Large-Scale Query-by-Image Video Retrieval Using Bloom Filters

André Araujo
Stanford University
afaraujo@stanford.edu

Jason Chaves
Stanford University
jchaves@stanford.edu

Haricharan Lakshman
Stanford University
hari.lakshman@stanford.edu

Roland Angst
Stanford University
rangst@stanford.edu

Bernd Girod
Stanford University
bgirod@stanford.edu

ABSTRACT

We consider the problem of using image queries to retrieve videos from a database. Our focus is on large-scale applications, where it is infeasible to index each database video frame independently. Our main contribution is a framework based on Bloom filters, which can be used to index long video segments, enabling efficient image-to-video comparisons. Using this framework, we investigate several retrieval architectures, by considering different types of aggregation and different functions to encode visual information – these play a crucial role in achieving high performance. Extensive experiments show that the proposed technique improves mean average precision by 24% on a public dataset, while being 4× faster, compared to the previous state-of-the-art.

Keywords

Bloom filters, large-scale, query-by-image, video retrieval

1. INTRODUCTION

This paper addresses the problem of searching large video databases using image queries – a technology which enables applications such as brand monitoring and content linking. The general task of linking images and videos has recently attracted much attention, for example for training event classifiers using web images [9, 10, 22] or localizing actions/tags in videos [3, 13, 31]. Our focus is a generic instance search problem, where no training data is available to learn classifiers for the query items. Traditionally, this query-by-image video retrieval task has been approached by constructing systems that index each frame [17, 21, 30] or each shot [4, 26, 39, 40, 41] in the database. In particular, shot-level indexing achieved remarkable success in recent editions of the TRECVID Instance Search challenge [24].

In contrast to most papers in this area [4, 17, 21, 26, 30, 39, 40, 41], we consider the large-scale setting of this problem, where indexing each frame or each shot entails prohibitive latency and memory costs. Our proposed retrieval architecture is presented in Fig. 1: we introduce a new stage (highlighted in orange), where the query is directly compared against long video segments (scenes). In subsequent stages of the pipeline, only a small number of scenes need to be considered for re-ranking, using shot- and frame-level information. This architecture enables much more scalable retrieval than traditional approaches – we demonstrate this experimentally, using three large-scale datasets. Similar to previous work [1, 28, 34, 38], we define a scene as a segment of video which contains interrelated shots and represents a semantic unit for a given type of content. We consider datasets where scenes correspond to video segments which are on average 2 to 8 *minutes* long (while shots are on average 3 to 8 seconds long).

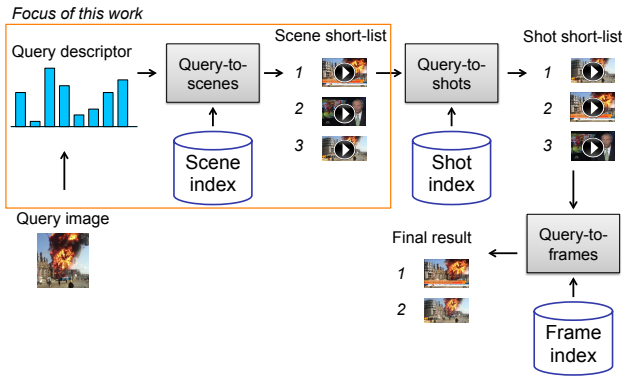

Figure 1: Block diagram of a large-scale query-by-image video retrieval system. First, a query image is represented as a descriptor which can be queried against the index of database scenes, to retrieve a short-list of scenes. Then, two refinement stages are conducted, to narrow down the matches to the shot and frame levels. In this work, our focus is on the stage where the query image is compared directly against scenes – this is the key to enable an efficient retrieval process.

While frame- and shot-level indexing received considerable attention in previous work, only recently have researchers started to investigate methods to compare images directly against scenes. In [1], Araujo et al. introduced scene-based descriptors which use Fisher vectors, enabling very compact databases and showing promising results. However, their work presents two important disadvantages: (i) their technique obtains limited retrieval accuracy, and (ii) their large-scale system employs linear search during query time, incurring substantial retrieval latency. Our work addresses these two aspects, to design a query-by-image video retrieval system which is scalable and achieves high retrieval accuracy.

Our main contributions are: (1) a new scene-based descriptor, which combines Fisher embedding and Bloom filters to enable up to 24% more accurate and 4× faster retrieval, compared to the previous state-of-the-art [1]; (2) a study of different feature representations and hashing schemes for Bloom filters, showing that point-indexed descriptors coupled with quantizer-based hashes provide the best performance among several tested configurations; (3) large-scale experiments using three datasets (among them two newly-

introduced datasets), comparing the proposed technique to the previous state-of-the-art, and to a frame-based baseline. Contrary to [1], these experiments use inverted index retrieval structures for all techniques under consideration, to obtain practical large-scale systems, with reduced latency.

2. BLOOM FILTERS FOR VIDEO RETRIEVAL BY IMAGE

We are interested in efficiently retrieving database scenes that contain visual information similar to a given query image. This problem is characterized by substantial asymmetry: while the query is a still image, database scenes are long segments of video containing diverse visual contents. In previous work, several hashing techniques addressed the symmetric problem of image retrieval using query images [27, 35]. To deal with the asymmetry of our problem, we model scenes as sets and images as items. We propose a generalization of hashing techniques, based on Bloom filters, to support efficient item-to-set comparisons. In this section, we briefly review the concept of Bloom filters, then introduce techniques that enable efficient large-scale retrieval.

2.1 Review of Bloom Filters

A Bloom filter (BF) [6] is a data structure designed for set membership queries, widely used in distributed databases and networking applications – for a review, see [7]. For a query item $q \in \mathcal{U}$ and a set of database items $\mathcal{S} \subset \mathcal{U}$, a BF is designed to respond to "is $q \in \mathcal{S}$?". If $q \in \mathcal{S}$, the answer is guaranteed to be correct (i.e., no false negatives); however, if $q \notin \mathcal{S}$, there is a small probability that the answer is incorrect (a false positive). This probabilistic response typically yields significant savings in memory – the total size of a BF can be much smaller than the combined size of all items it encodes. We consider two variants of BFs, described in the following.

Non-partitioned BF. In this case, the BF representation of $\mathcal{S}$ is a bit vector $\mathbf{b} \in \{0,1\}^{L_{np}}$, initialized to $\mathbf{b} = (0, 0, ..., 0)$. The number of bits that are used is $B_{np} = L_{np}$. Hash functions $h_1, h_2, ..., h_M$, with $h_m : \mathcal{U} \rightarrow \{1, 2, ..., L_{np}\} \forall m$, map an item to a single bit of $\mathbf{b}$. To insert a database item $x \in \mathcal{S}$ into the BF, we hash it M times and the bits $\mathbf{b}[h_1(x)]$, $\mathbf{b}[h_2(x)]$, ..., $\mathbf{b}[h_M(x)]$ are set to 1. This repeats for each database item, so more and more bits are set. Insertion of additional items is simple, but deletion is not possible. At query time, the BF responds that $q \in \mathcal{S}$ if $\mathbf{b}[h_1(q)] = \mathbf{b}[h_2(q)] = ... = \mathbf{b}[h_M(q)] = 1$, and $q \notin \mathcal{S}$ otherwise.

Partitioned BF. In this variant, the bit vector $\mathbf{b}$ is partitioned into M equal parts $\mathbf{b}_m$, each of length L_p. Each hash function h_m only produces bits in its respective partition $\mathbf{b}_m$. The total number of bits is $B_p = L_p \times M$. If $L_p = \frac{L_{np}}{M}$ (which leads to $B_p = B_{np}$), the false positive rate is asymptotically the same for partitioned and non-partitioned BFs.

Distance-sensitive BF. The BF introduced by [6] is designed to decide for the presence of an *exact* match in a database set. In general retrieval problems, the notion of approximate set membership queries might be more useful. Such queries are concerned with the question "is q near an item of $\mathcal{S}$?". For example, if we model a scene as a set and a frame as its item, a query image will unlikely be exactly the same as a frame, and a match may never occur. We want to find scenes that contain frames which are *similar* to the query image. Our application is thus more suitable to distance-sensitive Bloom filters (DSBF) [16], which address this problem, illustrated in Fig. 2. DSBFs are similar to standard BFs, but they are coupled to locality-sensitive hashes (LSH) – since in this case the hashes must map similar items to the same hash bucket with high probability.

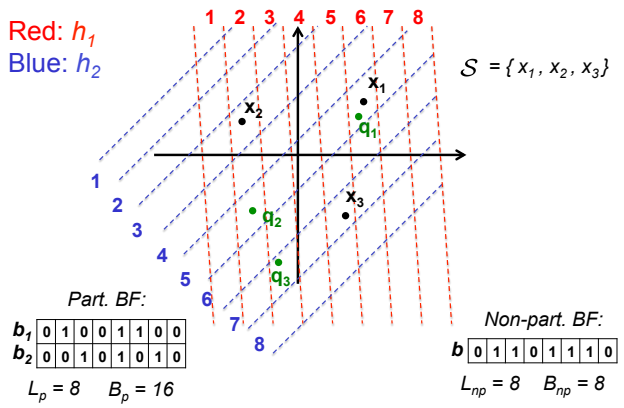

Figure 2: Illustration of a Bloom filter encoding set $\mathcal{S} = \{x_1, x_2, x_3\}$ in 2D. Two hash functions are shown ($M = 2$), in red and in blue, with bin numbers marked near the corresponding regions. Partitioned (left) and non-partitioned (right) BFs are presented. Examples of queries are shown in green. Consider that the BF should indicate $q \in \mathcal{S}$ if the query is close to a database item. Both partitioned and non-partitioned BFs indicate $q_1 \in \mathcal{S}$ (True Positive) and $q_2 \in \mathcal{S}$ (False Positive). For q_3, the non-partitioned BF indicates $q_3 \in \mathcal{S}$ (False Positive), and the partitioned BF indicates $q_3 \notin \mathcal{S}$ (True Negative).

BFs and hashing. As discussed in [7], BFs can be seen as a generalization of hashing, supporting more effective trade-offs between number of bits and false positive rate. In general, BFs obtain lower memory requirements than simple hashing schemes, given a target false positive rate.

2.2 BF-GD: Using Global Descriptors

First, we apply the BF framework to our problem in a straightforward way: query images are directly modeled as items, and database scenes as sets of video frames. For each scene, the constituent frames are hashed into a BF. A query image can then be matched against the BF of each scene.

To represent query images and video frames, we use global image descriptors – this method is denoted BF-GD. We use Fisher vectors (FV) [15], a common technique in multimedia retrieval, whose variants obtain high performance in many datasets [29, 33]. Note that this system discards the ordering of frames: the representation for a given scene is the same regardless of the ordering of its constituent frames. This is akin to the use of bag-of-words in image retrieval, where the representation is the same regardless of where local features appear in an image.

2.3 BF-PI: Using Point-Indexed Descriptors

In this section, we consider a different configuration of the BF framework. The motivation arises from noticing the two levels of aggregation at play when using BF-GD: local descriptors are first aggregated into FVs per frame, then FVs are aggregated per scene. It is not clear the impact of these two stages to the discriminativeness of the final scene descriptor. This leads us to remove one aggregation step, and directly encode Fisher-embedded local features into BFs (they are not aggregated per frame before hashing, as in BF-GD). Tao et al. [32] showed how a FV can be decomposed into the Fisher embedding of each local feature, leading to a point-indexed representation: instead of storing a FV, the

database stores an embedded version of each local feature.

The proposed technique is called BF-PI. Consider a local feature $\mathbf{x}$ and a FV with parameters $\{w_k, \mu_k, \sigma_k, k = 1 \dots K\}$ denoting the mixture probabilities, mean vectors and diagonal covariance matrices. As in [32], we employ the point-indexed representation of $\mathbf{x}$ using only the Gaussian from the FV which obtains the strongest soft-assignment probability. The point-indexed representation for $\mathbf{x}$ is a triplet:

$$\{r; \frac{\gamma_{\mathbf{x}}(r)}{\sqrt{w_r}}; \mathbf{d_x} = \sigma_r^{-1}(\mathbf{x} - \mu_r)\} \quad (1)$$

where r is the index of the Gaussian with strongest soft-assignment probability for $\mathbf{x}$, $\gamma_{\mathbf{x}}(r)$ is the value of that soft-assignment probability, and $\mathbf{d_x}$ is the scaled residual vector between $\mathbf{x}$ and the r-th Gaussian. With $\mathbf{x}$ represented in this manner, the bucket $h_r(\mathbf{d_x})$ in the BF is set to 1.

2.4 Hash Functions & Scoring

LSH families. We consider three LSH families. The standard metric for comparing FVs is cosine similarity, so a natural choice for this problem is the LSH family for cosine distance [8], which uses random hyperplanes – referred to as LSH-C. A second family of functions, denoted LSH-S, is a special case of LSH-C, where the components of random hyperplanes are either +1 or −1, picked at random. This family has been widely used in information retrieval [12, 18]. We also consider the LSH family for Hamming distance, denoted LSH-B. This function samples a bit from a binarized signature, and can be generalized to real-valued vectors by using random axis-aligned hyperplanes. In practice, we want to map each item to L buckets. To accomplish that, each of the M hash functions we use is composed of n hyperplanes, thus mapping each item to one out of $2^n = L$ buckets.

Domain of hash functions. The natural choice for the domain of hash functions is the original space where items lie. We denote hash functions of this type as vector-based hashes (VBH). For a FV with K Gaussians, and local descriptors having d dimensions, FVs lie in $\mathcal{R}^{K \times d}$. Thus, in the BF-GD case, $h_{VBH} : \mathcal{R}^{K \times d} \rightarrow 2^n$. Another possibility is to divide FVs into chunks corresponding to their Gaussians, and hash each chunk separately. We denote hash functions of this type as Gaussian-based hashes (GBH), $h_{GBH} : \mathcal{R}^d \rightarrow 2^n$. In the BF-PI version, we are interested in hashing d-dimensional point-indexed descriptors into 2^n buckets. Thus, GBH is also applicable to this version.

Quantizer-based hashing. Recent work shows that quantization outperforms random hashes for approximate nearest neighbor tasks [5, 25]. With that motivation, we employ K-means to construct a vector quantizer (VQ), and use it as a hash function: an item is inserted into the bucket corresponding to the centroid it is closest to.

Scoring. At query time, the query image is processed in the same way video frames are processed at indexing time. To score scenes, we explore two techniques. We restrict the presentation to the case of BF-GD, using a non-partitioned BF (scoring for other configurations is similar). First, we consider scoring based on the number of hash matches ($S^{\#}$). Given the query image descriptor $\mathbf{q}$ and the m-th hash function, the score $S_v^{\#}$ of database scene v is updated as:

$$S_v^{\#} := S_v^{\#} + \mathbf{b}_v[h_m(\mathbf{q})] \quad (2)$$

In other words, the score of scene v is incremented if its $h_m(\mathbf{q})$-th bucket is set. Another option is to use TF-IDF, as is common in information retrieval: for the same case as above, the score S_v^T of clip v can be computed as:

$$S_v^T := S_v^T + \mathbf{b}_v[h_m(\mathbf{q})] \cdot \frac{w_{h_m(\mathbf{q})}^2}{(\sum_l \mathbf{b}_v[l] w_l^2)^{\alpha}} \quad (3)$$

where w_l corresponds to the IDF weight of bucket l and $(\sum_l \mathbf{b}_v[l] w_l^2)^{\alpha}$ denotes a normalization factor, where α is empirically chosen ($\alpha = 0.5$ corresponds to L_2 normalization).

3. EXPERIMENTS

Datasets. Our experiments use 3 datasets (2 of them introduced in this work). The existing Stanford I2V (SI2V) dataset is the largest for this research problem [2]. It contains news videos, and query images are collected from the web. The new Video Bookmarking (VB) dataset uses the same videos as SI2V, but the queries contain displays with a frame of a video being played. This models the case where a user wants to retrieve the video being played, e.g., to resume playback in a different device. The third dataset, also introduced in this work, contains lecture videos (LV), with queries being clean images of slides. In all cases, we extract 1 frame per second. Extensive experiments are conducted on reduced dataset versions: SI2V-600k, VB-600k and LV-600k (each containing 600k frames and 160 hours of video). Large-scale experiments use SI2V-4M, VB-4M (4M frames and 1,079 hours of video) and LV-1.5M (1.5M frames and 408 hours of video). More than 200 queries are used per dataset. To train auxiliary structures (e.g., GMM, PCA), we use independent datasets. All datasets are presented in greater detail in Appendices A and B.

Local and global descriptors. For local descriptor extraction, we detect Hessian-affine keypoints [23] and describe them using SIFT [19]. Using PCA, the dimensionality of SIFT descriptors is reduced to $d = 32$. For FVs [15] and point-indexed FVs [32], we use $K = 512$ Gaussians.

BF parameters. We set $M = K = 512$, which is experimentally shown as a good choice. We vary n, the number of bits obtained per hash function. For a given n, an item can be mapped into 2^n buckets in the BF. For TF-IDF scoring, we experiment with $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ in the 600k datasets. We tune n and α in the smaller-scale experiments, and use their optimal values for large-scale experiments.

Performance assessment. We follow the evaluation procedure from previous work [1, 2], to obtain comparable numbers: results are evaluated using mean Average Precision (mAP). For configurations which use LSH functions, we run the experiments 10 times, and report the averaged mAP.

Results: BF-GD. Fig. 3 presents BF-GD results in the SI2V-600k dataset. First, note that GBH outperforms VBH significantly: this can be understood since FVs aggregate different types of visual information per Gaussian, and the correlation between different Gaussians might be weak. Fig. 3 also compares partitioned (P) and non-partitioned (NP) BFs. For a fair comparison, we should have $B_p = B_{np}$: P-BF using $M = 512 = 2^9$ bit vectors of length 2^n should be compared to NP-BF using a bit vector of length 2^{n+9}. In this case, P-BF outperforms NP-BF. Finally, Fig. 3 shows that LSH-B outperforms LSH-C and LSH-S. This might be surprising, as LSH-B is much simpler than LSH-C and LSH-S. However, these results agree with previous work [27] which shows that this simple technique outperforms spectral hashing [37] and LSH-C for image retrieval. Overall, though, BF-GD obtains limited mAP, showing that a straightforward BF aggregation method may not be the best choice for this problem.

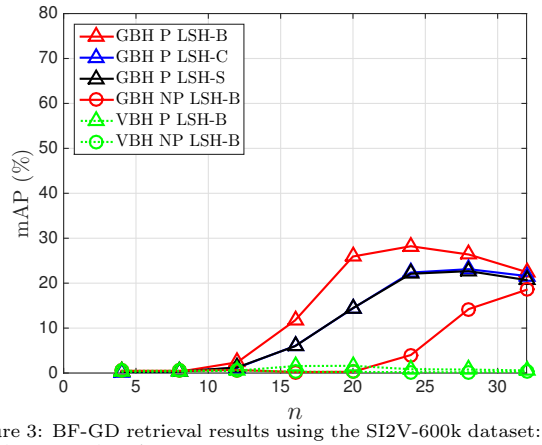

Figure 3: BF-GD retrieval results using the SI2V-600k dataset: mAP as a function of n. All curves use scoring based on the number of hash matches. Comparison of partitioned (P) versus non-partitioned (NP) BFs; GBH versus VBH hashes; LSH-B versus LSH-C and LSH-S.

	SI2V-600k	VB-600k	LV-600k
Scene FV* (DoG) [1]	47.33	-	-
Scene FV*	50.01	62.17	**66.71**
BF-GD LSH-B	35.73	38.53	36.02
BF-PI LSH-B	70.46	65.76	45.05
BF-PI VQ	**73.75**	**67.67**	63.66

Table 1: Summary of retrieval results (mAP in %) for the 600k datasets. All techniques use Hessian-affine keypoints, except for [1], which uses difference-of-Gaussian (DoG) keypoints. The BF techniques presented here use GBH hashes, partitioned BFs and TF-IDF.

Results: BF-PI. Fig. 4 compares the different hashing and scoring techniques, when using BF-PI. BF-PI provides a substantial improvement in mAP, compared to BF-GD, of more than 30%. This demonstrates the benefit of removing the aggregation per frame before hashing. This figure shows that LSH-B outperforms LSH-C and LSH-S also when BF-PI is used. Fig. 4 further introduces results using the TF-IDF scoring method and VQ hashes. In this case, we use $n \leq 16$ to limit memory and computational complexity. VQ-based hashing improves retrieval performance compared to the scheme that makes use of LSH-B. BF-PI using $n = 16$, coupled with VQ-based hashing and TF-IDF scoring, outperforms all other BF configurations we experimented with.

Comparison to state-of-the-art. Tab. 1 presents summarized results for experiments on the 600k datasets. We compare the proposed methods against the state-of-the-art in scene-based signatures: Scene FV* [1] (binarized FVs with 2048 Gaussians), which used difference-of-Gaussian keypoints, on the SI2V dataset (we use their public code to reproduce these results). We also implement Scene FV* using Hessian-affine keypoints (which boosts performance of [1]). The proposed BF-PI scheme, coupled with VQ hashing, outperforms other approaches significantly for SI2V-600k, by 26% mAP compared to [1]. In the VB-600k dataset, it also outperforms other approaches, but with a smaller margin: 5.50% better than Scene FV*. In the LV-600k dataset, BF-PI VQ is slightly worse than Scene FV*, by 3.05%.

Large-scale experiments compare our best method to the state-of-the-art [1] (using Hessian-affine keypoints, since it improves retrieval performance), and to a technique which indexes every frame in the database using binarized FVs with 512 Gaussians, denoted Frame FV*. Results are presented in Tab. 2, including latency and memory figures. Contrary to previous work [1], we use inverted index retrieval structures for all techniques, to obtain practical large-scale systems. For

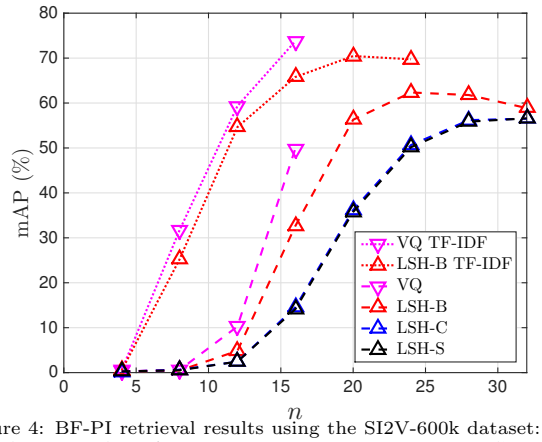

Figure 4: BF-PI retrieval results using the SI2V-600k dataset: mAP as a function of n. All curves use partitioned BFs and GBH. Comparison of VQ versus LSH hashes, and different scoring techniques.

	mAP (%)	Latency (secs)	Memory (GB)
SI2V-4M dataset			
Frame FV*	72.44	0.4118	20.59
Scene FV* [1]	49.71	0.1643	**3.01**
BF-PI VQ (ours)	**74.08**	**0.0431**	10.76
VB-4M dataset			
Frame FV*	75.97	0.4423	20.59
Scene FV* [1]	67.37	0.2106	**3.01**
BF-PI VQ (ours)	**76.25**	**0.1101**	10.76
LV-1.5M dataset			
Frame FV*	64.21	0.1984	7.67
Scene FV* [1]	64.47	0.0365	**0.42**
BF-PI VQ (ours)	**67.60**	**0.0357**	1.20

Table 2: Summarized results for large-scale experiments, comparing the proposed BF-PI technique against the previous state-of-the-art [1] and against a reference frame-based technique. All methods use inverted index structures and Hessian-affine keypoints. Retrieval latency results are per query, using one core on an Intel Xeon 2.4GHz.

Scene FV* and Frame FV*, we use the Multi-Block Indexing Table (MBIT) [11, 36] method, which was introduced to provide speedup over simple linear search for binarized FVs (we used MBIT's source code, which is available online). For BF-PI, we can represent it in an inverted index format, which is straightforward. For BF-PI and Scene FV*, we re-rank the top scene results using shot-based FV*s, as in [1]. It can be seen that BF-PI is much faster (up to 4×) and much more accurate (up to 24%) than Scene FV*, in all datasets. BF-PI also outperforms Frame FV*: up to 10× faster retrieval with slightly improved mAP. This demonstrates the improved scalability of our technique, compared to the common approach of frame-based indexing, which has been widely used for query-by-image video retrieval.

4. CONCLUSION

In this work, we explore aggregation of visual information from scenes into Bloom filters, enabling efficient and effective video retrieval using image queries. First, we show that a straightforward application of Bloom filters to our problem, using global image descriptors, obtains limited retrieval accuracy. Our best-performing scheme adapts the Bloom filter framework: the key is to hash discriminative local descriptors into scene-based signatures. The techniques are evaluated by considering different hash functions and score computation methods. Large-scale experiments show that our system achieves high retrieval accuracy and reduced query latency in three datasets, outperforming the previous state-of-the-art.

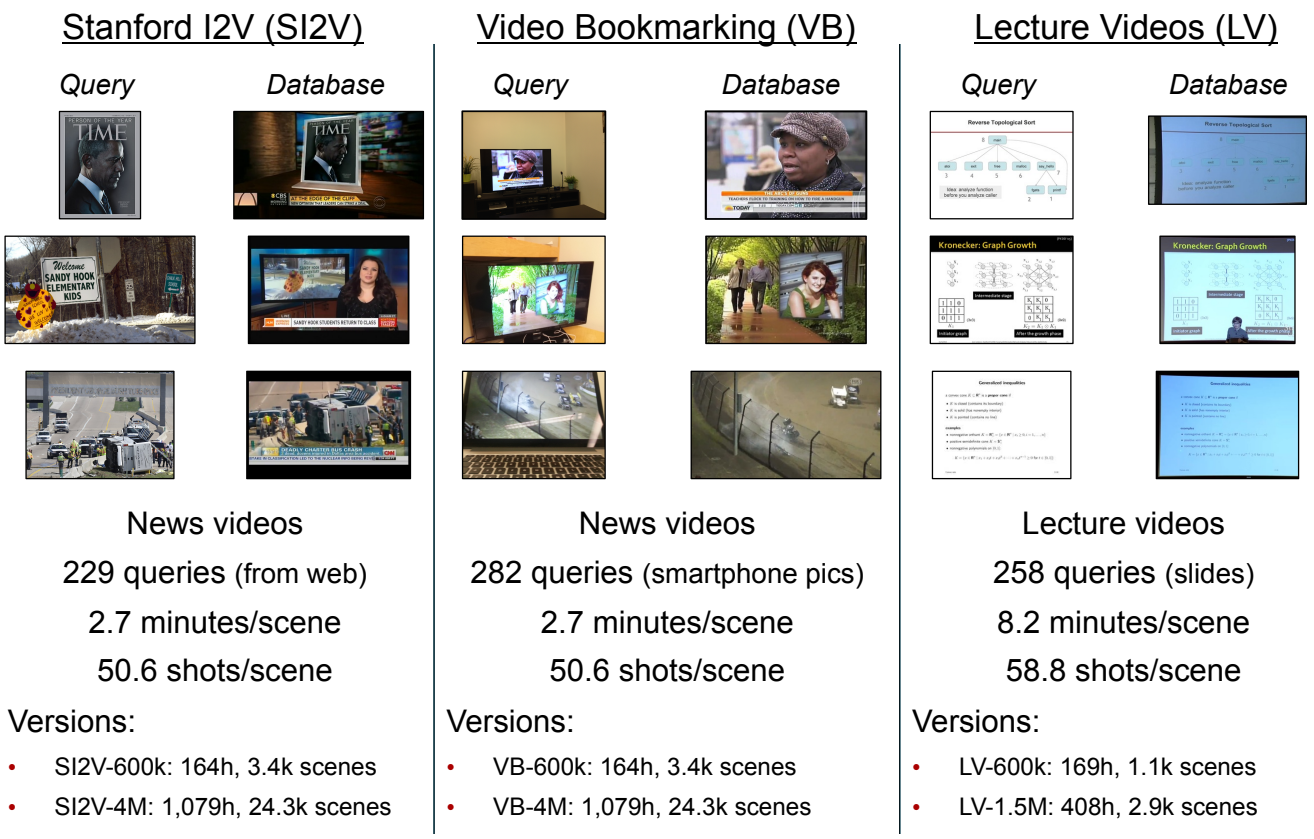

Figure 5: Illustration of the query-by-image video retrieval datasets used in this work. Left: Stanford I2V (SI2V) dataset. Center: Video Bookmarking (VB) dataset. Right: Lecture Videos (LV) dataset.

Appendix A: Datasets for Query-by-Image Video Retrieval

This section presents the large-scale query-by-image video retrieval datasets used in this paper. They are illustrated in Fig. 5. Two new datasets are introduced: Video Bookmarking and Lecture Videos. They will be released to the community, to encourage comparative experiments.

Stanford I2V (SI2V)

This dataset was introduced in [2], and is publicly available since 2015, at https://purl.stanford.edu/zx935qw7203. This is the largest available dataset for query-by-image video retrieval experiments. The video clips correspond to scenes: in this case, they are news stories from 39 recurring newscasts, and the query set is composed of 229 images collected from the web, with annotated ground-truth. For example, a video clip could start with an anchor in a studio, then it would transition to a reporter in the field, then original footage of an event would be presented, then the story would be wrapped up by the anchor in the studio. The clips are on average 2.7 minutes long and contain on average 50.6 different shots. In this work, we experiment with two versions of this dataset. In both versions, the full query set is used. The dataset versions are: (i) SI2V-600k: a version of the dataset containing 3,401 clips, with 164 hours of video and 589,965 frames extracted at 1 frame per second (fps); (ii) SI2V-4M: a large-scale version of the dataset containing 24,317 clips, with 1,079 hours of video and 3,972,602 frames extracted at 1fps.

Video Bookmarking (VB)

This new dataset makes use of the same database videos as the SI2V dataset, but it introduces a different query set. The application modeled by this dataset corresponds to the case where a user is interested in retrieving the video being played, e.g., to resume playback in a different device. The queries contain displays (from TVs or computers) with a frame of a video being played. Smartphone and tablet cameras are used to generate these queries, and the number of queries in this dataset is 282. The VB-600k and VB-4M versions of this dataset are based on the same database videos as the SI2V dataset. In both versions, the full query set is used.

Lecture Videos (LV)

This new dataset contains lecture video segments, which have been collected from 21 engineering-related university courses, in 2013 and 2014. Each database video clip corresponds to a lecture segment that introduces a single topic (i.e., each clip corresponds to a scene for this type of content). For example, a video clip would present the notion of locality-sensitive hashing, in the context of a lecture that presents different techniques to mine large datasets of documents. The clips are on average 8.2 minutes long and contain on average 58.8 different shots. The queries are 258 clean images of slides which are shown during a particular lecture. In this work, we experiment with two versions of this dataset. In both versions, the full query set is used. The dataset versions are: (i) LV-600k: a version of the dataset containing 1,166 clips, with 169 hours of video and 607,747 frames extracted at 1fps;

(ii) LV-1.5M: a large-scale version of the dataset containing 2,985 clips, with 408 hours of video and 1,478,978 frames extracted at 1fps.

Discussion

These three datasets present very different characteristics. SI2V and VB present much larger variety of content than LV – the average number of shots per clip is similar, but the average clip duration is much longer for LV. Regarding queries, in the LV dataset the matching object covers the entire query image, while this is not usually the case for the SI2V and VB datasets. In SI2V, the database ground-truth matching information usually occupies only a fraction of the total frame, while in the other two datasets usually it covers the entire frame.

Appendix B: Training Datasets

This section presents the independent datasets which were used for training auxiliary retrieval structures in this paper. Two new datasets are introduced: LV-Training and Slideshare-1M. They will be released to the community, to encourage comparative experiments.

Flickr60k

This standard dataset (introduced by Jégou et al. [14]) contains 67,714 images, provided together with 140 million SIFT [19] descriptors, extracted from keypoints detected using the Hessian-Affine keypoint detector [23]. These data were used to train Gaussian mixture models and principal component analysis for the SI2V and VB datasets.

LV-Training

This new dataset has been used to train Gaussian mixture models and principal component analysis used for retrieval with the LV dataset. The data is composed of slides for 12 engineering-related courses, different from the ones used in the LV dataset. A total of $8,619$ slides compose this training dataset.

MIRFLICKR-1M

This dataset contains 1 million images downloaded from Flickr under the creative commons license [20]. It has been used to train quantizers for the Bloom filter technique, when performing retrieval on the SI2V and VB datasets.

Slideshare-1M

This new dataset was used for training quantizers for the Bloom filter technique, when performing retrieval on the LV dataset. 1 million slides were collected from Slideshare using the available API, with tags related to engineering and science, as the LV dataset (which is the target dataset in this case) contains slides mostly related to these topics.

5. REFERENCES

[1] A. Araujo, J. Chaves, R. Angst, and B. Girod. Temporal Aggregation for Large-Scale Query-by-Image Video Retrieval. In *Proc. ICIP*, 2015.

[2] A. Araujo, J. Chaves, D. Chen, R. Angst, and B. Girod. Stanford I2V: A News Video Dataset for Query-by-Image Experiments. In *Proc. ACM MMSys*, 2015.

[3] L. Ballan, M. Bertini, G. Serra, and A. Del Bimbo. A Data-Driven Approach for Tag Refinement and Localization in Web Videos. *Computer Vision and Image Understanding*, 140:58–67, 2015.

[4] N. Ballas, M. Redi, A. Hamadi, B. Gao, B. Labbe, B. Merialdo, C. Zhu, L. Chen, B. Safadi, N. Derbas, Y. Tang, A. Benoit, H. L. Borgne, B. Mansencal, E. Dellandrea, J. Benois-Pineau, P. Lambert, P. Gosselin, M. Budnik, T. Strat, D. Picard, S. Ayache, G. Quenot, and C. Bichot. IRIM at TRECVID 2014: Semantic Indexing and Instance Search. In *Proc. TRECVID*, 2014.

[5] R. Balu, T. Furon, and H. Jégou. Beyond Project and Sign for Cosine Estimation with Binary Codes. In *Proc. ICASSP*, 2014.

[6] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13, 1970.

[7] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 2004.

[8] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proc. ACM Symposium on Theory of Computing*, 2002.

[9] J. Chen, Y. Cui, G. Ye, D. Liu, and S.-F. Chang. Event-Driven Semantic Concept Discovery by Exploiting Weakly Tagged Internet Images. In *Proc. ICMR*, 2014.

[10] L. Duan, D. Xu, and S.-F. Chang. Exploiting Web Images for Event Recognition in Consumer Videos: A Multiple Source Domain Adaptation Approach. In *Proc. CVPR*, 2012.

[11] L.-Y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao. Overview of the MPEG-CDVS Standard. *IEEE Transactions on Image Processing*, 25(1), 2016.

[12] M. Henzinger. Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms. In *Proc. ACM SIGIR*. ACM, 2006.

[13] M. Jain, J. C. van Gemert, T. Mensink, and C. G. Snoek. Objects2action: Classifying and Localizing Actions Without any Video Example. In *Proc. ICCV*, 2015.

[14] H. Jégou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proc. ECCV*. Springer, 2008.

[15] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Pérez, and C. Schmid. Aggregating Local Image Descriptors into Compact Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9), 2012.

[16] A. Kirsch and M. Mitzenmacher. Distance-Sensitive Bloom Filters. In *Proc. Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2006.

[17] D.-D. Le, C.-Z. Zhu, S. Poullot, V. Q. Lam, D. A. Duong, and S. Satoh. National Institute of Informatics, Japan at TRECVID 2011. In *Proc. TRECVID*, 2011.

[18] J. Leskovec, A. Rajaraman, and J. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[19] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[20] B. T. M. Huiskes and M. Lew. New Trends and Ideas in Visual Concept Detection: The MIR Flickr Retrieval Evaluation Initiative. In *Proc. ICMR*, 2010.

[21] J. Meng, J. Yuan, J. Yang, G. Wang, and Y.-P. Tan. Object Instance Search in Videos via Spatio-Temporal Trajectory Discovery. *IEEE Transactions on Multimedia*, 18(1), 2016.

[22] M. Merler, B. Huang, L. Xie, G. Hua, and A. Natsev. Semantic Model Vectors for Complex Video Event Recognition. *IEEE Transactions on Multimedia*, 14(1), 2012.

[23] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1), 2004.

[24] P. Over, G. Awad, J. Fiscus, M. Michel, D. Joy, A. Smeaton, W. Kraaij, G. Quenot, and R. Ordelman. TRECVID 2015 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *Proc. TRECVID*, 2015.

[25] L. Paulevé, H. Jégou, and L. Amsaleg. Locality Sensitive Hashing: A Comparison of Hash Function Types and

Querying Mechanisms. *Pattern Recognition Letters*, 31(11), 2010.

[26] Y. Peng, J. Zhang, X. Huang, M. Sun, X. He, P. Tang, Y. Zhao, J. Zhao, J. Qi, and J. Zhang. PKU-ICST at TRECVID 2015: Instance Search Task. In *Proc. TRECVID*, 2015.

[27] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-Scale Image Retrieval with Compressed Fisher vectors. In *Proc. CVPR*, 2010.

[28] Z. Rasheed and M. Shah. Scene Detection In Hollywood Movies and TV Shows. In *Proc. CVPR*, 2003.

[29] M. Shi, Y. Avrithis, and H. Jégou. Early Burst Detection for Memory-Efficient Image Retrieval. In *Proc. CVPR*, 2015.

[30] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. ICCV*, 2003.

[31] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal Localization of Fine-Grained Actions in Videos by Domain Transfer from Web Images. In *Proc. ACM MM*, 2015.

[32] R. Tao, E. Gavves, C. G. M. Snoek, and A. W. M. Smeulders. Locality in Generic Instance Search from One Example. In *Proc. CVPR*, 2014.

[33] G. Tolias, Y. Avrithis, and H. Jégou. Image Search with Selective Match Kernels: Aggregation Across Single and Multiple Images. *International Journal of Computer Vision*, 2015.

[34] B. Truong, S. Venkatesh, and C. Dorai. Scene Extraction in Motion Pictures. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1), 2003.

[35] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to Hash for Indexing Big Data – A Survey. *Proceedings of the IEEE*, 104(1), 2016.

[36] Z. Wang, L.-Y. Duan, J. Lin, T. Huang, W. Gao, and M. Bober. Component Hashing of Variable-Length Binary Aggregated Descriptors for Fast Image Search. In *Proc. ICIP*, 2014.

[37] Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *Proc. NIPS*, 2008.

[38] M. Yeung, B.-L. Yeo, and B. Liu. Segmentation of Video by Clustering and Graph Analysis. *Computer Vision and Image Understanding*, 71(1), 1998.

[39] C.-Z. Zhu, Y.-H. Huang, and S. Satoh. Multi-Image Aggregation for Better Visual Object Retrieval. In *Proc. ICASSP*, 2014.

[40] C.-Z. Zhu, H. Jegou, and S. Satoh. Query-Adaptive Asymmetrical Dissimilarities for Visual Object Retrieval. In *Proc. ICCV*, 2013.

[41] C.-Z. Zhu and S. Satoh. Large Vocabulary Quantization for Searching Instances from Videos. In *Proc. ICMR*, 2012.