Noname manuscript No.
(will be inserted by the editor)

Video Classification with Densely Extracted HOG/HOF/MBH Features: An Evaluation of the Accuracy/Computational Efficiency Trade-off

J. Uijlings · I.C. Duta · E. Sangineto · Nicu Sebe

Received: date / Accepted: date

Abstract The current state-of-the-art in video classification is based on Bag-of-Words using local visual descriptors. Most commonly these are Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF) and Motion Boundary Histogram (MBH) descriptors. While such approach is very powerful for classification, it is also computationally expensive. This paper addresses the problem of computational efficiency. Specifically: (1) We propose several speed-ups for densely sampled HOG, HOF and MBH descriptors and release Matlab code; (2) We investigate the trade-off between accuracy and computational efficiency of descriptors in terms of frame sampling rate and type of Optical Flow method; (3) We investigate the trade-off between accuracy and computational efficiency for computing the feature vocabulary, using and comparing most of the commonly adopted vector quantization techniques: k-means, hierarchical k-means, Random Forests, Fisher Vectors and VLAD.

Keywords Video Classification, HOG, HOF, MBH, Computational Efficiency

J. Uijlings
University of Edinburgh, UK
E-mail: jrr.uijlings@ed.ac.uk

I.C. Duta
DISI, University of Trento, Italy
E-mail: duta@disi.unitn.it

E. Sangineto
DISI, University of Trento, Italy
E-mail: enver.sangineto@unitn.it

N. Sebe
DISI, University of Trento, Italy
E-mail: sebe@disi.unitn.it

1 Introduction

The Bag-of-Words method [10,37] has been successfully adapted from the domain of still images to the domain of video by using local, visual, space-time descriptors (e.g. [25,13,22,35,36,45]). Successful applications range from Human Action Recognition [25,24,32] to Event Detection [38] and Concept Classification [39,38]. However, analysing video is even more computationally expensive than analysing images. Hence, in order to deal with the enormous, growing amount of digitalized video it is important to have not only accurate, but also computationally efficient methods.

In this paper we take a powerful, commonly used Bag-of-Words pipeline for video classification and investigate how we can make it more computationally efficient while sacrificing as little accuracy as possible. The general pipeline is visualised in Figure 1. In this pipeline we focus on densely sampled local visual descriptors only, since dense sampling has been found to be more accurate than keypoint-based sampling, both in images [20] and in video [46]. As type of local visual descriptors, we focus on the standard ones, which are based on local 3D volumes of Histograms of Oriented Gradients (HOG) [11], Histograms of Optical Flow (HOF) [12,25] and Motion Boundary Histograms (MBH) [12]. For transforming the set of local descriptors extracted from a video into a fixed-length vector necessary for classification, we compare a variety of techniques: k-means, hierarchical k-means, Random Forests [4,16], Fisher Vectors [31] and Vector of Locally Aggregated Descriptors (VLAD) [19]. Starting from this pipeline, this evaluation paper makes the following contributions:

Fast Dense HOG/HOF/MBH. We exploit the nature of densely sampled descriptors in order to speed

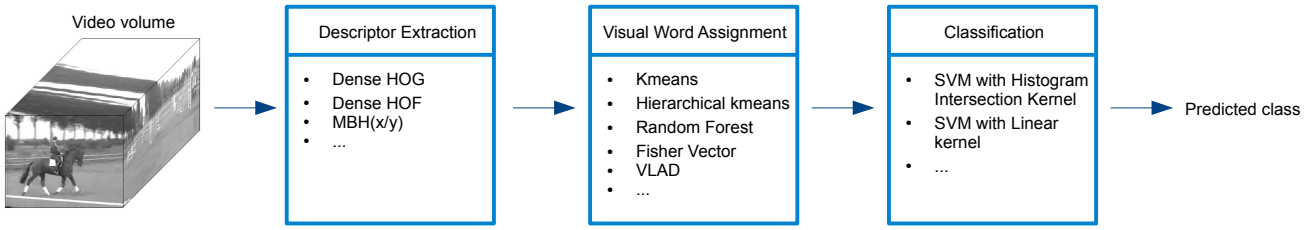

Fig. 1: General framework for video vlassification using a Bag-of-Words pipeline. The methods evaluated in this paper are instantiated in this diagram.

up their computation. HOG, HOF and MBH descriptors are created from subvolumes. These subvolumes can be shared by different descriptors similar to what was done in [42]. In this paper we generalize their idea of reusing subregions to 3 dimensions. Matlab source code will be made available[1].

Evaluation of frame subsampling. Videos consist of many frames, making them computational expensive to analyse. However, subsequent frames also largely carry the same information. In this paper we evaluate the trade-off between accuracy and computational efficiency when subsampling video frames.

Evaluation of Optical Flow. Calculating optical flow is generally expensive and takes up much of the total HOF and MBH descriptor extraction time. But for optical flow there is also a trade-off between computational efficiency and accuracy. Moreover, optical flow methods are generally tested against optical flow benchmarks such as [2,7], but it is not immediately obvious that methods which perform well on these benchmarks would automatically also yield better HOF and MBH descriptors. Therefore in this paper we evaluate optical flow methods directly in our task of interest: video classification. Specifically, we compare the optical flow methods of Lukas-Kanade [28], Horn-Schunk [17], Farnebäck [15], Brox 04 [5], and Brox 11 [6].

Evaluation of descriptor encoding. The classical way of transforming a set of local visual descriptors into a single fixed-length vector is by using a k-means visual vocabulary and assign local descriptors to the mean of the nearest cluster (e.g. [10]). However, both hierarchical k-means and Random Forests [30,42] are viable fast alternatives. Furthermore, the Fisher Vector [31] significantly outperforms classical k-means representation in many tasks, whereas VLAD [19] can be considered a simplified non-probabilistic version of the Fisher Vector [33] and it is computationally more efficient. In this paper we evaluate the accuracy/efficiency trade-off of all five methods above in the context of video classification.

[1] `http://homepages.inf.ed.ac.uk/juijling/index.php#page=software`

2 Related Work

The most used local spatio-temporal descriptors are modelled after SIFT [27]: each local video volume is divided into blocks, for each block one aggregates responses (either oriented gradients or optical flow), and the final descriptor is a concatenation of the aggregated responses of several adjacent blocks. Both Dalal et al. [12] and Laptev et al. [25] proposed to aggregate 2D Oriented Gradient Responses (HOG) and Optical Flow responses (HOF). Additionally, Dalal et al. [12] also proposed to calculate changes of optical flow, or Motion Boundary Histograms (MBH). Both Scovanner et al. [36] and Kläser et al. [22] proposed to measure oriented gradients also in the temporal dimension, resulting in 3-dimensional gradient responses. Everts et al. [14] extended [22] to include colour channels. As Wang et al. [46] found little evidence that the 3D responses of [22] are better than HOG, in this evaluation paper we implemented and evaluated the descriptors which are most widely used: HOG, HOF and MBH.

Wang et al. [46] evaluated several interest point selection methods and several spatio-temporal descriptors. They found that dense sampling methods generally outperform interest points, especially on more difficult datasets. As this result was earlier found in image analysis [20,34], this paper focuses on dense sampling for videos. In [46] the evaluation was on accuracy only. In contrast, this paper focuses on the trade-off between computational efficiency and accuracy.

Recently, Wang et al. [45] proposed to use dense trajectories. In their method, the local video volume moves spatially through time; it tries to stay on the same part of the object. Additionally, they use changes in opti-

cal flow rather than the optical flow itself. They show good improvements over normal HOG, HOF and MBH descriptors. Nevertheless, combining their dense trajectory descriptors with both normal HOG, HOF and MBH descriptors still gives significant improvements over dense trajectories alone [21,45]. In this paper we focus on HOG, HOF and MBH. Note that we evaluate the accuracy/efficiency trade-off for several optical flow methods which may be of interest also when using dense trajectories.

In [34], Sangineto proposes to use Integral Images [44] to efficiently compute densely extracted SURF features [3] in *still images*. The work of Uijlings et al. [42] proposes several methods to speed up the Bag-of-Words classification pipeline for *image* classification and provides a detailed evaluation on the trade-off between computational efficiency and classification accuracy. In this paper we perform such evaluation on *video* classification. Inspired by [42] we propose accelerated densely extracted HOG, HOF and MBH descriptors and provide efficient Matlab implementations. Additionally, we evaluate various video-specific aspects such as frame sampling rate and the choice of optical flow method.

The Fisher Vector [31] has been shown to outperform standard vector quantization methods such as k-means in the context of Bag-of-Words. On the other hand, the recently proposed VLAD descriptors [19] can be seen as a non-probabilistic version of Fisher Vectors which are faster to compute [19,33]. In this paper we evaluate the accuracy/efficiency trade-off using Fisher Vector and VLAD in the context of video classification.

3 Bag-of-Words for Video

In this section we explain in detail the pipeline that we use. We mostly use off-the-shelf yet state-of-the-art components to construct our Bag-of-Words pipeline, which is necessary for a good evaluation paper. Additionally, we explain how to create a fast implementation of densely sampled HOG and HOF descriptors, and also implicitly for MBH, being MBH based on HOG and Optical Flow. We make the HOG/HOF/MBH descriptor code publicly available.

3.1 Descriptor Extraction

In this section we describe the details of our implementation for dense extraction of HOG, HOF and MBH descriptors. Specifically, in Section 3.1.1 we show how HOG and HOF can be efficiently extracted and aggregated from video blocks. Then, in Section 3.1.2 we deal with MBHs, which are largely based on HOG. Finally, since in this paper we compare our implementation with the widely used available code of Laptev [25], in Section 3.1.3, we show the parameters we have adopted in using Laptev's code. Both ours and the Laptev' system work on grey-values only. Note that Laptev's implementation does not include MBH descriptors, thus the comparison performed in our experiments only concerns HOG and HOF.

3.1.1 Fast Dense HOG/HOF Descriptors

For both HOG and HOF descriptors, there are several steps. First one needs to calculate either gradient magnitude responses in horizontal and vertical directions (for HOG), or optical flow displacement vectors in horizontal and vertical directions (for HOF). Both result in a 2-dimensional vector field per frame. Then for each response the magnitude is quantized in o orientations, usually $o = 8$. Afterwards, one needs to aggregate these responses over blocks of pixels in both spatial and temporal directions. The next step is to concatenate responses of several adjacent pixel blocks. Finally, descriptors have to be normalized and sometimes PCA is performed to reduce their dimensionality, often leading to computational benefits or improved accuracy.

To calculate gradient magnitude responses we use HAAR-features. These are faster to compute than Gaussian Derivatives and have proven to work better for HOG [11]. Quantization in o orientations is done by dividing each response magnitude linearly over two adjacent orientation bins.

We use the classical Horn-Schunk [17] method for optical flow responses as a default. We use the version implemented by the Matlab Computer Vision System Toolbox. Additionally, we evaluate four other optical flow methods: Lucas-Kanade [28], also using the Matlab Computer Vision System Toolbox, the method of Färneback [15], using OpenCV[2] with the mexopencv interface[3], Brox 04 [5], and Brox 11 [6] using the author's publicly available code.

Both HOG and HOF descriptors are created out of blocks. By choosing the sampling rate identically to the size of a single block, one can reuse these blocks. Figure 2 shows an example on how a video volume can be divided into blocks. Once responses per block are computed, descriptors can be formed by concatenating adjacent blocks. In this paper we use descriptors of 3 by 3 blocks in the spatial domain and 2 blocks in the temporal domain, as shown in blue in Figure 2, but these parameters can be easily changed. Hence each block is

[2] `http://opencv.org`

[3] `https://github.com/kyamagu/mexopencv`

reused 18 times (except for the blocks on the borders of the video volume).

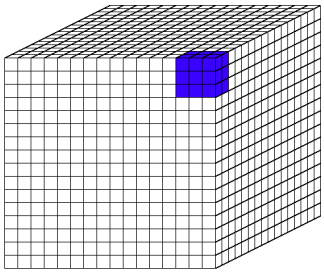

Fig. 2: Blocks in a video volume can be reused for descriptor extraction. In our paper descriptors consist of 3 by 3 blocks in space and 2 blocks in time, shown in blue.

To aggregate responses over space we use the Matlab-friendly method proposed by [42]: Let R be an $N \times M$ matrix containing responses in a single orientation (be it gradient magnitude or optical flow magniture). Let B_N and B_M be the number of elementary blocks from which HOG/HOF features are composed. Now it is possible to construct (sparse) matrices O and P of respectively $B_N \times N$ and $M \times B_M$ such that $ORP = A$, where A is a $B_N \times B_M$ matrix containing the aggregated responses for each block. O and P resemble diagonal matrices but are rectangular and the filled in elements follow the 'diagonal' of the rectangle instead of positions (i, i). By proper instantiation of these matrices we perform interpolation between blocks, which provides the descriptors some translation invariance. For integration over time we add the responses of the frames belonging to a single block. For more details we refer the reader to the work of [42].

In this paper, we extract descriptors on a single scale where blocks consist of 8 by 8 pixels by 6 frames, which at the same time is our dense sampling rate. Descriptors consist of 3 by 3 by 2 blocks. Both for HOG and HOF the magnitude responses are divided into 8 orientations, resulting in 144 dimensional descriptors. PCA is performed to reduce the dimensionality by 50% resulting in 72 dimensional vectors. Afterwards, normalization is performed by the L1-norm followed by the square root, which effectively means that Euclidean distances between descriptors in fact reflect the often superior Hellinger distance [1].

3.1.2 Motion Boundary Histograms Descriptor

Another commonly used descriptor for video classification tasks is Motion Boundary Histogram (MBH), proposed by Dalal et al. [12], who proved its robustness to camera and background motion. The intuitive idea of MBH is to represent the oriented gradients computed over the vertical and the horizontal optical flow components. The advantage of such representation is that constant camera movements tend to disappear and the description focuses on optical flow *differences* between frames (*motions boundaries*).

In more detail, the optical flow's horizontal and vertical components are separately represented using two scalar maps, which can be seen as gray-level "images" of the motion components. Histograms of oriented gradients are then computed for each of the two optical flow component images, using the same approach used for computing HOG in still images. Taken into account only flow differences, the information about changes in motion boundaries is kept and the constant motion information is removed, which leads to the cancelation of most of the effects of camera motion.

In our MBH implementation we follow the pipeline suggested in [12] and mentioned above. Once computed the horizontal and vertical optical flow components, histograms of oriented gradients are computed on each image component using the same efficient approach and the same parameters shown in Section 3.1.1. Also the block-based aggregation step is analogous to what described in Section 3.1.1.

The outcome of this process is a pair of horizontal (MBHx) and vertical (MBHy) descriptors [12], each one composed of 144 dimensions. We separately apply PCA to both MBHx and MBHy and we obtain two vectors of 72 dimensions each. The (PCA-reduced) MBHx and MBHy vectors can then be either separately used in the subsequent visual word assignment and classification stages (Figure 1) or combined in order to get a unique descriptor. In [45] the authors state that late fusion of MBHx and MBHy gives a better performance than concatenating the two descriptors before the visual word assignment step. Hence in this paper we will report results for MBHx and MBHy separately, and a late fusion of the two which we simply denote as MBH. This late fusion combines the outcomes of the two (independent) classifications with equal weights. Finally, in Section 4.6, we will also show results concerning a late fusion strategy involving all the descriptors (MBHx, MBHy, HOG and HOF).

3.1.3 Existing HOG/HOF Descriptors

We use the existing implementation of Laptev et al. [25]. We use the default parameters as suggested by the authors, which compared to our descriptors are as follows: They perform a dense sampling at multiple scales. At

the finest scale, blocks are 12 by 12 pixels by 6 frames, sampling rate is every 16 pixels by every 6 frames. They consider 8 spatial scales and 2 temporal scales for a total of 16 scales, where each scale increases the descriptor size by a factor of $\sqrt{2}$. In the end, they generate around 33% less descriptors than our single scale dense sampling method.

Unlike our descriptor extraction, the implementation of [25] uses 4 orientations for HOG and 5 orientations for HOF, resulting in respectively 72 and 90 dimensional descriptors.

3.2 Visual Word Assignment

We use five different ways of creating a single feature representation of a set of descriptors extracted from a single video: k-means, hierarchical k-means, Random Forests [4,16], VLAD [19] and Fisher Vectors [31].

For hierarchical k-means we use the implementation made available by VLFeat [43]. For the regular k-means assignment, we make use of the fact that the descriptors are L2-normalised: Euclidean distances are proportional to dot products (cosine of angles) between the vectors. Hence finding the minimal euclidean distance is equivalent to finding the maximal dot product, yet more efficient to compute [42]. For both hierarchical k-means and regular k-means, we use 4096 visual words. For hierarchical k-means, we learn a hierarchical tree of depth 2 with 64 branches per node of the tree (preliminary experiments showed a large decrease in accuracy when using a higher depth with fewer branches, but only marginal improvements in computational efficiency, data not shown). We normalize the resulting frequency histograms using the square root, which discounts frequently occurring visual words, followed by L1-normalization.

Random Forests are binary decision trees which are learned in a supervised way by randomly picking several descriptor dimensions at each node with several random thresholds and choose the one with the highest Entropy Gain. We follow the recommendations of [42], using 4 binary decision trees of depth 10, resulting in 4096 visual words. The resulting vector is normalized by taking the square root followed by L1.

The Fisher Vector [18] as used in [31] encodes a set of descriptors D with respect to a Gaussian Mixture Model (GMM) which is trained to be a generative model of these descriptors. Specifically, the set of descriptors is represented as the gradient with respect to the parameters of the GMM. This can be intuitively explained in terms of the EM algorithm for GMMs: Let G_λ be the learned GMM with parameters λ. Now use the E-step to assign the set of descriptors D to G_λ. Then the M-step yields a vector F with adjustments on how λ should be updated to fit the data (i.e. how the GMM clusters should be adjusted). This vector F is exactly the Fisher Vector representation. We follow [31] and normalize the vector using a square root of the absolute values and afterwards keep the original sign ($(sign(f_i))\sqrt{|f_i|}$), followed by L2. In this paper we use two common cluster sizes for the GMM: 64 and 256 clusters [31]. Without a spatial pyramid [26], for our 72 dimensional HOG/HOF/MBHx/MBHy features this will yield vectors of 9,216 and 36,864 dimensions respectively. While not comparable with the dimensionality of other methods, Fisher Vectors (and VLAD) allow for linear Support Vector Machines rather than Histogram Intersection or χ^2-kernels. Hence efficiency-wise, the simpler classifiers will compensate for the larger dimension of the feature vectors.

The recently proposed VLAD [19] representation can be seen as a simplification of the Fisher Vector [19, 33] in which: (1) a spherical GMM is used, (2) the soft assignment is replaced with a hard assignment and (3) only the gradient of G_λ with respect to the mean is considered (first order statistics). This leads to a lower dimensional representation, half of the dimensions of a Fisher Vector, in which second order statistics are also used. Following [19] we use for VLAD the same normalization scheme used for Fisher Vectors: We square-root the VLAD vectors while keeping their sign, followed by L2-normalisation. For good comparison to the Fisher Vectors, we use a dictionary of 128 and 512 clusters respectively, leading to features of dimensionality identical to the Fisher Vectors: 9,216 and 36,864 dimensions.

We use the Spatial Pyramid [26] in all our experiments. Specifically, we divide each video volume into the whole video and into three horizontal parts which intuitively roughly corresponds to a ground, object, and sky division (in outdoor scenes).

3.3 Classification

For classification we use Support Vector Machines which are powerful and widely used in a Bag-of-Words context (e.g. [10,26,42,43]). For k-means, hierarchical k-means, and Random Forests, we use SVMs with the Histogram Intersection kernel, using the fast classification method as proposed by [29]. For the Fisher Vector and VLAD, we use linear SVMs. For both types of SVMs, we make use of the publicly available LIBSVM library [8] and the fast Histogram Intersection classification of [29].

4 Experiments

Our baseline consists of densely sampled HOG, HOF and MBH(x/y) descriptors, all consisting of blocks of 8 by 8 pixels by 6 frames. For HOF and MBH(x/y), optical flow is calculated using Horn-Schunk. Gradient and flow magnitude responses are quantized in 8 bins. The final descriptors consist of 3 by 3 by 2 blocks. PCA always reduces dimensionality of descriptors by 50%. We use a spatial pyramid division of $1 \times 1 \times 1$ and $1 \times 3 \times 1$ [26] (we have no temporal division). Normalisation after word assignment is done by either taking the square root while keeping the sign followed by L2 for the Fisher Kernel, or by the square root plus L1 for all other methods. We use SVMs for classification, with either a linear kernel for the Fisher Vectors or histogram intersection kernel for all other visual word assignment methods.

Starting from our baseline we perform four experiments: (1) We compare five different visual word assignment methods: k-means, hierarchical k-means, Random Forests, VLAD and the Fisher Kernel; (2) We compare our densely extracted descriptors with the descriptors provided by Laptev et al. [25]; (3) We evaluate the efficiency/accuracy trade-off by subsampling video frames for the descriptor extraction process; (4) For HOF and MBH(x/y) descriptors, we compare five different optical flow implementations: Horn-Schunk, Lukas-Kanade, Farnebäck [15], Brox 04 [5] and Brox 11 [6].

All timing experiments are performed on a single core of an Intel(R) Xeon(R) CPU E5620 2.40GHz. We use mainly Matlab, but most toolboxes used by us have mex-interfaces to c++ implementations for critical functions. All implementations are heavily optimized for speed. Since the computation involves many common operations that use standardized and optimized libraries (e.g. convolutions, matrix multiplications) on large quantities of data, virtually the entire time is spend on core calculations while the overhead is negligible; using only c++ will not result in noticeable differences in the overall timing results presented in this paper.

Based on our experiments we provide two recommendations, one for real-time video classification and one for accurate video classification. Finally we give a comparison with the state-of-the-art.

4.1 Dataset

We perform all experiments on the UCF50 Human Action Recognition dataset [32]. This dataset contains 6600 realistic videos taken from Youtube and as such has large variations in camera motion, object appearance and pose, illumination conditions, scale, etc. The 50 human action categories are mutually exclusive and include actions such as biking, diving, drumming, and fencing. The frames of the videos are 320 by 240 pixels. The video clips are relatively short with a length that varies around 70-200 frames. The dataset is divided in 25 predefined groups. Following the standard procedure we perform a leave-one-group-out cross-validation and report the average classification accuracy over all 25 folds. Optimization of the SVM slack parameter is done for every class for every fold on the training set (containing 24 groups).

4.2 Visual Word Assignment

In this experiment we compare the following visual word assignment methods: k-means, hierarchical k-means, Random Forests, VLAD and Fisher Vector. K-means, hierarchical k-means and Random Forests are similar in the sense that the final vector represents visual word counts. To compare these methods we ensure that all have 4096 visual words. For k-means this means performing clustering with k=4096. For hierarchical k-means we use a hierarchy of depth 2 with 64 branches at each node. The Random Forest consists of 4 trees of depth 10. We choose to base our Fisher Vectors on standard sizes for the number of clusters: 64 and 256 clusters [31,9]. While Fisher Vectors are of higher dimensionality, the vectors work with linear classifiers. This means that Fisher Vectors are best compared with the other visual word assignment methods in terms of the accuracy/efficiency trade-off. Similarly, we adopted 2 standard cluster sizes for VLAD: 128 and 512 dimensions respectively [19] and we used linear classifiers as well.

The accuracy and computational efficiency for the various word assignment methods for our HOG, HOF and MBH(x/y) features are presented in Figure 3 and Table 1. The first thing to notice is that the Fisher Vector with 256 clusters has the best accuracy of 0.765 for HOG, 0.795 for HOF, 0.796 for MBHx and 0.804 for MBH, while taking 3.39 seconds per video (per descriptor type). K-means has also good accuracy at 0.728 for HOG, 0.791 for HOF, 0.782 for MBHx and 0.8 for MBH. However, the computational time is at 1.81 seconds per video. This means that the Fisher Vector (with 256 clusters) for video classification is superior in accuracy but slightly slower compared to k-means. For computational efficiency, the Random Forest is by far the fastest and takes 0.1 seconds per video. The hierarchical k-means (hk-means) is four times slower at 0.47 seconds per video, and performs slightly worse on HOG (0.718 hk-means vs. 0.729 RF) but significantly better on HOF (0.780 hk-means vs. 0.732 RF) and on MBHx,

	k-means	hk-means	RF	FV 64	FV 256	VLAD 128	VLAD 512
HOG Acc	0.728	0.718	0.729	0.746	0.765	0.653	0.671
HOF Acc	0.791	0.780	0.732	0.779	0.795	0.751	0.783
MBHx Acc	0.782	0.774	0.738	0.767	0.796	0.749	0.774
MBHy Acc	0.772	0.763	0.739	0.759	0.787	0.737	0.765
MBH Acc	0.800	0.791	0.765	0.786	0.804	0.769	0.792
sec/video	1.81	0.51	0.10	1.10	3.39	0.19	0.47
frame/sec	108	387	1910	180	58	1011	415

Table 1: Trade-off accuracy/efficiency for the following visual word assignment methods: k-means, hierarchical k-means (hk-means), Random Forest (RF), Fisher Kernel with 64 and 256 clusters (FK 64 and FK 256). Assignment time for HOG and HOF is the same.

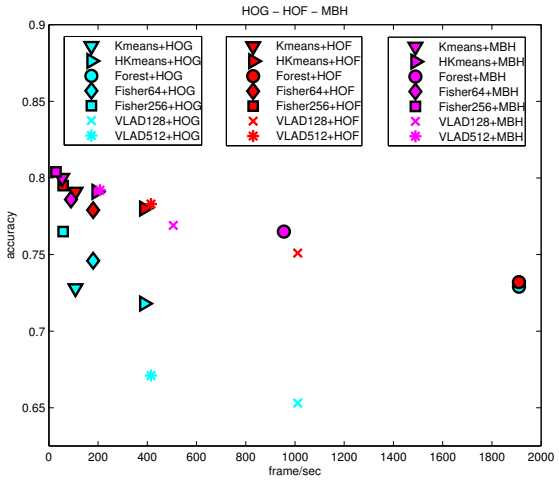

Fig. 3: Accuracy/Efficiency trade-off for various word assignment methods and features. For a better readability of the figure, we omitted the results concerning MBHx and MBHy (see Table 1).

MBHy and MBH (respectively, 0.774 vs 0.738, 0.763 vs. 0.739 and 0.791 vs 0.765).

In terms of classification time per video, we measure 0.017 seconds per video when using the fast Histogram Intersection based classification for SVMs [29] for k-means, hk-means, and Random Forests. We measure 0.001 seconds per video for the linear classifier used on the Fisher Vector representation with 256 clusters. This means that the classification time is negligible compared to the word assignment time and is of little concern for video classification.

For the remainder of this paper, we choose to perform our evaluation on two word assignment methods: the Fisher Vector, which yields the most accurate results, and hk-means, which is the second fastest after Random Forests, while its accuracy for HOF and MBH(x/y) is much higher than using Random Forests.

4.3 Comparison with Laptev et al.

In this experiment we compare the publicly available code from [25] with our implementation. We compare only to the dense sampling option as [46] has already proven that dense sampling outperforms the use of space-time interest points. Moreover, only HOG and HOF features are used for comparison because the code in [25] does not include any implementation for MBH features. Results are presented in Figures 4 and 5 and in Table 2.

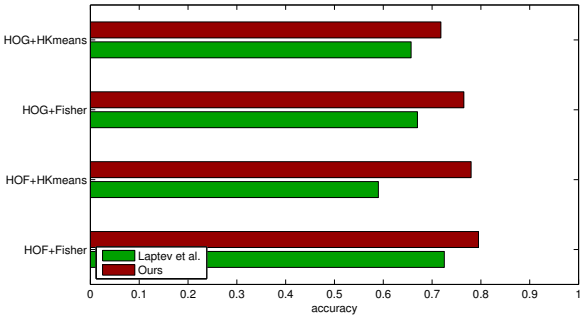

Fig. 4: Accuracy comparison between [25] and our HOG/HOF descriptors

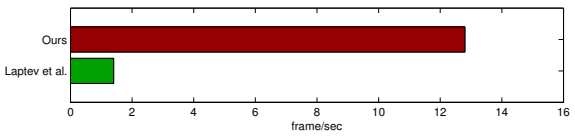

Fig. 5: Computational Efficiency comparison between [25] and our HOG/HOF descriptors

The results show that for all settings there is a significant difference in accuracy between the dense implementation of [25] and our method. For the Fisher Vector, HOG descriptors yield 0.670 accuracy for [25] and 0.765 accuracy for our implementation and HOF

	hk-means		FV 256		efficiency	
	HOG	HOF	HOG	HOF	sec/vid	frame/sec
[25]	0.657	0.590	0.670	0.725	141	1.4
ours	0.718	0.780	0.765	0.795	15	12.8

Table 2: Comparing the dense HOG/HOF implementation of [25] and ours. The descriptor extraction time is measured for extracting both HOG and HOF features, as the binary provided by [25] does always both. Descriptor extraction time is independent of the visual word assignment method (RF or FV 256).

descriptors yield 0.725 accuracy for [25] and 0.795 accuracy for our implementation. These are accuracy increases of 9% and 7% respectively. Similar differences are obtained using hk-means. Part of the difference can be explained by the fact that we sample differently: because we reuse blocks of the descriptors, our sampling rate is defined by the size of a single block. This means we sample descriptors every 8 pixels and every 6 frames at a single scale, whereas [25] samples every 16 pixels and every 6 frames at 10 increasingly course scales. For our method this yields around 150 descriptors per frame or around 29,000 descriptors per video whereas [25] generates around 90 descriptors per frame or around 17,500 descriptors per video, which means we generate 66% more descriptors. While this may seem unfair towards [25], in this paper we are interested in the trade-off between accuracy and computational efficiency, which makes the exact locations from where descriptors are sampled irrelevant.

In terms of computational efficiency our method is more than 9 times faster: their method takes 141 seconds per video while our method takes 15 seconds per video. Our method is faster because we reuse blocks in our dense descriptor extraction method. Note that because the method of [25] samples fewer descriptors, visual word assignment time is faster. But by using [25] the overall computation time will be completely dominated by descriptor extraction.

To conclude, our implementation is significantly faster and significantly more accurate than the version of [25].

4.4 Subsampling Video Frames

In video, subsequent video frames largely contain the same information. As the time for descriptor extraction is the largest bottleneck in video classification, we investigate how the accuracy behaves if we subsample video frames and hence speed-up the descriptor extraction process.

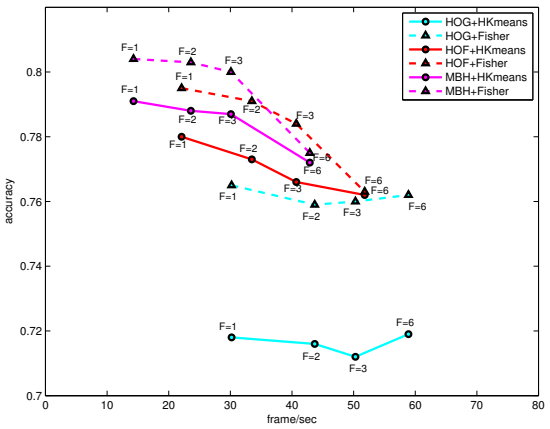

Fig. 6: Trade-off accuracy/efficiency when varying sampling rate. F stands for frames per block and it is directly related to sampling rate.

For a fair comparison, we want the descriptors always to describe the same video volume. In our baseline, each descriptor block consists of 8 by 8 pixels by 6 frames. To subsample in such a way that every block describes the same video volume regardless of the sampling rate, we do the following: if we sample every 2 frames, we aggregate responses over 3 frames (i.e. of frame 2, 4 and 6). When sampling every 3 frames, we aggregate responses over 2 frames (i.e. frame 2 and 5), and when sampling every 6 frames in which we only consider a single frame per descriptor block (i.e. frame 3). Results are presented in Figure 6 and Table 3.

For HOG descriptors, subsampling video frames has surprisingly little effect on the accuracy, both for hk-means and Fisher Vectors: using Fisher Vectors, a sampling rate of 1 yields an accuracy of 0.765 while a sampling rate of 6 yields 0.762 accuracy. The result of hk-means is basically constant, with slight oscillations. In terms of computational efficiency, a significant speed-up is achieved: sampling every 6 frames instead of every frame gives a speed-up from 6.5 seconds per video to 3.3 seconds per video.

For HOF descriptors, subsampling has a bigger impact: For the Fisher Vector accuracy is 0.795 using a sampling rate of 1, maintains a respectable 0.791 accuracy at a subsampling rate of 2 frames, while dropping significantly to 0.763 for sampling every 6 frames. Accuracy with hk-means is less affected and drops from 0.78 at sample rate of 1 to 0.762 at sample rate 6. Again, a good speed-up is obtained by subsampling. While descriptor extraction takes 8.9 seconds when using every frame, a sampling rate of 2 yields a factor 1.5 speed-

	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	hk-means	0.718	0.716	0.712	0.719
	FV 256	0.765	0.759	0.760	0.762
	sec/vid	6.5	4.5	3.9	3.3
	frame/sec†	30.2	43.7	50.3	58.9
HOF	hk-means	0.780	0.773	0.766	0.762
	FV 256	0.795	0.791	0.784	0.763
	sec/vid	8.9	5.9	4.8	3.8
	frame/sec†	22.1	33.5	40.7	51.8
MBHx	hk-means	0.774	0.767	0.769	0.758
	FV 256	0.796	0.794	0.788	0.771
	sec/vid	9.4	6.1	5.0	3.9
	frame/sec†	20.9	32.1	39.4	50.7
MBHy	hk-means	0.763	0.757	0.752	0.741
	FV 256	0.787	0.785	0.772	0.750
	sec/vid	9.4	6.1	5.0	3.9
	frame/sec†	20.9	32.1	39.4	50.7
MBH	hk-means	0.791	0.788	0.787	0.772
	FV 256	0.804	0.803	0.800	0.775
	sec/vid	13.7	8.3	6.5	4.6
	frame/sec†	14.3	23.6	30.1	42.9

Table 3: Trade-off between frame sampling rate and accuracy. We keep video volumes from which descriptors are extracted the same for all sampling rates. †Frames/second is measured in terms of the total number of frames of the video, not in terms of how many frames are actually processed during descriptor extraction.

up while sampling every 6 frames yields a factor 2.34 speed-up.

The remaining rows of Table 3 present results obtained with different combinations of the vertical and the horizontal components of the Motion Boundary Histograms (Section 3.1.2). Note that when calculating both components of the MBH features, the optical flow has to be calculated only once, so computation time is faster than simply adding the times of MBHx and MBHy.

We observe a particular order of accuracy among these three combinations: using the only horizontal component (MBHx) always results in a higher accuracy than using the only vertical component (MBHy), independently of whether Fisher Vectors or hk-means is used as word assignment method. This sharp difference is probably due to the fact that in the test videos the horizontal motion is more frequent than the vertical one. Moreover, as expected, late fusion of the two components (MBH), always outperforms using MBHx only. Concerning the drop of accuracy depending on the sample rate, for all the three descriptor combinations (MBHx, MBHy, MBH) and both word assignment methods (Fisher Vectors and hk-means), the accuracy loss as a function of the sample rate is similar to what happens with HOF and much higher than HOG. We believe that this is due to the fact that HOG are basically "static" features, representing the appearance of a given image window independently of possible motion information. As a consequence, they are less affected by optical flow errors (which is used to compute both HOF and MBH(x/y)) and better exploit the redundancy of consecutive video frames.

As for HOG and HOF and also for MBH(x/y) and MBH, we observe a significant computational efficiency gain using subsampling. For instance, sampling every 6 frames yelds a factor of 2.4 speed-up for MBH(x/y) and a factor of 3 speed-up for MBH with respect to using all the frames.

To conclude, HOG descriptors can be sampled every 6 frames with negligible loss of accuracy yielding a speed-up of a factor 2. HOF and MBH descriptors can be sampled every 2 frames with negligible loss of accuracy yielding a speed-up of a factor 1.5 and 1.7 respectively. When speed is more important than accuracy, both HOF and MBH descriptors can also be sampled every 6 frames leading to 1-3% accuracy loss while gaining a significant speed-up of a factor 2.3-3.

4.5 Choice of Optical Flow

The results reported in the previous section show that both the HOF and the MBH(x/y) descriptors are much more expensive to extract than the HOG descriptors (Table 3). This is because calculating the optical flow is computationally expensive. Additionally, not much research has been done on how different optical flow methods affect HOF/MBH descriptors. Therefore in this experiment we evaluate five available optical flow implementations to investigate both their computational efficiency and accuracy. In particular, we compare: (1) Farnebäck [15] from OpenCV using the mexopencv interface, (2) Lucas-Kanade [28] and (3) Horn-Schunk [17] from the Matlab Computer Vision Systems Toolbox, (4) Brox 04 [5] and (5) Brox 11 [6] using the available author's code.

Results are presented in Tables 4 and 5. Specifically, while in Table 4 we used the same setting adopted in the other experiments of this paper, in Table 5 we downscaled the frame resolution of all the videos by a factor of 4 (i.e., using 80×60 pixel frames) and we subsampled every 6 frames (see Section 4.4). This scale and

time subsampling was necessary in order to process our large video dataset with both Brox 04 and Brox 11, two state-of-the-art dense optical flow methods not able to process videos in real time. In fact, processing all the frames of our 6600 videos at full spatial resolution with Brox 11 would require a few months.

With the original frame resolution (Table 4), and with both hk-means and Fisher Vectors, the three computationally feasible optical flow methods have the same ranking in terms of accuracy. For the Fisher Vector, Horn-Schunk performs best at an accuracy of 0.795, followed by Lucas-Kanade at an accuracy of 0.747, while the method of Farnebäck performs relatively poorly with an accuracy of 0.641. These results show that the optical flow method is crucial to the performance of the HOF descriptor: the choice of optical flow affects the results by up to 15%(!).

In terms of computational efficiency, Lucas-Kanade is the fastest at 27 frames/second, followed by Horn-Schunk at 22 frames per second, while Farnebäck is slower with 10 frames/second. However, while Lucas-Kanade is faster, its trade-off between efficiency and accuracy is not good: As seen in Table 3 Horn-Shunk with a frame sampling rate of 2 outperforms the Lukas-Kanade results in Table 4 in both speed (33 frames vs 27 frames) and accuracy (0.77 vs 0.75).

Table 5 reports results when we subsample frames and reduce the frame size by a factor 4, enabling comparison with the Brox methods. Note that for a fair comparison these times include the computation for reducing the frame sizes (although these times are negligible compared to the total description extraction time). It can be seen that both Brox methods are better than Farneback, but surprisingly not better than the Horn-Shunk and Lucas-Kanade method. One explanation is that this is due to the low resolution of the frames, which makes dense optical flow extraction not sufficiently accurate. Another possibility is that optical flow methods performing better on optical flow benchmarks are not necessarily optimal for use in classification; reducing mistakes in most parts of the flow may introduce artefacts elsewhere that negatively affect results in a classification framework.

In terms of computational efficiency, Brox 11 is the slowest, followed by Brox 04: even subsampled on reduced frames Brox 04 still processes only 27 frames/sec. In contrast to results without downsampling, Farnebäck is here the fastest method. Apparently, there is some overhead in the Matlab optical flow implementations.

To conclude, the choice of optical flow method drastically influences the power of the resulting HOF descriptor and it is not necessarily correlated with the performance on optical flow benchmarks. Additionally, many optical flow methods aim for accuracy rather then computational efficiency (e.g. Sun et al. [41] provide a very good overview for accuracy but do not report computational efficiency). Indeed, except the Horn-Schunk, Lucas-Kanade, and Farnebäck methods we did not find any other freely available optical flow method fast enough for use in our classification pipeline. Our evaluation shows that the Horn-Schunk method has the best trade-off between accuracy and computational efficiency and that subsampling every two frames works better than switching to Lucas-Kanade optical flow. Horn-Schunk is therefore the current method of choice.

4.6 Recommendations for Practitioners

Based on the results of the previous experiments, we can now give several recommendations when accuracy or computational efficiency is preferred. For calculating Optical Flow, Section 4.5 showed that the Matlab implementation of Horn-Schunk is always the method of choice. In terms of frame sampling rate, for HOG descriptors we always recommend a sampling rate of every 6 frames. For HOF descriptor, if one wants accuracy we recommend a sampling rate of every 2 frames and if one wants computational efficiency we recommend a sampling rate of 6. The same holds for MBH(x/y) descriptors. For the word assignment method, the Fisher Vector is the method of choice for accuracy. For computational efficiency there are two candidates: hierarchical k-means and the Random Forest. Observe first that the descriptor extraction time is the most costly phase of the pipeline: Extracting HOF descriptors with a sampling rate of 6 frames takes 3.8 seconds per video to compute. And while the Random Forest is five times faster than hierarchical k-means, the difference is only 0.41 seconds per video, which is very small compared to the descriptor extraction phase. Furthermore, Table 1 showed a significant drop of accuracy from 0.780 for hierarchical k-means to 0.732 for Random Forests (and a similar drop of accuracy is observed with MBH(x/y)). Therefore we recommend using hierarchical k-means for a fast video classification pipeline.

We found that late fusion of the classifier outputs gave slightly better results than early fusion of the descriptors (e.g. concatenating HOG and HOF). Hence in our recommendations we perform a late fusion with equal weights.

We tested different descriptor combinations, using equal-weights-based late fusion and with the goal of selecting: (1) the most accurate set of descriptors, possibly taking into account the complementarity of appearance/motion information of different features, and (2)

	Horn-Schunk	Lucas-Kanade	Färneback
hk-means	0.780	0.750	0.652
FV 256	0.795	0.747	0.641
sec/video	8.8	7.2	19.0
frame/sec	22	27	10

Table 4: Comparison of different optical flow methods used to compute HOF features. Results obtained with no frame subsampling and at full original spatial resolution (320 × 240 pixels).

	Horn-Schunk	Lucas-Kanade	Färneback	Brox 04	Brox 11
hk-means	0.713	0.681	0.529	0.548	0.552
FV 256	0.718	0.697	0.542	0.638	0.652
sec/video	2.9	2.8	0.76	7.2	12.4
frame/sec	68	69	257	27.4	16

Table 5: Comparison of different optical flow methods used to compute HOF features. Results obtained subsampling a frame every 6 and at reduced spatial resolution (80 × 60 pixels).

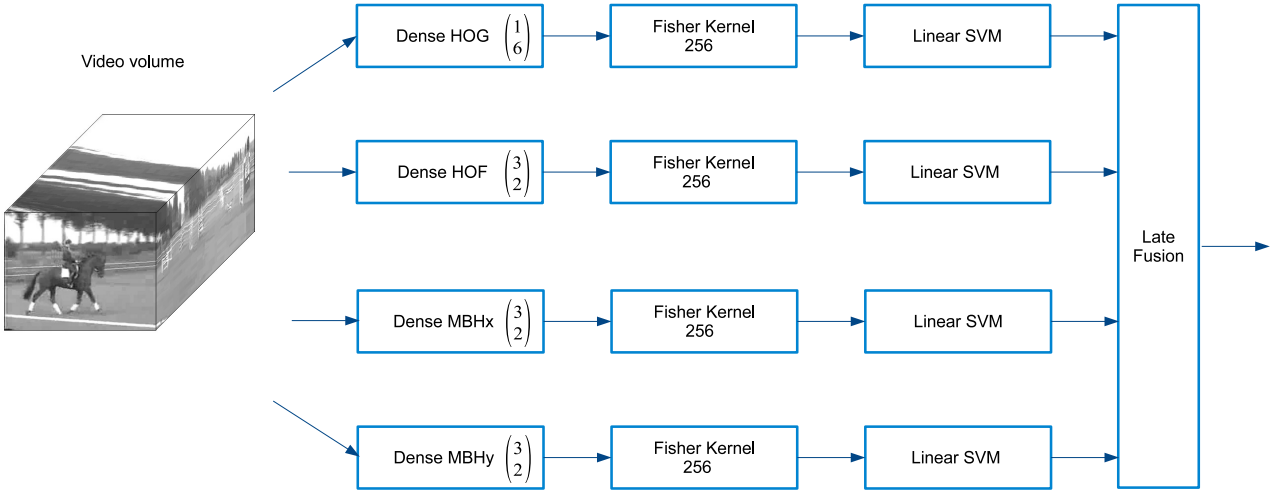

Fig. 7: Recommended pipeline for accurate video classification. This pipeline yields an accuracy of 0.818 on UCF50 while processing 9 frames per second.

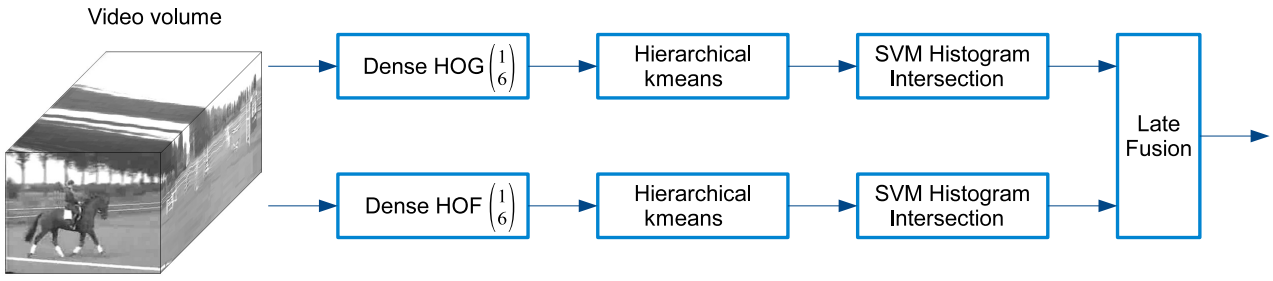

Fig. 8: Recommended pipeline for realtime video classification. This pipeline yields an accuracy of 0.790 on UCF50 while processing 28 frames per second.

Method	Accuracy
Wang et al. [45] (2013)	0.856%
This paper	0.818%
Reddy et al. [32] (2012)	0.769%
Solmaz et al. [40] (2012)	0.737%
Everst et al. [14] (2013)	0.729%
Kliper-Gross et al. [23] (2012)	0.727%

Table 6: Comparison with the State-of-the-Art.

the fastest solution with a sufficiently good accuracy degree. The final recommended pipelines are visualised in Figures 7 and 8.

The most accurate pipeline (Figure 7) combines all the descriptors we adopted in this paper: HOG, HOF, MBHx and MBHy. HOG are extracted using all the frames, while HOF and MBH(x/y) are extracted with a sampling rate of 2. The word assignment method used in this case is the Fisher Vector. Using this pipeline we can process 11 frames per second (for video frames of 320 by 240 pixels) at an accuracy of 0.818 on UCF50. Conversely, our recommended pipeline for computational efficiency (Figure 8) is based on late fusion of only HOG and HOF, both extracted with a sampling rate of 6 and using hk-means. This second pipeline can process 28 frames per second at a respectable accuracy of 0.790.

4.7 Comparison with State-of-the-Art

In this section we compare our descriptors to the state-of-the-art. Results of several recent works are given in Table 6. This comparison is done in terms of accuracy only, as most compared methods evaluate accuracy only. *This paper* in Table 6 indicates the late fusion of all the descriptors (HOG, HOF, MBH(x/y)): see Section 4.6 and Figure 7.

As can be seen, the method of [45] yields the best results. This method is a combination of Dense Trajectories and STIP features [25]. As our results are better than [25], we expect that a combination of dense trajectories with our method would increase results further. In general, our method yields good performance compared to many recently proposed methods, which shows that we provide a strong implementation of densely sampled HOG, HOF and MBH(x/y) descriptors.

5 Conclusion

This paper presented an evaluation of the trade-off between computational efficiency and accuracy for video classification using a Bag-of-Words pipeline with HOG, HOF and MBH descriptors. Our first contribution is a strong and fast Matlab implementation of densely sampled HOG, HOF and MBH descriptors, which we make publicly available.

In terms of visual word assignment, the most accurate method is the Fisher Kernel. Hierarchical k-means is more than 6 times faster while yielding an accuracy loss of less than 2% and is the method of choice for a fast video classification pipeline. HOG descriptors can be subsampled every 6 frames with a negligible loss in accuracy, while being 2 times faster. HOF and MBH descriptors can be subsampled every 2 frames with negligible loss in accuracy, being 1.5 - 1.7 times faster. When speed is essential, HOF and MBH descriptors may be subsampled every 6 frames.

For the HOF and MBH descriptors, we showed that the choice of optical flow algorithm has a large impact on the final performance. The difference between the best method, Horn-Schunk, and the second best method, Lucas-Kanade, is already 5%, while the difference with Färneback is a full 15%. Brox 04 and Brox 11 are computationally very demanding, and cannot be used in a real time video classification scenario.

Compared to the state-of-the-art, the Dense Trajectory method of [45] obtains better results. Nevertheless, the huge difference for the choice of optical flow methods suggests this would also influence dense trajectories. Furthermore, Dense Trajectories still benefit from a combination with normal HOG, HOF and MBH desciptors [21,45]. Finally, comparisons with other recent methods on UCF50 shows that we provide a strong implementation of dense HOG, HOF and MBH descriptors to the community.

References

1. R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.
2. S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *IJCV*, 2011.
3. H Bay, A Ess, T Tuytelaars, and L Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110:346–359, 2008.
4. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
5. Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36, 2004.
6. Thomas Brox and Jitendra Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011.
7. D.J. Butler, J. Wulff, G.B. Stanley, and M.J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.
8. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on*

Intelligent Systems and Technology, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
9. K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.
10. G Csurka, C R Dance, L Fan, J Willamowski, and C Bray. Visual Categorization with Bags of Keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004.
11. N Dalal and B Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
12. N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.
13. P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.
14. I. Everts, J. van Gemert, and T. Gevers. Evaluation of color STIPs for human action recognition. In *CVPR*, 2013.
15. G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, 2003.
16. P Geurts, D Ernst, and L Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.
17. B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
18. T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.
19. H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010.
20. F Jurie and B Triggs. Creating Efficient Codebooks for Visual Recognition. In *ICCV*, 2005.
21. S. Karaman, L. Seidenari, A. Bagdanov, and A. del Bimbo. L1-regularized logistic regression stacking and transductive CRF smoothing for action recognition in video. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.
22. A. Kläser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.
23. O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, 2012.
24. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011.
25. I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
26. S Lazebnik, C Schmid, and J Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006.
27. D G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60:91–110, 2004.
28. B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, 1981.
29. S Maji, A C Berg, and J Malik. Classification using Intersection Kernel Support Vector Machines is Efficient. In *CVPR*, 2008.
30. F Moosmann, E Nowak, and F Jurie. Randomized Clustering Forests for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:1632–1646, 2008.
31. F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *ECCV*, 2010.
32. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. In *Machine Vision and Applications*, 2012.
33. J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.
34. E. Sangineto. Pose and expression independent facial landmark localization using dense-SURF and the Hausdorff distance. *PAMI*, 2013.
35. C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICIP*, 2004.
36. P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.
37. J Sivic and A Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.
38. A F Smeaton, P Over, and W Kraaij. Evaluation campaigns and TRECVID. In *ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, 2006.
39. C G M Snoek, M Worring, J Gemert, J Geusebroek, and A Smeulders. The Challenge Problem for Automated Detection of 101 Semantic Concepts in Multimedia. In *ACM MM*, 2006.
40. B. Solmaz, S. Assari, and M. Shah. Classifying web videos using a global video descriptor. *Machine Vision and Applications*, 2012.
41. D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2013.
42. J R R Uijlings, A W M Smeulders, and R J H Scha. Real-time Visual Concept Classification. *IEEE Transactions on Multimedia*, 12, 2010.
43. A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM MM*, 2010.
44. P Viola and M Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings CVPR*, volume 1, pages 511–518, 2001.
45. H. Wang, A. Kläser, C. Schmid, and C. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103:60–79, 2013.
46. H. Wang, M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.