Realtime Video Classification using Dense HOF/HOG

J.R.R. Uijlings
DISI, University of Trento, Italy
jrr@disi.unitn.it

I.C. Duta
DISI, University of Trento, Italy
duta@disi.unitn.it

N. Rostamzadeh
DISI, University of Trento, Italy
rostamzadeh@disi.unitn.it

N. Sebe
DISI, University of Trento, Italy
sebe@disi.unitn.it

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) and Histogram of Optical Flow (HOF) descriptors. While such system 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 and HOF 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 the video representation, using either a k-means or hierarchical k-means based visual vocabulary, a Random Forest based vocabulary or the Fisher kernel.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Miscellaneous

Keywords

Video Classification, HOG, HOF, Computational Efficiency

1. INTRODUCTION

The Bag-of-Words method [7, 31] has been succesfully adapted from the domain of images to the domain of video (e.g. [21, 10, 18, 29, 30, 38]). Succesful applications range from Human Action Recognition [21, 20, 28] to Event Detection [32] and Concept Classification [33, 32]. However, analysing video is even more expensive than analysing images. Hence in order to be able to deal with the enormous, growing amount of digitalized video it is important to have computationally efficient systems.

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. The general pipeline is visualised in Figure 1. It has been shown that Bag-of-Words works better using densely sampled descriptors rather than descriptors taken at key-points, both in images [16] and in video [39]. Therefore we focus on densely sampled descriptors. As type of descriptors, we focus on the standard ones, which are based on local 3D volumes of Histograms of Oriented Gradients (HOG) and Histograms of Optical Flow (HOF). For the final representation of the video we use Random Forest and the Fisher Kernel by default. Starting from this pipeline, this paper makes the following contributions:

- We exploit the nature of densely sampled descriptors in order to speed up their computation. HOG and HOF descriptors are created from subvolumes. These subvolumes can be shared by different descriptors similar to what was done in [36]. In this paper we generalize their idea of reusing subregions to 3 dimensions.
- 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.
- Calculating optical flow is generally expensive and takes up much of the total HOF descriptor extraction time. But for optical flow there is also a trade-off between computational efficiency and accuracy. Whereas accuracy for optical flow is mostly measured on optical flow benchmarks such as [2, 4], in this paper we investigate this trade-off directly on our task of interest: video classification. Specifically, we compare the optical flow methods of Lukas-Kanade [24], Horn-Schunk [14], and Farnebäck [12].
- A k-means visual vocabulary is the most commonly used Bag-of-Words technique. However, it has been shown in [26, 36] that Random Forests [3, 13] are a computationally efficient alternative. On the other hand, the Fisher kernel [27] has been shown to be more accurate. This paper compares the accuracy/efficiency trade-off in the context of video classification for all three methods.

Matlab code for HOG and HOF descriptors is available[1].

ICMR '14 April 01 - 04 2014, Glasgow, United Kingdom
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2782-4/14/04 ...$15.00.
http://dx.doi.org/10.1145/2578726.2578744.

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

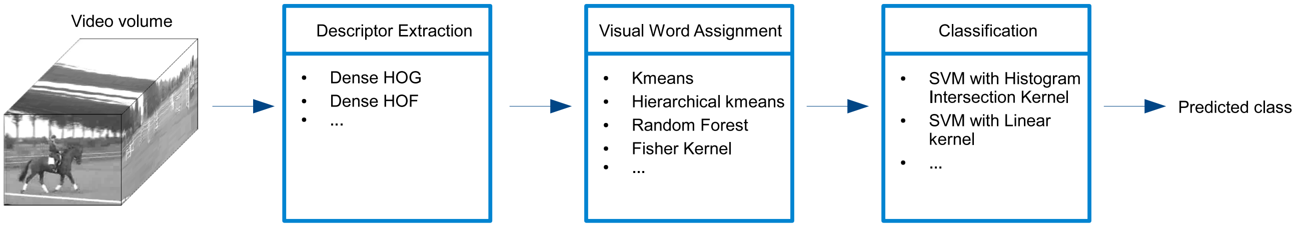

Figure 1: General Framework for Video Classification using a Bag-of-Words pipeline. The methods evaluated in this paper are instantiated in this diagram. For both HOG and HOF descriptors we evaluate several variations.

2. RELATED WORK

The most used local spatio-temporal descriptors are modelled after SIFT [23]: 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. [9] and Laptev et al. [21] proposed to aggregate 2D Oriented Gradient Responses (HOG) and Optical Flow responses (HOF). Additionally. Dalal et al. [9] also proposed to calculate changes of optical flow, or Motion Boundary Histograms. Both Scovanner et al. [30] and Kläser et al. [18] proposed to measure oriented gradients also in the temporal dimension, resulting in 3-dimensional gradient responses. Everts et al. [11] extended [18] to include colour channels. As Wang et al. [39] found little evidence that the 3D responses of [18] are better than HOG, in this evaluation paper we implemented and evaluated the descriptors which are most widely used: HOG and HOF.

Wang et al. [39] 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 [16], this paper focuses on dense sampling for videos. In [39] the evaluation was on accuracy only. In contrast, this paper focuses on the trade-off between computational efficiency and accuracy.

Recently, Wang et al. [38] 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 optical flow rather than the optical flow itself. They show good improvements over normal HOG and HOF descriptors. Nevertheless, combining their dense trajectory descriptors with both normal HOG and HOF descriptors still gives significant improvements over dense trajectories alone [17, 38]. In this paper we focus on HOG and HOF. Note that we evaluate the accuracy/efficiency trade-off for several optical flow methods which may be of interest also when using dense trajectories.

The work of Uijlings et al. [36] 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 their work we propose accelerated densely extracted HOG and HOF descriptors and provide an implementation. Additionally, we evaluate various video-specific aspects such as frame sampling rate and the choice of optical flow method.

The Fisher Kernel [27] has proven to consistently outperform standard vector quantization methods such as k-means in the context of Bag-of-Words. In this paper we evaluate the accuracy/efficiency trade-off using the Fisher Kernel 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. We make the HOG/HOF descriptor code publicly available.

3.1 Descriptor Extraction

In this paper we compare HOG and HOF descriptors using two implementations. The first is the fast implementation which we created ourselves. The second is the widely used available code of [21]. Both methods work on grey-values only.

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 [8]. Quantization in o orientations is done by dividing each response magnitude linearly over two adjacent orientation bins.

We use the classical Horn-Schunk [14] method for optical flow responses as a default. We use the version implemented by the Matlab Computer Vision System Toolbox. Additionally, we evaluate two other optical flow methods: Lucas-Kanade [24], also using from the Matlab Computer Vision System Toolbox, and the method of Färneback [12], using OpenCV[2] with the mexopencv interface[3].

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

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

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

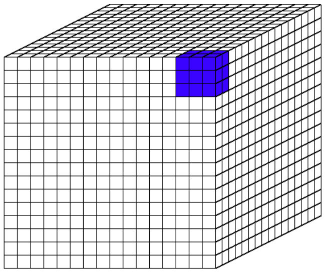

Figure 2: Blocks in a video volume can be reused for descriptor extraction. In our paper descriptors consist of 3 by 3 blocks in space and 2 blocks in time, shown in blue.

To aggregate responses over space we use the Matlab-friendly method proposed by [36]: 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 [36].

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 *Existing HOG/HOF Descriptors*

We use the existing implementation of Laptev et al. [21]. 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 [21] 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 four 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, and the Fisher Vector [27].

For both k-means and hierarchical k-means we use the implementation made available by VLFeat [37]. In both cases 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 features 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 [36], 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 [15] as used in [27] 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 Kernel representation. We follow [27] 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 [27]. Without a spatial pyramid [22], for our 72 dimensional HOG/HOF features this will yield vectors of 9,216 and 36,864 dimensions respectively. While not comparable with the dimensionality of other methods, Fisher Vectors 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 feature vectors.

We use the Spatial Pyramid [22] 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. [7, 22, 36, 37]). 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 [25]. For the Fisher Vector, we use linear SVMs. For both types of SVMs, we make use of the publicly available LIBSVM library [5] and the fast Histogram Intersection classification of [25].

4. EXPERIMENTS

Our baseline consists of densely sampled HOG and HOF descriptors, both consisting of blocks of 8 by 8 pixels by 6 frames. For HOF, 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 $1 \times 3 \times 1$ [22] (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 four different visual word assignment methods: k-means, hierarchical k-means, Random Forests, and the Fisher Kernel. (2) We compare our densely extracted descriptors with the descriptors provided by Laptev et al. [21]. (3) We evaluate the efficiency/accuracy trade-off by subsampling video frames for the descriptor extraction process. (4) For HOF-descriptors, we compare three different optical flow implementations: Horn-Schunk, Lukas-Kanade, and Farnebäck [12].

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 [28]. 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, and the Fisher Kernel. 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 [27, 6]. 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.

The accuracy and computational efficiency for the various word assignment methods for both our HOG and HOF features are presented in Figure 3 and Table 1. The first thing to notice is that the Fisher Kernel with 256 clusters has the best accuracy of 0.765 for HOG and 0.795 for HOF, while taking about 4 seconds per video (per descriptor type). K-means has also good accuracy at 0.726 for HOG and 0.791 for HOF. However, the computational time is at 13 seconds per video more than three times slower. This means that the Fisher Kernel (with 256 clusters) for video classification is superior in both accuracy and efficiency compared to k-means. For computational efficiency, the Random Forest is by far the fastest and takes 0.11 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).

In terms of classification time per video, we measure 0.017 seconds per video when using the fast Histogram Intersection based classification for SVMs [25] for k-means, hk-means, and Random Forests. We measure 0.001 seconds per video for the linear classifier used on the Fisher Kernel 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 Kernel, which yields the most accurate results, and the Random Forest, which is the fastest.

4.3 Comparison with Laptev et al.

In this experiment we compare with the publicly available code from [21] with our implementation. We compare only to the dense sampling option as [39] has already proven that dense sampling outperforms the use of space-time interest points. Results are presented in Figure 4 and 5 and in Table 2.

The results show that for all settings there is a significant difference in accuracy between the dense implementation of [21] and our method. For the Fisher Kernel, HOG descriptors yield 0.670 accuracy for [21] and 0.765 accuracy for our implementation and HOF descriptors yield 0.725 accuracy for [21] and 0.795 accuracy for our implementation. These are accuracy increases of 9% and 7% respectively. Similar differences are obtained using Random Forests. 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 [21] 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 [21] generates around 90 descriptors per frame or around 17,500 descriptors

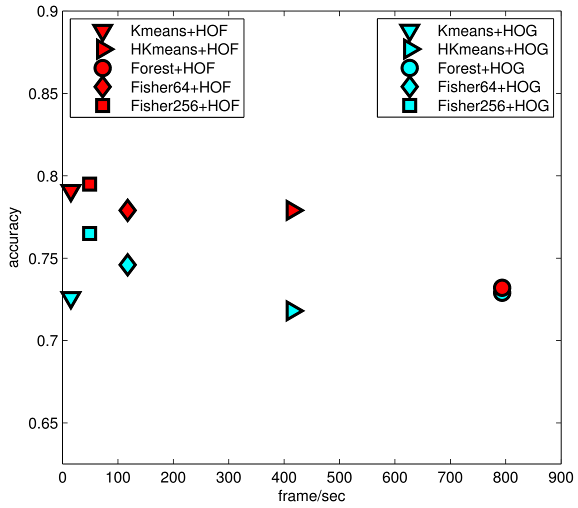

Figure 3: Accuracy/Efficiency trade-off for various word assignment methods.

	k-means	hk-means	RF	FK 64	FK 256
HOG Acc	0.726	0.718	0.729	0.746	0.765
HOF Acc	0.791	0.780	0.732	0.779	0.795
sec/video	13.08	0.47	0.11	1.67	3.99
frame/sec	15	415	1734	118	49

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.

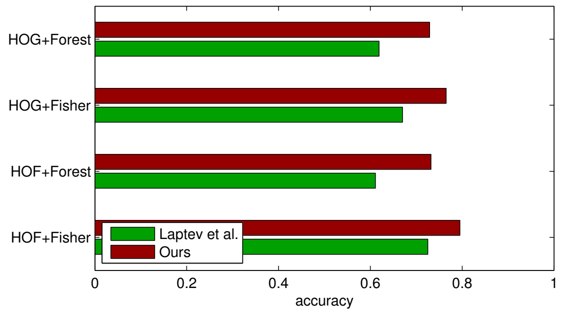

Figure 4: Accuracy comparison between [21] and our HOG/HOF descriptors

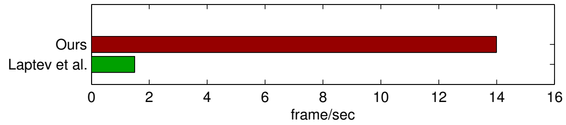

Figure 5: Computational Efficiency comparison between [21] and our HOG/HOF descriptors

	RF		FK 256		efficiency	
	HOG	HOF	HOG	HOF	sec/vid	frame/sec
[21]	0.619	0.611	0.670	0.725	132	1.5
ours	0.729	0.732	0.765	0.795	14	14

Table 2: Comparing the dense HOG/HOF implementation of [21] and ours. The descriptor extraction time is measured for extracting both HOG and HOF features, as the binary provided by [21] does always both. Descriptor extraction time is independent of the visual word assignment method (RF or FK 256).

per video, which means we generate 66% more descriptors. While this may seem unfair towards [21], 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 132 seconds per video while our method takes 14 seconds per video. Our method is faster because we reuse blocks in our dense descriptor extraction method. Note that because the method of [21] samples fewer descriptors, visual word assignment time is faster. But by using [21] 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 [21].

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.

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 the Random Forest and the Fisher Kernel: using the Fisher Kernel, a sampling rate of 1 yields an accuracy of 0.765 while a sampling rate of 6 yields 0.762 accuracy. The result of the Random Forest similarly goes slightly down from 0.729 to 0.720. In terms of computational efficiency, a significant speed-up is achieved: sampling every 6 frames instead of every frame gives a speed-up from 5.7 seconds per video to 1.8 seconds per video.

For HOF descriptors, subsampling has a bigger impact: For the Fisher kernel 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 a Random Forest is less affected and drops from 0.732 at sample rate of 1 to 0.722 at sample rate 6. Again, a good speed-up is obtained by subsampling. While descriptor extraction takes 8.3 seconds for sampling every frame, a sampling rate of 2 yields a factor 1.72 speed-up while sampling every 6 frames yields a factor 3.6 speed-up.

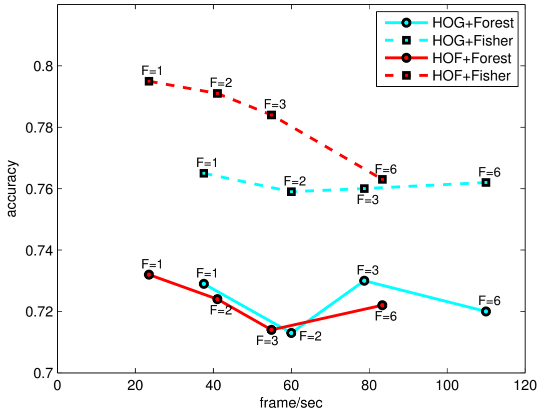

Figure 6: Trade-off accuracy/efficiency when varying sampling rate. F stands for frames per block and is directly related to sampling rate.

HOG	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
	RF FK 256	0.729 0.765	0.713 0.759	0.730 0.760	0.720 0.762
	sec/vid frame/sec†	5.7 37.6	3.3 60.0	2.5 78.7	1.8 110.0
HOF	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
	RF FK 256	0.732 0.795	0.724 0.791	0.714 0.784	0.722 0.763
	sec/vid frame/sec†	8.3 23.5	4.8 41.1	3.6 54.9	2.3 83.4

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.

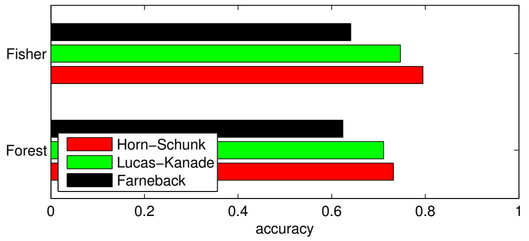

Figure 7: Classification Accuracy for various optical flow methods.

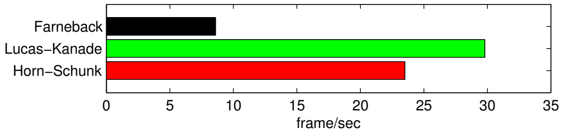

Figure 8: Computational efficiency for various optical flow methods

	Horn-Schunk	Lucas-Kanade	Färneback
RF FK 256	0.732 0.795	0.711 0.747	0.624 0.641
sec/video frame/sec	8.3 24	6.5 30	22.8 9

Table 4: Comparison of different optical flow methods. Horn-Schunk and Lucas-Kanade use the Matlab implementation. The dense Färneback method is taken from OpenCV.

To conclude, HOG descriptors can be sampled every 6 frames with negligible loss of accuracy yielding a speed-up of a factor 3.2. HOF descriptors can be sampled every 2 frames with negligible loss of accuracy yielding a speed-up of 1.72. When speed is more important than accuracy, HOF descriptors can also be sampled every 6 frames leading to 1-3% accuracy loss while gaining a significant speed-up of a factor 3.6.

4.5 Choice of Optical Flow

The HOF descriptors are much more expensive to extract than the HOG descriptors. This is because calculating the optical flow is computationally expensive. Additionally, not much research has been done on how different methods of optical flow affect HOF descriptors. Therefore in this experiment we evaluate three available optical flow implementations to investigate both their computational efficiency and accuracy. In particular, we compare Farnebäck [12] from OpenCV using the mexopencv interface, and Lucas-Kanade [24] and Horn-Schunk [14] from the Matlab Computer Vision Systems Toolbox. Results are presented in Figure 7, Figure 8 and Table 4.

As can be seen, for both the Random Forest and the Fisher Kernel the optical flow methods have the same ranking in terms of accuracy. For the Fisher Kernel, 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, Farnebäck is the slowest at 9 frames per second, followed by Horn-Schunk at 24 frames per second, while the Lucas-Kanade implementation is best at 30 frames per second.

To conclude, the choice of optical flow method drastically influences the power of the resulting HOF descriptor. Considering the many recent advances in optical flow (see Sun et al. [35] for a good overview), it would be worthwhile to evaluate a larger variety of optical flow methods in the context of video classification. In this paper, the Horn-Schunk method has superior performance. While the Lucas-Kanade method

Method	Accuracy
Wang et al. [38] (2013)	0.856%
This paper	0.809%
Reddy et al. [28] (2012)	0.769%
Solmaz et al. [34] (2012)	0.737%
Everst et al. [11] (2013)	0.729%
Kliper-Gross et al. [19] (2012)	0.727%

Table 5: Comparison with the State-of-the-Art.

is faster, its accuracy/efficiency trade-off is poor: If computational efficiency is important, instead of using the Lucas-Kanade method which yields 0.747 accuracy at 30 frames per second, we recommend using Horn-Schunk with a frame sampling rate of 2, yielding 0.791 accuracy at 41 frames per second (Table 3).

4.6 Recommendations for Practitioners

Based on the results of the previous experiments, we can now give several recommendations when accuracy of 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. For word assignment method, the Fisher Kernel 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 2.3 seconds per video to compute. And while the Random Forest is four times faster than hierarchical k-means, the difference is only 0.36 seconds, 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. 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 (i.e. concatenating HOG and HOF). Hence in our recommendations we perform a late fusion with equal weights.

The final recommended pipelines are visualised in Figures 9 and 10. Both combine HOG and HOF descriptors. Our recommended pipeline for accuracy is able to process 13 frames per second (for video frames of 320 by 240 pixels) at an accuracy of 0.809 on UCF50. Our recommended pipeline for computational efficiency processes 39 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 5. This comparison is done in terms of accuracy only, as most compared methods evaluate accuracy only.

As can be seen, the method of [38] yields the best results. This method is a combination of Dense Trajectories and STIP features [21]. As our results are better than [21], 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 and HOF 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 and HOF descriptors. Our first contribution is a strong and fast Matlab implementation of densely sampled HOG and HOF 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 8 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 3 times faster. HOF descriptors can be subsampled every 2 frames with negligible loss in accuracy, being 1.7 times faster. When speed is essential, HOF descriptors may be subsampled every 6 frames.

For the HOF 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%. In the near future we plan to investigate several more optical flow methods on a larger variety of datasets.

Compared to the state-of-the-art, the Dense Trajectory method of [38] obtains better results. Nevertheless, the huge difference for the choice of optical flow methods suggest this would also influence dense trajectories. Furthermore, Dense Trajectories still benefit from a combination with normal HOG and HOF desciptors [17, 38]. Finally, comparisons with other recent methods on UCF50 shows that we provide a strong implementation of dense HOG and HOF descriptors to the community.

6. 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] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] D.J. Butler, J. Wulff, G.B. Stanley, and M.J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[5] 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`.

[6] 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.

[7] G Csurka, C R Dance, L Fan, J Willamowski, and C Bray. Visual Categorization with Bags of Keypoints.

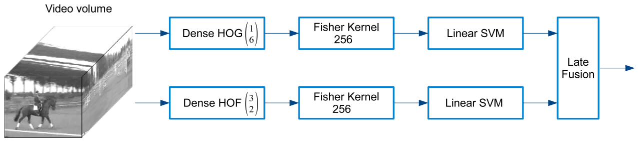

Figure 9: Recommended pipeline for accurate video classification. This pipeline yields an accuracy of 0.809 on UCF50 while processing 13 frames per second.

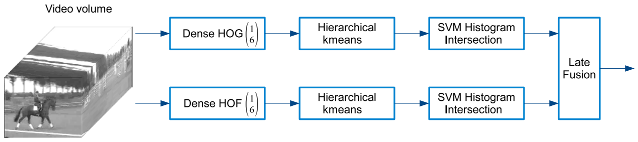

Figure 10: Recommended pipeline for realtime video classification. This pipeline yields an accuracy of 0.790 on UCF50 while processing 39 frames per second.

In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004.

[8] N Dalal and B Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[9] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.

[10] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[11] I. Everts, J. van Gemert, and T. Gevers. Evaluation of color STIPs for human action recognition. In *CVPR*, 2013.

[12] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, 2003.

[13] P Geurts, D Ernst, and L Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[14] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[15] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[16] F Jurie and B Triggs. Creating Efficient Codebooks for Visual Recognition. In *ICCV*, 2005.

[17] 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.

[18] A. Kläser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

[19] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, 2012.

[20] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011.

[21] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[22] S Lazebnik, C Schmid, and J Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006.

[23] D G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60:91–110, 2004.

[24] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, 1981.

[25] S Maji, A C Berg, and J Malik. Classification using Intersection Kernel Support Vector Machines is Efficient. In *CVPR*, 2008.

[26] 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.

[27] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *ECCV*, 2010.

[28] K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. In *Machine Vision and Applications*, 2012.

[29] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICIP*, 2004.

[30] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.

[31] J Sivic and A Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.

[32] A F Smeaton, P Over, and W Kraaij. Evaluation campaigns and TRECVID. In *ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, 2006.

[33] 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.

[34] B. Solmaz, S. Assari, and M. Shah. Classifying web videos using a global video descriptor. *Machine Vision and Applications*, 2012.

[35] D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2013.

[36] J R R Uijlings, A W M Smeulders, and R J H Scha. Real-time Visual Concept Classification. *IEEE Transactions on Multimedia*, 12, 2010.

[37] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM MM*, 2010.

[38] 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.

[39] H. Wang, M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.