SCALABLE ACTION LOCALIZATION WITH KERNEL-SPACE HASHING

Andrei Stoian[⋆] *Marin Ferecatu*[⋆] *Jenny Benois-Pineau*[†] *Michel Crucianu*[⋆]

[⋆]CEDRIC-Cnam, 292 Rue St. Martin, Paris, France
[†]LABRI, Cours de la Libération, Bordeaux, France

ABSTRACT

To detect and locate complex human actions in video, one trains a detector for each target class and applies it to the video content. This approach can scale to large video databases if the application of the detector can be made *sublinear* in the size of the database. Sublinear retrieval methods have been successfully explored for query-by-example but few were devised for these more challenging *queries by detector*. We put forward here a novel approximate search method that relies on LSH to support query-by-detector. We evaluate our method on a recent large action localization dataset and show it has significantly better efficiency than linear search.

Index Terms— Action localization, query-by-detector, LSH, hyperplane queries, kernel indexing

1. INTRODUCTION

Large video databases contain many action classes of potential interest. For example, actions are very relevant content items in historical and cultural videos; a historian, a researcher or some other user should be able to define action classes and search for occurrences of such actions in a large cultural database. In a video-surveillance application, an investigator may also have to define new specific action classes and search for their occurrences in a potentially high volume of video records. In such scenarios, it can be prohibitively time consuming or excessively expensive to perform a new exhaustive scan of the entire database every time a new class detector is built. It is then necessary to devise methods supporting the *scalable* application of a detector to the data, i.e. methods that are sublinear in the size of the database. Such methods should avoid an exhaustive scan and only apply the detector to a (hopefully) small part of the data, where the detector is relatively likely to provide a positive answer. While the scalability of query-by-example was thoroughly considered in the literature, there is comparatively little work on the scalability of what we shall call *query-by-detector*. In this work we address this problem by proposing a method that supports sublinear retrieval of complex human actions.

Action detection and localization was addressed in the recent literature (see e.g. [1, 2, 3, 4]), but not on large scale datasets. Actions are usually represented by the statistical distribution of local features that describe shape and motion in video patches. In [1, 2], Bags of Visual Words (BoVW) histograms are used to model the statistics of features in video segments, while [3, 4] employ instead Fisher Vectors. Both approaches involve high dimensional vector descriptions. Support vector machine (SVM) classifiers are learned from the annotated examples and detection is performed by applying the SVM to a window that slides over the entire video database. Thus, all the windows in the database are evaluated by the detector and the positive detections are eventually ranked to produce the result list.

There exist more efficient alternatives for applying a detector to a database. A method for approximating a linear SVM decision function using locality-sensitive hashing (LSH [5]) is introduced in [6]. The normal vector to the SVM hyperplane is used as a query and its hash is obtained. The approximation to the decision function employs the Hamming distance between the query hash and the hashes of the non-empty buckets. The complexity is linear in the size of the database but the approximation allows to significantly accelerate the retrieval of the data points classified as positive by the detector.

To best deal with large databases, exhaustive detection (i.e. sliding the detection window over the entire video database) should nevertheless be avoided. Several proposals focus on sublinear methods that aim to find the data points whose image in feature space is close to the normal vector to the SVM hyperplane. Since for most kernels employed $\mathcal{K}(x, x) = \text{constant}$, these points maximize the SVM decision function. The KDX index structure in [7] defines rings around the normal vector and indexes them according to the angle to the central vector. A second level of the index is used within each ring. A sublinear exact search solution for such SVM-based queries was proposed in [8] and claimed to improve over KDX. The data is clustered in feature space and, for each cluster, rings are built with the kernel space neighbors of the cluster prototypes, by order of their distance. Querying this index structure with the normal vector requires testing all prototypes and then the corresponding cluster rings close to the query in order to accumulate the results. However, as we shall see later, the relevant data to be retrieved is not necessarily close to the normal vector in feature space. We can also mention here the method in [9] for exact sublinear retrieval with hyperplane queries, shown to be efficient but in rather low-dimensional spaces.

Alternatively, SVM-based active learning using an ambiguousness criterion requires retrieval of unlabeled data that is close to the decision boundary (which is a hyperplane in feature space). If the amount of unlabeled data is large, sublinear retrieval methods are needed. Several such solutions, based on LSH, are proposed in [10, 11, 12]. They rely on the fact that data points that are close to the decision hyperplane have a low inner product with the normal vector to the hyperplane. Such approaches can be interesting if the target class is defined online, e.g. with relevance feedback, but this is not the scenario we consider here.

In Section 2 we first show how to adapt the approach in [6] to create a linear time "exhaustive approximate" (EA) search method for queries that are nonlinear SVM detectors. We then introduce a novel approximate sublinear method for answering such queries. In Section 3 we provide the experimental validation of our sublinear method and compare it to both exhaustive approximate and exhaustive exact (EE) search.

This work is supported by the French National Research Agency grant ANR-11-IS02-001 within the joint French-Mexican project Mex-Culture. We are grateful to the Institut National de l'Audiovisuel for providing the video database.

2. PROPOSED APPROACH

We start by briefly presenting the general action detection and localization method we employ, then we focus on the scalability issue where our contributions are.

2.1. Action localization method

Video description. We employ the method in [13] to detect and describe salient video patches. We quantize these descriptors into a visual dictionary of $d = 4000$ words and compute an L1 normalized BoVW histogram for each frame. The videos are over-segmented into a series of time windows of $L = 30$ frames, shifted by 5 frames. For each window, the frame histograms are averaged to produce its final description.

Action detection in sliding windows. For detection, an SVM classifier is trained for each action class on annotated examples. The SVM decision function used to score results is

$$f(v) = \sum_{j=0}^{p} \beta_j \mathcal{K}(y_j, v) + b_q \tag{1}$$

Since each window is described by a BoVW histogram, we use the histogram intersection (HI) kernel:

$$\mathcal{K}(x, y) = \sum_{i=1}^{d} \min\{x_i, y_i\} \tag{2}$$

Note that $\langle \Phi(x), \Phi(x) \rangle = \mathcal{K}(x, x) = 1$. All the windows in the database are evaluated by f and the positive detections are retained for further processing. We use this "exhaustive exact search" (EE search) as a baseline. Its complexity is linear in the size of the database.

Post-processing. Multiple overlapping windows can have positive detection scores. To obtain the final detection boundaries, all positive windows whose overlap is above a threshold $\tau_{merge} = 50\%$ are merged by using the union of their bounds. The resulting *detection window* A is assigned the sum of scores of the composing windows: $S(A) = \sum_i f(v_i)$.

2.2. Exhaustive approximate search

We first adapt the method in [6] to kernel space hashing using the Random Maximum Margin hashing (RMMH) functions proposed in [14]. With RMMH, LSH functions are hyperplanes in the feature space. To construct each hyperplane, M data points are chosen at random and randomly labeled as positive or negative examples. An SVM is learned from these examples and its decision function is one atomic hash function used to build the index:

$$h(v) = \text{sgn}\left(\sum_{i=0}^{m} \alpha_i^* \mathcal{K}(x_i^*, v) + b \right) \tag{3}$$

where x_i^* are the m support vectors of one hash SVM, α_i^* their Lagrangian multipliers and $\mathcal{K}(x, y)$ is the employed kernel. A hash table is built from D such atomic functions obtained on independent random samples of M points. To retrieve nearest neighbors, the query q (in input space) is hashed and its hash $H(q)$ is computed as the concatenation of the atomic hash values: $\{h_1(x), ..., h_D(x)\}$. The probability that the hash value of data v is equal to that of the query q is proportional to the inner product between $\Phi(v)$ and $\Phi(q)$ [15]:

$$Pr\left[h(q) = h(v)\right] = 1 - \frac{1}{\pi} \cos^{-1}\left(\frac{\Phi(q) \cdot \Phi(v)}{||\Phi(q)||\,||\Phi(v)||} \right) \tag{4}$$

We build D RMMH functions with the HI-kernel and compute the hash of each data v_i, $i \in \{1..N\}$, obtaining binary vectors $H(v_i)$. However, in our case the query corresponds to the SVM detector in Eq. (1) and is represented by the normal vector q of the learned hyperplane in *feature* space. This vector is a linear combination of the feature space images of the learning examples, $q = \sum_{j=1}^{p} \beta_j \Phi(y_j)$, so its discrete hash value $h(q)$ can be computed according to:

$$\begin{aligned} g(q) &= \sum_{i=0}^{m} \alpha_i^* \left\langle \Phi(x_i^*), \sum_{j=1}^{p} \beta_j \Phi(y_j) \right\rangle + b \\ &= \sum_{i=0}^{m} \alpha_i^* \sum_{j=1}^{p} \langle \Phi(x_i^*), \beta_j \Phi(y_j) \rangle + b \\ &= \sum_{i=0}^{m} \alpha_i^* \sum_{j=1}^{p} \beta_j \mathcal{K}\left(x_i^*, y_j\right) + b \\ &= \vec{\alpha^*} K \vec{\beta}^T + b \\ h(q) &= \text{sgn}\left(g(q)\right) = \text{sgn}\left(\vec{\alpha^*} K \vec{\beta}^T + b \right) \end{aligned} \tag{5}$$

where $K_{ij} = \mathcal{K}\left(x_i^*, y_j\right)$, x_i^* ($i \in \{1..m\}$) are the support vectors of the RMMH function, y_j ($j \in \{1..p\}$) are the support vectors of the action detector, $\vec{\alpha^*} = (\alpha_1^*, ..., \alpha_m^*)$ and $\vec{\beta} = (\beta_1, ..., \beta_p)$.

We take from [6] the approximate decision value of the detector, $\hat{f}_q(v)$, as a function of the Hamming distance between the hashes of the query and of the data point, $d_H(H(q), H(v))$:

$$\hat{f}_q(v) = \cos\left(\pi \frac{d_H(H(q), H(v))}{D} \right) ||q|| + b_q \tag{6}$$

We do not need to compute $\hat{f}_q(v)$ for any v, we just sort the buckets in ascending order of their Hamming distance to the query. All data points in the top buckets are retrieved and evaluated by the decision function in Eq. (1). As many buckets are returned as necessary to reach the desired recall. Note that $\text{sgn}(\hat{f}_q(v))$ does not give accurate positive detections, the estimator can have a large probability of false positives.

2.3. Scalable retrieval

The method introduced in Section 2.2 can be efficient enough for medium-size video databases but requires the evaluation of the Hamming distance between the hash of the query and every bucket. For very large databases, the number of buckets should increase linearly with the size of the database. So the complexity of this method is linear in the size of the database. To scale to very large databases it is then necessary to devise sublinear methods.

If the feature-space images of the data points for which f (the decision function of the SVM detector) takes positive values were in the neighborhood of the normal vector to the SVM hyperplane, then a potentially good solution would be to use a method like Multi-probe LSH [16]. Multi-probe LSH proposes to search those buckets whose spatial boundaries are close to $g(q)$. Assuming the hash functions are independent, each $h(q)$ value can be flipped to produce hashes that have lower probability than the query's hash. In [16] a

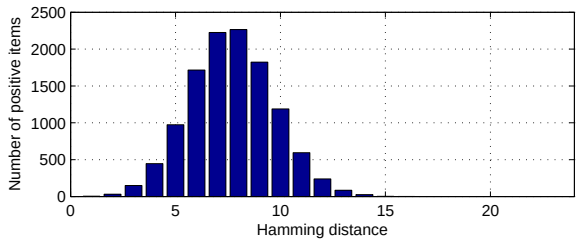

Fig. 1. Positive results are far from query's hash bucket

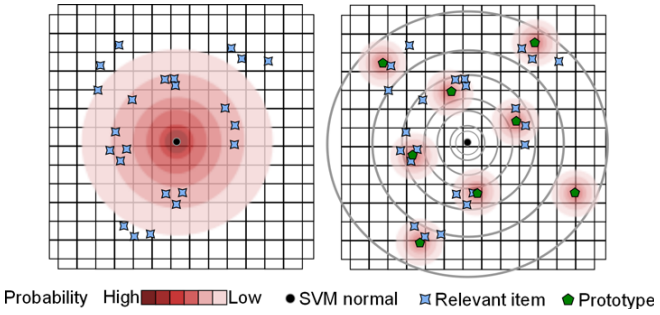

Fig. 2. Left: probing from the query requires traversing a large number of buckets to reach positive data points. Right: probing from prototypes should bring us closer to the positives

probing algorithm that searches buckets in decreasing order of their probability is given. Multi-probe LSH allows to reduce the number of different hash tables required to reach a given level of recall.

We evaluated the distribution of Hamming distances between the hashes of the data points for which f takes positive values and the hash of the query. Figure 1 shows that the maximum of this distribution is far from $d_H = 0$. To arrive at a Hamming distance $s = d_H(H(q), H(v))$, for one hash table, Multi-probe LSH should explore $\sum_{n=1}^{s} \binom{D}{n}$ hashes. The assumption made by Multi-probe LSH that the probability of finding relevant results is an isotropic normal distribution centered on the query is clearly wrong; see the illustration on the left side of Figure 2.

Since the query itself is not a good starting point, we propose to use instead the prototypes of the positive training examples as starting points. We use kernel K-means clustering to obtain K clusters on the positive training examples and for each cluster k we obtain the most central element, the cluster prototype p^k:

$$p^k = \text{argmax}_j \sum_{i=1}^{n^k} \mathcal{K}(y_j^k, y_i^k) \tag{7}$$

Our assumption is that cluster prototypes are more representative of the positive data points than the query (the normal vector to the SVM hyperplane). So, by starting to sample from the buckets of the prototypes should allow to reduce the total number of probes needed to reach a required level of recall. Finally, we apply the multi-probe algorithm of [16] for each of the prototypes.

The method we propose is illustrated in the right side of Figure 2. This picture shows the case of using two hash functions that give a range of integer values. In our case we have D hash functions that give only binary values, but the idea is the same. Note that the advantage of using cluster prototypes over cluster centers is in computation time. For a cluster center we would need to use Eq. (5) while for the prototypes we can directly use Eq. (3).

Cost reduction with query expansion. To increase recall we can increase the number of clusters and the probes around each cluster prototype. However, further probing is computationally expensive, while data points that are farther from the prototypes have lower probability of being positive. A better informed way to direct the search for candidates would be helpful. In the context of action localization we can use the fact that temporal neighbors of windows (data points) detected as positive have a high probability of being positive themselves. Also, since we have the $H(v)$ values of the temporal neighbors, we can apply a simple filter by using Eq. (6). This filtering is fast and has a very low false negative rate, so it eliminates only a negligible number of good candidates. The approximate test is only applied to the temporal neighbors of windows already detected as positive.

For details, a listing of the algorithm of our sublinear method can be found on our website[1].

3. EXPERIMENTAL EVALUATION

Dataset. We evaluate the effectiveness and efficiency of our action localization method on the Corrida dataset[2] of 77 hours of video. With the video representation described in Section 2.1, the database contains 1.3M windows, each represented by a 4000-dimensional BoVW histogram. The annotations for two actions were employed: (1) BullChargeCape—in the context of a bull fight, the bull charges the torero who dangles a cape to distract the animal, and (2) HorseRiding—one or several persons riding horses. The dataset is split into two parts. For training and parameter validation, there are 2 hours of video containing 85 examples of BullChargeCape and 50 of HorseRiding. For testing, there are 75 hours of video in which we attempt to identify 570 instances of BullChargeCape and 344 of HorseRiding.

Metrics. Following current practice, we evaluate action localization like a retrieval problem: detection windows A having positive scores S(A) are sorted by decreasing scores and a precision - recall curve is obtained. Average Precision (AP) is then computed for each class. A result window is positive if it overlaps the ground truth annotations. We aim to localize actions only in time, so only temporal overlap is measured. Ground truth annotations B_i are marked as detected if they are at least 50% inside a detection window A, $|A \cap B_i|/|B_i| > 0.5$, and successive annotations cover at least 20% of the time span of the detection window: $\sum_i |A \cap B_i|/|A| > 0.2$.

For the experiments we ran, the number of hash tables was $L = 16$ and the number of hash functions $D = 24$ and $M = 32$.

We first measure the recall while varying the number of prototypes $K = \{5, 15, 25\}$ and the number of probes $P = \{5, 10, 20, 30\}$ (Figure 3). We compare the scalable retrieval (SR below) method in Section 2.3, with or without query expansion (QE), to the exhaustive approximate (EA below) search in Section 2.2 and take the exact exhaustive (EE below) search as a reference. Thus, for a window retrieved by SR or by EA search, we check its $f(v)$ value to see if it was truly a positive result. Recall is then measured as the number of positive items retrieved divided by the total number of items marked positive by EE Search.

For the BullChargeCape class, the results show that SR obtains the recall of EA search by testing only a small additional number of data points. Moreover, we see that by using QE and $K = 5$

[1] http://cedric.cnam.fr/∼stoiana/supp.pdf

[2] http://mexculture.cnam.fr/xwiki/bin/view/Main/Datasets

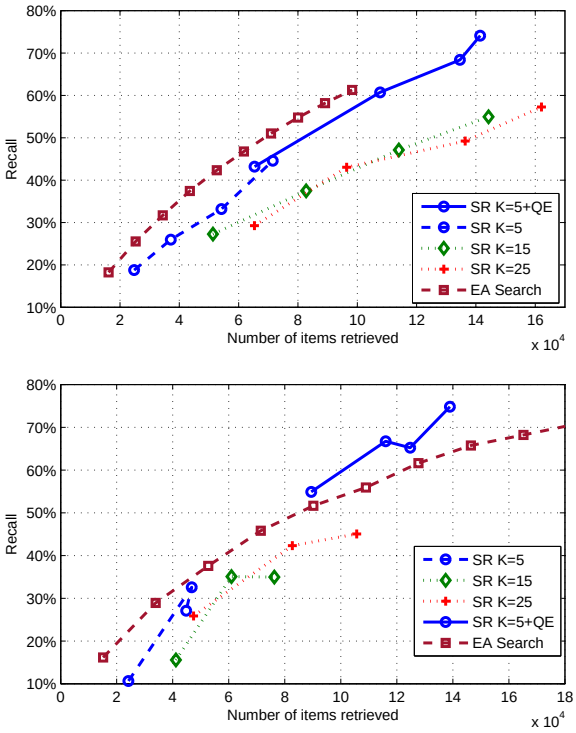

Fig. 3. Recall comparison for BullChargeCape (top) and HorseRiding (bottom). Lines correspond to variable numbers of probes at fixed K

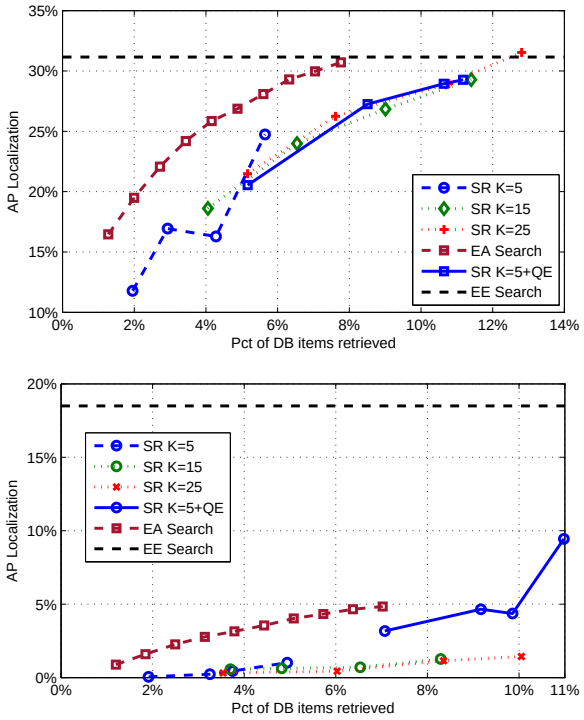

Fig. 4. Localization Average Precision: BullChargeCape (top) and HorseRiding (bottom)

clusters we obtain better results than by using more clusters. For HorseRiding, by using QE (label 'K=5+QE') we have higher recall with fewer data points retrieved than by EA search. For these results we did 2 query expansion iterations.

We now check the Average Precision for localization of SR (Figure 4). For $K = 5, P = 30$ with QE (label 'K=5+QE') for BullChargeCape SR achieves almost the same AP as the EE method while examining 11% of the database. Note that the percentage of data points considered positive according to EE is 4.3% of the database. With $K = 25, P = 30$ we obtain the same AP as EE search. SR examines 3% more data points than EA search to achieve the same AP.

For HorseRiding, neither method reaches the AP of the EE search but SR achieves 10% AP (average over 5 runs) by examining 11% of the database. For this class, the percentage of data points considered positive according to EE is 3.3% of the database. Again, SR examines 3% more data points than EA search to achieve the same AP. The major difficulty for this class is the quality of the detector (EE search): it only provides an AP of 18.3%. Even though we obtain high recall with respect to the detector (75%) with both SR and EA search (Figure 3), the post-processing window fusion method appears to be too sensitive to the absence of the remaining 25% of the positive video sequences.

We give a theoretical analysis of the complexity of our method. To run the SR method with query expansion we first need to hash the query (SVM normal vector). This takes $L \times D \times N_{SV}$ kernel computations. Next, to hash the prototypes of the clusters, we need to run $K \times L \times D \times M$ kernel computations. The complexity is thus dependent on the number of training vectors of the query SVM. For EA search the complexity is linear in the size of the database, even though only Hamming distances are to be computed. Thus, we expect SR to scale better to very large databases.

4. CONCLUSION

We presented a method that allows to find in a potentially large database most of the instances of a complex class without having to check more than a fraction of the data. The class is defined here by an SVM detector obtained on training examples. We showed that this method can approach the effectiveness of exact exhaustive search while being much more efficient since it only examines a fraction of the data. The method is not dependent on kernel type and parameters, nor on database size. For databases of medium size, we have also shown that an approximate exhaustive search method can be faster. To improve upon this work, we plan to explore better ways to sample from the distribution of potentially relevant hash buckets.

5. REFERENCES

[1] Olivier Duchenne, Ivan Laptev, and Josef Sivic, "Automatic annotation of human actions in video," in *Proc. of the Intl. Conf. on Computer Vision (2009)*, 2009, pp. 1491–1498.

[2] A. Gaidon, Z. Harchaoui, and C. Schmid, "Actom Sequence Models for Efficient Action Detection," *CVPR 2011 - IEEE Conference on Computer Vision & Pattern Recognition*, pp. 3201–3208, June 2011.

[3] D. Oneata, J. Verbeek, and C. Schmid, "Action and Event Recognition with Fisher Vectors on a Compact Feature Set,"

in *ICCV 2013 - IEEE Intenational Conference on Computer Vision*, Sydney, Australia, Dec. 2013, pp. 1817–1824, IEEE.

[4] Dan Oneata, Jakob Verbeek, and Cordelia Schmid, "Efficient Action Localization with Approximately Normalized Fisher Vectors," in *CVPR 2014 - IEEE Conference on Computer Vision & Pattern Recognition*, Columbus, OH, United States, June 2014, IEEE.

[5] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, New York, NY, USA, 2004, SCG '04, pp. 253–262, ACM.

[6] Saloua Litayem, Alexis Joly, and Nozha Boujemaa, "Hash-based support vector machines approximation for large scale prediction," in *Proceedings of the British Machine Vision Conference*. 2012, pp. 86.1–86.11, BMVA Press.

[7] Navneet Panda and E.Y. Chang, "KDX: an indexer for support vector machines," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 6, pp. 748–763, June 2006.

[8] Youngdae Kim, Ilhwan Ko, Wook-Shin Han, and Hwanjo Yu, "iKernel: Exact indexing for support vector machines," *Information Sciences*, vol. 257, no. 0, pp. 32–53, 2014.

[9] Arijit Khan, Pouya Yanki, Bojana Dimcheva, and Donald Kossmann, "Towards indexing functions: Answering scalar product queries," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2014, SIGMOD '14, pp. 241–252, ACM.

[10] Prateek Jain, Sudheendra Vijayanarasimhan, and Kristen Grauman, "Hashing hyperplane queries to near points with applications to large-scale active learning," in *Advances in Neural Information Processing Systems 23*, J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, Eds., pp. 928–936. Curran Associates, Inc., 2010.

[11] Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang, "Compact hyperplane hashing with bilinear functions," in *International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[12] Sudheendra Vijayanarasimhan, Prateek Jain, and Kristen Grauman, "Hashing hyperplane queries to near points with applications to large-scale active learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 276–288, Feb. 2014.

[13] H. Wang, A. Kläser, C. Schmid, and C.L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, May 2013.

[14] A. Joly and O. Buisson, "Random maximum margin hashing," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 873–880.

[15] Michel X. Goemans and David P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, Nov. 1995.

[16] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li, "Multi-probe LSH: Efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd International Conference on Very Large Data Bases*. 2007, VLDB '07, pp. 950–961, VLDB Endowment.