2013 IEEE Conference on Computer Vision and Pattern Recognition

Sampling Strategies for Real-time Action Recognition

Feng Shi, Emil Petriu and Robert Laganière
School of Electrical Engineering and Computer Scinece
University of Ottawa, Ottawa, On, Canada
{fshi098, petriu, laganier}@site.uottawa.ca

Abstract

Local spatio-temporal features and bag-of-features representations have become popular for action recognition. A recent trend is to use dense sampling for better performance. While many methods claimed to use dense feature sets, most of them are just denser than approaches based on sparse interest point detectors. In this paper, we explore sampling with high density on action recognition. We also investigate the impact of random sampling over dense grid for computational efficiency. We present a real-time action recognition system which integrates fast random sampling method with local spatio-temporal features extracted from a Local Part Model. A new method based on histogram intersection kernel is proposed to combine multiple channels of different descriptors. Our technique shows high accuracy on the simple KTH dataset, and achieves state-of-the-art on two very challenging real-world datasets, namely, 93% on KTH, 83.3% on UCF50 and 47.6% on HMDB51.

1. Introduction

Action recognition in videos has recently been a very active research area due to its wide applications such as intelligent video surveillance, video retrieval, human-computer interaction and smart home. State-of-the-art approaches [23, 24, 30] have reported good results on human action datasets. Among all the methods, local spatio-temporal features and bag-of-features(BoF) representations achieved remarkable performance for action recognition. Laptev and Lindeberg [12] were first to introduce space-time interest point by extending 2D Harris-Laplace detector. Schüldt *et al*. [25] built on Harris corner detector with automatic scale selection to detect salient sparse spatio-temporal features. To produce denser space-time feature points, Dollár *et al*. [4] used a pair of 1D Gabor-filter to convolve with a spatial Gaussian to select local maximal cuboids. Willems *et al*. [33] proposed Hessian3D detector and extended SURF descriptor to detect relatively denser and computationally efficient space-time points. A recent trend is the use of dense sampled feature points [26, 32] and trajectories [30] for action recognition.

While impressive progress has been made, there are still some problems that need to be addressed. First, most existing action recognition methods use computationally expensive feature extraction, which is a very limiting factor considering the huge amount of data to be processed. Also, sparse interest point representations may miss important aspects of the scene and therefore do not generate enough relevant information for classification. In contrast, dense sampling methods can provide a very large number of feature patches and thus can potentially produce excellent performance. The best results are observed when the sampling step size decreases [30, 32]. However, the increase in the number of processed points adds to the computation complexity even if simplifying techniques are used, such as integral video and approximative box-filters.

Most interest point detectors used for action classification are extended from 2D space domain. They were originally designed for feature matching, not for selecting the most discriminate patches for classification. Interest point detectors [32] or selected features [14] by unsupervised learning have been shown to be very useful for simple KTH dataset [25] with single, staged human actions and uncorrelated backgrounds. We argue that it is more suitable to include the background information for real-life challenging datasets [10, 18, 22] because some of their background features are highly correlated with the foreground actions(*e.g.* diving with water background and skiing with snow background), and thus provide discriminative information for the foreground categories.

It should also be noted that the bag-of-features model only contains statistics of unordered features, and any information concerning temporal ordering and spatial structure is lost. A more discriminative method should include global structure information and ordering of local events.

To overcome these challenges, we proposed an efficient method for real-time action recognition. Inspired by the success of random sampling approach in image classification [21], we use random sampling for action recognition.

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.335

For computation efficiency, we use a Local Part Model [26] to extract features. This Local Part Model provides a better representation of spatio-temporal activities because it includes both structure information and ordering of local events. We evaluated our random sampling strategy combined with Local Part Model on publicly available datasets and show real-time performance and state-of-art accuracy.

2. Related works

A recent trend is to use dense sampling over sparse interest points for better performance. Dense sampling has shown to produce good results for image classification [1, 15]. For action recognition, Wang *et al*. demonstrate in [32] that dense sampling at regular space-time grids outperforms state-of-the-art interest point detectors. Similar results have also been observed in [26, 30]. Compared with interest point detectors, dense sampling captures most information by sampling every pixel in each spatial scale. However, such approaches are often computationally intractable for large video datasets.

Uniform random sampling [21], on the other hand, can provide performances comparable to dense sampling. A recent study [29] shows that action recognition performance can be maintained with as little as 30% of the densely detected features. Mathe and Sminchisescu also show similar results in [19]. Given the effectiveness of the uniform sampling strategy, one can think of using biased random samplers in order to find more discriminant patches. Yang *et al*. [34] are able to identify more features on the object of interest by using a prior distribution over patches of different locations and scales. Liu *et al*. [16] select the most discriminative subset from densely sampled features using the AdaBoost Algorithm. [19, 29] are based on the idea that eye movement of the human viewers is the optimal predictor of visual saliency. They measured the eye movement of human observers watching videos, and used the data to produce an "empirical" saliency map. By using such saliency maps, they pruned 20-50% of the dense features and achieved better results. The requirement of prior eye movement data renders such methods impractical for real applications. In addition, because of computational constraints, these methods didn't explore high sampling density schemes to improve their performance.

As for real-time action recognition algorithms, both Ke *et al*. [7] and Willems *et al*. [33] use approximative box-filter operations and integral video structure to speed-up the feature extraction. Patron-Perez and Reid [2] employ a sliding temporal window within the video and use first-order dependencies to effectively approximate joint distribution over feature observations given a particular action. Yeffet and Wolf [35] efficiently classify the actions with Local Binary Patterns and an approximate linear SVM classifier. Yu *et al*. [36] extend the efficient 2D FAST corner detector to the 3D domain V-FAST detector, and applied semantic texton forests for fast visual codeword generation.

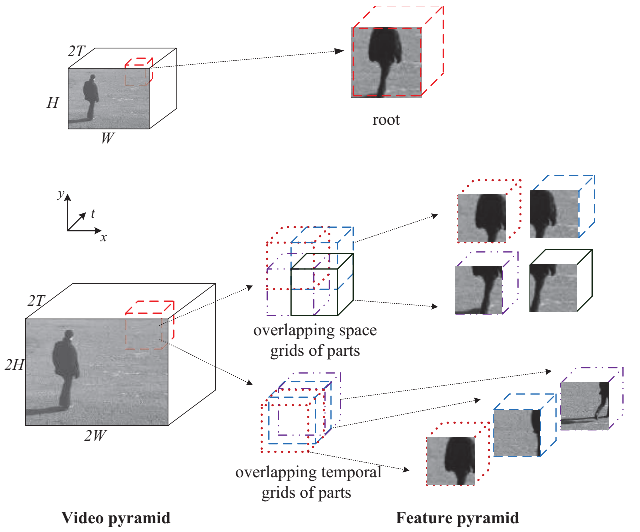

2T
H
W
root
y
t
x
2T
2H
2W
overlapping space grids of parts
overlapping temporal grids of parts
Video pyramid
Feature pyramid

Figure 1: Example of Local Part Model defined with root filter and overlapping grids of part filters.

3. Real-time action recognition approach

Our method builds on our previous work based on Local Part Model [26]. However, instead of using dense sampling, we focus on real-time applications by using random sampling. Nowak *et al*. [21] have shown that the most important factor governing performance is the number of patches sampled. While the performance of dense sampling is improved as the sampling step size decreases [30], such approach becomes rapidly computationally intractable due to the very large number of patches produced. To overcome such problem, our approach increases the sampling density by decreasing the sampling step size, and at the same time controls the number of sampled patches. In addition, we experimentally found that, with proper sampling density, state-of-the-art performance can be achieved by randomly discarding up to 92% of densely sampled patches.

3.1. Local Part Model

Inspired by the *multiscale, deformable part model* [5] for object classification, we proposed a 3D multiscale part model in [26]. However, instead of adopting deformable "parts", we use "parts" with fixed size and location on the purpose of maintaining both structure information and ordering of local events for action recognition. As shown in Figure 1, the local part model includes both a coarse primitive level *root* feature covering event-content statistics and higher resolution overlapping *part* filters incorporating structure and temporal relations.

Spatial resolution	Cuboid[4]	Dense[32]	Ours
80 x 60			767
160 x 120			4,295
360 x 288	44	643	27,950

Table 1: Average number of generated features per frame for different methods. Our numbers are based on a video length of 160 frames. The numbers of Cuboid and Dense are based on the report [32].

We found that, in addition to preserving local temporal context, this approach applies well to the context of dense sampling. Under the local part model, a feature consists of a coarse global *root* filter and several fine overlapped *part* filters. The *root* filter is extracted on the video at half the resolution. For every coarse *root* filter, a group of fine *part* filters are acquired from the full resolution video at the locations where the *root* filter serves as a reference position. For sampling, the dense sampling grid is determined by the *root* filter applied on half the spatial resolution of the processed video. At this resolution, it can achieve very high sampling density with far less samples. In addition, no drop in performance is observed due to this lower resolution because the fine-grained information is still included in the *part* filters.

3.2. Integral video and descriptors

Following the ideas of [7, 9, 33], we use integral video for fast 3D cuboid computation. With integral video, the volume of any 3D patch can be calculated from 8 additions, independent of the patch size. In our method, for each clip, we compute two integral videos, one for the *root* filter at half resolution, and another one for the *part* filters at full resolution. The descriptor of a 3D patch can then be computed very efficiently through 8 additions multiplied by the total number of *root* and *parts*. Apart from descriptor quantization, most cost associated with feature extraction is spent on accessing memory through the integral videos.

For computational efficiency consideration, we first test the HOG3D [9] descriptor. The HOG3D is based on simple spatio-temporal 3D gradients which are cheap to compute. We also test our method with HOG/HOF [13] and MBH [30] descriptors, which involve computationally expensive dense optical flow calculation. Therefore, we down-sampled the UCF50 and HMDB51 videos to half the spatial resolution for all our experiments. Since we use random sampling, no feature detection is required. The feature computation time is mainly spent on the feature description.

3.3. Sampling strategy

3.3.1 Dense Sampling Grid

We perform uniform random sampling on a very dense sampling grid. We follow the same multi-scale dense sampling grid as in [26] but with denser patches. A feature point is determined by 5 parameters (x, y, t, σ, τ). A 3D video patch centred at (x, y, t) is sampled with a patch size determined by the multi-scale factor (σ, τ). The consecutive scales are obtained by multiplying σ and τ by a factor of $\sqrt{2}$. In our experiments with HOG3D, we set the minimal spatial size to 16 x 16 pixels and minimal temporal size to 10 frames. With a total of 8 spatial scales and 2 temporal scales, we sampled the video 16 times.

A key factor governing sampling density is the overlapping rate of sampling patches. We explore very high sampling density with 80% overlap for both spatial and temporal sampling. Table 1 shows the comparison of the average number of generated features per frame for different methods. The features produced with cuboid and dense sampling in [32] are sampled from videos with resolution of 360 x 288 pixels. At same resolution, we generate 43 times more features than the dense sampling method in [32].

3.3.2 Random Sampling Strategies

For an image of size $n \times n$, the number of possible sampled patches is n^4 [11]. Nowak *et al*. have shown in [21] that the performance is always improved as the number of randomly sampled patches is increased with as many as 10000 points per image. For video recognition, such an approach would be computationally prohibitive. Therefore, we have to use some strategies to reduce number of sampled points per frame and at the same time maintain an adequate sampling density.

One solution is to do sampling at lower spatial resolution. As discussed above, the Local Part Model is well suitable for maintaining sampling density. By using it, the dense sampling grid is determined by the *root* filter, which is applied at half the resolution of the processed video.

As stated above, we use half the video resolution for UCF50 and HMDB51 in our experiments. Table 2 shows the average number of dense points (the third column) per video for different datasets. It also includes the average percentage of random samples *vs*. the total number of densely sampled points. For example, the average video size of HMDB51 is 182 x 120 pixels and 95 frames, and we randomly sample 10000 patches from the dense grid of 87,249 points. The dense grid is decided by *root* filter, which is performed on half the video size (91 x 60 pixels and 95 frames).

We randomly select 4000, 6000, 8000 and 10000 features for each video, and report the classification results for

Dataset	Avg. video size	Dense samples	10,000 samples
KTH	80 x 60 x 94	72,324	13.83%
UCF50	80 x 60 x 199	129,150	7.74%
HMDB51	91 x 60 x 95	87,249	11.46 %

Table 2: The sampling percentage of 10,000 random samples *vs*. total points (the third column) of dense sampling for different datasets.

each of them. They are chosen uniformly from the dense grid, so samples at finer scales predominate. We set the maximal video length to 160 frames. If the video is larger than 160 frames, we simply divide it into several segments, and select features at same rate for each segment. The 10k random samples represent 7.74-13.83% of the total dense points (depending on dataset). Our sampling density is much higher than [30, 32]. Also, compared with [29] which randomly discarded up to 60-70% of dense points, we obtain a much higher pruning rate.

4. Experiments

To demonstrate the performance of our sampling strategy, we evaluated our method on three public action benchmarks, the KTH [25], the UCF50 [22] and the HMDB51 [10] datasets. We randomly sampled 3D patches from the dense grid, and used them to represent a video with a standard bag-of-features approach. To generate codewords, we randomly selected 120,000 training features, and used k-means to cluster them into 4000 and 6000 visual words.

The sampled 3D patches are represented by descriptors, and the descriptors are matched to their nearest visual words with Euclidean distance. The resulting histograms of visual word occurrences are fed into a non-linear SVM with histogram intersection kernel [28]. For multi-class SVM, we used LIBSVM [3] one-versus-one approach implemented by max-wins voting.

To combine multiple channels of different descriptors, most methods use RBF-χ^2 kernel [31, 37]:

$$K(x_i, x_j) = \exp(-\sum_c \frac{1}{A^c} D(x_i^c, x_j^c)), \qquad (1)$$

where $D(x_i^c, x_j^c))$ is the χ^2 distances between the samples for the c-th channel, and A^c is the mean value of the χ^2 distance between the training samples for the c-th channel. While this approach produced "comparable results" [37], we argue that it is intuitive to add more weight to descriptors with higher classification power. For instance, in our experiments, the MBH outperform HOG on HMDB51 dataset by a large margin (43.0% *vs*. 21.0%). We propose a histogram intersection kernel for multi-channel classification:

$$K_{IH}(x_i, x_j) = \sum_c \frac{w^c}{max(w^c)} min(x_i^c, x_j^c), \qquad (2)$$

where w^c is classification accuracy for the c-th channel, which can be learnt from the training data. $max(w^c)$ is the maximal value from w^c of all channels.

One advantage of this approach is its computational efficiency. Given $w_i min(a, b) = min(w_i a, w_i b)$ for positive numbers, we can concatenate the weighted histograms of all channels, and use a single efficient intersection kernel SVM [17].

To compensate for the random sampling, we repeated every experiment 3 times, and report average accuracy and standard deviation over 3 runs.

4.1. Datasets

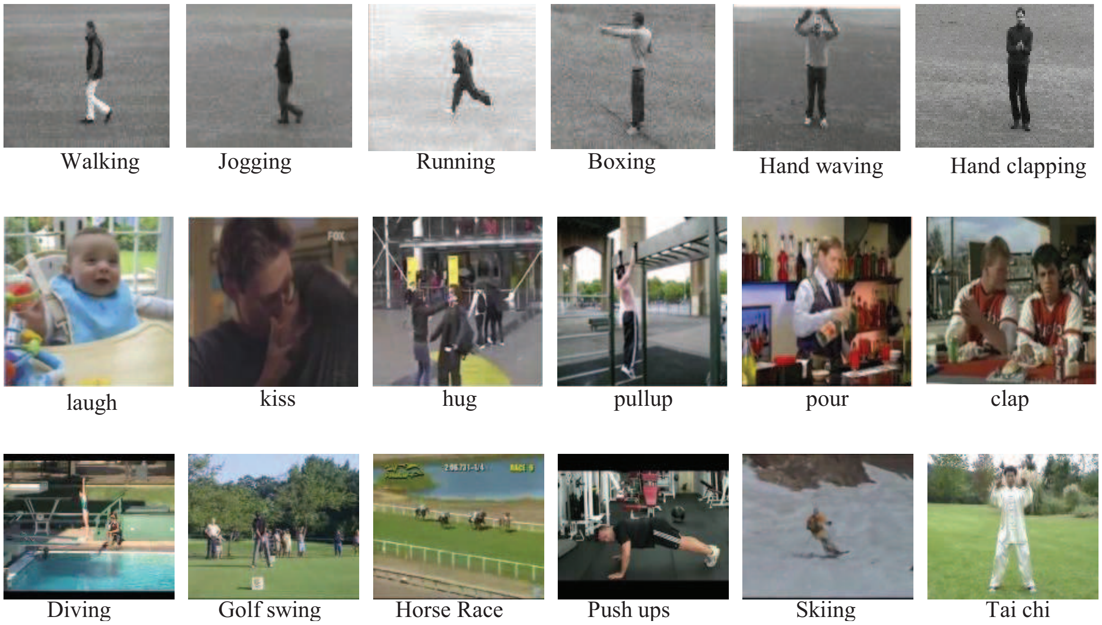
The **KTH** dataset [25] is an older dataset. It contains six action classes: walking, jogging, running, boxing, hand waving and hand clapping. Each action is performed by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. The background is static and homogeneous. In our experiment, we followed the experimental setup as those in [30, 32] by dividing the videos into testing set (2,3,5,6,7,8,9,10 and 22) and training set (the remaining subjects). We trained a non-linear SVM on training set and report the average accuracy over six classes on testing set.

The **UCF50** dataset [22] contains 50 classes and 6680 realistic videos taken from YouTube. The videos are grouped into 25 groups, where each group consists of a minimum of 4 action clips. The video clips in the same group may have similar background or be played by the same subjects. The dataset is very large and relatively challenging due to camera motion, cluttered background, large scale variations, etc. We report 5-Fold-Group-Wise Cross-Validation in Table 3 and Leave-One-Group-Out Cross-Validation in Table 4.

The **HMDB51** dataset [10] is by far the largest human action dataset with 51 action categories, with at least 101 clips for each category. It is perhaps the most realistic and challenging dataset. The dataset includes a total of 6,766 video clips extracted from Movies, the Prelinger archive, Internet, Youtube and Google videos. Three distinct training and testing splits have been selected from the dataset, with 70 training and 30 testing clips for each category. We used the original non-stabilized videos with the same three train-test splits as the authors [10], and report the mean accuracy over the three splits in all experiments.

4.2. Parameters

There are few parameters for our method, which determine the feature dimensions.



Walking Jogging Running Boxing Hand waving Hand clapping

laugh kiss hug pullup pour clap

Diving Golf swing Horse Race Push ups Skiing Tai chi

Figure 2: Sample of frames from KTH (first row), HMDB51 (second rom) and UCF50 (last row). The frames from KTH share same background for all categories. The cluttered background are shown on both HMDB51 and UCF50 datasets.

Local part model. The *root* filter of *local part model* is sampled from the dense sampling grid of the processed video at half the resolution. For each *root* patch, we sampled 8 (2 by 2 by 2) overlapping *part* filters from the full resolution video. Both *root* and *part* patches are represented with a descriptor. The histograms of 1 *root* patch and 8 *part* patches are concatenated as one local ST feature. Therefore, each feature is 9 times the dimension of a single descriptor.

HOG3D. We tested our method with three different HOG3D feature dimensions: 60, 96 and 144. At 60 dimension, the parameters are: number of histogram cells $M = 1$, $N = 3$; number of sub-blocks $2 \times 2 \times 2$; and polyhedron type icosahedron(20) with full orientation. At 96 dimension, the parameters are: number of histogram cells $M = 2$, $N = 2$; number of sub-blocks $2 \times 2 \times 3$ for KTH, $1 \times 1 \times 3$ for UCF50 and HMDB51; and polyhedron type dodecahedron(12) with full orientation. At 144 dimension, the parameters are: number of histogram cells $M = 2$, $N = 3$; number of sub-blocks $2 \times 2 \times 2$ for KTH, $1 \times 1 \times 2$ for UCF50 and HMDB51; and polyhedron type dodecahedron(12) with full orientation. For all cases, the cut-off value is $c = 0.25$. With one HOG3D descriptor at dimension of 60, 96 or 144, our local part model feature (with 1 *root* filter and 8 *part* filters) has a dimension of 540, 864 or 1296, respectively.

HOG, HOF and **MBH.** The minimal patch size is $16 \times 16 \times 14$ for HOG and $20 \times 20 \times 14$ for HOF and MBH. Each patch is subdivided into a grid of $2 \times 2 \times 2$. With 8 bins quantization, one descriptor of HOG, HOF , MBHx or MBHy has a dimension of 64 ($2 \times 2 \times 2 \times 8$). Our local part model feature has a dimension of 576 (9×64). For MBH, we simply concatenate MBHx and MBHy into a descriptor with size of 1152.

4.3. Results

Table 3 presents our experimental results with HOG3D descriptor for all three datasets. All the tests were run with the parameters listed in previous section. For each video, we randomly sampled 4000, 6000, 8000 and 10000 3D patches as features, and performed classification using a standard Bag-of-Feature approach with 4000 and 6000 codewords, respectively. The feature dimensions of 540, 864 and 1296 were tested. To compensate the sampling randomness, all the tests were run 3 times. The mean accuracy and standard deviation are given.

In general, the best performance is observed with 6000 codewords combined with 10000 features per video. The performance is almost always improved as the number of patches sampled from the video is increased. This is consistent with the results of random sampling for image classification [21]. Also, 6000 codewords give better result than 4000 codewords. For feature dimension, there is no clear performance advantage when using 1296 over 864. However, both of them show better results than a dimension of 540. In practice, it is preferable to use dimension of 864 for better computational efficiency without sacrificing the performance.

On both KTH and UCF50 datasets, the best result is achieved using 6000 codewords and 10000 sampled patches

Feature size	Samples	KTH		UCF50		HMDB51	
		4k words	6k words	4k words	6k words	4k words	6k words
540	4000	87.6% ± 0.33	87.9% ± 0.42	67.3% ± 0.30	67.2% ± 0.14	31.3% ± 0.67	31.6% ± 0.40
	6000	87.9% ± 0.85	88.8% ± 0.77	67.0% ± 0.22	67.8% ± 0.06	32.1% ± 0.31	32.0% ± 0.12
	8000	88.8% ± 0.61	88.1% ± 0.60	67.1% ± 0.18	67.6% ± 0.25	32.2% ± 0.16	32.8% ± 0.50
	10000	88.6% ± 0.42	88.8% ± 0.31	67.0% ± 0.21	67.6% ± 0.14	32.5% ± 0.28	32.6% ± 0.27
864	4000	91.9% ± 0.37	92.8% ± 0.24	69.8% ± 0.26	70.1% ± 0.45	33.0% ± 0.22	33.1% ± 0.21
	6000	92.4% ± 0.12	92.5% ± 0.12	70.2% ± 0.11	71.1% ± 0.34	33.8% ± 0.24	33.9% ± 0.59
	8000	92.7% ± 0.29	92.8% ± 0.13	70.1% ± 0.16	71.0% ± 0.20	34.2% ± 0.26	34.0% ± 0.42
	10000	92.7% ± 0.29	**93.0%** ± 0.29	70.0% ± 0.08	**71.7%** ± 0.18	34.7% ± 0.40	34.6% ± 0.01
1296	4000	91.7% ± 1.40	92.7% ± 0.31	70.0% ± 0.37	70.1% ± 0.30	33.8% ± 0.38	33.9% ± 0.20
	6000	91.9% ± 0.31	92.4% ± 0.27	70.7% ± 0.17	71.0% ± 0.08	34.0% ± 0.39	34.8% ± 0.48
	8000	92.0% ± 0.71	92.7% ± 0.41	70.2% ± 0.29	71.3% ± 0.15	34.7% ± 0.64	35.6% ± 0.28
	10000	92.4% ± 0.24	92.7% ± 0.18	70.4% ± 0.21	71.3% ± 0.27	34.8% ± 0.28	**35.6%** ± 0.25

Table 3: Average accuracy on all three datasets with 4000, 6000, 8000 and 10000 randomly sampled features per video. The table gives the mean and standard deviation over 3 runs with 4000 codewords and 6000 codewords, respectively. 5-fold group wise cross-validation is used for UCF50. Note: if video has more than 160 frames, more features are sampled at the same sampling rate as the first 160 frames.

at a feature dimension of 864. For HMDB51, the best result is obtained using 6000 codewords and 10000 samples at dimension 1296.

One very important observation from Table 3 is that all values of the standard deviation are very low. Such low standard deviation demonstrates effectiveness and consistency of our methods in spite of the random sampling. As discussed on Section 3.3.2, 10000 random samples represent 7.74-13.83% of the dense points. The consistency of the results at such high rate of randomness can be explained by the very high sampling density used by our approach.

4.4. Comparison to state-of-the-art

KTH is a simple dataset with six classes sharing the same homogeneous, uncorrelated backgrounds. The best performances have been obtained on the methods which focus on the foreground human motion. Such systems include feature point approaches [4, 25, 33] and template based approaches [23]. Because we randomly select the features, all parts of the scene have equal probability. The similarity of the background in all classes may reduce the classification power. This is consistent with conclusion in [37] that, for "easier" datasets, using foreground and background features together does not improve the performance for image classification. Nevertheless, we obtain 93%, which is better than the uniform dense sampling methods [13, 26, 32].

Table 4 shows the comparison of our method with the state-of-the-art. We use the parameters listed in Section 4.2. We concatenate MBHx and MBHy into a single MBH descriptor and use 6000 codewords. For other descriptors, we use 4000 codewords. For multiple channels, we combined all 4 descriptors (as shown in Table 4) with the proposed histogram intersection kernel. The HOG3D has a feature dimension of 864. The results show consistently good performance on HOG3D, HOF and MBH descriptors. We obtained 21.0%(HMDB51) on HOG. This is probably due to the reduction in resolution we applied, and we observed better result (24.8%) when using full size video.

On **HMDB51** and **UCF50**, we achieved classification results similar to the current state-of-the-art method [31], but by analysing the videos at half the resolution. Also, by randomly sampling cubic patches, we are able to use integral video to accelerate the processing. In case of [31], the use of curve trajectories limits them to use integral image only. Our approach is then significantly faster.

Our method demonstrates very good performance on large scale challenging datasets with more realistic scenarios. One possible explanation for such good performances on real life videos resides in our random sampling conducted on an extremely dense sampling grid. For 10000 patches per video on HMDB51, we have around 100 features per frame, which is similar as [31]. However, our sampling density is much higher because the sampling is performed on one quarter size of that in [31]. Compared with interest point detectors, we have more patches sampled from the test videos, and with uniform random sampling our method also includes correlated background information. Such background information may improve discriminative power for recognition on real-life videos.

Our parameters are optimized for real-time process at half the spatial resolution. We expect the performance to improve further with parameter tuning in the case of full resolution videos.

4.5. Computation efficiency

Table 5 summarizes average computation speed at different stages for HMDB51 dataset when using HOG3D de-

Feature size	Speed (frames per second)						
	Integral video	HOG3D & Sampling		Brute Force BoF matching		Total fps	
				4k words	6k words	4k words	6k words
540	65.57	384.48	cpu	84.00	56.06	34.54	29.61
			gpu	266.12	180.63	49.94	46.59
864	62.66	394.10	cpu	53.37	36.21	28.014	23.20
			gpu	166.24	113.37	45.72	41.23
1296	62.45	393.12	cpu	37.57	25.62	23.60	18.78
			gpu	113.62	76.19	41.51	35.30

Table 5: Average computation speed with four cores at different stages in frames per second for HMDB51 dataset. 10000 features are sampled. Note: the classification stage is not included.

Descriptor	Feature size	Speed (frames per second)						Mean accuracy
		Integral video	Sampling	Flann BoF matching		Total fps		4k words
				4k words	6k words	4k words	6k words	
MBH	1152	41.19	192.4	267.14	252.78	30.79	29.92	41.1% ± 0.23
HOG3D	864	71.88	159.60	290.81	282.60	42.22	41.69	33.3% ± 0.19

Table 6: Average computation speed with single core at different stages in frames per second for HMDB51 dataset. The 10K samples are used in the experiment, and the optical flow computation for MBH is included in "Integral video".

Method		HMDB51	UCF50
HMDB51 [10]		23.2%	47.9%
ActionBank [23]		26.9%	57.9%
MIP [8]		29.17%	72.68%
Subvolume [24]		31.53%	-
MRP [6]		40.7%*	-
GIST3D [27]		29.2%*	73.7%*
UCF50 [22]		27.02%*	76.90%*
Dense trajectories [31]		46.6%%*	**84.5%***
Ours	HOG	21.0%±0.28	58.6%±0.16
	HOF	33.5%±0.31	69.7%±0.12
	HOG3D	34.7%±0.40	72.4%±0.02
	MBH	43.0%±0.11	80.1%±0.39
	Combined	**47.6%**±0.29*	83.3%±0.15*

Table 4: Comparison of average accuracy on UCF50 and HMDB51 with state-of-the-art methods in the literature. Those marked with * are results with combined descriptors. Leave One Group Out Cross-validation is used for UCF50.

scriptor. Similar speed is observed on KTH and UCF50 datasets, on which we test with same spatial resolution. The computation time was estimated on an Intel i7-3770K PC with an AMD HD7770 GPU @1050 Hz. Our prototype is implemented in C++. The run-time estimates for "Integral video" and "HOG3D & random sampling" were obtained on CPU parallelized with OpenMP. The speed for integral video varies little, and it only depends on the size of videos. Because there is no feature detection for random sampling, the introduction of integral video has greatly improved the speed for random sampling and HOG3D. We simply use brute-force match to assign HOG3D descriptors to their closest visual words. Brute-force matching is the most time-consuming step, but very suitable for GPU computing. We test it with and without GPU. The results show that our system runs at over 30 frames per second with low feature dimensions by using only CPU and at high feature dimensions using GPU.

We also tested single core speed with the same parameters as the experiments in Table 4, but using FLANN [20] for bag of word matching instead of brute-force. The FLANN can speed up matching process by 40x at a price of 0.5% to 2% drop in final classification results. The detail results are listed in Table 6.

5. Conclusions

This paper has introduced a sampling strategy for efficient action recognition. We also proposed a histogram intersection kernel to combine multiple channels of different descriptors. We introduced the idea of using very high sampling density for efficient and accurate classification. Compared with existing methods, a major strength of our method resides in its very high computational efficiency. Our results show its effectiveness and efficiency on various descriptors, and achieves state-of-the-art on two realistic large scale datasets, UCF50 and HMDB51.

References

[1] A. Agarwal and B. Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, 2006. 2

[2] I. R. Alonso Patron-perez. A probabilistic framework for recognizing similar actions using spatio-temporal features. In *BMVC*, 2007. 2

[3] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011. 4

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proc. 2nd Joint IEEE Int Visual Surveillance and Performance Evaluation of Tracking and Surveillance Workshop*, pages 65–72, 2005. 1, 3, 6

[5] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, pages 1–8, 2008. 2

[6] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*, pages 425–438, 2012. 7

[7] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, volume 1, pages 166–173, 2005. 2, 3

[8] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, pages 256–269, 2012. 7

[9] A. Klser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, pages 995–1004, 2008. 3

[10] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011. 1, 4, 7

[11] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, pages 1–8. IEEE, 2008. 3

[12] I. Laptev and T. Lindeberg. Space-time interest points. In *Proc. Ninth IEEE Int Computer Vision Conf*, pages 432–439, 2003. 1

[13] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, pages 1–8, 2008. 3, 6

[14] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, pages 3361–3368, 2011. 1

[15] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43:2944, 2001. 2

[16] L. Liu, L. Shao, and P. Rockett. Human action recognition based on boosted feature selection and naive bayes nearest-neighbor classification. *Signal Processing*, pages 1521–1530, 2012. 2

[17] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, pages 1–8. IEEE, 2008. 4

[18] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, pages 2929 – 2936, 2009. 1

[19] S. Mathe and C. Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *ECCV*, pages 842–856, 2012. 2

[20] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340, 2009. 7

[21] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV (4)*, pages 490–503, 2006. 1, 2, 3, 5

[22] K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications Journal*, pages 1–11, September, 2012. 1, 4, 7

[23] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, pages 1234–1241, 2012. 1, 6, 7

[24] M. Sapienza, F. Cuzzolin, and P. H. T. and. Learning discriminative space-time actions from weakly labelled videos. In *ECCV*, 2012. 1, 7

[25] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR (3)*, pages 32–36, 2004. 1, 4, 6

[26] F. Shi, E. M. Petriu, and A. Cordeiro. Human action recognition from local part model. In *Proc. IEEE Int Haptic Audio Visual Environments and Games (HAVE) Workshop*, pages 35–38, 2011. 1, 2, 3, 6

[27] B. Solmaz, S. M. Assari, and M. Shah. Classifying web videos using a global video descriptor. *Machine Vision and Applications*, pages 1–13, 2012. 7

[28] M. J. Swain and D. H. Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991. 4

[29] E. Vig, M. Dorr, and D. Cox. Space-variant descriptor sampling for action recognition based on saliency and eye movements. In *ECCV*, pages 84–97, 2012. 2, 4

[30] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176, 2011. 1, 2, 3, 4

[31] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. Technical report, INRIA, 2012. 4, 6, 7

[32] H. Wang, M. M. Ullah, A. Klser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, pages 127–137, 2009. 1, 2, 3, 4, 6

[33] G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, pages 650–663, 2008. 1, 2, 3, 6

[34] L. Yang, N. Zheng, J. Yang, M. Chen, and H. Chen. A biased sampling strategy for object categorization. In *Proc. IEEE 12th Int Computer Vision Conf*, pages 1141–1148, 2009. 2

[35] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *ICCV*, pages 492–497, 2009. 2

[36] T.-H. Yu, T.-K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *BMVC*, pages 1–12, 2010. 2

[37] J. Zhang, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73:2007, 2007. 4, 6