

# Object Search in Videos

Chaitra Jambigi

Department of Computational and Data Sciences

Indian Institute of Science, Bengaluru, India

chaitraj@iisc.ac.in

## Abstract

*[] The aim of this project is to implement a technique for object retrieval which searches for all the occurrences of an object in a video, given a query image of the object. From the given video frames, the user has to select an object of interest, and the method will retrieve all those frames of the video containing the selected object. A text retrieval approach using Bag of Words (Bow) model is used for object search. Feature descriptors are extracted from the video frames and each frame is represented as a vector using the Term Frequency-Inverse Document Frequency (TF-IDF) weight matrix. Based on similarity with query frame, frames are ranked. A novel approach is proposed which represents the image frames as a set, and uses Jacard similarity to rank the frames. An approximate method is used for computing Jacard similarity, which is the Min-Hash algorithm. Results based on ranking is shown for both algorithms, and comparisons based on Precision values is made.*

## 1. Introduction

Information retrieval often demands quick and accurate access to our needed information from the entire collection of data. It is the process of finding relevant data from the collection of resources [4] This can be thought of in many different contexts like searching for a book in a library, searching for a word in a dictionary, searching for some article on the internet. As the amount of available information grows, it becomes more and more difficult to search for the relevant information. Thus there is a need for efficient information search techniques.

### 1.1. Problem Statement

We can use the same analogy and define our problem of Object search in a video. Suppose we have a 100's of hours video and we want to know a particular object or a person appears at which frame in the entire video, then it is a tedious task to watch the entire video and waiting to see the

object of our interest appear in the video. So this project aims to implement a technique which retrieves frames containing a particular object with ease speed and accuracy [2] The tool takes an input video, does the preprocessing of video and extracts information from it which is all part of offline process. Then at run time, the user selects an object to be searched, and the tool retrieves the frames which contain the object. Object search problem is also often called as Instance search in computer vision literature. The object search here is different from object tracking as we are not interested in how a particular object is moving in the video. Also it is different from object detection as it does not contain the detection of category-level objects [2] For example, if we want to look for an actress in a movie, we are not interested in human detection, but we are interested in where the actresses appears in the movie. Some good examples of object search include finding a particular logo, or special landmark like Eiffel Tower. Example of Object search can be seen in Figure 1 and 2.

### 1.2. Challenges involved

The object to be queried may be very small in size and be present in only few frames in the video. Then searching for such object becomes difficult. The object may be partially or fully occluded in some frames. Also there may be issues due to camera view point variations and illumination changes. If a particular person is to be detected, he/she might be in a different pose or at a different angle, as compared to the query image, thus identifying him/her becomes difficult.

### 1.3. Approach used

Now a very obvious question would be what information we should extract from the video frames so that we can match it with the new query object. Typically an object can be represented by a set of overlapping regions, each represented by a viewpoint invariant region descriptor computed from the region's appearance [2] Descriptors can be computed for each frames. These descriptors are then clustered into K clusters and each frame is represented as a vec-



Figure 1. Example of selecting a Query object. In the figure to right is the zoomed query object

tor with weights given by number of clusters in that frame. Similar vector is computed for the query image and recognition of a new object is done by nearest neighbor matching of the frame vectors. This type of approach can be related to the text retrieval. As an analogy to text retrieval, each word can be related to cluster of descriptors. Thus using the Bag of words (BoW) model of text retrieval, we can effectively solve this problem of object search similar to text retrieval.

The benefit of this approach is that matches are effectively pre-computed so that at run-time, frames containing any particular object can be retrieved with no delay [2] Thus any object occurring in the video can be retrieved even though there was no specific interest in this object when descriptors were built.

In this project for the baseline implementation of [2], we are extracting the Harris Laplace interest points in each video frame and the interest point is represented by a 128 dimensional SIFT descriptor. Descriptors are computed for each frame. It is followed by vector quantizing the descriptors by clustering them in some K clusters. Each of the descriptors in video frames are then assigned to the nearest cluster, thus creating a visual vocabulary. Once the vocabulary is built, each video frame is represented by a vector with values given by the tf-idf weighting values, as in text retrieval case. Whenever a new query object is searched for, its feature descriptors are computed and it is also represented as a vector using tf-idf weightings. Then the cosine similarity is computed between this query object and all other frames to give a ranking for top matches. The detailed description for the implementation will be covered in section 2.

#### 1.4. Novel approach

The choice of image representation and the distance metric used affects the data stored per frame and time to retrieve the matching frames at run-time. In the approach using tf-idf, the vector representation of each frame is found to be a sparse one. Not all descriptors belong to each frame, thus many of the entries are 0 still stored for each frame. Thus

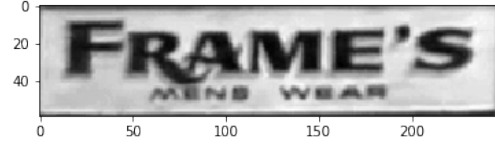


Figure 2. Query object to be searched

a new similarity measure called Jaccard's similarity is used to calculate the similarity between the query object and all other frames. Then the frames are ranked based on similarity measure and top k frames are retrieved. Here we need to store only those distinct clusters to which a descriptor belongs in each frame. Also this method gives better confidence values during object retrieval. Details of this approach will be covered in section 3.

#### 1.5. Chapter outline

Section 2 gives the detailed description of baseline work which is the implementation of [2] Section 3 gives a detailed description of the novel approach proposed. Section 4 covers various experiments carried out and results obtained.

#### 1.6. Review of Text Retrieval

Text retrieval systems generally employ a number of standard steps [3]: the documents are first parsed into words and words are represented by their stems, for example, 'talk', 'talking', 'talks' would be represented by 'talk'. A corpus contains many documents and each document contains many words. A unique identifier is assigned to each word and each document is represented by a vector with components given by the frequency of occurrence of the words the document contains. Also the components are weighted by their inverse document frequency. All these steps are carried out in advance and the set of vectors representing the documents are organized as inverted file [1] for efficient retrieval. We can establish an analogy between text retrieval and object search by relating the video of frames with corpus of documents, a frame with descriptors to a document with words and the cluster of descriptors as visual words. This is a very strong analogy and based on this analogy, BoW model is used for Object search.

### 2. Algorithm Description

#### 2.1. Viewpoint invariant descriptors:

For a given video, frames are captured at the rate of 1 frame per second. It is observed that there is not much movement within 1 sec in a video, thus for our purpose, we can effectively use 1 frame per second to do the processing.

On each of the captured frame, Harris Laplace interest points are detected. For a 1280x720 pixel video frame, the

number of interest points per frame are around 600. Each interest point is then represented by a 128 dimensional vector using SIFT descriptor. SIFT has superior performance than other descriptors because it is designed to be invariant to a shift of a few pixels in the key points. Analysis is done on grey scale images and there not much difference is observed when color information is used.

## 2.2. Building a visual vocabulary

The objective here is to vector quantize the descriptors into clusters which will be the visual words for text retrieval [2]. All the descriptors from all the frames are clustered using K means clustering and we get K centroids. Each of this cluster acts like a visual word. For a 2 minute video, we get around 90 frames with 1 fps. And total of around 40K descriptors, which are clustered in 2K clusters. After experimentation, the number of cluster was chosen empirically to maximize the matching performance.

The Mahalanobis distance metric is used in K means clustering. Mahalanobis distance is the distance between two points in multivariate space. This distance is used when the vectors are correlated. If the vectors are uncorrelated Mahalanobis distance is same as Euclidean distance. Experiments were carried out using Euclidean distance, but it didn't give good results. The Mahalanobis distance between any 2 vectors  $x_1$  and  $x_2$  is given by,

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)}$$

where  $\Sigma$  is the covariance matrix between the data points

## 2.3. Creating TF-IDF Matrix

In text retrieval each document can be represented as a vector of word frequencies. Usually weighting is applied to the frequency calculation. The most common way of describing a text document as a vector is by using the Term frequency- Inverse document frequency (TF-IDF) weighting scheme. Say we have some  $v$  words in a vocabulary then each document  $d$  can be represented by a vector having  $v$  components. Now since in our visual words analogy, we have K clusters, each cluster acts like 1 word, thus each of our video frame can be represented as a vector having K components as,

$$v_d = (t_1, \dots, t_i, \dots, t_K)^T$$

where the weighted components are given as,

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Describing the notations and tf-idf calculation in terms of bag of visual words model, first term in product is Term frequency and second term is inverse document frequency.

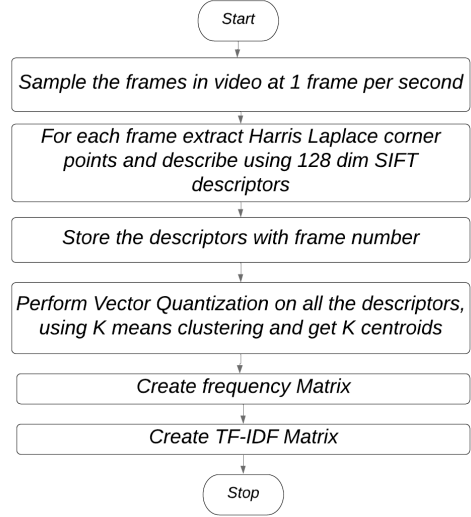


Figure 3. Flowchart of the offline process for Object search using TF-IDF weighting

$n_{id}$  is number of descriptors in  $i^{th}$  cluster for  $d^{th}$  frame,  $n_d$  is the total number of descriptors in frame  $d$ ,  $n_i$  is the number of documents which contain cluster  $i$  at least once,  $N$  is the total number of frames

As per [2] the intuition to use this weighting scheme can be that the word frequency weights words occurring often in a particular document and thus describes it well, whilst the inverse document frequency down weights words that appear often in the database. Thus using the above formula, TF-IDF matrix is calculated for the entire video frames. Figure 3 gives the flowchart of this offline process. The retrieval stage is the run time or online process.

## 2.4. Retrieval Stage

At the retrieval stage, the user selects the query object to be searched for. The descriptors are calculated for this object and similar vector quantization is performed for this query object. Thus we get a vector representation of size K for this object. Now we need to find the similarity between this weighted vector which represents query object with the frame vectors which is represented by the TF-IDF matrix. Mathematically this similarity is computed by simple cosine similarity given by,

$$f_d = \frac{v_q^T v_d}{\sqrt{v_q^T v_q} \sqrt{v_d^T v_d}}$$

where  $v_q$  is the query object vector,  $v_d$  is one of the frame vector, and  $f_d$  is the similarity for that frame. Once the similarity is computed for all the frames, all frames are ranked according to their similarity.

Performance of the method can be judged by seeing top L frames, or else adding a threshold on similarity values which can also be called as confidence values. Here we have taken threshold to be 50%

### 3. Novel approach proposed

#### 3.1. Vector Quantization

A new way of vector quantization of the video frame is proposed. The process upto finding the K centroids remains same, but here the vector representation and similarity calculation is changed. In the previous approach, we had to calculate the tf-idf values for each frame, it needed a K length vector for each frame. Thus if there are 1000s of frames, we need to store a matrix of size  $1000 \times K$ . However this matrix is a sparse matrix as each frame won't have all the K clusters to have some value. Naturally most of the components would be 0 leading to unnecessary storage and comparison. Thus a new approach can be used where for each frame, we store only the number of distinct clusters to which the descriptors of that frame belong. Thus each frame is now represented as a set of clusters which are present in that frame as,

$$s_d = \{t_1, \dots, t_i, \dots, t_{l_d}\}$$

where  $l_d$  is the number of clusters present in  $d^{th}$  frame and  $t_i$  is the cluster number from 1 to K

This is a weaker representation as each vector does not store the number of descriptors/features but only whether a cluster descriptor is present or not. So the novelty here lies in recognizing that by denoting frames as sets, we can still compute the similarity effectively.

#### 3.2. Jaccard's similarity

Now that we have each frame as a set, we can find out the similarity between the frames and query object set, using a set similarity metric. We have used Jaccard's similarity which is defined between two sets A and B as,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

and

$$0 \leq J(A, B) \leq 1$$

The distance between 2 frames is computed as the similarity between them which is defined as the ratio of number of elements in intersection over number of elements in Union. So far this looks good, but the issue comes when we look at the compute requirement involved in doing this similarity measure between a huge number of frames. To compare 2 frames with N elements requires comparisons equal to

$$\binom{N}{2} \approx \frac{N^2}{2}$$

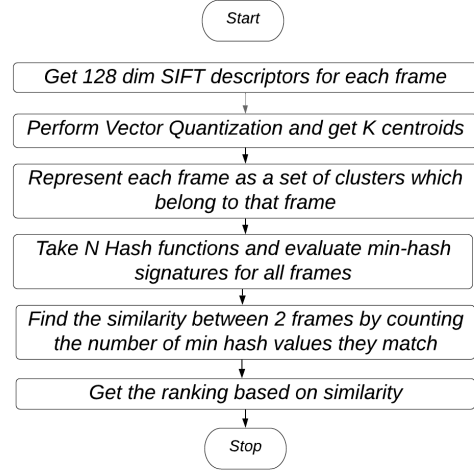


Figure 4. Flowchart for Object search using Min-Hash algorithm

and doing this for 1000s of frames is a computationally expensive task. Thus to solve this issue, Jaccard's similarity can be approximated by a fast randomized algorithm called Min-Hash algorithm.

#### 3.3. Min-Hash algorithm

Min-Hash algorithm effectively approximates Jaccard's similarity. Algorithm is described below and why it gives Jaccard's similarity is explained in next section.

This algorithm uses Hash functions. A Hash function is any function that can be used to map data of arbitrary size to a fixed size values. It basically projects a value from a set with many number of elements to a set with a fixed number of elements. Eg.  $h(x) = (ax + b)\%c$  The value of  $h(x)$  lies between 0, c. Thus by changing values of a, b we get different permutations of hash functions.

If the size of set is L then we have a set of L values corresponding to 1 hash function. Find the minimum value (that is why the name of algorithm is min-hash) of this set and store it. In min-hash algorithm, we take N such hash functions, and compute the minimum of hash values for a set. Thus after applying N hash functions on a set of size L, we get an output set of size N. This set is the min-hash signature of the input set. We need to compute the min hash signatures for all the frames in our case. Then we compare each of the frames by counting the number of signature components in which they match. The higher the number of matching components, the more is the similarity between the frames. Since this is an approximation to Jaccard's similarity, the accuracy of result increases as we increase the number of hash functions used, N. But it comes with a compromise on time and space requirements for computation




	Query Image 1		Query Image 2	
				
	<b>TFIDF</b>	<b>MIN HASH</b>	<b>TFIDF</b>	<b>MIN HASH</b>
<b>Result 1</b>	Frame 57, confidence = 100 	Frame 57, confidence = 100 	Frame 7, confidence = 100 	Frame 7, confidence = 100 
<b>Result 2</b>	Frame 49, confidence = 44.408 	Frame 22, confidence = 63.61 	Frame 49, confidence = 70.57 	Frame 57, confidence = 35.48 
<b>Result 3</b>	Frame 22, confidence = 43.647 	Frame 14, confidence = 50.5 	Frame 54, confidence = 39.96 	Frame 2, confidence = 32.43 
<b>Result 4</b>	Frame 14, confidence = 37.489 	Frame 41, confidence = 42.9 	Frame 21, confidence = 29.5 	Frame 54, confidence = 29.43 

Figure 5. Top 4 retrieved frames with confidence values, for 2 query objects, for both TF-IDF and MinHash algorithms










	HOG Features	Colour Histogram of spatial pyramid	Colour Histogram
<b>Rank 1</b>	Frame 39, Distance: 0.647876 	Frame 41, Distance: 0.59874742375131 	Frame 14, Distance: 0.51794405488485 
<b>Rank 2</b>	Frame 51, Distance: 0.240054 	Frame 56, Distance: 0.562999870549471 	Frame 11, Distance: 0.7497014054884877 
<b>Rank 3</b>	Frame 54, Distance: 0.413427 	Frame 71, Distance: 0.57076748785552 	Frame 10, Distance: 0.3609351815364013 

Figure 6. Retrieved frames for the Query Object 'Frames' from figure 2 using Global features

### 3.4. Why Min-Hash works

It can be shown that the expected value of Min-Hash similarity (number of matching components divided by total number of components in signature) equals the Jaccard similarity. Eg there are 2 sets,  $A, B$

$$A = \{1, 2, 3, 4\}, B = \{3, 4, 5, 6\}$$

If we take just 1 hash function and compute the min-hash signature for both the sets, then the probability that the min-hash value for both the sets is same is equal to the ratio of common components to the total components, which is the

ratio of intersection of the sets to its Union.

$$P(\text{Both setshavesamehashvalue}) = \frac{\text{Numberofcommoncomponents}}{\text{Totalnumberofdistinctcomponents}} = \frac{2}{6} = p = J(A, B)$$

Thus for a single hash function, the probability that the hash value will be same is the Jaccard's similarity. Then if we take  $N$  hash functions and find the expected value of the event that both the sets have same hash values,

$$\text{Numberofcomponentsthatareequal} = N * p$$

$$\frac{\text{Expectednumberofequalcomponents}}{\text{Totalcomponents}} = \frac{\text{Numberofequalcomponents}}{N} = \frac{N * p}{N} = p = J(A, B)$$

Thus the expected value of the min-hash similarity between two sets is equal to Jaccard similarity and we are evaluating the expected value as the approximation to Jaccard similarity.

## 4. Experiments and results

### 4.1. Experiments

Rigorous experiments were conducted on 2 videos, one is the video clip from Ground Hog day movie and another a generic video of logos from YouTube. Around 90 frames were extracted from GroundHog day video and 70 frames

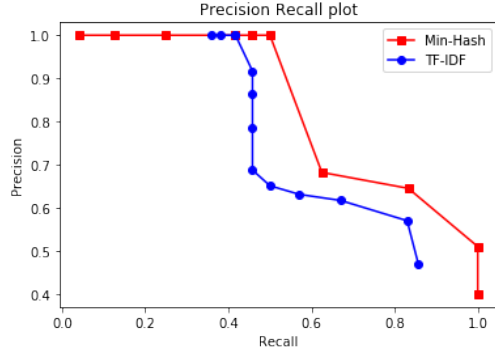


Figure 7. Recall precision plot

from Logo video. Both the videos were processed using both TF-IDF and Jaccard similarity. Also Global features like HOG, Color Histogram, Color histogram using spatial pyramids were used. However Global features didn't give satisfactory results. The major bottleneck in the experiment was performing K means clustering which was very time consuming operation. Thus experiments were carried out on fewer number of frames.

## 4.2. Results

Figure 5 shows the outputs from the Logo video. Total 4 Query objects were chosen to test the performance. Out of them 2 are displayed in this report, the Query Object 1 and Query Object 2. Top 4 retrieved frames with their confidence values for both the algorithms is shown. It can be seen that the actual frame has very high confidence and Incorrect frames have low values. Thus using an optimal threshold of 50% we can efficiently find the objects. Also 4 Query objects were selected from GroundHog day movie and frames having above threshold 50% were retrieved for both the algorithms. Also Global features were extracted for each frame to check if this method works well on Global features. Figure 6 shows top 3 retrieved frames for Query object which is in Figure 2, the 'frames' object. However, all the 3 Global features method perform poorly. Figure 7 shows the precision recall plot for both Min Hash and TF-IDF algorithms. Table at the end shows the Recall precision values for all methods.

## 5. Conclusion

From the confidence values, it can be seen that Jaccard's similarity has lesser number of False Positive's for each of the Query object. The area under the Precision-Recall plot gives the Precision value. Thus we can see that using Jaccard similarity we get a higher precision value than using TF-IDF values. Also from the Recall Precision values, Min Hash algorithm has higher precision than TF-IDF. And we can see that Global features don't work very well for this

problem, possibly because there is total loss of spatial information. Thus 2 methods were demonstrated to perform object search in Videos. Future work can involve searching for objects outside the movie. Also using Spatial information we can improve the ranking accuracy.

	TF-IDF	Min-Hash	Color Histogram	Spatial Pyramid
Precision	0.786	1	0.32	0.26
Recall	0.458	0.458	0.79	0.75

Table 1: Recall precision values

## References

- [1] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [2] J. Sivic and A. Zisserman. *Video Google: Efficient Visual Search of Videos*, pages 127–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [3] I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes : compressing and indexing documents and images / Ian H. Witten, Alistair Moffat, Timothy C. Bell*. Van Nostrand Reinhold ; Thomas Nelson Australia, New York : South Melbourne, Vic. :, 1994.
- [4] Z. Zhang. Phd thesis: Improving instance search performance in video collections. *SIGMultimedia Rec.*, 10(2):12:12–12:12, Aug. 2018.