Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

Problem 1: Texture Analysis

I. Abstract and Motivation

Texture analysis analyze textures in images. There are multiple subdivided tasks under this area, such as texture classification, texture segmentation. People study this technique to encode and utilize textures in further image processing procedures. People can understand textures in images, and this technique help machines to simulate human visual system. In this problem I am asked to do two basic tasks in texture analysis area using classic methods.

II. Approach and Procedures

Laws Filters

Laws designed several filters to analyze textures, these filters are called laws filters. Filters are generated by multiply two of all 1D kernels. Each kernel is a 1D array and can detect one certain pattern in one direction of images. For kernel size of 5, there are 5 kinds of kernels

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

Thus, there are total 25 laws filters of size 5 * 5, one filter can be generated by matrix multiply operation $\mathrm{F} = \mathrm{K}_1{}^T K_2$, where K_1 and K_2 is one of 5 kernels. All kernels are called

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

Since each kernel detects one certain pattern in one direction, so for a filter, the first kernel detects certain pattern vertically and the second kernel detects another pattern horizontally. After applying one of these filters on a given image, it will give a response image, and this response indicates how this combined 2D pattern embedded in given image. So, we can get total 25 response images.

Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

Pre-processing: Subtracting Local Average

However, it is better to perform an addition operation before applying laws filters. This operation subtracts the local average from each pixel value to get new pixel values. The local average is calculated by averaging values within a window around given pixel. The window size is fixed for all pixels, and it should be larger than laws filters' size. All pixels in local region will have a zero mean. The purpose of this is to eliminate the effect of luminance changes on subsequent results using laws filters. So that response values in different regions in a response image can be used to compare with each other.

Energy Measures

Since one pixel can not show any texture, so we average all response values in a small region to represent the response strength of each pixel. And this value is called the pixel's energy. The default window size for averaging is 15 *15, and it can be changed. Since both positive and negative response values exist, adding them together may not be a good idea, so each value should be replaced by its absolute value or square to accommodate the averaging operation before averaging. The differnet magnitudes metrics here will lead significant difference in final results in classification or segmentation tasks. Here in my program, squared magnitude is selected according to the supplemental requirement of this problem. And also Laws said squared magnitude gives better results [1].

For each filter, each pixel will have an energy measurement. So, there will be 25 energy measurements for all 25 vectors. Then compile these energy measurements into a 25D vector, and this vector becomes the high level representation of the texture around the pixel.

Optional: Combining Symmetric Pairs

There some symmetric pairs of laws filters, some time they can be combined since they detect 2D patterns with similar underling structure. As a result, one 25D vector can be reduced to a 15D vector. The replacing value of a symmetric pair is the average of their original value. All 15 dimensions are L5L5, E5E5, S5S5, W5W5, R5R5, L5E5/E5L5, L5S5/S5L5, L5W5/W5L5, L5R5/R5L5, W5R5/R5W5, E5S5/S5E5, E5W5/W5E5, E5R5/R5E5, S5W5/W5S5, S5R5/R5S5.

However, this procedure is optional, if one texture always appears in one direction, a vector without combining symmetric pairs may represent that texture better.

Normalization

For texture segmentation task, it is hard to compare different vectors get from same input image, since the range of response varies largely. The values in the vector should be normalized to accommodate further comparation.

We can see that all kernels except L5 has a zero mean, and pixel values also have a zero mean after subtracting local average, so energy values will have a zero mean. L5L5 energy value have a positive mean, so we can divide all other 24 values by the value of L5L5 to normalize all 25 dimensions to value range from 0 to 1.

Standardization

In texture classification task, we have several sample images. Each image contains only one kind of texture. So, each image can be represented using one feature vector rather than feature vectors for every pixel. This can be done by a procedure similar to the average energy measurement of one pixel. All the energy values of response are averaged to get one dimension of the feature vector.

Then we combine all feature vectors together to from a matrix that each row representing a sample and each column representing a feature dimension. Feature vectors are transposed then stacked together. Before further processing procedures, values in this matrix should be replaced by dimensionless quantities otherwise the comparation one dimension between different samples will be meaningless. This is done for all columns of the matrix individually. The equation for each element x_i is

$$x_i' = \frac{x_i - \bar{x}}{std(x)}$$

where $\bar{x}$ is the mean value of all samples and $std(x)$ is the standard deviation of all samples.

Singular Value Decomposition (SVD)

SVD decompose a matrix into three parts

$$\mathrm{A}_{m\times n} = U_{m\times m}\Sigma_{m\times n}{V_{n\times n}}^T$$

It is similar to eigen decomposition and can be performed to non-square matrices. $\Sigma_{m\times n}$ contains descending order singular values in its diagonal positions. It can be seen as geometrical transformations of original matrix.

Principal Component Analysis (PCA)

PCA using fewer dimensions to represent high dimensional data with fewer loss. It reduce the dimension of data and make the data easier to analyze and understand. SVD can be used for PCA, by discarding insignificant singular values in matrix $\Sigma_{m\times n}$, so the original matrix can be estimated by

$$\mathrm{A}_{m\times n} = U_{m\times r}\Sigma_{r\times r}{V^T}_{r\times n}$$

Then reduce the dimension of A by

$$\widetilde{A_{r\times n}} = {U^T}_{r\times m}\mathrm{A}_{m\times n} = (U^T U)\Sigma_{r\times r}{V^T}_{r\times n}$$

That is the dimension of matrix A is reduced from m to r, with minimal loss.

Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

K-means

K-means is an unsupervised machine learning algorithm. It can partition data set in high dimensional space into k clusters. It runs multiple iterations and adjust the center of each cluster by averaging data value belonging that cluster, then redistribute data values to its nearest cluster by computing the distance to all cluster centers. The original k-means algorithm initializes k cluster centers by randomly selecting k data values as centers. Another modified algorithm called k-means++ has different initializing strategies, it chooses the mean of all data values as the center of first cluster, then for following clusters, the center is initialized as the farthest data value from selected cluster centers as the center of the new cluster.

However, since the initializing procedure introduce some randomness, k-means may return different results among multiple times of run, so to get a result with better partition (a partition with minimal total distance to cluster centers), the k-means should be run multiple times and record the best result as the final result.

(Bonus option) I also implemented k-means in Matlab by myself and it works, however, its performance is not as fast as the built-in one and takes a long time for sub problem b & c since there are more feature vectors. That is because the Matlab programming language itself is inefficient. So, I set an argument to determine whether using mine k-means, the default choice is the built-in one.

Texture Classification

There are 12 sample texture images. I extract their texture vectors with different values of arguments and then use k-means to classify them. The classification results are compared to the ground truth classification to see correct rates.

Texture Segmentation

In this texture segmentation task, an image combining 7 different textures is given. And we can see there are 7 connected regions by eyes. I am asked to segment the texture by using laws filters and k-means. Now each pixel is represented by a 25D feature vector. Set k in k-means to 7 and feed all vectors into k-means algorithm, then I can get a segmentation result for all pixels. To illustrate segmentation results, a segmentation result image is created with pixels in same cluster assigned to one gray value, and total 7 different gray values.

Advanced Texture Segmentation Techniques

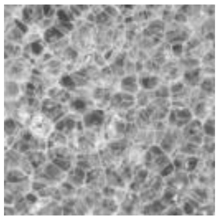
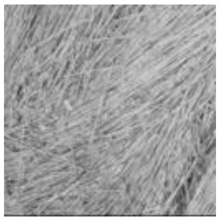
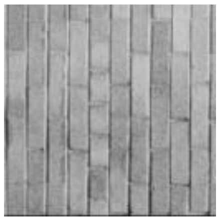
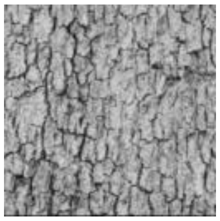
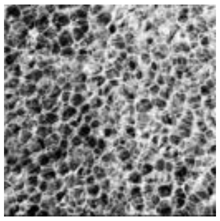
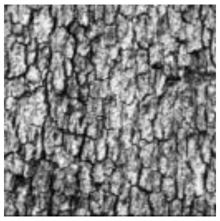
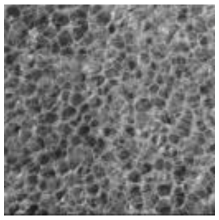
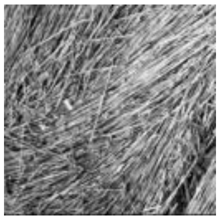
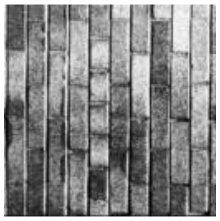
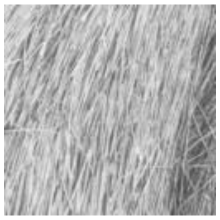
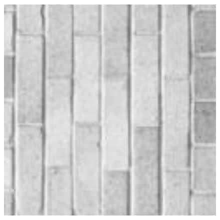
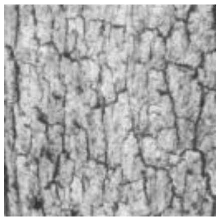
I first use PCA to see if the results are better.

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

III. Experimental Results

Texture Classification

12 textures are

					
Texture 1	Texture 2	Texture 3	Texture 4	Texture5	Texture 6
					
Texture 7	Texture 8	Texture9	Texture 10	Texture 11	Texture 12

Local average window size here is 15 * 15 (default of Laws Energy Measurement approach), local mean window size is the same as the local average window size.

Truth segmentation result is [1 2 3 4 1 4 1 2 3 2 3 4], this is invisible to the program.

The order of filters of 25D vectors is [L5L5 E5L5 S5L5 W5L5 R5L5 L5E5 E5E5 S5E5 W5E5 R5E5 L5S5 E5S5 S5S5 W5S5 R5S5 L5W5 E5W5 S5W5 W5W5 R5W5 L5R5 E5R5 S5R5 W5R5 R5R5].

The order of filters of 15D vectors is [L5L5 E5E5 S5S5 W5W5 R5R5 L5E5/E5L5 L5S5/S5L5 L5W5/W5L5 L5R5/R5L5 W5R5/R5W5 E5S5/S5E5 E5W5/W5E5 E5R5/R5E5 S5W5/W5S5 S5R5/R5S5].

Standard deviations of 25 feature dimensions after normalization and before standardization through 12 samples

Standard deviations	Min deviation	Max deviation
0 0.0091 0.0031 0.0028 0.0117 0.0067 0.0010 0.0004 0.0004 0.0019 0.0062 0.0005 0.0002 0.0002 0.0011 0.0091 0.0006 0.0003 0.0003 0.0016 0.0495 0.0029 0.0013 0.0017 0.0086	0.0002121858570399640 (E5S5)	0.049488370090861 (R5W5)

Combining Symmetric Pairs DISABLED & PCA3D DISABLED (25D)

Feature matrix (after standardization)

```
   1.0000    1.0271    0.7244    0.4190  -0.0921  -1.1657    0.2260    0.1508    0.0345  -0.1103  -1.1176  -0.2104  -0.1635  -0.1652  -0.1803  -0.9923  -0.4417  -
```

```
0.3656 -0.3084 -0.2673 -0.8942 -0.5817 -0.5616 -0.4846 -0.3878
   1.0000    0.6227    1.0318    1.3066    1.5885  -1.1894    1.3190    1.5551    1.6264    1.6799  -0.6120    1.4564    1.6988    1.7294    1.7344  -0.4816    1.3435
1.7109    1.7643    1.7506 -0.4811    0.8610    1.4938    1.7362    1.7589
   1.0000 -1.3134 -1.2871 -1.2048 -0.8406    0.0616 -1.3713 -1.1414 -1.0030 -0.8731    1.2242 -1.2377 -1.0460 -0.9200 -0.8167    1.3778 -1.0728 -0.9713
-0.8705 -0.7814    1.4879 -0.8344 -0.8991 -0.8326 -0.7780
   1.0000 -0.2369 -0.3018 -0.3454 -0.3679 -0.3237 -0.2849 -0.3488 -0.3858 -0.4351 -0.5331 -0.3578 -0.4032 -0.4286 -0.4623 -0.6170 -0.4047 -0.4481 -
0.4605 -0.4668 -0.6201 -0.3768 -0.4762 -0.4849 -0.4571
   1.0000    1.1156    0.8975    0.6649    0.4944  -0.6176    0.4936    0.3474    0.2684    0.2146  -0.7883    0.1042    0.0764    0.0611    0.0441  -0.8029  -0.1772  -
0.1354 -0.0822 -0.0505 -0.8043 -0.3799 -0.3184 -0.2326 -0.1476
   1.0000    0.1229    0.0814    0.0961  -0.0237    0.5742    0.0918  -0.0087  -0.0398  -0.1056  -0.1572    0.0193  -0.0719  -0.1169  -0.1731  -0.3632  -0.0352  -
0.1201 -0.1712 -0.2216 -0.4133 -0.0492 -0.1703 -0.2282 -0.2343
   1.0000    1.3302    1.1630    0.9589    0.7291  -0.1739    0.7435    0.5747    0.4816    0.3858  -0.5889    0.3470    0.2822    0.2390    0.1838  -0.6781    0.0126
0.0257    0.0554    0.0535 -0.7352 -0.2565 -0.2094 -0.1444 -0.0919
   1.0000    0.9484    1.3503    1.6586    1.9177  -0.8497    1.6809    1.9734    2.0880    2.1761  -0.4652    1.8011    2.1304    2.2287    2.2865  -0.3762    1.6775
2.1518    2.2865    2.3458 -0.3939    1.1542    1.9340    2.2730    2.3703
   1.0000 -1.1770 -1.1293 -0.9935 -0.6556    1.6771 -1.1944 -0.9768 -0.8201 -0.6529    2.0238 -1.0529 -0.8861 -0.7486 -0.6261    2.0407 -0.8735 -0.8004
-0.6995 -0.6125    2.0489 -0.6381 -0.7128 -0.6568 -0.6075
   1.0000  -0.8168  -1.0492  -1.1697  -1.2469    1.0771    0.1082  -0.5064  -0.7286  -0.8328    0.9191    0.8956  -0.0460  -0.4247  -0.6075    0.8931    1.6007
0.4690 -0.1172 -0.4266    0.8826    2.4709    1.3671    0.3913 -0.1504
   1.0000 -1.3003 -1.0585 -0.9217 -0.9220    1.5255 -1.4244 -1.1751 -1.0544 -0.9542    0.8960 -1.2894 -1.0855 -0.9636 -0.8752    0.7628 -1.1199 -1.0040
-0.8939 -0.8124    0.6236 -0.8933 -0.9228 -0.8288 -0.7681
   1.0000 -0.3226 -0.4223 -0.4691 -0.5809 -0.5954 -0.3878 -0.4443 -0.4671 -0.4923 -0.8006 -0.4752 -0.4855 -0.4907 -0.5076 -0.7631 -0.5091 -0.5124 -
0.5028 -0.5108 -0.7009 -0.4761 -0.5244 -0.5077 -0.5064
```

All possible segmentation results

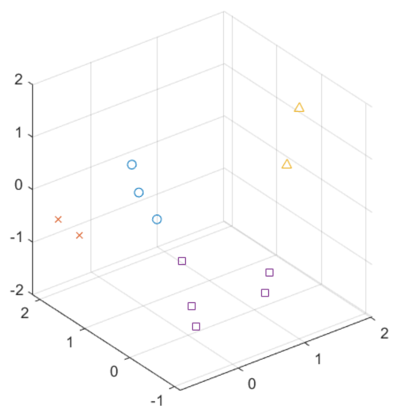
Correct rates	Segmentation results
0.9167	1 2 3 4 1 4 1 2 3 4 3 4
0.8333	1 2 3 4 1 1 1 2 3 1 3 4
0.8333	1 2 3 4 1 1 1 2 3 4 3 4
0.6667	1 2 3 1 1 1 1 2 3 4 3 1 (Segmentation with minimal total distance)
0.5833	1 2 3 1 1 1 1 2 3 1 4 1
0.5833	1 2 3 1 1 1 1 2 4 1 3 1
0.5833	1 2 3 1 4 4 4 2 3 4 3 1
0.5833	1 2 3 1 1 1 1 4 3 3 3 1
0.5833	1 2 3 3 1 1 1 2 4 4 3 3
0.5833	1 2 3 1 1 1 1 4 3 1 3 1
0.5833	1 2 3 1 4 4 4 2 3 3 3 1
0.5000	1 2 3 1 4 1 4 2 3 3 3 1
0.3333	1 1 2 1 1 1 1 1 3 4 2 1
0.3333	1 2 1 1 1 1 1 3 1 4 1 1
0.3333	1 1 2 2 1 3 1 1 3 4 2 2
0.2500	1 1 2 1 1 1 1 1 2 3 4 1

Combining Symmetric Pairs DISABLED & PCA3D ENABLED (25D→3D)

Feature matrix (after standardizing)

```
  -0.8874  -0.6167    0.9802
  -0.5618    1.5976  -0.7173
    1.4726  -0.0331    0.5537
  -0.5635  -0.9175  -0.6630
  -0.8129  -0.1003    1.1688
  -0.3776  -0.4319  -0.1622
  -0.7532    0.1436    1.5358
  -0.5009    2.1668  -0.7551
    2.0172    0.4967    1.0705
    0.9130  -0.3674  -1.4273
    0.6924  -0.8059  -0.6930
  -0.6381  -1.1319  -0.8911
```

Figure (Segmentation with minimal total distance; this figure is the 2D projection of a 3D figure which may lose information)



All possible segmentation results

Correct rates	Segmentation results
0.9167	1 2 3 4 1 4 1 2 3 3 3 4
0.8333	1 2 3 4 1 4 1 2 3 4 4 4 (Segmentation with minimal total distance)
0.6667	1 2 3 3 1 3 1 2 3 4 3 3
0.5833	1 2 3 1 1 1 1 2 3 4 4 1
0.5833	1 2 3 3 1 3 1 2 3 4 4 3
0.5833	1 2 3 2 1 2 1 2 3 4 4 2
0.5833	1 2 3 3 1 3 4 2 3 3 3 3
0.5833	1 2 3 2 1 2 1 2 4 2 2 2
0.5833	1 2 3 3 1 3 1 4 3 3 3 3
0.5833	1 2 3 1 1 1 1 4 3 3 3 1
0.5000	1 2 3 1 1 1 1 2 4 1 1 1
0.5000	1 2 3 3 4 3 4 2 3 3 3 3
0.5000	1 2 3 2 1 2 1 1 3 4 4 2

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

0.5000	1 2 3 2 4 2 4 2 3 2 2 2
0.4167	1 2 1 3 1 3 1 4 1 3 3 3
0.4167	1 2 1 3 1 3 1 2 1 4 4 3
0.4167	1 2 3 1 3 1 3 2 3 4 4 1
0.4167	1 2 2 3 1 3 1 2 2 4 4 3
0.4167	1 2 3 1 1 1 3 4 3 1 1 1
0.3333	1 1 2 3 1 3 1 1 4 3 3 3
0.3333	1 1 2 1 1 1 1 1 3 4 4 1
0.2500	1 1 2 3 1 3 1 1 2 4 4 3

Combining Symmetric Pairs ENABLED & PCA3D DISABLED (25D→15D)

Feature matrix (after standardization)

```
1.0000   0.2260 -0.1635 -0.3084 -0.3878   0.2534 -1.1207 -1.0619 -1.0254 -0.3825 -0.0405 -0.2504 -0.4699 -0.2725 -0.4080
1.0000   1.3190   1.6988   1.7643   1.7589 -0.3836 -0.1442 -0.0981 -0.1174   1.7592   1.5435   1.5848   1.4158   1.7384   1.6931
1.0000 -1.3713 -1.0460 -0.8705 -0.7780 -1.9146   0.8627   1.2388   1.4428 -0.8151 -1.2241 -1.1249 -1.0135 -0.9570 -0.9090
1.0000 -0.2849 -0.4032 -0.4605 -0.4571 -0.7180 -1.0133 -0.8894 -0.7918 -0.4805 -0.3630 -0.4279 -0.4771 -0.4435 -0.4959
1.0000   0.4936   0.0764 -0.0822 -0.1476   0.9972 -0.5048 -0.7360 -0.7693 -0.1454   0.2252   0.0206 -0.1707 -0.0417 -0.1605
1.0000   0.0918 -0.0719 -0.1712 -0.2343   0.8247 -0.1729 -0.4104 -0.4690 -0.2270   0.0062 -0.0402 -0.0855 -0.1199 -0.1811
1.0000   0.7435   0.2822   0.0554 -0.0919   1.8150 -0.0130 -0.4712 -0.6297 -0.0486   0.4670   0.2366 -0.0008   0.1292 -0.0305
1.0000   1.6809   2.1304   2.2865   2.3703   0.4863   0.3088   0.1648   0.0676   2.3296   1.9332   2.0061   1.8619   2.2123   2.2119
1.0000 -1.1944 -0.8861 -0.6995 -0.6075   0.0894   2.1644   2.1341   2.1199 -0.6411 -1.0442 -0.9176 -0.7680 -0.7840 -0.7103
1.0000   0.1082 -0.0460 -0.1172 -0.1504 -0.0345   0.5864   0.6560   0.6574 -0.0064   0.2381   0.6204   1.3777   0.0413   0.4865
1.0000 -1.4244 -1.0855 -0.8939 -0.7681 -0.2655   0.5453   0.5895   0.4536 -0.8284 -1.2685 -1.1778 -1.0944 -0.9953 -0.9509
1.0000 -0.3878 -0.4855 -0.5028 -0.5064 -1.1499 -1.4988 -1.1160 -0.9387 -0.5139 -0.4729 -0.5296 -0.5756 -0.5074 -0.5453
```

All possible segmentation results

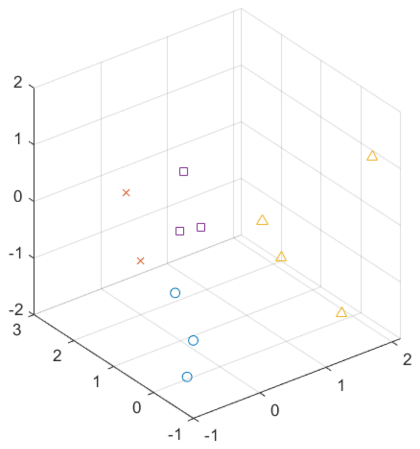
Correct rates	Segmentation results
0.8333	1 2 3 4 1 1 1 2 3 4 3 4
0.6667	1 2 3 1 1 1 1 2 3 4 3 1
0.5833	1 2 3 1 4 4 4 2 3 4 3 1 (Segmentation with minimal total distance)
0.5833	1 2 3 3 1 1 1 2 4 4 3 3
0.5833	1 2 3 1 1 1 1 2 3 1 4 1
0.5833	1 2 3 1 1 1 4 2 3 4 3 1
0.5833	1 2 3 1 1 1 1 4 3 1 3 1
0.5833	1 2 3 1 1 1 1 2 4 1 3 1
0.5833	1 2 3 3 1 1 1 2 4 1 3 3
0.3333	1 1 2 1 1 1 1 1 3 1 4 1
0.2500	1 2 1 1 3 3 3 4 3 3 1 1

Combining Symmetric Pairs ENABLED & PCA3D ENABLED (25D→15D→3D)

Feature matrix (after standardization)

```
 -1.0139 -0.5515   0.0121
 -0.1585   1.7278 -0.8434
   1.4575 -0.6209 -1.4268
 -0.7761 -0.6108 -0.9049
 -0.7651 -0.2436   0.8490
 -0.4621 -0.2801   0.7982
 -0.6268 -0.1317   1.7791
   0.0094   2.3849 -0.0179
   2.1256 -0.2811   0.8761
   0.6551   0.0405   0.2637
   0.4783 -0.7333   0.0539
 -0.9233 -0.7003 -1.4390
```

Figure (Segmentation with minimal total distance; this figure is the 2D projection of a 3D figure which may lose information)



All possible segmentation results

Correct rates	Segmentation results
0.8333	1 2 3 4 1 1 1 2 3 3 3 4
0.7500	1 2 3 4 1 1 1 2 3 1 1 4
0.7500	1 2 3 4 1 1 1 2 2 2 2 4
0.6667	1 2 3 4 1 1 1 2 4 4 4 4
0.6667	1 2 3 3 1 1 1 2 4 2 3 3
0.5833	1 2 3 1 4 4 4 2 3 3 3 1 (Segmentation with minimal total distance)
0.5833	1 2 3 3 1 1 1 2 4 4 3 3
0.5833	1 2 3 1 1 1 1 4 3 3 3 1
0.5833	1 2 3 1 1 1 1 2 4 3 3 1
0.5833	1 2 3 3 1 1 1 2 4 3 3 3
0.5833	1 2 3 3 1 1 4 2 3 3 3 3
0.5833	1 2 3 1 3 3 3 2 3 3 3 4

Zongjian Li

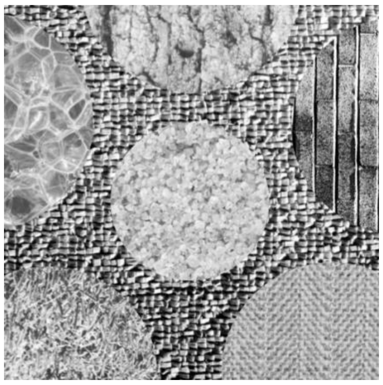
6503-3789-43

zongjian@usc.edu

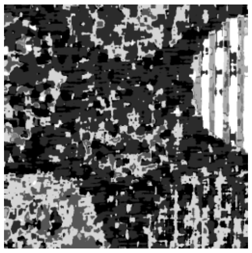
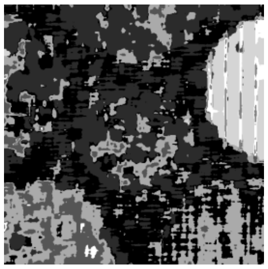
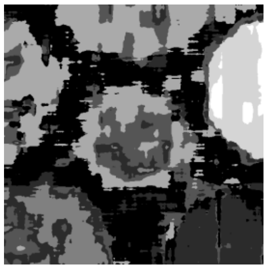
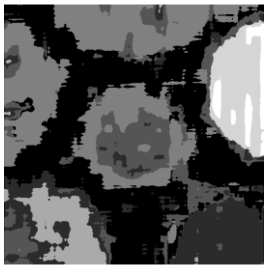
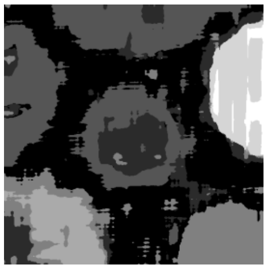
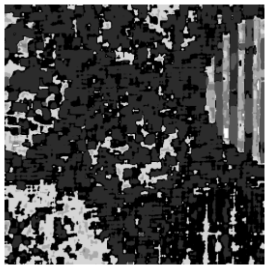
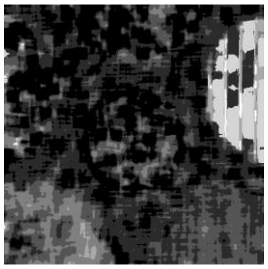
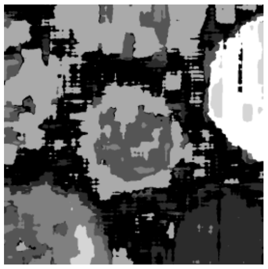
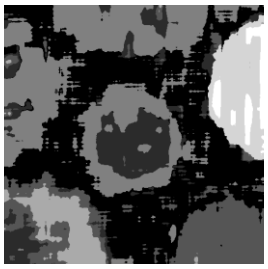
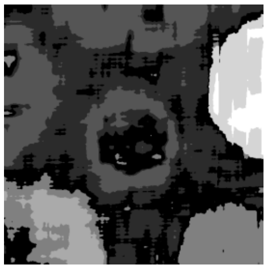
0.5833	1 2 3 3 1 1 1 4 3 3 3 3
0.5833	1 2 3 1 1 1 1 2 4 4 3 1
0.5833	1 2 3 2 1 1 1 2 4 3 3 2
0.5833	1 2 2 3 1 1 1 2 2 2 2 4
0.5000	1 2 3 3 1 1 1 2 4 1 1 3
0.5000	1 2 3 3 1 1 1 2 4 4 1 3
0.5000	1 2 3 1 4 4 4 2 3 1 1 1
0.5000	1 2 2 3 1 1 1 2 4 2 2 3
0.5000	1 2 3 1 1 1 1 2 4 4 4 1
0.5000	1 2 3 1 4 4 4 2 3 4 4 1
0.5000	1 2 3 1 1 1 1 2 4 4 1 1
0.5000	1 2 3 1 4 4 4 2 2 2 1 1
0.5000	1 2 3 3 1 1 1 2 4 4 4 3
0.5000	1 2 3 1 1 1 1 2 4 1 1 1
0.5000	1 2 3 1 4 4 4 2 3 4 1 1
0.5000	1 2 3 2 1 1 1 1 4 3 3 2
0.5000	1 2 3 1 1 1 1 4 3 1 1 1
0.5000	1 2 3 1 4 4 4 2 4 4 3 1
0.5000	1 2 3 1 1 1 4 2 4 4 3 1
0.5000	1 2 3 2 1 1 1 2 4 4 4 2
0.5000	1 2 3 1 4 4 4 2 2 2 2 1
0.5000	1 2 3 2 1 1 4 1 3 3 3 2
0.4167	1 2 3 1 4 4 4 2 4 4 4 1
0.4167	1 2 3 2 1 1 1 1 4 1 1 2
0.4167	1 2 2 3 1 1 4 2 2 2 2 3
0.4167	1 2 3 2 1 1 1 1 4 4 4 2
0.4167	1 2 3 1 2 2 2 2 4 2 2 1
0.4167	1 2 3 1 3 3 3 2 4 3 3 1
0.4167	1 2 3 3 1 1 4 2 4 1 1 3
0.4167	1 2 3 1 3 3 3 4 3 3 3 1
0.4167	1 2 3 1 2 2 2 2 4 3 3 1
0.4167	1 2 3 1 4 4 4 2 1 1 1 1
0.4167	1 2 3 3 4 4 4 2 4 4 1 3
0.4167	1 2 3 1 1 1 1 4 1 1 1 1
0.4167	1 2 2 3 1 1 1 2 4 1 1 3
0.4167	1 2 3 3 1 1 1 4 1 1 1 3
0.4167	1 2 2 3 4 4 4 2 2 2 2 3
0.4167	1 2 3 2 1 1 4 1 3 1 1 2
0.3333	1 2 1 1 3 3 3 4 3 3 3 1

0.3333	1 2 2 1 3 3 3 2 4 2 2 1
0.3333	1 2 3 1 2 2 2 2 4 4 4 1
0.3333	1 2 1 1 1 1 3 4 3 1 1 1
0.3333	1 2 2 1 3 3 3 2 4 2 1 1
0.3333	1 1 2 1 1 1 1 1 3 4 4 1
0.2500	1 2 1 1 3 3 3 2 4 4 4 1
0.2500	1 2 1 1 3 3 3 2 4 4 1 1
0.2500	1 2 1 1 3 3 3 2 4 1 1 1
0.2500	1 1 2 3 1 1 1 1 4 1 1 3
0.2500	1 2 2 2 1 1 3 1 4 1 1 2
0.2500	1 2 1 1 3 3 3 4 3 3 1 1
0.1667	1 2 2 2 3 3 3 3 4 3 1 2
0.1667	1 1 2 1 3 3 3 3 4 3 3 1
0.1667	1 2 2 1 3 3 3 3 4 4 2 1
0.0833	1 1 2 1 3 3 3 3 4 4 2 1
0.0833	1 1 2 1 3 3 3 4 4 4 4 1
0.0833	1 1 2 1 3 3 3 3 4 4 4 1
0.0833	1 1 2 1 3 3 3 1 4 1 1 1
0.0833	1 1 2 2 3 3 3 4 4 4 1 2

Texture Segmentation

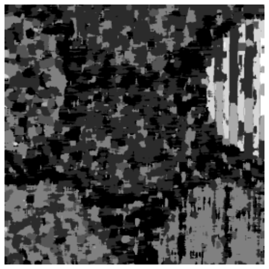
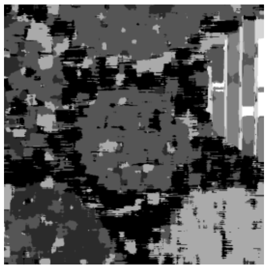
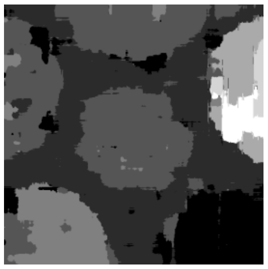
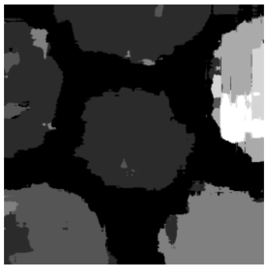
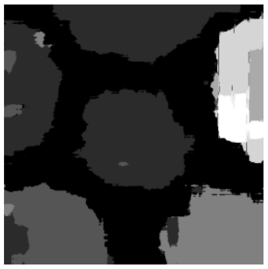


Composite texture image

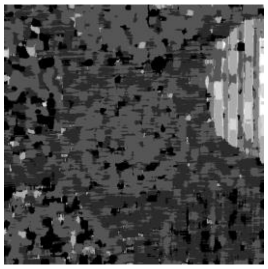
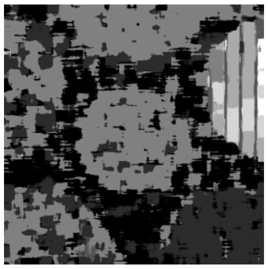
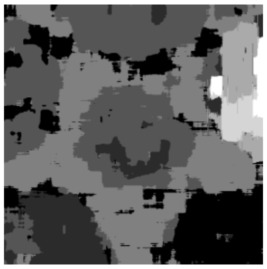
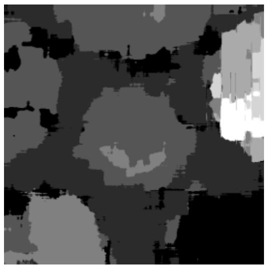
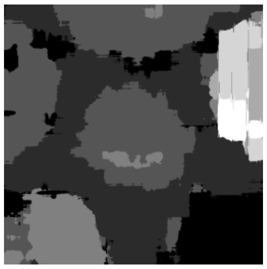
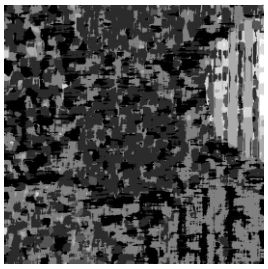
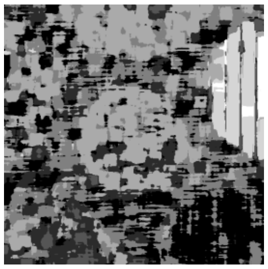
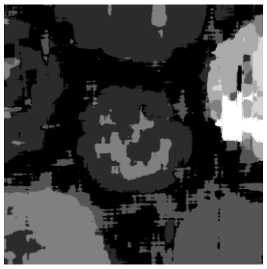
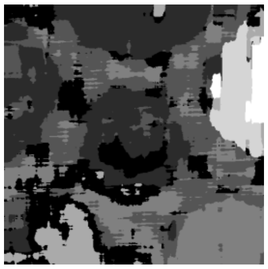
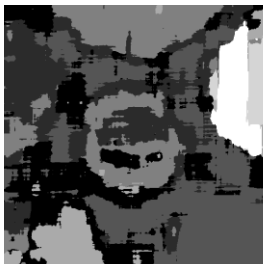
				
Segmentation (square, dim=25, k=7, window size=15)	Segmentation (square, dim=25, k=7, window size=23)	Segmentation (square, dim=25, k=7, window size=39)	Segmentation (square, dim=25, k=7, window size=47)	Segmentation (square, dim=25, k=7, window size=55)
				
Segmentation (square, dim=15, k=7, window size=15)	Segmentation (square, dim=15, k=7, window size=23)	Segmentation (square, dim=15, k=7, window size=39)	Segmentation (square, dim=15, k=7, window size=47)	Segmentation (square, dim=15, k=7, window size=55)

Advanced Texture Segmentation Techniques

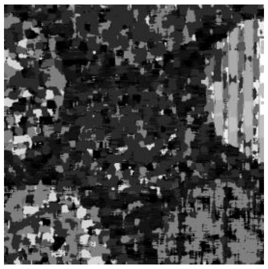
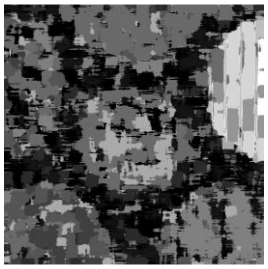
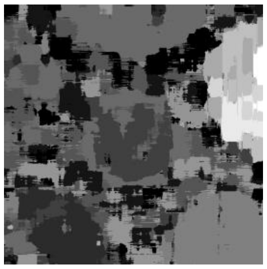
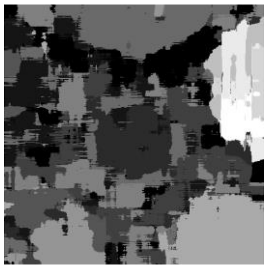
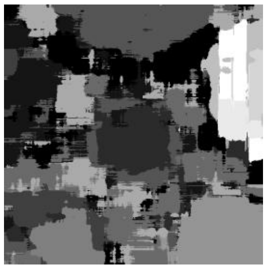
PCA

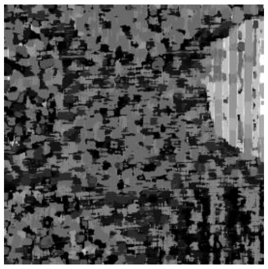
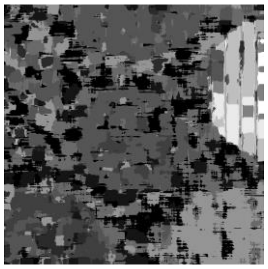
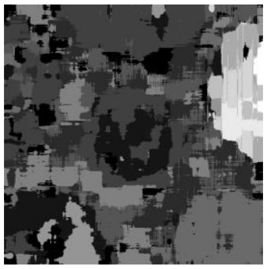
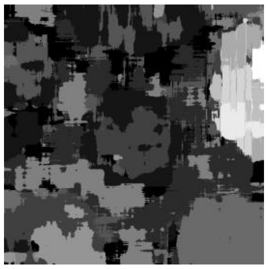
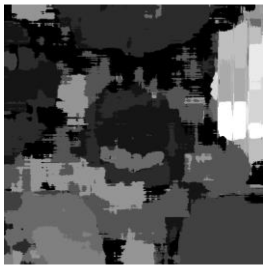
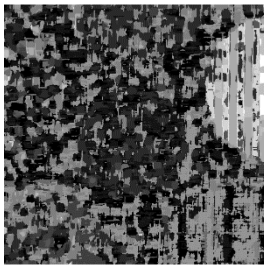
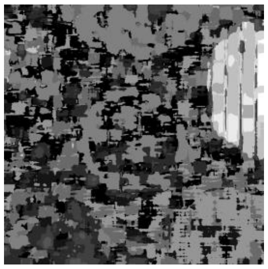
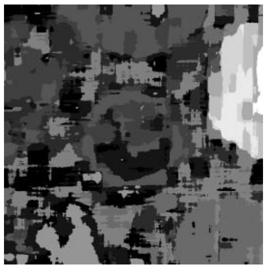
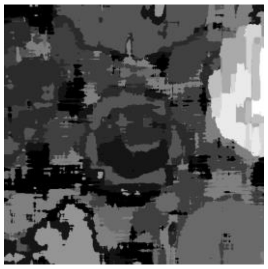
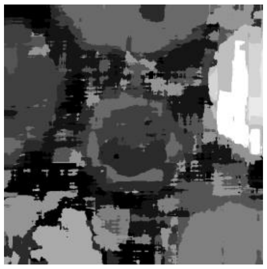
Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

Segmentation (square, dim=25, k=7, window size=15, pca_dim=12)	Segmentation (square, dim=25, k=7, window size=23, pca_dim=12)	Segmentation (square, dim=25, k=7, window size=39, pca_dim=12)	Segmentation (square, dim=25, k=7, window size=47, pca_dim=12)	Segmentation (square, dim=25, k=7, window size=55, pca_dim=12)
				
Segmentation (square, dim=25, k=7, window size=15, pca_dim=8)	Segmentation (square, dim=25, k=7, window size=23, pca_dim=8)	Segmentation (square, dim=25, k=7, window size=39, pca_dim=8)	Segmentation (square, dim=25, k=7, window size=47, pca_dim=8)	Segmentation (square, dim=25, k=7, window size=55, pca_dim=8)
				
Segmentation (square, dim=25, k=7, window size=15, pca_dim=4)	Segmentation (square, dim=25, k=7, window size=23, pca_dim=4)	Segmentation (square, dim=25, k=7, window size=39, pca_dim=4)	Segmentation (square, dim=25, k=7, window size=47, pca_dim=4)	Segmentation (square, dim=25, k=7, window size=55, pca_dim=4)

Segment into 7+6 classes (because there are 7 textures + 6 boundaries, so I tried this)

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

Segmentation (square, dim=25, k=13, window size=15, pca_dim=12)	Segmentation (square, dim=25, k=13, window size=23, pca_dim=12)	Segmentation (square, dim=25, k=13, window size=39, pca_dim=12)	Segmentation (square, dim=25, k=13, window size=47, pca_dim=12)	Segmentation (square, dim=25, k=13, window size=55, pca_dim=12)
				
Segmentation (square, dim=25, k=13, window size=15, pca_dim=8)	Segmentation (square, dim=25, k=13, window size=23, pca_dim=8)	Segmentation (square, dim=25, k=13, window size=39, pca_dim=8)	Segmentation (square, dim=25, k=13, window size=47, pca_dim=8)	Segmentation (square, dim=25, k=13, window size=55, pca_dim=8)
				
Segmentation (square, dim=25, k=13, window size=15, pca_dim=4)	Segmentation (square, dim=25, k=13, window size=23, pca_dim=4)	Segmentation (square, dim=25, k=13, window size=39, pca_dim=4)	Segmentation (square, dim=25, k=13, window size=47, pca_dim=4)	Segmentation (square, dim=25, k=13, window size=55, pca_dim=4)

IV. Discussion

[Problem 1 (a) 2] Which feature dimension has the strongest discriminant power? Which has the weakest? Please justify your answer.

Theoretically speaking the feature dimension with minimal standard deviation has the strongest discriminant power, because that makes data out of the cluster more obvious to be identify. And oppositely, the feature dimension with maximal standard deviation has the weakest discriminant power, because all the data in that cluster already spread among large special area, if there is another data near that area, it is hard to tell whether it belongs to that cluster. I calculated the standard deviations of each dimension through 12 samples and the results are shown above., however I do not think it is the optimal place to calculate standard deviations which is requested by the homework problem, because this is done before clustering 4 kind of textures and it seems meaningless. I think it will be meaningful if I calculate them with knowing truth of classification and then calculate the standard deviation of the average feature vector class by class. And then find the weakest and the strongest dimension for each class.

[Problem 1 (a) 4] Discuss the effectiveness of feature dimension reduction over K-means. Report your results and compare them with the reality (by eyes).

If I run k-means without multiple trails, the classification result will be different, all the results are listed above. Some gives very good correct rates, and some are bad. The texture10 is hard to classify, all the results with a high correct rate can not classify it to the truth cluster. The texture of that image is straw, and it is quite different from other two straw textures. Actually, it seems that other two straw textures (texture2 and texture8) are derivate from the same image. Same to texture3 and texture9, texture4 and texture6, texture5 and texture7. These pairs seem to be easy to classify into a same cluster in all results. Others are relatively difficult to give a right classification.

However, after multiple trails of k-means, the final result is stable. Although it can not give the best result comparing with reality in all possible results, the returned result tends to have a high correct rate. A stable result is also welcome than a lottery result. The only disadvantage of this improvement is it takes a little bit longer time. (The program cannot return the result with the highest correct rate, since the truth is unknown to the program, thus computing correct rates is impossible.)

The Combining Symmetric Pairs procedure seems useless for this problem and always give worse results although it is common to use with laws filters. Maybe that is because the orientation of features is important for discriminating feature, since some texture images are derivate from a same source and no rotation is performed.

PCA gives a better result, so it is useful. It mitigates interference from unrelated features and improves the accuracy of feature resolution.

Texture Segmentation's Window Size and Speed

None of attempts gives a correct segmentation, although the texture segmentation is clear by eyes. That means the raw Laws Energy Approach is not a powerful approach. However, with larger and larger window size for each pixel, the segmentation result seems cleaner and cleaner. That is because with larger window size, it takes larger patterns in texture into consideration, such as cracks of the brick texture. So, the window size should not be too small. While, the time consumption increases largely, so the window size should not be too large. And also, too large window size may cover several regions of textures and make it hard to segment.

Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

Advanced Texture Segmentation Techniques

PCA gives better (cleaner) results, but not correct results also. The new dimension after PCA can not be too small, it is better to determine by knowing about the ordered singular values from SVD procedure.

Although small holes can be filled by using flood-fill and assign pixels to their nearest neighbor's color, but I do not think it is good, because you can do that by knowing there is no small regions for textures in advance. I think making that assumption is not applicable in all situations.

Trails about increasing the K for k-means from 7 to 13 does not give clearer results, but it tends to no longer classify two different large regions of textures into a same class. However, these results are not convincing me as good results.

Problem 2: Image Feature Extractor

I. Abstract and Motivation

Feature extraction can represent the image local information using fewer dimensional. And these features can be easily comparing with others, to find whether there is a good match. With a good designed feature extractor, extracted features can also be matched even if images are geometric transformed. So that this technique is commonly used in industry to identify objects in digital images.

II. Approach and Procedures

Scale Invariant Feature Transform (SIFT)

SIFT [2] is an advanced feature extractor. It can give robust feature representation "invariant to image scaling, translation and rotation, and partially invariant to illumination changes and affine or 3D projection." [2] It uses different sizes of scaled image to build a pyramid and this pyramid helps it to extract significant features in all sampling rates. So that it is invariant to image scaling. And it extracts features by using only local information, so it is invariant to image translation. Then it records the direction of local gradient, so it is invariant to image rotation. The feature comparation is not precise comparation that make it have some tolerance for illumination changes and affine or 3D projection. I will not go over all the procedures of SIFT here.

SIFT is a patented algorithm. Even OpenCV implemented SIFT, but it is unavailable in public releases because of latent legal risk. The only common way to use SIFT in OpenCV is to compile it from scratch and link the library to my project, because not only you need to enable the *contrib* library, but also compile with NON_FREE compiling argument enabled. however, it will be hard to grade if I use C++. So, I use Python with pre-compiled wheel file with command "pip install opencv-contrib-python =3.4.2.16". The version should not be very new, because the newer versions of public releases does not enable the NON_FREE compiling argument.

Feature Extraction

In OpenCV, key point information is stored in a dedicate data structure which can provide key point's coordinate, angle, size, response, octave information. The feature of a key point is represented by a 128-dimension vector and this vector is stored separately.

There are some arguments for SIFT for detecting key points, I use the default values provided by OpenCV since they works well.

Matching Features

After I use SIFT extracted features in two images, I need to match them to find good matches. With at least 3 matches, we can find the relationship between two images. To find matches, I use brute-force matcher (BFMatcher) provided by OpenCV. The BFMatcher is not fast but precise. It works very simple, for each feature in the first image, it iterates through all the features in the second image to see whether they are close to each other. The distance between two feature vectors is defined by a distance function, for this problem the Euclidean distance (L2-norm) is suitable. There is another argument called Cross-check, by enabling this, only if the feature in the first image is the best match of the matched feature in the second image among all features in the first image will result in a final match.

There are two matching functions, the most intuitive one is the default match function only returns the best matched feature. The other one is Knn match function, it returns top K best matches for each feature. Commonly, K is set to 2. If the distance of these two matches is close, it indicates that this is a bad match and should be discarded. I use the default matcher in my program. The matching results is stored in a dedicate data structure, and the structure has a distance field indicating the distance of two feature vectors.

Image Matching

The problem asks me to find the largest scale in river image 1, the scale here is defined as the L2-norm of a feature vector.

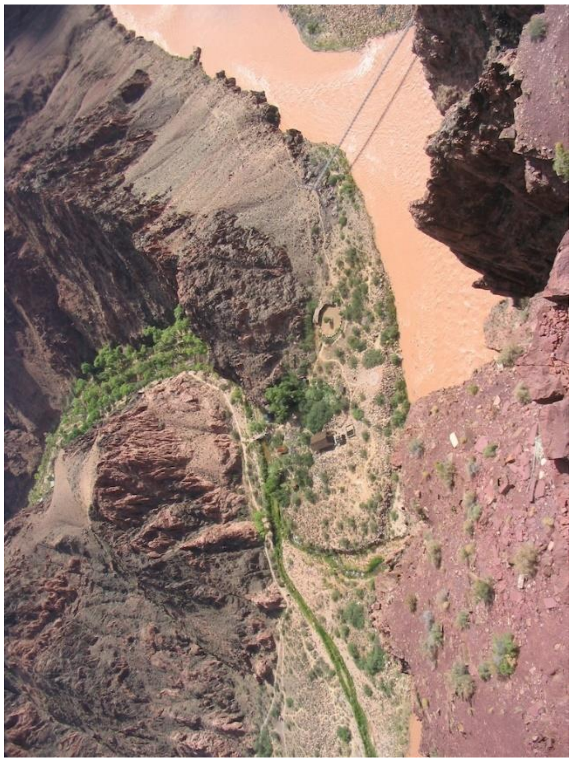
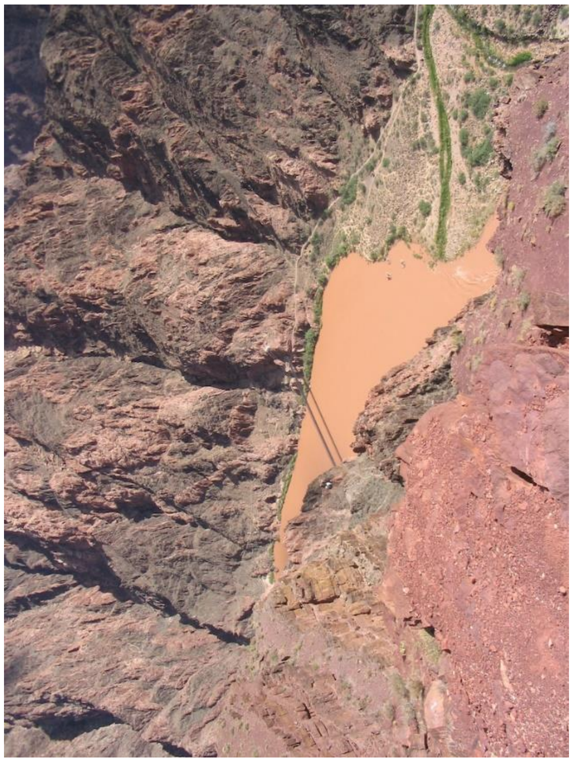
After I get all the key points in river image 1, then I pick the key point having the largest scale. And then use this key point to find the nearest key-point in river image 2. The distance here is defined as the L2-norm of two feature vectors. I found them and printed their information given from OpenCV.

Bag of Visual Words (BoVW)

The algorithm Bag of Visual Words (BoVW) is inspired by Bag of Words (BoW) in Natural Language Processing (NLP) area. However, it is easy to extract words in plain text and hard to extract "words" in an image. Here, features serve as words' role in BoW. So, for BoVW, a feature extractor is indispensable. To build a bag of features, we need an image database with labeled images. However, this homework problem is quite different, it requires unsupervised machine learning, so labels of images are not needed. I first collect all features from all images in the dataset together. Here, features are extracted by using SIFT. Then clustering these features into 2 clusters using k-means. The k-means runs multiple time to return a result with minimal total distance. After I have the coordinates of the centers of 2 clusters, I can then calculate the clustering histogram of the features in each image. Additionally, images have different numbers of key points, so I normalized bins in each histogram by dividing the number of key points in that image. However, this make the histograms acting like probability distribution graphs, but maybe more reasonable. These are steps for building a bag of visual words. Then we have a test image. Extract its features and cluster these features using same cluster centers and draw a histogram. We may know something by comparing this histogram and histograms of the training data.

III. Experimental Results

Image Matching

	
River image 1 (river1.raw)	River image 2 (river2.raw)

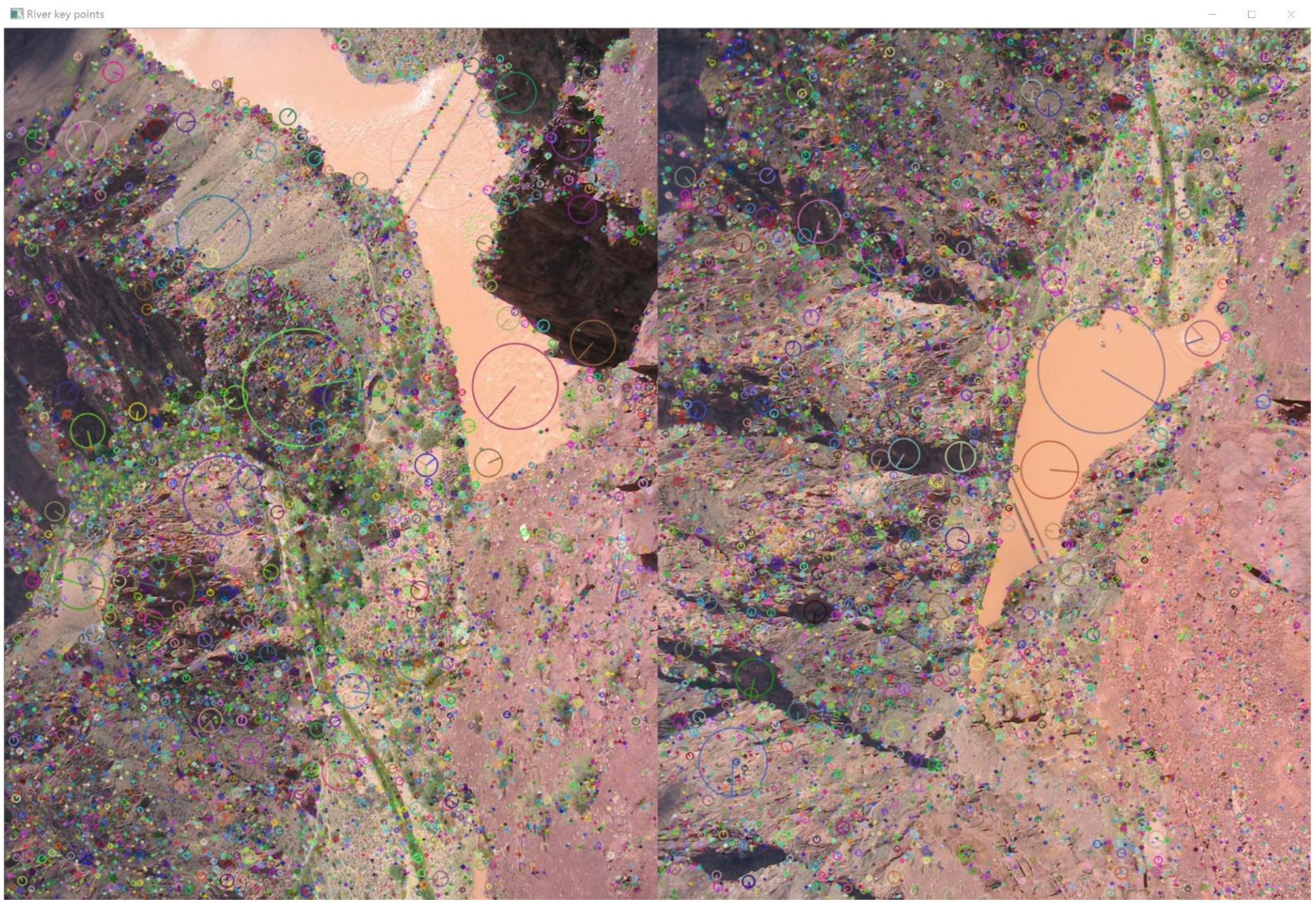
Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019



Key points in river image 1 & 2	
[pt=(330.8982849121094, 913.8211059570312), size=4.586584, angle=37.770386, response=0.027991, octacv=983552]	[pt=(477.3910217285156, 163.93267822265625), size=4.582687, angle=55.971649, response=0.027413, octacv=918016]
Key point with the largest scale in river image 1	closest neighboring key-point in river image 2

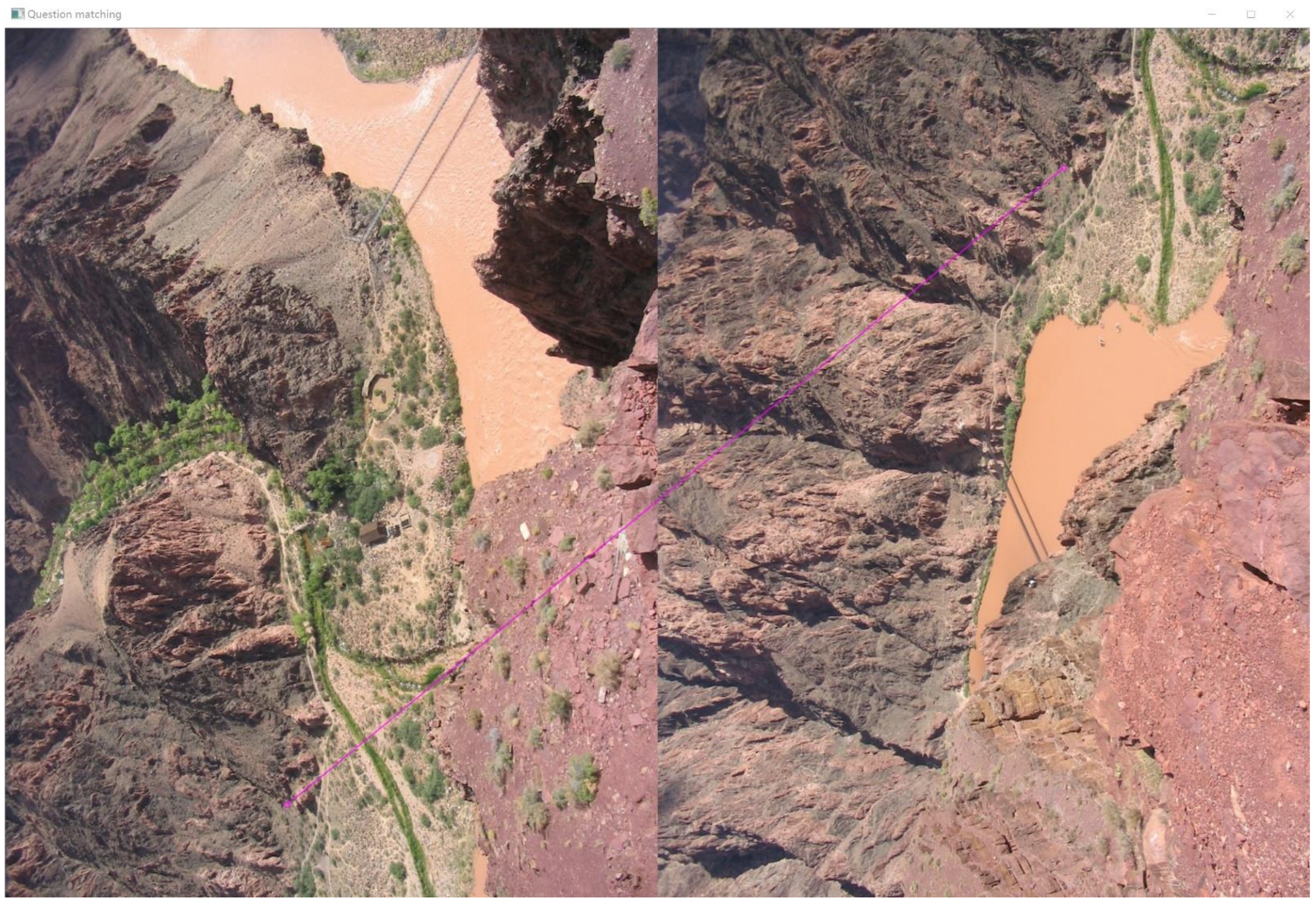
Zongjian Li

6503-3789-43

zongjian@usc.edu

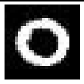
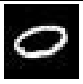
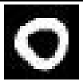
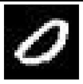
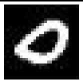
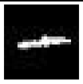
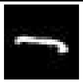
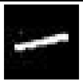
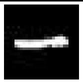
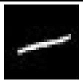
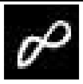
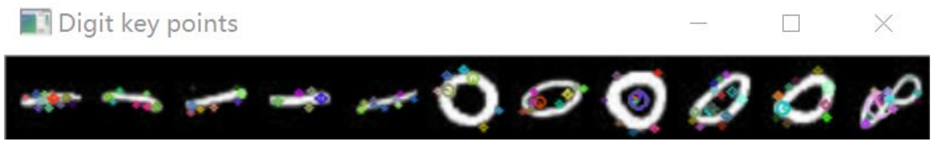
EE 569 Homework #4

March 19th, 2019

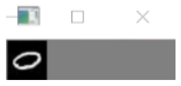
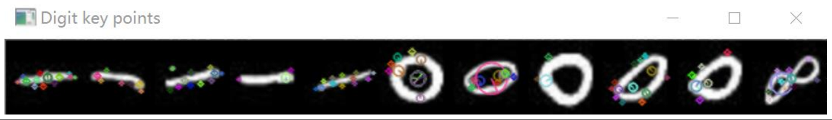


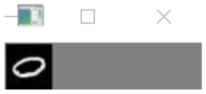
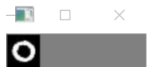
Matching result

Bag of Words

										
zero-1	zero-2	zero-3	zero-4	zero-5	one-1	one-2	one-3	one-4	one-5	eight
Key points (f=1.5)										

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

Cluster 1 center (f=1.5)	[1.491 2.86 0.772 2.702 2.807 0.123 0.246 1.14 60.018 8.474 1.228 0.895 0.456 1.526 0.825 17.684 119.14 9.316 0.263 2.509 17.649 8.596 11.772 85.684 18.614 1.439 4.702 35.123 78.684 19.526 22.684 43.439 2.053 1.877 0.316 1.789 4.614 0.263 0.351 1.053 116.632 11.018 0.474 0.298 0.561 0.105 0.14 12.754 146.754 19.632 0.754 5.684 30.368 7.088 3.895 51.825 22. 2.158 7.158 49.491 130.596 25.298 9. 11.965 2.316 0.93 0.298 0.368 4.614 0.965 0.211 1.088 118.491 12.825 0.088 0.07 0.544 0.14 0.281 9.86 146.93 55.509 3.632 7.263 30.228 5.386 0.789 17.491 23. 11. 5.86 28.614 130.737 48. 4.228 1.912 1.842 0.789 0.263 0.07 3.754 2.281 0.105 0.86 59.526 18.86 0.912 0.088 0.526 0.877 0.298 6.649 117.912 86.667 15.228 9.193 17.053 2. 0.281 7.211 21.193 44. 15.632 22.912 78.807 34.842 3.544 0.596]
Cluster 2 center (f=1.5)	[20.25 10.125 4.018 4.054 5.536 2.375 4.054 9.75 69.232 19.911 3.893 2.571 14.857 14.768 22.446 51.25 23.179 10.071 3.75 18.875 68.768 42.107 31.857 25.857 5.161 6. 8.839 10. 30.232 19.482 12.732 4.929 46.5 8.911 3.732 3.804 7.768 7.143 3.946 20.357 118.054 24.054 9.571 15.893 28.464 16.571 21.893 67.411 44.786 18.857 11.964 35.089 111.625 58.125 27.964 22.286 10.286 7.464 10.696 15.875 59.554 24.143 8.304 9.071 47.607 19.518 4.446 3.911 8.304 8.161 7.196 10.911 115.304 63.982 24.786 20.839 27.625 17.482 14.089 20.446 45.964 19.821 26.821 59.161 108.946 39.107 13.357 19.232 8.232 4.643 10.036 33.018 58.464 13.589 9.607 9.179 19.036 12.321 9. 2.714 7.25 4. 1.375 3.321 73.143 54.411 29.696 11.75 13.018 2.393 1.857 11.321 25.607 28.518 35.589 38.857 64.071 22.161 4.196 6.357 2.607 3.232 11.232 20.821 29.464 13.393 8.036 5.839]
Histograms (f=1.5, unnormalized)	[[7, 6], [6, 3], [5, 4], [3, 4], [6, 6], [6, 3], [3, 11], [6, 5], [7, 10], [8, 4]]
Eight Histogram (f=1.5, unnormalized)	[4, 15]
Nearest Histogram (f=1.5, unnormalized)	[3, 11]
Nearest Image (f=1.5, unnormalized)	
Key points (f=2.0)	
Histograms (f=1.5, normalized)	[[0.5384615384615384, 0.46153846153846156], [0.6666666666666666, 0.3333333333333333], [0.5555555555555556, 0.4444444444444444], [0.42857142857142855, 0.5714285714285714], [0.5, 0.5], [0.6666666666666666, 0.3333333333333333], [0.21428571428571427, 0.7857142857142857], [0.5454545454545454, 0.45454545454545453], [0.4117647058823529, 0.5882352941176471], [0.6666666666666666, 0.3333333333333333]]

Eight Histogram (f=1.5, normalized)	[0.21052631578947367, 0.7894736842105263]
Nearest Histogram (f=1.5, normalized)	[0.21428571428571427, 0.7857142857142857]
Nearest Image (f=1.5, normalized)	
Key points (f=2.0)	[1.222 0.463 0.333 0.167 0.111 0.204 0.315 0.704 52.444 2.241 0.074 0.093 0.093 0.074 0.926 17.778 107.222 3.944 0.037 1.63 14.556 5.574 20.185 87.037 20.778 1.556 3.704 25.907 77.204 20.389 36.907 42.407 1.815 1. 0.426 0.241 0.148 0.222 0.37 1.019 110.852 7.648 0.074 0.056 0.093 0.074 0.315 13.944 149.722 17.481 0.685 4.481 26.037 5.148 12.074 66.537 25.722 4.963 10.444 37.463 115.704 25.296 26.019 29.389 1.741 1.037 0.463 0.185 0.056 0.056 0.352 1.056 109.37 14.259 0.259 0.019 0.019 0.056 0.167 6.648 149.722 72.278 12.907 5.87 26.519 3.981 0.463 15.5 25.093 28.833 26.667 30.852 114.148 40.259 5.074 4.481 1.426 0.611 0.296 0.037 0.019 0.037 0.296 0.981 52.556 14.611 0.722 0.148 0.148 0.13 0.5 4.185 104.167 80.259 18.185 5.037 15.204 1.944 0.222 5.296 18.593 44.407 31.759 18.833 74.407 26.407 2.537 0.667]
Cluster 2 center (f=2.0)	[19.908 9.031 5.4 2.862 7.523 3.462 4.4 13.215 60.908 23.769 7.462 2.846 13.492 12.108 25.677 45.138 29.523 19.015 6.031 16.308 61.262 42.323 31.308 18.615 5.769 10.877 8.954 9.185 38.246 22.123 9.954 3.569 45.369 10.631 2.938 5.077 8.292 9.231 5.231 23.708 105.031 23.6 12.492 17.185 25.077 20.154 27.385 68.369 49.231 21.077 11.862 31.2 99.446 57.262 28.908 22.215 9.646 12.754 8.923 15.246 66.108 25.862 7.077 7.062 45.569 24.262 5.185 8.262 9.308 8.185 5.692 10.231 104.077 67.754 28.769 21.385 24.631 18.354 14.308 22.169 46. 19.354 30.815 64.354 99.662 31.477 12.4 22.662 10.615 6.215 7.477 31.015 61.554 12.477 6.677 7.877 18.646 13.538 6.723 4.138 9.246 5.908 4.2 4.6 61.538 49.538 29.4 10.677 11.723 5.815 5.185 14.954 25.754 22.831 35.062 43.354 53.831 19.292 4.954 11.046 5.877 2.738 8.692 21.692 29.831 12.308 6.862 6.031]
Histograms (f=2.0, unnormalized)	[[9, 9], [4, 5], [9, 9], [2, 2], [10, 6], [4, 7], [0, 15], [2, 1], [10, 7], [4, 4]]
Eight Histogram (f=2.0, unnormalized)	[4, 11]
Nearest Histogram (f=2.0, unnormalized)	[4, 7]
Nearest Image (f=2.0, unnormalized)	

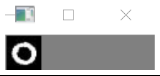
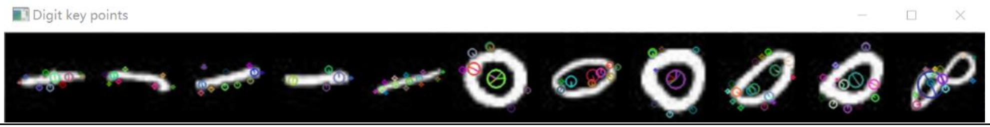
Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

Histograms (f=2.0, normalized)	[[0.5, 0.5], [0.4444444444444444, 0.5555555555555556], [0.5, 0.5], [0.5, 0.5], [0.625, 0.375], [0.36363636363636365, 0.6363636363636364], [0.0, 1.0], [0.6666666666666666, 0.3333333333333333], [0.5882352941176471, 0.4117647058823529], [0.5, 0.5]]
Eight Histogram (f=2.0, normalized)	[0.26666666666666666, 0.7333333333333333]
Nearest Histogram (f=2.0, normalized)	[0.36363636363636365, 0.6363636363636364]
Nearest Image (f=2.0, normalized)	
Key points (f=3.0)	 Digit key points
Cluster 1 center (f=3.0)	[1.239 1.451 2.394 0.662 0.169 0.254 0.394 0.789 49.915 4.127 0.775 0.07 0.07 0.141 0.718 19.831 107.366 4. 0.085 1.423 10.732 3.887 18.93 96.69 22.803 1. 3.155 26.394 62.958 18.225 37.958 51.423 2.662 1.732 1.535 0.493 0.127 0.197 0.366 0.915 113.732 8.62 0.296 0. 0.042 0.042 0.169 14.817 149.239 16.042 0.493 4.113 20.38 3.69 8.958 68.986 30.662 3.901 8.479 44.859 108.451 21.732 23.549 29.732 3.31 1.141 0.451 0.169 0.099 0.127 0.282 1.113 114.451 15.183 0.155 0.014 0.028 0.028 0.07 7.606 149.423 69.887 7.225 4.282 20.901 3.789 0.31 15.93 32.535 27.592 16.155 24.761 112.761 44.845 4.141 3.775 2.254 0.873 0.225 0.085 0.085 0.155 0.296 0.775 54.028 18.268 0.718 0.056 0.127 0.127 0.085 3.606 111.549 93.155 13.282 3.465 11.296 1.732 0.127 5.465 26.831 54.31 25.155 13.394 65.113 33.085 2.887 0.746]
Cluster 2 center (f=3.0)	[18.258 11.561 5.227 6.667 7.288 2.636 4.803 15.591 57.424 21.606 6.939 5.909 13.015 12.682 33.636 50.394 25.667 12.576 5.409 14.667 56.394 46.576 36.561 20.197 2. 4.576 7.288 9.576 32.924 22.848 7.318 3.455 51.106 14.136 3.303 8.561 8.47 6.727 3.258 27.152 105.621 28.545 15.258 23.318 29.091 18.182 27.076 66.5 52.803 28.591 13.591 28.561 99.864 62.712 27.909 17.182 6.379 5.53 4.894 11.318 70.106 31.894 4.227 6.803 51.909 25.318 3.621 6.076 9.379 7.894 8.303 15.182 106.136 67.439 28.985 21.667 28.894 22.909 17.242 27.045 50.848 21.803 31.045 62.409 95. 31.394 15.227 28.712 6.455 2.091 6.394 36.333 68.318 12.697 6.394 7.591 19.061 14.636 8.424 3.106 8.242 5.606 6.591 7.803 53.045 49.515 36.909 12.97 11.348 5.924 5.864 15.773 22.773 20.727 43.258 47.894 48.5 12.864 2.803 9.621 3.591 2.348 10.697 28.591 29.061 8.697 1.955 3.394]
Histograms (f=3.0, unnormalized)	[[7, 5], [6, 4], [8, 8], [4, 3], [10, 8], [6, 6], [1, 10], [8, 6], [10, 9], [11, 7]]
Eight Histogram (f=3.0, unnormalized)	[8, 14]
Nearest Histogram (f=3.0, unnormalized)	[8, 8]

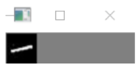
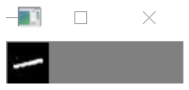
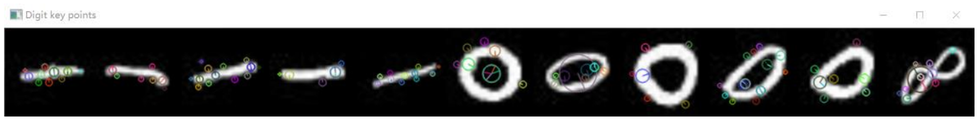
Zongjian Li

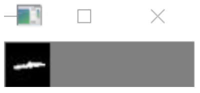
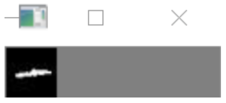
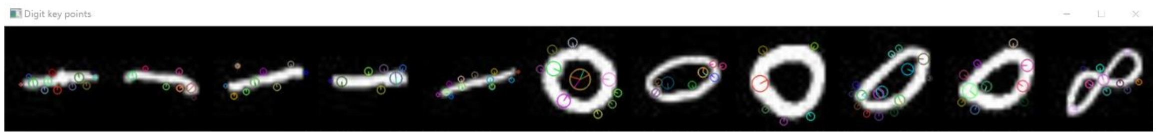
6503-3789-43

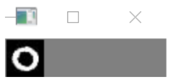
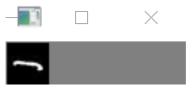
zongjian@usc.edu

EE 569 Homework #4

March 19th, 2019

Nearest Image (f=3.0, unnormalized)	
Histograms (f=3.0, normalized)	[[0.5833333333333334, 0.4166666666666667], [0.6, 0.4], [0.5, 0.5], [0.5714285714285714, 0.42857142857142855], [0.5555555555555556, 0.4444444444444444], [0.5, 0.5], [0.09090909090909091, 0.9090909090909091], [0.5714285714285714, 0.42857142857142855], [0.5263157894736842, 0.47368421052631576], [0.6111111111111112, 0.3888888888888889]]
Eight Histogram (f=3.0, normalized)	[0.36363636363636365, 0.6363636363636364]
Nearest Histogram (f=3.0, normalized)	[0.5, 0.5]
Nearest Image (f=3.0, normalized)	
Key points (f=4.0)	
Cluster 1 center (f=4.0)	[20.852 8.197 2.541 2.918 6.869 4.377 1.656 11.328 67.492 21.164 3.77 3.115 14.508 11.164 22.59 48.213 30.049 12.066 2.426 16.23 66.656 41.016 31.721 22.082 2.23 2.885 4.197 9.246 39.049 21.361 11. 2.82 47.033 10.246 0.705 3.098 7.77 8.869 3.213 21.82 114.033 21.738 9.623 16.951 26.77 19.541 24.23 70.279 51.344 19.738 11.41 32.508 107.77 55.41 27.672 23.295 5.803 6.902 8.689 13.508 67.689 24.393 7.377 5.541 49.049 23.18 3.279 5.656 8.574 7.787 4.672 9.869 113.131 67.754 24.967 21.115 27.016 18.803 12.787 21.164 48.885 19.803 27.18 62.377 107.951 35.672 12.279 21.934 5.475 4.344 6.918 29.656 64.344 13.213 7.574 6.574 21.508 13.41 5.59 2.262 9.295 5.672 3.738 2.672 68.836 53.541 27. 9.311 12.902 5.984 5.098 11.967 27.787 25.049 33.689 41.393 59.77 20.852 4.525 8.738 1.59 1.508 9.721 22.754 34.098 12.918 5.607 2.984]
Cluster 2 center (f=4.0)	[1.192 0.423 0.288 0.135 0.077 0.154 0.192 0.769 50.865 1.942 0.077 0.019 0.019 0.038 0.558 19.75 110.231 3.558 0.019 1.75 13.481 4.519 17.846 96.692 21.769 1.635 4.481 31.365 69.885 17.019 32.654 50.462 1.885 0.942 0.404 0.135 0.077 0.135 0.346 1.038 113.538 7.154 0.058 0.019 0.019 0.077 0.25 14.173 148.731 16.904 0.731 5.231 25.654 4.385 9.885 65.846 25.712 4.308 10.481 47.981 117.558 22.731 20.558 24.423 1.962 1.038 0.404 0.135 0.058 0.077 0.269 0.981 112.404 14.635 0.231 0.077 0.019 0.058 0.173 7.962 148.731 69.577 10.5 7.077 26.154 4.5 0.462 15.788 24.442 24.327 20.269 33.558 119.635 43.096 3.75 2.846 1.558 0.577 0.212 0.019 0.019 0.058 0.192 0.904 52.442 15.923 0.673 0.115 0.058 0.096 0.327 6.385 106.5 85.923 18.077 6.865 15.385 1.769 0.25 6.269 19.808 47.115 25.327 21.942 75.558 28.788 1.5 0.462]
Histograms (f=4.0, unnormalized)	[[9, 6], [5, 4], [6, 9], [3, 3], [7, 8], [6, 4], [10, 0], [1, 5], [9, 9], [5, 4]]

Eight Histogram (f=4.0, unnormalized)	[10, 4]
Nearest Histogram (f=4.0, unnormalized)	[9, 6]
Nearest Image (f=4.0, unnormalized)	
Histograms (f=4.0, normalized)	[[0.6, 0.4], [0.5555555555555556, 0.4444444444444444], [0.4, 0.6], [0.5, 0.5], [0.4666666666666667, 0.5333333333333333], [0.6, 0.4], [1.0, 0.0], [0.16666666666666666, 0.8333333333333334], [0.5, 0.5], [0.5555555555555556, 0.4444444444444444]]
Eight Histogram (f=4.0, normalized)	[0.7142857142857143, 0.2857142857142857]
Nearest Histogram (f=4.0, normalized)	[0.6, 0.4]
Nearest Image (f=4.0, normalized)	
Key points (f=5.0)	
Cluster 1 center (f=5.0)	[1.177 0.419 0.258 0.177 0.177 0.242 0.274 0.774 48.726 2.242 0.097 0.065 0.065 0.113 0.758 18.5 104.919 4.113 0.097 1.871 12.468 3.887 20.274 96.516 20.355 1.177 3.387 30.79 65.806 17.5 38.855 51.339 1.871 1.097 0.484 0.194 0.145 0.145 0.339 0.855 112.177 6.371 0.177 0.016 0.048 0.097 0.21 14.742 148.952 14.806 0.645 4.839 23.871 4.484 11.339 71.726 25.919 4.565 9.903 46.323 115.21 23.145 24.935 28.161 1.823 1.129 0.371 0.177 0.081 0.129 0.258 0.839 112.097 14.919 0.21 0.016 0.016 0.016 0.177 6.565 148.952 71.403 9.839 5.371 23.484 4.79 0.516 13.581 26.726 26.242 21.823 27.339 115.645 49.323 4.758 3.532 1.387 0.694 0.194 0.032 0.016 0.065 0.274 0.887 50.387 16.871 0.597 0.129 0.145 0.129 0.339 5.242 107.71 91.032 15.694 4.5 12.548 1.742 0.258 7.081 22.5 51.081 27.532 15.032 69.016 34.419 2.532 0.839]

Cluster 2 center (f=5.0)	[17.593 11.78 3.644 6.678 9.542 4.542 1.695 9.949 61.305 25.153 6.458 5.932 11.78 12.305 22.475 48.339 27.475 14.983 6.61 21.186 56.864 39.542 35.203 26.237 3.203 6.983 11.983 12.695 34.661 23.746 16. 5.593 41.695 10.678 0.881 6.458 8.475 9.169 3.39 21.424 107.542 24.22 9.966 16.407 22.475 17.237 26.576 68.254 50. 21.288 13.119 35.695 99.424 55.559 33.085 24.186 8.068 9.881 13.542 18.881 62.102 27. 12.814 11.593 41.271 22.22 3.797 6.305 9.102 8.559 4.661 10.169 106.593 73.339 27.627 19.508 21.542 16.695 12.559 20.39 46.644 27.627 34.322 60.373 94.424 37.797 13.966 21.644 9.424 5.525 13.339 35.915 59.305 17.695 13.39 11.729 15.237 12.288 5.847 2.898 9.695 6.305 4.458 2.831 54.627 53.085 28.864 9.085 10.085 5.864 2.814 11.949 24.373 33.373 42.508 37.932 45.373 20.034 4.119 9.203 4.746 5.051 17.932 24.831 27.068 13.847 8.831 6.322]
Histograms (f=5.0, unnormalized)	[[8, 7], [4, 5], [7, 6], [5, 3], [10, 5], [6, 7], [1, 9], [5, 1], [10, 10], [6, 6]]
Eight Histogram (f=5.0, unnormalized)	[5, 9]
Nearest Histogram (f=5.0, unnormalized)	[6, 7]
Nearest Image (f=5.0, unnormalized)	
Histograms (f=5.0, normalized)	[[0.5333333333333333, 0.4666666666666667], [0.4444444444444444, 0.5555555555555556], [0.5384615384615384, 0.46153846153846156], [0.625, 0.375], [0.6666666666666666, 0.3333333333333333], [0.46153846153846156, 0.5384615384615384], [0.1, 0.9], [0.8333333333333334, 0.16666666666666666], [0.5, 0.5], [0.5, 0.5]]
Eight Histogram (f=5.0, normalized)	[0.35714285714285715, 0.6428571428571429]
Nearest Histogram (f=5.0, normalized)	[0.4444444444444444, 0.5555555555555556]
Nearest Image (f=5.0, normalized)	

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #4
March 19th, 2019

IV. Discussion

[Problem 2 (a) I] From the paper abstract, the SIFT is robust to what geometric modifications?

Invariant to image scaling, translation, and rotation.

Partially invariant to illumination changes and affine or 3D projection.

It is also robust to noise images, however, adding noise is not a kind of geometric modification.

[Problem 2 (a) II] How does SIFT achieves its robustness to each of them?

Scaling: Using Gaussian kernel as smoothing kernel, which is good for scale space analysis. Building an image pyramid with different scale of image, and only key points with significant value in all levels of the pyramid is kept.

Translation: Key points encode its local region gradient information into a vector, all the further computation is based on this vector, so the position of this feature does not affect the results of the further calculation. Even the key point is moved by translation operation, the local gradient information will not change.

Rotation: Only key points with significant value in all levels of the pyramid is kept, so the key point is significant. The feature records one or more units of gradient orientation information. The orientation of small local regions is assigned to one of 36 bins by Gaussian-weighted average to build a histogram. The main orientation information is computed by selecting the majority of that histogram, which is more robust. With these orientation information, rotated features can also compare with original features and compute the rotation angle. Some key points may have multiple main orientations and can be considered as multiple features with same center and radius but different orientations, which can increase the matching accuracy. When encoding features,

Illumination: See next problem's answer.

Affine/3D projection and noise: SIFT is invariant to these situations is because SIFT has certain fault tolerance ability that can adapt small shifts. When SIFT encodes local gradient information in to a vector, it "use a bin size of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times the maximum model dimension for location.". So, we can see the segmentations has large step widths, these border bin sizes give it a good fault tolerance ability.

[Problem 2 (a) III] How does SIFT enhances its robustness to illumination change?

First, it uses gradient magnitude to encode features, this provides the basic ability of robustness to illumination change. Second, the robustness is enhanced by applying a threshold to the gradient magnitudes of each pixel. The threshold is defined by the 0.1 times the maximum possible gradient value. Gradient magnitude values higher than this threshold will be clipped to that threshold. So, the effect of illumination change is bounded. And, gradient orientation is not influenced by illumination change much.

[Problem 2 (a) IV] What are the advantages that SIFT uses difference of Gaussians (DoG) instead of Laplacian of Gaussians (LoG)?

The Gaussian smooth kernel and its derivatives is good for scale space analysis. DoG approximates LoG, and it needs less computation. So, using DoG speeds up the algorithm.

[Problem 2 (a) V] What is the SIFT's output vector size in its original paper?

8*4*4+8*2*2=160 elements. Two scale, 4*4 and 2*2 grid of locations on region with a radius of 8 pixels from the center. Each grid encodes 8 orientations.

[Problem 2 (b)] … Discuss your results, esp, the orientation of each key-points.

There are thousands of key points founded in both images, however it is not difficult to find the pair of key points requested by the homework problem. The attributes of these two key points are shown above. The matched local regions are the same region, which can be justified by eyes. However, two images are not the same, so that means the SIFT works for this case. Two key points have similar size, that means two images have similar scaling factors. And the key point in image 2 has basically the same but a little larger angle than that in image 1. I can see from eyes that the region in image2 rotates clockwise a little bit, so the difference of angles is reasonable. Note that the SIFT calculate feature vectors after rotating counter clockwise of key points' angles, so these angles are not the gradient orientation of that in 128D (OpenCV) feature vectors. Since they are the nearest pair of key points, their feature vectors must be similar, and I do not need to justify this by showing their lengthy feature vectors.

[Problem 2 (c)] … Discuss your observation.

First, for different scaling factors, the results are different. That is because the parameters of SIFT extractors in different trails are the same. So, it focusses on the same size of viewport, when image scale up, some small patterns become larger and finally become big solid color blocks that makes the feature extraction results be different.

Second, by using BoVW, two clusters learnt by k-means from 10 training images itself even cannot give good result to classify these training data, not to mention the digit eight image. Both unnormalized and normalized attempts do not work. In my imagine, feature for zero images representing holes and arcs, and feature for one images representing edges and toppings. However, after zooming in the images, the extractor finds more smooth edge features than characteristic features. Non-characteristic features dominate the decision, so this approach does not work well. While, I must zoom in, otherwise some images will be too small to extract any key points.

It is hard to say whether the eight digit image is close to one or zero, my result tells me that for small scaling factor like 1.5 and 1, it is close to zero which is intuitive since digit eight is built by two connected zeros. For larger scaling factors, the result is vacillating. The reason may be the eight is zoomed in a lot and contains more smooth arcs which make it a little bit closer to ones.

References

[1] K.I. Laws. Textured Image Segmentation. PhD thesis, University of Southern California, LA, 1980.

[2] Lowe, David G. "Object recognition from local scale-invariant features." iccv. Ieee, 1999.