SlideSearch: Slide-to-Lecture-Video Search System on ClassX Database

Mengyuan Yan
School of Engineering, Stanford University
Stanford, CA, 94305
mengyuan@stanford.edu

Abstract

A slide-to-lecture-video search system is implemented on Android tablet. The search system gets video clips from the SCPD database. Previous research on Fisher Vector and Residual Enhanced Visual Vector are combined to generate compact index for each database image. An augmented database of clean slide images is used to exploit the temporal redundancy of videos. Original database of video keyframes and augmented database of slides are compared and experiments show using augmented database can achieve significant memory reduction and precision increase.

1. Introduction

As the usage of computers and the internet penetrates deeper into peoples' lives, universities are following the trend offering their courses online. Platforms like Coursera and Udacity are offering large amounts of courses from universities as well as great tech companies. Stanford University also has its own online lecture video database called ClassX. It is developed by Stanford Center for Professional Development and it achieves lecture videos from classes in the school of engineering. The problem in these databases is that it is hard to search for the time when professor was talking about a particular concept, equation, etc. This forces people to scan through the entire video to find information they need, and makes online learning less efficient. To solve this problem, I developed a way to search for video chunks by taking a photo of a slide using mobile devices. Given photo of a slide, whether printed or displayed on a screen, the application leads user to the webpage which contains the matching video and informs user the time stamp to start watching.

The mobile slide-to-lecture-video search system adds to the mobile augmented reality family. Applications like google goggles have been able to recognize landmarks, paintings, common products, barcodes, etc. They are helping people with entertainment, e-commerse and other areas in their daily lives. The slide-to-lecture-video search system can help in the field of education. As a matter of fact, many students in universities are still using printed slides. Therefore, taking a picture of a slide is a convenient way to search for related information. Even if slides are viewed on computer, it is still more convenient to take a picture of the screen than using screen shot and copy it to a web-based application.

2. Related Works

The image-to-video search problem has been addressed by Sivic and Zisserman [7] a decade ago. A video is represented by a sequence of keyframes at specific frame rate. Therefore, image-to-video search is built on content based image retrieval. Bag-of-Words model [6] and Vocabulary Tree [4] are widely used for this task today. For large-scale image retrieval, it is important to build compact index for database images and reduce memory usage. It is desirable to fit the database indices in the RAM of a mobile device, so that users can process queries on their own device instead of sending their query to the server. This lowers the requirement for a strong server, reduces latency due to network data transmission, and protects user privacy. Recent progress in this direction includes the Vector of Locally Aggregated Descriptors (VLAD) [3], the Compressed Fisher Vector (CFV) [5], and the Residual Enhanced Visual Vector (REVV) [2]. In this project, I combined the thoughts in CFV and REVV to build compact indices for database images. Both work are based on the notion of visual words. Different from the Bag-of-Words and Vocabulary Tree which use 10k to 1M visual words, CFV and REVV schemes only use a few hundred to a few thousand visual words. Instead of building a histogram of words, the vector residual between local descriptors and visual words are aggregated to form compact signature for images. The CFV applied Fisher Kernel to image retrieval problem, and model local descriptors as i.i.d. random vectors generated from a Gaussian Mixture Model. The REVV uses mean aggregation for vector residuals, and exploit dimension reduction techniques like Linear Discriminate

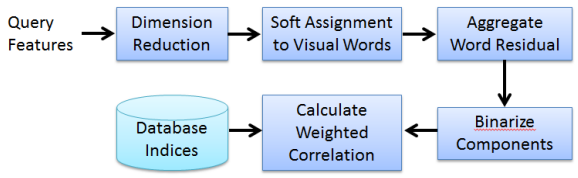

Figure 1: figure 1. Pipeline for generating a compact index for query image and compare it against database indices.

Analysis. Both CFV and REVV use binarization to achieve compact visual vectors to enable large-scale mobile visual search.

An important feature for videos is their temporal redundancy. Much progress is made in the direction to exploit the temporal redundancy of videos to aggregate local features and compress database files [1]. In the case of lecture videos, an easy way to exploit the temporal redundancy is to introduce an augmented database consisting of clean lecture slide images converted from pdf or powerpoint files. With the help of augmented database, the slide-to-lecture-video search becomes a two-step process: match the photo of a slide against the augmented database images, and link the database image to its corresponding video and time stamp with a preprocessed annotation file. This method reduces the memory requirement by a significant amount, and boosts retrieval precision.

3. Design of compact index for database image

Figure 1 shows the pipeline for generating a compact index for query image and comparing it against database indices. I will explain each block in detail in the following subsections.

3.1. Dimension Reduction

During codebook training, covariance matrix is calculated from all SIFT descriptors extracted from training images. Each descriptor is then reduced to a 32 dimension vector using Principle Component Analysis. When building database image indices and query image index, the same dimension reduction is applied to every descriptor. This dimension reduction would speed up the training process, and would help reduce the size of database image indices.

3.2. Word Residual Aggregation

Let us denote the set of descriptors extracted from a image as $X = \{x_t, t = 1...T\}$. T is the number of descriptors, with a typical value of a few hundreds or thousands. We assume that the x_t's are i.i.d random vectors generated from distribution p. The Fisher Kernel framework assumes that p is a Gaussian mixture model (GMM): $p(x) = \sum_{i=1}^{N} w_i p_i(x)$. Each Gaussian p_i can be viewed as a visual word and N is the vocabulary size. w_i is the mixture weight which reflects the percentage of descriptors that are quantized to this visual word. μ_i is the mean vector of the Gaussian p_i, which can be viewed as the word centroid. Σ_i is the covariance matrix of the Gaussian p_i. It is assumed that the covariance matrices are diagonal and we denote by σ_i^2 the variance vector. The GMM p is trained on a large number of images using Maximum Likelihood Estimation (MLE).

After the GMM is obtained from training, we want to build indices for database images. Each descriptor x_t is first soft assigned to visual words. $\gamma_t(i)$ is the probability that x_t belongs to Gaussian p_i.

$$\gamma_t(i) = \frac{w_i p_i(x_t)}{\sum_{j=1}^{N} w_j p_j(x_t)} \tag{1}$$

Word residuals $x_t - \mu_i$ for one visual word p_i are aggregated in the following manner:

$$F_i^X = \frac{1}{\sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(i) \frac{x_t - \mu_i}{\sigma_i} \tag{2}$$

Here the vector division $(x_t - \mu_i)/\sigma_i$ is calculated elementwise. The aggregated vector for each visual word is then concatenated and L2 normalized to form the index for image. This aggregation discounts the contribution of descriptors which are close to word centroids (with respect to its variance) thus have small $(x_t - \mu_i)/\sigma_i$, or are soft assigned with high probability to a visual word with large weight w_i. Both above cases indicate that the descriptor is frequent in training dataset, therefore its contribution to discriminating between images is small. This weighting is similar to the tf-idf weighting in text retrieval, where frequent words have small contributions towards representing the document.

3.3. Power law

In order to reduce the influence of large values in the vector before L2 normalization, power law should be applied elementwise to suppress large values. Power value $\alpha = 0.5$ is used from heuristics.

3.4. Binarization

The index for image constructed so far is high-dimensional and dense. It would consume much memory, and computing L2 distance between indices would be slow. Therefore, it is helpful to binarize the index elementwise to 1 or -1 according to their sign. As the Fisher Vector for each visual word is 32 dimensional, it can be packed to a 32-bit unsigned integer.

3.5. Calculate Weighted Correlation

To calculate the similarity between a pair of indices, we calculate the correlation:

$$\frac{1}{N_q N_d} \sum_{i \subset I_q \bigcap I_d} C(S_{q,i}, S_{d,i}) \quad (3)$$

Here $S_{q,i}$ and $S_{d,i}$ are the binarized Fisher Vector at the i th visual word for query image and database image, respectively. $C(S_{q,i}, S_{d,i}) = d_{PCA} - 2H(S_{q,i}, S_{d,i})$ and H is the Hamming distance which can be quickly computed with bitwise XOR and POPCNT. I_q and I_d are the sets of visual words that query image and database image has visited, respectively. $N_q = \sqrt{d_{PCA}|I_q|}$, $N_d = \sqrt{d_{PCA}|I_d|}$ are normalization factors. To make the correlation more discriminatory between matching and non-matching image pairs, We apply a weight to the correlation for each visual word. The weight would highlight large correlation, because larger correlation indicates higher possibility of a match. We define the weight $w(C) = P(match|C)$, which can be calculated by training with annotated matching and non-matching image pairs. The similarity score changes to

$$\frac{1}{N_q N_d} \sum_{i \subset I_q \bigcap I_d} w(C(S_{q,i}, S_{d,i}))C(S_{q,i}, S_{d,i}) \quad (4)$$

4. Mobile search system implementation

Figure 2 shows the pipeline for the practical mobile slide-to-lecture-video search system. The codebook training and database index generation are done on the server. The matches from clean slides to video information (url and time stamp) are manually annotated for 233 slides for demonstration. It could also be obtained by pairwise matching clean slides and video keyframes on the server given enough preprocessing time. The trained codebook, database index file, annotation file and all database image feature files are downloaded to the tablet. The query is processed on tablet, and the application would open a browser to show the returned matching video.

4.1. Training

The training dataset consists of 7304 images. Some sample images from the training dataset is shown in figure 3. They are natural images of buildings, people, natural view, etc. 11642 matching pairs are annotated as well as 11642 non-matching pairs. The most stable 300 SIFT features are extracted from each image to ensure that codebook training is not degraded by noisy features. 512 visual words are trained on this dataset. The trained codebook data is 330KB in size.

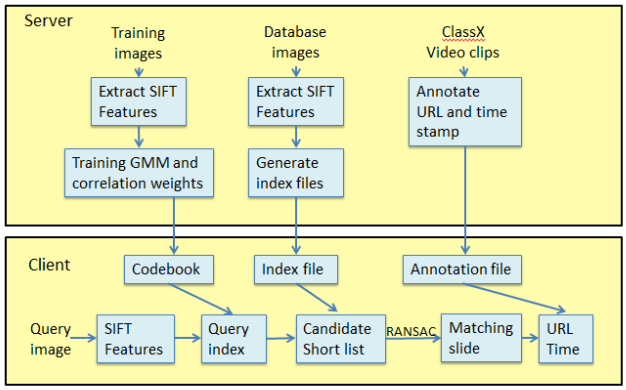

Figure 2: Pipeline for the mobile slide-to-lecture-video search system

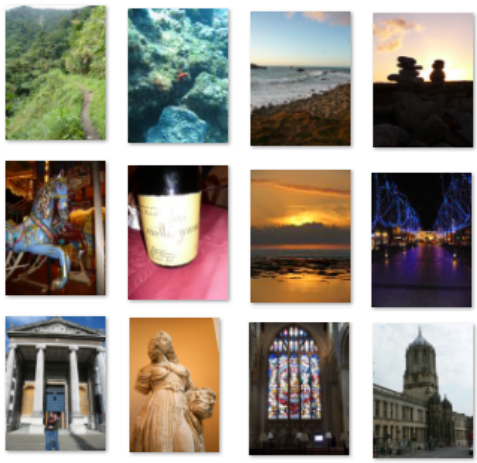

Figure 3: Sample images from the training set.

4.2. Database

The augmented database consists of 955 slide images that are from CS155 class in spring quarter 2013-14. The images are directly converted from pdf files. Some sample images are shown in figure 4. Aside from text contents, the slides also contain figures, flowcharts and images. The index file for 955 images is 2.4MB in size.

4.3. On Device Processing

An Android application is developed and tested on NVIDIA SHIELD tablet. Upon starting the application, codebook data and database index file are loaded into memory. The openCV cameraview handler would manage and render preview frames from the device's camera. The user could then move the tablet and fill the cameraview with a slide that he would like to search. When the user touches the

Figure 4: Sample images from the database.

screen, openCV cameraview handler would pass the preview frame from Java coded front-end to C++ coded native functions.

Because the NVIDIA version of opencv2.4.8 do not have impleamentation for SIFT/SURF, source code from local version of opencv2.4.9 is included to extract SIFT features from the input image. After that, index for the query image is generated and compared to database indices to return a ranked list of candidate matches. Top ten matching database images are then compared to the query image using RANSAC. L1 norm is used instead of L2 norm to match SIFT features in order to speed up computation. Related functions are from NVIDIA tegra-optimized opencv2.4.8 library. Finally, the database image with the highest number of inliers is regarded as the correct match, and its corresponding video URL and time stamp is retrieved from the annotation file. These information are returned back to the Java front-end. Since I haven't figure out a way to control video player embedded in the webpage, I choose to show the video time stamp in a dialog popped from the application, and then open the web browser to show the matching video clip. The user can move the time bar to the time indicated by previous dialog.

Several improvement could be done to optimize the performance on mobile device in terms of speed. First, GPU-computed SURF or ORB feature could be used to speed up local feature extraction. However, bacause text patches are extremely repetitive compared to natural images, test should be done to show that these features could be used to discriminate text images. Due to time constraint of the project I didn't explore along this path. Second, FLANN-based feature matching can be used to speed up the geometric verification step. However, FLANN-based feature matching does not have built-in support for cross-check which excludes many-to-one matches. Cross-check is proven to be crucial for reliable geometric verification for text images from previous experiments. Geometric verification failed to find the matching slide from other candidates when only ratio test is used but cross-check is unabled. Therefore, FLANN-based feature matching is not adopted in the mobile application. Third, short list returned for geometric verification can be shorter to save time on geometric verification. This is at the cost of losing retrieval precision. Considering the speed-precision trade-off, a short list of 10 image is chosen. This leads to 95% precision with geometric verification time 4.4s.

5. Experimental Results

5.1. The Mobile Slide-to-Lecture-Video Search Interface

Screen shots of the mobile application interface is shown in figure 5. On starting the application, camera preview frames are shown on the screen. The user can focus the camera on the slide he would like to search and tap on the screen. The application then takes the preview frame and search against the database. When result is returned, a dialog is popped up to show the time stamp of the matching video chunk. When the user clicks on "Got it" botton, the application would start a web browser to open the ClassX webpage containing the matching video. Stanford network login might be required to access ClassX videos.

The mobile application takes on average 8.3 seconds to perform a query. Among the 8 seconds, 0.24 second is spent on loading the codebook and database indices. 2.42 seconds are spent on extracting SIFT features from the input frame. The input frame is resized to have a width of 1000 pixel in order to limit the time of feature extraction. 1.32 second is spent on generating index for the input frame and comparing it to database indices. Finally, 4.4 seconds are used to perform geometric verification on 10 candidate database images. The codebook data used is 330KB and the database index file is 2.4MB. 880MB feature files for database images are stored on the sdcard, and about 1MB would be loaded for each database image involved in geometric verification. Using adb tools, the PSS total memory consumption is 26.75MB.

The accuracy of this searching application is very high. However, the system would fail if auto-focusing failed to capture a well-focused frame, or if the input frame has a lot of background clutter.

5.2. Effectiveness of Augmented Database

The following sections measure retrieval precision under various conditions using a Linux server instead of a mobile device in order to have repeatable systematic experiments. In this section, we compare the retrieval performance of searching a photo of a slide against the original database consisting of video keyframes, versus searching against the augmented database consisting of clean slide images. The query set is made up of 98 photos of slides, half of them are taken for printed slides and the other half are taken for

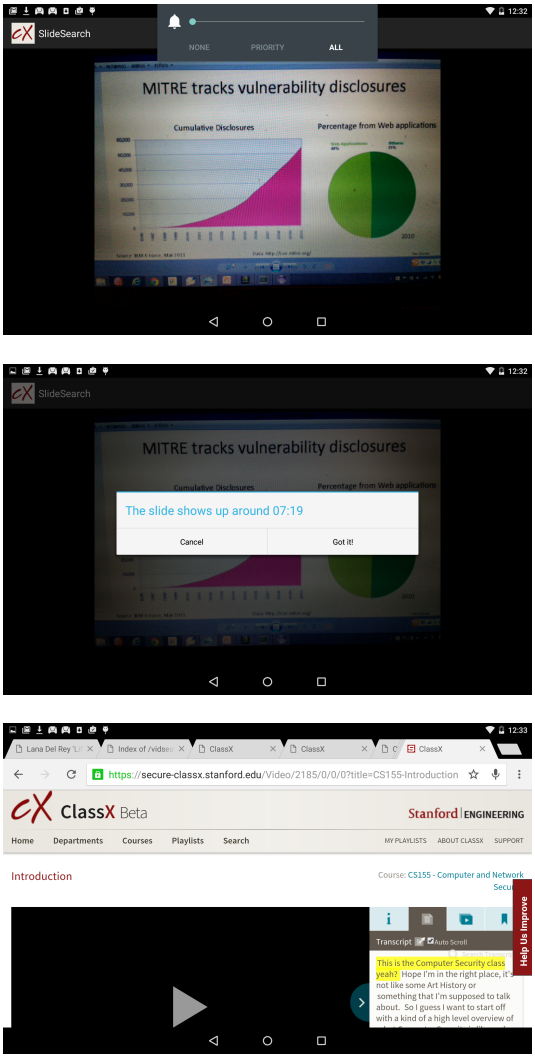

Figure 5: Screen shots of mobile slide-to-lecture-video search application.

slides shown on computer screen. Some photos have rotation and perspective changes, but they don't contain background clutter. A codebook of 512 visual words is used for both experiments. In the first experiment, database images are video keyframes extracted at 1 frame per second. The database has a total of 88720 images. After comparing query index with database indices, the top 300 keyframes are passed to RANSAC, and reranked by the number of inliers. The retrieval is successful if the top keyframe is within the correct video clip. Precision at one in this case is only 18%. In the second experiment, database images are 955 clean slide images. After comparing query index with database indices, the top 4 slide images are passed to RANSAC, and reranked by the number of inliers. The retrieval is successful if the top slide is the one shown in the photo. We chose short list size 4 in order to have similar

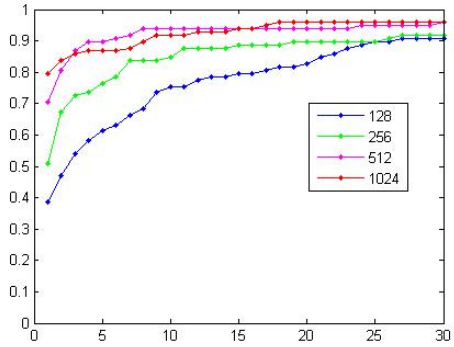

Figure 6: Precision at one for varying number of visual words and varying short list size.

ratio between short list size and database size. The precision at one is boosted to 90%. In addition, database index file is only 2.4MB compared to the 227MB index file for original database. It is a 100X memory saving.

5.3. Varying size of codebook

In this section, we compare the retrieval performance with different sizes of codebooks. The query set is made up of 98 photos of slides as mentioned in the previous section. Codebook with 128,256,512, and 1024 visual words are experimented. The augmented database of slide images is used and precision at one is calculated for each case with varying short list size. The result is shown in figure 6. From this figure, it is obvious that as the short list size increase, precision would increase. This is not surprising since we are using geometric verification to rerank candidate matches. Because geometric verification would almost certainly find the true match if it is within the short list, the possibility of finding the true match would increase with growing short list. Comparing the retrieval performance for different number of visual words, a larger vocabulary would lead to higher precision, especially when the short list size is small. However the performance tend to saturate when the number of visual words increases from 512 to 1024. This could be because the database is rather small. Considering the fact that database index file size and search time would grow linearly with the number of visual words, there is a speed/time-precision trade off in choosing a proper codebook size. Since our database is small, memory and speed are not important problems for our mobile application, we choose to use 512 visual words in our mobile search system.

5.4. Robustness under Background Clutter

I created another query set which has 30 images of slides with various background clutter. Types of clutter include other slides, other non-slide text, and non-text objects. Samples of query images can be found on the left side of figure

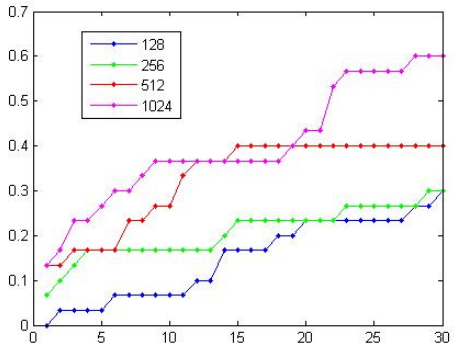

Figure 7: Precision at one for test images with background clutter.

8. The system is tested on this dataset in the same way as described in the previous section. Precision at one is calculated for various vocabulary size and short list size. The result is shown in figure 7.

Comparing figure 7 with figure 6, the precision has dropped significantly. For short list smaller than 5 images, the precision has dropped to below 30%. Analyzing the result for different types of clutter, the syetem is more robust to non-text clutter than text clutter, and there is no obvious difference between slide text clutter and non-slide text clutter. A possible reason for the precision decrease is that descriptors extracted from clutter text get mixed up with descriptors from our region of interest during residual aggregation, and makes the index less like the index of original slide. descriptors from non text clutter are less likely to be assigned to the same visual word as descriptors from text patches, thus would have less influence on the index. Some trial into solving this problem of background clutter is discussed in section 5.

6. Discussion and Future Work

As mentioned in the previous section, the mobile slide-to-lecture-video search system is not very robust for background clutter, especially text clutter. Investigation into this direction is limited by the time of this project. As a first step, I am able to segment the printed paper or the computer screen from other non-text background by detecting white pixels in the input photo. The algorithm first converts color image into grayscale image by taking the minimum of three color channel values. Taking the minimum puts penalty on object that are bright but have hue. Then the grayscale image is sent to detect MSER or to binarization using threshold. Some results using threshold binarization are shown in figure 8.

More processing is needed to segment the slide region we want out of other texts. This can be done by first finding

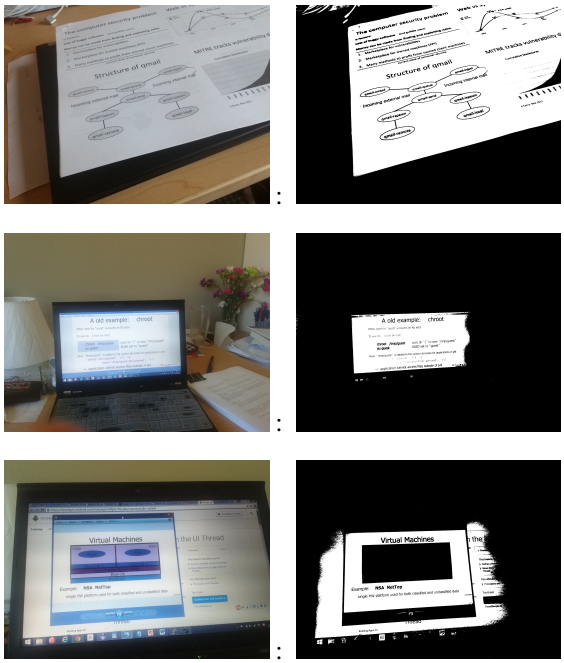

Figure 8: Segmentation of paper or computer screen from input photos.

the direction of text lines by Hough transform and rectifying the image, and then projecting the binary image to horizontal and vertical direction to find blank margins around the slide. Due to limitation of time this is not implemented well to achieve satisfying results.

Test is also done in taking photos of only part of a slide. Preliminary results show that the system could retrieve the right video when non-trivial amount of slide is shown (more than two lines of text). Systematic test can be done to investigate the robustness and limitation in matching parts of a slide to lecture videos.

7. Conclusion

In this project, I have combined the previous research on Fisher Vectors and Residual Enhanced Visual Vector to generate compact indices for database images to achieve efficient slide-to-lecture-video search. In order to exploit the temporal redundancy of videos, an augmented database of clean slide images is used instead of the database of video keyframes. Using augmented database has led to 100X memory reduction and has boosted the retrieval precision from 18% to 90%. An Android Application is implemented and tested on NVIDIA SHIELD tablet.

8. Acknowledgement

Special thanks to Andre Araujo for helpful discussions during this project and for sharing part of the code to build compact index for images. In addition, thanks should be given to Huizhong Chen for helpful suggestions on segmenting slides and analyzing layouts.

References

[1] 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.

[2] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual enhanced visual vector as a compact signature for mobile visual search. *Signal Processing*, 93(8):2316 – 2327, 2013. Indexing of Large-Scale Multimedia Signals.

[3] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311, June 2010.

[4] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168, 2006.

[5] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391, June 2010.

[6] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477 vol.2, Oct 2003.

[7] J. Sivic and A. Zisserman. Video google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin Heidelberg, 2006.