EE569 Digital Image Processing

# HOMEWORK#4 Report

**Issued: 3/7/2021** **Due: 11:59PM, 3/28/2021**
**Name: Siyu Li** **ID:2455870216**
**E-mail: lisiyu@usc.edu**

## Problem 1: Texture Analysis (35%)

## (a) Texture Classification – Feature Extraction (15%)

### I. Approach and Procedures

Texture is a kind of pattern that follows a certain distribution of pixels. Laws Filter **[1]** can be applied to filter a image and get a feature map which can distinguish different textures and patterns from it. There are 5 1D Laws filters which are L5, E5, S5, W5 and R5 respectively. L5 is a low pass filter which can eliminate or restrict high frequency components such as abrupt pixel change, noise and edges. E5, S5 and W5 are bandpass filters which only allow the component with a certain range of frequency to pass. And R5 is highpass filter which maintains high frequency information such as edge and contour. To perform Laws filter on a image which is a two-dimensional object, we apply tensor product operation between these 5 different basic filters to get the 25 2D filter masks.

To get the feature map. We apply the 25 2D filters on the original image to get a 25-layer feature map whose single layer has the same size as original image. And to get the energy feature vector of the feature map, all pixel values on each layer of the feature map are powered, summed up and normalized to output a single constant. Therefore, for each image, we have a 25-dimension feature vector.

To analyze the discriminant power of each feature, we use the 25-dimension feature vectors of all images to calculate inter-class and intra-class variance and measure the discriminant power by the ratio between inter-class and intra-class variance. Inter-class variance is a measurement of how easy one class can be separated from other classes by a certain feature. Meanwhile, intra-class is a measurement of how close the samples in the same class cluster together by a certain feature.

Different features have different capability to classify and too many features can lead to undesired computational complexities and too much redundant information of little use. Therefore, PCA (Principle Component Analysis) is applied to do feature reduction. PCA can discard features with less information and maintain those with more information and here we reduce the 25-dimension feature to 3-dimension by it. PCA is realized in the code by myself with the function of **PrincipleComponentAnalysis.m** and standardization is applied to get better result.

After feature extraction by Laws filter and feature reduction by PCA, nearest neighbor classifier is applied to do classification on the test set images. Nearest neighbor classifier is a supervised machine learning algorithm which performs classification by using the label of training set point with the nearest distance as the class of the test set point. In this problem, to avoid the impact of scale and deviation difference among different classes, Mahalanobis distance is used to calculate the distance between test set point and training set point.

## II. Experimental Results

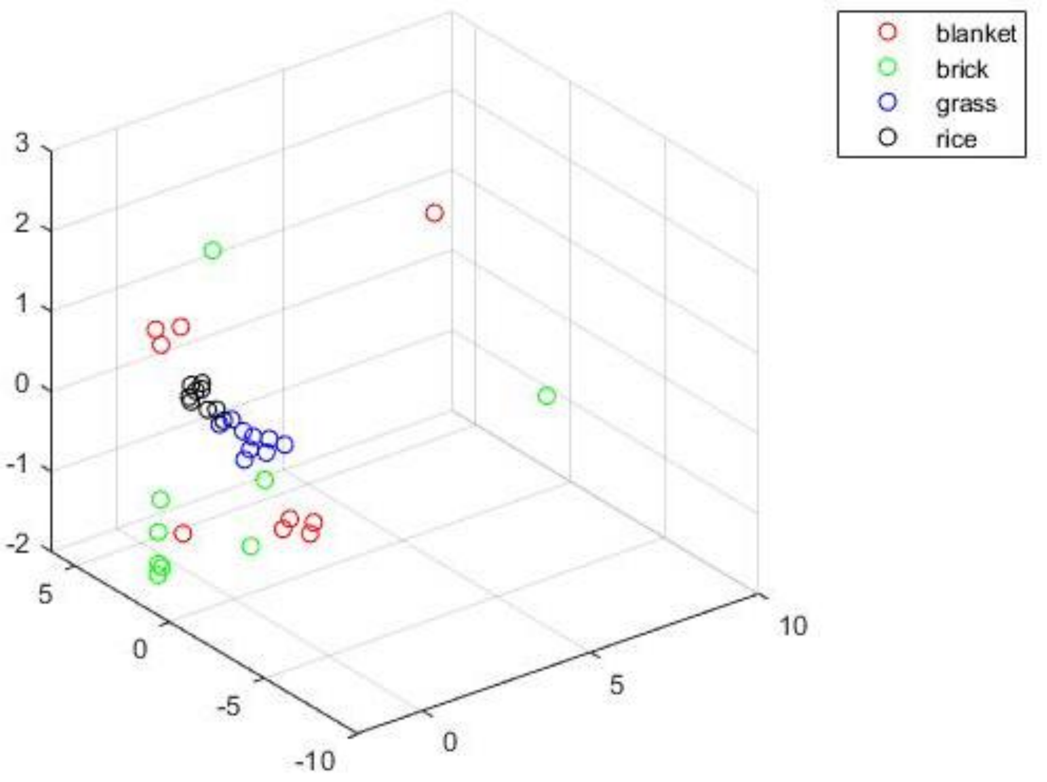


**Fig 1.1 3D Scatter Plot of Feature Distribution in Each Class After PCA**

**For the result of feature discriminant power**, the feature with the **largest** discriminant power is calculated by the filter of **L5E5 and E5L5** while the feature with the **least** one is calculated by **L5W5 and W5L5** filter.

For the reduced 3D feature vector of the training set, they are plotted in the 3D scatter plot in **Fig 1.1**. From Fig 1.1, we can observe that the features of brick and blanket are more scattered while those of grass and rice cluster closely together, which means the class of brick and blanket will be more likely to be misclassified than the other two. This phenomenon in blanket and brick may result from the more complicated texture pattern and bigger disparities among images in the same class of the training set.

As for the nearest neighbor result on the test set, the original output of test set image

**labels by my function is 3 2 1 3 1 1 2 4 3 2 4 2**. Here **1,2,3,4 respectively represent class blanket, brick, grass and rice**. Meanwhile, **the original labels of 12 images in the test set are 3 2 1 3 1 1 2 4 3 2 4 4**. For the reason that the order of image labels has not been changed during calculation, we can compare the two label series directly to distinguish whether the classification of each image is correct. Finally, **we find in the 12th image, the original image label is 4 (rice) while the classified label is 2 (brick)**. This may result from the scattered distribution of feature in brick, which makes it possible in some case that the distance from test set point with label rice closer to points of brick in training set. **In conclusion, only 12th image is misclassified and the total classification error rate is 1/12 (8.3%).**

## (b) Advanced Texture Classification --- Classifier Explore (20%)

### I. Approach and Procedures

For the **unsupervised** learning part, kmeans method which is unsupervised and iteration based is applied to do the classification. Kmeans is implemented by initializing a certain number of cluster centroid as the center of each class in feature space and then giving label of other point in the test set based on the distances to the original cluster centroid. After all the point in the test set get their labels, new cluster centroids are calculated based on the average of labels of all points classified previously. Then the two steps above are repeated back and force until the result converges. Due to the randomness in assigning the initial label, every time the classified label to each class may be different so we should adjust the classified label to original label manually to do comparison. Here original 25 dimensional features and 3 dimensional features acquired by PCA are applied to do the classification by the kmeans functions by myself.

For the **supervised** learning part, SVM (Support Vector Machine) and RF (Random Forest) are applied to do the classification. Random Forest is implemented by combining several decision trees together to avoid overfitting. Meanwhile PCA is implemented by using the kernel trick and points on the decision boundary to do classification. In this problem, feature reduction is applied to reduce feature from 25 dimensions to 3 dimensions to avoid overfitting and both of the two methods are implemented by function from Matlab.

### II. Experimental Results

For the **unsupervised learning part**, labels calculated by **kmeans by myself** based on **25D feature** are **2 4 4 2 4 1 3 2 2 4 1 1**. Here **label 1,2,3,4 are respectively rice, grass, brick and blanket**. While **test set label based on label order of this classification are 2 3 4 2 4 4 3 1 2 3 1 1**. The error rate is **1/3 (33.3%)** in this classification. The major errors exist in the misclassification between brick and blanket, which results from the very scattered and overlaid feature distribution of the two classes in Fig 1.1.

After **feature reduction by PCA**, labels calculated by **kmeans of myself** based on **3D feature are 2 2 3 2 2 4 1 4 2 1 4 4**. Here the labels **1,2,3,4 are respectively brick, grass, blanket and rice**. The **test set label based on label order of this classification are 2 1 3 2 3 3 1 4 2 1 4 4**. The error rate is **1/4 (25%)** which is a noticeable improvement compared to that of 25D feature before PCA. Because PCA maintains the most useful features and discards the less useful ones.

Then, for the supervised learning part **SVM**, labels calculated by **SVM (Linear)** by 3D feature are **3 3 1 3 2 4 2 4 3 2 4 4**. Here **1,2,3,4 respectively represent class blanket, brick, grass and rice**. The original labels of test set are **3 2 1 3 1 1 2 4 3 2 4 4.** The error rate is **1/4 (25%)**. When using the **Gaussian kernel**, the labels calculated are **3 1 1 3 1 1 2 4 3 2 4 4**. The error rate is only **1/12 (8.3%)**. The reason of this result is that when feature is reduced, it is difficult for SVM to find a linear decision boundary while by using Gaussian core which using kernel trick to project data points to higher dimension feature space to find a boundary and then project back to original space we can get an ideal decision boundary very easily.

For the supervised learning part **RF**, **labels** calculated by RF with **3 trees** are **3 1 1 3 1 1 2 4 3 2 4 4.** The original labels of test set are **3 2 1 3 1 1 2 4 3 2 4 4.** We can find that the error rate is **1/12 (8.3%)**. In this case, the large number of decision trees in random forest alleviates the issues of overfitting very well and performs great.

## Problem 2: Texture Segmentation (30%)

## (a) Basic Texture Segmentation (20%)

### I. Approach and Procedures

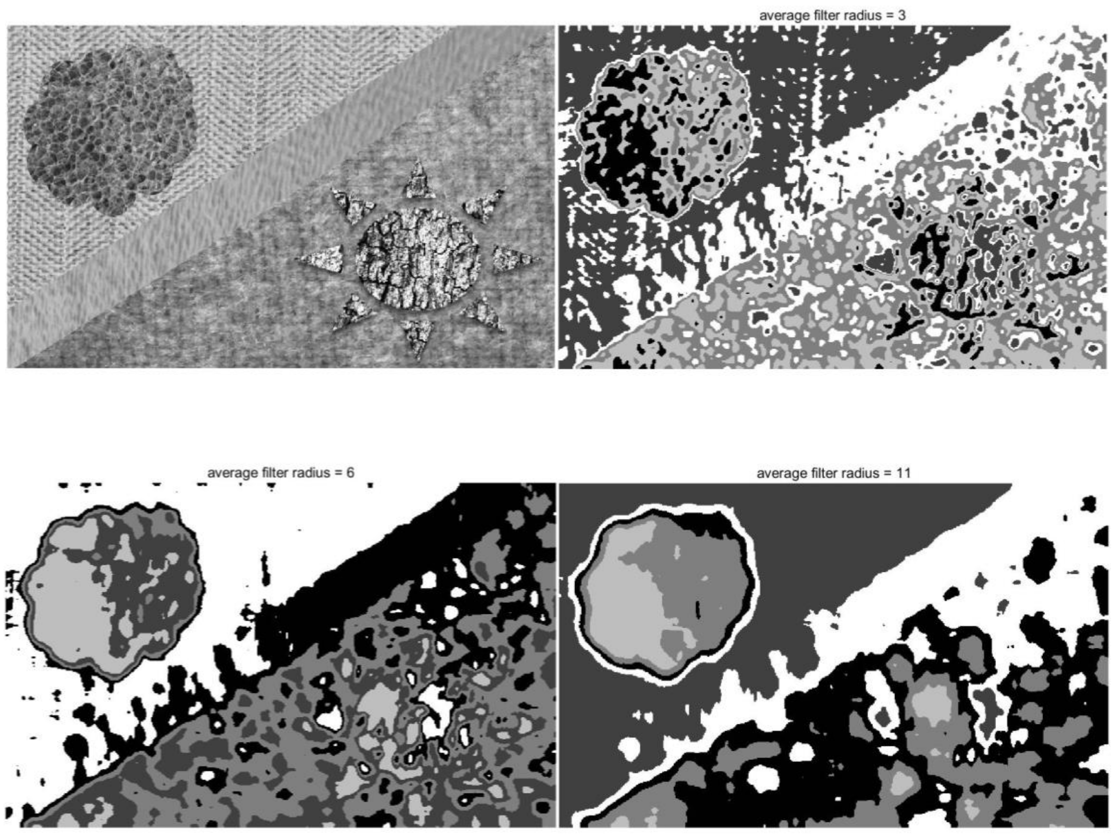
For the basic texture segmentation, we first calculate the feature map by 25D Laws filter which is illustrated in Problem 1 previously. And based on the result acquired, **subtraction of global mean is not used** to extract as many feature information as possible. Due to the property that the class of texture is determined by the pixel pattern of a certain-sized region rather than a certain pixel, filtering is applied to the feature map and we get a filtered feature map. Here the filter radius plays a very important row. With small filter radius, it is not enough to distinguish the texture pattern and many pixels will be misclassified even if edge information will be well preserved. Meanwhile, with large filter radius, pixels are less likely to be classified into the wrong class but edge information will be damaged severely which makes original object in the image hard to identify.

Then, for the reason that the L5L5 filter core does not have a zero mean and feature extracted from it is useless for classification, the L5L5 feature from the filtered feature map is used to normalize the entire filtered feature map and **L5L5 layer is removed** from the 25D filtered feature map and **only 24D filtered feature map remains**.

To input the data with proper size, data reshape is applied to transform m*n*24 filtered **24D** feature map to a feature matrix with size 24*(m*n) which is the input of kmeans classification function. Here m and n are respectively row and column numbers of original image. After kmeans classification, we get a label vector of size 1*(m*n) which contains the labels assigned to each pixel. To show the image correctly by the label vector, we first reshape it into a label map sized m*n and then assigned different values to different labels to get proper luminosity and contrast in the final output image.

## II. Experimental Results

We applied the segmentation feature map filtering step by both average filter and Gaussian filter with sigma=10 with different radius. For the reason that kmeans algorithm is an unsupervised learning method, the output label of each class is random, so the gray scale of same label in the result of average and Gaussian filter output may be different. The result of **Average filter output** is in **Fig 2.1** as follows.


average filter radius = 3
average filter radius = 6
average filter radius = 11

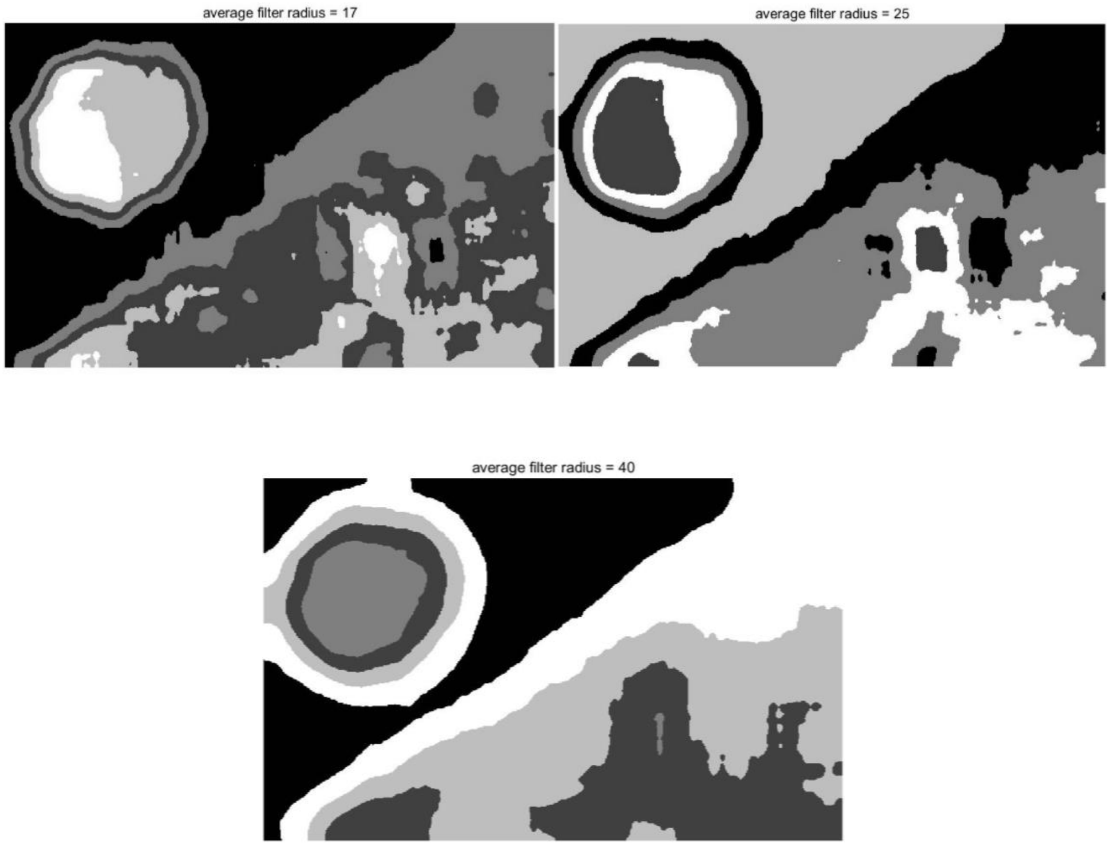
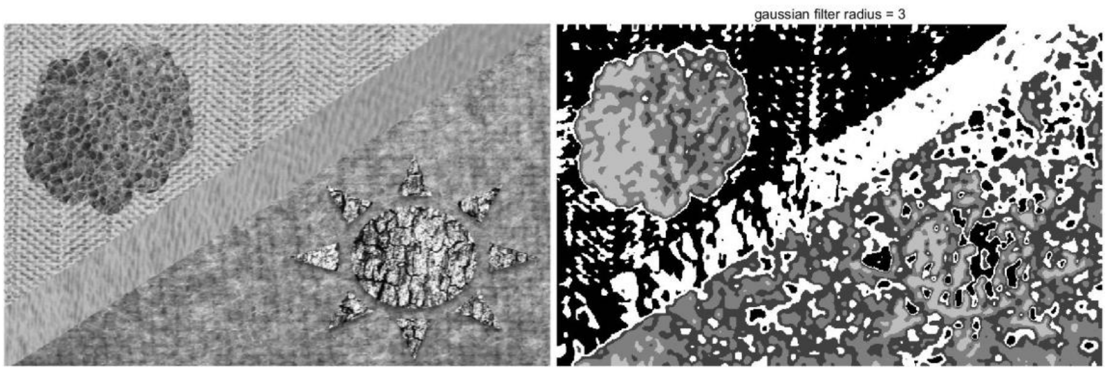

average filter radius = 17
average filter radius = 25
average filter radius = 40

**Fig 2.1 Texture Segmentation Result by Average Filter with Different Radius**

From the Fig 2.1 we can observe that the result of segmentation is not so ideal and we cannot see the original object clearly. Result with smaller filter radius tend to have more detailed information when compared with original composite image but there are lots of small holes which corrupt the segmentation result seriously. Result with larger filter radius preserve less edge and detail information but there are fewer small holes. The result of Gaussian filter with sigma=10 is just a little bit better than the average filter. It is shown in Fig 2.2. Therefore, more advanced method such as PCA needs to be implemented to get better result.



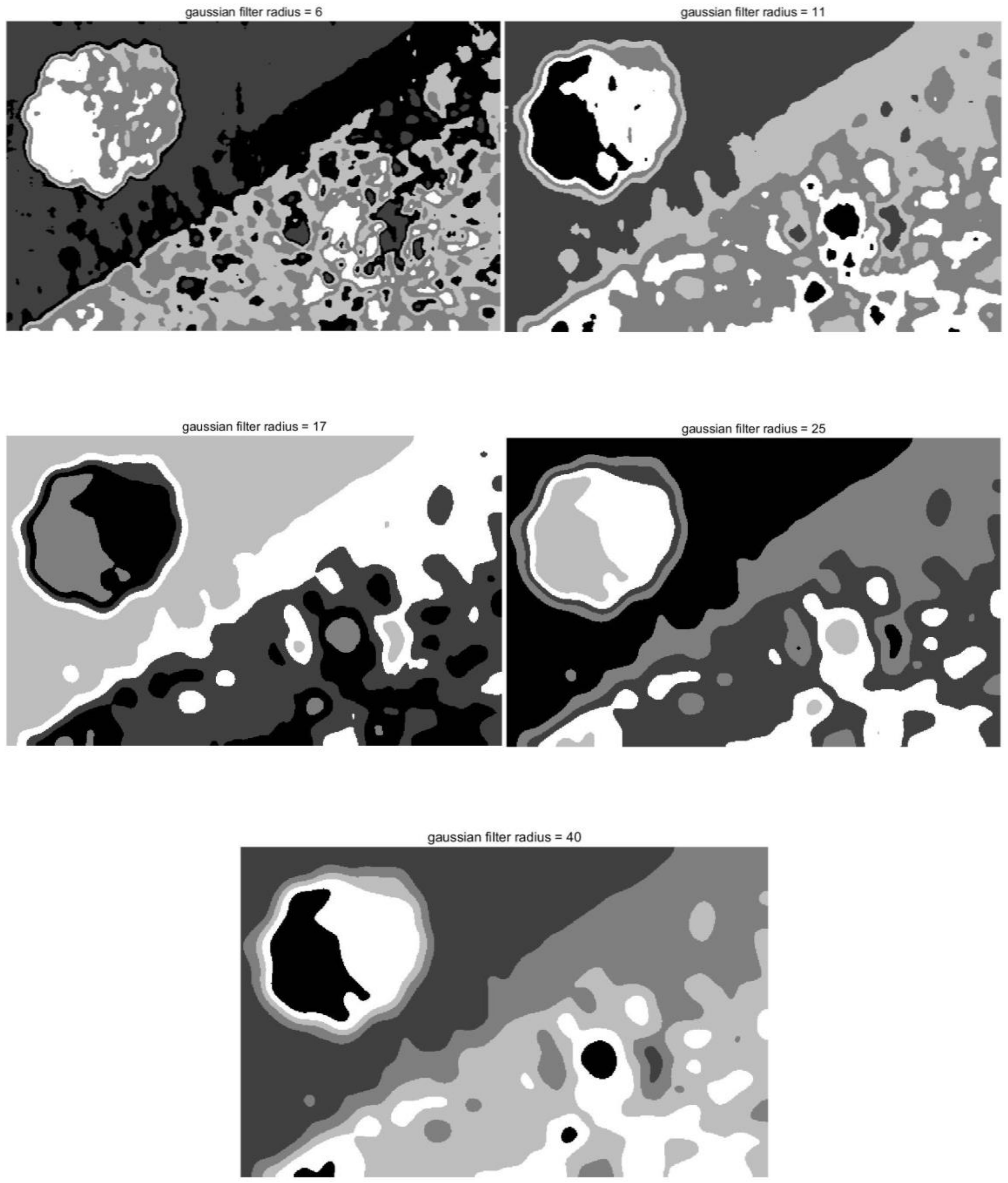


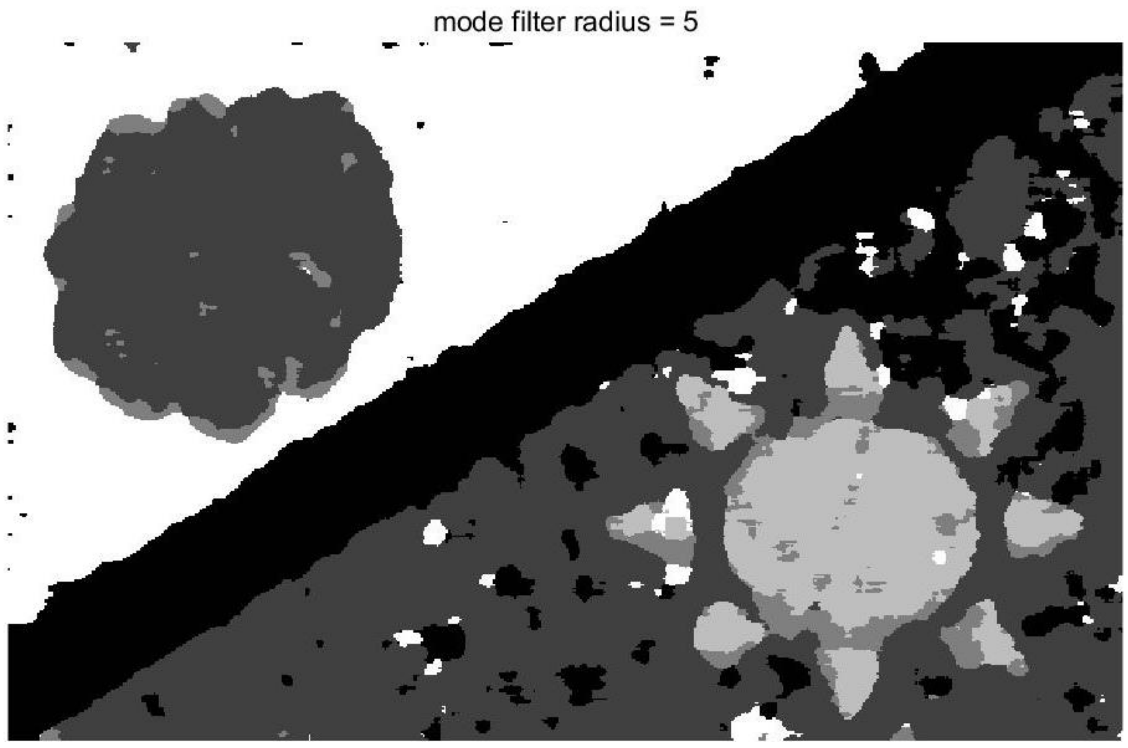
**Fig 2.2 Texture Segmentation Result by Gaussian Filter (sigma=10) with Different Radius**

## (b) Advanced Texture Segmentation (10%)

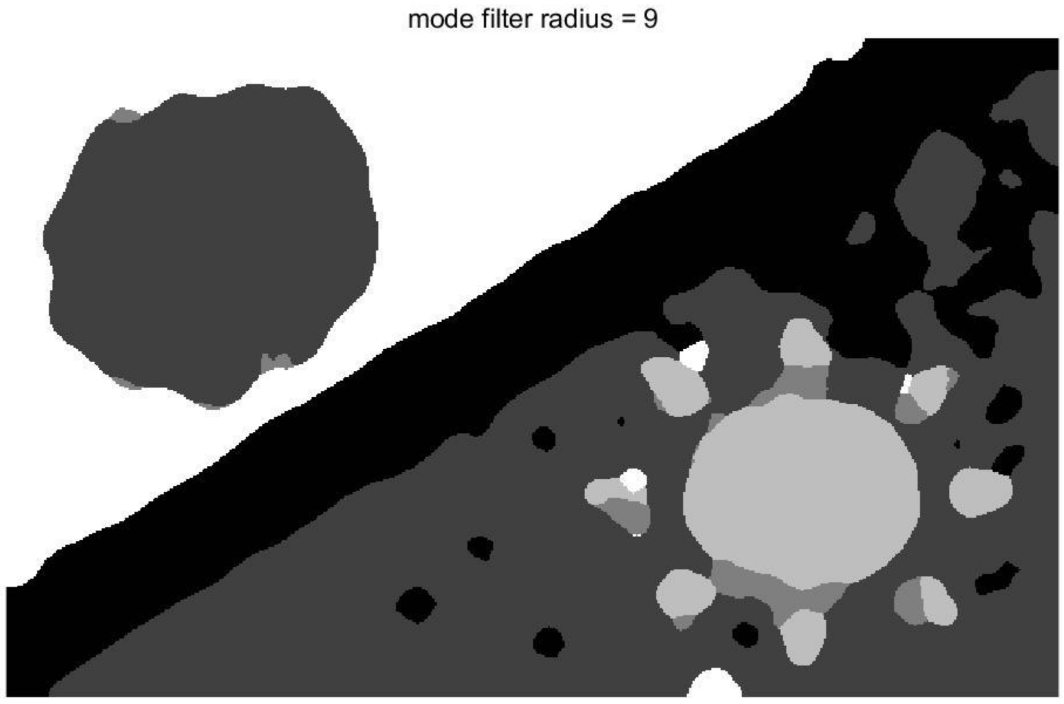
### I. Approach and Procedures

In this step, global mean subtraction is not applied in order to get more useful feature information. And we used 24D filtered feature information which is the same case as the previous problem2 a. For the improvement made in this step, **PCA** is applied to maintain useful features and discard relatively not useful features to avoid misclassification. After get the segmentation result enhanced by PCA, **we applied mode filter which select the most common label in the neighbor of a certain pixel as the actual label**. This can eliminate the small holes on the original segmentation result effectively. There is a trader off between edge information preservation and number of small holes when different mode filter radius is applied. With small mode filter radius, only the very close neighbor is considered, this prevents the elimination of holes with relatively bigger size even though it keeps more detailed information which makes the object easier to recognize. While with larger mode filter radius, the holes are well fixed but detailed information is also impaired. Therefore, tuning the parameters of the filter radius and the number of features left after PCA is extremely important in this step.

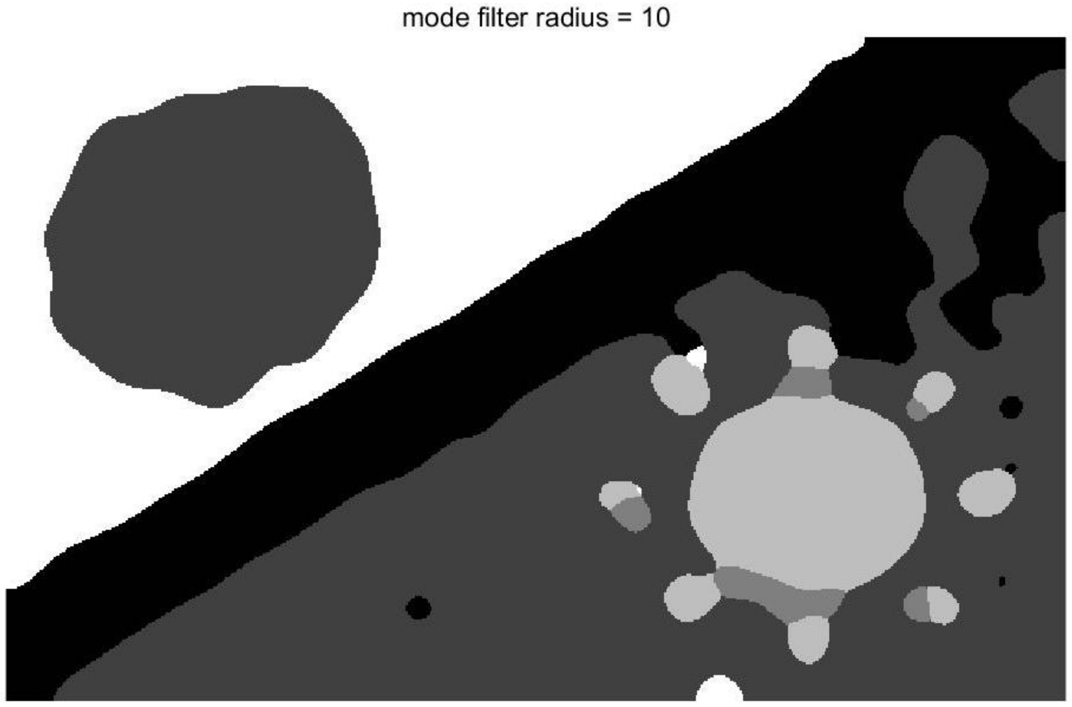
## II. Experimental Results



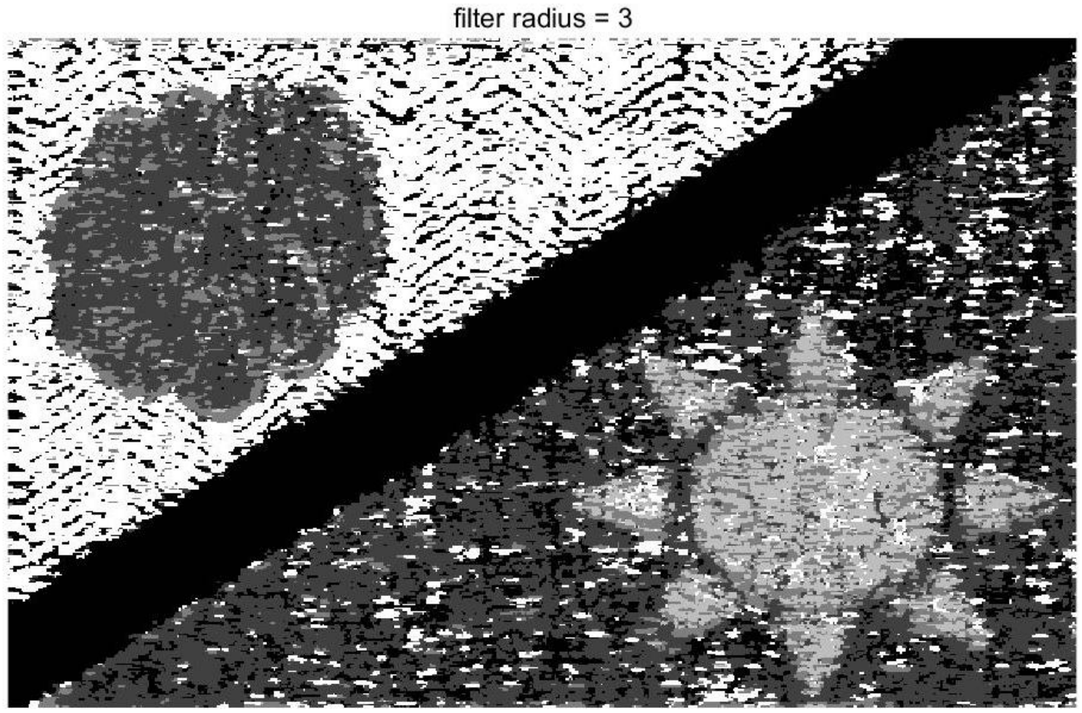
**(a) Segmentation Result After PCA with Fix of Mode Filter Sized 5**



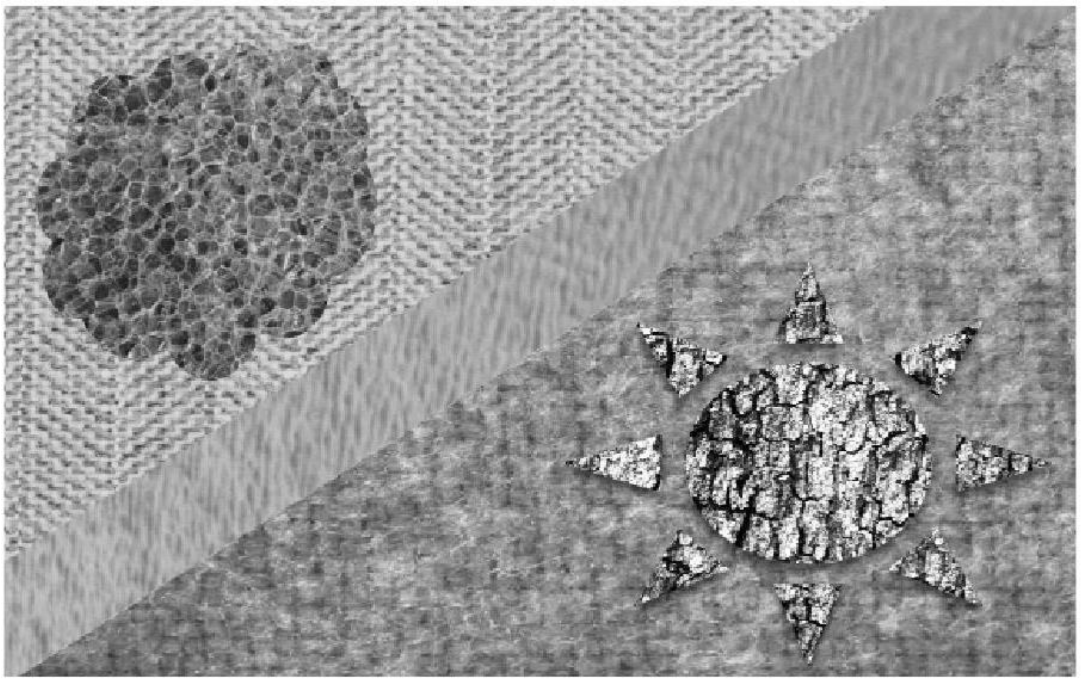
**(b) Segmentation Result After PCA with Fix of Mode Filter Sized 9**



**(c) Segmentation Result After PCA with Fix of Mode Filter Sized 10**



**(d) Segmentation Result After PCA with Feature Map Filter Sized 3**



**(e) Original Composite Image for Comparison**

**Fig 2.3 Results of Advanced Texture Segmentation**

After trying lots of filter size on the original feature map and number of features remained after PCA, we find that by using Gaussian filter with radius=3 and sigma=10 and PCA with 3 largest features remained we can get the **optimal original segmentation result after PCA which is shown in Fig 2.3 (d).** When compared with result in Fig 2.1 and Fig 2.2, the segmentation result after PCA is much better because we can see the contour of object very easily especially in the sun area with several small affiliated regions and the edge position is nearly the same as the original composite image in Fig 2.3 (e). Meanwhile in Fig 2.1 and Fig 2.2, the contour and edge information are totally messed up compared to the original image.

The remaining problem is to fix the small holes which can make the segmentation result noisy. Mode filter which selects the most common label in the neighbor of a certain pixel as the actual label with different filter size is applied to the optimal original segmentation result in Fig 2.3 (d) and we get **Fig 2.3 (a)(b)(c).** When the mode filter radius is 9 and 10, we can get the best result which balances the contour and object information and number of holes well. With the larger mode filter size such as 9 or 10, the boundary of two adjacent regions is also enhanced well for there is not any noise on the boundary separating line. **Therefore, my final advanced image segmentation results are in Fig 2.3(b) and Fig 2.4(c).**

In conclusion, the result of final segmentation is not perfect. The major reason is that the features Laws filter provides cannot satisfy the requirement of segmenting such a complex image because in image segmentation based on traditional two step method the first step feature selection is fundamentally important. Good features will have a decisive impact on the success of segmentation. And for this image, it is also more difficult than average because the affiliated regions of the sun in bottom right are very hard to be segmented and there is so much similarity between the texture in different regions. Even though the textures overall are different in different regions, if we look at feature on a certain very tiny part, the texture looks nearly the same. This can account for why there are so many holes which represents misclassified pixels.

## Problem 3: SIFT and Image Matching (35%)

## (a) Salient Point Descriptor (Basic: 10%)

### I. Approach and Procedures

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

The SIFT **[2]** is robust to image scaling operation, image rotation and image location change.

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

SIFT achieves scale invariance by searching stable features throughout all possible scales. The key point location is acquired by searching the extrema on the difference map of two neighbor scale maps.

Rotation robustness is achieved by calculating orientation of each key point. Key point is acquired by local extrema detection firstly and then further detection is applied by removing key point with lower contrast and on the edge. For the existing key point after key point selection, orientation information is calculated by the ratio of differences in two directions from the difference map acquired from two nearby scale image. The orientation constant assignment to each key point endows the invariance to image rotation.

Location invariance is achieved by calculate orientation histograms over the 4*4 sampled region. Therefore, a sample can move up to 4 positions and still contribute the same histogram.

**3. How does SIFT enhances its robustness to illumination change?**

SIFT achieve its illumination robustness change by local image descriptor which obtained by absorbing its neighborhood region histogram. During the process of calculating local image descriptor, all the feature vectors are normalized to unit length. This can eliminate the negative impact of illumination disparity among different images. Moreover, this method applies gradient value rather than the original value, which also prevent the result from illumination change. And to remove the influence by non-linear illumination changes, some values in the previous unit feature vector are reduced and then feature vector is normalized again.

**4. What the advantages that SIFT uses Difference of Gaussians (DoG) instead of Laplacian of Gaussians (LoG)?**

DoG is an approximation to the LoG so DoG is relatively computationally efficient.

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

The original vector sizer is 128 (4*4*4). The key point in the paper apply a 4*4 array of histogram which contains 8 orientation bins each.

## (b) Image Matching (Basic: 15%)

### I. Approach and Procedures

**1.**Feature detection by SIFT is applied by vl_sift **[3]** function in the toolbox to the grayscale image of Dog_1 and Dog_3. Then, to pick the key-point with the largest scale in Dog_3 and find its closest neighboring key-point in Dog_1, we first sort the value in the 3rd row in feature matrix output by vl_sift and get the index of the feature with maximum scale. And we use this index to locate the position of this feature in the descriptor matrix. Then we use the function of vl_kdtree build and vl_kdtreequery with the values of largest scale feature in descriptor3 and the grayscale Dog_1 image to search the matched feature point of largest scale point of Dog_3 in Dog_1. Finally, plot the matched points in the two images respectively.

**2.** Matching and showing the corresponding SIFT pairs between the Dog_1 and Dog_3, Dog_3 and Dog_2, Dog_3 and Cat, Dog_1 and Cat are implemented by vl_sift and vl_ubcmatch.

## II. Experimental Results

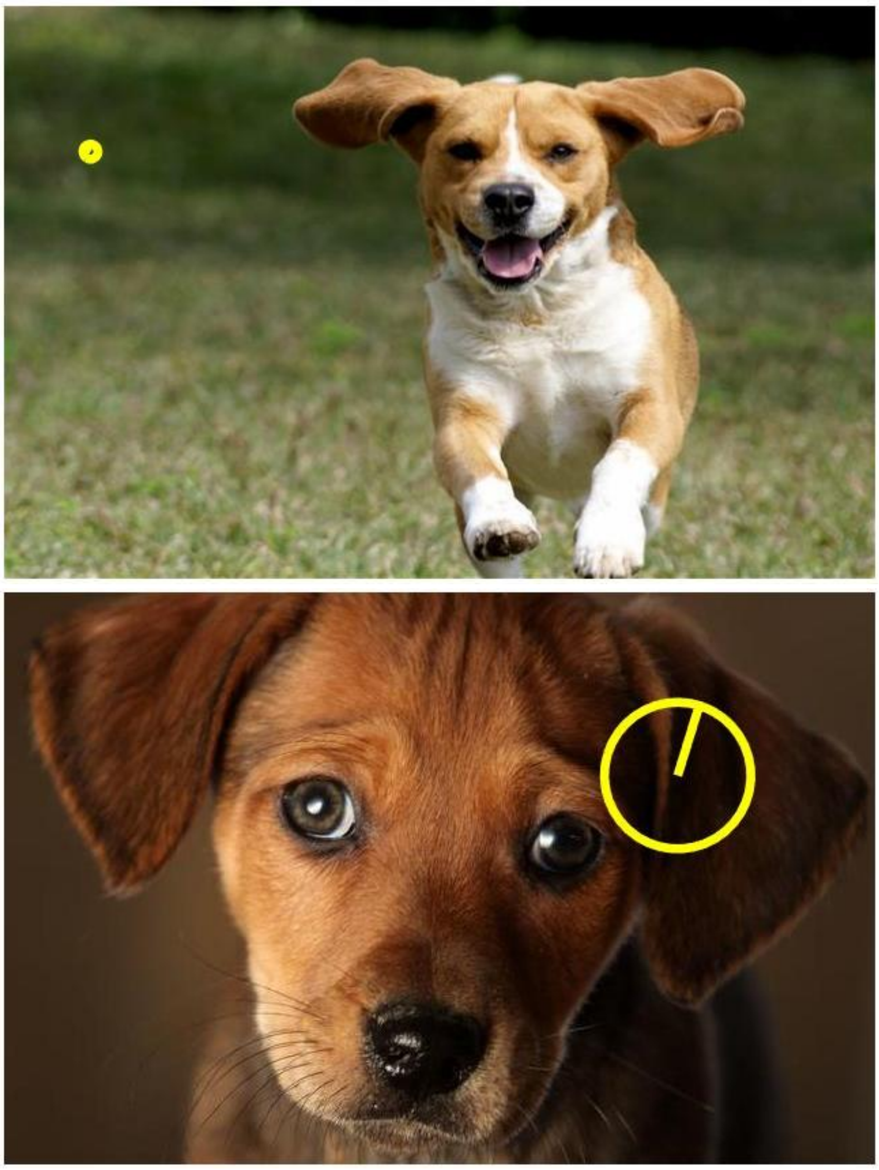


**Fig 3.1 Largest Scale Point in Dog_3 (bottom) and Its Matched Point in Dog_1 (upper)**

**1.** From the result in Fig 3.1 we can observe that with the huge difference between the two images, even if we searched the matched feature points calculated by very robust SIFT method, the two points in the image still look entirely different. There is no similar feature or pattern at all.

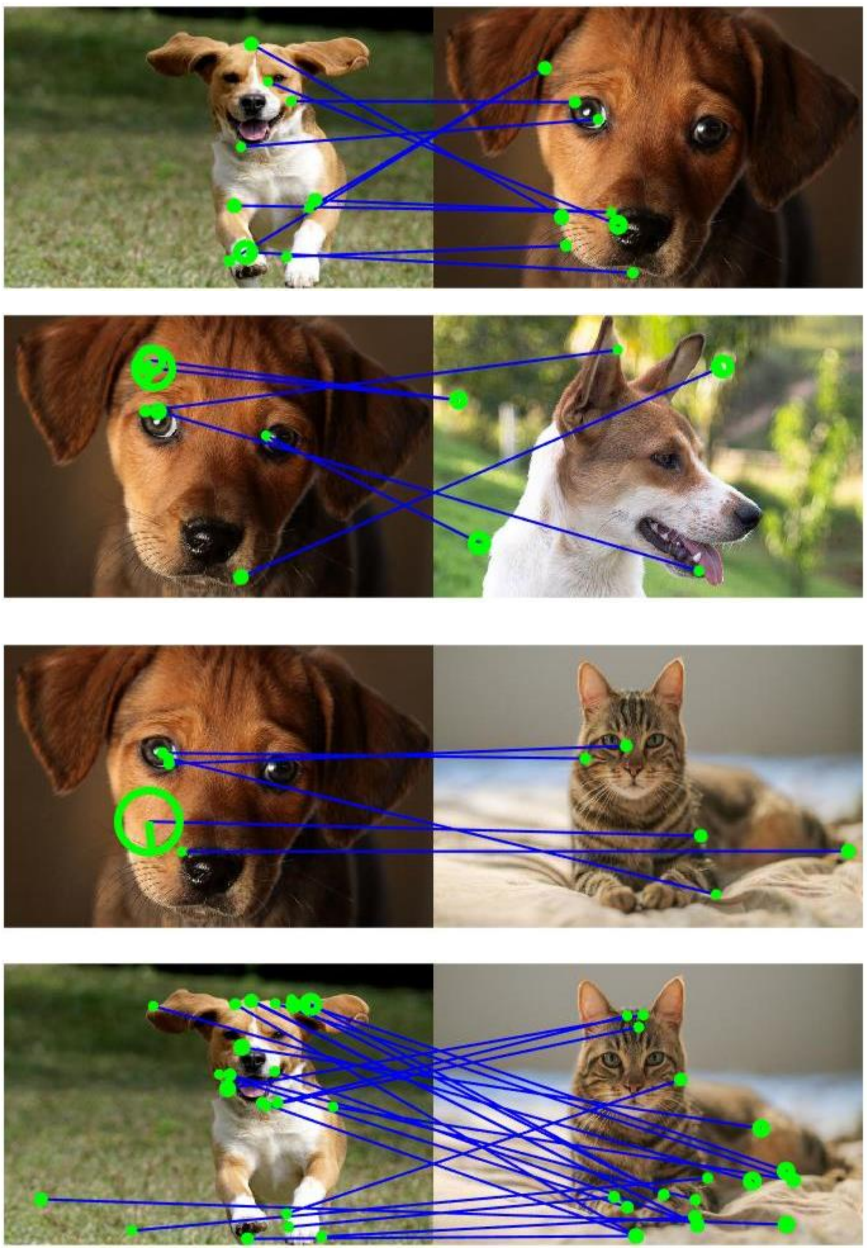


**Fig 3.2 Matched SIFT Pairs between Dog_1 and Dog_3 (1st), Dog_3 and Dog_2 (2nd), Dog_3 and Cat (3rd), Dog_1 and Cat (4th)**

**2.** In this step, proper parameters and threshold are set to show better match result. From Fig 3.2, we can observe that some of the matched points are correct such as points on the eyes and the contour of dog. But due to the huge disparity among images and background noise, some of the points are not matched well. For example, the points on the background are matched to the points on the dog or cat.

## (c) Bag of Words (10%)

### I. Approach and Procedures

Bag of words which is a method in natural language processing regards each kind of feature as word in codebook and all the feature set as the codebook itself. In this problem, descriptors calculated by vl_sift is used by the kmeans to get the cluster center matrix which contains codeword. Then the dimension of cluster center matrix is reduced by PCA from 128 to 20. In order to match Dog_3's codewords with other images, we need to give order to the center vector matrix because the output labels of kmeans classifier are entirely random. We can align all the feature in different images together by this center reorder process. Then, we can just calculate the distance of feature points in each image to the cluster centers which can be regarded as the codewords in codebook and count each kind of feature to build a feature histogram of each image.

## II. Experimental Results

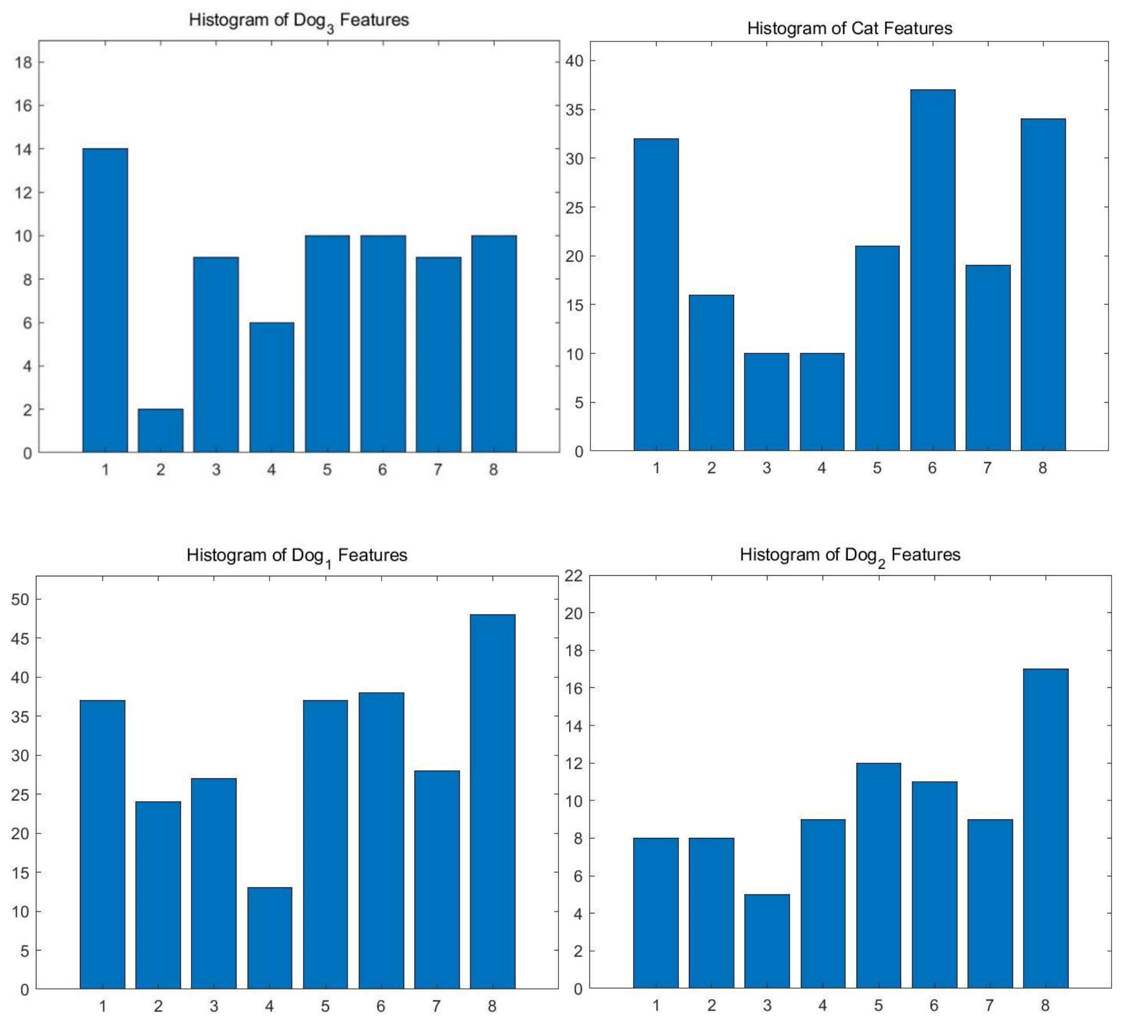


**Fig 3.3 Matched Codeword Histogram of Image Dog_3, Cat, Dog_1 and Dog_2**

The result of Matched Codeword Histogram of Image Dog_3, Cat, Dog_1 and Dog_2 is shown in Fig 3.3. We can observe that histograms of dog features have some distribution similarity even if due to detail difference the histogram distribution is not exactly the same. And the histogram of cat features is a little bit different from that of dog's even if they share some common feature with dog. The more distribution similarities there are between two images, the more likely we can find matched SIFT points between two images.

**References:**

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

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

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