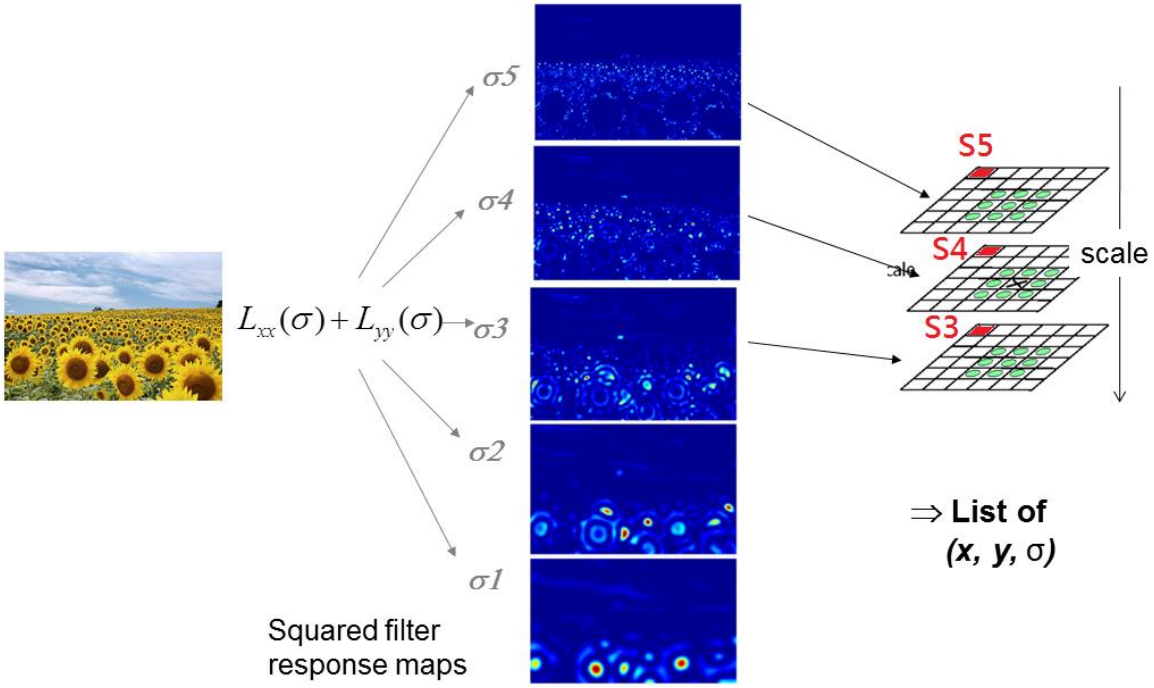
ECS189G Computer Vision Problem Set 3

WEI-CHIH CHEN

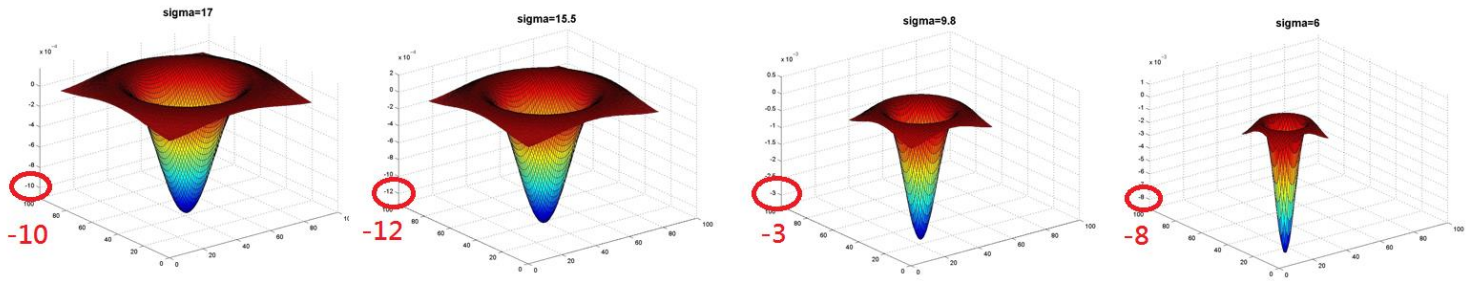
ID: 912448776

I. Short programming example

1. (a) Take any positions that are local maxima in scale-space: That means in any position (x,y) which is local maxima in scale-space we count it interest point. In below graph, S3, S4 and S5 are in the same (x,y). One of them gets local maxima so one of them is selected as interest point. However, this position gets relative low f value compare to truly interest point, although we still can select truly interest points. So **this method is repeatability** but **NOT Distinctiveness.**

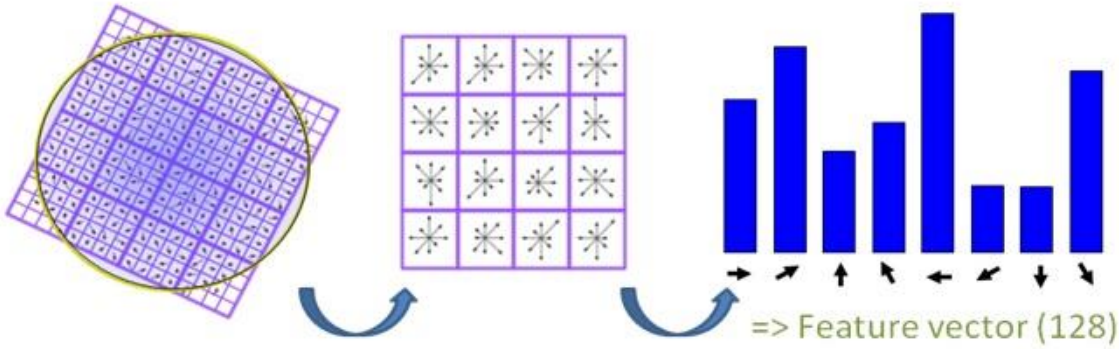


(b) Take any position whose filter response exceeds a threshold: In below graph, for different sigma size, the local maxima are different. No matter what value we chose can't represent truly interest point. Therefore, this method is **NEITHER Repeatability NOR Distinctiveness**.



Interest points can be selected correctly by local maxima in both position and scale space.

2. The magnitude of specific gradient direction. For every SIFT descriptor it records 8 gradient direction of 16 near blocks. So it has 8x16=128 values. We can show it in graph below. This graph is from: https://gilscvblog.wordpress.com/2013/08/18/a-short-introduction-to-descriptors/



3. It uses X position, Y position, scale and rotate four dimension. To recognize an object by spatial verification, Generalized Hough Transform tries to get enough votes for some local features to decide whether an object is found. Since in a new image we can't make sure the whether target object rotate and whether target object is scale in different size, we need to add these two parameters into Hough Transform dimension. And include original 2D local dimension, there are four dimensions in Hough space.

II. Programming: Video search with bag of visual words

1. Raw descriptor matching

   My method:

   a) Initialize data: `load('twoFrameData.mat')`

   b) Let user set region in first image. I used `selectRegion` to do this.

   c) Computer nearest raw SIFT descriptors:

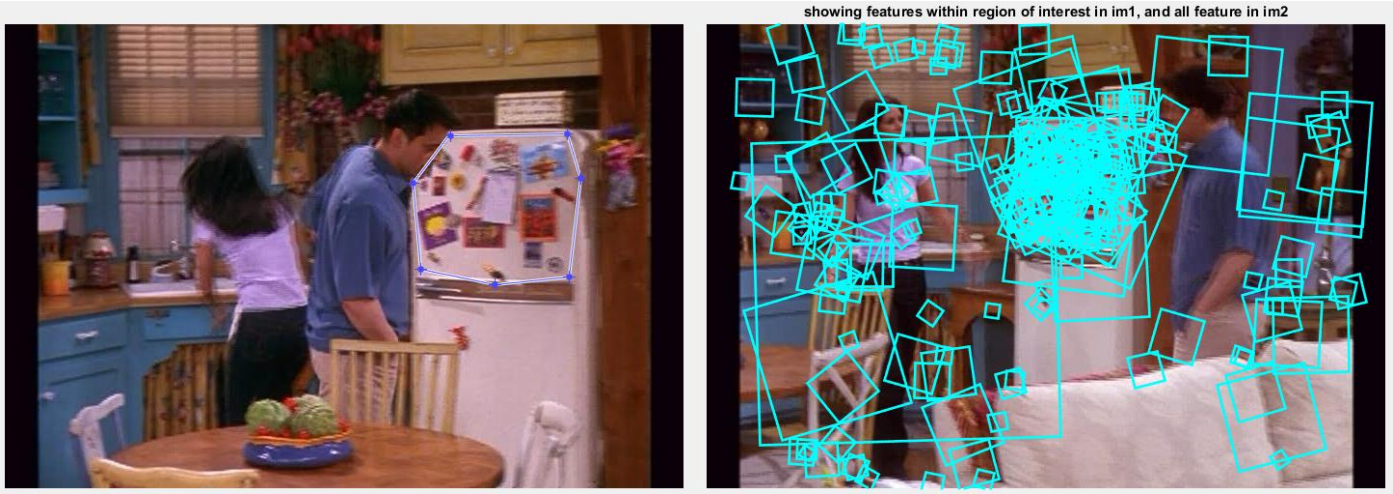
      (1) Each features which are found within b) should find a minimal distance between SIFT descriptors in second image. I use `dist2(x, c)` to find minimal SSD.

      (2) If the distance is > 0.3, we decide it mismatch.

      (3) If the ratio = $\frac{\text{best SSD}}{\text{second best SSD}}$ > 0.7, we decide it's ambiguous and not shown on image.

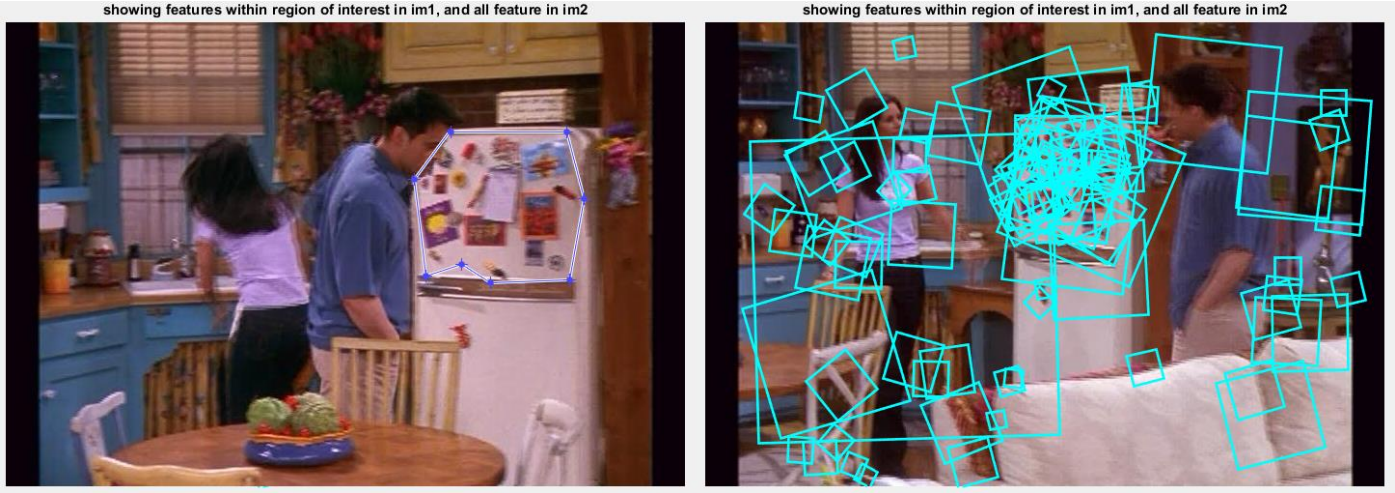
   d) Display only selected features on second image: I used `displaySIFTPatches` to implement this concept.

Result1: No threshold, No ratio. If there are 100 features selected in first image, there are also 100 matched features in second image without eliminating unsuitable mismatch. So the minimal SSD also may happen outside of the selected region.

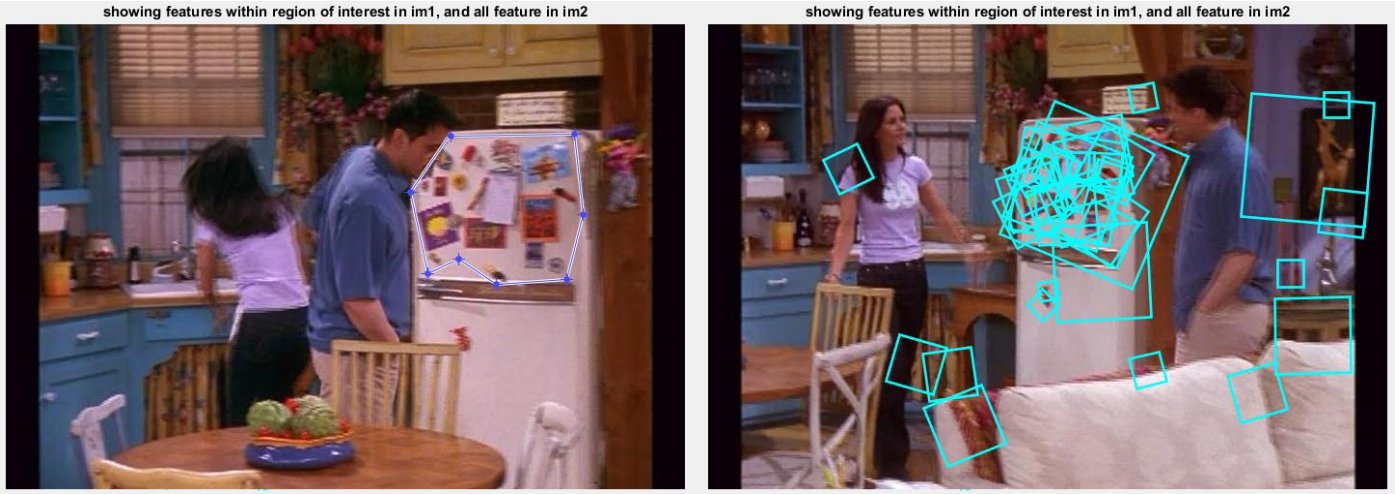


Result2: Threshold < 0.3, No ratio. We can eliminate a lot of mismatch that caused by larger SSD.

Result3: No threshold, ratio < 0.7. This one is used to deduct similar features. For example, features selected on blinds of windows are less than result1 which looks similar.

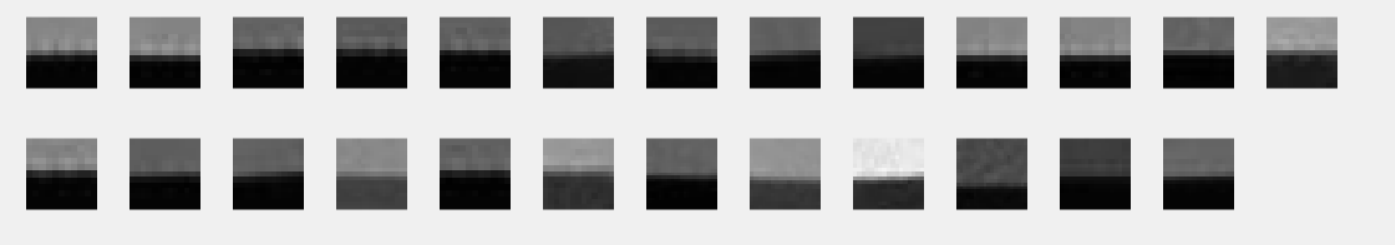


Result3: Threshold < 0.3, ratio < 0.7. We can eliminate most of the unrelated features.

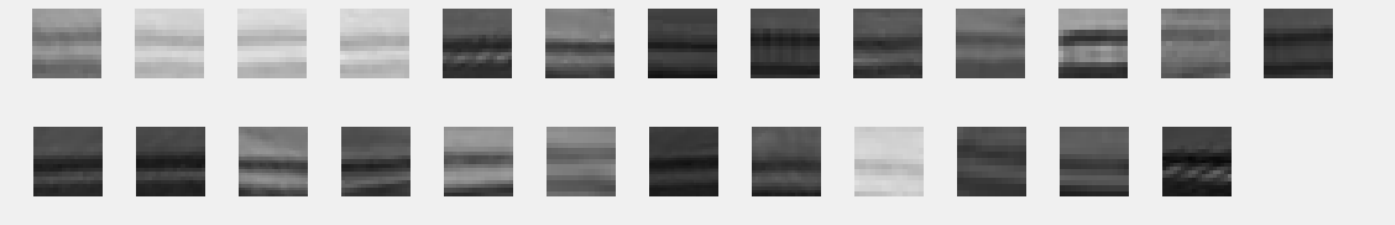


2. Visualizing the vocabulary
   a) [Initial] Initial data and randomly select 100 SIFT descriptors for each frames. So there are total 661,200 SIFT called `alldescriptors` are selected as training set for kmeans.

   b) [Initial] Find information of every selected SIFT which includes frames and order of that descriptor come from. I called it `alldescriptorsinfo`.
   c) [kmeans] Using `alldescriptors` as training set for kmeans. I used `kmeansML.m` to train my k cluster.
   d) [visual words] Get two visual words: one is the densest to center the other is 40th dense to center.
      (1) By membership, cataloging `alldescriptors` into K clusters.
      (2) Sorting each clusters by distance between centers.
      (3) Selected a densest to center, V1, and the other is 40th dense to center, V2, by sorted distance. (If I choose first and second dense cluster, the patch looks like similar.)
   e) [show patch] We have V1 and V2 clusters. Show the first 25 patch which are closet to center. By b), we can trace back the frames and order of every descriptor. Using `getPatchFromSIFTParameters` function to show each patch and for loop for 25 times.



Looks like the same but the features are not exactly the same.



3. Full frame queries
   a) Load kMeans matrix from previous question as center of visual words.
   b) Calculate bag of words of each frame which are illustrated in figure 1 and figure 2 below:
      Loop every frame
      (1) Load SIFT descriptors from each frame
      (2) Calculate distance between 1500 kMeans center and SIFT descriptors. In there, I used `dist2()` function.
      (3) Quantize every SIFT into a visual words and calculate its bag of words by `histc` function.

Figure 1

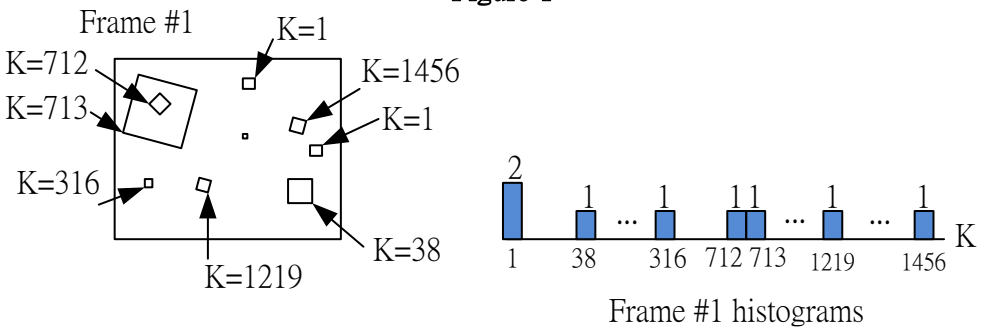


Figure 2

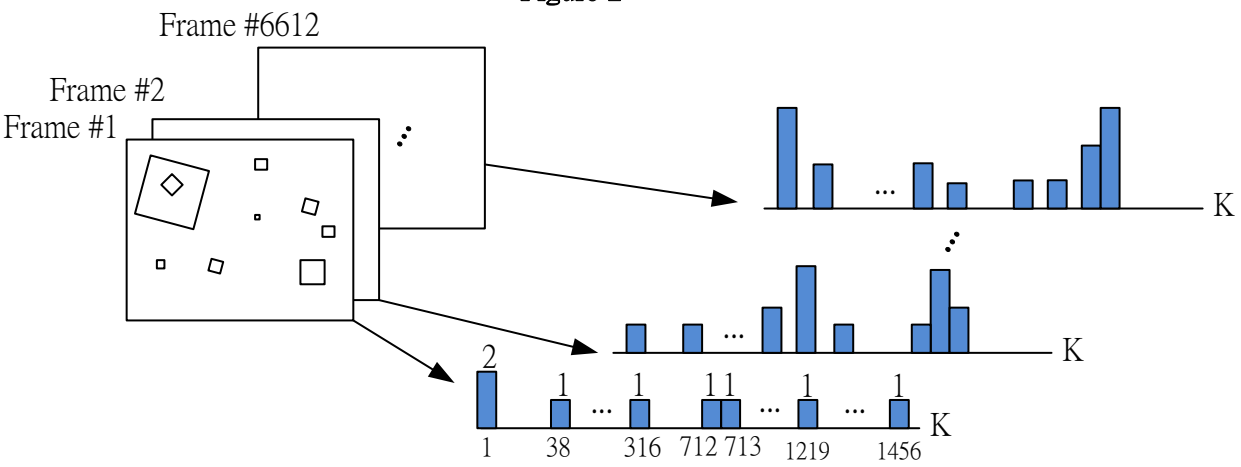
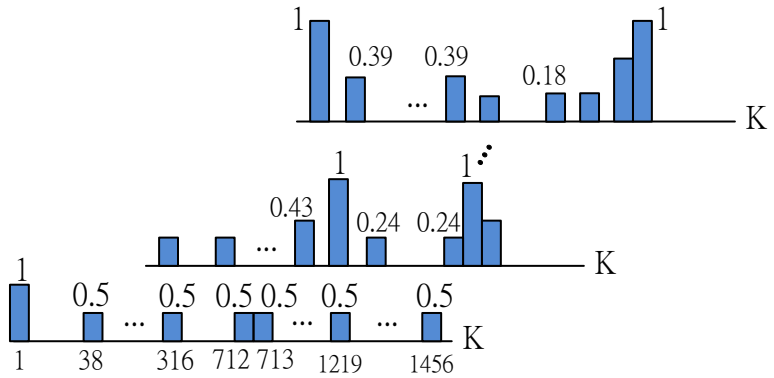


Figure 3 Normalized



   c) Normalized all bag of words histogram, shown in above figure 3.
   d) Find similarity frame:
      Select a target frame. Use `dot()` function to calculate similarity with the other 6611 frames.
      The top five high score are selected frames.
   e) Print six frames together.

Explain:

We can tell that this method works well. In the below Frame # 795, even most of the frames are not near that frame, we still can pick them correctly. But in most case, the most similar frames are the frames that nearby the selected frame.

[Frame # 795]

[Frame # 1500]

[Frame # 4500]

4. Region queries

   There are two methods to implement this question. First is building an inverted file index. The other method is using bag of words. For first methods, the result is more reliable. But building an inverted file index takes huge computation effort because of we have to include every features in every frames. If we have a high performance computer, we can choose this method.

   In this homework, I choose second method that is using bag of words to represent a selected region and use it to compare with other 6611 frames by cosine similarity. This method does not fully make sense since I compare a region with an image frame. However, I use `dot()` function to compute similarity. If there is some feature in an image frame not in selected region, the computing result of that feature is zero which means dissimilar. This is the same concept for us to compute frame to frame compute similarity in question three.

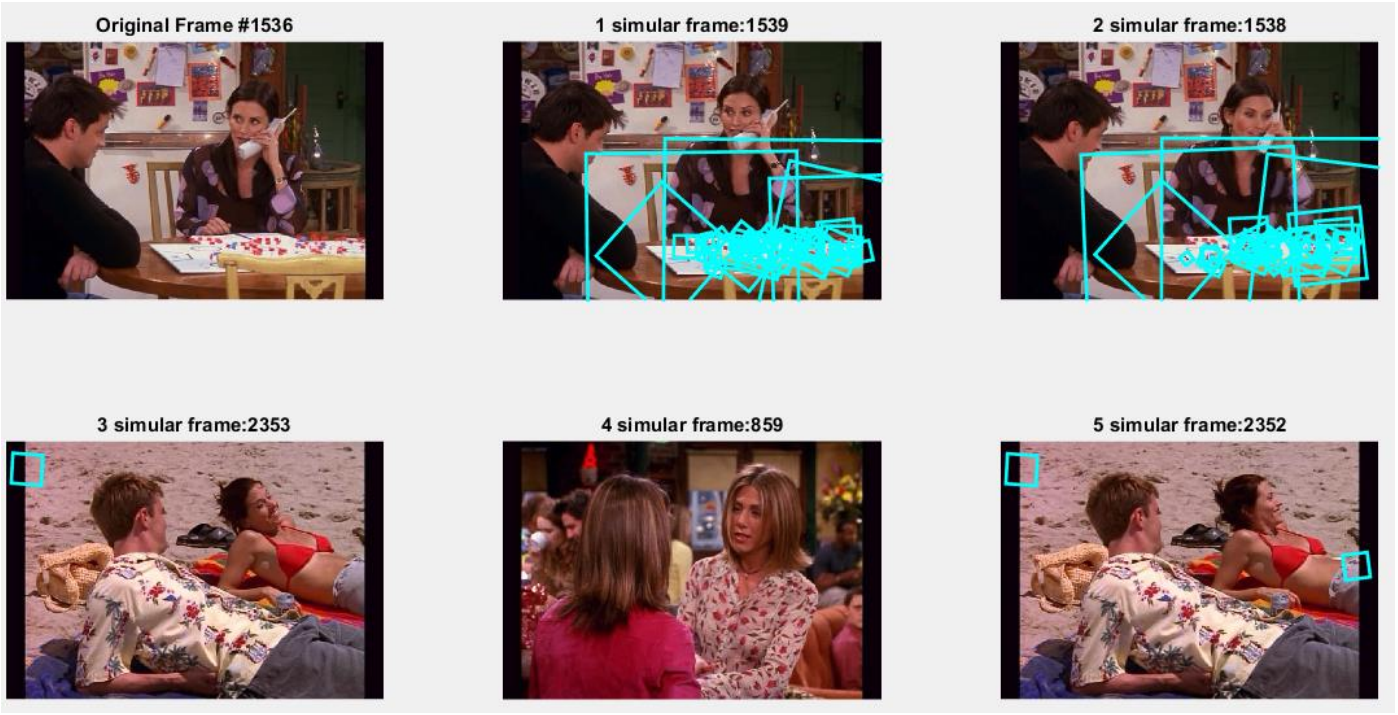
   But this some **disadvantages**. If there is some features not in the target object in other frames, it still increases cosine similarity score that interference our decision. The smaller the selected region is, the more interfere causes. Even worse, because we only compare part of frame with a whole frame, the cosine similarity score is relative low. In question 3, top five similarity is higher than 0.7, but in this question this value down to 0.5 or even worse.

   In below four examples, most top two frames are the target we want, but we may select some frames that are totally unrelated to the selected region. I marked the similar features in selected frames. There are even **NO** similar features in selected frame but selected because of relative high cosine score. That means **noise features truly interfere with our cosine score**. We can improve it by tf-idf method or spatial verification method in optional questions.

Below are my algorithm and the results.

a) Load Bag of Word matrix
b) Select region in a target frames. In here, I used `selectRegion` function.
c) After normalizing features within selected region as bag of words histogram, compare it with other 6611 frames. And find the top five cosine similar score.
d) Print the top five frames with similar feature patches. I used `displaySIFTPatches` function.

[Frame #1536]



[Frame #1598]

Original Frame #1598

1 simular frame:1274

2 simular frame:1337

3 simular frame:1276

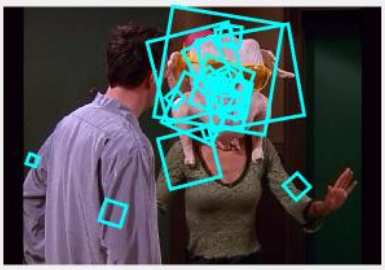
4 simular frame:1378

5 simular frame:1370

[Frame #2554]

Original Frame #2554

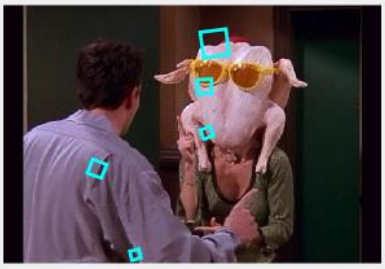
1 simular frame:2561



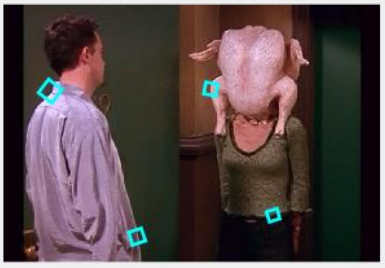
2 simular frame:2569

3 simular frame:2486

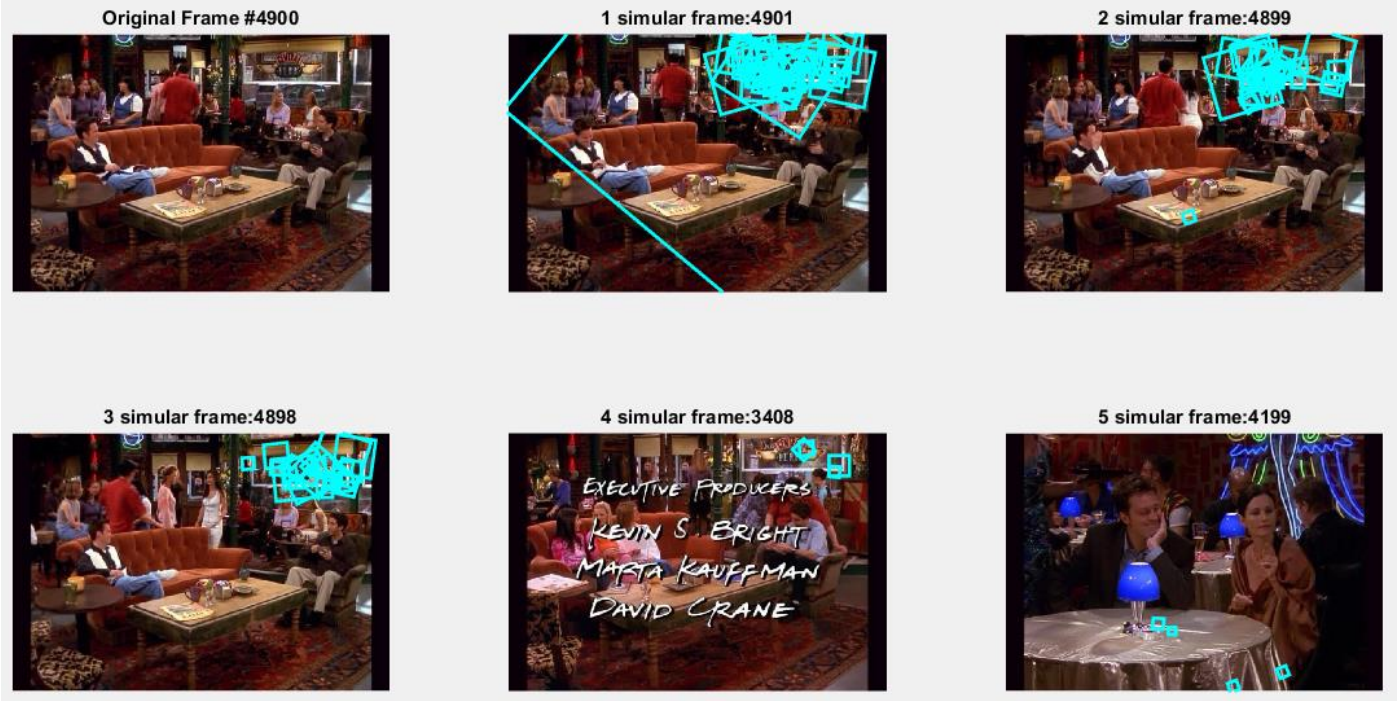
4 simular frame:2576



5 simular frame:2524



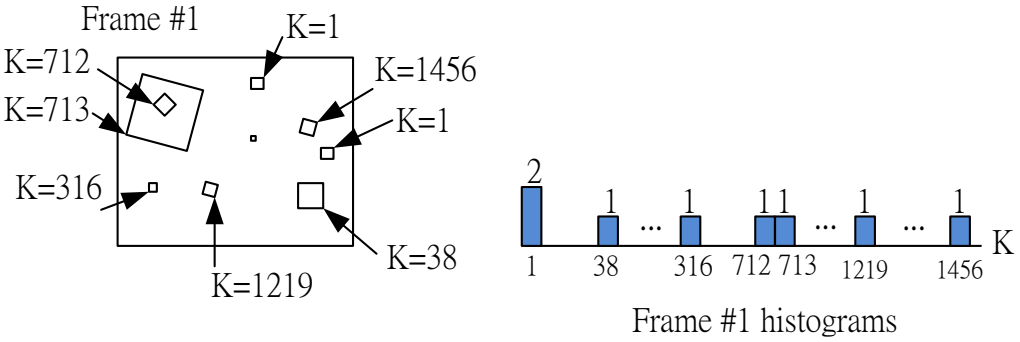
[Frame #4900]



III. OPTIONAL: Extra credit

1. Stop list and tf-idf: This one looks like question 4 Region queries but use tf-idf method to ignore very common words. Below is my algorithm.
   a) Load Bag of Words matrix from the result of question 2.
   b) Calculate Bag of Words with tf-idf, concepts are illustrated in below graph:
      (1) Calculate $t_i = \frac{n_{id}}{n_d} log \frac{N}{n_i}$ which ti is a weight of a visual word in new bag of words of a frame.
      We need to count ni first which is the number of documents word I occurs in whole database.
      (2) If a visual words occurs in every frames, $n_i = N$ and $log \frac{N}{n_i} = 0$. So this word does not shown in bag of words. And following the formula, we calculate Bag of Word of all frames.
      (3) Normalized bag of words.
   c) Selected region in a specific frames. Using b) method to calculate bag of words with tf-idf of selected region. In here, I used `selectRegion` function.
   d) After normalizing features within selected region as bag of words histogram, compare it with other 6611 frames. And find the top five cosine similar score.
   e) Print the top five frames with similar feature patches. I used `displaySIFTPatches` function.

Figure 1: Original bag of words



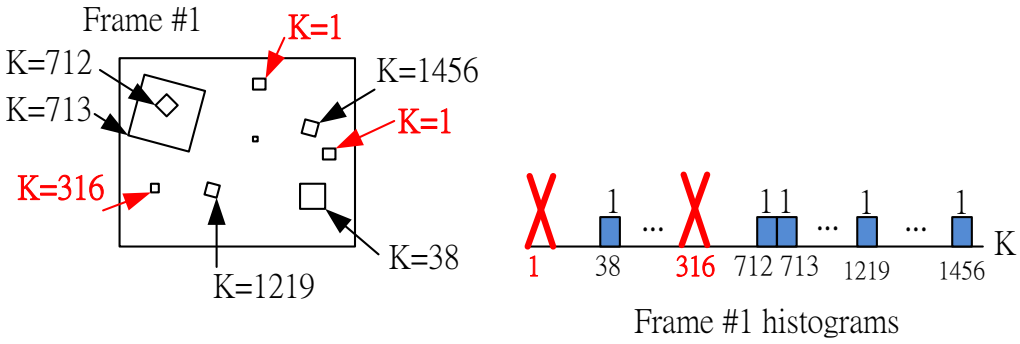


Figure 2: New bag of words Ex: After calculating all t_i, supposed we get t_1 and t_{316} are zero. So we eliminate them in bag of words histogram.

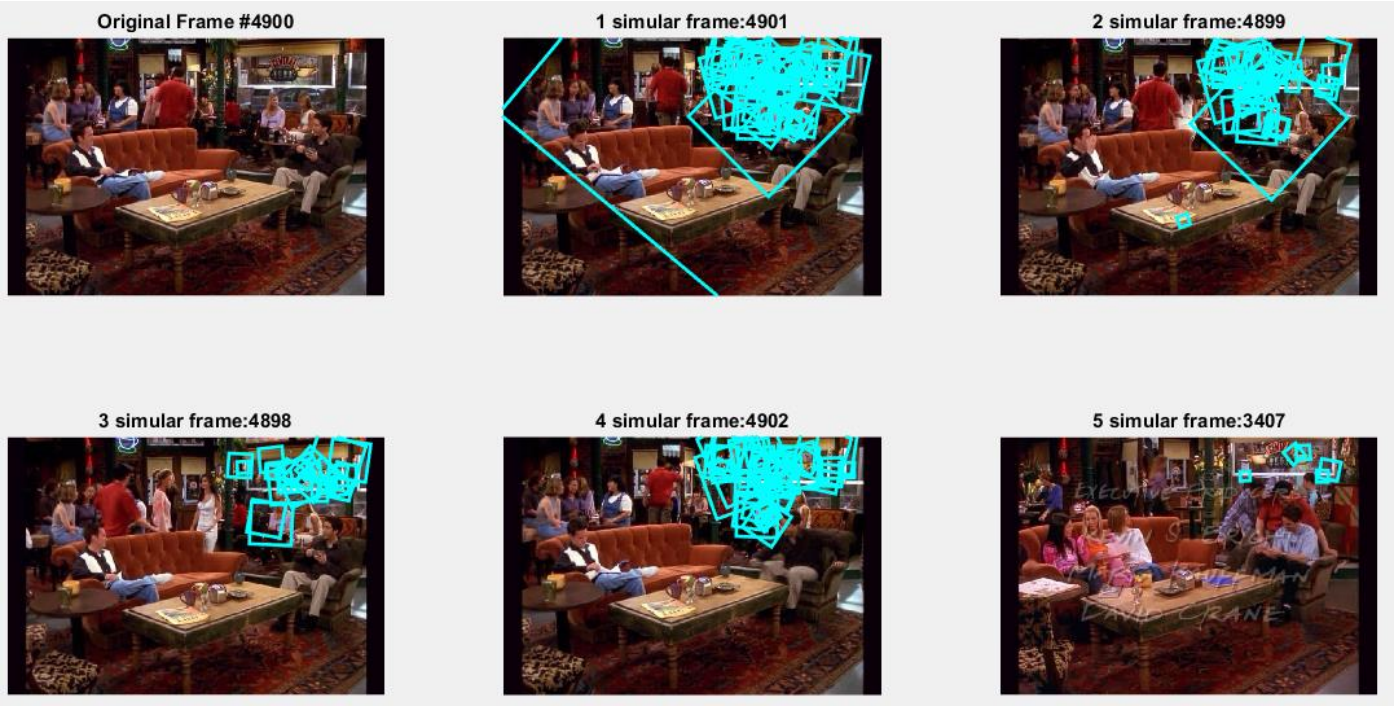
Using this algorithm, we can the result is more robust than question 4. But like the third examples frame #2121, we can easily select some objects that are similar with part of original features. To eliminate those cases, we should use spatial verification.

[NOTE] My code is in `regionQueriesextraone.m`

[Frame #1536]


Original Frame #1536
1 simular frame:1540
2 simular frame:1537
3 simular frame:1538
4 simular frame:1541
5 simular frame:1059

[Frame #4900]


Original Frame #4900
1 simular frame:4901
2 simular frame:4899
3 simular frame:4898
4 simular frame:4902
5 simular frame:3407

[Frame #2121]


Original Frame #2121
1 simular frame:2120
2 simular frame:5741
3 simular frame:5725
4 simular frame:3119
5 simular frame:3118