VIDEO SEARCH

Introduction - What is SIFT?

SIFT stands for **Scale-Invariant Feature Transform**. It is a popular computer vision algorithm used for feature extraction and object recognition tasks. SIFT was introduced by David Lowe in 1999 and has since become widely used in various applications.

The primary goal of SIFT is to find distinctive and **robust features** in an image that are **invariant to changes in scale, rotation, illumination, and partial occlusion.** These features are essential for matching and recognizing objects in different images, even when the objects appear under different conditions.

The SIFT algorithm can be broken down into several steps:

- **Scale-space extrema detection**: This step involves finding potential interest points in the image by looking for local extrema in the scale space, which represents the image at multiple scales.
- **Keypoint localization**: The algorithm refines the detected extrema by discarding low-contrast keypoints and those located on edges, as they may not be stable and repeatable.
- **Orientation assignment**: For each keypoint, an orientation is assigned based on the local image gradient direction. This allows SIFT to be invariant to image rotation.
- **Keypoint descriptor generation**: A feature descriptor is computed for each keypoint, which encodes information about the local image appearance around the keypoint location. This descriptor is designed to be invariant to changes in scale, rotation, and illumination.
- **Keypoint matching**: To recognize objects or find correspondences between images, the SIFT descriptors from different images are compared using distance metrics like Euclidean distance or cosine similarity. Keypoints with similar descriptors are considered as potential matches.

Raw Descriptor Matching

(Allow a user to select a region of interest in one frame, and then match descriptors in that region to descriptors in the second image based on Euclidean distance in SIFT space.)

Here we use the **roipoly** class given in **selectRegion.py** to select a region of interest in the image by drawing a polygon using mouse. The getIdx function is then used to give the indices of the SIFT features whose centers fall within the center of the selected region.

After extracting the keypoint descriptors from the selected region of the first image, we now have to find the matching descriptors in the second image. For this purpose we use the Ratio Test.

The Ratio Test is a technique used in feature matching to improve the accuracy and robustness of matching descriptors between two images. The ratio test compares the distance to the closest feature with the distance to the second closest feature. If the ratio of the two distances is below a certain threshold, the match is considered valid. Otherwise, it is rejected.

By using the ratio test, ambiguous matches are reduced, as it ensures that the closest match is significantly better than the next best match. This helps improve the accuracy and reliability of feature matching, especially in cases where some descriptors are more discriminative than others.

The matched features are then shown in the image as squares using Matplotlib.pyplot.
The final output is shown below.

Visualizing the vocabulary

(Build a visual vocabulary. Display example image patches associated with two of the visual words.)

Here, we build a visual vocabulary using the K- Means clustering algorithm.

In the context of machine learning, a **visual vocabulary** refers to a technique used in computer vision and image analysis to represent visual content in a more manageable and efficient manner. It involves creating a collection of visual features, often referred to as "visual words" that capture distinctive patterns or characteristics present in images. These visual words act as building blocks for representing and recognizing visual content.

K-means algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group.

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.

2. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

To implement K-Means clustering, we make use of **KMeans** in **sklearn.cluster.**

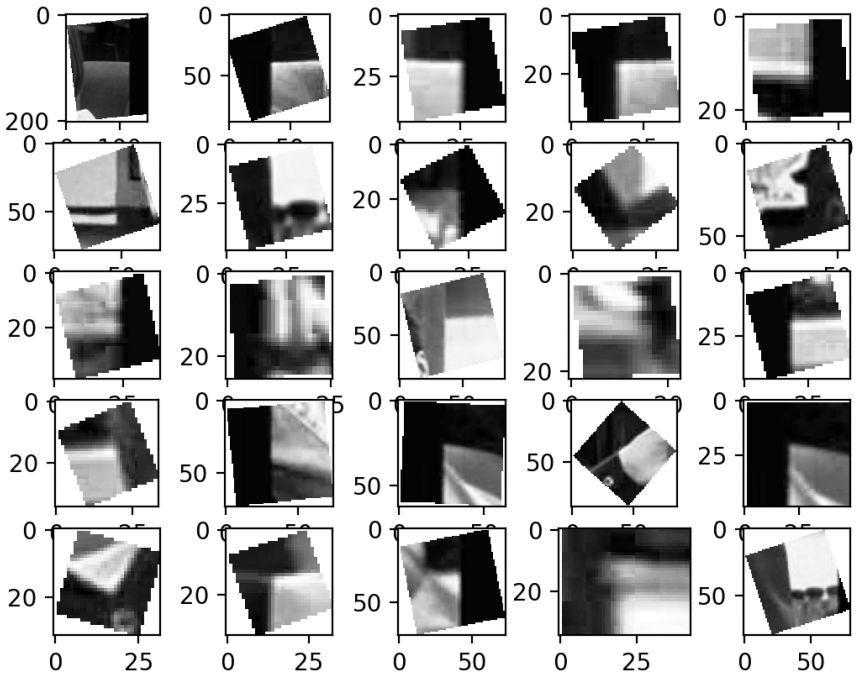
Further in the program, we use first n images, say 15 images and find out sift features corresponding to two cluster centers(visual words).

We also make use of getPatchFromSIFTParameters.py to extract the patch corresponding to the SIFT feature corresponding to a specific visual word.

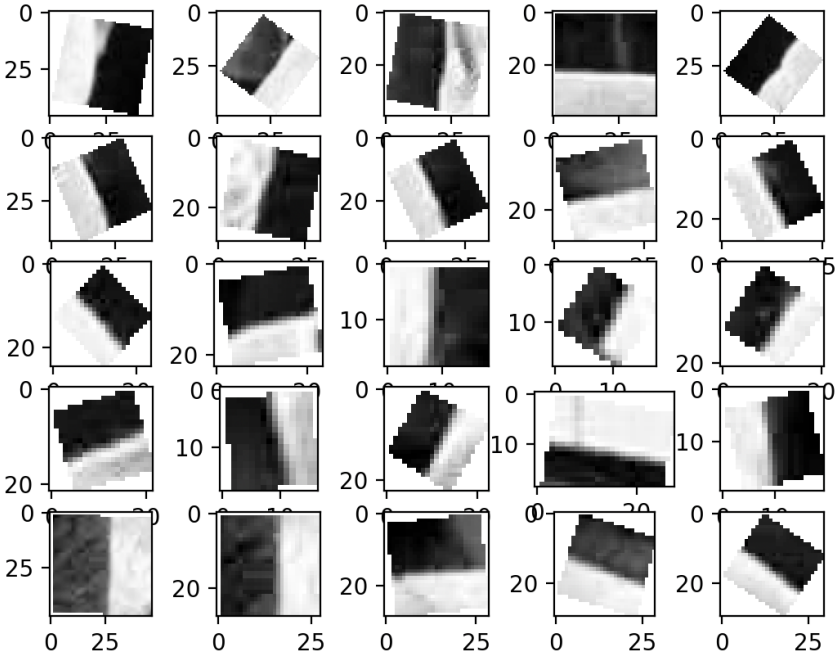
We first show the SIFT patches corresponding to the chosen two visual words simultaneously with different coloured squares, in different images. One such image with depicted SIFT features is shown below.

Then we show 25 of the extracted SIFT features corresponding to each of the two visual words.

Visual Word 1



Visual Word 2



Full frame queries

(Choose 3 different frames from the entire video dataset to serve as Queries and display the M=5 most similar frames to each of these queries)

Using the k-means model, as we specified in the previous subtask, we build a bag **of visual words histograms** for some images(say first 300 images). In computer vision, a *bag of visual words* is a vector of occurrence counts of a vocabulary of local image features. For this purpose, we use kmeans.predict() function which gives a list of cluster centers corresponding to the descriptors present in the image. Using this information we can easily build a histogram, or a **frequency vector** for all images.

Similarity(S) between two images(Ii and Ij) is shown by **normalized scalar product** of their frequency vectors(F).

$$S(I_i, I_j) = F(I_i) \cdot F(I_j) / ||F(I_i)|| \, ||F(I_j)||$$

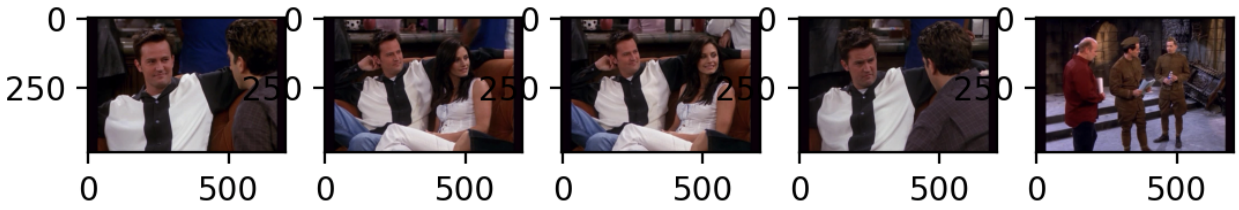
where S() is the similarity score. ||F(Ii)|| is the L2 norm of F(Ii).

To calculate the L2 norm of a vector, take the square root of the sum of the squared vector values. Another name for the L2 norm of a vector is Euclidean distance.

Now we find the most similar images to the query images as the ones which have the top 5 highest similarity scores.

An example is shown below:

5 Most Similar Images



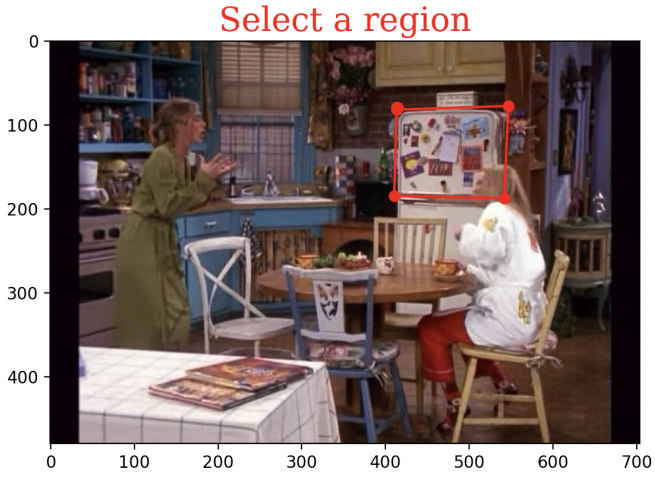
Region queries

(Select query regions from within 4 frames (which may be different than those used above) to demonstrate the retrieved frames when only a portion of the SIFT descriptors are used to form a bag of words.)

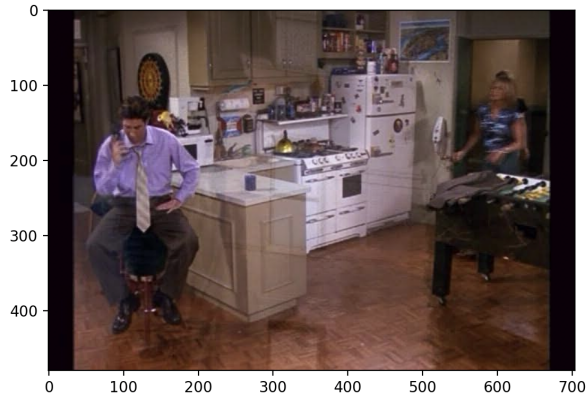
We select a query region in a frame and build a bag of visual words histograms for the features that are centralized in the selected frame. For all the other frames we build the histogram for all SIFT features in the frame.

Now we create a special dot product function to check similarity between the two images wherein we take the dot product of only those values in the two histograms, whose frequency in the query images' histogram is non-zero. Basically, what we are doing is that we are looking at how much two images are similar with reference to only those features that have their center in the query region in the query image.

We then show five most similar images according to the above criteria of similarity.
For example,



Showing 5 most similar images according to the algorithm:



NOTE: In our code, we have trained our K-Means Model for a very small subset of image data provided. So errors are inevitable!