

EE 569: Homework #4**Issue: 3/04/2019 Due: 11:59PM 03/19/2019**

Name: Wenjun Li

USC ID: 8372-7611-20

Email: wenjunli@usc.edu

Submission Date: 03/19/2019

Problem 1: Texture Analysis (50%)**(a) Texture Classification (15%)****1. Motivation**

Texture analysis consists of texture feature extraction and texture feature analysis. If we want to analyze the textures inside an image, we need to use some techniques to extract them. Since texture features are statistical data of the gray scale image, we can apply Laws Filters to gray image to extract local features. Texture extraction and texture classification are extremely important in the area of computer vision. For example, texture classification can help us classify a large image database automatically, which can save a lot human resource. Below I will explain Laws Filters in detail.

2. Approach

Laws filter have five 1-D kernels and thus they can form twenty-five 2-D kernels. The five 1-D Laws filter are shown in below table. To generate twenty-five 2-D Laws, we use $A^T A$, where A denotes the 1-D Laws filter kernel. For example, E5L5 can be calculated as:

$$E5L5 = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5(Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

Table 1 Five 1-D Laws Filter Kernels

Before we do texture classification, we need to do some preprocessing to images. In order to reduce illumination effects, we subtract image mean.

Then, the 2nd step of texture classification is to do texture extraction. Using the 25 2-D laws filter, we can obtain 25 filtered image and get a 25-D feature vector for an image.

The 3rd step is called feature averaging. After we obtain the 25-D feature vector, we need to average the feature of all image pixels. Since the filtered image may have negative value, here we take the average of square value for each pixel. The 25-D filtered feature vector has a dimension of 25x128x128. After we take the average value for each dimension, we have a 25x1x1 vector for each image in those 12 texture

images. Then we combine these 12 averaged feature vectors together and we now have a matrix of 25×12 . Here, 25 is the vector dimension, and 12 is the number of images to be classified.

The 4th step is using PCA (principal component analysis) to reduce those 25-D vector to 3-D, i.e. we will have a 3×12 matrix as output. Using scikit-learn package in python, we can easily calculate PCA and obtain the final 3×12 matrix. So, what is PCA? As we know, a high dimensional feature matrix usually has lot of redundancy. As a result, the covariance matrix of this high dimensional feature matrix has many zeros (or close to zero) eigenvalues, which can be discarded safely. PCA can help us remove these redundant eigenvalues and eigenvectors. Also, PCA decompose a matrix by orthogonal transformation and thus it can produce uncorrelated features.

The last step to do texture classification is do clustering to classify the 3×12 matrix. Here, we can treat this 3×12 matrix as 12 matrices and each of them has a dimension of 3. K-means clustering is a type of unsupervised data analysis algorithm. Below are the procedures of k-means clustering.

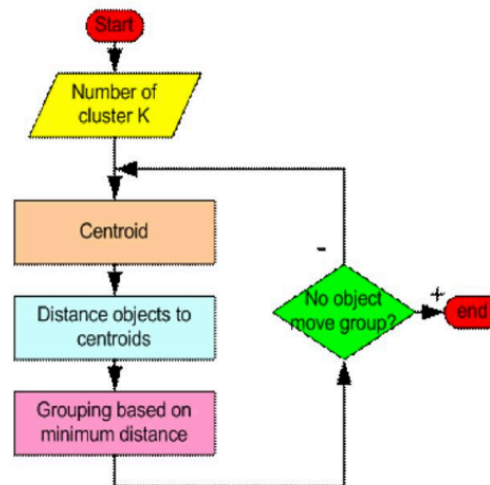


Figure 1 K-Means Clustering Algorithm

First, K-means choose k sample center randomly and compute the distance from each point to these clustering center points. Then, it will classify each point to the center that nearest from it. Then, k-means continue to compute the distances from each point to those clustering center points. Next, k-means will do grouping once again. By doing above steps repetitively, k-means will final classify all the points.

As a classic clustering algorithm, k-means has following advantages: it converges fast; high efficiency; perform well when groups are dense and sparse. At the same time, k-means also has some shortcomings: it need to input group number before-time and it depends on initialization very much.

3. Result

As you can see from below result, both 25-D and 3-D feature vector can produce a good k-means clustering result. They correctly classify 11 texture images and is only wrong with texture 10. This is because texture 10 is too similar to another texture. When I set the cluster number to be 5, texture10 will stay by its own and other texture images will be correctly classified.

```
clustering result of 25-D
[0 2 1 3 0 3 0 2 1 1 1 3]

clustering result of 3-D
[0 2 1 3 0 3 0 2 1 1 1 3]
```

Figure 2 K-means Clustering Results (Clustering Number=4)

The result of cluster number = 5 is shown below. As you may see, texture 10 is classified by its own, i.e. it is close to none other features.

```
clustering result of 25-D
[0 2 1 3 0 3 0 2 1 4 1 3]
clustering result of 3-D
[0 2 1 3 0 3 0 2 1 4 1 3]
```

Figure 3 K-means Clustering Results (Clustering Number=5)

4. Discussion

Q1: Which feature dimension has the strongest discriminant power? Which has the weakest?

A1: The feature dimension gives strongest discriminant is the most important feature in clustering. When we do PCA using scikit in python, we can rank their importance from high to low. In this problem, we reduce feature vector from 25-D to 3-D, then the 3 left feature dimensions are the strongest discriminant.

```
[ [ 4.45694183e-01 -2.34731086e-01 -3.80926662e-03]
  [-1.00202285e+00  6.44597987e+00 -3.47164917e-01]
  [ 1.38205532e+01 -2.48862562e-01 -2.96415371e-01]
  [-1.70939604e+00 -1.07577738e+00 -2.88889131e-01]
  [-1.87354614e+00 -8.02014694e-02 -1.00356605e-01]
  [-7.03162322e-01 -1.11917885e+00 -7.66196961e-01]
  [-2.20303037e+00 -1.00481332e+00 -2.91070511e-01]
  [ 1.15585994e-01 -9.40488162e-02  2.94301531e+00]
  [-1.77600507e+00 -8.68471326e-01 -2.58846232e-01]
  [-1.27314825e+00  3.26987325e-01  8.98548412e-02]
  [-1.63286141e+00 -1.08169297e+00 -5.19336590e-01]
  [-2.20866090e+00 -9.65189415e-01 -1.60784569e-01] ]
```

Figure 4 Strongest Discriminant Feature Dimensions

Q2: Compare 25-D clustering and 3-D clustering?

A2: Basically, the clustering quality of 25-D and 3-D are basically the same. They both can produce a good classification result. The major difference is the computation speed. Since 3-D clustering needs a much less computation than 25-D, so, when we applied PCA to do dimension reduction, we will have a fast computation speed and convergence speed.

Discussion: Something about K-Means

As a classic clustering algorithm, k-means has following advantages: it converges fast; high efficiency; perform well when groups are dense and sparse.

However, k-means also has some shortcomings at the same time: it need to input group number before-

time and it depends on initialization very much. Since K-means is randomly initialized, so, it will produce different results in different trail. Also, when the initialization is not good, k-means cannot produce a correct classification result.

(b) Texture Segmentation (20%)

1. Motivation

When we are done with image texture feature extraction, the next thing we can do is texture segmentation. Texture segmentation is to split an image which consists of multiple kinds of texture into different parts. Texture segmentation is critical in some application area, such as digital map generation, object tracking, etc.

2. Approach

Similar to the steps we used in prob.1.(a), we will first subtract the image mean and then apply 25 Laws filters to the input image first and get 25 filtered gray scale images. What's different from prob.1.(a) is that we do not have 12 images to cluster here, instead, we need to "generate" images to be clustered.

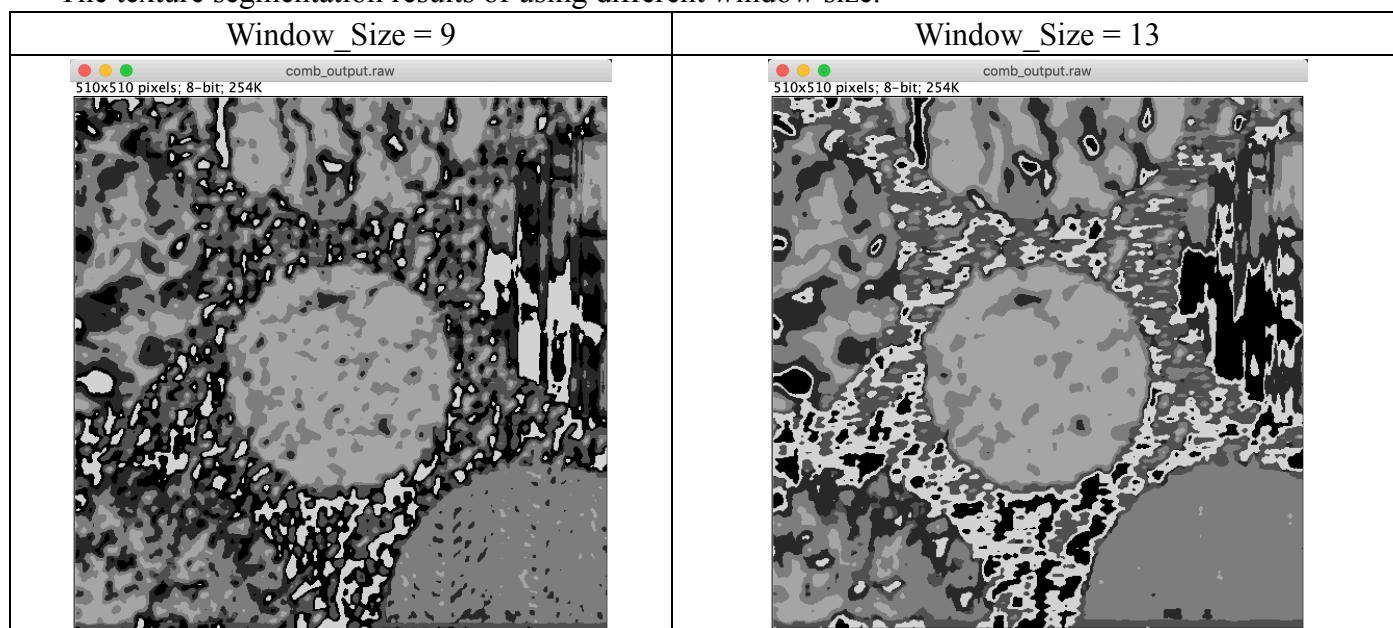
The 2nd step is energy feature computation. Here we will use a window with size of $N \times N$ to compute the energy measure for each pixel based on the results from step 1. In other words, for each feature dimension, we need to compute the square sum of all the neighbor pixels' values, and then normalize the square sum with the size of the window ($N \times N$). N is typically much larger than 5, so that we can "generate" images to be clustered. This procedure can be also referred as "macro-window energy computation". After this step, we will obtain a 25-D energy feature vector for each pixel.

The next step is energy feature normalization. In order to let each feature contributes approximately proportionately when computing Euclidean distance, we need to normalize all the feature energy to the range of $[0, 1]$.

After all the above processing, the final step is k-means clustering. We implement k-means clustering on comb.raw and set the cluster number as 7. To denote segmented regions, I assign each group with different pixel value, specifically (0, 42, 84, 126, 168, 210, 255).

3. Result

The texture segmentation results of using different window size.



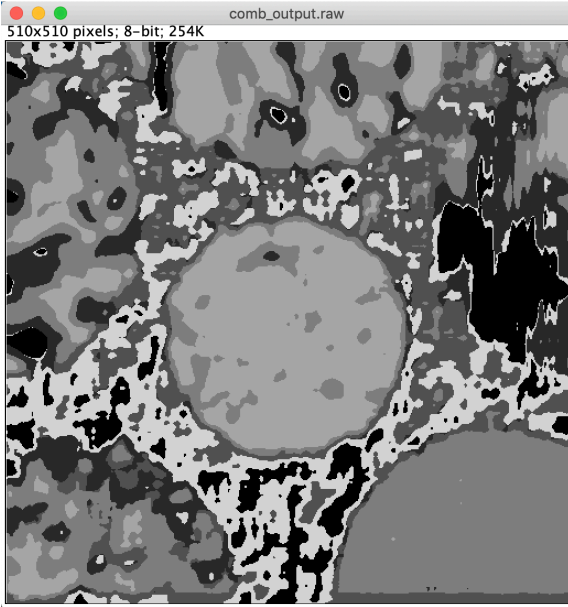
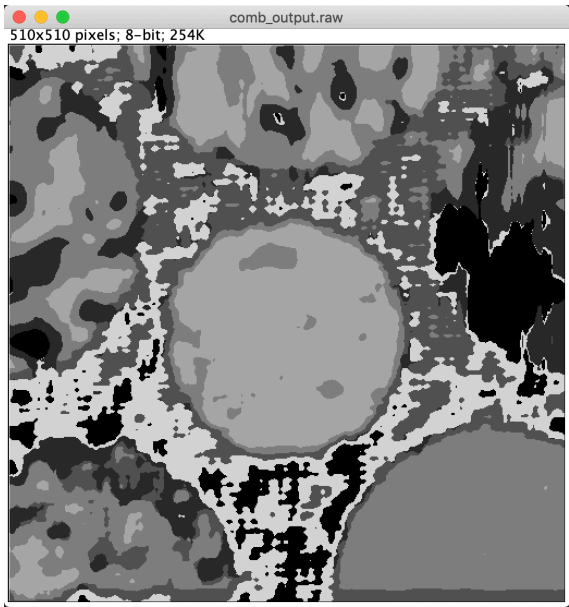

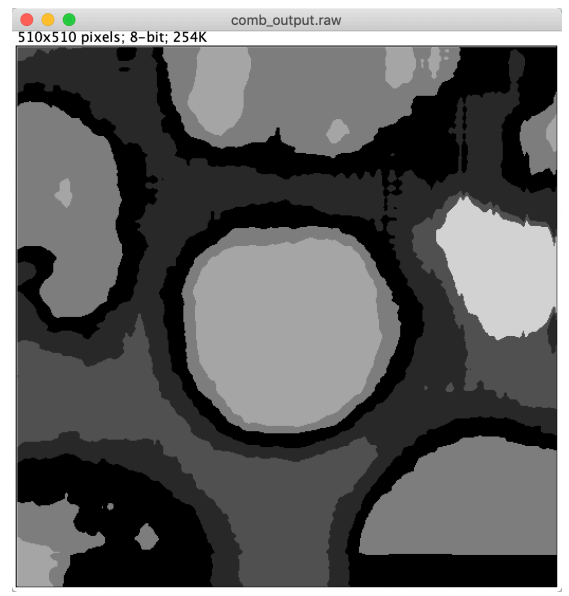
Window_Size = 17	Window_Size = 21
	
Window_Size = 41	Window_Size = 61
	

Table 2 Texture Segmentation Result Images

As is shown in the above table, I used 6 different window size to do energy feature computation and their performances are also different.

The result image with window size = 21 has relatively the best segmentation result. When the window size is too small, like =9 or =13, the segmented regions will have some holes inside a smooth region. And these holes will be mostly removed when we increased the window size. This is easy to explain: when the window size is small, the window will only consider a small neighbor and make a decision based on this small neighbor region; when the window size becomes larger, it will take larger neighbor region into consideration and make a better classification decision.

However, when the window size is too large, like =41 or = 61, it also leads to a bad segmentation result. Although those segmented regions will become smoother, but the boundaries will be blurring and fuzzy. Imagine this: when the window size is too large, it will probably contain 2 kinds of texture regions (or even more), and in this situation, this algorithm can't make a good decision for the center pixel.

So, choosing a suitable window size is critical here. Here, I found the window size of 21 has the best texture segmentation result. Hence, I developed some methods to improve the segmentation quality for this problem and I will discuss it in the next part, i.e. Prob.1.(c).

4. Discussion

As you can see from the result image above, the segmentation result is acceptable. You can find 7 segmented regions from the result image and you can distinguish their boundaries (some of them are a little blurring or fuzzy). However, there are still some defects in the segmented regions. Inside a certain region, there are some dots with other gray value, which lower the segmentation result quality.

I will talk about some methods to improve segmentation quality in next part.

(c) Advanced Texture Segmentation Techniques

1. Motivation

It is clear that the simple segmentation algorithm is not good enough. To improve the segmentation quality, we have several alternatives:

- Adopt PCA for feature reduction and cleaning;
- Develop a post-processing technique to merge small holes;
- Enhance the boundary of 2 adjacent regions by focusing on the texture properties in these 2 regions only.

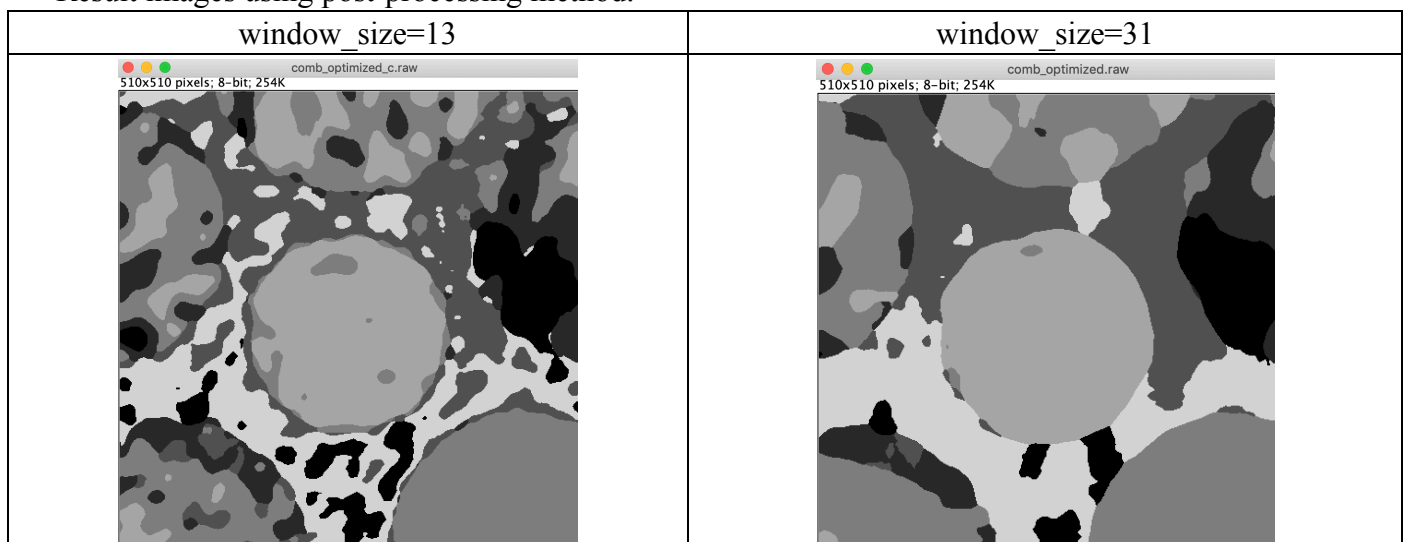
2. Approach

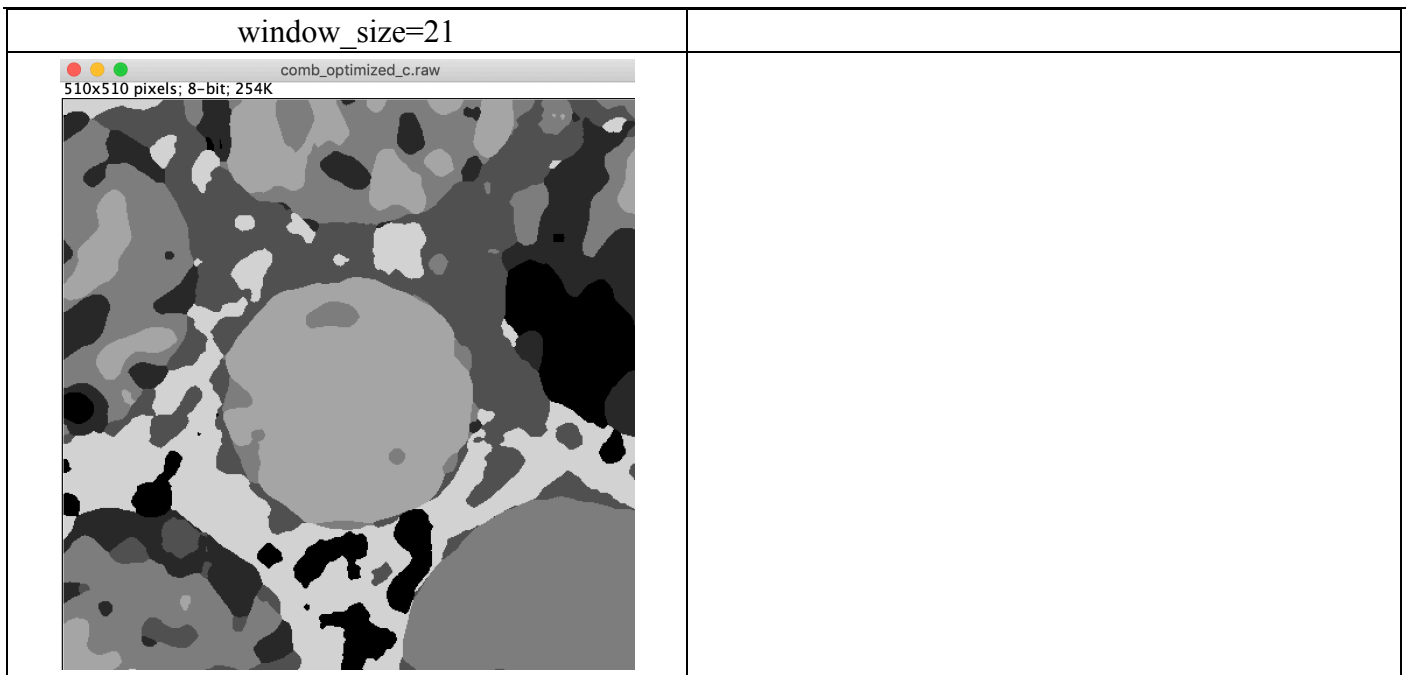
The first method I developed is a post-processing method to remove those apparent holes. I use a window with size 30x30 to scan the whole result image (result image in prob.1.(2)) and assign the center pixel with the most frequently pixel value in the window. This method can easily remove those apparent holes inside a smooth region. As you can see from the result image below, the segmentation quality is improved. You can distinguish these segmented regions more easily and clearly.

Second, I applied PCA to reduce feature vector dimension, so that the K-means will focus on those most important features and make the segmentation result better. This method can make the boundaries smoother and remove most of the holes inside a segmented region.

3. Result

Result images using post-processing method.





4. Discussion

As you can see from the above result image table, when I applied my post-processing algorithm, the segmentation result will be smoother. The boundaries are clear and not fuzzy. The segmentation quality is improved by this method.

Problem 2: Image Feature Extractor (50%)

(a) SIFT (20%)

1. Motivation

Since feature extraction is a critical task for computer vision, researchers developed lots of different feature extraction methods. One of those methods is SIFT, known as Scale Invariant Feature Transform. SIFT can extract features from an image accurately and quickly, and it can compare these features with a new image, to determine whether there are matching features in the new image. SIFT is robust to geometrical transformation, so it is very useful and effective.

2. Approach

Here I just list the main steps of SIFT.

1. Building Gaussian pyramid by applying different Gaussian kernels;
2. Using DoG to find local extrema and treat them as key points;
3. Removing those key points with low local contrast and unstable edge response;
4. Assigning key points with gradient direction;
5. Generating 128-D descriptor for each key point.

3. Result

This problem contains no result. (Skip...)

4. Discussion

I. From the paper abstract, the SIFT is robust to what geometric modifications?

SIFT is robust to image scaling, translation, rotation.

II. How does SIFT achieves its robustness to each of them?

- **Scale**
SIFT uses multi-scale filtering. It uses Gaussian kernel and its derivatives as smoothing kernels for scale space analysis.
- **Rotation**
Select key locations at maxima and minima of a difference of Gaussian function applied in scale space. First, we need to build an image pyramid by resampling between each level. Then, we locate key points at regions and scales by using DoG to find local maxima/minima.

In short, we locate key points as maxima/minima of DoG function in order to achieve rotation-invariant. Each key-point is assigned with a canonical orientation.

- **Translation**
(Same as Rotation)

Select key locations at maxima and minima of a difference of Gaussian function applied in scale space. First, we need to build an image pyramid by resampling between each level. Then, we locate key points at regions and scales by using DoG to find local maxima/minima.

In short, we locate key points as maxima/minima of DoG function in order to achieve rotation-invariant. Each key-point is assigned with a canonical orientation.

III. How does SIFT enhances its robustness to illumination change?

First, this is done by thresholding the gradient magnitudes at a value of 0.1 times the possible maximum possible gradient value. Because illumination variation may lead to significant changes to gradient magnitude but less changes to gradient orientation. So, by times 0.1, we reduce the influence of gradient magnitude.

Second, the canonical orientation is determined by the peak in a histogram of local image gradient orientation. In such case, SIFT can be illumination-robust.

IV. What are the advantages that SIFT uses difference of Gaussians (DoG) instead of Laplacian of Gaussians (LoG)?

First, DoG is faster than LoG.

Also, DoG is robust to scale transformation while LoG is not. LoG is a blob detector and when the blobs scale change, LoG will give a response for blobs in different scale, i.e. not robust to scale change.

V. What is the SIFT's output vector size in its original paper?

Output vector size = $8*4*4 + 8*2*2 = 160$

(b) Image Matching (20%)

1. Motivation

After we done with features extraction, we need to match features between original image and new input image. Image matching is also useful in real world.

2. Approach

SIFT can also do image matching. When we calculate the Euclidean distance between 2 sets of 128-D descriptors (original image and new image): if the Euclidean distance is small, then there is a high similarity between original image and new image; on the contrary, if the Euclidean distance is large, then there is a small similarity between original image and new image. By setting a threshold to the Euclidean distance, we can do image matching on our own settings.

3. Result

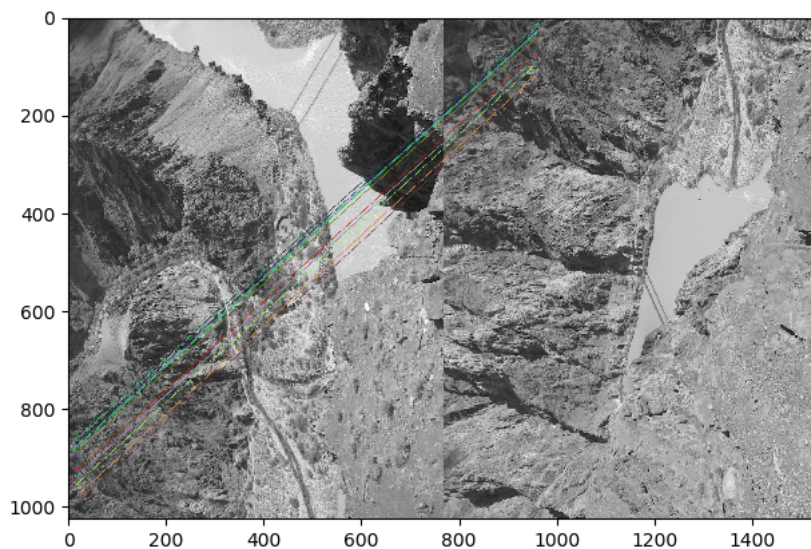


Figure 5 SIFT Result Image

My result image above can find matching key points between river1 and river2.

4. Discussion

Q: Discuss your results, esp. the orientation of each key-points.

A: My result can find some certain features, extract them and match them.

(c) Bag of Words (10%)

1. Motivation

In the real world, images and their content (objects in the image) cannot be represented as matrix or numbers. In order to convert images and their contents to data that we can do classification to, we need to introduce Bag of Words method. Bag of Words can represent images and objects as a N dimensional vector, so that we can do image processing, classification, segmentation on images.

2. Approach

The first step of Bag of Words is feature detection. To do this feature detection, we can implement SIFT to extract key points from the original image as a visual vocabulary. Then, we will apply k-means to those samples and cluster them into different group and thus generate a word dictionary.

After above procedure, a feature in the original image is represented as a “word”. Different features have different corresponding words and we will finally obtain a dictionary of features. This dictionary of features is treated as a N dimensional vector. To measure the similarity between images, we just count the number of matched words. Usually, we express this criterion in histogram to show how many words are matching.

3. Result

4. Discussion

The SIFT performance here is not very good, because the “8” in the image does not look like “0” or “1”. When we plot the histogram of matched features, we can see that both “0” and “1” do not have many correct matches. SIFT is not as good as some other advanced feature extraction and matching algorithm in this specific problem.

Reference

- [1] William K. Pratt. 2007. Digital Image Processing: PIKS Scientific Inside. Wiley-Interscience, New York, NY, USA.
- [2] Lowe, David G. "Object recognition from local scale-invariant features." iccv. Ieee, 1999.