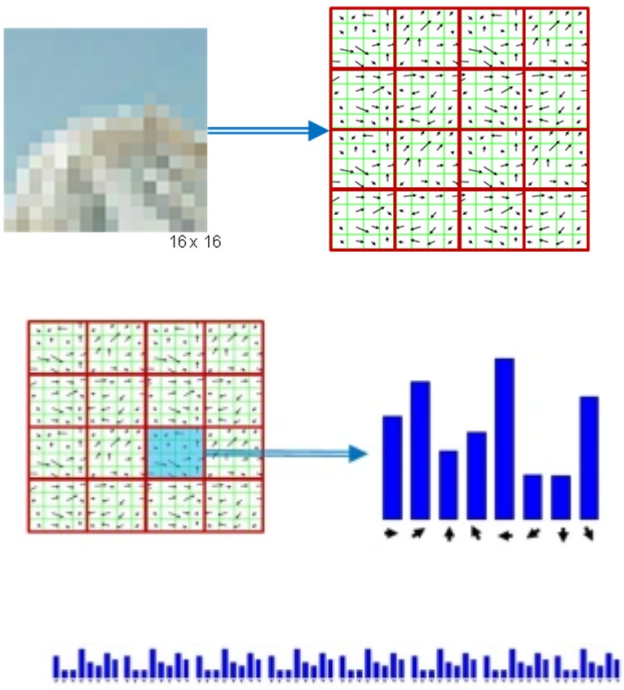
AGV

TASK-2(Video search)

RAW DESCRIPTOR MATCHER

Every SIFT value have a vector with 128 *1 as dim to represent itself which is invariant to rotation or scaling or illumination.
It represents Histograms of Oriented Gradients (HOG)



Above value is stored in128*1 array so each sift can be uniquely represented by this feature vector.
Hence to find similarity between two sift we compare their descriptor feature vector.

$$||a - b|| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

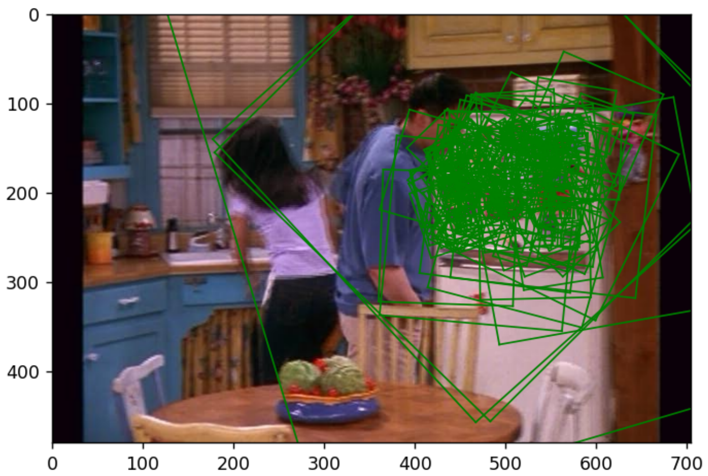
In my work I have implemented the Euclidean distance method.

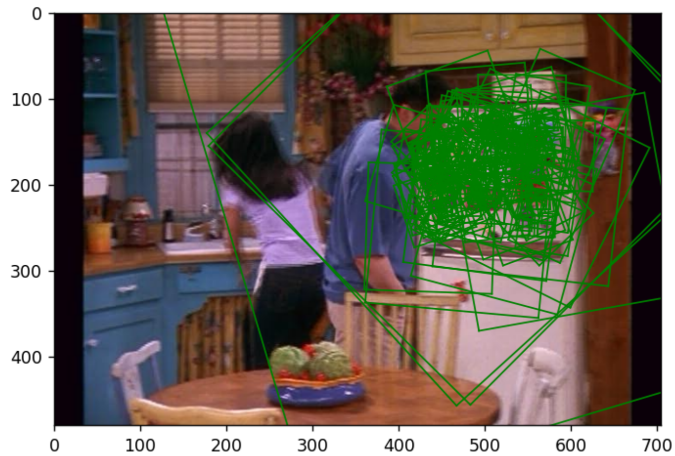
And then extracted the most similar sift in terms of distance in 128 dimensional space and stored its index in the matches list then showed it.
But not all sift need to be matched in the second image so we need to search only inside a fixed distance which can be controlled by threshold
So a sift to be matched from a sift its Euclidean distance should be less than 0.4
I have taken min for each sift to make sure only one sift matches to other because simply comparing threshold was making output very sensitive.

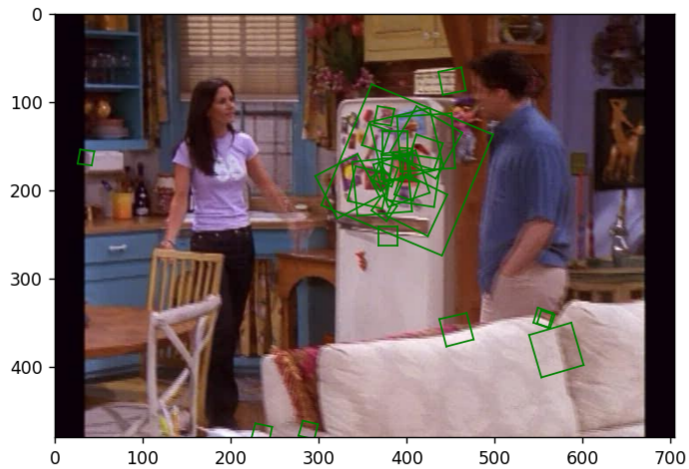
OUTPUT

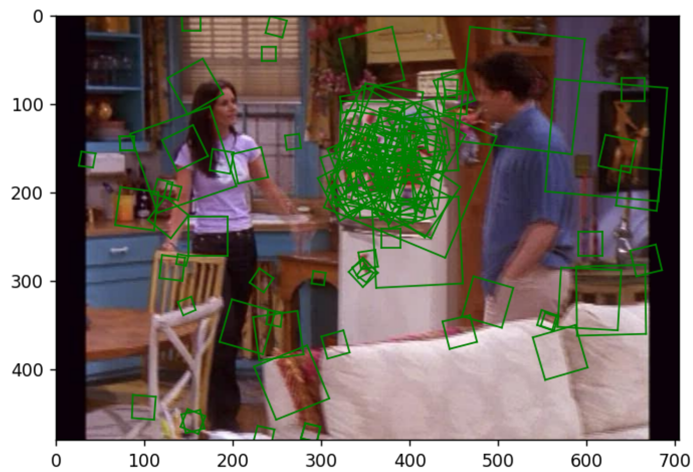
For threshold =0.4

For threshold=0.6









VISUAL VOCABULARY

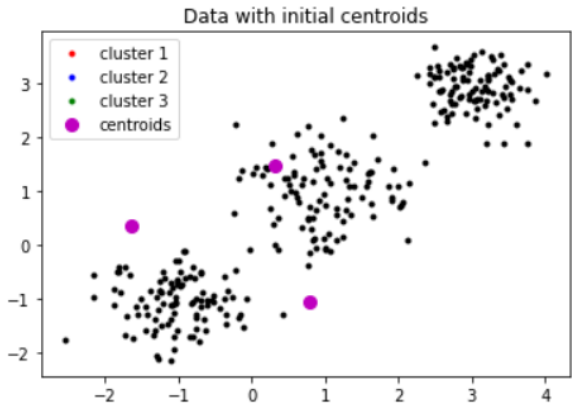
Here I have created a bag of words using all sift descriptors stacked from a number of frames 700 in my implementation.
Algorithms used:

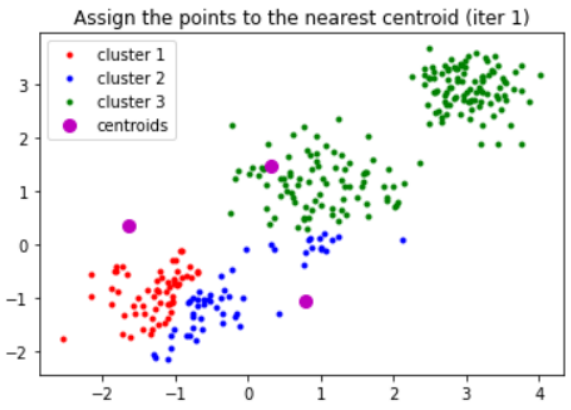
KMeans

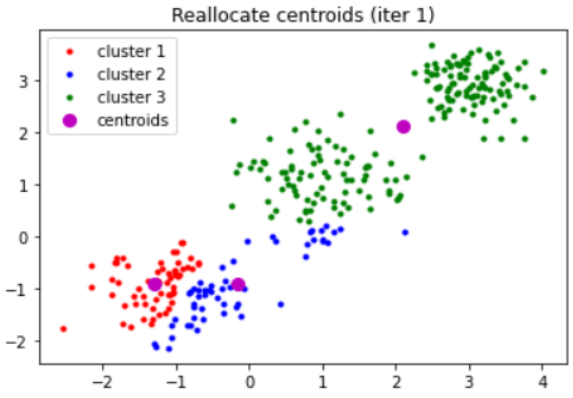
K-means is an unsupervised classification algorithm, also called clusterization, that groups objects into *k* groups based on their characteristics. The grouping is done minimising the sum of the distances between each object and the group or cluster centroid. The distance usually used is the quadratic or euclidean distance.

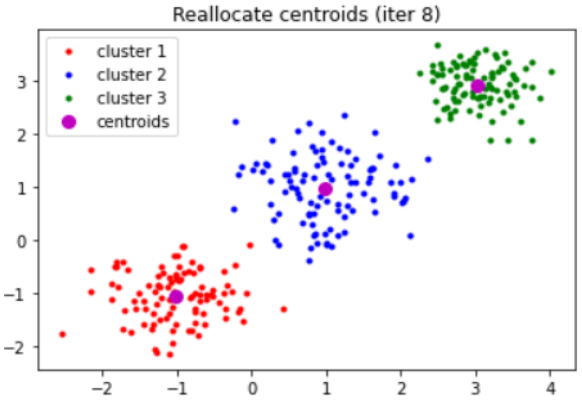
The algorithm has three steps:

1. **Initialization:** once the number of groups, *k* has been chosen, *k* centroids are established in the data space, for instance, choosing them randomly.
2. **Assignment of objects to the centroids:** each object of the data is assigned to its nearest centroid.
3. **Centroids update:** The position of the centroid of each group is updated taking as the new centroid the average position of the objects belonging to said group.









Final clustering after 8 iterations

In my implementation the dataset is of 816155 descriptors with 128*1 dimension each and a total of 1000 centre clustering centres.
So algorithms assign all 816155 points to a cluster of 1000 points.
Forming a bag of words.
Then I have chosen a particular clustering centre and showed all sifts associated with it.
Used a function load_model which is storing the created model using pickle which i can import again rather than recreating model again.

OUTPUT(showing sifts of 2 cluster centre respectively in 2 pages)








FULL FRAME QUERIES

To find similarity between two frames using its descriptors

1) Fit the descriptors in Kmeans to get bag of words for that frame
2) Create histogram for that frame which will be 1*1000 array showing number of sift found at each K_centre.
3) Do a normalised dot product to find similarity

$$similarity = \cos\theta = \frac{a.b}{||a||||b||} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2}\sqrt{\sum b_i^2}}$$

Higher this product is, the more the vectors are similar.

4) Doing this process for all images
5) Then sort all frames according to this score and take the top 5 frames, while making sure the same frame for which we are searching doesn't come under consideration.

Example with similarity score

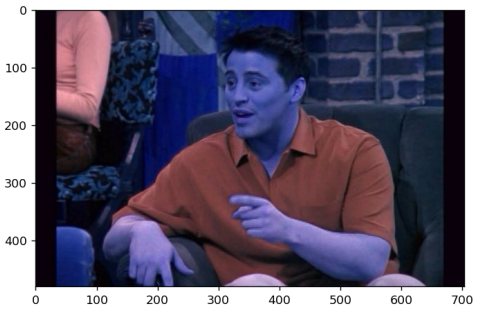
0.654940943

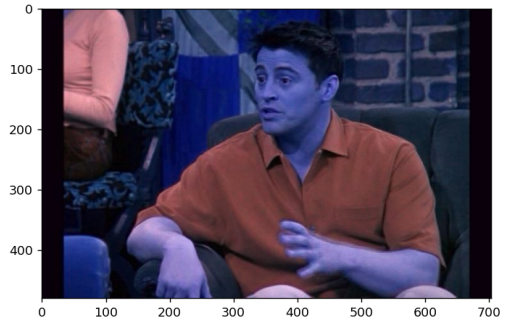
0.572391863

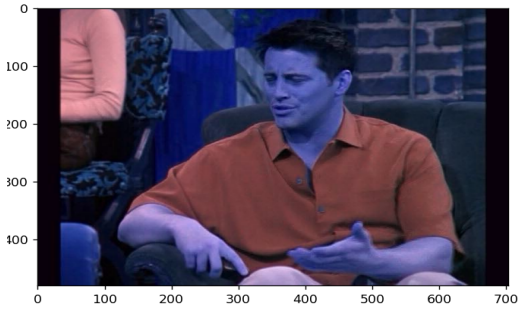
0.550399588

MORE OUTPUT

Query 1







Query 2

REGION QUERIES

Algorithm:

1) Get indexes of all sift in the selected region
2) Fit all the descriptors in KMeans giving us an associated bag of words for each sift.
3) Build a histogram containing only those centres which are associated with indexes found in the region , but use the entire image to create a histogram.
4) Then iterate through all the frames and using centres associated with founded indexes build histogram of frames.
5) Take a normalised dot product of two histograms and calculate the score.
6) Take the best 5 frames according to the score.

Example

Sample region

SIFT CENTRES FOUND

[8, 9, 67, 68, 69, 169, 172, 175, 176, 178, 179, 182, 259, 260, 261, 268, 375, 386, 392, 399, 400, 404, 405, 579, 582, 583, 590, 592, 610, 617, 865, 866, 870, 877, 884, 886, 892, 915, 935, 936, 938, 942, 1313, 1314, 1321, 1324, 1338, 1341, 1359, 1360, 1366, 1380, 1382, 1383, 1797, 1805, 1806, 1810, 1812, 1815, 1816, 1817, 1824, 2079, 2080, 2081]

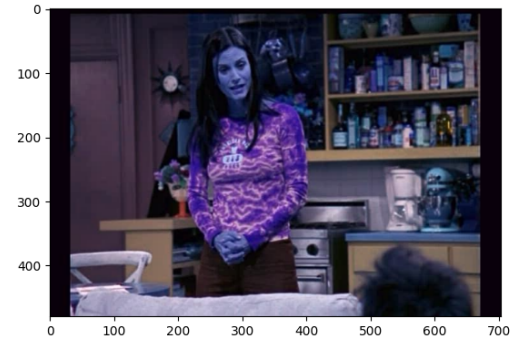
CLUSTERS CENTRES FOUND

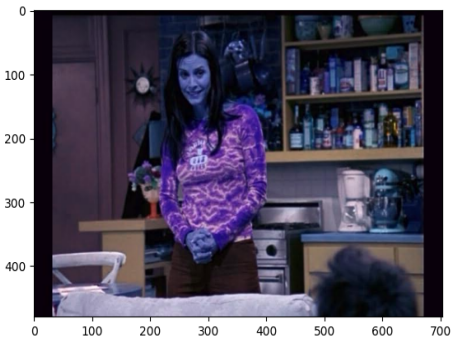
[561, 417, 134, 499, 538, 422, 975, 452, 747, 211, 96, 927, 219, 897, 630, 522, 598, 848, 60, 158, 590, 912, 992, 419, 331, 666, 149, 199, 554, 453, 777, 451, 976, 148, 863, 18, 899, 440, 677, 559, 190, 26, 847, 683, 330, 999, 72, 782, 870, 902, 225, 542, 920, 153, 472, 184, 394]

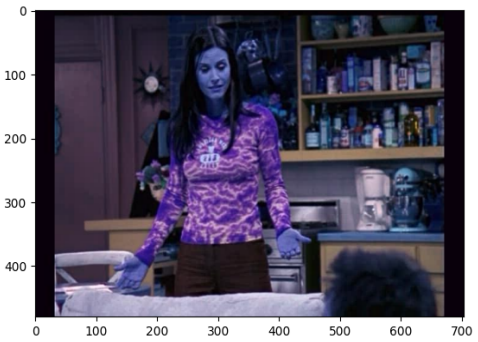
HISTOGRAM

[2. 4. 3. 3. 3. 3. 7. 4. 4. 5. 4. 5. 5. 4. 1. 3. 4. 1. 4. 2. 4. 3. 5. 1.
 2. 1. 4. 6. 2. 3. 2. 5. 1. 2. 2. 1. 2. 4. 3. 4. 3. 3. 1. 2. 2. 3. 3. 4.
 1. 1. 6. 1. 1. 4. 3. 2. 2.]

MATCHES (with scores)

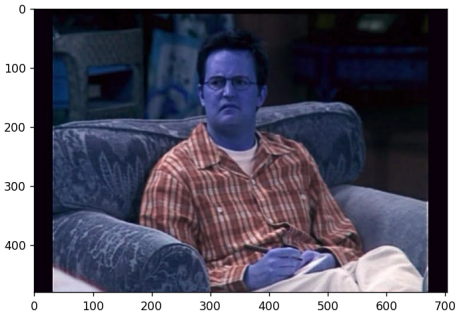




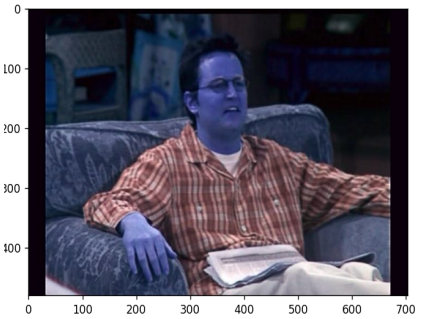


0.876292920452252 0.8566988073298427 0.8472683033511745

OUTPUTS

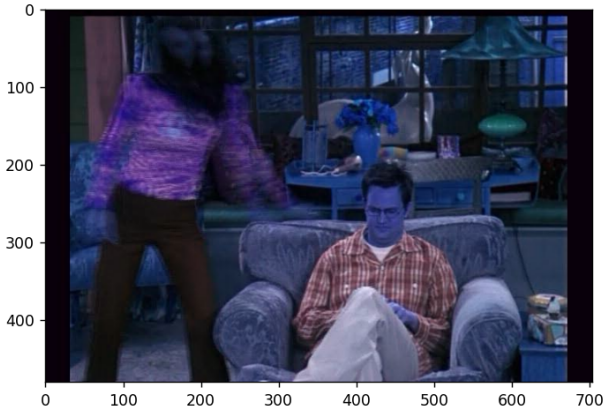


.9999999999999999(1)



0.9047457383408332(5)

0.821660740536164(9)

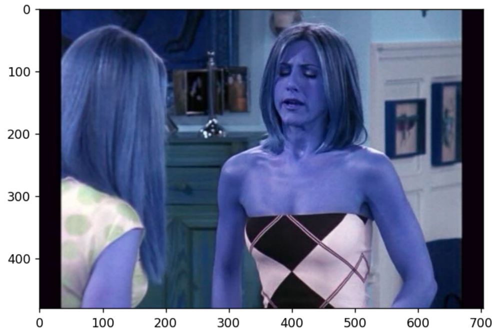


(output with different background)
0.7929706158492906(12)

MORE OUTPUT

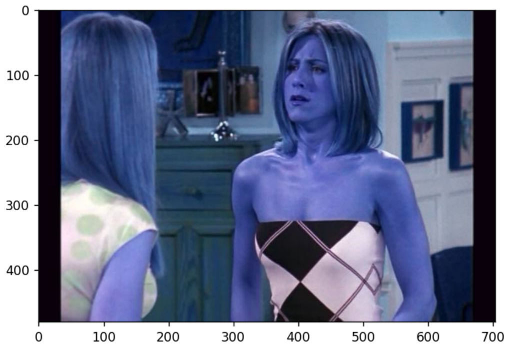

0 100 200 300 400
0 100 200 300 400 500 600 700

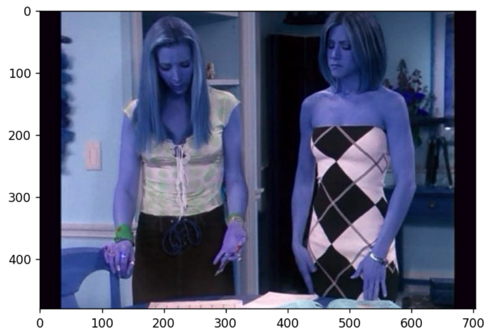

0 100 200 300 400
0 100 200 300 400 500 600 700

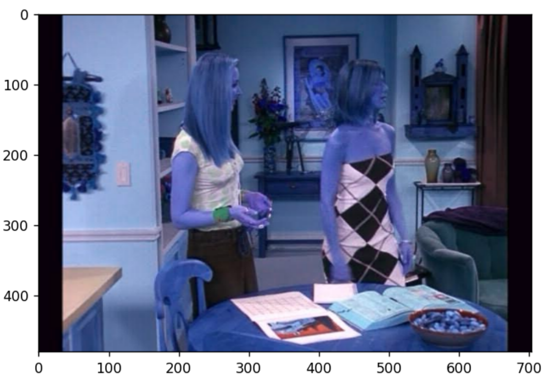

0 100 200 300 400
0 100 200 300 400 500 600 700

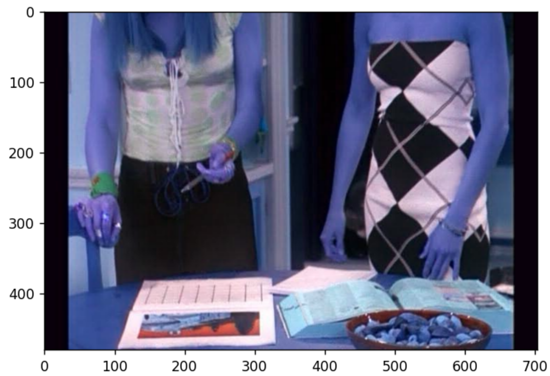

0 100 200 300 400
0 100 200 300 400 500 600 700


0 100 200 300 400
0 100 200 300 400 500 600 700


0 100 200 300 400
0 100 200 300 400 500 600 700


0 100 200 300 400
0 100 200 300 400 500 600 700


0 100 200 300 400
0 100 200 300 400 500 600 700


0 100 200 300 400
0 100 200 300 400 500 600 700