EE569 Introduction to Digital Image Processing

Homework Report #4

Name: Boyang Xiao **USC ID:** 3326730274 **Email:** boyangxi@usc.edu

I. Problem 1: Texture Analysis

1.1. Part (a) Texture Classification – Feature Extraction

1.1.1. Abstract and motivation

Texture is a very important and usual feature in the realm of image processing. An image texture is a set of metrics calculated in image processing designed to quantify the perceived texture of an image. It usually contains periodical (or half-periodical) patterns which are of high frequency of low frequency. The image texture can give us information about the spatial arrangement of color or intensities in an image or selected region of an image. Therefore, the analysis regarding the image texture is very important if we want to analyze an image and obtain key information from it.

In this part of problem, the basic image texture analysis based on Law's filters will be implemented and discussed. Different dimensions of features will be extracted by the analysis methods and will be visualized to show the results.

1.1.2. Approaches and procedures

There are three steps to conduct the texture analysis in this part of problem:

- Filter bank response computation
- Energy feature averaging
- Feature reduction

They will be elaborated in the following section.

- **Filter bank response computation**

The first step to analyze the image is to convolve the image with Law's filters. The 1D 5×5 filter kernels are given in the Table 1 below. To form the convolution 2D filter kernels, matrix multiplication has to be applied to one 1D kernels with each other and that will make 25 of 2D filter kernels.

After we obtain all the 25 filtering kernels, we apply them to the texture image pixels and get 25 filtered images.

Table 1: 1D Law's filter kernels

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

- **Energy feature averaging**

To average the energy feature, we sum up all the 128*128 squared filtering responses with the same filter in the same filtered image and divide it by 128*128. The filtering results for a single texture image should be 128*128*25. After the energy feature averaging, the feature dimension should be reduced to 1*25.

- **Feature reduction**

The feature dimension has been reduced to 25. In order to reduce the dimension further and pick the top three features, the principal component analysis (PCA) method will be applied to the 25D features.

PCA picks a small group of features from a larger group of features, which are called principles components. This method is widely used in data exploration and dimension reduction. In this specific problem, PCA will be applied to reduce the feature dimension from 25 to 3, which means the 1*25 feature vector will be turned into a 1*3 feature vector, which contains the most important features for this single image.

1.1.3. Experimental results

To tell which feature dimension has the strongest discriminant power and which has the weakest discriminant power, the averaged 25D feature vector from the whole 36 images are displayed as following.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
75857	154.76	25.57	22.12	104.34	146.08	4.16	1.14	1.12	5.17
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
26.11	1.26	0.44	0.47	2.16	22.06	1.22	0.48	0.52	2.39
21	**22**	**23**	**24**	**25**					
87.36	4.78	1.96	2.12	10.37					

From the table above, the feature dimension that has the **strongest** discriminant power is #1, which is the **L5L5 dimension**. And the feature dimension that has the **weakest** discriminant power is #13, which is the **S5S5 dimension**.

The extracted features of 36 training images in 3 dimensions are shown in Figure 1 below, where the blue dots are blanket textures, the black dots are brick textures, the green dots are grass textures and the red dots are the stone textures.

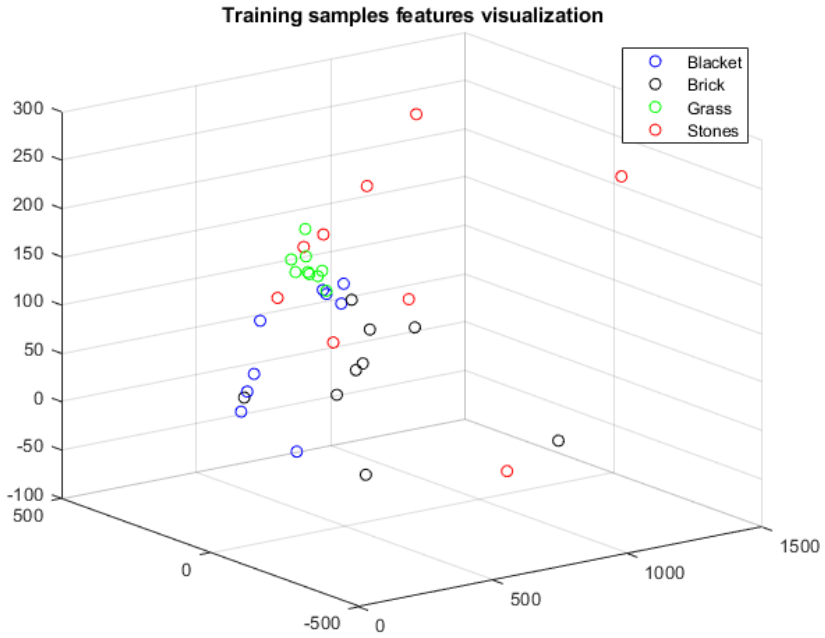

Figure 1: Training images 3D features visualization

1.1.4. Discussions

All the questions have been answered in above sections.

1.2. Part (b) Advanced Texture Classification --- Classifier Exploration

1.2.1. Abstract and motivation

From the section above, the features of texture images have been extracted and reduced to 3 dimensions. A simple classification based on Mahalanobis distance has also been applied to the image samples to classify the texture patterns. In this part of problem, two advanced texture classification methods will be introduced and implemented, which are k-means clustering and random forest. The main idea of each classifier will be discussed and the accuracies will be shown in this part.

1.2.2. Approaches and procedures

1.2.2.1. K-means clustering --- unsupervised learning

The k-means clustering method is a classic unsupervised learning algorithm which is designed to partition a group of samples into k clusters depending on the multi-dimension features of the sample points. It does not depend on training samples and labels to train the model, but generates and adjusts the clustering by the test samples directly.

The first step to start the K-means clustering is to randomly pick k samples as the initialization of the centroids. After iterations, the picked centroids will be adjusted to maximize the smallest distance to the centroids picked and to decide the next iteration's centroid. The clustering procedure will continue until the boundary conditions are reached. In this stage, the samples should be divided into k clusters.

1.2.2.2. Support vector machine(SVM) --- supervised learning

Support vector machine is a very popular supervised learning model with associated learning algorithms that analyze data for classification and regression analysis. SVM is one of the most robust prediction methods and it has many advantages: it's effective in high dimensional spaces and in cases where number of dimensions is greater than the number of samples and it uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

In this part, we will only use SVM to do the classification task. Training samples and labels will be input to train the SVM model and test samples will be used to test the accuracy of the trained model.

1.2.2.3. Random Forest (RF) --- supervised learning

Another popular model in the realm of supervised learning is Random Forest, which has already been introduced in the previous homework. Random Forest (RF) model is a typical kind of ensemble training, which means it use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In RF model, it uses multiple decision trees to do the classification and then obtain the results synthesizing the results from all the decision trees. If the classification is discrete, the RF model uses the mode from all the results. And if the classification is continuous, the RF model uses mean value of all the results to calculate the final result.

In this part, training sample images will be input to train the RF model and the test images will be used to calculate the model accuracy.

1.2.3. Experimental results

The classification accuracies of different methods are shown in Table 2. Below.

Table 2: Accuracies for different classification methods

Classification method	25D features for k-Means	3D features for k-Means	SVM	RF
Accuracy	0.500	0.583	0.667	0.833

1.2.4. Discussions

From the experimental results, the unsupervised learning has quite a low accuracy while the supervised learning obtains a higher accuracy. The two unsupervised learning methods have also obtained different accuracies. Since the 25D features may contain too much unnecessary information while the reduced 3D features still preserve the important information.

As for the two methods of supervised learning, RF clearly obtains a better result than SVM. However, since the training samples and testing samples are both too small in this specific case, this simple experiment cannot tell which method are better than the other one. Also, the initialization and hyperparameters settings can also influence the final results. Even when training the same model, different trainings can get different results. One thing has to mention is that when training the SVM and RF models, different hyperparameters are tried and only the highest accuracy is selected.

All the other questions and results are shown in the previous sections.

II. Problem 2: Texture Segmentation

2.1. Part(a) Basic Texture Segmentation

2.1.1. Abstract and motivations

In the previous section, we discussed the basic techniques to analyze the texture images and to classify different texture patterns. A very important application based on these techniques is Texture segmentation. The real-world images are actually all consisting of series of textures, such as grass, bricks, walls and other textures. To better analyze the images and obtain higher-level information from the images, segmentation of textures in these images is a very effective method.

In this part of problem, a basic method of texture segmentation based on the Law's filters and k-means clustering will be introduced and implemented.

2.1.2. Approaches and procedures

The texture segmentation method will be elaborated in four steps below:

- **Filter bank response computation:** The very first step is much similar to the first step in the Problem1. We will use 25 different Law's 2D filter to convolve the mosaic image and get the filtering responsive images.
- **Energy feature computation:** Since the object image is a whole mosaic image containing different kinds of mixed textures, we should use sliding windows to compute the energy averaging results within the windows but not use the whole energy averaging in the range of the whole image. We have to choose different window sizes to slide on the filtered images and use the window averaging result as the central pixel's value. After this step, every pixel will have a new 25D feature vector.
- **Energy feature normalization:** Since 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.
- **Segmentation**: Use K-means clustering to cluster all pixels in the image and get 6 different classes. Use different colors to mark the different textures in the image to observe the results.

In this part of problem, a very important factor to influence the results is the window sizes chosen to do the energy feature averaging. In the experiments, different window sizes will be tried and all of the results will be shown. Discussions regrading window sizes will be following.

2.1.3. Experimental results

Texture segmentation results with different window sizes are shown in the Figure 2 below.

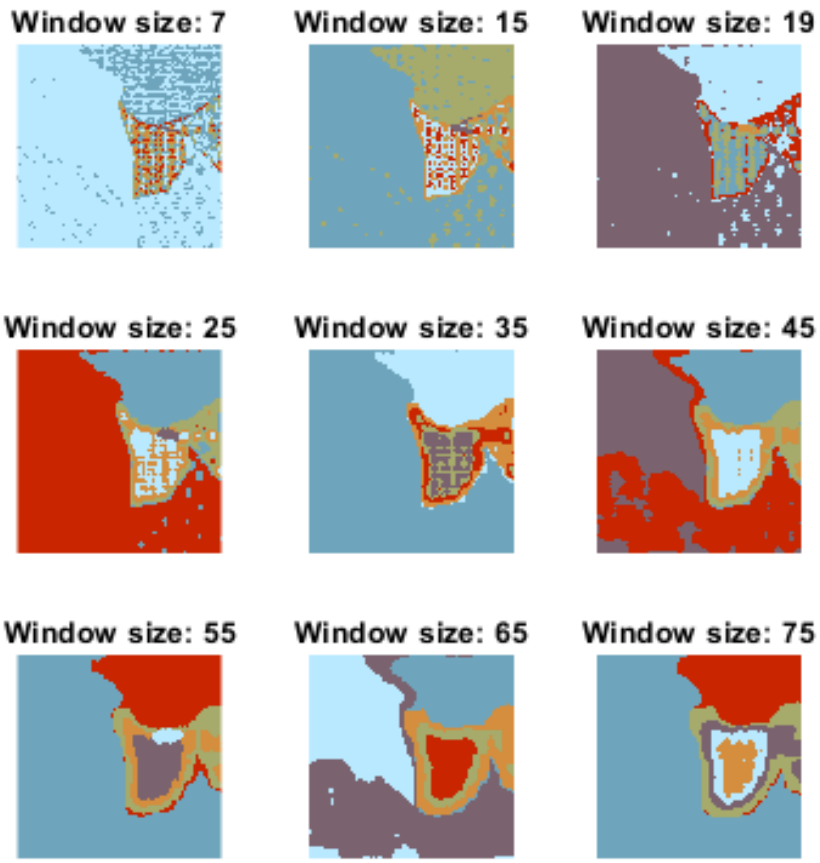

Figure 2: Basic texture segmentation with different window sizes

One thing that has be clarified is that the colors for different regions are different in different results because k-means clustering does not specify each cluster's label. To observe the results, we can simply focus on the region segmentation but not region's colors.

2.1.4. Discussions

From the experimental results shown above, the basic texture segmentation can roughly segment the textures and tell the difference from different regions. However, the details are still not satisfying, since there are some similar texture regions that are not segmented, such as the lower part of the images. For the textures that are greatly salient, they are easily classified.

Regarding the window sizes, if we use smaller window, the boundaries of the texture regions are clearly segmented, but there are many holes(errors) in the continuous texture regions. This happens because the window size is too small, and the texture patterns are greatly different from each other. Even in the same region, the texture pattern can also be different. Therefore, there are some errors in the same region.

However, if we use a larger window, the errors occurring in the same region are much less, but some similar regions cannot be distinguished and the edges of the texture regions are also blurred. This happens because the window size is too large and the sample features are much similar to each other in the similar or adjacent regions. Also, on the edges of the texture regions, the window sampling absorbs too much information from the pixels on the both sides of the regions and the edge pixels are much harder to be distinguished. Therefore, the edges are blurred.

In conclusion, we have to carefully pick the right window sizes to conduct the texture segmentation. From my perspective, the window size between 45 to 65 can obtain a quite good result in this specific case.

2.2. Part(b) Advanced Texture Segmentation

2.2.1. Abstract and motivations

In the last section, we implemented a basic texture segmentation method and used it to segment a composite texture image. Even though the segmentation method can roughly segment the different textures in the image, the results are not that satisfying, because there are some similar areas that are not distinguished and the edges of texture areas are also experiencing some errors.

In this part of problem, some improvements will be applied to the basic texture segmentation methods to augment the results. A very intuitive way to improve it is to apply the PCA to the 24D features to reduce the dimension (as what we did in section 1.1). Some redundant dimensions of features will be removed by PCA and the important dimensions will be preserved.

2.2.2. Approaches and procedures

In section 2.1.2, the first step to conduct the texture segmentation is to use to Law's filters to convolve the images and get 24D features' vector for each pixel. In order to reduce the feature dimensions, we can apply the PCA to the filtered images, resize the 512*512 images to a single row vector and do the same steps to the other feature dimensions. After we get a whole matrix with 24 columns which stand for 24 features, we apply the PCA to the matrix and get a reduced matrix with fewer columns. The matrix will be resize back to 512*512 images and continue with the remaining segmentation steps mentioned in section 2.1.2.

In this part of experiments, different dimension sizes and windows sizes are tried to optimize the segmentation results.

2.2.3. Experimental results

The segmentation results with different feature dimension sizes and window sizes are shown in the Figure 3 below. The feature dimension is reduced to 3, 7 and 15 using PCA method, and the window sizes are selected as 15, 41 and 65.

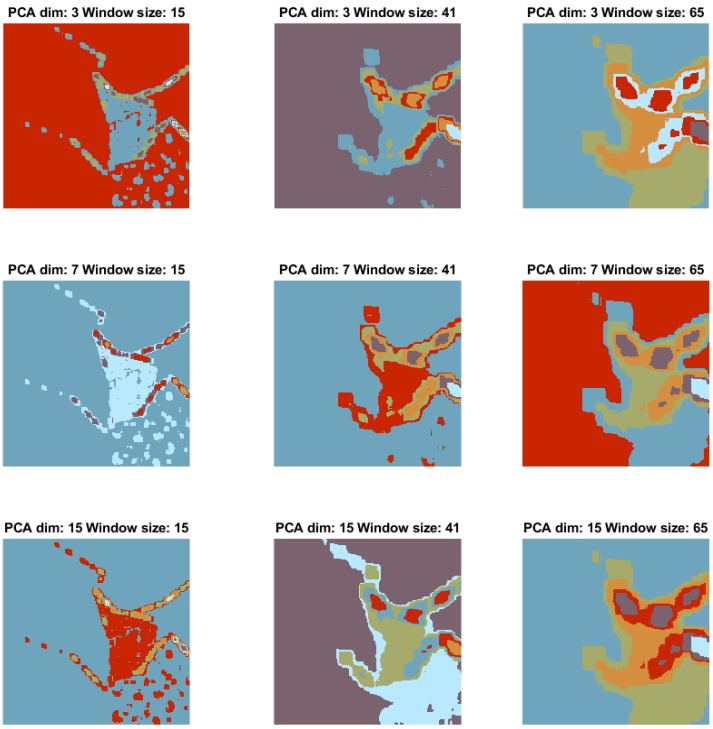

Figure 3: segmentation results with different feature dimension sizes and window sizes

2.2.4. Discussions

To better analyze the segmentation results, the original mosaic image is shown in Figure 4 below. Different textures are marked as #1-6.

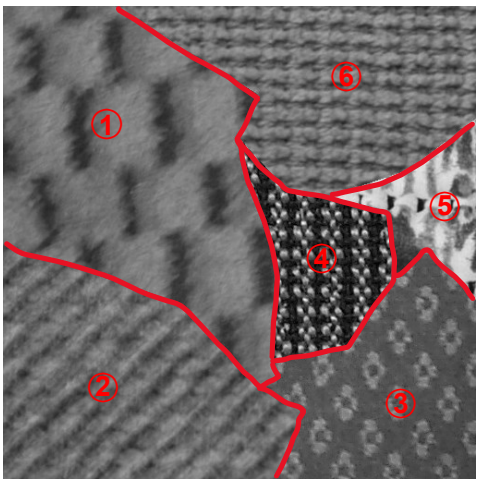

Figure 4: The original composite texture image

Compared to the basic texture segmentation method implemented in section 2.1, the most significant advantages that the improved segmentation method has is that it can tell the differences between some similar (but not the same) texture. For example, the basic method can hardly distinguish the region ①,② and ③ since these three regions are much similar to each other, but the improved method can at least recognize the boundaries of these three regions, especially using smaller window sizes (window size = 15).

Also, the purity (or accuracy) of each region is higher. The results from the basic segmentation have many holes or errors in a whole block of region, especially for region ④ and ⑤ which have very fancy patterns. However, the improved segmentation method can avoid this problem and obtain a pure region of textures.

Comparing the results from the improved segmentation method with different parameters, different PCA dimension settings can make little different difference. But when different window sizes are used, the results are quite different from each other. We cannot tell easily tell which parameter setting is the optimized, and we should try multiple parameters case by case.

To improve the segmentation results further, some advanced algorithms can be used. For example, we can use morphological processing to fill the holes and errors in a whole block of areas. Also, we can depend on the texture region boundaries detected by the experiments to segment the region.

All the questions' answers and results are shown in the previous sections.

III. Problem 3: SIFT and Image Matching

3.1 Part(a) Salient Point Descriptor

3.1.1 Non-programming questions

Q1: From the paper abstract, the SIFT is robust to what geometric modifications?

Answer: The SIFT is robust to image scaling, image rotating and also the affine distortion.

Q2: How does SIFT achieve its robustness to each of them?

Answer: SIFT is robust to image scaling because SIFT conduct the scale-space extrema detection ahead of all the steps. The scale-space extrema detection convolves the image to obtain a series of different scale images, so SIFT can be robust to different image scaling. And SIFT is robust to image rotating because SIFT do the orientation assignment, in which One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation.

Q3: How does SIFT enhance its robustness to illumination change?

Answer: SIFT is robust to illumination change because SIFT has a Keypoint Descriptor stage where the local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of change in illumination.

Q4: What are the advantages of using Difference of Gaussians (DoG) instead of Laplacian of Gaussians (LoG) in SIFT?

Answer: The first reason is that DoG is particularly efficient to compute because it can be computed by simple image subtraction while LoG need to be computed in any case for scale space feature description. The second reason is that DoG provides a close approximation to the scale-normalized LoG.

Q5: What is the SIFT's output vector size in its original paper?

Answer: The output vector size is 128.

3.2 Part(b) Image Matching

3.2.1 Abstract and motivation

As introduced in the last section, SIFT is very powerful tool to detect the salient points in the images. Even if the images has experienced some geometric transformation like scaling or rotating, the invariant salient points can still be detected and matched. In this part of problem, SIFT will be applied to multiple images to extract the features. After the salient points are extracted in the images, they will be compared and matched.

3.2.2 Approaches and procedures

In this part of problem, SIFT is implemented in an open-source library named VLFeat. The vl-sift(Image) function take the gray scale images as an input and returns both the image's keypoints frames and also the descriptor vectors.

To match the two images, we pick the keypoint with the largest scale in one of the image and find among the keypoints in the other image that have the nearest Euclidean distance on the dimensions of their descriptors(128 dimensions). Then these two points can match.

Multiple images matches are processed in this way and the results will be shown in the following section.

3.2.3 Experimental results

The keypoints detected from Cat_1.raw and Cat_Dog.raw are displayed in Figure 5 below. The keypoints disks are marked as green circles with a line in them to indicate the orientation. Since the keypoints are too many for each image and the visualization is hard to observe, **I randomly picked 200 keypoints among all the keypoints in the same image** to display.

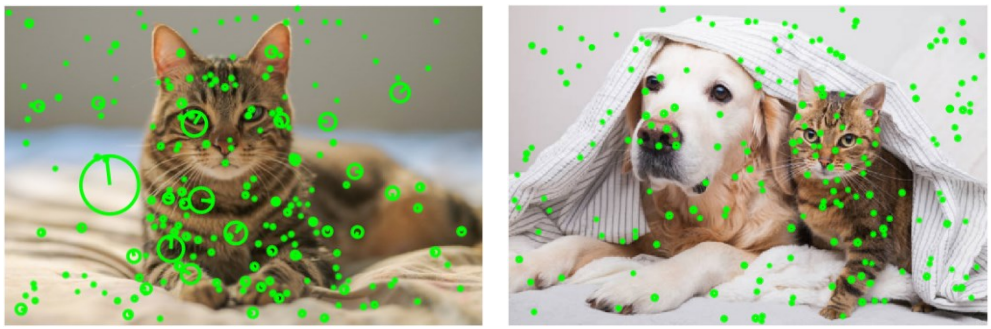

Figure 5: Keypoints detected from Cat_1.raw(left) and Cat_Dog.raw(right)

The largest scale keypoint in Cat_1.raw and its matching point in Cat_Dog.raw is shown in Figure 6 below.

Figure 6: The largest scale keypoint in Cat_1.raw(left) and its matching point in Cat_Dog.raw(right)

The largest scale keypoint in Dog_1.raw and its matching point in Cat_Dog.raw is shown in Figure 7 below.

Figure 7: The largest scale keypoint in Dog_1.raw(left) and its matching point in Cat_Dog.raw(right)

The largest scale keypoint in Cat_1.raw and its matching point in Cat_2.raw is shown in Figure 8 below.

Figure 8: The largest scale keypoint in Cat_1.raw(left) and its matching point in Cat_2.raw(right)

The largest scale keypoint in Cat_1.raw and its matching point in Dog_1.raw is shown in Figure 9 below.

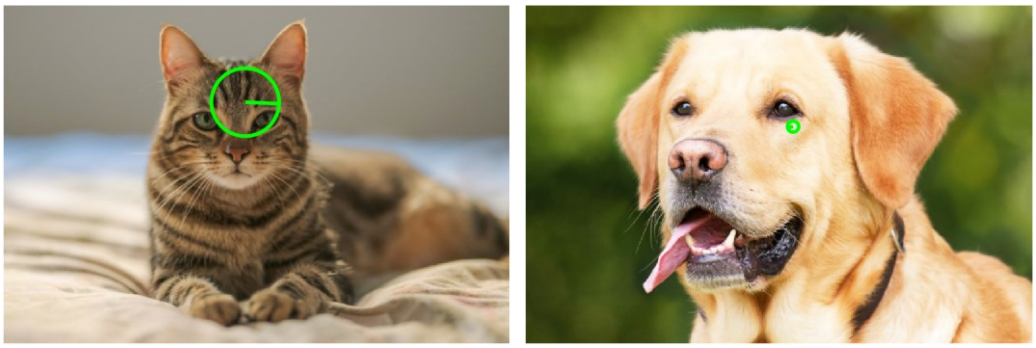

Figure 9: The largest scale keypoint in Cat_1.raw(left) and its matching point in Dog_1.raw(right)

3.2.4 Discussions

From Figure 5, we can observe the keypoints detected from the image, including their scales and orientation. The orientation is marked with a single line inside the frame disks. The orientation is calculated in the step called Orientation Assignment in the SIFT algorithm. In this step, each keypoint is assigned one or more orientations based on local image gradient directions. This is the key step in achieving invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

The Figure 6, 7, 8 and 9 show the SIFT keypoints matching from the largest keypoint in one image to the keypoint in another image. The matching is based on the smallest Euclidean distance between two descriptor vectors. Theoretically, the two matching points should have same patterns, such as they are both from the same part of the cats or dogs, since their descriptors are the closest to each other. However, the experimental results show the opposite way. The matching points' patterns are not exactly the same, and they are not located in the same area of the animals. They are somehow similar, such as they have similar illumination or texture patterns. The reason for this flaw might be that the keypoint with largest scale is always hard to find a matching point.

3.3 Part(c) Bag of words

3.3.1 Abstract and motivation

In computer vision, the bag-of-words model (BoW) can be applied to image classification or retrieval, by treating image features as words. A bag of visual words is a vector of occurrence counts of a vocabulary of local image features. To represent an image using the BoW model, an image can be treated as a document. Similarly, "words" in images need

to be defined too. To achieve this, it usually includes following three steps: feature detection, feature description, and codebook generation. A definition of the BoW model can be the "histogram representation based on independent features".

In this part of problem, BoW models will be applied to the same cats and dogs images used in the previous section. We will use the SIFT descriptors to classify the images and match images based on BoW models.

3.3.2 Approaches and procedures

To generate the BoW model, the first step is to detect the SIFT descriptors as introduced in section 3.2. SIFT can generate a large number of feature points with 128D feature vectors on these feature points. After that, I use PCA to reduce the feature vector's dimension from 128 to 20. Then the k-means clustering is applied to the 20D descriptor vectors to cluster all the feature points into 8 classes. The histogram generated according to the sample numbers in these 8 classes is the BoW model for this single images.

The steps above are repeated for each of the image, and obtain a BoW model for each image. After obtaining the BoW models for each image, we compare the Dog_2.raw to the other images' histograms using the histogram intersection method described below:

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

Where I_j is one of the Cat_1 or Dog_1 codewords histograms, M_j is the Dog_2 codeword histogram, and K is 8. The similarity between different histograms can be indicated by the similarity index calculated.

3.3.3 Experimental results

The codewords histograms are shown as Figure 10 below.

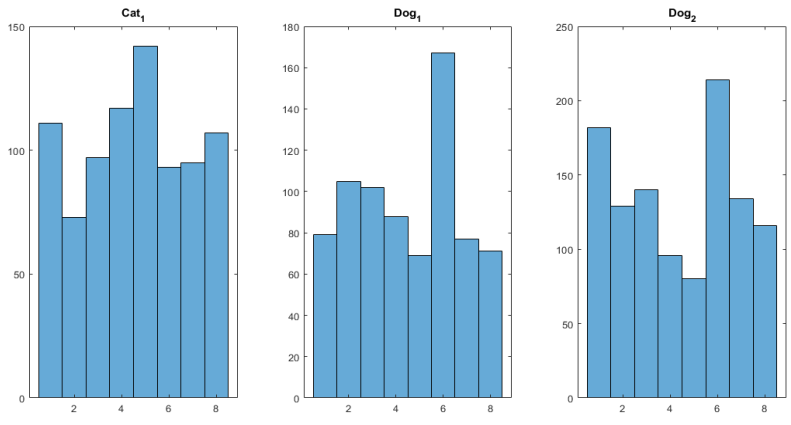

Figure 10: Codewords histogram for the three images

Using histogram similarity intersection, we can get the similarity level from Dog_2.raw to the other two images. The results are shown in Table 3 below:

Table 3: Codewords histogram similarity comparison

Images	Dog_2 and Cat_1	Dog_2 and Dog_1
Similarity	0.64055	0.69478

3.3.4 Discussions

From Table 3 shown above, the Dog_2 and Dog_1 can obviously match better because they get higher similarity index than that between Dog_2 and Cat_1. From the codewords histograms shown in Figure 10, we can also find that Dog_2's histogram has very similar patterns to Dog_1's histogram, while Cat_1's histogram has completely different patterns. Therefore, the BoW model can work well to classify different types of images in this case.