Vincent Chau
998947424
ECS189G
PS3

Problem Set 3

I. Short Answers

1.

The results using the Laplacian of Gaussian by taking any positions that are local maxima, the detect points are repeatable and distinctive. However, there are some tradeoffs, the detection of interest points is more repeatable when there is robust detection, and precise localization, if there were more points, robust to occlusion works with less texture. Furthermore, the more distinctive, it is less flexible meaning there will be less variants to maximize correct matches.

When using the Harris corner detector(thresholding) a good balance of distinctiveness and repeatability can be reached by increasing distinctiveness through using color information to describe regions. The color model helps the Harris corner detector to become less invariant to changes in lighting and this tradeoff would detect more distinctive regions.

2.

Each point in the SIFT detector space is a 128-dimensional. Each point signifies a local descriptor(SIFT vector). It consist of points that are extracted from a number of images to build clusters to map visual words.

3.

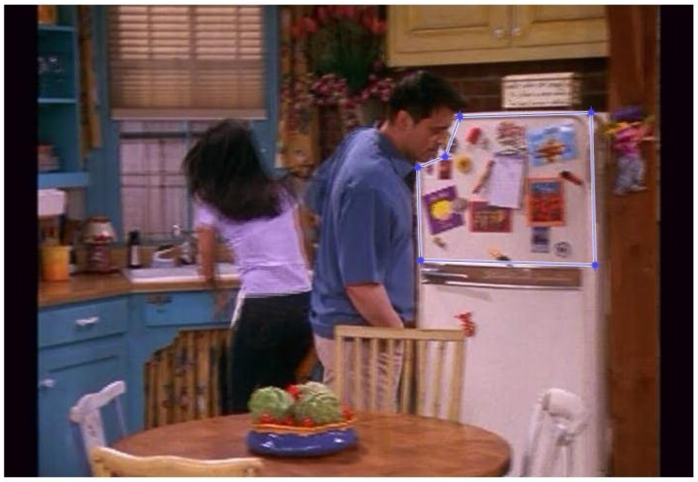
Hough transform can be used to cluster and vote on features that are consistent for all object poses that are similar to the feature. As for the Hough Space size, I feel like it depends on the implementation of the algorithm. But, I believe the hough space would be 4-Dimensional. Each dimension would be the position, scale, orientation, and votes for that specific feature.

Vincent Chau
998947424
ECS189G
PS3

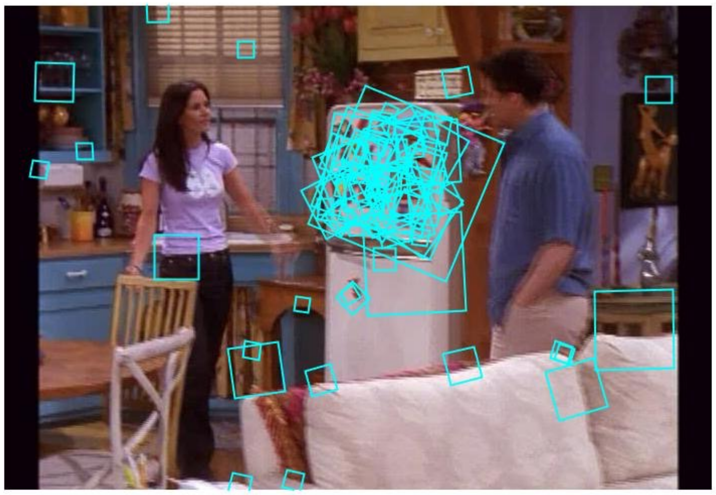
II. Programming Write-Up

1. **Raw Descriptor Matching**

Selected Region:



displaySIFTPatches:

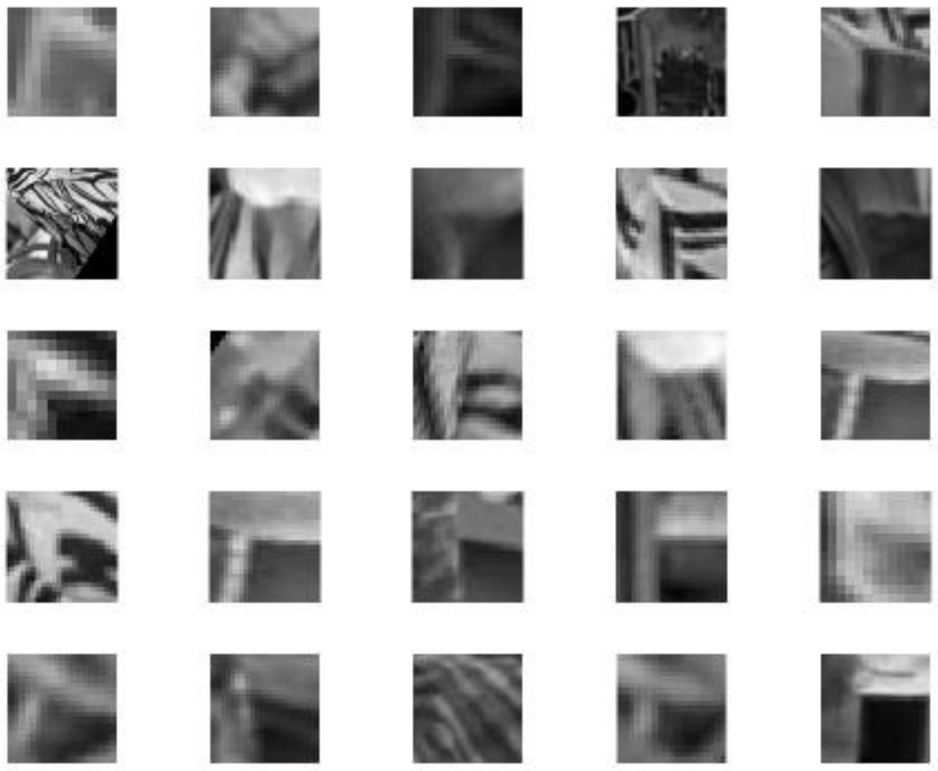


Explanation:

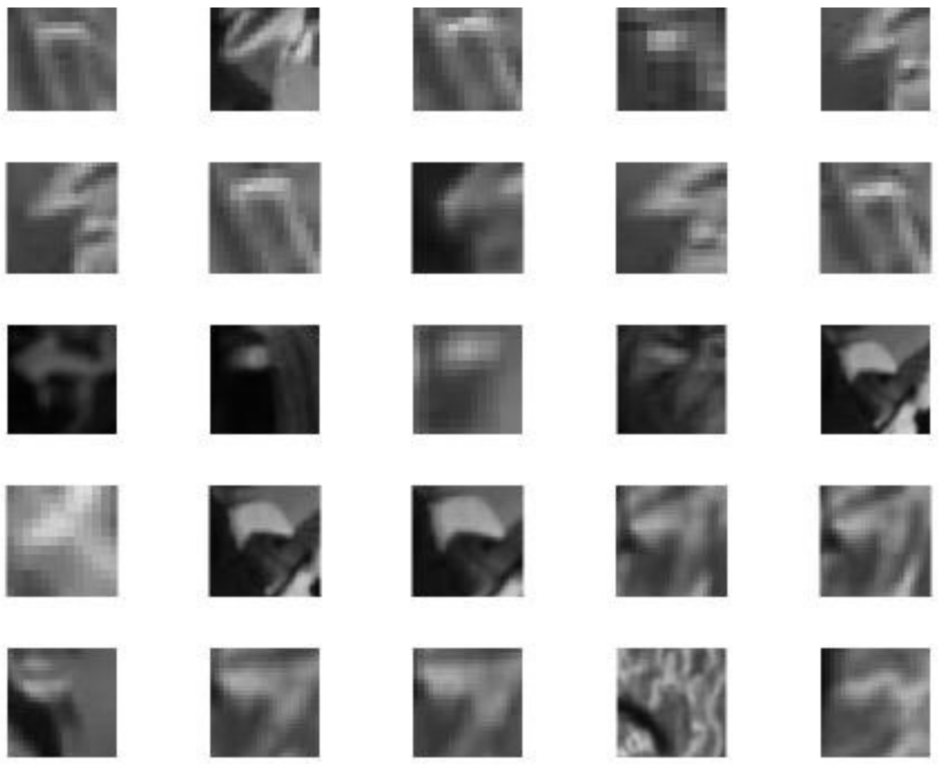
Implementation wise, I calculated the distance between the feature points to the closest feature points on the second image with the provided dist2(). For each row, I took the minimum of the distance and appended the features row-wise to the matrix matchFeatures. The column contains the indices that corresponds to the closest matching feature points on the second image. I then tried to achieve a more accurate result by taking the threshold of the mean. I took the means of the match features and thresholded the results by taking values that were less than 75% of the mean value, then displayed the patches with the provided function.

Vincent Chau
998947424
ECS189G
PS3
2. **Visualizing the vocabulary**

I. 25 Patches From Cluster



II. 25 Patches From Cluster



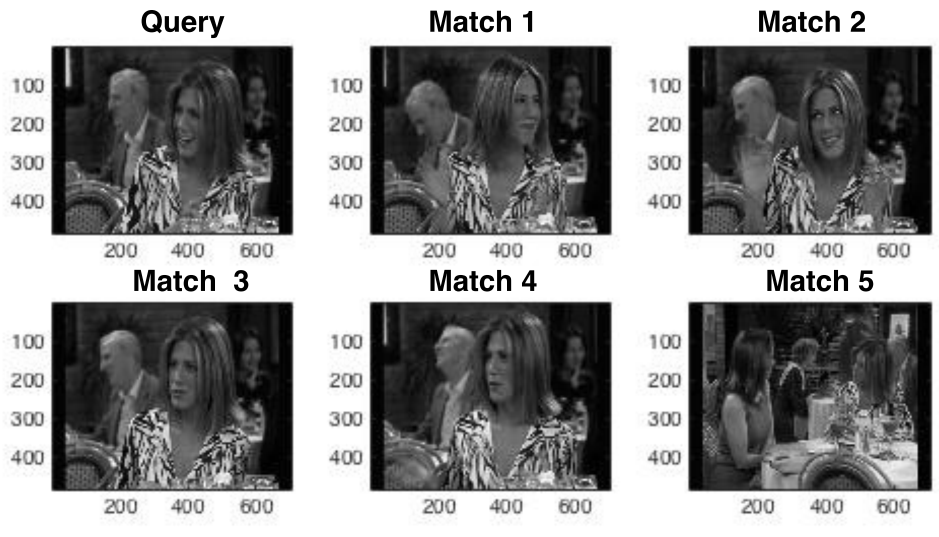
Vincent Chau
998947424
ECS189G
PS3

Explanation:

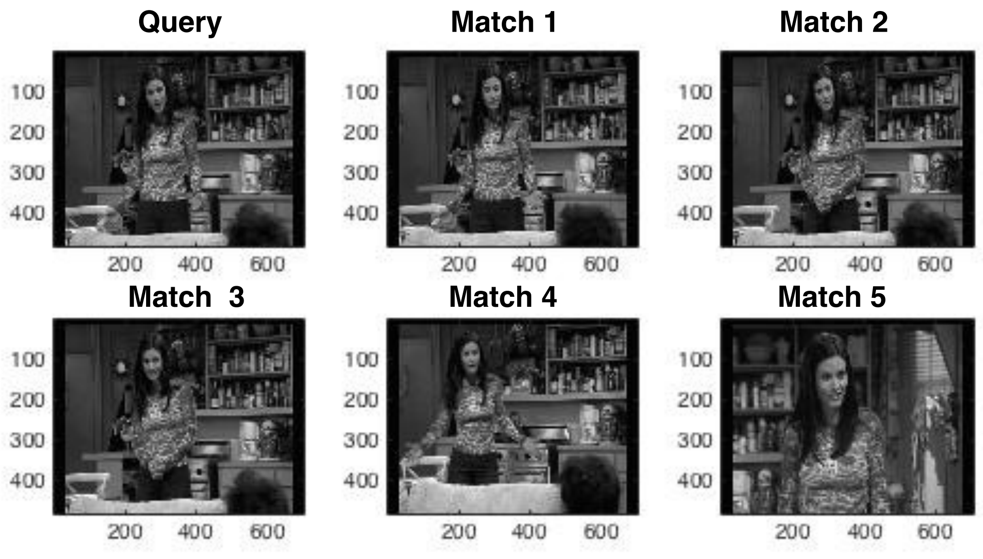
The bags of words above are correct as the patches look very similar to one another and belong to the same cluster. The feature in the image is as expected, since the patches are similar and differ by its scale, orientations. As for my implementation, I first extracted and stored the descriptors from 100 files. I did not run all of the files because the runtime was over 8 hours. I tried to fix this problem but I was unsuccessful. This may be due to my implementation on indexing the images and their features. I used three matrices to track images and for the look up of frames while loading in the images. The three matrices mapped the image ID's with its descriptors size, a matrix of gray scale images and it's ID, and a map of an image's path name and the amount of features. It was necessary for my implementation to store the images which is later used in part 3 and 4 to build the histogram. Next, I built the vocabulary by clustering with kmeansML, with k = 1500 and passing in the cumulative amount of descriptors. The descriptors are now clustered into k clusters. Each cluster is a word and next I sampled random words. After acquiring the selected descriptors, I extracted the patches from the frame and displayed the 25 patches each for the two words. As for the kMeans.mat file, i used it to save all the variables in the workspace instead because I wanted to improve runtime and have a collection of variables that would not be needed to recomputed later on.

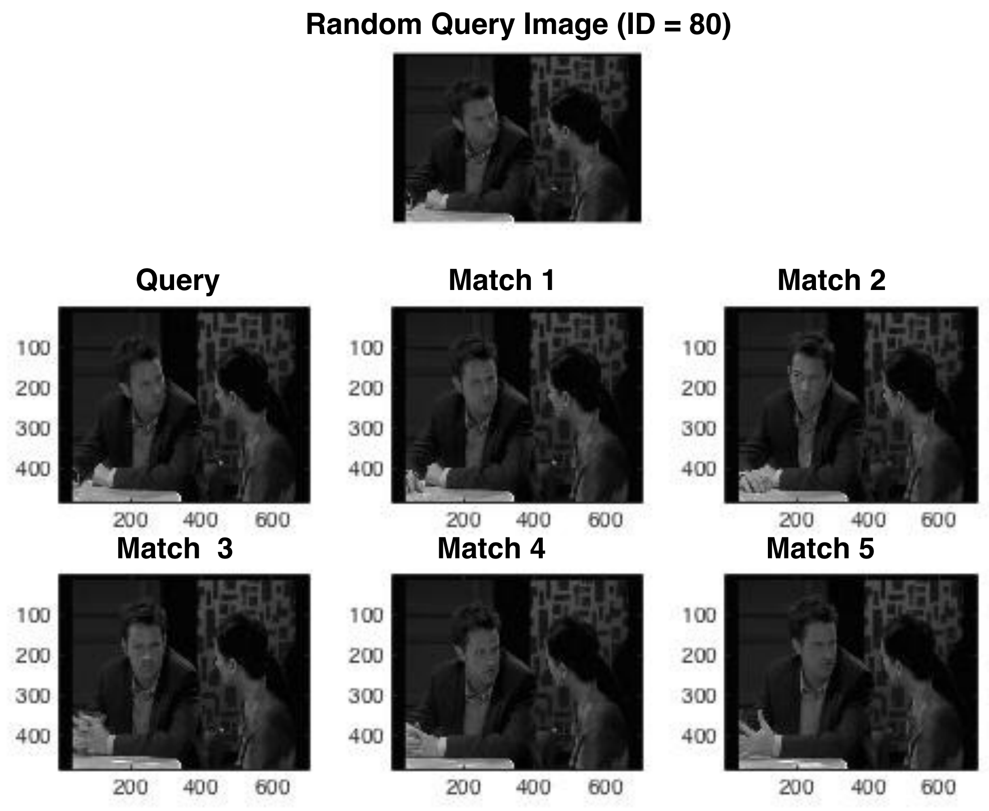
Vincent Chau
998947424
ECS189G
PS3
3. **Full frame queries**

Random Query Image (ID = 10)



Random Query Image (ID = 30)





Explanation:

The results of full frame queries are very accurate. Although one thing to take note is that I was unable to read all frames in visualize vocabulary because it took over 8 hours. All these results were produced with the frames I had read in (100 frames). I have not yet found a solution to read in all the frames and implement it through all files. Throughout this assignment, I used only the first 100 frames and I will need to figure out a more efficient way to store the frames. Another issue I had was the black frames. I stored the black frames and I had to skew my results because the black frames would show up on my subplot without the skew. As for the format of the subplot, the column in the second row is a copy of the original image, and the following are the 5 images with the least amount of difference in distance. The random frames were picked with queryID being 10, 30, 80. The results are correct for this dataset with only the first 100 frames.

Vincent Chau
998947424
ECS189G
PS3

The query images are matched correctly with frames with the most similar feature in the set of the first 100 frames. I was unable to search through all frames because my implementation was based on the image matrices i used in part 1, which stored the actual image by its index and I was unable to store all frames in a timely manner.

As for the calculations, I created a histogram with the return memberships from kMeans for each image, with its score. Next I iterated through the histogram and calculated with the formula in the lecture slide 14(indexing_instances.pdf):

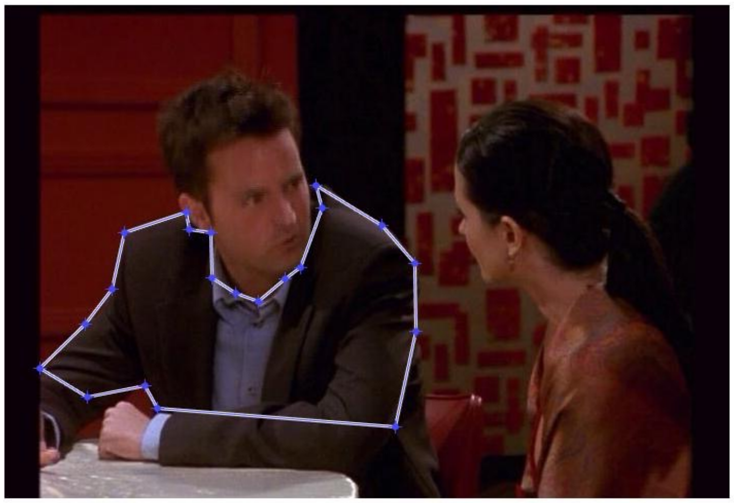
$$\langle d_j , q\rangle / \|d_j\| \|q\|$$

I kept the histogram from the query image as a constant and iterated through all the other images applying the formula, and finally sorted the rows of the matrix's distance and index. I then plotted the image images but had to skew the first 4 images because I had not filtered the black images in part a. I had to adjust the subplot to skip over the black images. The results were spectacular as each query yielded very relevant and similar frames.

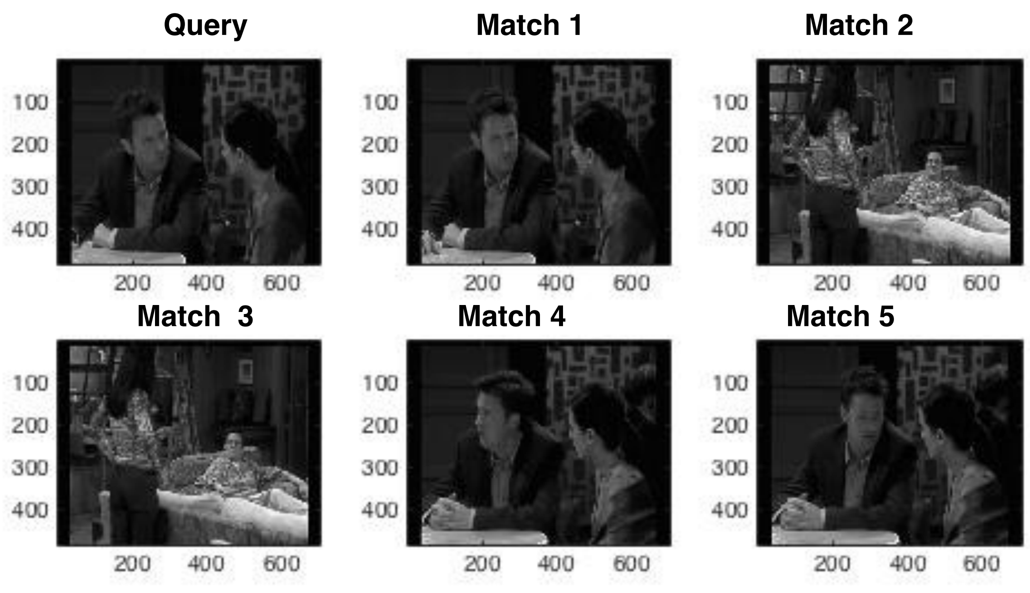
Vincent Chau
998947424
ECS189G
PS3
4. **Region queries**

(Good Cases)

Selected Region(queryID = 80)

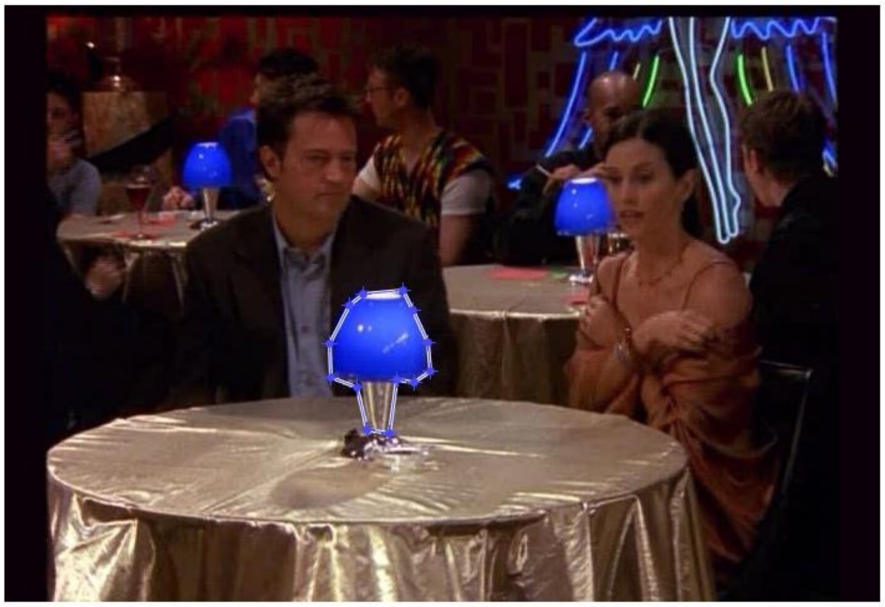


Random Query Image (ID = 80)

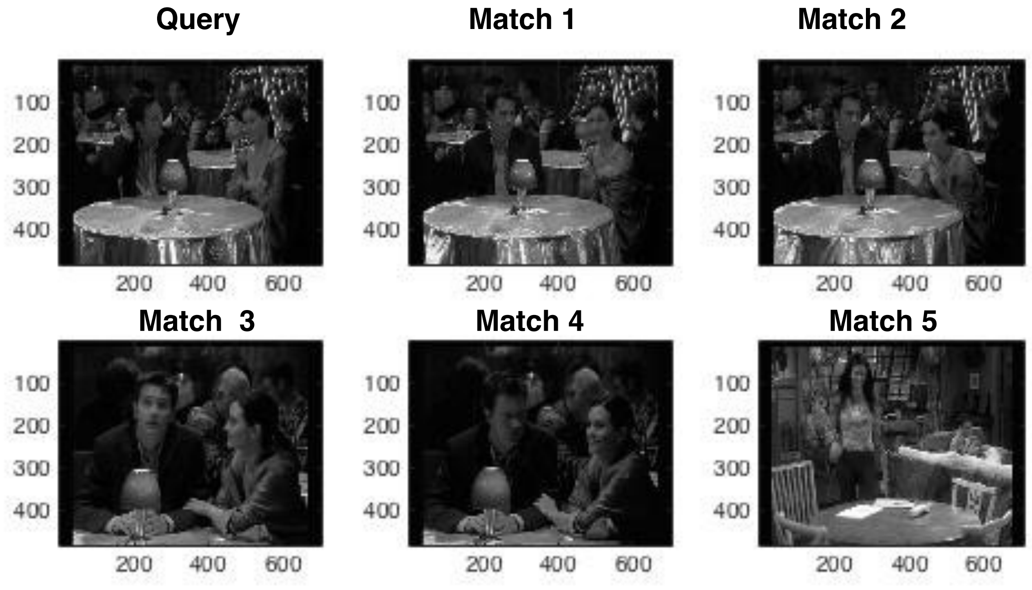


Vincent Chau
998947424
ECS189G
PS3

Selected Region(queryID = 69):

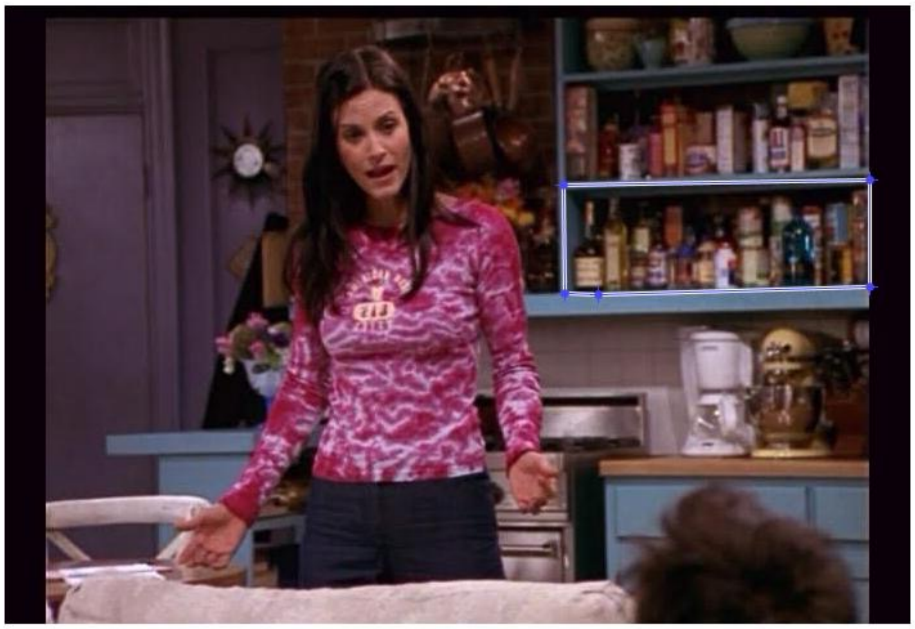


Random Query Image (ID = 69)

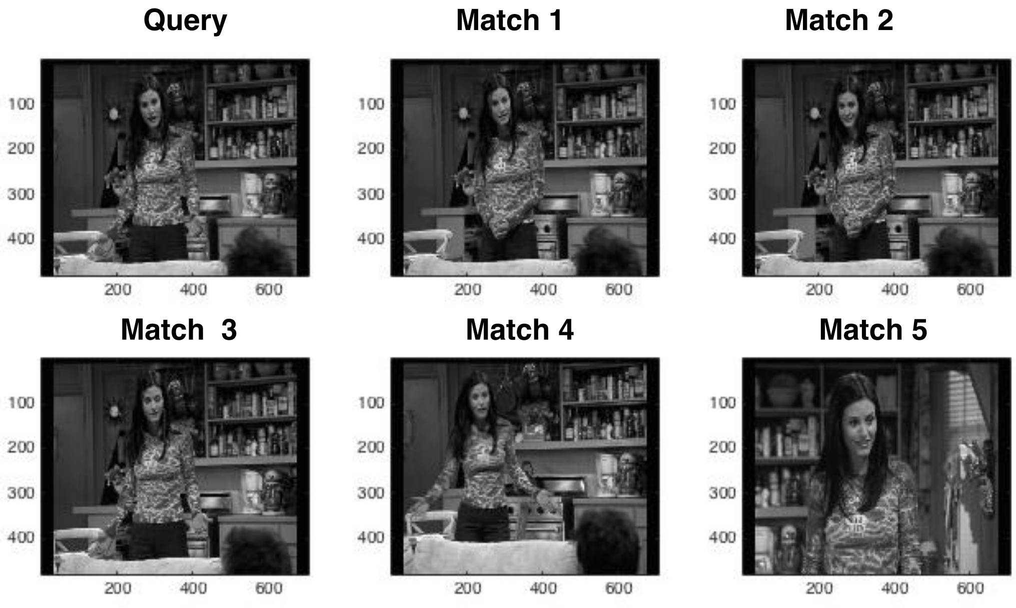


Vincent Chau
998947424
ECS189G
PS3

Selected Region(queryID = 30):

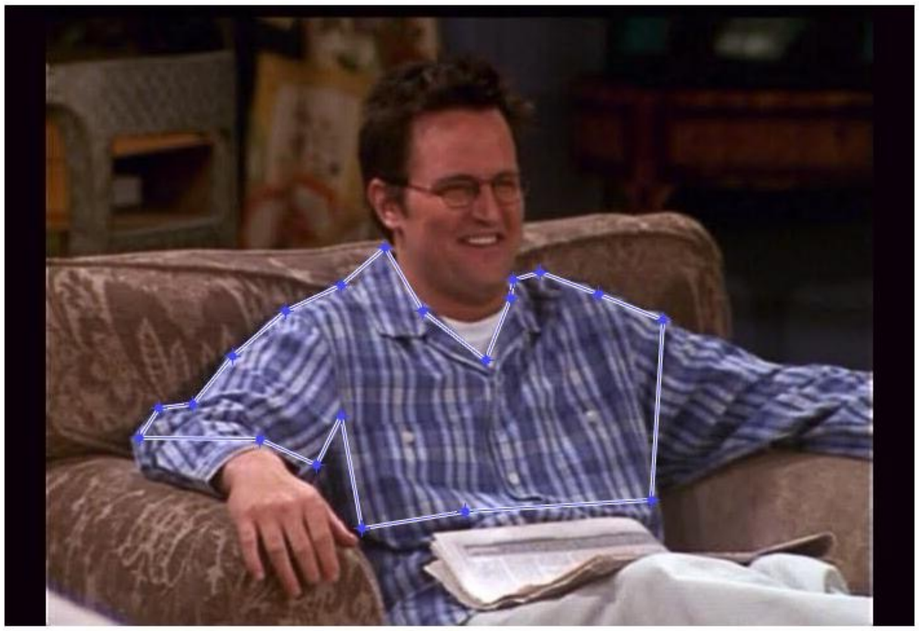


Random Query Image (ID = 30)

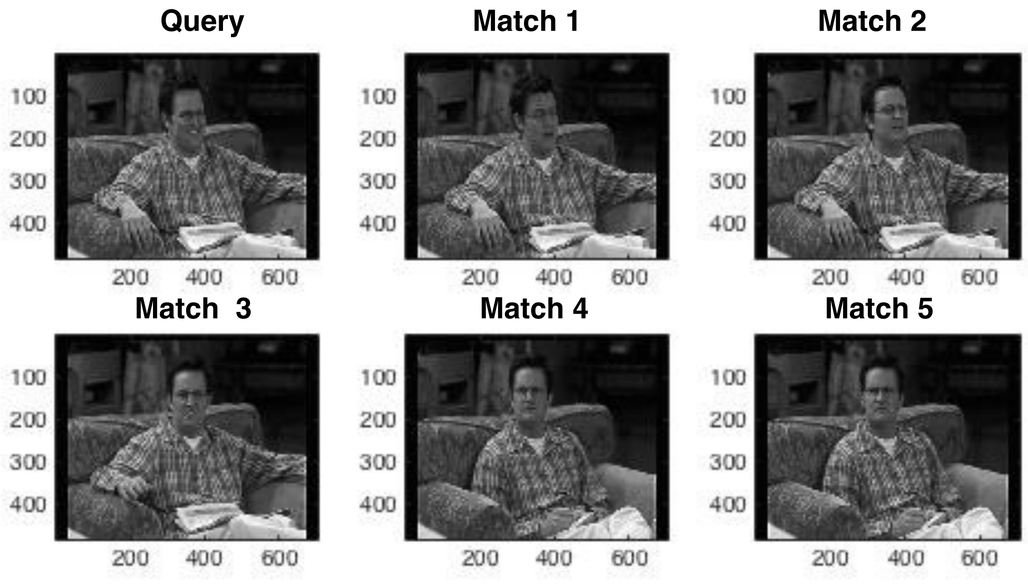


Vincent Chau
998947424
ECS189G
PS3

Selected Region(queryID = 28):



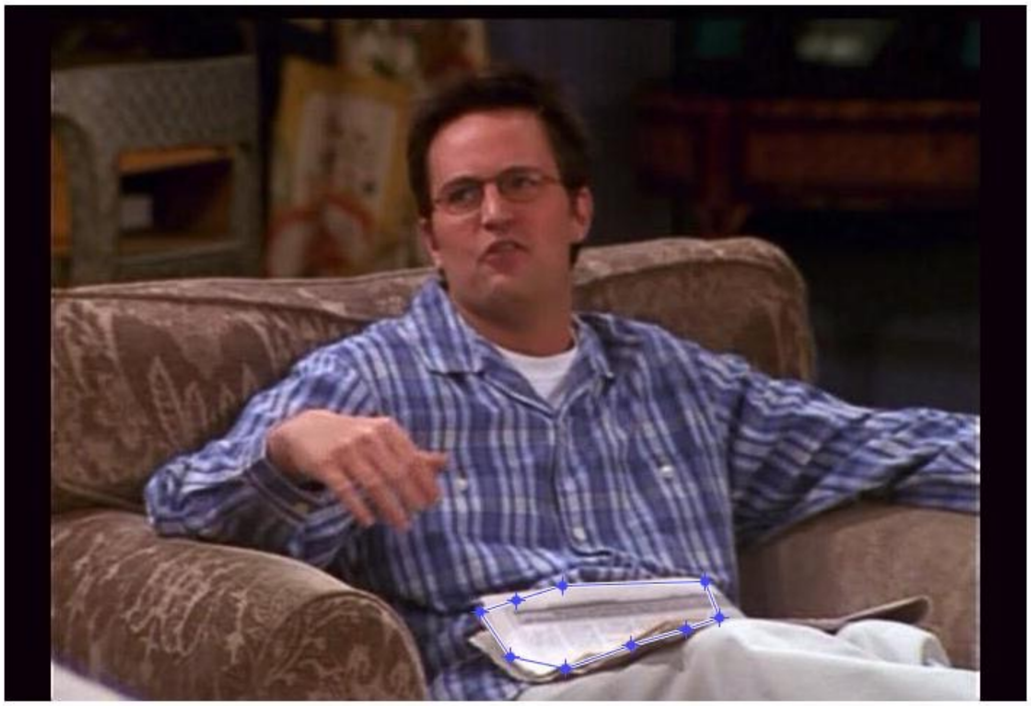
Random Query Image (ID = 28)



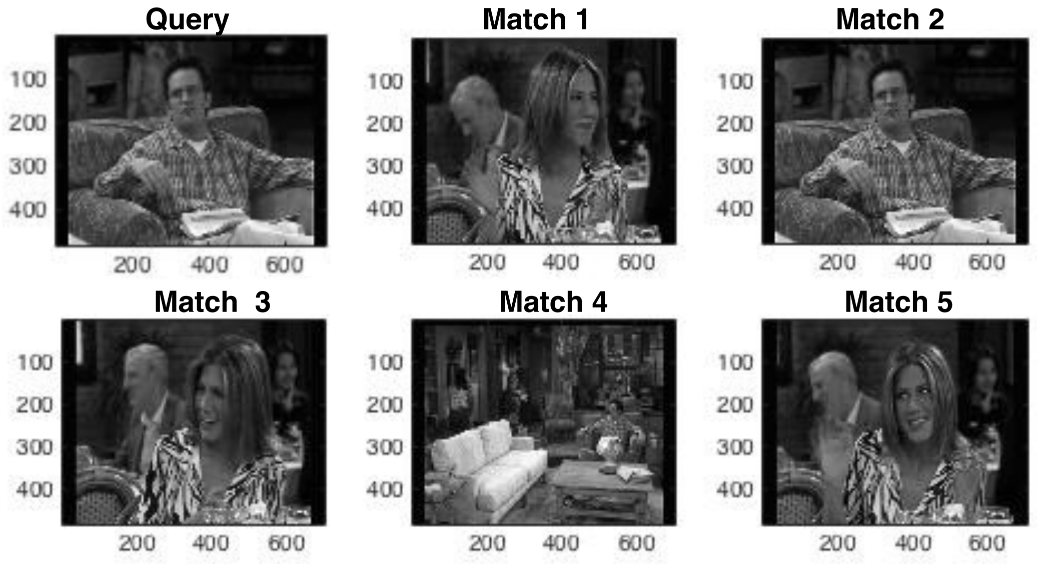
Vincent Chau
998947424
ECS189G
PS3

(BAD CASE)

Selected Region(queryID = 16):



Random Query Image (ID = 16)



Vincent Chau
998947424
ECS189G
PS3

Explanation:

The results were accurate, with query regions like the figure above (queryID 28). The region was specified selected with a nice cutout of his shirt and it pulled the most similar frames with very good accuracy. As for queryID 30, the selected region was the bookshelf and the calculated similarities yielded the 5 similar frames with the last match as a frame that is a closer-up image of the scene. Another good case is queryID 69 where the region is the blue lamp. The first four matches were accurate but the 5th image is not. This is due to the amount of frames with the blue lamp in my data set of 100, the same applies for queryID 80.

For a bad case, with the queryID 16 and selected region as the newspaper, the results were not accurate. The frames it matched with included Rachel's striped color shirt. The bag of words of the region would closely resemble the bag of words for Rachel's white colored, striped shirt. This explains the bad match up because the stripes closely resembles the lettering on the newspaper, because they are very similar.

For my implementation of this part of the problem set, I used the same method in part 3. Picked a query image based on the queryID and allowed the user to select a region. I then took the descriptors in the region and constructed a histogram with the scores. Again applying the dot product / norm formula, but this time used the region histogram as a constant. I iterated through the images and computed the similarity, then sorted and displayed the matches. I skewed the plot again like in part3 to filter out the black images.

Vincent Chau
998947424
ECS189G
PS3

Overall, my implementation seems to be working great for a data set of 100 frames. I still need to debug and find out a more efficient way to store the descriptors, images, and sift features. It takes more than 8 hours to read in all the frames, but using this implementation on a data set of 100 images shows amazing results.