Large-Scale Video Retrieval Using Image Queries

André Araujo and Bernd Girod, *Fellow, IEEE*

***Abstract*—Retrieval of videos from large repositories using image queries is important for many applications, such as brand monitoring or content linking. We introduce a new retrieval architecture, where the image query can be compared directly to database videos – significantly improving retrieval scalability, compared to a baseline system that searches the database on a video frame level.**

Matching an image to a video is an inherently asymmetric problem. We propose an asymmetric comparison technique for Fisher vectors and systematically explore query or database items with varying amounts of clutter, showing the benefits of the proposed technique. We then propose novel video descriptors that can be compared directly to image descriptors. We start by constructing Fisher vectors for video segments, by exploring different aggregation techniques. For a database of lecture videos, such methods obtain two orders of magnitude compression gain with respect to a frame-based scheme, with no loss in retrieval accuracy. Then, we consider the design of video descriptors which combine Fisher embedding with hashing techniques, in a flexible framework based on Bloom filters. Large-scale experiments using three datasets show that this technique enables faster and more memory-efficient retrieval, compared to a frame-based method, with similar accuracy. The proposed techniques are further compared against pre-trained convolutional neural network features, outperforming them on three datasets by a substantial margin.

***Index Terms*—Bloom filter, Fisher vector, large-scale, query-by-image, video retrieval.**

I. Introduction

VISUAL search applications have gained substantial popularity recently. In its most common form, this technology enables image-based querying against a database of images. This is typically used to retrieve information associated with specific objects from large databases, by comparing an image of an object (the query image) against a database of reference images. This technology has been widely used for recognition of products [1], [2] and locations [3], [4] – and it has also found its way to commercial applications [5], [6].

This work addresses a variant of the visual search problem, where the query is an image, and the database is composed of videos – such technology is relevant for numerous applications. For example, for brand monitoring, a company might want to find all appearances of specific logos or products in television broadcasts. In another application, users might snap a picture of a display to obtain information about the video that is being watched. In online education, a user might want to find a segment of a lecture video by using a specific slide as a query.

A naïve solution to this problem would involve indexing each database video frame independently – essentially treating the database of videos as a database of images, where the images would correspond to video frames. While such a simple solution can potentially obtain high retrieval accuracy, storing and comparing frame-based descriptors directly would entail prohibitively large data and complexity requirements in a large-scale setting. In contrast, our work introduces a new retrieval architecture, where in a first stage the query image can be directly compared to database video clips – significantly improving the scalability of the retrieval process.

A. Araujo is with Google Inc., Mountain View, CA 94043 USA (e-mail: andrearaujo@google.com).

B. Girod is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: bgirod@stanford.edu).

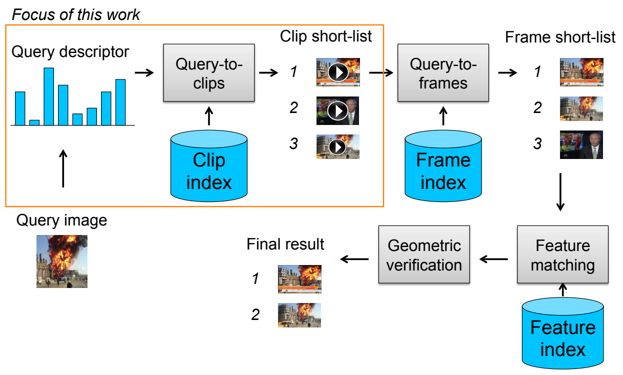


Figure 1: Block diagram of a large-scale system which searches video databases using image queries. First, a query image is represented as a descriptor, which can be queried against the index of database video clips – to retrieve a short-list of video clips. Then, two re-ranking stages narrow down the matches to the frame level and the local feature level. In this work, our focus is on the stage where the query image is compared directly against video clips in the database – this is the key to enable an efficient retrieval process. This stage corresponds to the part of the system which is highlighted in orange.

Fig. 1 presents the block diagram of a large-scale query-by-image video retrieval system. It is very important to quickly narrow down the search to a small set of video clips – which will be re-ranked in subsequent stages. As depicted in Fig. 1, the query image's descriptor is initially compared to an index that contains information on a video clip level. Then, the most promising video clips are inspected at a frame level, generating a ranking of video frames. Finally, the short-listed frames are compared to the query image in terms of local information, and a geometric verification step ensures that the visual information of the query is geometrically consistent with that of the retrieved database videos. The focus of this work lies on the first retrieval stage (highlighted in orange): the objective is to retrieve the most relevant video clips from a large database, using a query image.

Image-based video retrieval presents two main challenges: (i) Asymmetry: database videos comprise a temporal component, while query images do not – how can a retrieval system take this into account to design effective ways of comparing images to videos? (ii) Temporal aggregation: how can we combine information over seconds or minutes of video and obtain compact signatures that can be directly compared against images? These two challenges, which are clearly interrelated, are addressed by the main contributions of this work, described in the following.

Contributions.

- *Asymmetric comparison techniques for Fisher vectors*: We develop methods which test if an image is contained in another image (or contained in a video), for retrieval systems that use Fisher vectors. Existing Fisher vector comparison techniques are not optimized for testing if one image is contained in another, so we introduce a method that addresses this important problem configuration. We consider two different asymmetric problems which arise in practice, where the query item might be contained in one database item, or vice-versa. Experiments show substantial performance gain by using the proposed techniques.
- *Temporal aggregation using Fisher vectors*: We develop compact video representations using Fisher vectors. Temporal aggregation is investigated for long video segments, effectively removing temporal redundancy and enabling large-scale retrieval. To the best of our knowledge, this is the first work that addresses aggregation of visual information over long video segments in order to match against images.
- *Temporal aggregation using Bloom filters*: We develop video representations which use Bloom filters to aggregate visual information. This framework enables experimentation with different aggregation configurations, where visual information might be first aggregated per frame then per video, or simply directly aggregated per video. This technique achieves similar retrieval quality as a baseline technique that indexes every video frame in the database, while being much faster and more memory-efficient.

Initial results of our work have been presented in [7], [8]. In this paper, we study in much more depth the asymmetric Fisher vector (FV) comparison technique initially introduced in [7]. In particular, we introduce: (1) a new asymmetric comparison technique (for the case where database may be included in query), (2) the use of power normalization, with a new interpretation, (3) extensive experiments to analyze the effect of asymmetric comparison techniques with varying degrees of asymmetry, for both binarized and non-binarized FVs, (4) a new dataset to perform such experiments. We also provide much more in-depth experimental results for the FV-based temporal aggregation technique presented in [7], including: (1) experiments for three different datasets, making use of the asymmetric comparison techniques introduced in this work, and (2) a comparison against recent CNN-based descriptors. Compared to [8], this paper presents more comprehensive large-scale experiments using Bloom filters, including detailed results for different retrieval metrics as a function of the dataset size.

II. Related Work

Visual search is the problem of indexing and querying visual data, and we categorize its variants depending on the type of query and database information. Most work in visual search concerns the image-to-image (I2I) problem, where an image is queried against a database of images [9], [10], [11], [12]. The video-to-image (V2I) problem, relevant for augmented reality, refers to searching a database of images using query videos [13], [14]. Another variant is the case where the query is a video and the database is composed of videos (V2V) – widely used for content-based copy detection [15] and event retrieval [16]. This work focuses on the image-to-video (I2V) problem: a query image is used to find relevant database videos. We review related I2V work in the rest of this section.

Early work in the I2V problem simply applied I2I techniques for video search. In this case, the video database is simply treated as an image database of video frames. Sivic and Zisserman [17] introduced the bag-of-words (BoW) model by using it to index a database of movies. In their work, each video frame is indexed independently. Later, Sivic et al. [18] used the temporal consistency of the video database to find different views of the same object. This system enabled "object-level matching", i.e., when a user issues a query image presenting a specific view, video segments with all different object views might be retrieved.

The I2V research problem received attention with the introduction of the TRECVID challenge task "Instance Search" (INS), in 2010 [19]. In this task, given a query image set, systems are expected to find all occurrences of the query in a video database. The queries might represent a person, an object or a location, and the query set comprises up to four images per query. The query images are composed of regions-of-interest in frames from the same dataset. Early high-performing systems, by Le et al. [20], used color SIFT-based BoW with vocabulary trees. This system indexed videos by considering each frame independently. In follow-up work, Zhu and Satoh [21] showed that the performance of the top INS 2011 system is mainly due to the matching of the background between query and database (instead of matching the query object). This shows a limitation of the TRECVID dataset, whose query images are collected from videos very similar to the videos in the test dataset. The datasets introduced in our previous papers [22], [23] addressed these issues.

Zhu and Satoh [21] introduced the aggregation of SIFT descriptors into a single BoW for each shot. More recently, Ballas et al. [24] and Zhu et al. [25] reported improved retrieval performance when aggregating frame-based features per shot. Zhu et al. [25] specifically evaluated different shot aggregation methods. They concluded that the method which simply extracts a BoW global descriptor from all SIFT features from keyframes in the shot (average pooling) performs best. While previous work considers temporal aggregation over shots, within which there is high visual similarity between frames, in this work we extend such form of temporal aggregation to much longer video segments, which present varied visual contents. In subsequent work, Zhu et al. [26] proposed a system that makes use of query-adaptive asymmetrical dissimilarities, based on a BoW model, achieving top performance in INS 2013. In this work, we extend this idea to develop asymmetric comparison schemes for Fisher vectors, and we further consider two different asymmetric retrieval cases, with varying amounts of clutter.

III. Asymmetric Comparisons for Fisher Vectors

The Fisher vector (FV) [27] is a state-of-the-art global descriptor for image retrieval. Using this technique, the similarity of two images is measured by the similarity of two

FVs, one for each image. In some visual retrieval problems, however, one is not interested in measuring how similar the query and database items are, but rather if the query image is contained in a database image, or vice-versa. In this section, we introduce comparison techniques to address this scenario.

A. Review of Fisher Vectors

Let $X = \{x_t, t = 1 \dots T\}$ represent a set of T d-dimensional local descriptors, extracted from an image, and let $u_\lambda(X)$ denote a probability distribution for sets of local descriptors, with parameters λ. The Fisher kernel framework [28] describes X using the gradient vector of the log-likelihood:

$$G_\lambda^X = \frac{1}{T}\nabla_\lambda \log u_\lambda(X) \tag{1}$$

The Fisher kernel between sets X and Y is defined as:

$$K(X,Y) = (G_\lambda^X)^T F_\lambda^{-1} G_\lambda^Y \tag{2}$$

where F_λ denotes the Fisher information matrix. This kernel can be rewritten as an inner product between normalized vectors: $F_\lambda^{-1} = (L_\lambda)^T L_\lambda$. The FV of X is thus defined as:

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X \tag{3}$$

In practice, $u_\lambda(X)$ is simplified by assuming i.i.d. local descriptors. The distribution of x_t, denoted $u_\lambda(x_t)$, is modeled by a Gaussian mixture model (GMM) with diagonal covariance matrices [29]. Thus, $u_\lambda(X) = \prod_{t=1}^T u_\lambda(x_t)$, with $u_\lambda(x_t) = \sum_{k=1}^K w_k u_k(x_t)$, where $u_k(x_t)$ is a Gaussian density with mean vector μ_k and diagonal covariance matrix σ_k. In this case, $\lambda = \{w_k, \mu_k, \sigma_k, k = 1 \dots K\}$. Usually, only the gradients with respect to $\{\mu_k\}$ are taken into account [27], and F_λ^{-1} can be approximated as a whitening operation. The d-dimensional weighted gradient with respect to μ_k can be derived as:

$$\mathcal{G}_k^X = \frac{w_k^X}{\sqrt{w_k}} \sigma_k^{-1}(\mu_k^X - \mu_k) \tag{4}$$

Defining $\gamma_t(k) = \frac{w_k u_k(x_t)}{\sum_{j=1}^K w_j u_j(x_t)}$ as the soft assignment of x_t to the k-th Gaussian, w_k^X and μ_k^X can be written as:

$$w_k^X = \frac{1}{T}\sum_{t=1}^T \gamma_t(k) \tag{5}$$

$$\mu_k^X = \frac{\sum_{t=1}^T \gamma_t(k) x_t}{\sum_{t=1}^T \gamma_t(k)} \tag{6}$$

w_k^X corresponds to the proportion of local descriptors soft-assigned to Gaussian k, and μ_k^X to the weighted local descriptor vector corresponding to Gaussian k. The final vector $\mathcal{G}_\lambda^X$ is the concatenation of the $\mathcal{G}_k^X$ vectors, with total number of dimensions given by $K \times d$.

FVs undergo two normalization steps. The first step reduces the influence of bursty local features [30]. Perronnin et al. [31] apply the transformation $f(z) = \text{sign}(z)|z|^\beta$ to each component of $\mathcal{G}_\lambda^X$, where typically $\beta = 0.5$. This is known as signed square rooting (SSR). Another option is intra-normalization (IN), introduced by Arandjelović and Zisserman [32], where each $\mathcal{G}_k^X$ is L_2-normalized independently. In this work, we experiment with both of these normalization methods. The second step performs L_2-normalization of the entire FV. Thus, the comparison of FVs using the Euclidean distance is equivalent to using the cosine similarity. We denote by v_k^X the k-th d-dimensional normalized components of the FV (also called the k-th FV residual). The full FV is simply the concatenation of all v_k^X's, and we denote it by v^X.

Binarized Fisher vectors (FV*). Perronnin et al. [31] proposed to binarize FVs, in order to improve their scalability. In this proposed scheme, named compressed Fisher vector (CFV), each component of the FV is binarized depending on its sign, and an extra bit per Gaussian is used to encode w_k^X:

$$v_k'^X = \text{sign}\left(v_k^X\right) \tag{7}$$

$$b_k^X = \begin{cases} 1, \text{ if } w_k^X > \tau_w \\ 0, \text{ otherwise} \end{cases} \tag{8}$$

where the $\text{sign}(.)$ operation is applied component-wise. CFV's dimensionality is $K \times (d+1)$ bits. This simple binarization scheme has been shown to outperform more sophisticated binarization approaches [31], [14].

Fisher vector comparison schemes. The commonly used measure for comparing a query FV v^Q and a database FV v^D is the cosine similarity:

$$\frac{1}{\|v^Q\|_2 \|v^D\|_2} \sum_{k=1}^K (v_k^Q)^T v_k^D \tag{9}$$

where $\|.\|_2$ denotes the L_2 norm. Note that in a retrieval application, where the query image is compared to several database images, the term $\frac{1}{\|v^Q\|_2}$ can be omitted, since it is independent of database images. When FVs are L_2-normalized, the term $\frac{1}{\|v^D\|_2}$ can be omitted as well. Nevertheless, we include these terms in the presented expressions for greater clarity.

For binarized FVs, the cosine similarity measure leads to:

$$\frac{1}{\|v'^Q\|_2 \|v'^D\|_2} \sum_{k=1}^K (v_k'^Q)^T v_k'^D = \frac{1}{Kd}\sum_{k=1}^K (d - 2H_k) \tag{10}$$

where H_k denotes the Hamming distance between vectors $v_k'^Q$ and $v_k'^D$. The term $(d - 2H_k)$ is exactly equal to $(v_k'^Q)^T v_k'^D$. Also, note that $\|v'^Q\|_2 = \|v'^D\|_2 = \sqrt{Kd}$. Equation (10) does not approximate the original FV cosine similarity (9) well. A better approximation of the FV inner product [31] is:

$$\frac{1}{\sqrt{d\sum_{k=1}^K b_k^Q}\sqrt{d\sum_{k=1}^K b_k^D}} \sum_{k=1}^K b_k^Q b_k^D (d - 2H_k) \tag{11}$$

In this case, if either $b_k^Q = 0$ or $b_k^D = 0$, the score computation for Gaussian k is omitted. We refer to this scheme as Symmetric Gaussian Skipping (SGS), as it proposes to omit (skip) Gaussian components based on a criterion that depends on both the query's and the database's signatures. SGS was introduced in [31] in order to approximate the inner product between the FVs of Q and D: if either w_k^Q or w_k^D is small, the FV inner product for Gaussian k is close to zero and thus (11) approximates it as zero. Duan et al. [33] further extend (11):

$$\frac{1}{\left(d\sum_{k=1}^K b_k^Q\right)^\alpha \left(d\sum_{k=1}^K b_k^D\right)^\alpha} \sum_{k=1}^K b_k^Q b_k^D \beta_{H_k}(d - 2H_k) \tag{12}$$

where β_{H_k} is a trained weight that depends on H_k, and α defines a power law normalization.

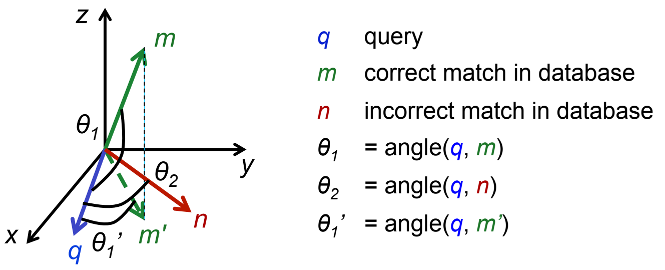


Figure 2: Simplified illustration of a common failure case when using FVs for asymmetric problems. In this case, the query q is supposed to have a small angle θ_1 compared to the correct match database item m, and a large angle θ_2 compared to the incorrect match database item n. However, $\theta_1 > \theta_2$. But the projected correct match database item m' is closer to q than n ($\theta_1' < \theta_2$) – note that in this toy example the vectors q and n live on the x-y plane. This shows that the database items should be compared to the query only based on the type of visual information the query contains (in this case, the vectors should be compared based on their projections to the x-y plane).

B. Asymmetric Fisher Vector Comparison Schemes

We introduce an asymmetric score computation for FVs, which we name Asymmetric Gaussian Skipping (AGS), in contrast to SGS. SGS was introduced in [31] such that the binarized FV inner product would better approximate the original FV inner product. In contrast, we introduce AGS as a technique that can be used with both binarized and non-binarized FVs.

1) Geometric Interpretation: Before introducing the specific technique for asymmetric comparisons, we illustrate the problem with a toy example. We consider an asymmetric application where the query item is mostly contained in a database item. Remember that each d-sized chunk of a FV corresponds to a different type of feature (since each chunk corresponds to a Gaussian residual in descriptor space). Fig. 2 illustrates the simplified setting in 3D where each component x, y, z corresponds to a different type of feature. The vectors q, m and n illustrate a query FV, a correct match database FV, and an incorrect match database FV. We would expect the angle θ_1 between q and m to be smaller than the angle θ_2 between q and n, but, in fact, $\theta_1 > \theta_2$. However, m's projection to the x-y plane, denoted m', is closer to q than n, since $\theta_1' < \theta_2$. This is a common failure case if asymmetric comparisons are not employed. It can be avoided if database items are compared to the query based only on their projections to the x-y plane. In other words, it is not important if a database item contains visual information represented by other Gaussians (clutter), since we are only interested in testing if it contains visual information from the query item.

This technique assumes that the clutter (visual information present in m but not in q) is mostly composed of features that are different from the features present in q. In practice, this assumption holds and the proposed technique works well.

2) Interpretation of Power Law Normalization: We introduce a new interpretation for the power law normalization from (12), which allows us to design improved asymmetric comparison techniques. Consider the usage of SGS for FVs. Adapting (9), the score between query and database FVs is:

$$\frac{1}{\left(\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2\right)^{\alpha} \left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{\alpha}} \sum_{k=1}^{K} b_k^Q b_k^D (v_k^Q)^T v_k^D \tag{13}$$

To the best of our knowledge, ours is the first work that proposes SGS for non-binarized FVs: in previous work, SGS has only been used with binarized FVs in order to better approximate the original FV inner product. (13) can be rewritten as:

$$\begin{aligned}&\left(\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2\right)^{0.5-\alpha} \left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{0.5-\alpha} \\ &\times \frac{1}{\sqrt{\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2}\sqrt{\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2}} \sum_{k=1}^{K} b_k^Q b_k^D (v_k^Q)^T v_k^D \\ &\propto \left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{0.5-\alpha} \cos.\mathrm{sim}(b^Q, v^Q, b^D, v^D)\end{aligned} \tag{14}$$

where $\cos.\mathrm{sim}(b^Q, v^Q, b^D, v^D)$ denotes the cosine similarity of v^Q and v^D, when residuals are selected by b^Q and b^D, respectively. The last step in this derivation uses the fact that $\left(\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2\right)^{0.5-\alpha}$ is fixed when comparing the query to any database image, so it simply multiplies the score of each database image by the same factor. The final expression reveals the effect of the power normalization α when using SGS: each database image's score is weighted by $\left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{0.5-\alpha}$. When $\alpha = 0.5$, the score reduces to the cosine similarity. When $\alpha < 0.5$, this weight boosts the score of database images with many selected Gaussians. As each Gaussian represents a different type of information, this weighting factor tends to favor database images which contain diverse features. While [34] proposed that $0 \leq \alpha \leq 0.5$, we explore even lower (i.e., negative) values of α, which can be helpful in some cases.

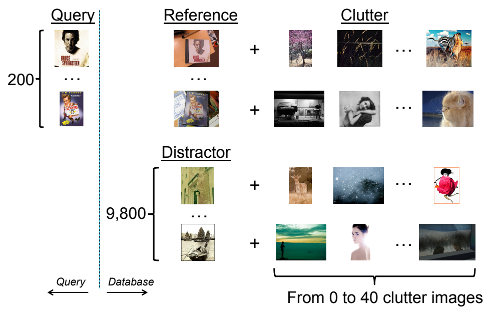
The interpretation of power law normalization as a weight which boosts the score of database images with large variety of features allows us to use such weighting in other comparison techniques, such as the one introduced in the following.

3) Query-Based AGS (QAGS): For the case where our retrieval goal is to test whether the query Q is contained in the database item D, we introduce query-based AGS (QAGS). For FVs, a simple QAGS score can be computed as:

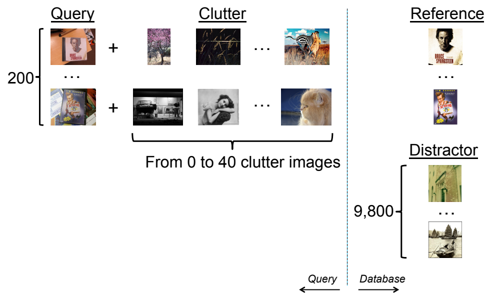
$$\frac{1}{\sqrt{\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2}\sqrt{\sum_{k=1}^{K} b_k^Q \left\|v_k^D\right\|_2^2}} \sum_{k=1}^{K} b_k^Q (v_k^Q)^T v_k^D \tag{15}$$

where b_k^Q is either zero or one, as in (8). This expression computes the cosine similarity of the query and database vectors projected to a subspace defined by the query image (as illustrated in Fig. 2): the residuals of both the query and database images are selected by b_k^Q, and the normalization factors take into account solely the selected residuals. This is approximately equivalent to extracting a FV that is parameterized by a GMM that uses only the selected Gaussians.

Note how (15) effectively penalizes database items in some important cases, while SGS (13) does not. For example, consider the case where, for a Gaussian k, $b_k^Q = 1$ and $b_k^D = 0$.



(a)



(b)

Figure 3: Illustration of the datasets used for experiments of Sec. III. (a) Asym-QCD (query contained in database), (b) Asym-DCQ (database contained in query). For both datasets, we evaluate different levels of asymmetry by using up to 40 clutter images.

In this case, the score $(v_k^Q)^T v_k^D$ is likely low. The SGS score (13) would hide this fact by ignoring this Gaussian and decreasing the normalization factor. However, the QAGS score (15) does not ignore this Gaussian and does not decrease the normalization factor: the score for this Gaussian is taken into account, and since it is low, the database item is penalized.

QAGS might also be beneficial even when the dataset does not contain any asymmetry: the Gaussians that have $b_k^Q = 0$ would contribute to the score if they are not skipped, even if the residuals in this case are originally low. If the intra-normalization technique is used, for example, these residuals are scaled to have unit norm and might end up contributing significantly to the score.

We can improve the QAGS scoring scheme by using the database-side weight derived in Subsec. III-B2:

$$\left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{0.5-\alpha} \times \frac{1}{\sqrt{\sum_{k=1}^{K} b_k^Q \left\|v_k^Q\right\|_2^2}\sqrt{\sum_{k=1}^{K} b_k^Q \left\|v_k^D\right\|_2^2}} \sum_{k=1}^{K} b_k^Q (v_k^Q)^T v_k^D \quad (16)$$

In this expression, the added weight helps the selection of relevant database images – as shown in experimental results, this improves retrieval performance. The QAGS score can be computed for binarized FVs (FV*s) in a similar manner.

4) Database-Based AGS (DAGS): If the asymmetry is reversed, i.e., when a database image might be contained in the query image, we adapt the asymmetric score accordingly:

$$\frac{1}{\left(\sum_{k=1}^{K} b_k^D \left\|v_k^Q\right\|_2^2\right)^{\alpha}\left(\sum_{k=1}^{K} b_k^D \left\|v_k^D\right\|_2^2\right)^{\alpha}} \sum_{k=1}^{K} b_k^D (v_k^Q)^T v_k^D \quad (17)$$

We refer to this scheme as database-based AGS (DAGS): in this case, residuals are selected based on the database image. This means that the comparison between the query and each database image is performed based on a different projection. The use of the power normalization α is crucial in this case, as for $\alpha < 0.5$ database images with more visual information are favored, as explained in Subsec. III-B2. As shown in experiments, results are poor if $\alpha = 0.5$: this corresponds to simply performing each comparison based on a different projection – in this case, spurious database images with small amount of visual information might obtain high score, and the retrieval system might not work well. For the binarized FV (FV*) case, we propose a similar DAGS score computation.

C. Experimental Evaluation

Datasets. We are interested in evaluating the impact of the proposed techniques to retrieval problems with varying degrees of asymmetry. We construct two datasets[1], illustrated in Fig. 3: (a) Asym-QCD, where the query image is contained in a database image, and (b) Asym-DCQ, where a database image is contained in the query image. The query images used in Asym-QCD are clean images of objects, and their corresponding correct database matches are images where the object is shown along with clutter. For the Asym-DCQ dataset, these two sets of images are reversed. Distractor images are added to expand the database to $10,000$ items. We construct several versions of the two datasets, simulating different amounts of asymmetry by adding a set of clutter images to query or database items – we denote the number of clutter images as C. For the Asym-QCD dataset, $C = 0$ to $C = 40$ clutter images are added to each database item and, for the Asym-DCQ dataset, $C = 0$ to $C = 40$ clutter images are added to each query item. For a set of images constituting one database item or one query item, we pool the local descriptors of all images and extract a single FV for the set. The query and reference images are collected from the Stanford Mobile Visual Search dataset [35], while distractor and clutter images are collected at random from the Holidays [36] and MIRFLICKR-1M dataset [37].

Detector, local and global descriptors. We use the Hessian-Affine keypoint detector [38], and SIFT local descriptors [39]. Using PCA, the dimensionality of SIFT descriptors is reduced to $d = 32$. For computation of FVs and FV*s, we use $K \in \{512, 1024, 2048\}$ Gaussians. We do not use the weights from (12) in the experiments using FV*, since their effect is complementary to the contributions introduced in this section.

Comparison techniques. We evaluate the proposed comparison techniques QAGS and DAGS against existing comparison techniques: baseline (no Gaussian skipping) and SGS. More specifically, for FVs, the baseline uses (9), SGS uses (13), QAGS uses (16) and DAGS uses (17). Similar expressions are used for FV*s.

Performance measure. Results are evaluated using Average Precision for each query, computed over the ranked list of the top 100 database items. Mean Average Precision (mAP) is reported for results over the query set. For each query, the objective is to retrieve the database item containing the corresponding reference image in the database.

[1]These new datasets are available at https://purl.stanford.edu/hg081bj1051.

C	0	1	2	5	10	20	40
FV baseline SSR	83.3	74.7	72.1	56.6	44.6	29.9	17.9
FV* baseline SSR	79.9	73.7	70.5	57.5	45.9	29.5	15.4
FV baseline IN	79.8	75.4	71.1	59.0	50.1	30.8	16.7
FV* baseline IN	79.9	73.7	70.5	57.5	45.9	29.5	15.4
FV QAGS SSR	85.4	78.2	75.9	61.4	47.2	32.9	19.1
FV* QAGS SSR	89.4	83.4	79.6	66.9	54.8	35.0	18.7
FV QAGS IN	**89.5**	**84.4**	**81.2**	**70.4**	**57.6**	**38.6**	**21.4**
FV* QAGS IN	89.4	83.4	79.6	66.9	54.8	35.0	18.7
FV DAGS IN	85.5	79.6	76.6	62.3	52.4	31.6	17.2
FV* DAGS IN	86.2	79.6	75.7	60.0	49.0	29.9	15.4
FV SGS IN	88.3	81.9	79.4	66.7	54.9	35.2	20.0
FV* SGS IN	87.9	81.2	79.3	64.3	54.0	33.4	18.3

Table I: Retrieval performance (% mAP) on the Asym-QCD dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), varying the number of added database clutter images (C), for K = 512. For each configuration, we report the best performance varying τ_w and α.

C	0	1	2	5	10	20	40
FV baseline SSR	89.1	79.5	76.3	67.1	46.0	30.0	15.0
FV* baseline SSR	86.5	78.3	76.3	61.4	38.7	23.7	9.2
FV baseline IN	88.8	79.4	78.5	63.1	43.3	25.2	10.5
FV* baseline IN	86.5	78.3	76.3	61.4	38.7	23.7	9.2
FV QAGS IN	92.9	84.6	81.1	68.6	50.9	32.5	18.8
FV* QAGS IN	91.8	83.5	79.7	66.0	46.5	30.8	16.9
FV DAGS SSR	90.4	82.2	78.8	71.7	51.7	34.8	21.7
FV* DAGS SSR	93.3	85.6	81.6	70.8	53.8	35.6	19.9
FV DAGS IN	93.5	**86.7**	**82.0**	**74.2**	**57.3**	**36.8**	**22.4**
FV* DAGS IN	93.3	85.6	81.6	70.8	53.8	35.6	19.9
FV SGS IN	**94.3**	86.0	81.7	72.0	54.8	34.7	21.1
FV* SGS IN	94.2	85.0	80.7	69.0	52.4	33.6	18.1

Table II: Retrieval performance (% mAP) on the Asym-DCQ dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), varying the number of added query clutter images (C), for K = 512. For each configuration, we report the best performance varying τ_w and α.

Results: query contained in database. First, we evaluate the usage of intra-normalization (IN). Tab. I presents results for FV and FV* using both SSR and IN. Performance using no Gaussian skipping (baseline) may degrade with IN, compared to the usage of SSR, since this technique gives too much importance to Gaussians with low w_k^X, i.e., Gaussians that are in practice not visited by local features. On the other hand, the FV QAGS IN technique improves performance compared to FV QAGS SSR, for all values of C: since in this scheme the Gaussians with low w_k^X are skipped, IN boosts retrieval performance by equalizing the importance of non-skipped Gaussians. Note that FV* SSR is exactly the same as FV* IN, since their difference is only the normalization, which does not change the binarized descriptor. In the rest of experiments using Asym-QCD, we make use of IN in all retrieval schemes.

Fig. 4a demonstrates the benefit of Gaussian skipping in the presence of small asymmetry (C = 0), by presenting QAGS and SGS results, compared to the non-skipping baseline, with K = 512. Fig. 4b shows that performance further improves for QAGS and SGS as α decreases, compared to using α = 0.5.

Fig. 5 presents the best performance for QAGS, SGS and the baseline scheme for each value of C. Since the x-axis uses a logarithmic scale, the results are presented as a function of the variable $C + 1$, such that the results using C = 0 can be seen in the graph. The benefit of using QAGS is clear, especially as K increases. QAGS systematically performs better than SGS as C increases. Note that the FV* results are very close to the FV results, which indicate that FV* is an effective approximation of FV. Overall, QAGS improves mAP by up to 25%, compared to the baseline scheme that does not perform Gaussian skipping.

Results: database contained in query. Tab. II shows consistent improvements when using DAGS with IN, compared to using DAGS with SSR. In the rest of experiments using Asym-DCQ, we make use of IN in all retrieval schemes.

Fig. 6a presents retrieval performance as a function of τ_w with α = 0.5, and Fig. 6b presents retrieval performance as a function of α, with τ_w = 10^{-4}, both plots considering C = 0. As expected, the results using DAGS with α = 0.5 are poor. DAGS results are much improved as α decreases. These plots show that retrieval performance can be much improved for C = 0 by using DAGS or SGS, compared to using the baseline FV scheme. Fig. 7 presents retrieval performance as C increases. The benefit of using DAGS is clear, especially as K increases. DAGS systematically performs better than SGS for C > 2. Overall, DAGS improves mAP by up to 25%, compared to the baseline scheme that does not perform Gaussian skipping.

IV. Temporal Aggregation

In this section, we design descriptors which can be used to compare video segments directly against query images. Fig. 8 presents the natural structure of video databases, serving to establish the nomenclature we use for different temporal units of video. A video segment is defined as a sequence of frames. Shots and scenes are two types of segments, which are delimited depending on the audiovisual contents of a video. A shot is a sequence of consecutive video frames taken without interruption by a single camera [40], [41]. Video frames within a shot are usually similar to each other. A scene is defined as a concise segment of video that contains interrelated shots and represents a semantic unit for the given type of content [41], [42]. In contrast to shots, scenes are longer video segments that contain diverse visual information. For example, in the context of news videos, scenes correspond to video segments that contain complete news stories. In this case, the scene often starts with an anchor shot, then cuts to a shot of a reporter in the field, etc. In another example, lecture videos, scenes correspond to video segments which present a single concept, or set of interrelated concepts. Both for news and lecture videos, scenes are typically several minutes long.

In this work, we are interested in designing scene-based descriptors for matching against query images. We introduce two techniques, in Subsecs. IV-A and IV-B, which lead to different trade-offs in terms of retrieval accuracy, latency and memory requirements. Experiments are presented in Subsec. IV-C. To the best of our knowledge, we are the first to study descriptors based on such long and diverse video segments for image-based retrieval. The techniques we develop can very efficiently scan the video database to find the scenes most likely to contain the query image. For a discussion on shot-based descriptors, we refer the reader to our previous work [7].

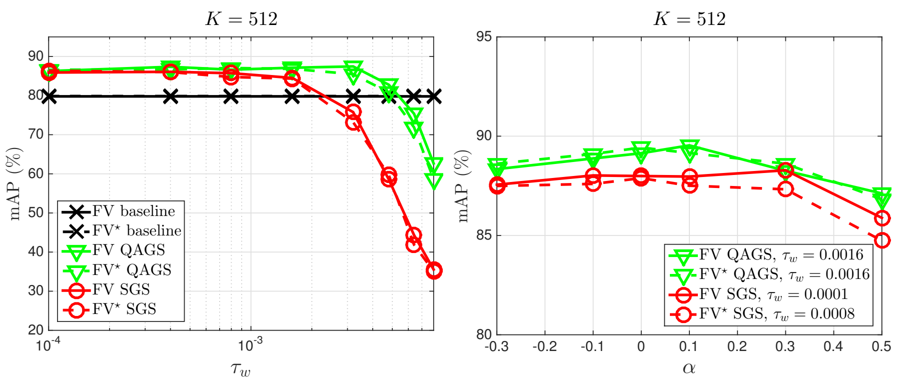


Figure 4: Retrieval results on the Asym-QCD dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), $K = 512$, $C = 0$, varying (a) τ_w (with $\alpha = 0.5$) and (b) α (with different values of τ_w).

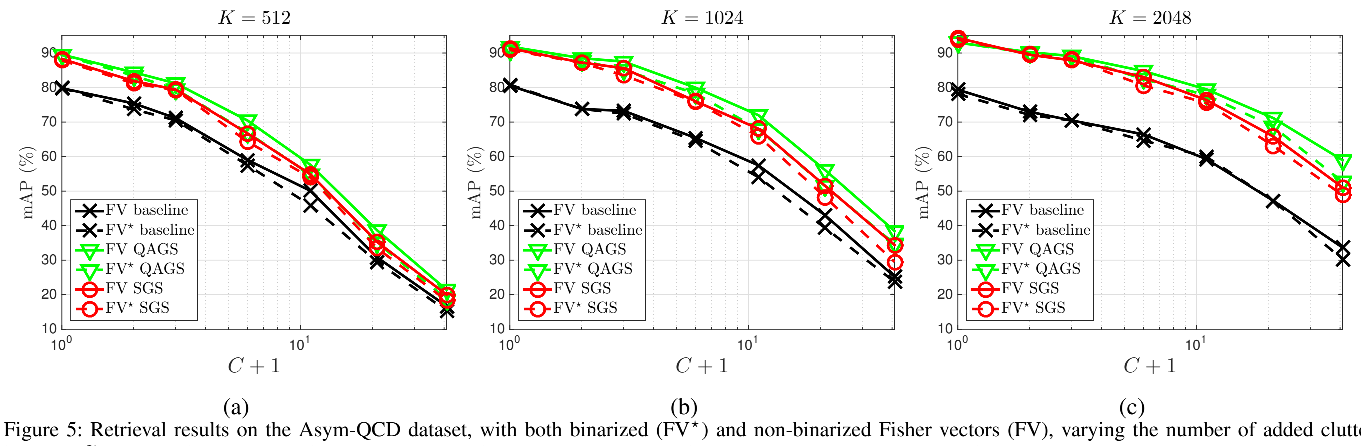


Figure 5: Retrieval results on the Asym-QCD dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), varying the number of added clutter images (C), for (a) $K = 512$, (b) $K = 1024$ and (c) $K = 2048$. For each configuration, we report the best performance varying τ_w and α.

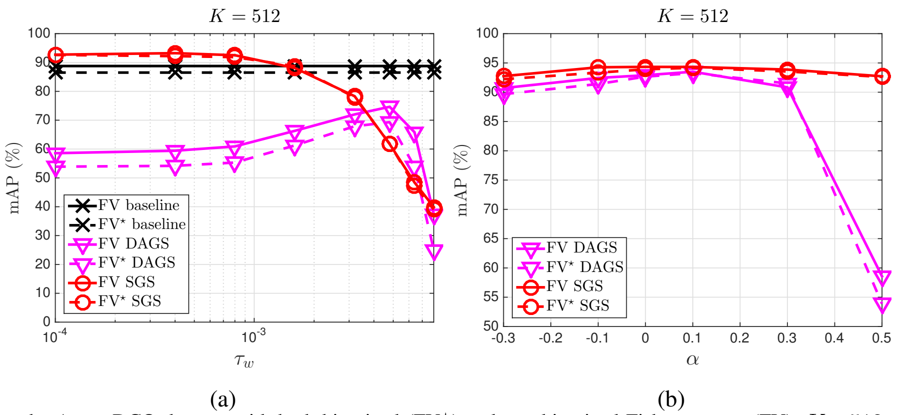


Figure 6: Retrieval results on the Asym-DCQ dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), $K = 512$, $C = 0$, varying (a) τ_w (with $\alpha = 0.5$) and (b) α (with $\tau_w = 10^{-4}$).

A. *Temporal Aggregation Using Fisher Vectors*

In this subsection, we consider temporal aggregation of visual information using the FV framework. Since it is not obvious how to extend FVs in order to generate scene-based signatures, we experiment with different approaches, described in the following.

Scene FV. Keypoints are detected from each frame independently, and local descriptors are extracted for each keypoint. In this technique, a FV is constructed by simply aggregating all local descriptors from the frames within a scene.

Scene FV with tracked features (Scene FV-TF). Using keypoints detected independently from each frame, we perform tracking to cluster similar keypoints from consecutive frames. The tracking algorithm works by comparing keypoints based on their locations and descriptors: two keypoints are considered part of the same track if their spatial and descriptor distances are small enough. Then, the local descriptors within a track are averaged within each scene and L_2-normalized. Finally, the averaged track descriptors in a scene are aggregated into a FV. Note that this mode inserts an early aggregation stage into the system: averaged descriptors over a track might lose some discriminative power before being aggregated into FVs.

Averaged shot FV (Avg. Shot FV). First, we extract FV signatures for each shot within a scene, by using a technique

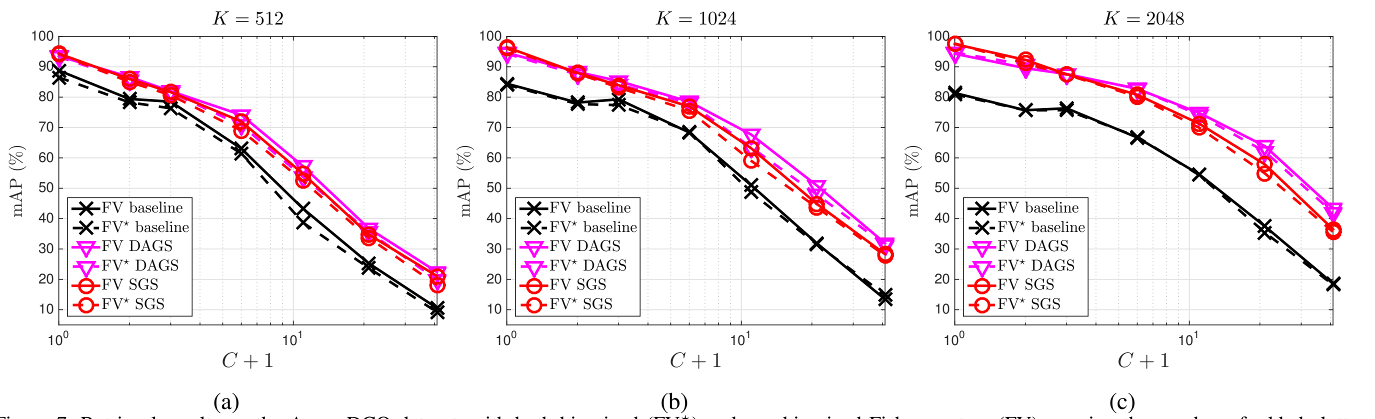


Figure 7: Retrieval results on the Asym-DCQ dataset, with both binarized (FV*) and non-binarized Fisher vectors (FV), varying the number of added clutter images (C), for (a) $K = 512$, (b) $K = 1024$ and (c) $K = 2048$. For each configuration, we report the best performance varying τ_w and α.

similar to Scene FV – except that the aggregation happens over a shot, instead of over a scene. These FV signatures are averaged, and the resulting vector is used to describe the scene.

Frame FV. This mode constitutes a simple algorithm which serves as a reference for the aggregation techniques we develop. In this case, FVs are constructed independently for each frame in a scene, and no scene-based aggregation is performed.

Note that the techniques Scene FV, Avg. Shot FV and Scene FV-TF generate a single FV per scene, while Frame FV generates multiple FVs per scene. If the same number of Gaussians is used in both cases, Frame FV requires much more memory and computation for retrieval, compared to the other techniques. The Scene FV-TF technique is similar in spirit to burstiness removal methods for I2I retrieval problems [43], [30], which showed retrieval accuracy gain. In our case, similar features from consecutive frames can be seen as temporal bursts – instead of spatial bursts addressed in [43], [30].

Finally, the Scene FV and Avg. Shot FV techniques discard information related to the ordering of frames. In other words, the representation for a given scene would be the same regardless of the ordering of its constituent frames. This is akin to the use of BoW or FVs in image retrieval, where the representation is the same regardless of where local features appear in an image.

B. Temporal Aggregation Using Bloom Filters

In this subsection, we consider a different approach to temporal aggregation. 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 the following, we review the concept of Bloom filters, then introduce techniques to enable efficient large-scale retrieval.

1) Review of Bloom Filters: A Bloom filter (BF) [44] is a data structure designed for set membership queries, widely used in distributed databases and networking applications – for a review, see [45]. 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.

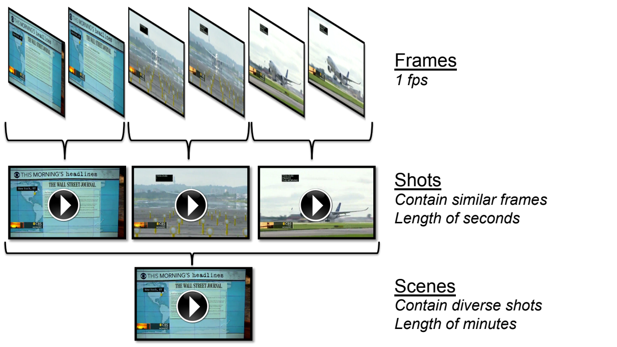


Figure 8: Temporal structure of videos. In this work, frames are extracted at 1 frame per second (fps). Shots are sequences of frames taken without interruption by a single camera, and in the databases we consider their length is on the order of seconds. Scenes are longer video segments that contain interrelated shots and represent a semantic unit for the given type of content (for example, for news content, a scene would correspond to a news story). In the databases we consider, their length is on the order of minutes.

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 $h_m : \mathcal{U} \to \{1,2,...,L_{np}\} \forall m$, 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 [44] 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

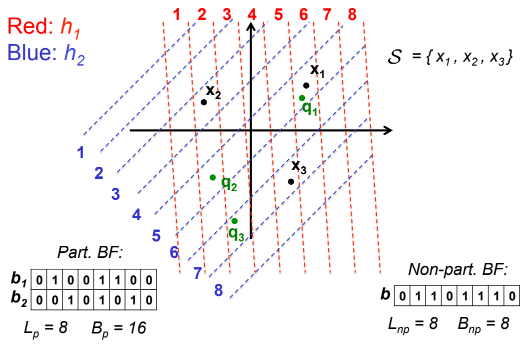


Figure 9: Illustration of a Bloom filter encoding set $\mathcal{S} = \{x_1, x_2, x_3\}$ in 2D. Two hash functions are shown ($M = 2$), in red and in blue, with bin numbers marked near the corresponding regions. Partitioned (left) and non-partitioned (right) BFs are presented. Examples of queries are 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).

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) [46], which address this problem, illustrated in Fig. 9. 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.

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 FV global descriptors – this method is denoted BF-GD.

3) BF-PI: Using Point-Indexed Descriptors: We also 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 the first aggregation step, by hashing embedded local descriptors into BFs directly. We make use of point-indexed representations, which were introduced by Tao et al. [47]. In [47], the authors show how a FV can be decomposed into the Fisher embedding of each local descriptor, leading to a point-indexed representation: instead of storing a FV, the database stores an embedded version of each local descriptor. Our proposed technique is called BF-PI. Consider a local feature x and a FV with parameters $\{w_k, \mu_k, \sigma_k, k = 1 \dots K\}$, as before. As in [47], we employ the point-indexed representation of x using only the Gaussian from the FV which obtains the strongest soft-assignment probability. The point-indexed representation for x is a triplet:

$$\{r; \frac{\gamma_x(r)}{\sqrt{w_r}}; d_x = \sigma_r^{-1}(x - \mu_r)\} \tag{18}$$

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

4) Hash Functions & Scoring:

LSH families. We consider three LSH families. The metric for comparing FVs is cosine similarity, so a natural choice for this problem is the family for cosine distance [48], 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. It has been widely used in information retrieval [49], [50]. We also consider the 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 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), $h_{GBH} : \mathcal{R}^d \rightarrow 2^n$. For BF-PI, we hash 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 [51]. 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 q and the m-th hash function, the score $S_v^{\#}$ of database scene v is updated as:

$$S_v^{\#} := S_v^{\#} + \mathbf{b}_v[h_m(q)] \tag{19}$$

In other words, the score of scene v is incremented if its $h_m(q)$-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 scene v can be computed as:

$$S_v^T := S_v^T + \mathbf{b}_v[h_m(q)] \cdot \frac{i^2_{h_m(q)}}{(\sum_l \mathbf{b}_v[l] i_l^2)^\eta} \tag{20}$$

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

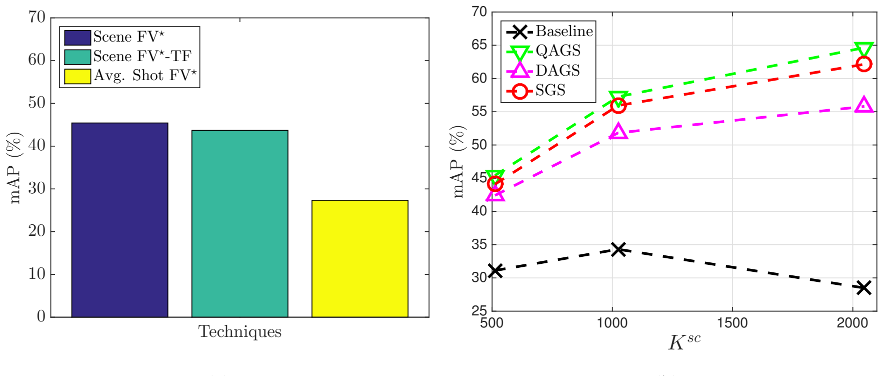


Figure 10: Retrieval results on the ClassX-600k dataset, using scene descriptors based on binarized Fisher vectors (FV⋆). (a) Comparison of different scene aggregation schemes, using QAGS with $K^{sc} = 512$ and $\alpha^{sc} = 0.5$. We report the best performance varying τ_w^{sc}. (b) Retrieval performance of the Scene FV⋆ scheme for different asymmetric scoring schemes, as a function of K^{sc}. For each data point, we report the best performance varying τ_w^{sc} and α^{sc}.

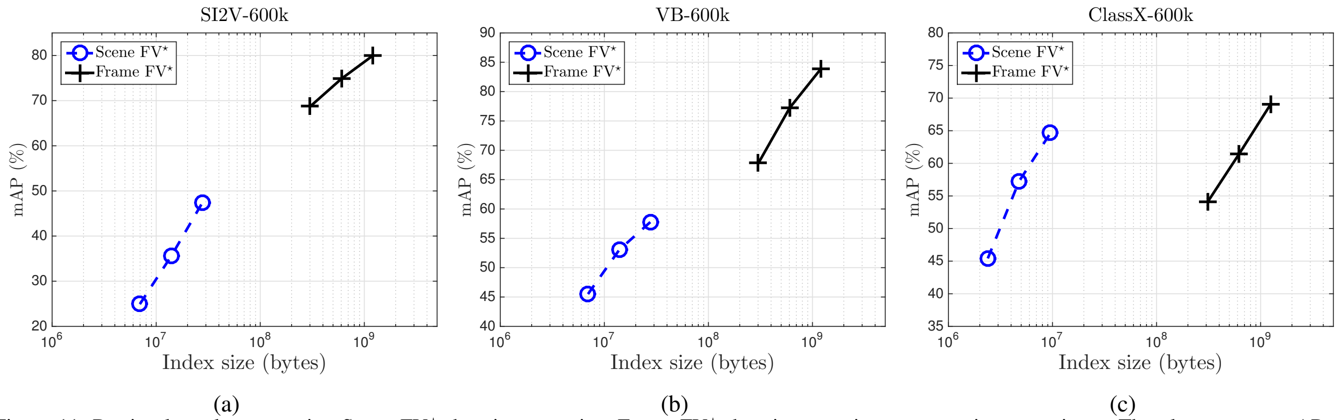


Figure 11: Retrieval results comparing Scene FV⋆ descriptors against Frame FV⋆ descriptors, using asymmetric comparisons. The plots present mAP as a function of the index size, for three datasets: (a) SI2V-600k, (b) VB-600k and (c) ClassX-600k. Each curve is drawn by varying the number of Gaussians in the FV construction. For each data point, we report the best performance varying τ_w^{sc} and α^{sc}.

C. Experimental Evaluation

Datasets. We consider 3 datasets. The Stanford I2V (SI2V) dataset is currently the largest dataset for this research problem [23]. It contains news videos, and query images are collected from the web. The Video Bookmarking (VB) dataset [8] 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, named ClassX, contains lecture videos [8], with queries being clean images of slides. In all cases, we extract 1 frame per second. In this section, we use the dataset versions SI2V-600k, VB-600k and ClassX-600k (each containing 600k frames and 160 hours of video). More than 200 queries are used per dataset. To train auxiliary structures (e.g., GMM, PCA), we use independent datasets [8].

Performance measure. We follow the evaluation procedure from previous work [7], [23], to obtain comparable figures: results are evaluated using mAP.

Detector and local descriptors. As in the previous section, we use the Hessian-affine detector [38], and describe keypoints using SIFT [39]. Using PCA, the dimensionality of SIFT descriptors is reduced to $d = 32$.

FV parameters. The scene descriptors introduced in Subsec. IV-A are evaluated in binarized format (FV⋆): the binarization is applied as the final step in scene signature construction, based on the sign of each component. To denote that we use the binarized version of these techniques, we simply substitute the term *FV* by *FV*⋆ – for example, the binarized version of Scene FV-TF is Scene FV⋆-TF. For computation of these signatures, we vary the number of Gaussians K^{sc} within $\{512, 1024, 2048\}$. The frame-based signatures are constructed using $K^{fr} \in \{128, 256, 512\}$ Gaussians. The variables τ_w^{sc} and α^{sc} correspond to the parameters used for asymmetric comparison computation when using these descriptors, following similar notation to Sec. III.

BF parameters. We set the number of Gaussians K^{BF} to 512 in all experiments using the BF framework. The number of hash functions M is chosen equal to K^{BF}, which is experimentally shown to achieve high performance. 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 $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$.

Results: FV-based temporal aggregation. Fig. 10 presents retrieval results on the ClassX-600k dataset. We do not make use of the weights from (12) in the experiments discussed in this paragraph – their effect is complementary to the techniques evaluated here. Fig. 10a compares the different scene-based FV aggregation methods – showing that Avg. Shot FV⋆ performs much worse than other methods, while Scene FV⋆ and Scene FV⋆-TF perform similarly. In the rest of the experiments using scene-based FVs, we make use of Scene FV⋆, due to

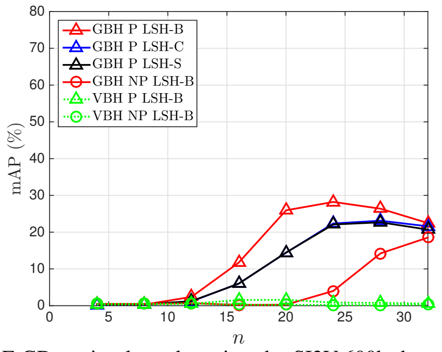


Figure 12: BF-GD retrieval results using the SI2V-600k dataset: mAP as a function of n. All curves use scoring based on the number of hash matches. Comparison of partitioned (P) versus non-partitioned (NP) BFs; GBH versus VBH hashes; LSH-B versus LSH-C and LSH-S.

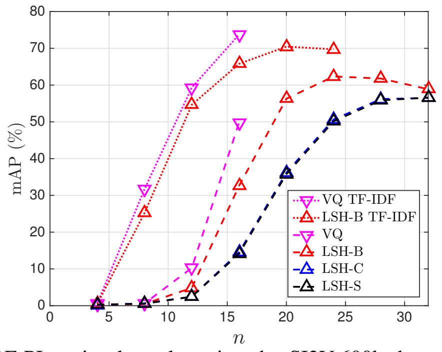


Figure 13: BF-PI retrieval results using the SI2V-600k dataset: mAP as a function of n. All curves use partitioned BFs and GBH. Comparison of VQ versus LSH hashes, and different scoring techniques.

	SI2V-600k	VB-600k	ClassX-600k
Frame-based techniques			
AlexNet FC6	48.43	18.18	3.70
AlexNet FC7	36.34	15.65	1.14
VGG16 FC6	34.41	6.99	1.53
VGG16 FC7	31.57	4.84	1.18
Frame FV	**71.51**	**70.40**	**55.02**
Scene-based techniques			
AlexNet FC6	7.09	10.38	1.20
AlexNet FC7	6.48	8.16	1.31
VGG16 FC6	6.71	3.75	1.29
VGG16 FC7	6.92	2.75	1.10
Scene FV	**13.04**	**34.00**	**30.60**

Table III: Retrieval results (mAP in %) on the 600k datasets, comparing the proposed FV-based methods (using asymmetric comparisons) against pre-trained CNN descriptors. All techniques generate descriptors with the same dimensionality ($4k$).

its simplicity and high performance. Fig. 10b evaluates the different FV comparison techniques (introduced in Sec. III) when using Scene FV* descriptors: baseline (no Gaussian skipping), QAGS, DAGS and SGS. The results show that the use of asymmetric comparisons (QAGS) is very important in this case. Fig. 11 presents retrieval results on the three datasets: mAP as a function of the index size. In these plots, we compare scene-based against frame-based descriptors. Scene FV* achieves excellent performance for the ClassX-600k dataset – it reduces the index size by approximately two orders of magnitude with no performance drop. For the SI2V-600k and VB-600k datasets, scene-based signatures achieve substantial memory savings (43×), but with a significant performance drop (more than 25% mAP in both cases).

Results: Comparison against pre-trained CNN features. Recently, it has been shown that features extracted using convolutional neural networks (CNN) achieve remarkable performance in image retrieval problems, even if the models are trained for a classification task [52], [53]. Tab. III presents a comparison of such pre-trained CNN features against the FV-based technique. We employ two widely-used models: AlexNet [54] and VGG16 [55]. Input frames are resized to 224×224 resolution, and features are extracted from the FC6 and FC7 layers (before ReLU) – as in previous work [52], [53]. For frame-based experiments, features are extracted for each frame and L_2-normalized. For scene-based experiments, features are extracted for each frame and sum-pooled within each scene (as in [56]), followed by L_2 normalization. In both cases, the CNN descriptor contains $4k$ floating-point dimensions. For a fair comparison, Tab. III presents FV-based results which use 128 Gaussians and no binarization, leading to exactly the same dimensionality. FV-based techniques outperform CNN-based techniques substantially, for both frame-based and scene-based experiments, in all datasets. Pre-trained CNN features optimized for an image classification task do not provide satisfactory performance for the image-to-video instance retrieval problems studied in this paper.

Results: BF-GD. Fig. 12 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. 12 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. 12 shows that LSH-B outperforms LSH-C and LSH-S. Overall, BF-GD obtains limited mAP, showing that a straightforward BF aggregation method may not be the best choice for this problem.

Results: BF-PI. Fig. 13 compares the different hashing and scoring techniques, when using BF-PI. BF-PI provides a substantial improvement in mAP, compared to BF-GD: more than 30%. This demonstrates the benefit of removing the aggregation per frame before hashing. Fig. 13 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. BF-PI using $n = 16$, coupled with VQ-based hashing and TF-IDF scoring, outperforms all other BF configurations we experimented with.

Comparison of temporal aggregation approaches. Tab. IV presents summarized results for experiments on the 600k datasets, using scene-based descriptors with the best configurations experimented here. For a fair comparison against Scene FV*, we add scoring weights (presented in (12)), which improve slightly the previously presented results. We also present Scene FV* results from previous work [7], where difference-of-Gaussian keypoints were used for retrieval on the SI2V dataset. We can see that the use of Hessian-affine

	SI2V-600k	VB-600k	ClassX-600k
Scene FV* (DoG) [7]	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 IV: Summary of retrieval results (mAP in %) for the 600k datasets. All techniques use Hessian-affine keypoints, except for [7], which uses difference-of-Gaussian (DoG) keypoints. The BF techniques presented here use GBH hashes, partitioned BFs and TF-IDF.

keypoints boosts performance of Scene FV*. The BF-PI scheme, coupled with VQ hashing, outperforms other approaches significantly for SI2V-600k, by 23.74% mAP compared to Scene FV*. In the VB-600k dataset, it also outperforms other approaches, but with a smaller margin: 5.50% better than Scene FV*. In the ClassX-600k dataset, BF-PI VQ is slightly worse than Scene FV*, by 3.05%. In the next section, we provide a thorough large-scale comparison of these different techniques.

V. Large-Scale Experiments

In this section, we present large-scale experiments. We first evaluate the top-performing scene-based descriptors introduced in Sec. IV. Then, we implement a practical query-by-image video retrieval system, suitable to large databases, using inverted index retrieval structures – performance is compared against a frame-based technique.

A. Comparison of Scene Descriptors

In this subsection, we present a comparison of the top-performing scene-based descriptors introduced in Sec. IV: Scene FV* and BF-PI. The parameters for these two techniques were selected as those which provided the best performance on the experiments from Sec. IV.

Datasets. We use large-scale versions of the datasets from Sec. IV, containing many more database video clips: SI2V-14M, VB-14M (14M frames and 3,801 hours of video) and ClassX-1.5M (1.5M frames and 408 hours of video).

Performance measure. As before, we use mAP to assess the quality of retrieval techniques, computed over the top-ranked 100 scenes. In this section, we are also interested in re-ranking the original list of scenes to generate a more accurate final list of results. For this use case, it is important to assess the proportion of relevant database scenes which are ranked among the top results in the initial list – regardless of their ordering, since the results will be re-ranked in a subsequent stage. To measure this type of result, we use Recall@100 (R@100). The drawback of mAP in this case is that it gives a much higher weight to top-ranked results. When using R@100, the scenes positioned at any rank in the results list receive the same weight. We use mean Recall@100 (mR@100) to evaluate retrieval results over the entire query set.

Results. Fig. 14 presents results on the three datasets, as the database size varies, measured using both mAP and mR@100. For the SI2V dataset, BF-PI performs much better than Scene FV* as the database size increases, in terms of both measures. In the VB dataset, the gap in retrieval quality between the two techniques is not very large. BF-PI outperforms Scene FV* in terms of mR@100 as the database grows; in terms of mAP, Scene FV* outperforms BF-PI at large scale. Results for the ClassX dataset show similar findings: in large-scale, BF-PI outperforms Scene FV* in terms of mR@100, while the opposite happens when considering mAP. Note that, in this dataset, the mR@100 performance is very high for all techniques.

B. Experiments with Scene Re-ranking

In this section, we present a practical query-by-image video retrieval system, which performs re-ranking to narrow down results to the shot level. For large databases, it is infeasible to use linear search. A scalable solution involves the use of inverted index structures, such that only a fraction of database items are considered during query time. For BF-PI, we can represent it in an inverted index format, which is straightforward. For Scene FV* and Frame FV*, we use the Multi-Block Indexing Table (MBIT) [57] method, which is a state-of-the-art inverted index technique suitable to binarized global descriptors. The choice of MBIT is due to its recent standardization by MPEG, as part of the Compact Descriptors for Visual Search (CDVS) effort [57].

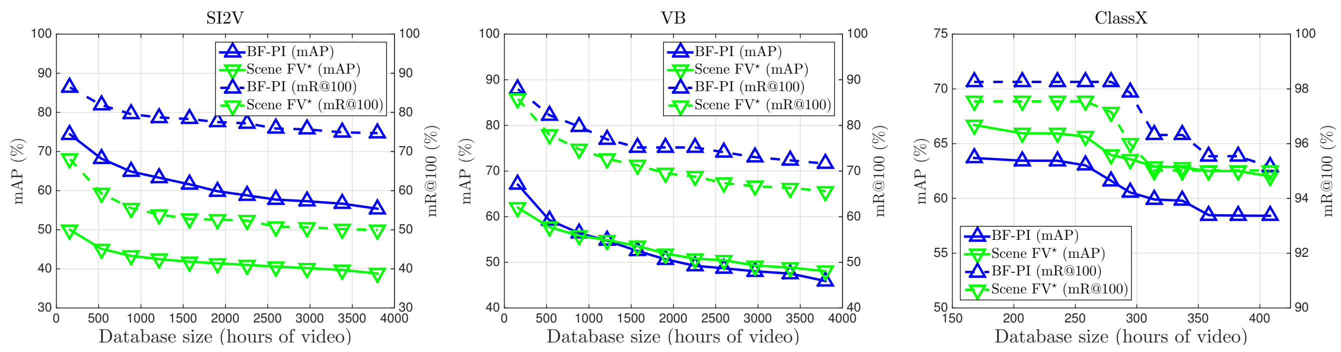
Datasets. In these experiments, we are interested in comparing the proposed retrieval techniques against a baseline system which indexes each frame in the database. For this reason, we make use of the large-scale SI2V-4M and VB-4M dataset versions (4M frames and 1,079 hours of video), such that the frame-based system fits in memory and can be properly compared. For the ClassX dataset, we make use of the same version as before: ClassX-1.5M.

Experimental setup. In this subsection, retrieval quality is assessed in terms of mAP. For the baseline Frame FV* technique, we selected parameters which provide the best performance on the 600k dataset versions. For BF-PI and Scene FV*, we re-rank the top scene results using shot-based FV*s, as in [7]. For the three compared techniques, we selected operating points which achieve similar mAP – all techniques are compared based on a similar level of retrieval quality.

Results are presented in Tab. V, including latency and memory figures. The proposed techniques enable high quality retrieval with much reduced resources, compared to the baseline Frame FV* technique. For the operating points considered in Tab. V, the method based on BF-PI achieves slightly improved mAP compared to the frame-based technique in all datasets, while at the same time obtaining a speedup of 9.6×, 4× and 5.6×, for the SI2V-4M, VB-4M and ClassX-1.5M datasets, respectively. The retrieval technique which makes use of Scene FV* enables large-scale retrieval with very compact databases – it achieves the smallest index size in all cases. The effectiveness of this technique depends on the dataset. In the ClassX-1.5M dataset, it obtains slightly improved mAP compared to the baseline, while being 5.4× faster and 18× more memory-efficient. This might seem to contradict the results from Sec. IV, where the same mAP performance was obtained with roughly two orders of magnitude index compression. The improvement in terms of index size is not as pronounced here because the size of the shot-based FV* index (used for re-ranking) is also taken into account in Tab. V.

VI. Conclusion

This work addresses the problem of querying large video databases by image. First, we introduced a new comparison



(a) (b) (c)

Figure 14: Retrieval performance as a function of the database size, measured in terms of mAP (left y axis, solid lines) and mR@100 (right y axis, dashed lines). The top-performing BF-PI and Scene FV* schemes are compared on the (a) SI2V-14M, (b) VB-14M, and (c) ClassX-1.5M datasets.

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

Table V: Summarized results for large-scale experiments, comparing the proposed techniques against the frame-based baseline. 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.

technique for Fisher vectors, which handles asymmetry of visual information. The basic idea is to carefully select the types of visual information to use in such comparisons, efficiently ignoring clutter that is typical in this case. Experimental results demonstrate up to 25% mAP improvement for two types of asymmetry.

Next, we introduced two different video descriptors that can be directly compared against image descriptors. These techniques can be seen as high-dimensional embeddings where images and videos are compared. To be useful, these embeddings are of much higher dimensionality than those that are commonly used when querying a database of images using images. We show that different embeddings (e.g., Scene FV or BF-PI) have different associated costs, in terms of retrieval latency and index size.

To construct Scene FVs, we perform a thorough evaluation of FV-based aggregation techniques. Scene FVs achieve excellent performance in the ClassX dataset, where high mAP can be obtained with a very memory-efficient index. The second video descriptor we introduced is constructed using BFs. We developed an aggregation technique where frame-based local descriptors are hashed into BFs – called BF-PI. The proposed techniques were evaluated at large scale, compared against a baseline frame-based method. Scene FVs enable very compact index sizes in all datasets, although with low mAP for some datasets. BF-PI achieves similar retrieval quality as the baseline in all datasets, while using a much smaller index (up to 6×) and reducing query time by up to 9.6×. We also presented a comparison of the proposed descriptors against recent pre-trained CNN features: our technique outperforms such CNN features substantially and consistently, on the three datasets considered in this work.

While the techniques presented here introduce specific methods to embed images and videos in a joint high-dimensional space, future work may focus on learning such embeddings directly from data. With the rise of deep learning techniques and large video datasets, we believe that this is a promising research direction.

REFERENCES

[1] S. Tsai, D. Chen, V. Chandrasekhar, G. Takacs, N.-M. Cheung, R. Vedantham, R. Grzeszczuk, and B. Girod, "Mobile Product Recognition," in *Proc. ACM MM*, 2010.
[2] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang, "Mobile Product Search with Bag of Hash Bits and Boundary Reranking," in *Proc. CVPR*, 2012.
[3] D. Chen, G. Baatz, K. Köser, S. Tsai, R. Vedantham, T. Pylvä, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk, "City-Scale Landmark Identification on Mobile Devices," in *Proc. CVPR*, 2011.
[4] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach, "Mobile Visual Location Recognition," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 77–89, 2011.
[5] Amazon Flow, http://flow.a9.com. Accessed Apr. 2016.
[6] Google Goggles, https://play.google.com/store/apps/details?id=com.google.android.apps.unveil. Accessed Apr. 2016.
[7] A. Araujo, J. Chaves, R. Angst, and B. Girod, "Temporal Aggregation for Large-Scale Query-by-Image Video Retrieval," in *Proc. ICIP*, 2015.
[8] A. Araujo, J. Chaves, H. Lakshman, R. Angst, and B. Girod, "Large-Scale Query-by-Image Video Retrieval Using Bloom Filters," *arXiv*, vol. 1604.07939, 2016.
[9] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Packing and Padding: Coupled Multi-index for Accurate Image Retrieval," in *Proc. CVPR*, 2014.
[10] E. Spyromitros-Xioufis, S. Papadopoulos, I. Y. Kompatsiaris, G. Tsoumakas, and I. Vlahavas, "A Comprehensive Study over VLAD and Product Quantization in Large-Scale Image Retrieval," *IEEE Transactions on Multimedia*, vol. 16, no. 6, 2014.
[11] S. Zhang, M. Yang, T. Cour, K. Yu, and D. Metaxas, "Query Specific Rank Fusion for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, 2015.
[12] L. Zheng, Y. Yang, and Q. Tian, "SIFT Meets CNN: A Decade Survey of Instance Retrieval," *arXiv*, vol. 1608.01807, 2016.
[13] M. Makar, V. Chandrasekhar, S. Tsai, D. Chen, and B. Girod, "Interframe Coding of Feature Descriptors for Mobile Augmented Reality," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3352–3367, 2014.
[14] D. Chen and B. Girod, "A Hybrid Mobile Visual Search System with Compact Global Signatures," *IEEE Transactions on Multimedia*, vol. 17, no. 7, pp. 1019–1030, 2015.
[15] M. Douze, H. Jégou, and C. Schmid, "An Image-Based Approach to Video Copy Detection with Spatio-Temporal Post-Filtering," *IEEE Transactions on Multimedia*, vol. 12, no. 4, pp. 257–266, 2010.
[16] S. Poullot, S. Tsukatani, A. Nguyen, H. Jégou, and S. Satoh, "Temporal Matching Kernel with Explicit Feature Maps," in *Proc. ACM MM*, 2015.
[17] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," in *Proc. ICCV*, 2003.

[18] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object Level Grouping for Video Shots," in *Proc. ECCV*, 2004.

[19] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A. Smeaton, W. Kraaij, and G. Quénot, "TRECVID 2010–An Overview of the Goals, Tasks, Data, Evaluation Mechanisms, and Metrics," in *Proc. TRECVID*, 2010.

[20] 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.

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

[22] A. Araujo, M. Makar, V. Chandrasekhar, D. Chen, S. Tsai, H. Chen, R. Angst, and B. Girod, "Efficient Video Search Using Image Queries," in *Proc. ICIP*, 2014.

[23] 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.

[24] 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.

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

[26] C.-Z. Zhu, H. Jégou, and S. Satoh, "Query-Adaptive Asymmetrical Dissimilarities for Visual Object Retrieval," in *Proc. ICCV*, 2013.

[27] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating Local Image Descriptors into Compact Codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, 2012.

[28] T. S. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers," in *Proc. NIPS*, 1998.

[29] F. Perronnin and C. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization," in *Proc. CVPR*, 2007.

[30] H. Jégou, M. Douze, and C. Schmid, "On the Burstiness of Visual Elements," in *Proc. CVPR*, 2009.

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

[32] R. Arandjelović and A. Zisserman, "All About VLAD," in *Proc. CVPR*, 2013.

[33] L. Duan, J. Lin, J. Chen, T. Huang, and W. Gao, "Compact Descriptors for Visual Search," *IEEE Multimedia*, vol. 21, no. 3, 2014.

[34] D. Chen and B. Girod, "Memory-Efficient Image Databases for Mobile Visual Search," *IEEE Multimedia*, vol. 21, no. 1, pp. 14–23, 2014.

[35] V. Chandrasekhar, D. Chen, S. Tsai, N.-M. Cheung, H. Chen, G. Takacs, Y. Reznik, R. Vedantham, R. Grzeszczuk, J. Bach, and B. Girod, "The Stanford Mobile Visual Search Data Set," in *Proc. ACM MMSys*, 2011.

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

[37] 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.

[38] K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[39] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, 2004.

[40] G. Boccignone, A. Chianese, V. Moscato, and A. Picariello, "Foveated Shot Detection for Video Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 365–377, 2005.

[41] M. Yeung, B.-L. Yeo, and B. Liu, "Segmentation of Video by Clustering and Graph Analysis," *Computer Vision and Image Understanding*, vol. 71, no. 1, 1998.

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

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

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

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

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

[47] 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.

[48] M. Charikar, "Similarity Estimation Techniques from Rounding Algorithms," in *Proc. ACM Symposium on Theory of Computing*, 2002.

[49] M. Henzinger, "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms," in *Proc. ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

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

[51] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality Sensitive Hashing: A Comparison of Hash Function Types and Querying Mechanisms," *Pattern Recognition Letters*, vol. 31, no. 11, 2010.

[52] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural Codes for Image Retrieval," in *Proc. ECCV*, 2014.

[53] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," in *Proc. CVPR Workshops*, 2014.

[54] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Proc. NIPS*, 2012.

[55] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, vol. 1409.1556, 2014.

[56] A. Babenko and V. Lempitsky, "Aggregating Local Deep Features for Image Retrieval," in *Proc. ICCV*, 2015.

[57] 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*, vol. 25, no. 1, pp. 179–194, 2016.

André Araujo is a Software Engineer at Google Inc., Mountain View, California. He received the Ph.D. degree in electrical engineering from Stanford University, California, in 2016. He is a recipient of the Fulbright Science & Technology Scholarship, the Kodak Fellowship and the Accel Innovation Scholarship. His current research interests include computer vision and multimedia systems.

He received B.S. degrees in electrical engineering from the Institut National des Sciences Appliqueés, Lyon, France, in 2007, and from the University of Campinas, Brazil, in 2008. He received the M.S. degree in electrical engineering from the University of Campinas, Brazil, in 2010.

Bernd Girod is the Robert L. and Audrey S. Hancock Professor of Electrical Engineering at Stanford University, California. He received the Engineering Doctorate degree from University of Hannover, Germany, and the M.S. degree from the Georgia Institute of Technology. Until 1999, he was a Professor with the Electrical Engineering Department, University of Erlangen–Nuremberg. He has authored over 600 conference and journal papers and six books, receiving the EURASIP Signal Processing Best Paper Award in 2002, the IEEE Multimedia Communication Best Paper Award in 2007, the EURASIP Image Communication Best Paper Award in 2008, the EURASIP Signal Processing Most Cited Paper Award in 2008, the EURASIP Technical Achievement Award in 2004, and the Technical Achievement Award of the IEEE Signal Processing Society in 2011. His research interests are in the area of image, video, and multimedia systems. As an entrepreneur, he was involved in numerous startup ventures, among them Polycom, Vivo Software, 8x8, and RealNetworks. He is a Fellow of the IEEE, a EURASIP Fellow, a member of the the National Academy of Engineering, and a member of the German National Academy of Sciences (Leopoldina).