W5L5	W5E5	W5S5	W5W5	W5R5	R5L5	R5E5	R5S5	R5W5	R5R5
Texture 1	23.234247	4.6825110	2.1976640	1.8063850	2.8931140	4.6563900	1.14569	0.615261	0.543338	0.911480	2.2360650	0.629166	0.366846	0.340821	0.596002	1.8570570	0.571519	0.353790	0.344430	0.624239	3.0070380	0.992611	0.650907	0.669362	1.2684020
Texture 2	15.519040	3.3037110	2.0008330	1.9259510	3.3339640	3.0939180	1.0554210	0.705943	0.718552	1.3054350	1.9633920	0.664501	0.458605	0.483221	0.948022	1.8790030	0.650536	0.465297	0.504723	1.0278850	4.22519	1.2137740	0.912609	1.0329280	2.1175560
Texture 3	16.621194	4.4736080	3.31508	3.6534520	7.0009330	1.6925420	0.360978	0.227988	0.234451	0.474318	1.1261190	0.206598	0.133033	0.137970	0.279465	1.0865340	0.203801	0.133234	0.140679	0.289675	3.1793560	0.408892	0.270087	0.291381	0.608785
Texture 4	23.468594	6.7505890	3.9780370	3.6511910	6.1144190	4.7540730	1.4080920	0.869330	0.840918	1.52497	2.6181890	0.774082	0.483874	0.472601	0.875657	2.4161390	0.724981	0.458069	0.452810	0.844226	4.9179660	1.3020250	0.836251	0.843531	1.6102970
Texture 5	42.057818	9.6580320	4.8892210	3.9949350	6.3130140	9.2229180	2.4742920	1.3526740	1.1989280	2.0053660	4.7171120	1.3507530	0.791898	0.737567	1.2900120	4.00038	1.2359560	0.764760	0.742231	1.3488630	7.1464640	2.2008030	1.4254330	1.4459380	2.7398020
Texture 6	37.257672	10.672662	6.2258550	5.6340540	9.3717850	7.3090420	2.2058610	1.3637470	1.3177120	2.4033310	3.8956840	1.2125590	0.763727	0.750567	1.4043120	3.5644950	1.1379410	0.725143	0.722346	1.3721710	6.62047	2.0481660	1.3286520	1.3556710	2.6589960
Texture 7	22.204380	5.1025030	2.5982240	2.1287460	3.3601380	4.8471780	1.3032860	0.717721	0.637370	1.0668270	2.5244110	0.712157	0.419269	0.390548	0.683647	2.1683510	0.653331	0.405365	0.393021	0.713837	4.1362860	1.1646540	0.756777	0.767413	1.44918
Texture 8	29.347879	6.2740220	3.7880610	3.6377180	6.3145750	5.7985960	2.0565240	1.3701370	1.3871990	2.5091330	3.4533610	1.3089360	0.905245	0.949446	1.8488910	3.2782250	1.2890810	0.929327	1.00544	2.0269070	6.2573210	2.3843810	1.8114760	2.0633110	4.2256870
Texture 9	43.238528	9.1141560	6.7507420	7.3943790	13.912112	3.7394160	0.900478	0.563597	0.582703	1.1947090	1.9297210	0.524479	0.339574	0.355980	0.739276	1.9029960	0.536469	0.353914	0.375567	0.789401	4.2254170	1.0991890	0.736089	0.796160	1.6858160
Texture 10	23.015021	4.8983070	3.2974780	3.42569	6.6216140	2.3702640	0.976126	0.759602	0.889031	1.9975260	1.0646410	0.428021	0.343537	0.423066	1.0391950	0.890206	0.327091	0.256623	0.319995	0.834625	1.4483590	0.505643	0.387378	0.4711	1.2521780
Texture 11	16.264758	4.8344790	3.1763660	3.0221690	4.5084640	1.3376630	0.332087	0.217935	0.234377	0.495369	0.779485	0.188805	0.125442	0.137253	0.300436	0.780822	0.187067	0.126896	0.1405	0.310427	1.5282950	0.379609	0.260953	0.293954	0.662665
Texture 12	26.865751	6.6678070	3.6361530	3.2245420	5.4255880	4.5457440	1.38110	0.773449	0.740840	1.3593010	2.3570780	0.697736	0.423219	0.415515	0.790853	2.1023390	0.640086	0.395435	0.395456	0.766898	3.55898	1.1281490	0.721796	0.744824	1.4777550

The plot of PCA is shown in Figure 1.
The kmeans result are as follows.
Groud truth: Bark: 4, 6, 12. Straw: 2, 8, 10. Brick: 3, 9, 11. Bubble: 1, 5, 7.
Use 25 dimension feature space to do the kmeans: 1, 2, 3, 1, 4, 4, 1, 1, 4, 1, 3, 1.
Use 3 dimension feature space to do the kmeans: 2, 2, 3, 1, 4, 4, 2, 4, 1, 2, 3, 1.

1.1.4 Discussion

Compute the variance of each feature over 12 images, we can conclude that L5L5 maintains the largest discriminant power, that could distingush each images. That is because the L5L5 filter collects the low-dimension information of the input image, which is dominant in the image. Beside, the Numerical value is the highest among the features. Thus, a small fluctuation of L5L5 will result in a large variance in the feature space. On the other hand, S5W5 and S5S5 maintains the smallest discriminant power, because they extract the high frequency information which is of small quantity in the images.

Table 2: Variance of the 25-D features

	L5L5	L5E5	L5S5	L5W5	L5R5	E5L5	E5E5	E5S5	E5W5	E5R5	S5L5	S5E5	S5S5	S5W5	S5R5	W5L5	W5E5	W5S5	W5W5	W5R5	R5L5	R5E5	R5S5	R5W5	R5R5
var	92.6800	5.3703	2.1778	2.4890	9.1832	5.1849	0.4499	0.1569	0.1422	0.4579	1.3863	0.1519	0.0599	0.0583	0.2046	1.0529	0.1383	0.0609	0.0637	0.2342	3.3524	0.4508	0.2239	0.2642	1.0159

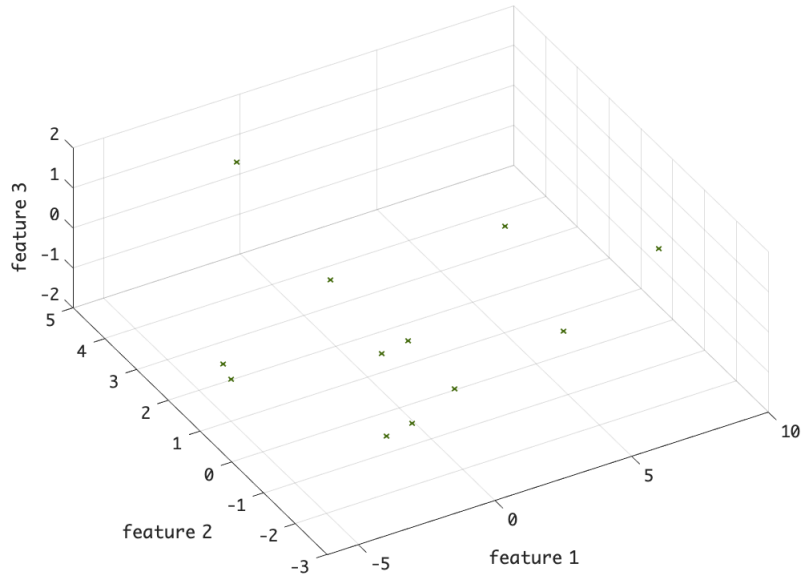


Figure 1: PCA Result

The kmeans method do not work very well in texture classification. With 25D feature space, it only cluster "brick" successfully and some "bubbles". It makes some mistakes that it cluster "straws" into "bubbles". That might because the feature representation of this two textures are not strong enough. With 3D feature, the result improves a little. Althought it still cannot seperate the "straws" and "bubbles", but it cluster the "Bark" and "brick" correctly. That might because the "brick" has strong vertical edge and horizontal edge information that distinguish among the textures pattern while the "Bark" also maintains strong vertical edge information.

1.2 Texture Segmentation

1.2.1 Abstract and Motivation

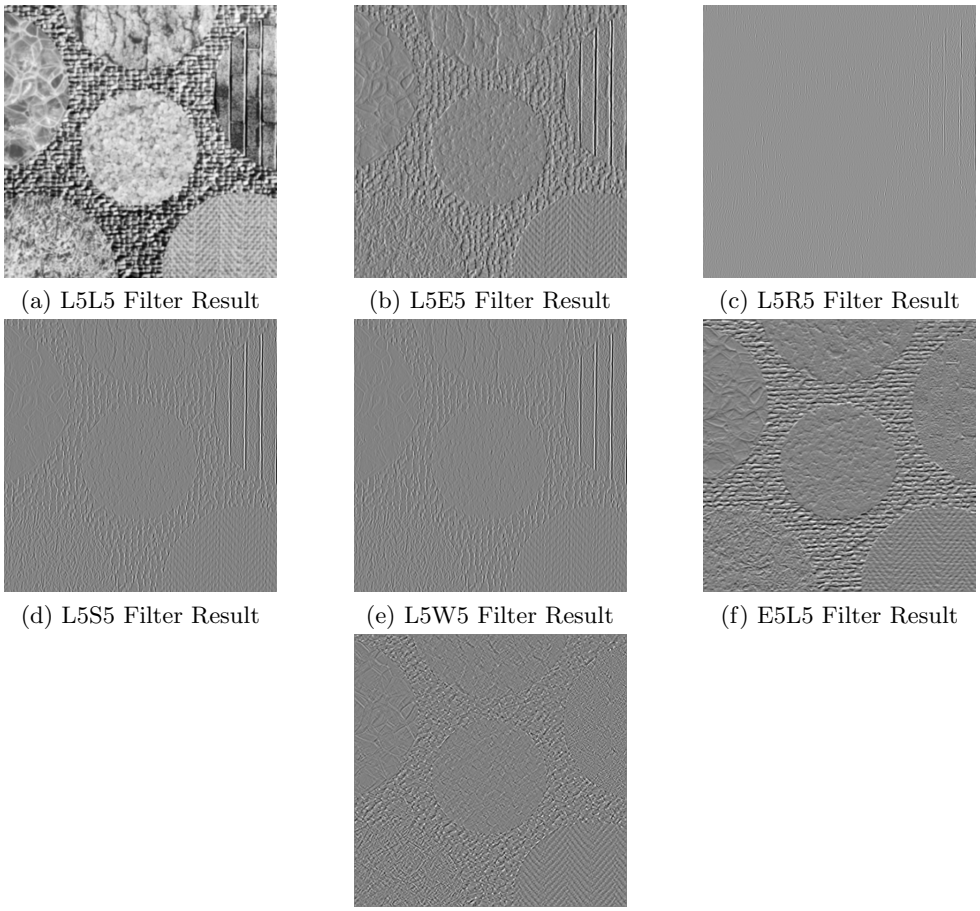
Base on the previous section, we use kmeans method to classify different feature base on the feature property. In this section, we use the same method – Apply 25 Laws' filter to segment the image contains various feature.

1.2.2 Approach and Procedures

1. Subtract the Local Mean
   First, subtract the local mean from each pixel within a 5 by 5 window to eliminate the brightness in the same texture.

$$I(\hat{i,j}) = I(i,j) - \frac{1}{N \times N} \sum \sum I(k,l) \tag{12}$$

2. Feature extraction
   As it is discribe in 1.1.2, apply 25 2-D Laws filters to extract the 25 feature of the entire image pixels. After this step, I got a $Row \times Column \times 25$ three-dimension array.



(a) L5L5 Filter Result (b) L5E5 Filter Result (c) L5R5 Filter Result

(d) L5S5 Filter Result (e) L5W5 Filter Result (f) E5L5 Filter Result

(g) E5E5 Filter Result

Figure 2: Some Filter Result

3. Compute Energy Map
   Use the Window to produce 25-d energy map.

$$Square\_sum = \sqrt{\frac{1}{M^2}\sum_{i=1}^{i=M}\sum_{j=1}^{j=M} I(i,j)^2} \tag{13}$$

   Where, M is the window size. Here, I choose $M = 7, 15, 17$. Use this value to represent the energy of the pixel located at the center of the window.

4. K-means cluster
   Apply k-means cluster with 25-D feature space. Set the number of cluster equals to 7. Output the result of the segmented regions by using 7 different grayscale intensity.

1.2.3 Experimental Result

The segmentation result is shown in Figure 3.

Figure 3: Segmentation Result, Window size = 15

1.2.4 Discussion

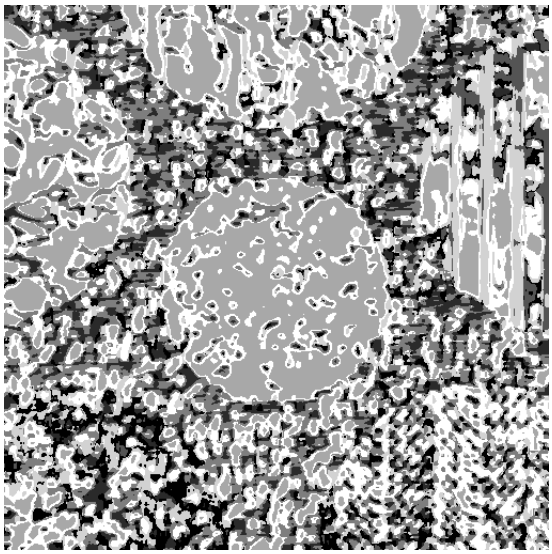


Figure 4: Segmentation Result ,Window size = 7

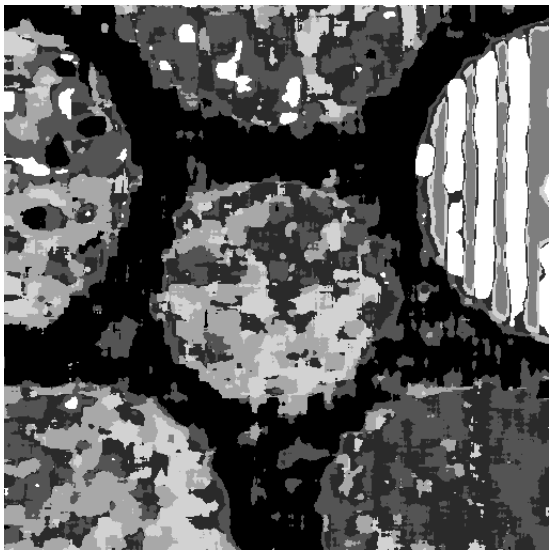


Figure 5: Segmentation Result ,Window size = 17

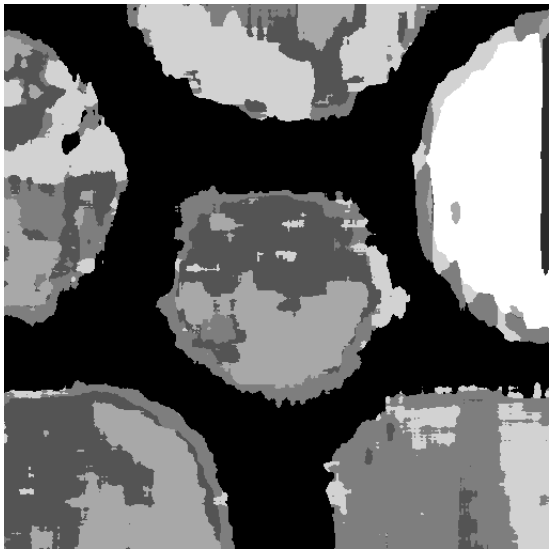


Figure 6: Segmentation Result ,Window size = 41

1.2.5 Discussion

Figure 4 to 6 shows the segmentation result is not quite good. From the figure, we can observe that the larger window size is, the less isolate holes remain in a paticular regions. When window size = 41, it shows the best result. Because it separates the most areas. There stll remains some holes in the image and the texture are not correctly classified. That might because the 25-D feature are not strong enought to represent the pixel information. Thus, PCA will not help much in this case.

1.3 Advanced Texture Segmentation Techniques

1.3.1 Abstract and Motivation

The result of 1(b) is not so clean and in this section I will come up with several idea to enhance the results.

1.3.2 Approach and Procedures

1. Adopt PCA
   Adapt a PCA method, reduce the 25-D feature space to low dimension feature space. Use the low dimension feature space to do the k-means and output the segmentation result. I try to reduce the dimension to 15, 7 and 3 and observe the result.

2. Post-processing
   Adapt median filter to merge the small hole and make the segmentation result more cleaning. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighbouthood entries. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire signal. For 2D images, the window size normally has an odd number of entries, like 3, 5 or 7. Here I choose the window size equal to 5. I filter the image for 30 times and the result are shown in the next section.

3. Enhance the Boundary

   Looking at boundary, the pattern of the boundary pixels are unique. First, itself is not the background color(which in this case is 0), then, it's 4-connected neighbour must contain a background color. Collect the boundary pixels of a texture pattern. Use the majority vote method to decide the boundary pixels' class.

1.3.3 Experimental Result

1.3.4 Discussion

As I mentioned above, the PCA method does not help much in improving the segmentation result. However, when taking a close look at the detail, **PCA does merge some small holes in to a continuous area**. But it still enlarege the boundary aroud the texture pattern. Although the segmentation result is not so good, but the median filter actually work well in merge the small hole in a specific area. From the Figure 10, we can see that the boundary is more clearly. And the small hole in the texture patter are merged. The k-means method works well in classify the texture located at the right-down area and the middle area.

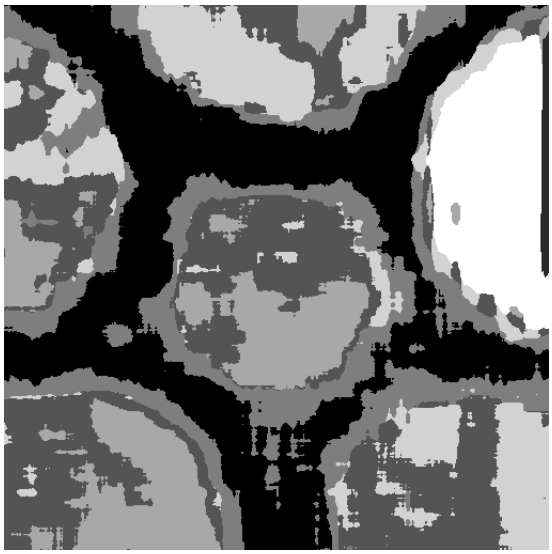


Figure 7: PCA to 3-D Result, Window size = 41

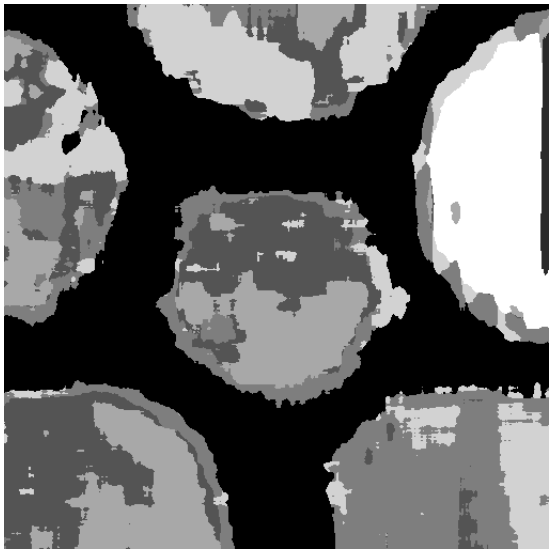


Figure 8: PCA to 7-D Result, Window size = 41

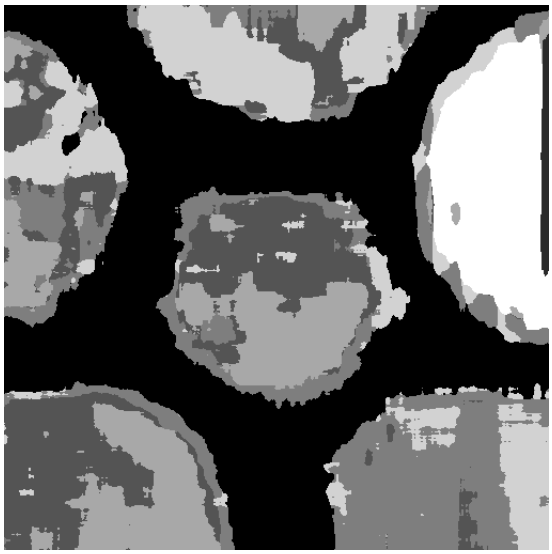


Figure 9: PCA to 15-D Result, Window size = 41

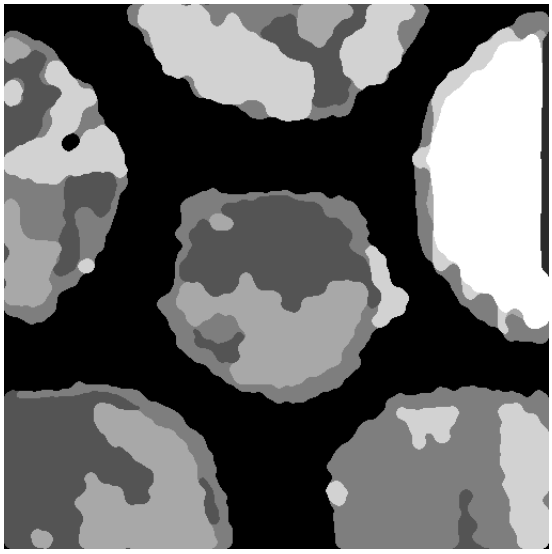


Figure 10: Median Filter Result, Window size = 41

2 Problem 2: Image Feature Extractor

2.1 SIFT

2.1.1 Abstract and Motivation

Scale-invariant feature transform (SIFT) is a computer vision algorithm used to detect and describe local features in image. It looks for the points that invariant in **translation, scaling and rotation** in spatial scale. This algorithm was published by David Lowe in 1999, and was summarized in 2004. In this section, I will introduce the basic procedure of the SIFT algorithm.

2.1.2 Approach and Procedures

The foundamental of the SIFT is as follow. The scaling-invariance and position-invariance is achieved by selecting key location at the maxima and minima of the DoG applied in scale space. This key location are treated as the potential key points. First, the input image is continuously filtered and downsampled by stage Gaussian kernel functions with different σ and building an image pyramid. After that, we need to remove the points that are not rotation-invariant. It mainly includes the following key steps.

1. Feature extraction
   In order to make the features have **position and scale invariance**, the detection of feature points is done in image scale space. The input image is convolved with Gaussian function through several level.

$$\begin{aligned} G(\sigma_1) &= (2\pi\sigma_1^2)^{-1} e^{-(x^2+y^2)/2\sigma_1^2} \\ G(\sigma_2) &= (2\pi\sigma_2^2)^{-1} e^{-(x^2+y^2)/2\sigma_2^2} \\ G(\sigma_3) &= (2\pi\sigma_3^2)^{-1} e^{-(x^2+y^2)/2\sigma_3^2} \\ G(\sigma_4) &= (2\pi\sigma_4^2)^{-1} e^{-(x^2+y^2)/2\sigma_4^2} \\ G(\sigma_5) &= (2\pi\sigma_5^2)^{-1} e^{-(x^2+y^2)/2\sigma_5^2} \end{aligned} \tag{14}$$

   Where $\sigma_1 = \sqrt{2}, \sigma_2 = 2, \sigma_3 = 2\sqrt{2}, \sigma_4 = 4, \sigma_5 = 4\sqrt{2}$.

   Sample the image with different size. For each specific size, the image are filtered by Gaussian filters with different σ. In order to detect stable and unique key points **more efficiently**, then subtract the adjacent gaussian smooth output image to approximate the Laplacian of Gaussian (LoG) with in the same octave. **The computation of DoG cost less than the Computation of LoG.** Then, for each pixels, we choose the 26 neighbourhood pixels to compare with. If it is the minima or maxima of the neighbourhood, we regard it as a potential key point.

2. Feature description
   After get these potential key points, the gradient direction distribution characteristic of the key point's neighborhood pixels is used to assign a direction to each image descriptors, so that the key point has **rotation invariance**.

$$M_{ij} = \sqrt{(A_{i,j} - A_{i+1,j})^2 + (A_{i,j} - A_{i,j+1})^2} \tag{15}$$

$$R_{i.j} = \tan^{-1}\left(A_{i,j} + A_{i+1,j}, A_{i,j+1} + A_{i,j}\right) \tag{16}$$

The pixel differences are efficient to compute and provide sufficient accuracy due to the substantial level of previous smoothing.

3. Removing
   By removing the potential points that not prossess the rotation-invariance, the key point achieves its robustness of rotation-invariant.

   (a) Detect "Low contrast point"
   Use Taylor expension method to compute the contrast. Compute the Hessian matrix $\boldsymbol{H}$ and its trace and determination.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{D}_{xx} & \boldsymbol{D}_{xy} \\ \boldsymbol{D}_{xy} & \boldsymbol{D}_{yy} \end{bmatrix} \tag{17}$$

$$tr(\boldsymbol{H}) = 2\boldsymbol{D}_{xy} \tag{18}$$

$$det(\boldsymbol{H}) = \boldsymbol{D}_{xy}^2 - \boldsymbol{D}_{xx}\boldsymbol{D}_{yy} \tag{19}$$

   if

$$\frac{tr(\boldsymbol{H})^2}{det(\boldsymbol{H})} > \frac{(r+1)^2}{r} \tag{20}$$

   remove it.

   (b) Remove points in non-salient region
   In order to make it as stable as possible, the orientation is determined by the peak in the histogram of local imgae gradient orientation. The orientation histogram has 36 bins covering 360 degree range of orientation, ans select the peak value to represent the orientation of the key point.

4. Illuminance Change
   SIFT algorithm guarantee its robustness to illuminance change by thresholding the gradient magnitudes with maximum possible gradient value times 0.1.

5. Output descriptor
   The orientation planes is formed by collecting the pixels that fall in a circle of radius 8 pixels around a key location. The orientation is defined by subtracting the key-points's orientation. For each key-points, use 8 orientation planes to sample over a 4 by 4 grid of locations, with a sample spacing 4 times that of the pixel spacing used for gradient detection. In order to sample the image at larger scale, the same process is repeated for a second level. But in this time, we use a 2 by 2 region. Thus, we got $8 \times 4 \times 4 + 8 \times 2 \times 2 = 160$ elements. SIFT's output vector size in its original paper contains 160 elements.

2.2 Image Matching

2.2.1 Abstract and Motivation

Image retrival is an important task in object recogonization area. In this section, I will implement the SIFT algorithm on OpenCV toolbox to extract the feature vector and do the Nearest Neightbour Search to do the image retrieval.

2.2.2 Approach and Procedures

1. Find the Key-point & Extract SIFT Feature
   Input the image *river1.raw* and *river2.raw* and extract the SIFT feature of the two images. The number of key points found by OpenCV_SIFT is shown in Figure 11. The feature extract by OpenCV_SIFT contains 128 dimensions.

```
number of key point extract from river1.raw:  8618
number of key point extract from river2.raw:  9082
```

Figure 11: number of key point found by OpenCV_SIFT

2. Find the Largest Scale Key-point
   The "largest scale key point" is defined by the key point's corresponding descriptor with the largest $L2$ norm. Thus, by traversing the entire SIFT descriptors of the *river1.raw* and caculating the $L2$ norm, I found the largest scale key point $\boldsymbol{kp}_1$. The parameters of $\boldsymbol{kp}_1$ is shown in Figure 12. The position of $\boldsymbol{kp}_1$ is marked in Figure 13.

```
The index of the largest scale key point in river1.raw:  (60.05622100830078, 476.3304443359375)
The orientation of the largest scale key point in river1.raw:  206.17921447753906
The diameter of of the largest scale key point neighbourhood in river1.raw:  6.630932807922363
```

Figure 12: The parameters of $\boldsymbol{kp}_1$

Figure 13: The position of $\boldsymbol{kp}_1$

3. Find the Closest Neighboring Key-point Use K-NN method and compute the $L2$ norm of the distance between $\boldsymbol{kp}_1$ and the key-points found in *river2.raw*. The $L2$ norm is defined by

$$\left\| \boldsymbol{kp}_1 - \boldsymbol{kp}^{'} \right\|_2 = \sqrt{(x_1 - x_1^{'})^2 + (x_2 - x_2^{'})^2 + ... + (x_D - x_D^{'})^2} \tag{21}$$

Where $\boldsymbol{kp}^{'}$ is the key-point of *river2.raw*. The $x_1, x_2 ... x_D$ are $D = 128$ features of the descriptor, i.e. the 128 entries in a row of the descriptor.

The key-point with the shortest $L2$ norm distance is regardest as the closest neighboring key-point of $\boldsymbol{kp}_1$. I try several number of nearest neighbour and the result are shown in the next section.

2.2.3 Experimental Result

The matching result of *river1.raw* and *river2.raw* is shown in Figuer 14, where I choose 500 key-points in each image. The Nearest Neighbour in *river1.raw* and it's orientation is circled in Figure 15. The parameter of the key-point is shown in Figure 16

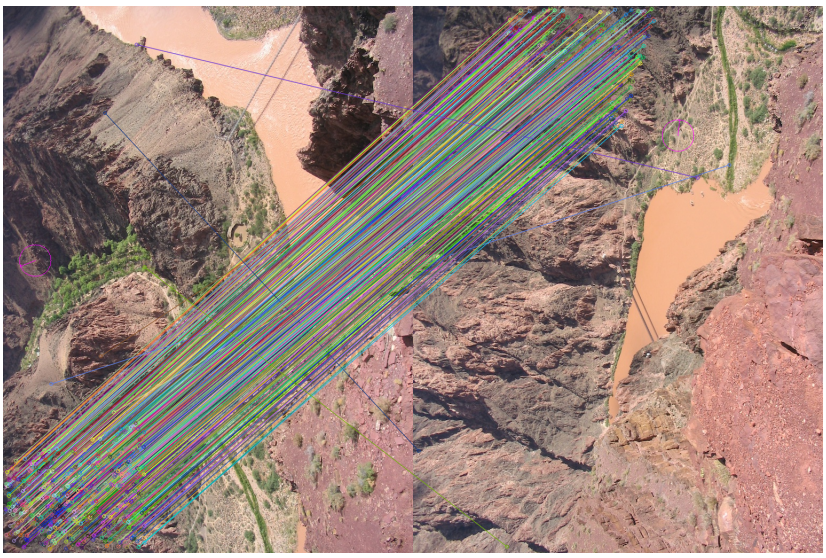


Figure 14: 500 key points Matching Result

The 2-Nearest Neighbour in *river1.raw* and it's orientation is circled in Figure 17. The parameter of the key-point is shown in Figure 18.

The 3-Nearest Neighbour in *river1.raw* and it's orientation is circled in Figure 19. The parameter of the key-point is shown in Figure 20

2.2.4 Discussion

The dominant orientation of the nearest point in *river2.raw* is 129.87, which is different from the $\boldsymbol{kp}^{'}$. Though it is true that in descriptor feature the dominant orientation is provided, this is not used to actually compare the similarity of the keypoints. The keypoints are primarily compared using the distance between the discriptor, which regardless of this dominant orientation are inherently orientation invariant. The dominant orientation are used in some retrieval/classification tasks as extra information used to improve matching scores.

Figure 15: 1-NN result

```
nearest point in river2:  (28.139734268188477, 466.1160583496094)
the orintation of the nearest point in river2:  129.87998962402344
The diameter of of the largest scale key point neighbourhood in river1.raw:  1.9627506732940674
```

Figure 16: Parameter of the 1-NN Result

Figure 17: 2-NN result

```
the orintation of the nearest point in river2:  129.87998962402344
The diameter of of the largest scale key point neighbourhood in river1.raw:  1.9627506732940674
nearest point in river2:  (496.109375, 242.12303161621094)
the orintation of the nearest point in river2:  65.8524169921875
The diameter of of the largest scale key point neighbourhood in river1.raw:  9.828125953674316
```

Figure 18: Parameter of the 2-NN Result

Figure 19: 3-NN result

```
the orintation of the nearest point in river2:  129.87998962402344
The diameter of of the largest scale key point neighbourhood in river1.raw:  1.9627506732940674
nearest point in river2:  (496.109375, 242.12303161621094)
the orintation of the nearest point in river2:  65.8524169921875
The diameter of of the largest scale key point neighbourhood in river1.raw:  9.828125953674316
nearest point in river2:  (411.07147216796875, 162.6121826171875)
the orintation of the nearest point in river2:  315.71728515625
The diameter of of the largest scale key point neighbourhood in river1.raw:  1.9489965438842773
```

Figure 20: Parameter of the 3-NN Result

2.3 Bag of Words

2.3.1 Abstract and Motivation

A Bag-of-Words model that encodes in graphs the local structures of a digital object. In this section, I use the samples of zeros and ones from MNIST to form a codebook with bin size of 2. Use the SIFT algorithm to extract the feature from the image and do k-means cluster. Then, do the classification for the *eight.raw* and show the histogram.

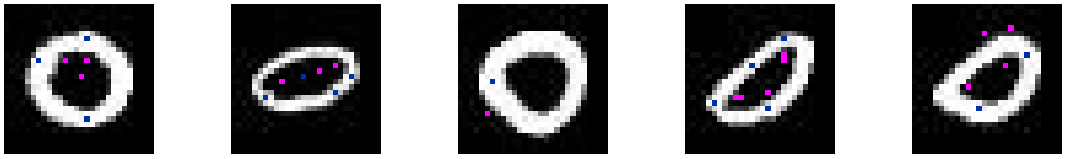
2.3.2 Approach and Procedures

1. Extract the SIFT feature
   Extrac the SIFT feature for each image *one_1.raw* ... *zero_5.raw*. Collect the descriptor for all the image and form a training data set. The OpenCV_SIFT find 54 key point in total. Thus, the training dataset is a 54×128 2-D array.

```
number of key point in 0:  8
number of key point in 0:  2
number of key point in 0:  12
number of key point in 0:  12
number of key point in 0:  7
number of key point in 1:  2
number of key point in 1:  3
number of key point in 1:  3
number of key point in 1:  0
number of key point in 1:  5
```

Figure 21: number of key-points

2. K-menas Cluster
   Train the training set with k-means cluster method and divide the training point into 2 clusters. The zeros' key points are shown in Figure 22 while the ones' key-points are shown in Figure 23. The blue dot represent the cluster of object, while the magenta dot reprent the cluster of background.

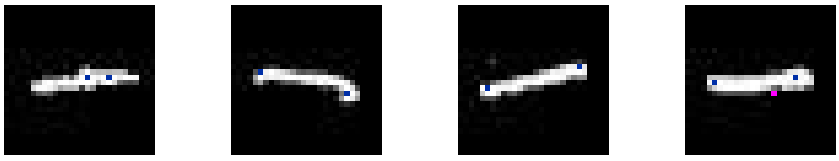


(a) zero_1 (b) zero_2 (c) zero_3 (d) zero_4 (e) zero_5

Figure 22: Zeros key-points

   Finally we got 2 centroid and 2 cluster for futher prediction.

3. Get the Histogram
   With the trained codebook, I can use it to get the histogram of the *eight.raw*. Extract the key-points of *eight.raw* and use the discriptor as the prediction dataset. Use the trained k-means cluster above to predict the cluster of the key-points. That is, caculated the $L2$ norm



(a) one_1 (b) one_2 (c) one_3 (d) one_4

Figure 23: Ones key-points

distance between the key-points and two centroids. Label the key-points with the shortest $L2$ norm distance of the centroid.

4. Caculate the Distance

   Regarding the histogram as a 2-dimension vector, caculate the distance between *eight.raw* and the codebook image. First, normalize the vector so that it has unit length. Then caculate the $L2$ norm of the distance and take an average. Use this distance information to classify the target image.

2.3.3 Experimental Result

OpenCV_SIFT extract 12 key points of the *eight.raw*. The cluster result are shown in Figure 24. The histogram of *eight.raw* is shown in 25. The histogram of Zeros is shown in 27. The histogram of Ones is shown in **??**.

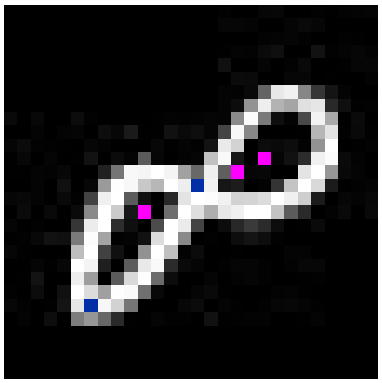


Figure 24: image of eight key-point

2.3.4 Discussion

From the Table 3, we can result that the *eight.raw* is closer to image zeros than ones. If there is only two digits in the bag, then *eight.raw* will be classified as Zeros.

The OpenCV_SIFT detect 12 points in *eight.raw*, however, only 5 of them are marked in the Figure 24. The reason is that some key-points **share the same coordinate but they are different in angle**.

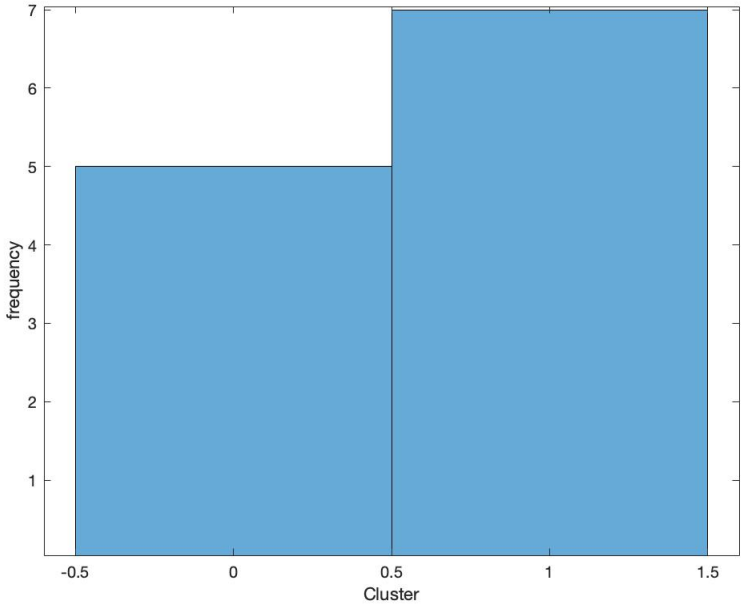


Figure 25: Histogram of *eight.raw*

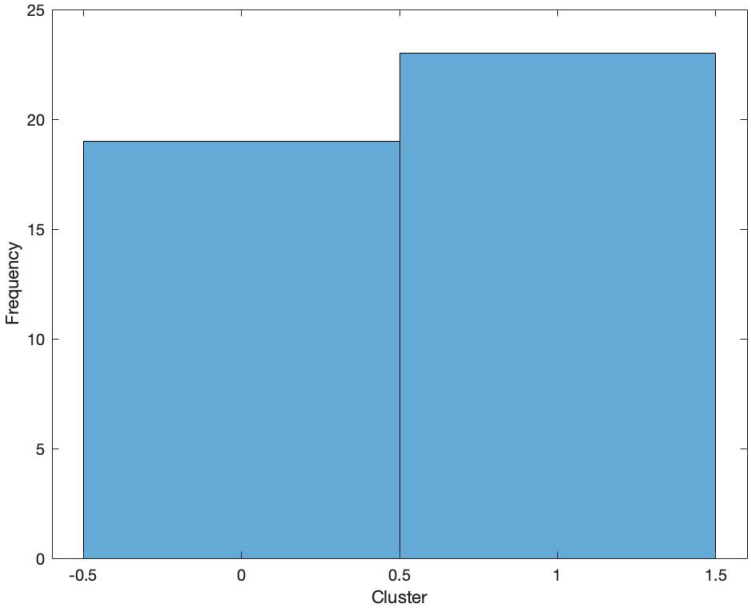


Figure 26: Histogram of Zeros

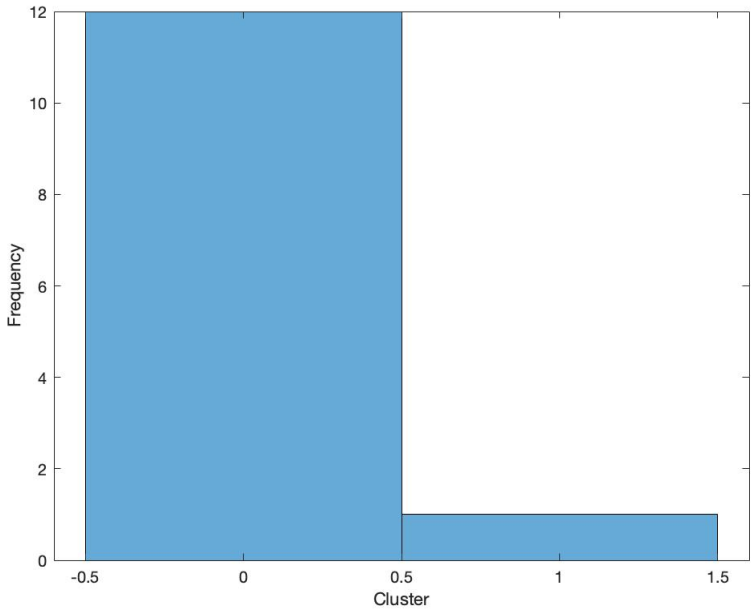


Figure 27: Histogram of Ones

Table 3: Distance Between *eight.raw* and bag of words

codebook	distance
zero_1	0
zero_2	0.4073
zero_3	0.1650
zero_4	0.1650
zero_5	0.2392
one_1	0.9152
one_2	0.9152
one_3	0.9152
one_4	0.6910

```
The index of key-point is:  (6.036602020263672, 22.00472068786621)
The orientation of key-point is:  356.158203125
The index of key-point is:  (10.599776268005371, 15.82100772857666)
The orientation of key-point is:  62.253753662109375
The index of key-point is:  (10.599776268005371, 15.82100772857666)
The orientation of key-point is:  239.60244750976562
The index of key-point is:  (10.69171142578125, 15.001066207885742)
The orientation of key-point is:  28.941009521484375
The index of key-point is:  (10.69171142578125, 15.001066207885742)
The orientation of key-point is:  220.13241577148438
The index of key-point is:  (14.336299896240234, 13.980986595153809)
The orientation of key-point is:  83.46072387695312
The index of key-point is:  (14.336299896240234, 13.980986595153809)
The orientation of key-point is:  255.73301696777344
The index of key-point is:  (17.977048873901367, 12.411479949951172)
The orientation of key-point is:  87.02120971679688
The index of key-point is:  (17.977048873901367, 12.411479949951172)
The orientation of key-point is:  213.2467041015625
The index of key-point is:  (19.008071899414062, 11.843730926513672)
The orientation of key-point is:  44.950286865234375
The index of key-point is:  (19.008071899414062, 11.843730926513672)
The orientation of key-point is:  80.05438232421875
The index of key-point is:  (19.008071899414062, 11.843730926513672)
The orientation of key-point is:  218.8138427734375
```

Figure 28: Parameters of the key-point in *eight.raw*