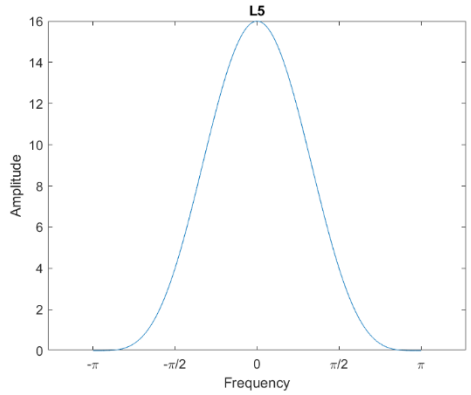
EE569 Digital Image Processing

HOMEWORK #4

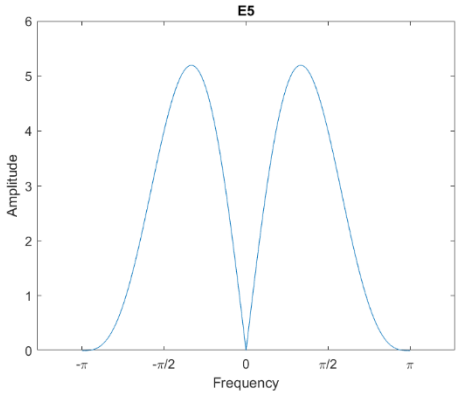
Zheng Wen
zwen1423@usc.edu

Problem 1: Texture Analysis and Segmentation

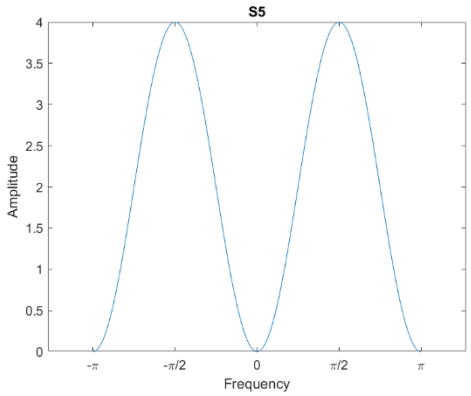
(a) Texture Classification



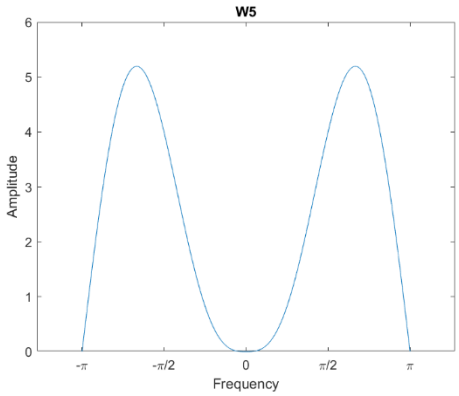
(a) Frequency Response of L5



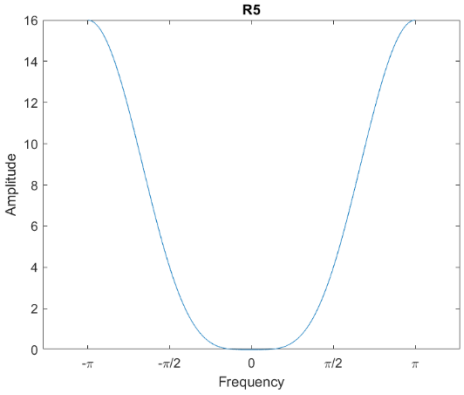
(b) Frequency Response of E5



(c) Frequency Response of S5



(d) Frequency Response of W5



(e) Frequency Response of R5

Fig. 1 Frequency Response of Laws Filter

Texture is a kind of pattern which follows certain periodical distribution of pixels, could be analysis and classified in some way. Laws Filter could be used to extract feature vectors from pixel in the texture image, images with similar textures will have similar feature vectors.

The frequency responses of the 5 1D Laws Filter are shown in Fig. 1. From Fig. 1, we could see that L5 is a low pass filter which will suppress abrupt pixel changes in the picture, such as edges, E5, S5 and W5 are band pass filters with different cut-off frequency and thus have different passband and stopband, and R5 is a high pass filter, which will preserve edges in the image and filter out the flat pixel region. These 1D filters are combined and the 2D masks are formed by the tensor product of them. There are 25 2D masks, convolve these masks with the reflection padded image and then the feature map of an image with 25 dimensions is constructed. For each dimension, there is a matrix containing the feature information with the same size as the image.

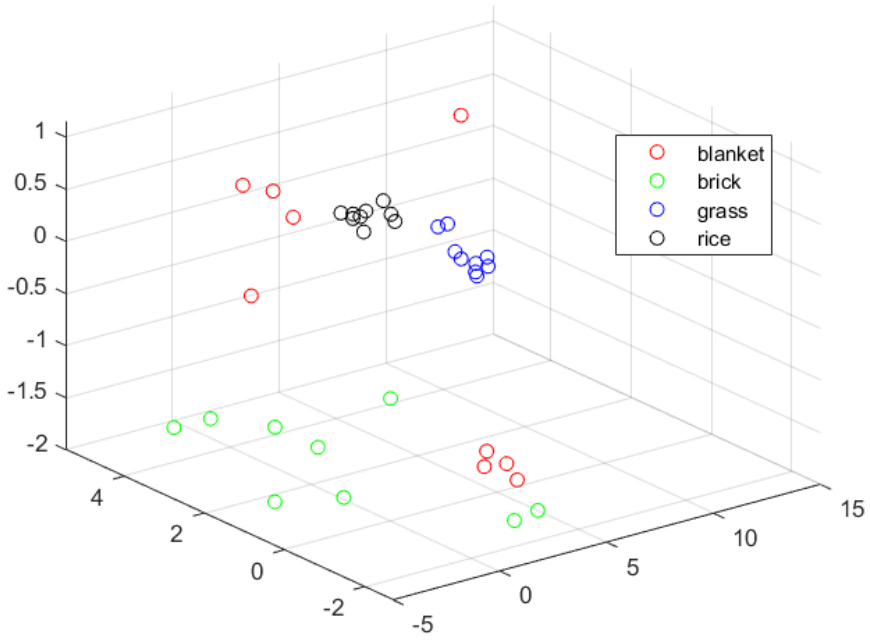


Fig. 2 Principle Component of Training Set

To get the energy of these feature map, which could be used to classify these textures, the absolute value of the feature map is averaged through each dimension feature map, and the 25-dimension energy vector is obtained for each image. Because the result of pairs with different sequences, such as L5E5 and E5L5, are different only because the different direction of these two filters, typically one filter the image vertically, and another filter the image horizontally, in other words, the pair with different sequences are transpose of each other[1], so the corresponding feature vector could be averaged and the 25D energy vector is reduced to 15D energy vector. To compare the discriminant power of each feature dimension, the ratio between inter-class variance and intra-class variance is compared, the larger the ratio is, the more discriminant power the dimension is, because a discriminant dimension should have the property that it could have

a large inter-class variance which means it could separate different classes away from each other, and a small intra-class variance, which means it could concentrate the samples in the same class tightly, with such a feature, the samples could easily be classified correctly. After checking the ratio, the feature with the strongest discriminant power is E5E5, and the feature with the weakest discriminant power is L5S5/S5L5 in this training set.

Using principle component analysis (PCA), the 15-dimension feature vector could be reduced to a fewer dimension, these components are composed of the linear combination of the original 15 dimensions, and the ones with little information will be dropped and those with large variance will be used to do classification. The distribution of the principle component of the training set is shown in Fig. 2.

PCA is implemented by Karhunen-Loeve transform[2], which directly decompose the covariance matrix of standardized samples into eigen value matrix and eigen vector matrix, and the three eigen vectors with three largest eigen values are used to do dimension reduction, reduce the 15 dimension features into 3 dimension features, the code is attached in appendix. In my approach, each feature is standardized by extract its mean and divided by its standard deviation[3] to obtain a better performance.

(b) Advanced Texture Classification

Classification is a common problem in machine learning, there are two patterns to solve this problem, one is supervised learning, in which labeled training set is used to tell the classifier the characteristic of each class, and the trained classifier is used on the test set, where the label of each sample is unknown, and the label of each sample is given by the trained classifier. The other is unsupervised learning, where test set is classified directly by the embedded structure without the help of training set. Both of these two classification methods could be used to classify the energy vector extracted by the procedure above.

Class Name	Grass	Blanket	Brick	Rice
Predict Label	1, 2, 6, 11, 12	3	4, 7, 10	5, 8, 9
True Label	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9

Table 1: Best Unsupervised Classification Result by K-means

For unsupervised learning, the k-means algorithm is used to classify this test set. K-means algorithm is a classification algorithm which at first select k data points randomly in the test set as k classification centers and classify other data points according to the distance between every data points and the selected classification centers, then the classification centers are re-computed by the average of classification results. Then new classification centers are assigned by the average vectors above. That's one iteration. After several classification, the points close to each other are classified as the same class. The process of classification is completed. The code is attached in appendix. K-means algorithm a simple unsupervised classification algorithm, the computational complexity is low and the classification result is reasonable. But it is not stable, the number of classes should be given ahead of time, and the algorithm is very sensitive to the

initial classification center, the outcomes also vary every test. The best result which obtain an error rate of **16.7%** is listed in table 1, there are only two samples are wrongly classified. Both the original feature and the feature after dimensional reduction by PCA could achieve this result. The difference between the original data and the dimensional reduced data could achieve the same best result, in which there are two samples in the test set are wrongly classified. The difference between these two processes is that the data after feature dimension reduction is more stable, after several experiments, the results achieved by feature dimension reduced data are less likely to vary than the original data, and the percent that the best results achieved by feature dimension reduced data is higher than the original data, which means PCA could reduce the noise in the original data and preserve the most of information embedded in the original data.

For supervised learning, support vector machine and random forest are used to classify the test set. Firstly, SVM is used to classify the test set, the built-in model in MATLAB is used. In the beginning, the energy of training set is abstracted, both the original 15-dimension energy vector and the 3-dimension energy vector after PCA are used to train the classifier. Besides, both linear support vector machine and kernel support machine are used to classify the test set, the result is shown in Table 2.

Kernel	Data	Class	Grass	Blanket	Brick	Rice	Error Rate
Linear	Original	Predict	1,6,11,12	2, 3	4, 7, 10	5, 8, 9	8.3%
		True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
	PCA	Predict	1, 6, 12	10	2, 4, 7	3,11,5,8,9	33.3%
		True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
Gaussian	Original	Predict	1,6,11,12	2, 3	4, 7, 10	5, 8, 9	8.3%
		True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
	PCA	Predict	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	**0%**
		True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	

Table 2: Classification Result with Different Kernel

Table 2 shows that SVM with Gaussian kernel could totally correctly classify the test data after dimensional reduction. But when using the linear kernel, PCA reduces the effect of classification, which is because even through PCA could reduce the noise in the original data, but it reduced the dimension of the original data, making SVM harder to find the linear decision boundary in the dimension reduced subspace, so the error rate of using PCA data is higher than using the original data when using linear kernel. However, the difference when using Gaussian kernel is that this kind of SVM is using kernel trick, which could project the original data into a feature space with higher dimension, and then the linear decision boundary could be easily found there, then the linear decision boundary in the higher-dimension feature space could be projected back to the original space, forming the non-linear decision boundary. So with the help of PCA, the noise in the original data is reduced which makes the difference between classes are more distinct, and

then with the help of kernel trick, the non-linear decision boundaries are found, and thus achieve 0% error rate.

Random forest is another supervised learning method, random forest is consist of several decision trees, each decision tree is trained in different sequence of features, and the cross entropy is minimized for each tree. The final result is voted by different decision trees. Table 3 shows the results by different random forests differ in the number of decision trees, all results in table 3 is achieved by the original data.

Number of Decision Tree	Class	Grass	Blanket	Brick	Rice	Error Rate
1	Predict	1,6,11,12	2, 3, 10	4, 7	5, 8, 9	16.7%
	True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
2	Predict	1, 6, 12	2, 11	3, 4, 7	5,8,9,10	16.7%
	True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
3	Predict	1,6,11,12	2, 4	7, 10	3, 5, 8, 9	25%
	True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	
4	Predict	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	**0%**
	True	1, 6, 12	2, 3, 11	4, 7, 10	5, 8, 9	

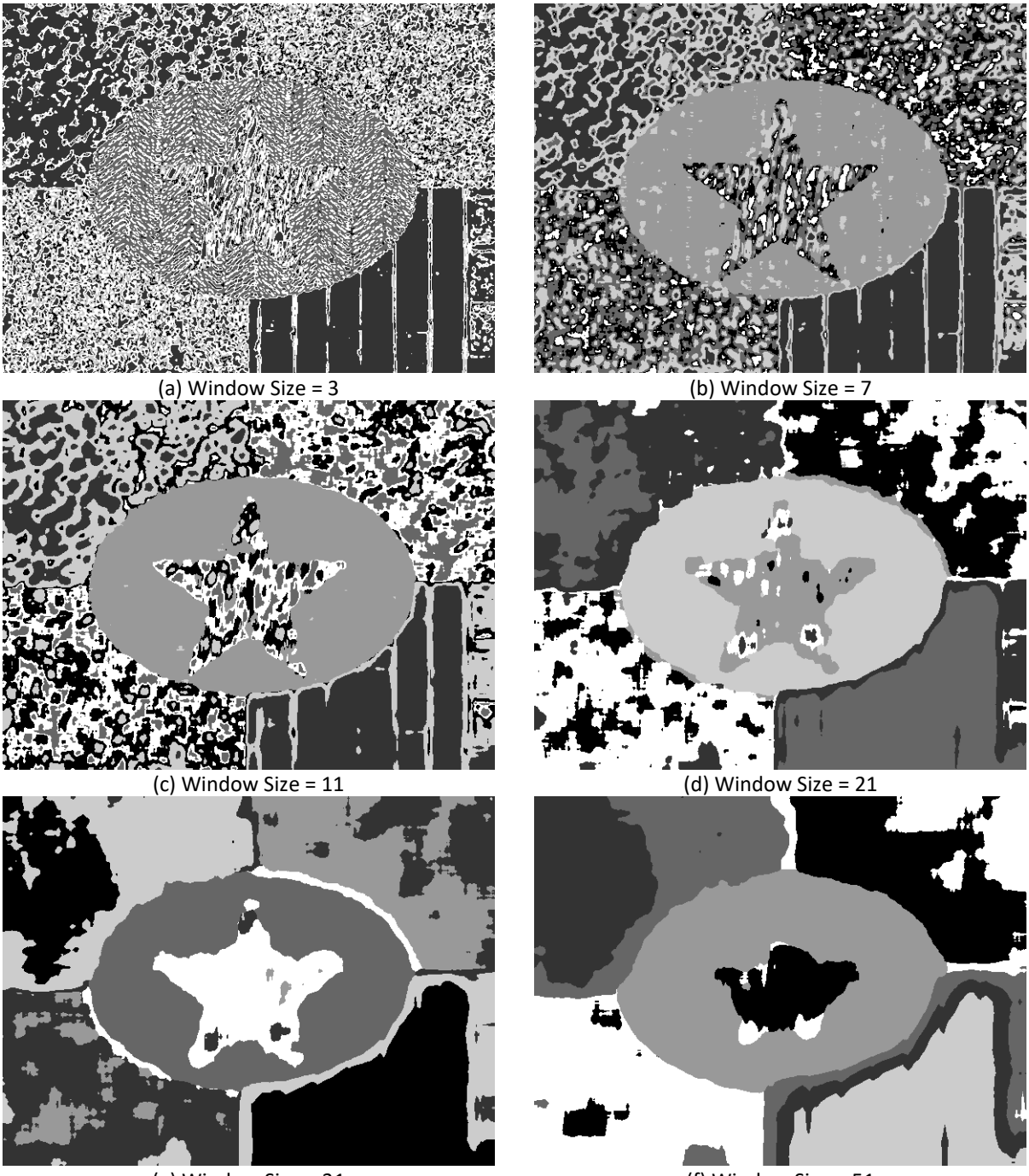
Table 3: Classification Result with Different Random Forest

Table 3 shows that when the number of decision tree is large enough, in this case, larger than 4, all the test set samples could be correctly classified. In this experiment, if the PCA data is used, the number of decision trees to achieve 0% error rate is larger than that of using the original data. The reason of this is that we reduce the dimension of the original data to 3 dimensions, there are still some critical information for decision tree to decide which the samples belong to. As for the noise, as the number of decision trees arises, the tolerance to noise also increases. That's why the result is different with that of SVM.

(c) Texture Segmentation

Texture Segmentation is another application of Laws filter. In an image where different kind of texture are mosaiced together, to figure out the region occupied by different texture, the feature map generated by the 25 Laws filter is obtained, then the pairwise results are averaged, so for the image, a 15-dimension feature map is generated. In order to figure out the region, not only the information of each pixel point should be taken into account, but also their neighborhood region, so for each pixel, a window is used to average the energy in the neighborhood, and thus the information for a single pixel is obtained. Next, because the L5L5 kernel does not have a zero mean, so the feature extracted by it is not useful for texture segmentation, all of the values are used to normalize the energy vector of each point, then the L5L5 dimension is discarded. After getting the 14-dimension normalized energy vector for each point in the image. Because the points with same texture has similar energy distribution, so in feature space, the points with same texture should near each other and far away from those from other textures. So, they could be

classified by k-means algorithm, and different classes are assigned with different gray scale. By changing the size of the window averaging the neighborhood window, different results could be obtained, as is shown in Fig. 3.



(a) Window Size = 3 (b) Window Size = 7

(c) Window Size = 11 (d) Window Size = 21

(e) Window Size = 31 (f) Window Size = 51

Fig. 3 Result of Texture Segmentation with Different Window Size

Fig. 3 shows that the larger the window size is, the blurrier the edges segmenting textures are, and also, the window size effect the classification result. With a proper window size, the textures

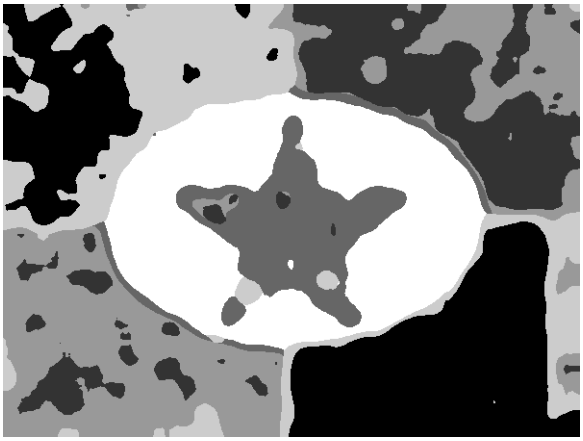
could be segmented properly with the same texture lies in the same region, and the edges between textures could be preserved well. According to [1] and the experimental result, the most proper window size is 31. In Fig. 3(e), a single gray scale in the same texture region majors in comparison with Fig. 3(a), (b), (c) and (d), and the boundary is relatively clearer than Fig. 3(f).

(d) Advanced Texture Segmentation

From the result above, the best result achieved by the procedure above is not so good. So PCA is used to refine the result. After getting the 14-dimensional energy vector, using PCA to do dimension reduction. Based on the results obtained in (c), all of the following results are obtained on the basic of window size = 31.

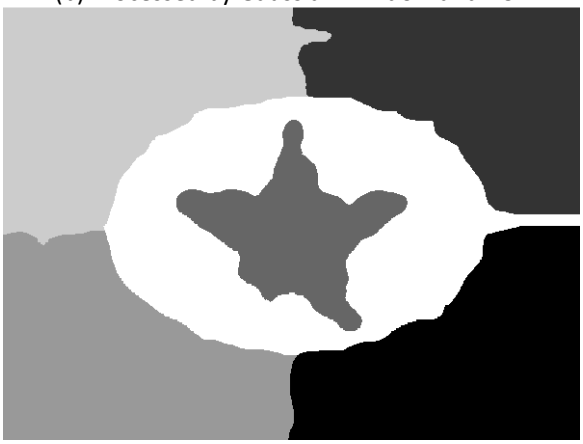
To get the most proper remaining dimension by PCA, several experiments are carried out and the final result shows that after PCA, if the first few principle components with the largest eigenvalue are preserved, the results do not vary too much. So, in this question, the final dimension remained is 5 and thus preserved the most information of the textures and the noise could be eliminated. The result by PCA is shown in Fig. 4(a). From the result, we could see that after PCA, the holes inconsistent with the feature is smaller, which means the classification result is better than using the original data.

(a) Processed After PCA



(b) Processed by Gaussian Window and PCA

(c) Setting all Small Region to -1



(d) Final Result

Fig. 4 Advanced Texture Segmentation

Next, inspired by the principle of filter, when computing the energy of a single pixel, the window is Gaussian weighted. In this way, the energy could focus more on the area nearby and also take the remote points into consideration. After several trail, the most proper sigma of Gaussian weight is selected, the result is shown in Fig. 4(b). From the result, we could see that the number of holes is less than the outcome using uniform window, it is better to do future process, such as filling in the holes by the pixels around it.

The last step is hole filling. To eliminate the holes in the image, every connected region in the image is counted, through observation, the area of the texture are the six largest regions in the image, so once the area of each hole is counted, the smaller ones could be filled by the pixels in the surrounding region. To count the area of each hole, breadth first search is used. After getting the area of each hole, the smaller ones are set to -1, as is shown in Fig. 4(c). Then traverse the image for several times until there is no -1 in the whole matrix, the value of the pixels is filled with the surrounding regions of the hole, the final result is shown in Fig. 4(d). From the result we could see that in general, the location of each features is recognized, but the edges segmenting features are hard to identify, which is caused by the size of window, the smaller the window size is, the sharper the edges are. But if the window size is too small, the features could not group in the similar region of the feature space. The best trade-off I could find is shown here.

Problem 2: Image Feature Extractors

(a) Salient Point Descriptor

1. The SIFT is robust to image scale and rotation, as well as location change, i.e. translation[4].

2. SIFT achieves scale invariance by searching for stable features in all possible scale, using scale space. The keypoint locations are detected by finding extrema in difference of two nearby scales separated by a constant multiplicative factor.

Rotation robustness is achieved by orientation assignment to each keypoint. Keypoints are selected by the Local Extrema Detection and then further selected by eliminating the keypoints with low contrast and those on edges. For the remaining keypoints, a consistent orientation to each keypoint is assigned, which makes the keypoint could be represented relative to this orientation and therefore achieve invariance to image rotation.

Location invariance is achieved by creating orientation histograms over 4*4 sample region. So that a gradient sample could shift up to 4 sample positions while still contributing to the same histogram. Besides, to avoid abruptly changes as a sample shifts smoothly from being within one histogram to another or from one orientation to another caused by boundary affects, trilinear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins.

3. SIFT achieve its robustness to illumination change by Local Image Descriptor, the local image descriptor of a region is obtained by adding some histogram of its neighborhood region, when adding, each feature vector is normalized to unit length, which could cancel contrast changes.

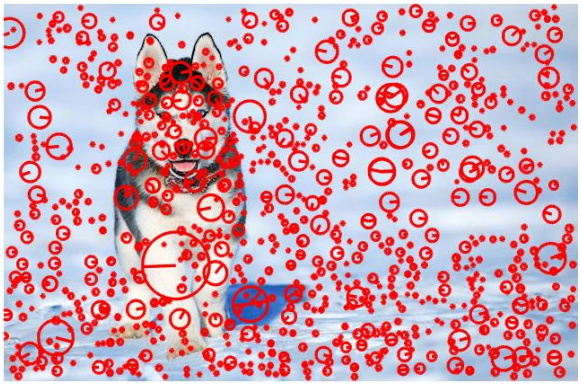
Besides, because the method uses gradient, so the brightness change will not affect the feature vector. To eliminate the difference caused by non-linear illumination changes, the value in the unit feature vector larger than 0.2 is reduced, then the feature vector is re-normalized, which makes the change in relative magnitudes caused by non-linear illumination changes less likely to affect the feature vector.

4. DoG has an efficient function to compute. Besides, DoG could give a close approximation to scale-normalized LoG $\sigma^2 \nabla^2 G$, whose maxima and minima produce the most stable image features compared to other possible image functions.

5. 4*4*8 = 128, each keypoint in the experiment uses a 4*4 array of histogram with 8 orientation bins in each.

(b) Image Matching

(a) All Keypoint in Husky_3



(b) All Keypoint in Husky_1

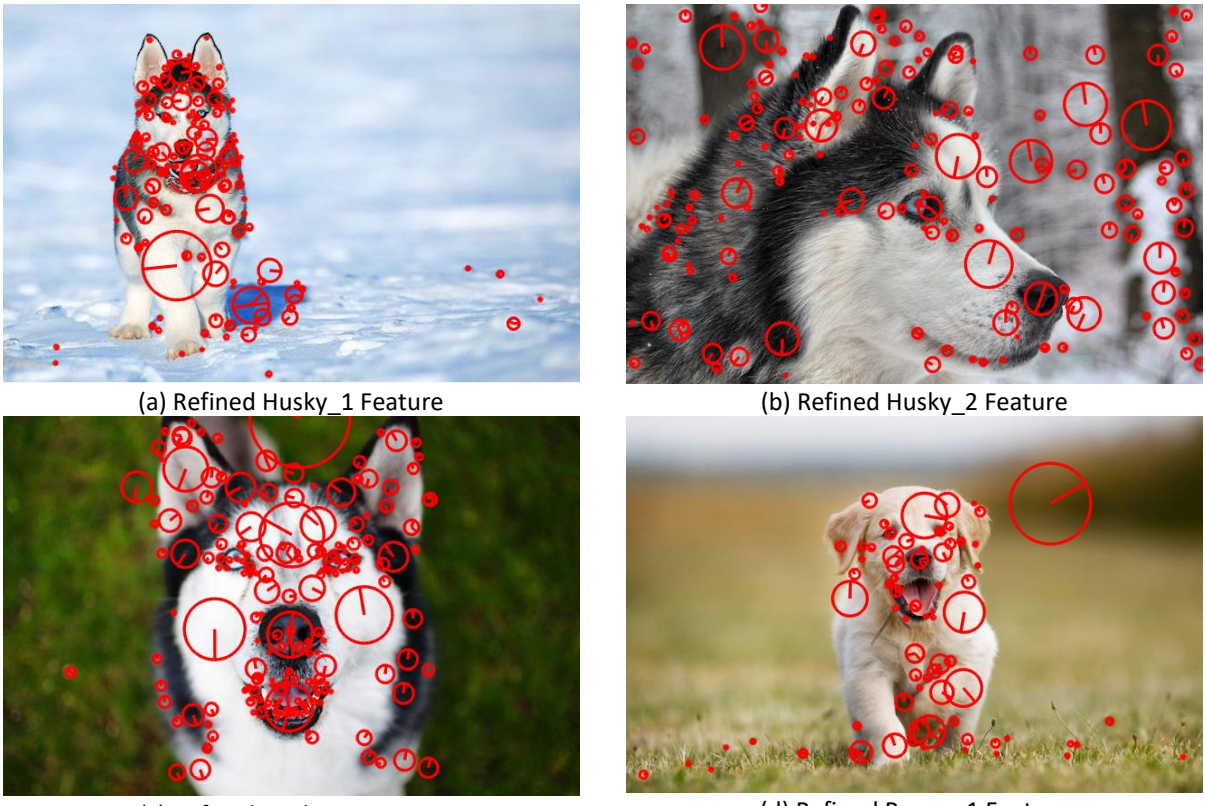
(c) Keypoint with Largest Scale in Husky_3

(d) Nearest one in Husky_1

Fig. 5 Keypoints in Husky_3 and Husky_1

1. Apply SIFT algorithm to Husky_3.jpg and Husky_1.jpg, all the keypoints extracted by SIFT algorithm is shown in Fig.5 (a) and (b), from the image, we could see that there are a lot of features in the background. In vl_sift() algorithm, a parameter called PeakThresh makes a great difference when extracting features in images. This parameter filters peaks of the DoG scale space that below a threshold[5], if the threshold is too small, there will be a lot of noise features in the background which are meaningless to be extracted, and if the threshold is too large, the

number of extracted features will decrease, leading to lack of features for post processing, another parameter named EdgeThresh taking control of the threshold to eliminate edges during detection below the threshold also have influence on the feature extraction results, but in this problem, there are almost no sharp edges in the image, different from pictures with a lot of edges such as building, so this parameter does not as effective as PeakThresh, to find the keypoint in Husky_3 and its best matching in Husky_1, the threshold should be large enough to eliminate other feature points with smaller scale. After selecting proper threshold, the keypoint with the largest scale is extracted and shown in Fig. 5(c) and (d), the radius of the circle denotes the scale in scale space of the keypoint and the direction of the radius denotes the direction of the key-point. To find the closest neighboring keypoint in Husky_1, the distance between feature vectors are computed. Because the data type processed by vl_sift() is Uint8, which will lead to the negative value be replaced by zero, and thus lead to the mismatching, so, before matching, the data type is converted to double type.



(a) Refined Husky_1 Feature

(b) Refined Husky_2 Feature

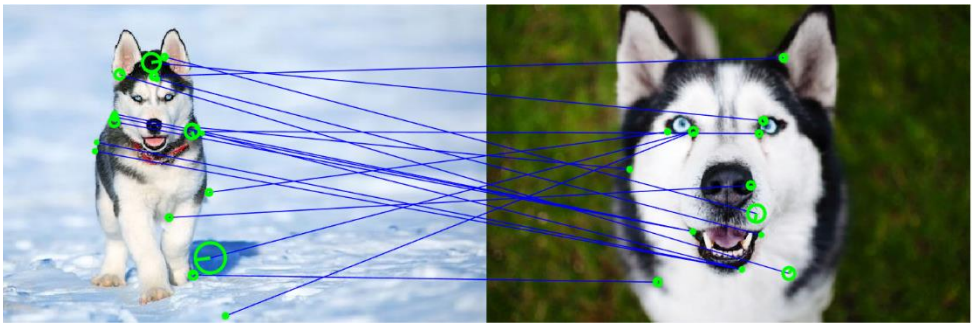
(c) Refined Husky_3 Feature

(d) Refined Puppy_1 Feature

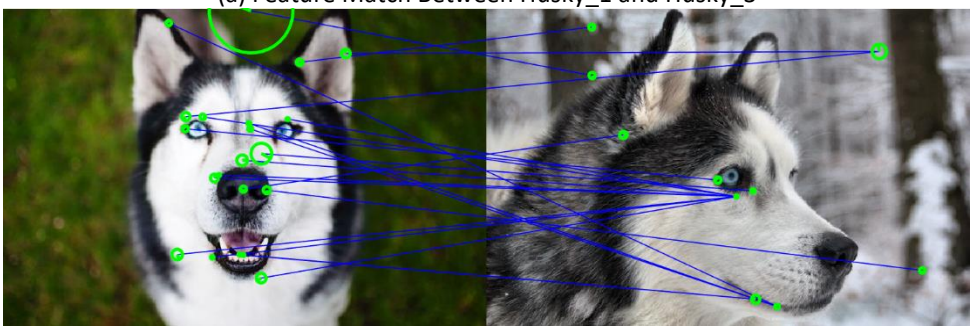
Fig. 6 Refined Feature Keypoint

Fig. 5(c) shows that the keypoint with the largest scale lies on the nose of the dog, intuitively, this point has the contrast of this point is relatively large in the whole image, and the gradient in its neighborhood is also very large, because the area around the nose is its white face, while the noise itself is black. To find the best matching point on Husky_1, the threshold of Husky_1 should also be set in a proper value, because the scale of Husky_1 is smaller than Husky_3, which means

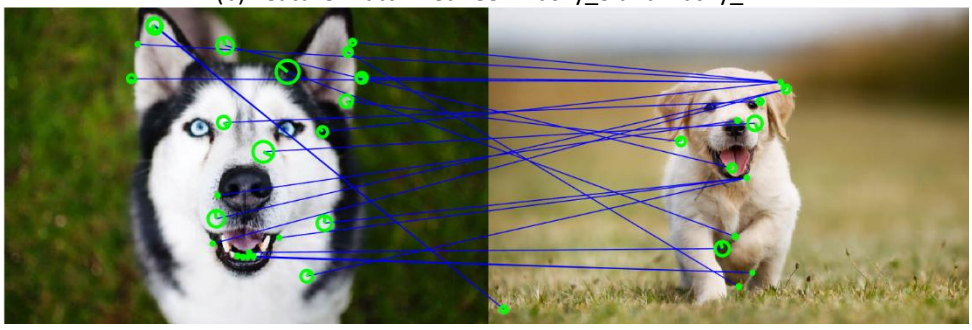
its easier to find such a region with larger gradient such as its eyes, mouth or the shadow around it. From the result, we could see that SIFT have a good scale invariance, translation invariance, and some tolerance to 3D transformation because the face in Husky_3 is larger than that of Husky_1, and they lies in different location on the two images, the view is also a little different, one is from the overlook, and the other is from front view.



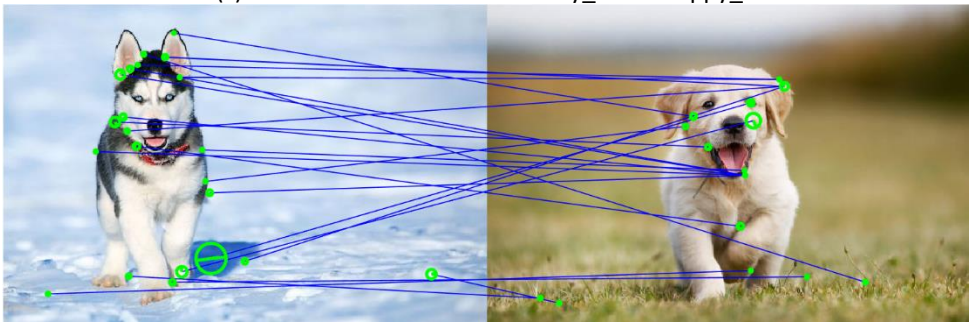
(a) Feature Match Between Husky_1 and Husky_3



(b) Feature Match Between Husky_3 and Husky_2



(c) Feature Match Between Husky_3 and Puppy_1



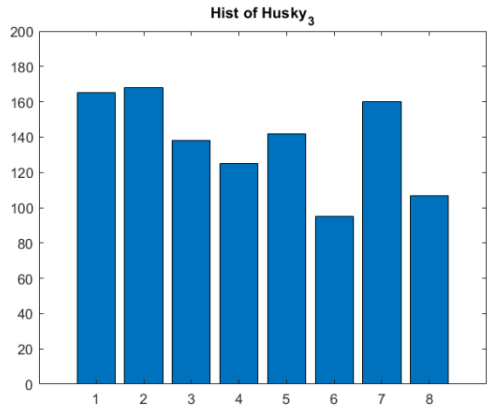
(d) Feature Match Between Husky_1 and Puppy_1

Fig. 7 Feature Match Result Between Different Pairs

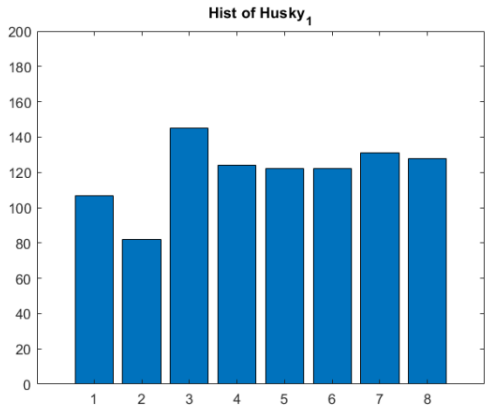
2. Before doing the image matching of these mentioned pairs, because there are a lot of unnecessary keypoints extracted by SIFT algorithm in the background of these images, so several thresholds are tried so that most of the features could lies on the body of these dogs, and also leaving enough number of features for image matching, the features extracted properly is shown in Fig. 6. Fig. 6 shows after tuning the threshold, most of the features lie on dogs, because the background of Husky_2 is more complicated than other picture, so it is harder to reduce features in the background, but from the image, most of the features are also concentrate on the dog.

Fig. 7 shows the required image matching result. Fig. 7(a) shows matching between Husky_1 and Husky_3 they are the same object with different scale and different view point, because of the different scale, the matched points are almost from the body edge of Husky_1 to the eye edges and mouth edge of Husky_3, because they are similar from a local view, and the background of Husky_1 is white, which will also make confusion when it interact with the black fur of Husky_1. There are also few points match well, which are from the edge of face of Husky_1 to that of Husky_3. Fig. 7(b) shows the matching between Husky_3 and Husky_2, they are the same object with similar scale but different view point, it's a kind of 3D distortion, it works better, even though the two photos are taken in different views, from the matching, we could see that the eyes and mouth are matched well, but there are also some confusion matchings caused by the background of Husky_2, and because the view point of Husky_3 makes the fur of the bottom of its face look like its eyes, so there are also some mismatching. Fig. 7(c) shows the matching between Husky_3 and Puppy_1, they are similar objects with different scale and different view point. This match is the worst in the four matching results, because the two image shares least in common. For example, the teeth of Husky_3 are matched to the shadow of Puppy_1, the mouth of Puppy_1 is matched to the edge of ear of Husky_3. But there is also something in common in the edges of the two dogs. Fig. 7(d) shows the matching result between Husky_1 and Puppy_1, they are about similar objects with the same scale and the same view point. Even they are not about the same object, because of the same scale and the same view point, there are also something the two images share in common, such as the edge of body and the edge of nose of the two dogs, which are matched well. But other parts are mismatched. The matching results above shows that the features extracted by SIFT has some invariance to 3D distortion, typically caused by changing of view point, and also some tolerance to scale changing, but still not powerful enough to extract the most significant feature which could decide whether two images are about the same object or not. Another shortcoming of SIFT algorithm is the value of threshold should decided by experience and the result is very sensitive to the threshold deciding peak of gradient and edges mentioned in (a), but the computation of SIFT is fast, and from Homework 3, we have already seen that SIFT works well when matching photos taken in the same circumstance with some regular view-point changing to form the panorama, which shows SIFT works well when doing some simple matching between the totally same objects, even with some change of view point or some change of illuminance.

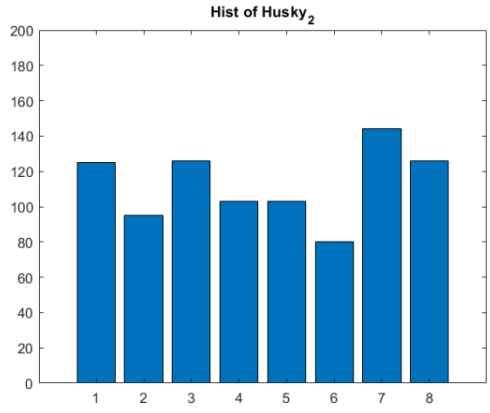
(c) Bag of Words



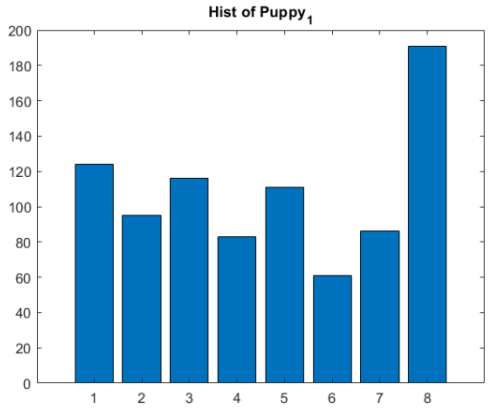
(a) Histogram of Husky_3



(b) Histogram of Husky_1
Distance from Husky_3: 114.9



(c) Histogram of Husky_2
Distance from Husky_3: 99.5

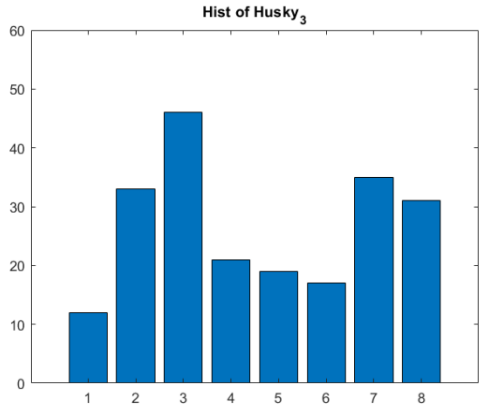


(d) Histogram of Puppy_1
Distance from Husky_3: 154.6

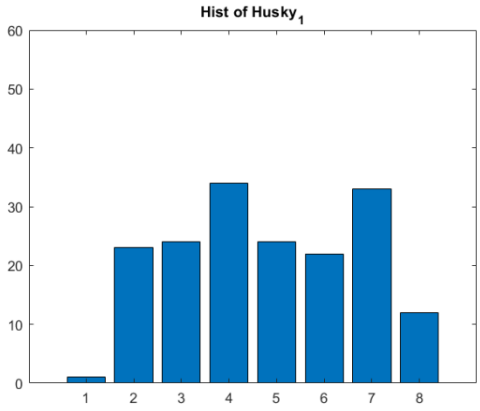
Fig. 8 Histogram (Codebook) by Whole Features

Bag of word is a common model in Nature Language Processing. Typically, a large amount of training articles is applied and different types of articles are put together to form a codebook by some keywords appearance in these articles, typically, different type of articles could have different frequency distribution on these keywords, so a test article could be classified by the frequency of the occurrence of these keywords in the codebook formed by the training set. This principle could also be used in image classification. To do image classification by bag of words, the first step to do is to extract some keypoints from each training image, the keypoints could be extracted by SIFT algorithm. These keypoints are extracted and then put together to form a whole feature space, and k-means algorithm is used here to find the cluster centers of these keypoints, and thus the codebook is formed by these cluster centers. Similar to natural language processing, different type of image tend to have different keypoint distribution on these cluster centers, so for each test image, the SIFT features are extracted and fit on these cluster centers, to observe the intersection of histogram of different images, a vector indicating the total number of occurrence of these SIFT features on the cluster centers could be obtained, and comparing this

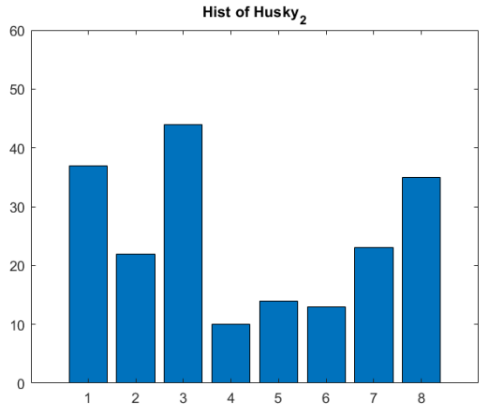
vector with those formed by articles in the training set, the image could be regarded as representing similar objects with the nearest one in the training set.



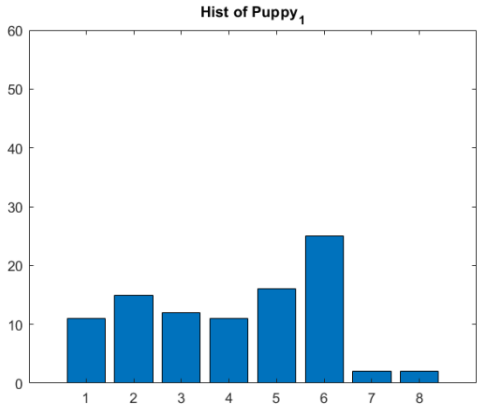
(a) Histogram of Husky_3



(b) Histogram of Husky_1
Distance from Husky_3: 39.93



(c) Histogram of Husky_2
Distance from Husky_3: 25.37



(d) Histogram of Puppy_1
Distance from Husky_3: 55.81

Fig. 9 Histogram (Codebook) by Refined Features

For this problem, the feature of these 4 images are extracted and put together, they are clustered by k-means into 8 clusters. To show the effect of SIFT, two groups of histogram are shown, one is obtained by using the whole set of SIFT features in Fig. 5(a) and (b), shown in Fig. 8, the other is obtained by using the refined feature in Fig. 6, shown in Fig. 9. The clusters are arranged by the distance between the cluster centers and the original point.

From Fig. 8 and 9, we could see that the distribution between Husky are similar in comparison with Puppy. The distance between these vectors are also calculated and shown above, which also suggests that the distributions of Husky's tend to share some common distribution. This is caused by the invariance of SIFT feature point. Fig. 8 shows that even through it is hard for us to match these features because of the background noise, the difference caused by this kind of noise is relatively smaller than the difference caused by different objects, features from the same objects are also share similar distribution on the whole. So, the distance between Husky_3 and the other

two pictures of Husky is smaller than the distance between Husky_3 and Puppy_1. The difference between Husky is not only caused by the different point of view, but also caused by the different background. Take Husky_2 as an example. There are some trees with black trunks and white snow on it, in multiscale viewpoint, these changes are likely be regarded as feature points by SIFT, but the background in Husky_1 or Husky_3 do not have such kind of changes. Another reason caused the difference is the different scale, more details of the face of Husky, which provide most of SIFT features, are shown in Husky_2 and Husky_3, the face of a Husky have a higher resolution, but Husky_1 tend to give a general look to a Husky, which caused the distance between Husky_3 and Husky_2 is less than that between Husky_3 and Husky_1. Fig. 9 with a more precise outcome also prove this.

Reference

[1] Laws KI. Textured image segmentation. University of Southern California Los Angeles Image Processing INST; 1980 Jan.

[2] https://en.wikipedia.org/wiki/Karhunen%E2%80%93Lo%C3%A8ve_theorem

[3] https://stats.stackexchange.com/questions/69157/why-do-we-need-to-normalize-data-before-principal-component-analysis-pca

[4] David G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60(2), 91-110, 2004.

[5] https://www.vlfeat.org/overview/sift.html

Appendix:

Code of PCA

```
function res = PCA_dim(X, dim)
      X = matDotDiv(X - matMean(X), matStd(X));
      covar = matMul(matTranspose(X), X);
      red_dim = dim - 1;
      [V,D] = eig(covar);
      V = V(:, end-red_dim: end);
      V = V(:, end:-1:1);
      res = matMul(X, V);
      res = matTranspose(res);
end
```

Code of kmeans

```
function y = k_means(X, k)
      [m,n] = size(X);
      rand_index = randperm(n);
      center = X(:, rand_index(1: k));

      y = zeros(n, 1);
      MAX_ITER = 50;
      for iter=1: MAX_ITER
          for i=1:n
              min_dis = Inf;
              for j=1:k
                  cur_dis = matNorm(X(:, i) - center(:, j));
                  if cur_dis < min_dis
                      min_dis = cur_dis;
                      y(i, 1) = j;
                  end
              end
          end
          new_cen = zeros(m,k);
          new_num = zeros(1,k);
          for i=1:n
              new_cen(:, y(i,1)) = new_cen(:, y(i,1)) + X(:, i);
              new_num(:, y(i,1)) = new_num(:, y(i,1)) + 1;
          end
          for i=1:k
              new_cen(:, i) = new_cen(:, i) / new_num(:, i);
          end
          center = new_cen;
      end
end
```