ECS 174: Computer Vision, Spring 2019 Problem Set 3

Gabriel Brusman, Wyatt Robertson

914060880, 913920853
University of California
Davis, CA 95616

1. Short Answer

1.1 *What exactly does the value recorded in a single dimension of a SIFT keypoint descriptor signify?*

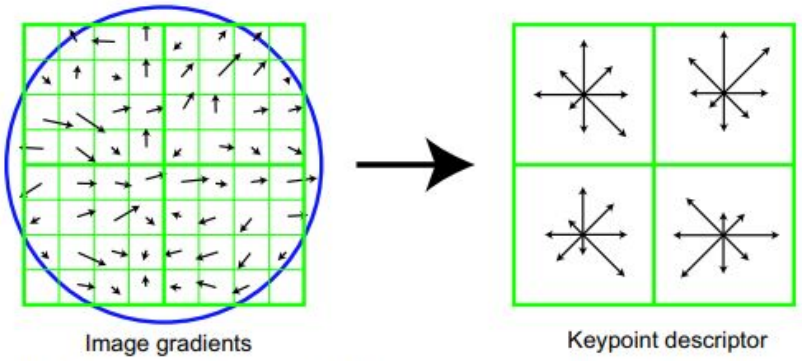


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

Source: https://people.eecs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf (the actual SIFT paper)

So we have a 4x4 array of histograms with 8 total orientation bins in each histogram which gives us a 4x4x8 = 128 element vector. Each individual value in one of these vectors signifies the gradient magnitude of one of the 8 directions in one of the 16 histograms.

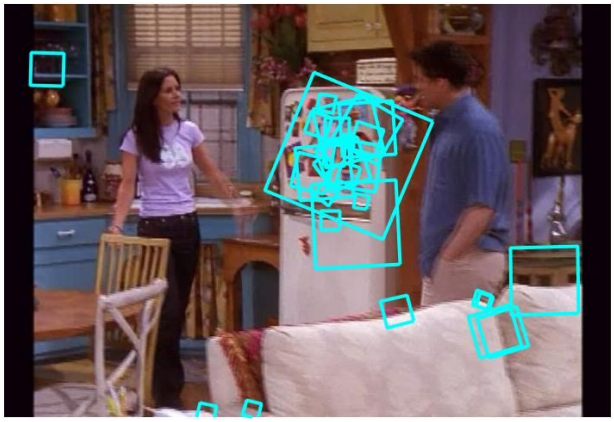
1.2 *A deep neural network has multiple layers with non-linear activation functions (e.g., ReLU) in between each layer, which allows it to learn a complex non-linear function. Suppose instead we had a deep neural network without any non-linear activation functions. Concisely describe what effect this would have on the network.*

Suppose we had a deep neural network without any non-linear activation functions. Then the network can only calculate linear combinations of values or linear combinations of linear functions. But, a linear combination of lines is also a line. This being the case, we would not need multiple layers in our network to train this model, since we only need one layer to learn a single line. This network would not still be considered a deep network since we would not need many layers to learn the resulting function.

2. Programming Problems

2.1 Raw Descriptor Matching

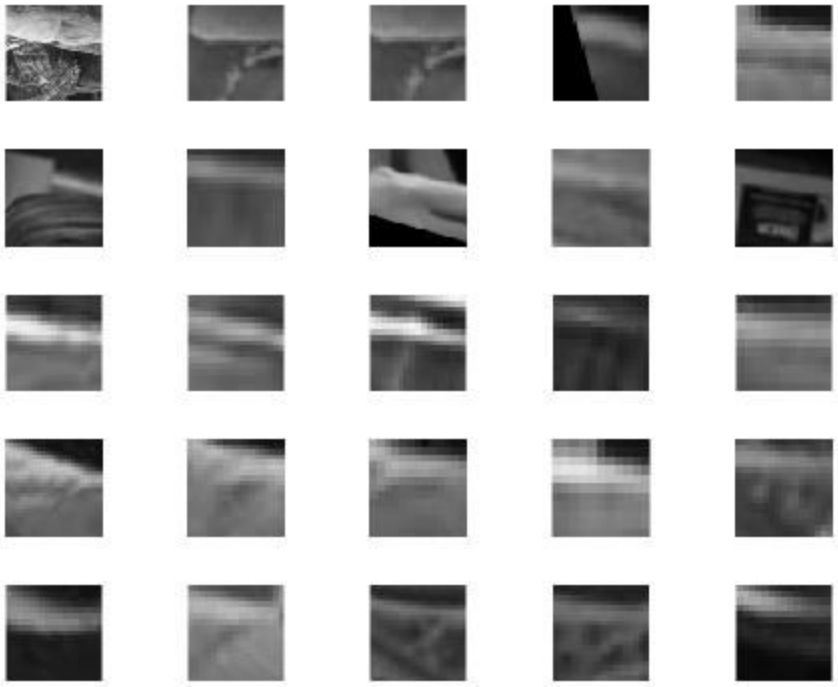
In this section we allow the user to select a region of interest in the left image and then match descriptors in that region to descriptors in the second image based on Euclidean distance in SIFT space.



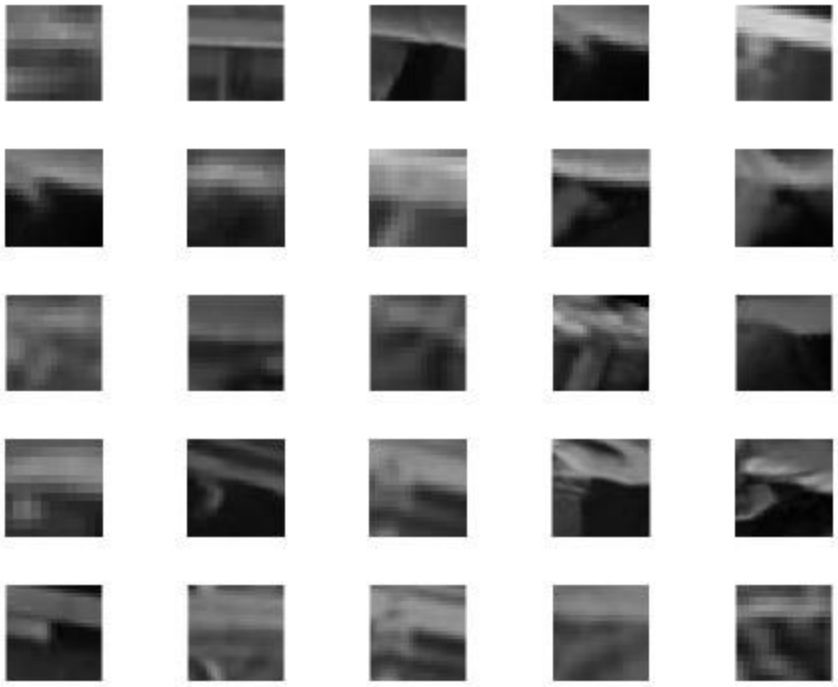
2.2 Visualizing Vocabulary

Using N (number of files) = 300, and K (number of clusters) = 1500, and using all descriptors from each file, we achieved the following results by sampling 2 random words from our visual vocabulary:

Word 1:



Word 2:



In the first word, we see that the patches are tend to have a lighter colored horizontal strip just below the top of the image. In addition, the bottom part of the image (below the lighter colored strip) is lighter than the top part of the image (above the lighter colored strip).
In the second word we can see that the patches are characterized by a white strip at the top of the patch and a lighter left region, and then a darker bottom-right region.

2.3 Full Frame Queries

In this section, we use 3 frames to serve as queries as we find their M=5 most similar frames (in rank order) based on the normalized scalar product between their bag of words histograms.

Frame 1:

As can be seen by the above figure, the algorithm did a good job finding similar frames to the query image. Since there were multiple frames of this scene, the algorithm was able to find 5 nearly identical frames to the query image, with most of the differences between them being in facial expressions and body movements, such as the arm of the woman on the right.

Frame 2:

Again, the algorithm did a good job finding similar frames since there were enough frames of this scene available in the dataset. The main differences between the frames are in the man's facial expressions and his and the other character's arm positions. Overall though, these frames are very similar.

Frame 3:

Here the algorithm still did a fairly good job picking out similar frames, but the last image (the least close to the query) is a bit different from the query image. The reason for this is because there were only 4 frames of the first image, with all of the characters on the couch, and so the algorithm had to find a frame from a different shot to use as the 5th image. That being said, the algorithm still did a good job finding a close match. Notice that the 5th image still contains a character sitting on a couch. Upon closer inspection it looks like the couch in the 5th image is the same couch as the one in the other images.

2.4 Region Queries

In this section we use 4 frames to get retrieved frames when only a portion of the SIFT descriptors are used to form a bag of words.

Image 1:

Results for Image 1:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

Here we selected the table and the chairs in our region. One of the frames, the 2nd closest image, was actually from the same shot as the query image, so the algorithm did a good job there. Interestingly, what the algorithm picked up in the other features was not the same table and chairs as in the query, but rather the objects that were placed on the table. Notice that in all of the images there are papers and other objects on top of the table. This result might be because the table in the query image was never sampled in our kMeans computation. We only used 300 frames to sample descriptors from, it's entirely possible that the specific table was not in any of our sampled frames, and therefore its descriptors never made it into our data.

Image 2:

Results for Image 2:

Here we sampled the entirety of the male character on the left. As expected, this region included enough descriptors for the algorithm to find very close matches, with every image it found being from the same shot as the query region. It helped that there were a lot of frames in the dataset from this specific shot in the video, so it was likely for our kMeans computation to pick a frame from this scene.

Image 3:

Results for Image 3:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

Here we selected only the mug being held by one of the characters in the background, and as expected the algorithm failed to find close matches. This could be for multiple reasons. The primary reason is probably that the region enclosing the mug is very small and likely did not have many descriptors attached to it. This meant that the algorithm did not have a lot of data to try to match with other images. Also, it's possible that there were no frames with mugs that were sampled in our kMeans computation. This would mean that there were no visual words in our visual vocabulary that could be used to describe the mug, and so the algorithm was clueless when trying to search for other frames with the mug. The best case scenario we could have hoped for with such a small region would be to match other frames with mugs in them, but the algorithm was not able to do that. Interestingly, the closest matched image contains a character who looks very similar to the one holding the mug in the query image (perhaps it's the same character), and he is also pictured in a similar posture, where he is sitting and slouched over a bit. We are not sure if that is the algorithm doing a good job, or just a coincidence.

Image 4:

Image 4 Results:

Here we selected the cabinet in the top right of the query image. The algorithm was able to find 5 frames from the exact same scene, so it did a good job. This region was large enough that it probably contained a lot of descriptors to help the algorithm find matches, and also there were a lot of frames containing this scene, so it is likely that our vocabulary contained data from this scene, and therefore from that cabinet.

2.5 Full frame queries

In this section we use two query frames to find the *M*=10 most similar frames to each frame based on the normalized scalar product between their bag of words histograms and the normalized scalar product between their AlexNet fully-connected layer 7 activation features respectively.

friends_0000004503.jpeg Results:

SIFT Bag of Words:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

Closest Image # 6

Closest Image # 7

Closest Image # 8

Closest Image # 9

Closest Image # 10

AlexNet DeepFC7:

friends_0000000394.jpeg

SIFT Bag of Words:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

Closest Image # 6

Closest Image # 7

Closest Image # 8

Closest Image # 9

Closest Image # 10

AlexNet DeepFC7:

Comparing SIFT bag-of-words with Deep Features

It is clear that the AlexNet results are much better than the SIFT Bag of Words results. In the first frame, the Bag of Words results got about 8 frames from the same scene as the query image, while AlexNet got all 10. In addition, when comparing the frames that AlexNet and the BoW chose, the frames and the ordering of the frames that AlexNet chose are even better than the good frames that the BoW chose. However, at least the first three chosen frames from BoW and AlexNet are the same, so the BoW didn't do a terrible job.

In the second frame it is also obvious that AlexNet significantly outperformed our SIFT Bag of Words method. The BoW matched about 5 of the matching images to the correct scene, where AlexNet again matched all 10.

One reason that AlexNet might have done better here is because it trained on far more data than the BoW was able to use. The BoW method only trained on 300 of the files from the given dataset. The AlexNet implementation very likely saw a much higher volume of data than the SIFT BoW method did. In addition, due to the fully connected and non-linear nature of AlexNet, it was likely able to learn better, more significant visual words/features to use for its search.

3. Extra Credit

A stop list was created to ignore commonly used words, and then tf-idf weighting was applied to the bags of words in order to reduce unnecessary information. An experiment was undergone on a previously used image to highlight the effects of the implemented list and tf-idf weighting.

Query Region:

Without stop list or tf-idf weighting:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

With stop list or tf-idf weighting:

Query Image

Closest Image # 1

Closest Image # 2

Closest Image # 3

Closest Image # 4

Closest Image # 5

As illustrated by the above figures, stop list and tf-idf weighting successfully determined the most similar frame in the data set and eliminated much of the variability caused by over-represented words. For example, closest image #2 for the original experiment was a still image of a lone female character, which has no obvious relationship to the query region which is of a coffee cup. As you can see, all six pulled frames contain drinking containers, whereas in the previous experiment only the original frame contained a drinking container.

4. Sources Cited

1. Lowe, David. “Distinctive Image Features from Scale-Invariant Keypoints”. 2004. Computer Science Department University of British Columbia, Vancouver, B.C., Canada.