

Name: Kuimu Ren  
USC ID Number:1473482531  
USC Email:kuimuren@usc.edu  
Submission Date: 03/28/2024

## Problem1

### Texture Analysis

#### Abstract and Motivation:

Spatial Texture analysis is a fundamental research area in the fields of image processing and computer vision, aimed at describing and understanding the texture features present in images. Texture typically refers to the repetitive and structured patterns found in surfaces.

#### Approach and Procedures:

First, we need to set up the Laws filter. Laws' filters are a set of 1D convolution kernels designed for texture analysis. These filters are used to capture various textural properties such as level, edge, spot, wave, and ripple in an image. The Laws' filters are constructed by combining five 1D kernels, each representing a different textural property:

L5:  $L5 = [1, 4, 6, 4, 1]$  Level filter, emphasizing areas with constant intensity.

E5:  $E5 = [-1, -2, 0, 2, 1]$  Edge filter, highlighting edges or sharp transitions in intensity.

S5:  $S5 = [-1, 0, 2, 0, -1]$  Spot filter, detecting small regions or spots with intensity variations.

W5:  $W5 = [-1, 2, 0, -2, 1]$  Wave filter, capturing repetitive patterns or wave-like structures.

R5:  $R5 = [1, -4, 6, -4, 1]$  Ripple filter, identifying repetitive ripples or oscillations in intensity.

Second, extend the image boundary with 2 pixels width for each edge by reflection and subtract image mean to reduce illumination effects. Then, use the 2D filter obtained above, convolute the input image and get a filtered image. Calculate the square sum as the representation of the energy using the following equation.

$$average\ energy = \sqrt{\frac{1}{MN} \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} I(i, j)^2}$$

M represents the height and N represents the width of the image. "Square sum" represents the texture property obtained by each filter.

Third, we can use PCA to reduce the dimensionality of the feature vectors from 25 to 3. PCA is a common technique used for dimensionality reduction. It accomplishes this by linearly transforming high-dimensional data into a lower-dimensional space while preserving the maximum variance in the data. Through dimensionality reduction, complexity and storage space of the data can be reduced, facilitating easier visualization and extraction of key features.

Fourth, using the nearest neighbor rule based on the Mahalanobis distance to realize texture classification. We need to do the same step for our test cases fist. Then, compute the mean vector and covariance matrix for each class in the training dataset. We iterate over each class, extract the features of samples belonging to that class, compute the mean vector and covariance matrix,

and store them separately. The Mahalanobis distance between the test feature vector and the class mean vector is then calculated using the corresponding covariance matrix. Finally, the nearest neighbor method is used to classify the test image based on Mahalanobis distance. For each test feature vector, it computes the Mahalanobis distance to each class mean vector and selects the class with the smallest distance as the predicted label for that test sample

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}$$

$$S = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})' (X_i - \bar{X})$$

D is Mahalanobis Distance and S is Covariance Matrix

Last, we can also use Unsupervised Method(K-means clustering) and Supervised Method(Support Vector Machine) to classify the 25-D features and 3-D features of the image.

## Experimental Results:

Shown below are the results of Problem 1:

```
image0: [1638.8458, 377.99118, 305.64099, 399.02979, 811.9281, 434.89185, 132.72066, 136.84881, 212.28074, 430.54227, 366.05682, 142.74715, 165.70465, 267.89413, 537.9519, 456.99402, 191.46107, 277.33392, 369.14526, 738.40802, 804.73236, 298.34091, 337.26
969, 537.2782, 1078.6328]
image1: [1647.2869, 663.89288, 482.48773, 616.16357, 1580.907, 1591.4648, 264.77112, 194.92691, 245.28789, 685.49687, 466.48086, 217.97633, 161.11975, 198.96526, 482.39175, 952.12781, 298.45117, 228.21181, 267.34174, 638.71827, 2441.3774, 828.39948, 682.5
274, 723.35974, 1706.8375]
image2: [1872.1382, 1586.4425, 615.80949, 673.49586, 886.8244, 1777.8744, 447.93472, 247.33231, 217.26851, 382.856, 1833.1525, 278.70673, 154.14139, 141.11286, 207.24025, 952.15784, 255.75711, 158.3269, 143.83988, 222.39187, 1353.6953, 381.78366, 235.1772
3, 235.68463, 411.56573]
image3: [12371.641, 1569.859, 773.28485, 778.83295, 1740.2786, 2148.6565, 438.74582, 224.70964, 214.82194, 471.55786, 1095.8385, 267.17499, 143.53855, 132.99658, 285.61707, 1083.6997, 277.36569, 152.44521, 138.42879, 283.8598, 2364.7478, 553.59845, 295.53
659, 267.92337, 525.99518]
image4: [13554.315, 1814.8826, 846.12689, 1121.3663, 2320.7119, 924.42944, 363.17767, 383.06388, 593.57379, 1190.1599, 895.36823, 412.18494, 475.96189, 768.14423, 1502.0875, 1203.2837, 563.92883, 668.88795, 1805.9211, 2079.8879, 2214.2649, 897.58459, 1802.
9659, 1573.2531, 3128.28]
image5: [1698.4131, 1285.8752, 729.56787, 827.40149, 1979.3815, 1779.952, 327.61978, 228.77069, 264.55627, 662.95319, 992.45331, 232.47264, 168.01871, 185.98073, 448.84761, 1867.5768, 288.98425, 195.44821, 216.48993, 499.76218, 2583.5485, 711.88788, 464.6
115, 487.43534, 1129.2842]
image6: [16691.7637, 1482.8901, 789.22449, 684.24286, 970.56769, 1797.9387, 468.83331, 262.27316, 228.88896, 324.74527, 1067.7882, 382.54587, 175.08658, 157.08929, 229.92984, 991.67426, 297.38585, 177.82916, 166.05429, 256.64151, 1471.4901, 468.91883, 295.
57687, 291.18488, 489.29968]
image7: [112286.055, 1633.8141, 696.72101, 624.56366, 1287.1722, 1574.7179, 373.82527, 287.33228, 212.27296, 466.3714, 701.31671, 213.73186, 135.86635, 148.63716, 348.55029, 631.84688, 222.28615, 153.92549, 176.76668, 418.93396, 1243.7649, 478.62886, 343.6
2299, 419.33887, 1861.7554]
image8: [14787.6254, 1648.6359, 1009.7929, 1045.8731, 1400.8994, 1765.3547, 425.92886, 236.99425, 285.56682, 332.28868, 1354.5016, 223.05438, 133.17238, 128.00821, 217.76576, 1382.3381, 194.31496, 124.76716, 128.43477, 225.3172, 1318.9562, 321.31238, 208.3
835, 217.03835, 485.48011]
image9: [14724.57, 1889.9786, 523.92725, 549.86334, 1289.5579, 2113.9823, 131.87576, 91.39711, 109.93815, 276.53931, 845.31738, 79.83577, 56.341864, 64.536446, 159.84413, 762.62213, 91.893862, 61.760878, 66.227805, 152.75555, 1478.4663, 212.21236, 134.538
33, 134.2882, 287.27181]
image10: [6963.2676, 1554.1385, 851.44886, 779.28911, 1282.3621, 1575.7943, 433.94987, 256.93423, 247.36723, 481.35458, 981.92383, 265.69559, 164.61731, 164.43877, 274.8624, 846.58964, 259.42682, 166.58223, 178.26544, 291.77292, 1287.9176, 411.64102, 273.
6532, 287.61344, 518.31628]
image11: [11485.888, 2557.9854, 1349.8484, 1342.4428, 2993.8386, 3889.8146, 766.65497, 451.79584, 485.73795, 1126.8946, 1683.3226, 491.18483, 321.85484, 367.84668, 877.91821, 1686.7339, 559.26544, 398.61795, 465.84882, 1138.1382, 3754.3849, 1341.8922, 981
.0255, 1289.7288, 2967.9646]
image12: [4188.2384, 1288.2883, 981.53723, 1014.3465, 1528.8274, 1573.8488, 357.05243, 218.7597, 226.42763, 426.85495, 1180.7028, 216.08284, 143.64867, 153.18253, 278.38338, 1102.3933, 218.2688, 149.69312, 162.01743, 298.31219, 1424.4971, 389.8385, 267.68
755, 295.83397, 566.67851]
image13: [7399.184, 918.88942, 736.42737, 911.16248, 1846.9954, 2683.6188, 196.74986, 133.45333, 158.11548, 286.15889, 1865.7919, 138.01997, 88.953354, 95.330784, 173.3297, 1736.576, 156.76848, 93.818555, 95.925575, 175.60828, 3133.4236, 326.13849, 179.13
2, 378.16771, 343.58595]
image14: [6136.3472, 1471.6783, 861.66223, 845.63825, 1389.8588, 1682.2231, 448.11414, 277.32367, 288.39713, 477.7597, 953.34961, 285.99442, 182.86455, 187.66728, 326.36481, 909.12653, 286.75598, 187.16847, 193.58047, 343.8965, 1413.7732, 468.1925, 314.66
522, 334.58768, 617.79213]
image15: [14833.802, 1786.1899, 636.5517, 544.19342, 1085.6359, 2827.984, 379.6152, 198.92274, 187.98314, 399.7851, 771.42346, 286.97281, 124.28996, 138.55954, 289.14059, 667.56757, 288.74862, 135.71718, 147.13886, 331.78729, 1381.5552, 448.88975, 298.7
2587, 333.1532, 759.93896]
image16: [14523.888, 1385.8792, 828.8236, 987.77612, 1489.175, 1586.8236, 372.54889, 222.78972, 221.51849, 414.35813, 1186.2374, 228.85767, 145.39809, 152.97972, 273.44345, 1142.1677, 226.63348, 151.83939, 161.1337, 287.8845, 1518.837, 411.94183, 271.83
69, 287.67227, 528.88835]
image17: [18775.229, 924.74341, 615.92841, 584.33569, 1855.8827, 1855.2393, 162.99483, 105.64679, 188.2317, 197.17392, 984.98723, 184.687, 66.528385, 66.668969, 128.58684, 783.52222, 112.87693, 78.637893, 69.714888, 125.36866, 1221.3342, 284.25833, 128.65
744, 126.95238, 233.63561]
image18: [6918.6289, 1551.5681, 838.94659, 67.62793, 1196.8497, 1576.125, 423.19684, 253.18539, 244.78286, 398.55914, 844.22089, 253.12271, 168.99962, 162.28528, 274.99881, 781.58655, 245.27451, 162.11588, 169.47537, 308.18324, 1226.8686, 403.31909, 274.
34158, 287.69788, 863.24811]
image19: [12557.849, 1217.9784, 548.85199, 539.42238, 1185.5885, 2361.2483, 412.12253, 219.54691, 222.39889, 485.8334, 1295.3469, 291.58133, 171.633, 181.27854, 406.89737, 1352.2764, 352.57465, 218.52292, 238.81646, 551.19348, 3112.5112, 888.98474, 564.71
986, 632.26884, 1486.8591]
image20: [1609.5511, 1172.391, 966.8897, 1883.8411, 1383.8391, 1712.5823, 483.8183, 229.85257, 287.67242, 349.44113, 1299.9688, 232.03588, 143.57619, 411.65884, 238.61531, 1262.8134, 232.89244, 149.28249, 158.85776, 257.99271, 1618.7551, 422.89212, 268.68
938, 278.20187, 486.53852]
image21: [12389.238, 946.81862, 462.41483, 873.8143, 819.57593, 2155.4814, 258.55487, 141.99582, 411.51872, 268.37867, 1085.4236, 166.58885, 180.48999, 180.56312, 187.79738, 889.59717, 186.68439, 114.85536, 114.29317, 211.38243, 1682.5277, 384.3157, 238.9
2235, 239.69662, 447.88673]
image22: [6183.8767, 1654.5563, 997.4953, 979.26263, 1592.5549, 1581.1869, 469.58897, 385.37918, 318.43491, 517.4873, 856.84979, 286.12421, 193.84125, 281.57927, 341.18387, 787.4729, 271.66379, 189.28334, 202.53861, 356.16614, 1188.8626, 420.44483, 380.39
989, 354.51385, 642.53815]
image23: [11349.25, 1999.5841, 959.52124, 1080.86, 2335.2534, 1966.856, 468.88939, 388.86221, 357.99362, 878.84375, 921.96198, 425.84695, 129.78576, 81.668744, 84.897633, 137.96631, 429.84749, 143.46112, 96.88312, 105.22319, 179.12586, 715.58167, 256.58516, 172.31
537, 181.82744, 328.89477]
image24: [4888.369, 638.65471, 632.5188, 374.4342, 638.78632, 1536.8488, 284.38829, 128.24734, 117.18268, 198.78212, 186.8873, 148.48157, 87.428635, 83.897736, 141.09517, 1832.8877, 167.38374, 188.29665, 96.613388, 63.21988, 1613.3667, 315.47998, 191.51
945, 185.76395, 328.37729]
image25: [7548.7329, 1537.866, 788.25763, 678.41144, 1882.1489, 1566.5293, 489.61227, 224.28833, 219.64919, 345.2417, 819.53156, 248.89853, 146.82386, 443.83691, 237.77788, 712.28412, 226.44664, 143.72827, 146.14586, 258.71519, 1843.4242, 351.83849, 231.7
4922, 242.69485, 444.79828]
image26: [1811.8838, 2484.5882, 1383.7286, 1287.8184, 2736.6789, 2888.9277, 616.31489, 398.67239, 427.68443, 972.49567, 1111.8884, 378.5715, 256.26074, 298.88582, 719.43646, 1115.3381, 483.98482, 296.83885, 368.63162, 888.71747, 2382.9124, 953.29584, 721.
```

Figure1: 25-D feature vector for each image

```
dimension discriminantPower:
dimension1: 0.607944
dimension2: 0.466016
dimension3: 0.767967
dimension4: 0.89884
dimension5: 0.841053
dimension6: 0.685375
dimension7: 0.511713
dimension8: 0.669006
dimension9: 0.83349
dimension10: 0.828258
dimension11: 0.958515
dimension12: 0.675848
dimension13: 0.854303
dimension14: 0.912575
dimension15: 0.887824
dimension16: 0.967779
dimension17: 0.813581
dimension18: 0.905337
dimension19: 0.916348
dimension20: 0.885276
dimension21: 0.750251
dimension22: 0.741242
dimension23: 0.819932
dimension24: 0.848805
dimension25: 0.775242
```

Figure2: discriminant power of each dimension in 25-D features vector

```
[-6749.936, -523.57269, 1275.7162;  
255.6255, 901.12707, 510.66843;  
-1166.717, -397.76498, -469.70648;  
4215.6313, 279.28592, -667.77692;  
-4608.9404, 3363.02, 2172.2588;  
1545.8577, 969.22821, -119.11047;  
-1534.7037, -186.29642, -469.68292;  
2945.645, -326.01175, 552.11499;  
-3425.29, 196.2282, -906.79022;  
6443.1265, -1183.7158, -210.46414;  
-1287.4419, -241.52553, -180.36911;  
3565.2068, 4323.8022, -270.85178;  
-4118.3823, 274.28662, -536.25128;  
-672.84058, 999.71417, -2045.7469;  
-2035.2794, 87.761383, -210.0959;  
6584.4897, -724.09674, 239.06796;  
-3700.5991, 253.43521, -579.14313;  
2472.8254, -1158.2434, -287.77328;  
-1342.4395, -272.84567, -100.76254;  
4462.9917, 1079.059, -362.22934;  
-3588.7063, 289.62231, -795.45148;  
4114.0303, -865.05298, -333.26541;  
-2138.675, 102.58337, -46.408226;  
5236.3687, 1744.9067, 953.53717;  
1485.4243, -2623.0535, 943.53345;  
-5856.9351, -1701.7515, 292.85666;  
-2036.1925, 65.529884, -76.10833;  
-1966.7269, 414.284, 435.76865;  
893.71014, -1701.0784, 596.08911;  
-4186.2275, -635.58197, -578.38367;  
-737.5174, -644.71063, -60.01004;  
1789.7703, 2711.9614, 332.39117;  
1275.9301, -2262.4702, 851.34924;  
708.18011, -521.1134, -384.44357;  
-784.78351, -659.35211, 7.6219926;  
3943.4934, -1517.5981, 527.85181]
```

Figure3: 3-D feature vector for each image

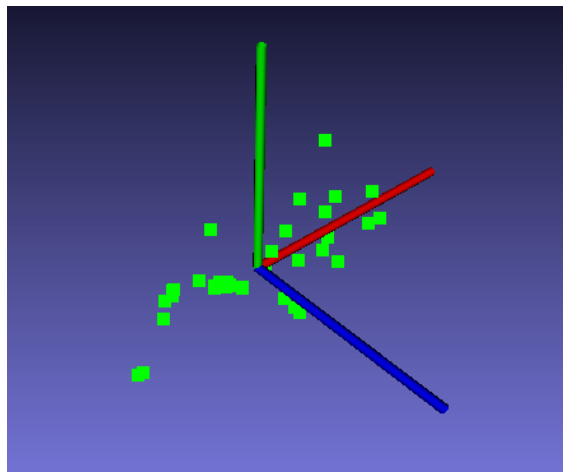


Figure4: reduced 3-D feature vectors of train data in the 3-D feature space

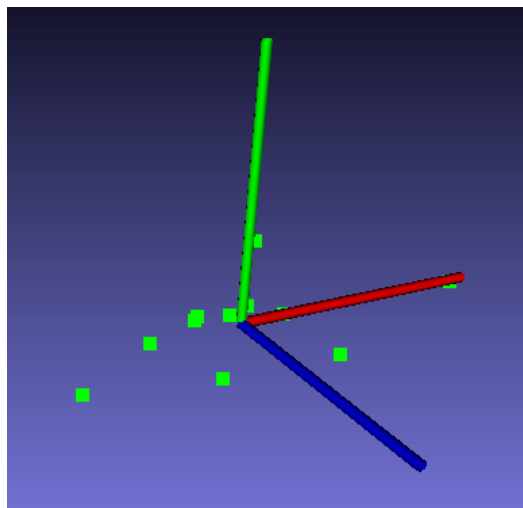


Figure5: reduced 3-D feature vectors of test data in the 3-D feature space

```
25-D classify:
2 2 2 1 2 3 3 2 3 2 2 3
error rate :0.5
3-D classify:
4 1 1 4 2 1 1 2 1 2 1 1
error rate :0.75
```

Figure6: test error rate by the nearest neighbor rule

```
25D classify:
2 1 3 3 4 3 3 2 3 3 1 3
error rate :0.416667
3D classify:
4 2 3 3 1 3 3 4 3 3 2 3
error rate :0.666667
```

Figure7: test error rate by K-means

```
25D classify:
4 1 4 1 4 3 3 4 2 2 2 3
error rate :0.166667
3D classify:
2 1 1 4 2 4 4 2 4 4 1 4
error rate :0.75
```

Figure8: test error rate by SVM

### Homework Answer and Discussion:

We used 5x5 Laws Filters to extract the response vectors and calculate the average energy feature, then we can obtain the discriminant power of the 25-D feature by computing the energy feature. Because discriminant power is equal to intra-variety divided by inter-variety, we can find that the smaller discriminant power is the better feature is found. From figure2, we can find that the feature of dimension2 is the best, while the feature of dimension16 is the worst.

Reduce the feature dimension from 25 to 3 using the principal component analysis (PCA). Then, from figure4, 5, we can observe the distribution of reduced 3-D feature vectors in 3-D feature space of train data and test data respectively. We can find that the distribution of features according to the X-axis is more obvious. The distribution of features according to the other two axes is not obvious.

Then we will use the nearest neighbor rule, K-means and SVM for texture classification. We can obviously find from figure6,7,8 that the nearest neighbor rule and K-means methods are not very effective on the samples we tested. The correct rate is maintained at about 50%, while the error rate of SVM is only 17%. We can also find that compared with the data after using PCA dimensionality reduction, the accuracy of the original data is higher, which is likely because the reduction from 25D to 3D leads to the loss of a lot of important information carried by energy feature, thus affecting the judgment of the classifier

## Problem2

### Texture Segmentation

#### Abstract and Motivation:

Texture segmentation is a fundamental problem in the fields of image processing and computer vision. It aims to partition an image into regions with similar texture properties based on the texture features present in the image. Texture refers to visual characteristics in an image that exhibit local spatial repetition, such as color, brightness, texture orientation, and frequency.

#### A) Basic Texture Segmentation

##### Approach and Procedures:

First, we need to subtract the local mean and extract response vectors just like we did in problem one, then we can get a 25-D response vectors.

Second, we need to compute the energy measure for each center pixel based on step1. We have to choose different window sizes to slide on the response vectors and use the window averaging result as the central pixel's value by following equation. Then we can get a 25-D energy map.

$$average\ energy = \sqrt{\frac{1}{MN} \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} I(i,j)^2}$$

Third, because all of the filters' responsive results have zero-mean except for L5L5 filter, all the other filter's results can be normalized and L5L5 results can be discarded. Then we can get a 24-D feature vector.

Last, apply K-means cluster with 24-D feature vector, and we can get 5 different classes. Using different colors to mark the different textures in the image to observe the results.

#### Experimental Results:

Shown below are the results of Problem 2(a):

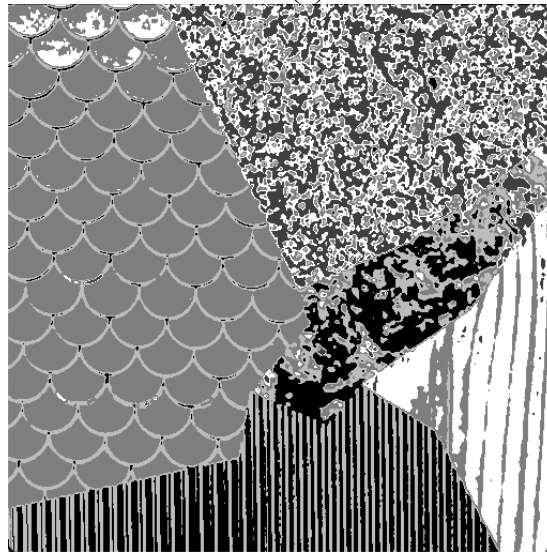


Figure1: Texture Segmentation with windowSize(3)



Figure2: Texture Segmentation with windowSize(9)



Figure3: Texture Segmentation with windowSize(15)



Figure4: Texture Segmentation with windowSize(35)



Figure5: Texture Segmentation with windowSize(55)

#### **Homework Answer and Discussion:**

Figure 1 to 5 shows the segmentation result of different window size. From the figure, we can observe that the larger window size is, the better the effect of image segmentation. When window size = 55, it shows the a great result. However, we can find that there is a region in the graph that is not segmented correctly. It is likely that the boundary of the graph is separated into a class by K-means algorithm, and the middle region may be ignored because it is too small or not obvious.

#### **B) Advanced Texture Segmentation**

##### **Approach and Procedures:**

First, all the steps are the same as in A.

Second, before we perform K-means, we need to perform PCA dimensionality reduction on the 24-d future vector, and then we can obtain the 3-d future vector, which can then be classified using K-means algorithm.

Third, we can also apply morphological closing operation on image result. This operation expands and then corrodes the regions in the image, effectively filling small holes or gaps within the regions.

##### **Experimental Results:**

Shown below are the results of Problem 2(b):



Figure6: Texture Segmentation with windowSize(15) by using PCA

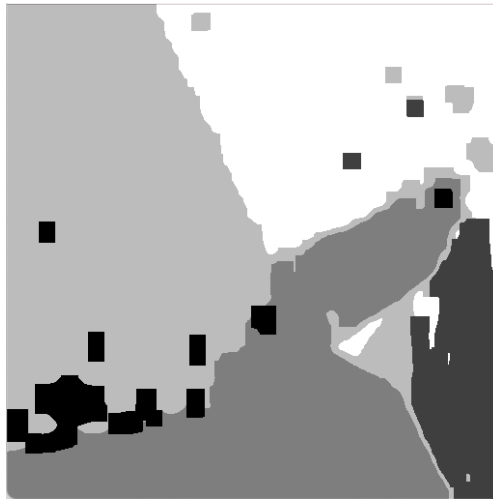


Figure7: Texture Segmentation with windowSize(15) by using PCA and post-processing



Figure8: Texture Segmentation with windowSize(35) by using PCA





Figure9: Texture Segmentation with windowSize(35) by using PCA and post-processing



Figure10: Texture Segmentation with windowSize(55) by using PCA



Figure11: Texture Segmentation with windowSize(55) by using PCA and post-processing

**Homework Answer and Discussion:**

Figure 6,8,10 shows the texture segmentation applied to different Windows after PCA dimensionality reduction. Figure 7,9,11 shows the texture segmentation applied to different Windows after PCA dimensionality reduction and post-processing. We can find that the effect of texture segmentation does not change much after PCA processing, so PCA is not suitable for this experiment. On the contrary, by developing a post-processing technique to merge small holes, some misclassifications classified to outlying areas are eliminated, showing better results.

# Problem3

## SIFT and Image Matching

### Abstract and Motivation:

The primary goal of the SIFT algorithm is to extract image features that are invariant to scale, rotation, and illumination changes, and utilize them for tasks such as image matching, object recognition, and reconstruction.

### (a) Salient Point Descriptor

#### Homework Answer and Discussion:

1. The SIFT algorithm is robust to various geometric modifications, including scale changes, rotation, translation, affine transformations and so on.
2. Scale Changes: SIFT features are detected and described at multiple scales, allowing them to be robust to changes in the size of objects in the image. This means that the same feature can be detected at different scales.  
Rotation: SIFT features are invariant to rotation, meaning they can be detected even when the image has been rotated.  
Translation: While not explicitly invariant to translation, SIFT features can still be matched across images with slight translation variations.  
Affine Transformations: SIFT features are robust to affine transformations such as shearing and stretching, although they may not be completely invariant to these transformations.
3. SIFT's robustness to illumination changes stems from its focus on local gradient-based descriptors, its scale-invariant feature detection approach, and its local matching strategy, all of which reduce the reliance on absolute pixel intensities and enhance the algorithm's ability to capture and match distinctive local image structures.
4. The use of the Difference of Gaussians in the SIFT algorithm offers computational efficiency, smoother scale space representation, improved localization of keypoints, and robustness to noise, making it a preferred choice for scale-invariant feature detection and description.
5. The SIFT algorithm outputs a feature vector of size 128 for each keypoint detected in the image.

### (b) Image Matching

#### Approach and Procedures:

I will extract SIFT features from images and visualize the matching between the keypoint with the largest scale in one image and its closest neighbor in another image.

First, we need to extract SIFT features, including keypoints and descriptors, from the input images and then we use the descriptor to select the keypoints that do well.

Second, we need to identify the keypoint with the largest scale in the first image and records its index.

Third, find the keypoint in the second image that is closest to the keypoint with the largest scale in the first image.

Fourth, Draw the keypoints corresponding to the largest scale in the first image and the closest neighboring keypoint in the second image.

Fifth, matches the extracted features.  
Last, draws and displays the matching results.

### Experimental Results:

Shown below are the results of Problem 3(b):



Figure1: largest scale in Cat\_1 and closet neighboring key point in Cat\_3

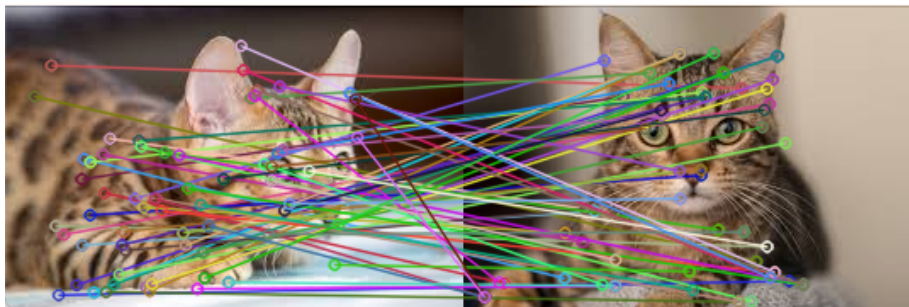


Figure2: corresponding SIFT pairs between the Cat\_1 and Cat\_3



Figure3: largest scale in Cat\_3 and closet neighboring key point in Cat\_2

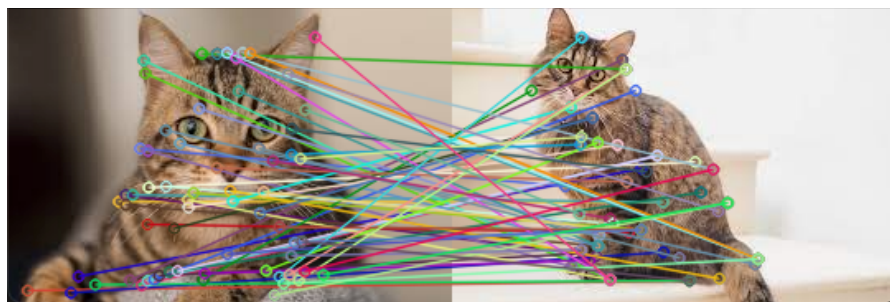


Figure4: corresponding SIFT pairs between the Cat\_3 and Cat\_2



Figure5: largest scale in Dog\_1 and closet neighboring key point in Cat\_3



Figure6: corresponding SIFT pairs between the Dog\_1 and Cat\_3

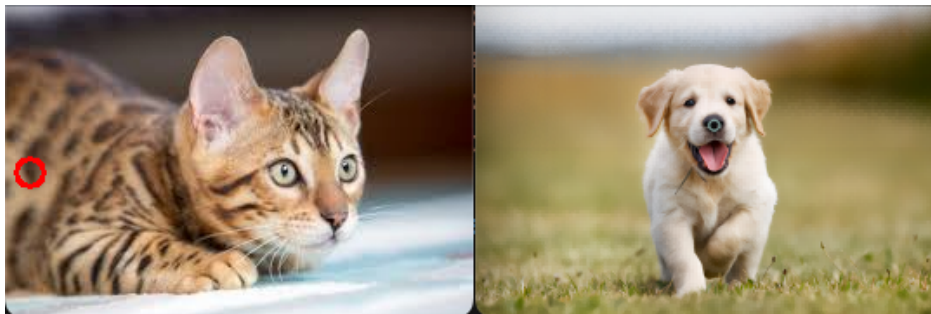


Figure7: largest scale in Cat\_1 and closet neighboring key point in Dog\_1



Figure8: corresponding SIFT pairs between the Cat\_1 and Dog\_1

### Homework Answer and Discussion:

From figure 1 to 8, we can find that in SIFT, matching may not perform well between different objects or the same object viewed from significantly different angles. The success or failure of matching depends on the similarity between objects and the extent of viewpoint differences. For objects that are similar and have similar viewpoints, matching tends to be more

accurate as they have similar descriptors in feature space. However, when there are significant differences between objects or large changes in viewpoint, SIFT may produce mismatches or fail to find matches because of the large differences in their feature descriptors, making it difficult to find suitable correspondences.

### (c) Bag of Words

#### Approach and Procedures:

First, we need to extract SIFT features from all four image and, then reduce dimension of the 128-d feature vector to the 20-d feature vector by PCA.

Second, we need to apply K-means clustering to create 8 dictionary's visual words.

Third, for each feature in each of four images, we need to calculate the nearest neighbor of cluster center in the dictionary.

Fourth, then for each of four images, build a histogram of length k (frequency of words in that image).

Last, calculate the similarity index between different image and compare codeword histogram intersection of the three other images with Cat\_3 image codeword histogram.

$$Similarity\ Index = \frac{\sum_{j=1}^k \min(I_j, M_j)}{\sum_{j=1}^k \max(I_j, M_j)}$$

#### Experimental Results:

Shown below are the results of Problem 3(a):

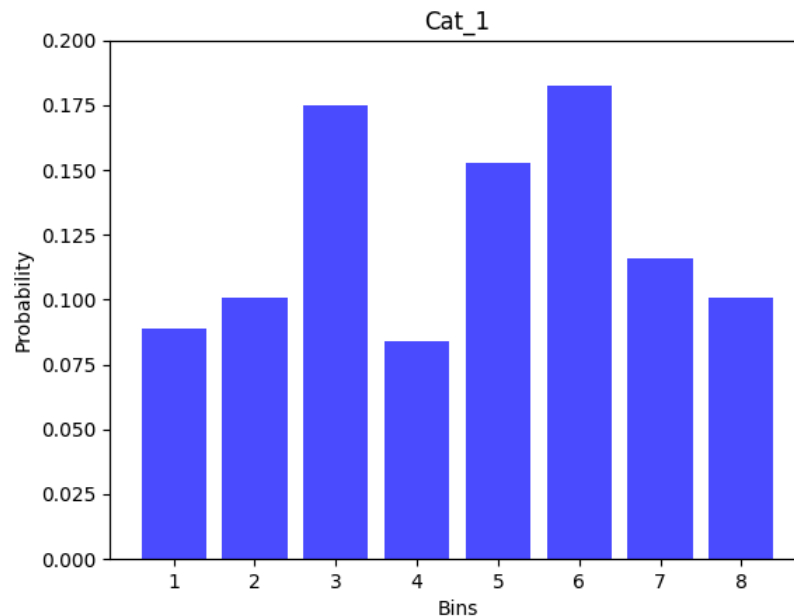


Figure1: codeword histogram for Cat\_1

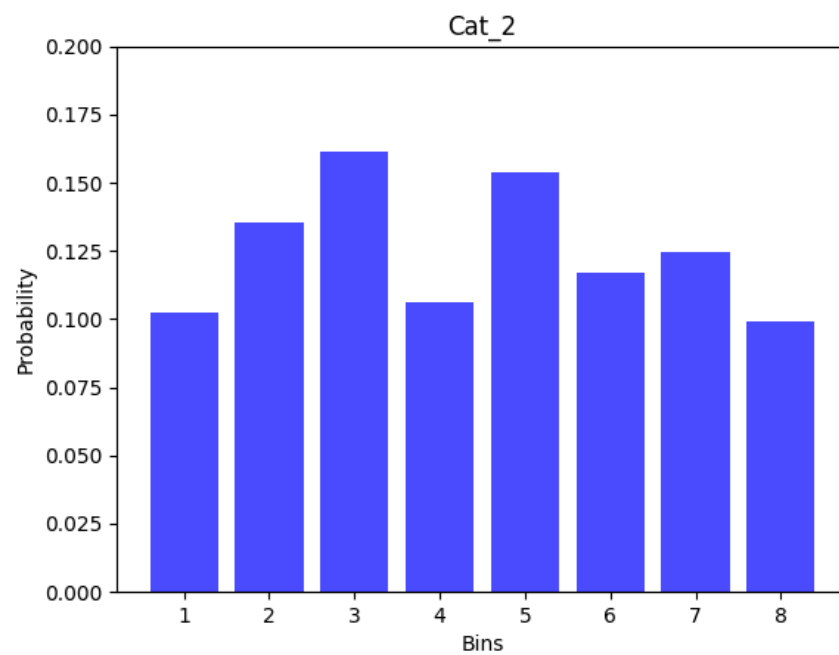


Figure2: codeword histogram for Cat\_2

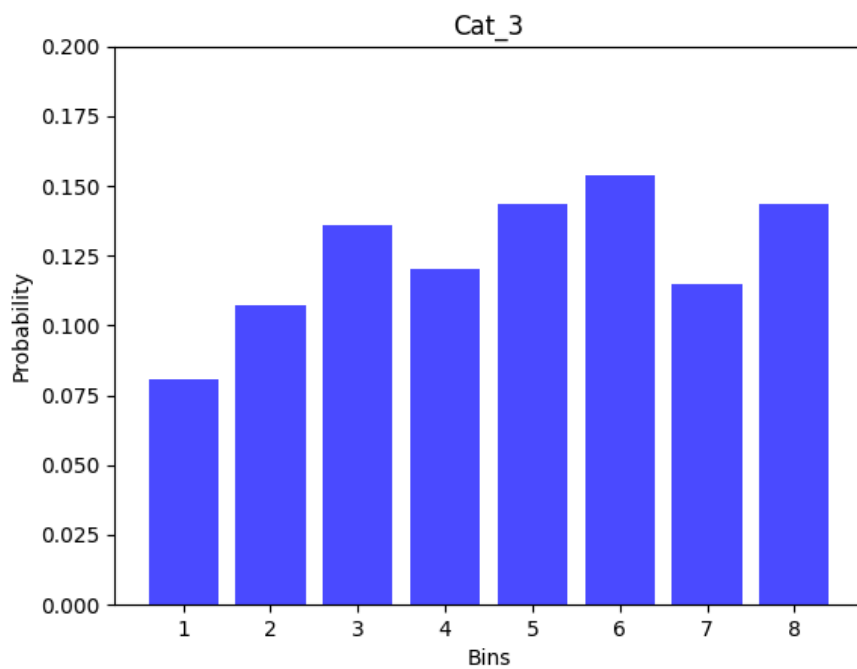


Figure3: codeword histogram for Cat\_3

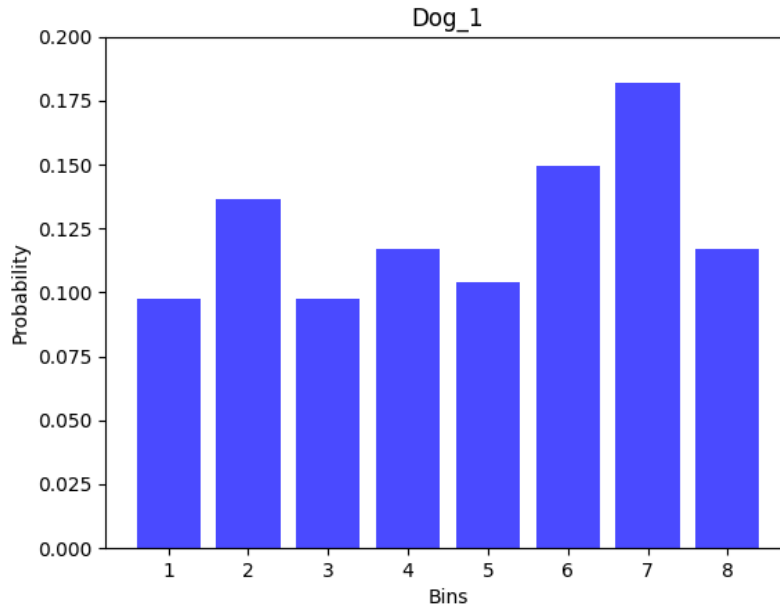


Figure4: codeword histogram for Dog\_1

```
similarity between cat_3 and cat_1
0.843245
similarity between cat_3 and cat_2
0.8258
similarity between cat_3 and dog_1
0.79741
```

Figure5: similarity between different image

### Homework Answer and Discussion:

From figure 1 to 4, we can find the histogram of different image, by comparing the histogram intersection, the similarity between images can be quantified to determine which images are more similar to Cat\_3 images. From figure 5, we can find the similarity index between Cat\_3 and other images. We can obviously find that the similarity of pictures of the same object is greater than that of pictures of different objects, but there may be some similarity between images of different objects, especially in similar scenes.