EE 569 Homework 4 Write Up

Armin Bazarjani

March 28th, 2021

1 Texture Analysis

1.1 Motivation

Texture analysis is the subfield of image processing that is focused with the analysis and classification of a perceived texture within an image. This gives us insight into what the image looks like. As humans, we are capable of differentiating between different textures with relative ease. However, the process of coding this knowledge into a computer is quite obtuse and not incredibely well-defined.

The benefits from being able to quantify these somewhat intuitive differences between different textures like smooth, sharp, rough, silky can be found in various applications like medical image processing, geo-spatial analysis, and automated inspection.

1.2 Approach and Procedures

1.2.1 PCA

At a very high level, principal component analysis (PCA) tries to break down a large number of points, or features, into a smaller group called principle components. It is a technique that can be used both for data exploration as well as dimension reduction.

In an essence PCA constructs a new set of characteristics that do a good job of summarize our old ones, where the new characteristics are constructed from the old ones. From a geometric perspective, PCA tries to find the line of best fit between features that maximize variance and minimize reconstruction loss.

1.2.2 K-Means

K-Means clustering is an unsupervised machine learning algorithm that attempts to partition n data points into k clusters. It works by first randomly initializing the k centroids, and then it will iteratively update the centroids until on of two stopping criteria is met. Those being (1) the centroids have stabilized

and there is no change in their values, or (2) we have hit the maximum number of iterations.

The algorithm typically works by assigning each observation to one of the clusters 1-k with the nearest mean distance, the most common distance metrics are L_1 and L_2. After each iteration through the dataset, the centroids for each cluster are recalculated.

1.2.3 Support Vector Machines

Support vector machines (SVMs) are a supervised machine learning algorithm that can perform both classification and regression. For our purposes, we are only interested in using them for classification. Without getting bogged down too much by the details, and SVM attempts to find the maximum-margin hyperplane, between two distributions. Where it essentially learns the line that creates the greatest amount of separation between two classes equally.

There are two tricks commonly applied to SVM's. The first is the kernel trick where you can use a non-linear function to map the points into a higher dimensional space. Thus, a linear seperator in that space will be a non-linear one once we come back down to our feature space. Another commonly used trick is to combine multiple SVM's together for multi-class classification. Because the model is defined for binary classification we can make a classifier for each of our k classes and have it disntinguish between our current class, k, and everything else. We can then combine all of these classifiers together.

1.2.4 Texture Classification - Feature Extraction

One method of classifying textures is to use Law's filters to extract the features (we used 25 filters). These features are extracted for each pixel, so we are left with 25 feature maps. Once the we have the relevant features extracted into feature maps, we get the average energy of each feature map and put the value into a 25D vector. At this point, we also decided to apply PCA to further reduce the feature space from 25 dimensions to 3. After all this, we are left with a 3D vector for all of our different images both training and testing. In order to classify the test images, we calculate the Mahalanobis distance from the 3D test feature vector to all of the 3D test feature vectors in our training set, after taking the mean based on the label, we can choose to label the test sample with the one that got the smallest distance.

Implementing this was relatively straightforward. After we construct our 25 Laws filters from multiplying the different variations of the 5x5 Laws filters, I convolved each filter on each image to get the feature mapping response. From these responses, I took the average energy ($\frac{1}{N}\sum_{i=1}^{N} I_i^2$) of each feature mapping and put it into a 25D vector. After applying PCA I get a 3D vector using the first three principal components. Then, I simply calculated the Mahlanobis

distance from each test vector to the training set vectors. I found it easier to split the training set into 4 separate matrices after I applied PCA to make the operation somewhat quicker.

1.2.5 Texture Classification - Classifier Explore

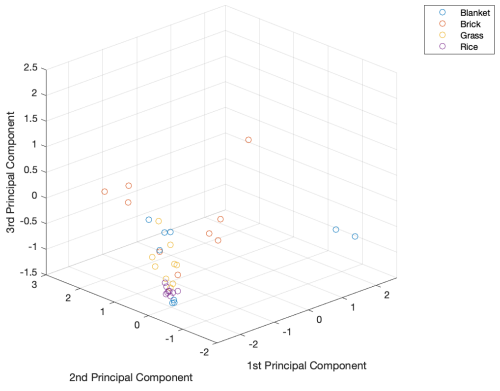
For this part of the assignment, we take the feature vectors that we got from part (a), and we apply some machine learning classification techniques to them. Those being (1) K-Means clustering, (2) Random Forest Classfication, and (3) Support Vector Machine Classification. Because we are using the same features, I simply saved all of the features I calculated above, through using the Laws filters and applying PCA into a few .mat files so I could access them quicker.

As far as the process goes, everything was pretty easy. Because we were allotted the use of packages, I took advantage of the machine learning library that MATLAB has and it made my life a whole lot easier.

For getting the accuracy of the K-Means classifier, I kept track of what cluster labels correspond with what "real" labels. I then took the maximum correspondence in order to say what cluster represents what label. This was a handy process because if you run K-Means multiple times, you will get different cluster assignments, so one time the cluster associated with 3 will correspond to "blanket", while on another iteration it will be 2.

1.3 Experimental Results

I will show the plot of the reduced 3D feature vector for the training images. Please find the accuracy of the method below in section "1.6.1 Discussion - Part (A)".



(a) PCA Visualization

Figure 1: Reduced feature space to the first three principal components

Method	K-Means (25-D)	K-Means (3-D)	Random Forest	Support Vector Machine
Accuracy	0.54	0.56	0.75	0.833

Please look below in section "1.6.2 Discussion - Part (B)" to see all of my final accuracy results for each machine learning method. However, I will also reproduce them here in a more readable format.

1.4 Discussion

1.4.1 Part (A)

I have reported the results of using feature extraction, dimension reduction with PCA, and nearest neighbor classification using the Mahalanobis distance in section 1.5 "Experimental Results". After visually inspecting the test images to see the true labels, I can see that this process correctly classified 7/12 images, to get an accuracy of 0.583, or similarly an error rate of 0.42.

Through my observation I have seen that this method has a tough time distinguishing between blanket/grass and grass/rice. This can also be seen in the plot in section 1.5 "Experimental Results" where I plot all of the training points in the reduced dimension along the first three principal components.

With regard to discriminant powers. I found that the best discriminant power was associated with the feature in dimension 10, or the $E_5^T R_5$ kernel. The feature associated with with the worst discriminant power was in dimension 11, or the $S_5^T L_5$ kernel. The reason I think this is the case is because the first kernel, when looked at in the energy domain shows peaks and valleys, which many of the patterns show too. The second kernel in the energy domain is simply a peak, which not many of the patterns display as the textures are very intricate.

1.4.2 Part (B)

Unsupervised Classification
For unsupervised classification, we only used the K-Means unsupervised classifier. Due to the nature of this algorithm, the results can be very sporadic and jumpy. A large part of this has to do with how we initialize the centroids. Through experimentation, I found that the default initialization method MATLAB uses tends to be the best, and the worst is random initialization.

I also noticed, that the distance metric that is used to calculate a points distance from a centroid or cluster is equally as important as the initialization. Through testing the available distance metrics that MATLAB allows, I found the 'city-block' distance metric consistently produced the best results, boasting 0.54 accuracy using the 25D feature vector and 0.56 using the 3D, PCA-reduced, feature vector.

Personally, I would opt for the feature dimension reduction technique as it is consistent. Especially because the benefits of averaging over multiple K-Means iterations only gains 2-3 more correct classifications. I personally think the more stable result of the feature dimension reduction technique is better.

Supervised Classification I tested the *Random Forest Classifier* with a few different numbers of trees (10, 20, 50, 100, 200, 500, 1000). I ran each classifier multiple times to get a rough idea of the average testing accuracy when using this supervised learning method. On average each classifier got an accuracy of around 0.75 (or an error score of 0.25). This is a substantial improvement over both the K-Means unsupervised learning method, as well as the feature extraction method with nearest neighbor in part (a).

When testing the *Support Vector machines* I applied the same methodology. I tested two different solvers (ISDA, SMO) and three different kernels (linear, rbf, polynomial). The non-linear kernels, those being the rbf and polynomial kernels consistently outperformed the linear kernel. The highest accuracy I got was using the polynomial kernel with the SMO solver, with the accuracy being 0.83 (or an error of 0.17).

2 Texture Segmentation

2.1 Motivation

Texture segmentation is the process of visually identifying different textures within an image, and then segmenting the different textures within the image as well. It is an interesting problem that can be approached many different ways, the current approach we used was to operate directly on the image. The benefits of texture segmentation again, stretch beyond our toy datasets. It is beneficial in geo-spatial analysis and medical imaging to name two.

2.2 Approach and Procedures

2.2.1 Basic Texture Segmentation

The previous sections covered all of the relevant background information that we needed up until this point. The big difference now is how we calculate the average energy and how we use it to construct a feature vector. The main idea behind the approach we use, is that we are now treating each pixel separately and we want to construct a feature vector for each pixel. Whereas in the feature classification section where the entire image was of a single feature, we got a feature vector for all of the different images.

The first step is the exact same as before, we apply the Laws filters on top of each pixel. The second step is where things start to get a little spicy. Instead of computing the average energy of an entire image, we now use a "window"

around the current pixel. We do this with all 25 laws filters and we end up with a 25D feature vector for each pixel in the image that we want to segment.

The next source of divergence comes from the fact that we now also are not concerned with non zero-mean $L_5^T L_5$ filter. Instead, we use it's value to normalize all of the other values inside of our feature vector, then we discard it. So, in the end we are left with a 24D feature vector.

The final step is to perform K-Means segmentation on all of our pixels with their respective feature vectors, setting our number of clusters, k, to be equal to the number of textures within the image.

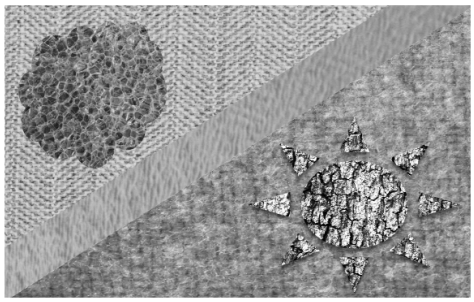
For application, I didn't really change much at all from problem one. The only thing I did differently was after I used the laws filters to get the 25 feature maps of each pixel's response, I mirror padded the image in accordance with the size of the window I was going to be using for the energy extraction.

2.2.2 Advanced Texture Segmentation

For advanced texture segmentation, one of the recommendations was to use PCA on our 24D feature vector and drop it down, similar to what we did in problem 1. Following this approach I tested three different down sampling sizes (3,5, and 7).

2.3 Experimental Results

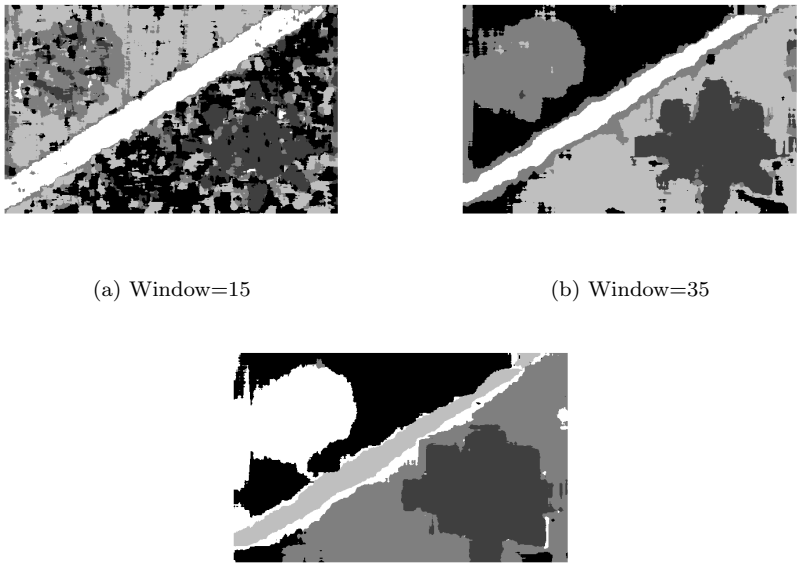
The original image is as shown:



(a) Original Image

Figure 2: Original composite image to be segmented

Now, I will show some of the different segmentation results using cityblock (L_1) distance.



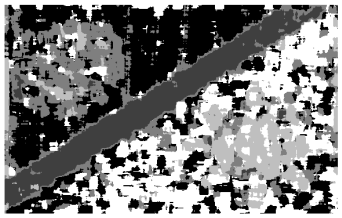
(a) Window=15

(b) Window=35

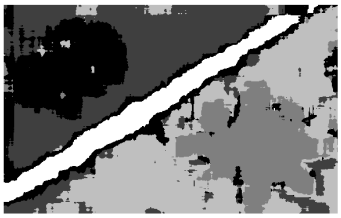
(c) Window=55

Figure 3: Segmentation results of L_1 distance with different window sizes

Now, I will show some of the different segmentation results using squared euclidean (L_2) distance.



(a) Window=15

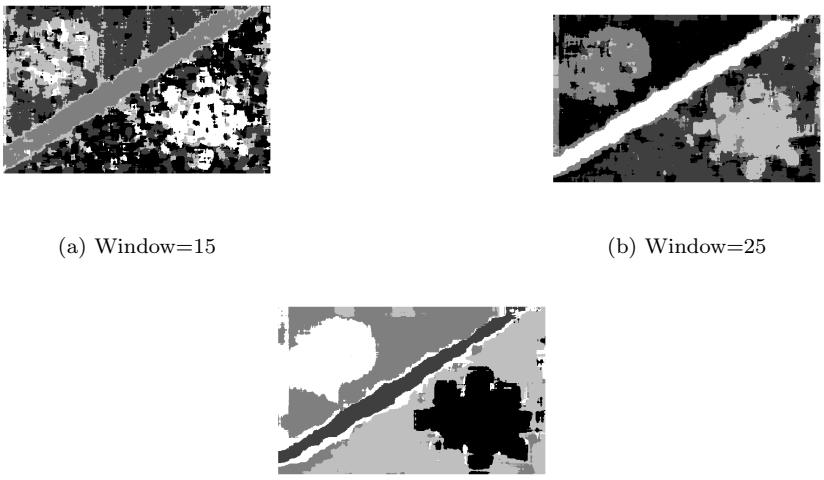


(b) Window=35

(c) Window=55

Figure 4: Segmentation results of L_2 distance with different window sizes

Now, I will show the different PCA results

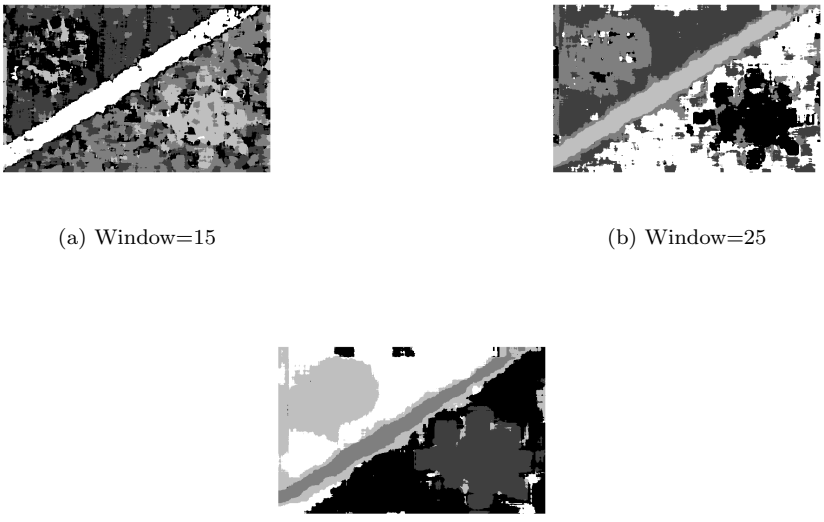


(a) Window=15

(b) Window=25

(c) Window=45

Figure 5: Segmentation results of PCA-3 with different window sizes

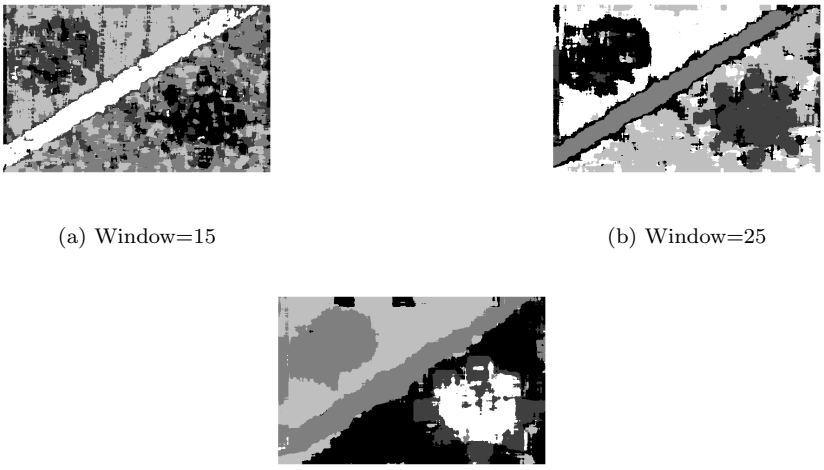


(a) Window=15

(b) Window=25

(c) Window=45

Figure 6: Segmentation results of PCA-5 with different window sizes



(a) Window=15

(b) Window=25

(c) Window=45

Figure 7: Segmentation results of PCA-7 with different window sizes

2.4 Discussion

2.4.1 Part (A)

As you can see above in section "2.3 Experimental Results" I tested out a few different things to see what impact they would make on the final results of this segmentation approach. The things I tested were changing the window size, and what distance metric K-Means clustering should use.

I personally could not see too many differences between the two distance metrics (L_1 and L_2). However, it is very clear the difference that the window size makes. As our window size for calculating the energy gets larger our final result is much less noisy as we merge more space together. The tradeoff is that we miss out on the finer details of the structures. Whereas with a smaller window size we can capture these details, but we also add a lot of noise to our segmentation results.

2.4.2 Part (B)

As you can see in section "2.3 Experimental Results" I have shown three different window sizes (15, 25, and 45) along with 3 different pca sub-sample dimensions (3, 7, and 9). None of them look much better than the others nor do they show a significant improvement, or any improvement really, over the original method.

3 SIFT and Image Matching

3.1 Motivation

As we have seen thus far, feature extraction is a very important aspect of computer vision, image processing, and just general image understanding in general. The ability to be able to extract salient points and features within an image has important implications in a diverse range of fields such as object recognition, navigation, video tracking, image stitching, image matching, scene understanding, among others.

Feature detection and extraction relies on algorithms that can use abstract image information at each point in the image to determine if there is a salient feature or not. Usually these algorithms are designed to be invariant to scale, shape, rotation, illumination, and to some extent camera viewpoint.

3.2 Approach and Procedures

3.2.1 SIFT

Scale Invariant Feature Transform (SIFT) is a feature detection algorithm that was developed by David Lowe. Initially it was published in 1999 in ICCV under the title "Object recognition from local scale-invariant features". The algorithm was then modified, the modified algorithm was published in 2004 in IJCV under the title "Distinctive image features from scale-invariant keypoints". The second publication gained more traction and proved to be a very useful algorithm as SIFT is now very widely recognized and used, the two papers garnering an impressive sum of 82,565 citations. I will go into much more detail about how the SIFT algorithm works and why it does such a good job in the discussion section below titled "3.4.1 Discussion - Salient Point Descriptor". However, I will also give a brief overview right now.

The main stages of SIFT consist of: (1) scale-invariant feature detection, (2) feature matching and indexing, (3) cluster identification by hough transform voting, (4) cluster identification by hough transform voting, (5) model verification by linear least squares, and finally (6) outlier detection.

3.2.2 Image Matching

I implemented the image matching problem through using the vl_sift toolbox for MATLAB that implements many important and useful computer vision algorithms. Thankfully, the package makes it really easy to see the score of each SIFT feature. To find the nearest neighbor feature, I found the lowest squared euclidean distance between the feature vector of the highest scoring feature in dog3 and one in dog1.

3.2.3 Bag of Visual Words

The bag of visual words is inspired by the bag of words representation that was popularized by natural language processing (NLP). It works by constructing a feature representation of an image and then combining combining said features into a codebook of sorts. The technique is very helpful with image classification as well as measuring image similarity, among other things. Essentially, it is a method to quantize the feature space. The quantization occurs from clustering the extracted features (a typical clustering method used is K-Means), we then use the clusters as psuedo labels for our features when we create our histogram. After clustering we create a histogram of frequency of features (cluster assignments) in that location. We can then use the histograms for classification.

I implemented the bag of words method, by applying the SIFT features for each image, applying PCA to reduce the feature space from 128 to 20. Then, I combined all of the PCA reduced feature spaces for the different images into a single matrix while keeping note of the proper dimensions for each one's start and end. Next, I run K-Means on the combined matrix with 8 different clusters. I then split the combined feature matrix using the saved starting and ending locations to get the cluster for each feature. After getting the cluster assignment for each feature for each image, I constructed the frequency histogram for each image by looping through the feature cluster assignment and adding up the number of points in each cluster.

3.3 Experimental Results

The largest scale SIFT feature in dog3 matched with the nearest neighbor SIFT feature in dog1 is shown below:

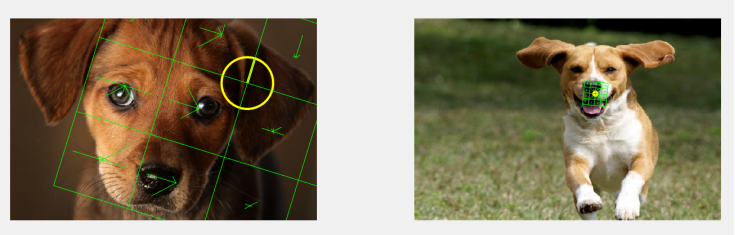
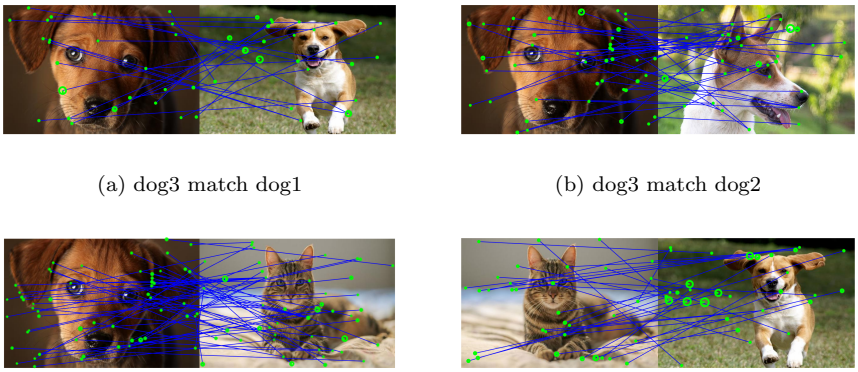


Figure 8: Largest scale keypoint in dog3 matched with nearest neighbor keypoint in dog1

The four different SIFT image pairings can be shown below:



(a) dog3 match dog1

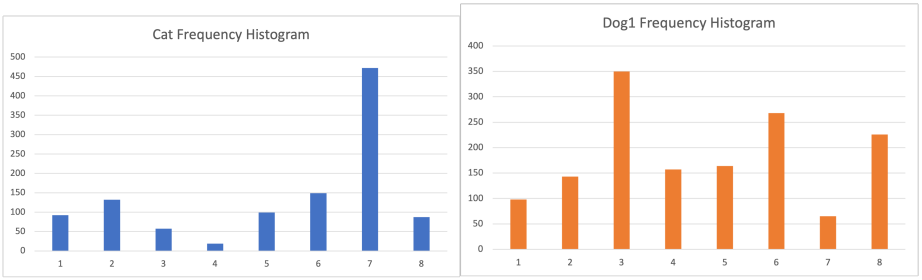
(b) dog3 match dog2

(c) dog3 match cat

(d) cat match dog1

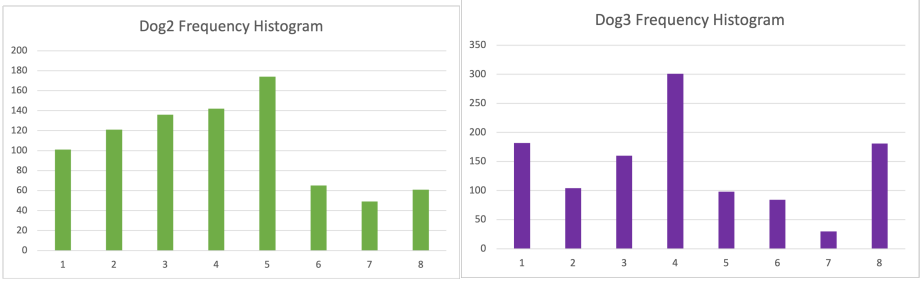
Figure 9: SIFT feature pairings for different images

The different frequency histograms for all the images are as following.



(a) Cat Histogram

(b) Dog1 Histogram



(c) Dog2 Histogram

(d) Dog3 Histogram

Figure 10: Frequency histograms

The table showing the similarity index between dog3 and all the other images are following:

Comparison	dog3,dog1	dog3,dog2	dog3,cat
Similarity Index	0.54	0.59	0.34

3.4 Discussion

3.4.1 Salient Point Descriptor

Question 1 - The geometric modifications that SIFT is robust to are image scale, rotating the image, as well as a substantial range of affine distortions.

Question 2 - The SIFT algorithm is robust to image scale and orientation through its scale-space extrema detection. Essentially, this method repeatedly convolves the initial image with Gaussians to produce a set of different scale space images. The algorithm then implements the difference-of-Gaussian function to identify potential interest points.

Question 3 - SIFT enhances its robustness to illumination change in its keypoint descriptor phase where the local image gradients are measured in the region around the keypoint and then transformed to become invariant to many things, illumination among them. The step taken to discount illumination is to normalize the feature vector to unit length, threshold the values in the unit vector so that each can be no larger than 0.2, and then re-normalize to unit length.

Question 4 - The main advantage of using difference-of-Guassian (DoG) over Laplacian-of-Gaussian (LoG) is that DoG is much more efficient to compute. On top of that the DoG function provides a very close approximation to the scale-normalized LoG function. This is beneficial as it will be used during the scale-space extrema detection step.

Question 5 - The output vector has 128 features.

3.4.2 Image Matching

Question 1 - The image showing the largest scale SIFT feature in dog3 matched with the nearest neighbor SIFT feature in dog1 is shown above in section "3.3 Experimental results". We can see that the location which had the highest score in dog3 was right on the crease of the right ear. Which suggests why the orientation is slightly off center and tilted to the right so as to follow the crease. The nearest neighbor feature in dog1 is on the right side of the nose. If we were to look at the orientation, we can see that is slightly tilted to the right. The right side of the nose is curved up and to the right, the same angle of the crease of the highest scoring dog SURF feature.

Question 2 - In my opinion the feature matching worked best between dog3 and dog1, second best was with dog3 and dog2, and matching the dogs with

the cats didn't work really well. Also, I feel like it's worthwhile to mention that my definition of matching "well" means the keypoint pairings actually seem like they match one another. It's obvious how the dogs and cats don't really match anywhere. However, the dogs match well on the snout, eyes, and ears. I believe dog3 and dog1 match the best because both of the dogs have downturned ears.

3.4.3 Bag of Visual Words

The results from the bag of visual words method can be seen above in section "3.3 Experimental Results". As expected the similarity index between dog3 and dog1 is the highest, and it is lowest between dog3 and cat. I believe the reasons for this are because if we look at the matched SURF feature pairings between the differnent photos, we can see that dog3 and dog1 share the most similarities whereas dog3 and cat share the least. Again, most likely because dog3 and dog1 have downturned ears and are both front facing, and comparing a dog and a cat should naturally not have many common features.