EFFICIENT VIDEO SEARCH USING IMAGE QUERIES

A. Araujo[1], *M. Makar*[2*], *V. Chandrasekhar*[3], *D. Chen*[1], *S. Tsai*[1], *H. Chen*[1], *R. Angst*[1] *and B. Girod*[1]

[1] Stanford University, USA [2] Qualcomm Inc., USA [3] Institute for Infocomm Research, Singapore

ABSTRACT

We study the challenges of image-based retrieval when the database consists of videos. This variation of visual search is important for a broad range of applications that require indexing video databases based on their visual contents. We present new solutions to reduce storage requirements, while at the same time improving video search quality. The video database is preprocessed to find different appearances of the same visual elements, and build robust descriptors. Compression algorithms are developed to reduce system's storage requirements. We introduce a dataset of CNN broadcasts and queries that include photos taken with mobile phones and images of objects. Our experiments include pairwise matching and retrieval scenarios. We demonstrate one order of magnitude storage reduction and search quality improvements of up to 12% in mean average precision, compared to a baseline system that does not make use of our techniques.

Index Terms— efficient video search, image-based retrieval

1. INTRODUCTION AND RELATED WORK

Visual search has become a widely studied topic. However, a large body of visual content, namely videos, cannot be searched visually using today's commercial systems. In this work, we explore problems that arise when one **searches videos** using **image queries**, which presents many practical use cases. Users might take a picture of a video to obtain information related to it. In online education, a user might want to find the segment in a lecture video where a particular slide is presented. A company might want to find all appearances of its logo or a particular product in television broadcasts. The fact that today's commercial visual search systems do not index videos is to a large extent a consequence of the enormous redundancy of video when treated simply as a collection of individual frames. We address this issue by introducing a method that reduces system's storage requirements dramatically while boosting video search quality.

Our contributions are the following: (I) A method of finding and combining different appearances of the same visual element, with mathematical justification for both image pairwise matching and image retrieval experiments – deriving more robust, temporally-coherent descriptions. Our method improves video search quality by up to 12% in mean average precision while achieving one order of magnitude storage savings. Also, our method is more memory-efficient than even a simple 10 times coarser video search technique, with mean average precision improvements of 61.4% in this case. These results show that we achieve a better discriminativeness-invariance trade-off for this application. (II) A compression algorithm that reduces significantly the storage requirements for descriptors and that, when combined with other compression techniques, allows for the previously mentioned storage savings. (III) We provide a comparison between several different ways of combining different visual element appearances, with ours ranking among the best. Also, somewhat surprisingly, this comparison shows that keeping a descriptor for only one of the appearances might harm search quality. (IV) We show that tracking keypoints is crucial to obtain better video search quality and further reduced storage needs. Our experiments show that the new descriptions do not work as well if we simply detect keypoints independently for each frame.

Fig. 1: Examples of query images and ground truth database frames from the *CNN2h* dataset, introduced in this work.

Related work. Previous work has explored video search by image. Chen et. al. [1] introduced a technique to find a video from images captured with mobile phones. TRECVID's "Instance Search" task [2] has evaluated systems that find repeated objects, persons or locations in a large video collection [3, 4]. Our method differs from [1, 3, 4] since we explicitly combine different appearances of the same feature to construct a more robust representation – which also makes our database storage very compact.

Sivic and Zisserman illustrated the effectiveness of the Bag-of-Words framework with a movie [5], and enhanced search for regions of video frames with different tracking methods [6]. By using a Temporally Coherent Detector (TCD) [7], and matching frames to join tracks, our approach is reminiscent of [6]'s, in the sense that it uses short- and long-range tracking methods. However, our usage of TCD makes the keypoint tracks more stable due to systematic tracking on the canonical patch level – compared to [6]'s method of tracking only when a keypoint is missing. One of our key findings demonstrates that, when using temporal aggregation, TCD keypoints perform better than keypoints that are detected every frame (as used in [6, 5]). This is shown by superior retrieval and pairwise matching results using a smaller storage footprint. Compared to [8, 6, 5], our dataset is much larger and much more challenging: it contains 139 queries, from photos taken with significant distortions and images of objects collected from the web – compared to [6, 5, 8]'s 6 queries, taken from regions of frames in the same database video.

Sivic and Zisserman [5] argue that averaging descriptors in a track improves signal-to-noise ratio, but without evaluating this idea. We build on top of this idea to introduce a good justification for averaging and demonstrate quantitative improvement in search quality and database storage when using averaged descriptors. Descriptor averaging was also explored by Takacs *et al.* [9] in a landmark recognition application. Our work differs from [9] since we work with video search, use temporal coherence for averaging, justify the use of averaging and compare different aggregation options.

On the compression side, recent work considered compression of descriptors [10, 11, 12], inverted index structures [13] and locations [14]. In this work, we focus on storage gains which arise naturally

*M. Makar performed the work while at Stanford University.

when using the same robust aggregated descriptor for a large number of keypoints in the database video frames. Moreover, we design compression algorithms to encode indices that make for a large portion of storage needs.

2. FINDING AND AGGREGATING VISUAL ELEMENTS

We use the expression 'visual element' to indicate image patches that represent the same part of a specific visual structure in different frames. For example, two image patches corresponding to a specific wheel of a specific car in two different frames represent the same visual element. But two descriptors that represent parts of different visual structures (e.g., a car and a tree) do not represent the same visual element, even if they are close in terms of descriptor distance. Our objective is two-fold. First, to generate a more robust representation of a visual element, by aggregating their appearances throughout the video under different conditions, such as different viewpoints and illumination. Second, to enable search in very large video datasets, by dramatically reducing descriptor storage needs.

We collect different appearances of visual element m in a Temporally Aggregated Patch Set (TAPS) t_m, which we define as a set of patches $\mathbf{p}_i, i \in \{1, ..., n_m\}$, where $n_m = |t_m|$. These patches represent visual element m under different conditions. Each patch is assigned to exactly one TAPS. We represent TAPS t_m by a TAPS descriptor, $\boldsymbol{\phi}_m$, which we define as a function of patches that are assigned to t_m, i.e., $\boldsymbol{\phi}_m = f(t_m)$. A TAPS descriptor is the aggregated representation we use for all patches of the same visual element. In the following, we present how to detect and track visual elements, then show their TAPS descriptors can be computed.

2.1. Keypoint Detection and Tracking

We experiment with two different ways of finding keypoints in video frames. The first approach detects keypoints for each frame independently (ID - Independent Detection), as in [5, 8, 6, 3, 4]. The second option is to use a Temporally Coherent Detector (TCD) [7]. TCD tracks keypoints on the canonical patch level (i.e., it searches for the best location, scale and orientation for the keypoint localization in the subsequent frames). When using ID, keypoints' locations, scales and orientations present significant variations in a single track, due to the sensitivity of the detector. In contrast, TCD generates keypoints whose patches and descriptors are much more similar within the same track [15].

In order to establish correspondences between tracks that are not contiguous in time (due to temporary occlusion, for example), we use a feature-based method for matching images, using a ratio test followed by a geometric consistency check, similar to [16]. This is similar to the long-range tracking approach of [6]. We use a parameter, N_{pm}, to denote the number of frames within which we allow matching operations. In our experiments, we set it to 10 or 50 (equivalent to 1 or 5 seconds, respectively). This image matching procedure is also used with ID mode to find feature tracks, for simplicity. A TAPS is created containing all different patches within a track, for each of the tracks that are found with this method.

2.2. TAPS description

After collecting repeated visual elements into a TAPS, we represent each patch from a TAPS using the same TAPS descriptor. We consider different ways to generate a TAPS descriptor:

Keep One (KO). We keep one of the patches from each TAPS, and use it to compute a descriptor.

Patch Average (PA). Patches are rotated and scaled according to their keypoint. Then, they are averaged and a descriptor is computed from the mean patch. This is equivalent to averaging the gradients of the different patches before descriptor computation.

Minimum Distance (MD). We keep all descriptors of the different patches in a TAPS. To calculate the distance from a query descriptor to a TAPS, we select the minimum distance between the former to each of the descriptors of the latter – i.e., this mode calculates a best case distance for comparison of a query descriptor and a TAPS. Note that in this case we do not reduce descriptor storage requirements, since we still need to keep each of the original descriptors.

Descriptor Average (DA). We extract descriptors from each patch in a TAPS and average them. This follows naturally for pairwise matching and retrieval scenarios, as follows.

Pairwise Matching. Consider matching a query image against a video frame, where we find putative feature matches using the ratio test from [16]. Consider matching the query descriptor $\mathbf{q}$ to the TAPS t. An intuitive way to capture contributions from different patches in a TAPS is to consider the expected square distance between $\mathbf{q}$ and t. We can use Lowe's ratio test [16] in its square version, by using square distances and a square threshold. Note that t is a set of samples from the probability distribution for patches generated from the same visual element. Let $\mathbf{U} \in R^D$ (D being the descriptor dimensionality) be a random vector denoting the descriptor of a patch drawn from the probability distribution sampled by t:

$$E_{\mathbf{U}}[\|\mathbf{q} - \mathbf{U}\|^2] = E_{\mathbf{U}}[\mathbf{q}^T\mathbf{q} - 2\mathbf{q}^T\mathbf{U} + \mathbf{U}^T\mathbf{U}] \quad (1a)$$

$$= \mathbf{q}^T\mathbf{q} - 2\mathbf{q}^T\boldsymbol{\mu} + \sum_i [\sigma_i^2 + E_{U_i}[U_i]^2] \quad (1b)$$

$$= \|\mathbf{q} - \boldsymbol{\mu}\|^2 + \sum_i \sigma_i^2 \quad (1c)$$

where $\boldsymbol{\mu}$ is the mean descriptor from t's patches, σ_i the standard deviation of component i of descriptors of patches from t and $\|\mathbf{x}\|$ is the L_2 norm of $\mathbf{x}$. This computes the dissimilarity between each query descriptor and each video frame's TAPS. Both terms in (1c) can be computed by estimating and storing, for each TAPS, the mean $\boldsymbol{\mu}$ and one other number, $\sum_i \sigma_i^2$, a measure of intra-TAPS variance.

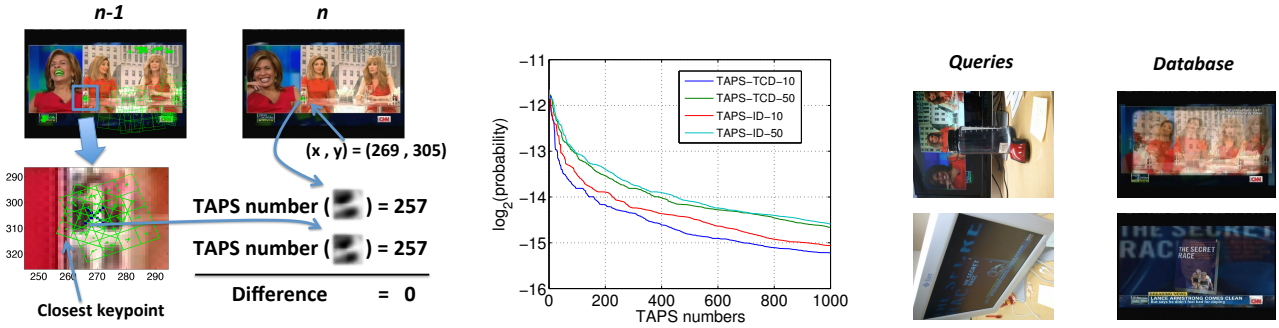
Retrieval. Consider a Bag-of-Words (BoW) model for retrieval, to search a large number of video frames without matching the query to each frame. Consider the construction of a codebook using a vector quantizer. In this case, we want to minimize $\sum_j dist(\mathbf{x}_j, \hat{\mathbf{x}}_j)$, where $\mathbf{x}_j$ represents a descriptor, $\hat{\mathbf{x}}_j$ its reproduction vector and $dist(.)$ being a nonnegative distortion measure. We want to use the average square distance between a TAPS and a reproduction vector. For a dataset with M TAPS's, K reproduction vectors and assignments $\mathbf{c}$ (with $c_m \in \{1, 2, ..., K\}$, the assignment for TAPS m):

$$\sum_m dist(\mathbf{U}_m, \hat{\mathbf{x}}_{c_m}) = \sum_m E_{\mathbf{U}_m}[\|\mathbf{U}_m - \hat{\mathbf{x}}_{c_m}\|^2] \quad (2a)$$

$$= \sum_m \|\boldsymbol{\mu}_m - \hat{\mathbf{x}}_{c_m}\|^2 + \sum_{m,i} \sigma_{mi}^2 \quad (2b)$$

where $\boldsymbol{\mu}_m$ represents the mean descriptor from the m-th TAPS and σ_{mi}^2 the variance of the i-th component of TAPS m. The term $\sum_{m,i} \sigma_{mi}^2$ is independent of both the codebook and the assignments. Thus, a codebook trained using the K-means algorithm with the average of descriptors from the same visual element is equivalent to using the expected square distances introduced in (1c). The assignment of a TAPS to one of the codewords can also be done without regard for the intra-TAPS variance term: it will be a constant factor added to the expected square distance between a TAPS and each centroid. Note that we only include a TAPS for codebook training once, instead of once for each patch in the database. This makes K-means training much faster, since there are usually one order of magnitude fewer TAPS than descriptors. It also allows for redundancy removal that may harm codebook training, and decreases quantization noise by making the different appearances of each visual element be quantized to a single codeword. This is related to the work of [17, 18].

Intra-TAPS variance. In practice, the intra-TAPS variance is small compared to the distance from the query feature to the average descriptor, not affecting matching results significantly (as seen in 4.2). To measure its importance, we collected 4M patch pairs from query images and database frames, and measured $\sum_i \sigma_i^2 / \|\mathbf{q} - \boldsymbol{\mu}\|^2$. Using TCD with $N_{pm} = 50$, the average ratio is 5.3%. In all cases, the intra-TAPS variance term can be discarded, as shown in 4.2.



(a) (b) (c)

Fig. 2: (a) Illustration of the predictive coding algorithm. For each keypoint in frame n, the algorithm finds the spatially closest keypoint in frame $n-1$ and encodes the difference in TAPS numbers. (b) Distribution for the 1,000 most frequent TAPS numbers, for different N_{pm} and detection methods. (c) Example of pairs that are succesfully matched using TAPS but not with frame-based descriptors.

3. REDUCING STORAGE REQUIREMENTS

The main components for storing our retrieval system (described in 4.3) are: retrieval structure and its index, features, and their locations. In a visual search application with a database of videos, the storage cost can quickly become prohibitive. We use the method of [13] to store an inverted index more efficiently. To store keypoints' locations, we use a simple uniform quantizer with step size of 2, followed by Arithmetic Coding [19]. In the rest of this section, we discuss how to store descriptors with as small of a cost as possible.

The main advantage of our approach to reducing storage needs is the fact that we can use a TAPS descriptor to represent multiple database features. By representing a database feature by TAPS t_m, we need to store the TAPS number, m, that indicates which TAPS it corresponds to. We are interested in reducing storage requirements by storing TAPS descriptors and each frame's TAPS numbers, instead of each frame's descriptors. In our dataset, TAPS numbers make for 32% of total descriptor storage, if using 32-bit unsigned integers.

We propose a lossless compression scheme to encode these numbers more efficiently, significantly reducing storage cost. First, note that the distribution of TAPS numbers is far from uniform – Fig. 2(b). This allows for more efficient encoding. We use an Arithmetic Coder to efficiently encode a frame's sequence of TAPS numbers.

In the pairwise matching step of our retrieval pipeline, we use descriptors to rerank the results returned by the retrieval structure. In our case, we use TAPS descriptors, instead of the original descriptors. Thus, we need to efficiently access TAPS numbers in order to know which TAPS descriptors to use. However, if we can afford some delay, we can further reduce storage by the use of predictive coding. We define an Encoding Sequence as a sequence of frames in which all TAPS numbers from a given frame are predicted by TAPS numbers from the previous frame – except for the first frame in the sequence. The Encoding Sequence Size (ESS) is dependent on how much delay one can afford before having access to TAPS numbers of the required frame. The worst-case delay before starting to decode TAPS numbers from the required frame is $T \times (ESS - 1)$, where T is the decoding delay per frame. There is a trade-off between storage requirements and system delay. Note that $ESS = 1$ corresponds to encoding TAPS numbers for each frame individually.

Predictive coding of TAPS numbers is done as follows. For each keypoint in frame n, we find the spatially closest keypoint in frame $n-1$. We then encode the sequence of differences in TAPS numbers with an Arithmetic Coder. This process is illustrated in Fig. 2(a).

4. EXPERIMENTS

We introduce the *CNN2h* dataset[1], composed of 2 hours of CNN video. We provide annotated ground truth query results for 139 queries composed of photos taken with mobile phones and tablets from displays showing the video (with substantial geometric and photometric distortions), along with pictures of objects collected from the web. We provide 2,951 true and 21,412 false ground-truth matching pairs of query images and database frames, creating a pairwise matching experiment. The frames are sampled at 10fps – a total of 72,000 frames. Fig. 1 presents sample queries and some of their corresponding database frames. The new dataset is needed since other datasets [5, 8, 6, 2] are either too small or use queries that are only regions in a video frame (none of them being pictures taken with cameras or clean images of products). Also, [2]'s queries include an indication of its type (object, person or place).

Our experiments use SIFT [16] detector and descriptor, with a budget of 400 features per frame, selected by highest Difference of Gaussian peaks. Using TCD [7], only 7.3% of frames had keypoints detected independently – all others used keypoint tracking. For compression of TAPS numbers, we used $ESS = \{1, 10\}$.

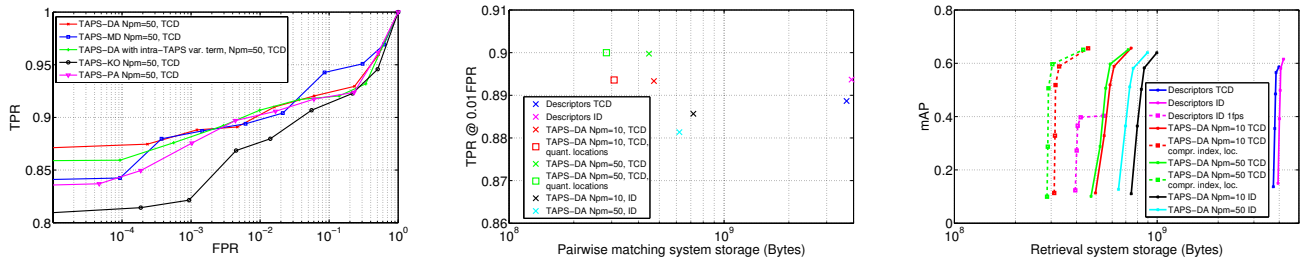
4.1. Tracking and storage

Table 1 presents tracking and descriptor storage results and Fig. 2(b) presents the distribution of the most frequent TAPS numbers. TAPS's tend to capture more patches as N_{pm} increases. The ratio of number of TAPS's to number of descriptors is smallest for higher N_{pm}'s and when using TCD. Significant storage savings are obtained using TAPS, compared to storing all descriptors in the database: using TCD, 93.44% with $N_{pm} = 50$. Note the importance of compressing TAPS numbers: storage is reduced by up to 28.18% compared to storing 4 bytes per number. Decoding delays can become significant with $ESS = 10$, but can be avoided if using $ESS = 1$. Storage needs for different retrieval systems using a Scalable Vocabulary Tree (SVT) [20] with 1M nodes are shown in Table 2. Location coding saves 75% of storage for locations. Inverted index compression obtains 95.37% gains for TAPS with TCD using $N_{pm} = 50$. Overall, we construct a system to index 72,000 frames using 412MB, compared to a baseline of 4,003MB – savings of 89.7%. This is even more memory-efficient than using descriptors extracted at 1fps with ID mode, as shown in Table 2, with savings of 20.8%.

4.2. Pairwise Matching

In this experiment, we match a query image to a video frame using SIFT with ratio test [16] and RANSAC using an affine model. The score is the number of inliers from the estimated model. We draw an ROC curve comparing the different approaches. The important operating points of this curve are between 10^{-2} and 10^{-3} False Positive Rates (FPR) – typical operating points for applications. We compare the different TAPS description modes, using TCD, in Fig. 3(a), for $N_{pm} = 50$. First, note that the KO mode performs substantially worse than all others. This is intuitive since it does not take into account different appearances of the same keypoint. Second, the intra-TAPS variance term does not affect much pairwise match-

[1] It can be accessed at http://purl.stanford.edu/pj408hq3574



(a) (b) (c)

Fig. 3: (a) Comparison of TAPS description modes, using $N_{pm} = 50$ and TCD keypoints. The important operating points are between 10^{-3} and 10^{-2} FPRs. (b) Pairwise matching results: TPR and storage requirements at FPR = 10^{-2}. (c) Retrieval results for different systems compared by storage cost. For each configuration, we vary the SVT's size. Note that the two left-most curves use compressed inverted index and locations.

N_{pm}	Detection mode	Number of descriptors	Number of TAPS's	Total storage - all database descriptors (MB)	Total storage - TAPS with 32-bit TAPS numbers (MB)	Total storage - TAPS with ESS = 1 (MB)	Total storage - TAPS with ESS = 10 (MB)	Worst case delay for ESS = 10 (s)
10	TCD	27,299,281	1,834,880	3332.43	328.12	286.24	241.69	4.59
50	TCD	27,299,281	1,638,880	3332.43	304.20	260.50	218.48	4.77
10	ID	28,793,066	3,566,072	3514.78	545.15	502.11	466.55	9.45
50	ID	28,793,066	2,781,996	3514.78	449.44	403.79	371.32	7.20

Table 1: Tracking and descriptor storage results for different N_{pm}'s and keypoint detection methods. Note the significant storage gain when using TAPS with compression of TAPS numbers, compared to storing all descriptors – approximately one order of magnitude when using TCD. Decoding delays are measured on an Intel Xeon 2.4GHz processor.

Component	Descriptors ID, baseline (MB)	Descriptors ID, compressed inv. index and locations (MB)	Descriptors TCD, baseline (MB)	Descriptors TCD, compressed inv. index and locations (MB)	TAPS-DA TCD, $ESS = 10$, $N_{pm} = 50$ (MB)	TAPS-DA TCD, $ESS = 10$, $N_{pm} = 50$, compressed inv. index and location (MB)	Descriptors ID, baseline, 1 fps (MB)
SVT	135.63	135.63	135.63	135.63	135.63	135.63	135.63
Inv. Index	133.11	24.23	126.22	15.24	123.66	5.73	10.30
Descriptors	3514.78	3514.78	3332.43	3332.43	218.48	218.48	351.48
Locations	219.67	53.94	208.28	51.74	208.28	51.74	21.97
Total	4003.19	3728.56	3802.56	3535.04	686.05	**411.58**	519.38

Table 2: Storage requirements for different retrieval systems using a Scalable Vocabulary Tree with 1M nodes. Our system indexes video at 10fps. It achieves savings of 89.7% with respect to the baseline and is more memory-efficient than using descriptors at 1fps with ID mode (right-most column).

ing performance. Third, the MD mode performs comparably to the DA mode – so we do not need to store all different visual element appearances. Lastly, PA and DA perform on par. We choose to use DA as it is well-justified for both pairwise matching and retrieval experiments, presents best pairwise matching performance and can be stored with much reduced requirements. We plot search quality against storage requirements for the different detection modes and N_{pm}'s, with TAPS using DA description mode, in Fig. 3(b). For this application, no retrieval structures are used, so we only need to store descriptors (or TAPS descriptors and TAPS numbers) and locations. TAPS's allow for 92.76% reduced storage at improved search quality for TCD with $N_{pm} = 50$ and quantized locations. Note that TCD-based TAPS's require lower storage than ID-based ones – as TCD keypoints are more coherent. The reason why the use of TAPS improves search quality is better temporal coverage due to more robust descriptors. Fig. 2(c) shows some pairs for which performance is improved using TAPS – difficult matches, since they are near a shot transition. Matching fails for frame-based descriptors, but the (implicit) temporal constraint imposed by TAPS helps finding more feature matches.

4.3. Retrieval

We employ a BoW retrieval method using an SVT, as in [20]. Two methods are compared: 1) We train an SVT based on all descriptors of the dataset (similar to [6, 8], but with larger and more scalable codebooks), and 2) We train an SVT based on the TAPS descriptors using DA mode. We evaluate retrieval performance for a range of SVT sizes, fixing the SVT's branch factor to 10, and varying the number of levels from 2 to 6. When querying an image, we score database frames using TF-IDF and output the top-ranked ones in a short list that will go through a final pairwise matching stage. Philbin *et al.* [21] showed that there can be significant loss in search quality by quantizing descriptors – for that reason, our retrieval system uses descriptors in a separate pairwise matching stage (using the procedure described in 4.2). We rerank frames from the SVT's short list by the number of inliers after geometric verification. We limit the number of pairwise matching operations to 50, to avoid retrieval delays. Frames that are close in time tend to be ranked close together in the SVT's short list. Pairwise matching frames that are too close in time would be a waste of resources, since frames close in time are very similar, and there is little value in providing results too close in time. We do not perform pairwise matching for a given frame if another frame, within 1 second, has already been pairwise matched. For performance assessment, we use mean average precision (mAP). For each query, we calculate the maximum possible number of matches in the database by discounting the 1 second window, and calculate AP based on the number of correct results in the 50 top-ranked positions after pairwise matching. Fig. 3(c) presents results for the retrieval experiment. Again, we observe improvements by using TAPS due to better temporal coverage, by up to 12% (0.07 mAP). When comparing based on storage costs, the advantage of using TAPS is clear, with a reduction of up to 89.7%. Our system performs significantly better than if just using descriptors with ID mode at 1fps: it presents 61.4% retrieval improvement (0.25 mAP) with 20.8% lower storage cost.

5. CONCLUSION

This work considers video search using image queries. We use temporally coherent keypoints to build better descriptors, in a mathematically justified manner, improving video search quality and reducing storage needs. We show that descriptor averaging is among the best description modes. A new compression algorithm reduces significantly storage cost of indices that make for a large portion of total storage needs. In a retrieval experiment, search quality improves by up to 12% mAP while reducing storage cost by up to 89.7%. Lastly, we show that temporally coherent keypoints are essential to achieve best video search quality at the lowest possible storage cost.

6. REFERENCES

[1] D. Chen, N.-M. Cheung, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod, "Dynamic selection of a feature-rich query frame for mobile video retrieval," in *Proc. ICIP*, 2010.

[2] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, and G. Quenot, " TRECVID 2013 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics," in *Proc. TRECVID*, 2013.

[3] C. Schulze and S. Palacio, "Retrieving Objects, People and Places from a Video Collection: TRECVID'12 Instance Search Task," in *Proc. TRECVID*, 2012.

[4] A. Bursuc, T. Zaharia, O. Martinot, and F. Preteux, "ARTEMIS-UBIMEDIA at TRECVid 2012 : Instance Search Task," in *Proc. TRECVID*, 2012.

[5] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Proc. ICCV*, 2003.

[6] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object Level Grouping for Video Shots," *IJCV*, vol. 67, no. 2, 2006.

[7] M. Makar, S. Tsai, V. Chandrasekhar, D. Chen, and B. Girod, "Interframe Coding of Canonical Patches for Mobile Augmented Reality," in *Proc. ISM*, 2012.

[8] J. Sivic and A. Zisserman, "Video Google: Efficient visual search of videos," *Toward Category-Level Object Recognition*, vol. 4170, 2006.

[9] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *Proc. MIR*, 2008.

[10] R. Ji, J. C. L.-Y. Duan, H. Yao, Y. Rui, S.-F. Chang, and W. Gao, "Towards Low Bit Rate Mobile Visual Search with Multiple-Channel Coding Categories and Subject Descriptors," in *Proc. ACM Multimedia*, 2011.

[11] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "LDAHash: Improved Matching with Smaller Descriptors," *PAMI*, vol. 34, no. 1, 2012.

[12] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, Y. Reznik, R. Grzeszczuk, and B. Girod, "Compressed Histogram of Gradients: A Low-Bitrate Descriptor," *IJCV*, vol. 96, no. 3, 2012.

[13] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod, "Inverted Index Compression for Scalable Image Matching," in *Proc. DCC*, 2010.

[14] S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, J. Singh, and B. Girod, "Location Coding for Mobile Image Retrieval," in *Information Systems Journal*, 2009.

[15] M. Makar, *Interframe Compression of Visual Feature Descriptors for Mobile Augmented Reality*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, 2013.

[16] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, vol. 60, no. 2, Nov. 2004.

[17] F. Jurie and W. Triggs, "Creating Efficient Codebooks for Visual Recognition," in *Proc. ICCV*, 2005.

[18] R. Ji, H. Yao, X. Xie, and Q. Tian, "Vocabulary Hierarchy Optimization and Transfer for Scalable Image Search," *IEEE Multimedia*, vol. 18, no. 3, 2011.

[19] A. Moffat, R. Neal, and I. Witten, "Arithmetic coding revisited," *ACM Trans. on Inf. Syst.*, vol. 16, no. 3, 1998.

[20] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," in *Proc. CVPR*, 2006.

[21] J. Philbin, M. Isard, J. Sivic, and A. Zisserman, "Descriptor learning for efficient retrieval," in *Proc. ECCV*, 2010.