Motion Interchange Patterns for Action Recognition in Unconstrained Videos

Orit Kliper-Gross[1], Yaron Gurovich[2], Tal Hassner[3], and Lior Wolf[2]

[1] The Weizmann Institute of Science, Israel
[2] Tel-Aviv University, Israel
[3] The Open University, Israel

Abstract. Action Recognition in videos is an active research field that is fueled by an acute need, spanning several application domains. Still, existing systems fall short of the applications' needs in real-world scenarios, where the quality of the video is less than optimal and the viewpoint is uncontrolled and often not static. In this paper, we consider the key elements of motion encoding and focus on capturing local changes in motion directions. In addition, we decouple image edges from motion edges using a suppression mechanism, and compensate for global camera motion by using an especially fitted registration scheme. Combined with a standard bag-of-words technique, our methods achieves state-of-the-art performance in the most recent and challenging benchmarks.

1 Introduction

Among all visual tasks that are virtually effortless for humans, Action Recognition remains a clear challenge for Computer Vision. As real-world object recognition, face identification, image similarity and visual navigation systems demonstrate improved capabilities and become useful in many practical scenarios, the performance of video-based Action Recognition systems seem too fragile to leave the research lab.

Much of the advancement in the other Computer Vision domains has been attributed to the emergence of better image descriptors, the better use of statistical learning methods, and the advent of large and realistic benchmarks that help push performance boundaries. In Action Recognition too, as detailed in Section 2, many contributions have been made in each of these directions. Paradoxically, and somewhat in contrast to other visual perception tasks, new benchmarks seem to play a disillusioning role and clearly demonstrate the shortcomings of current Action Recognition techniques. This is especially true for inferring human action in unconstrained videos. In such videos, actors' identity and clothes, scene background and illumination, camera viewpoint and motion, and image resolution and quality all change between the various clips of the same action, leading to a significant drop in performance.

An emerging trend due to the difficulty of recognizing action in videos is the use of 3D sensors for Action Recognition. Such recognition systems heavily rely on identifying the 3D pose of the various body parts at each frame. While such

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 256–269, 2012.

approaches show clear value, the 3D sensors and constrained imaging conditions limit their successful use beyond entertainment and human-machine interactions. Specifically, such systems cannot index or annotate conventional video data sets, monitor surveillance footage, or identify motion from a far range.

In this work we rethink the description of actions and propose a comprehensive solution[1]. Key to our method is the focus on encoding motion interchanges, i.e., the creation of a signature that captures at every time point and at every image location both the preceding motion flow and the next motion element. This is done using a patch-based approach (a.k.a self-similarity) and local pattern encoding. To decouple static image edges from motion edges, we incorporate a unique suppression mechanism, and to overcome camera motion, we employ a motion compensation mechanism that is tailor-made to our descriptors. A bag-of-words approach is then used to pool this information from the entire video clip, followed, when appropriate, by a learned metric technique that mixes and reweighs the various features.

2 Related Work

Action Recognition is a central theme in Computer Vision and existing work is only briefly covered here. For more comprehensive surveys we refer to [1, 2].

Over the years Action Recognition methods have been designed to employ information ranging from high-level shape representations, to low-level appearance and motion cues. Several early attempts relying on high-level information, include explicit models of human bodies in motion [3], silhouettes [4] or 3D volumes [5]. A recent paper by [6], turned back to these early attempts and constructed a bank of action templates used to create a high-dimensional representation for effective recognition in several challenging Action Recognition data sets. In recent years, however, three general low-level representation schemes have been central in Action Recognition systems. These are the local descriptor, optic flow, and dynamic-texture based representations.

Local Descriptors. These methods begin by seeking coordinates of space-time interest points (STIP) [7]. The local information around each such point is then represented using one of several existing or adapted feature point descriptors. A video is then represented using, for example, a bag-of-words representation [8]. Some recent examples of such methods include [9–11]. This approach has proven effective on a number of recent, challenging data sets (e.g., [12]), yet one of its chief drawbacks is the reliance on a suitable number of STIP detections in each video; videos supplying too few (e.g., videos of subtle motion) may not provide enough information for recognition. Videos with too much motion (e.g., background, textured motion such as waves in a swimming pool) may drown any informative cues for recognition.

[1] The source code for our method and the scripts used to run it on the various benchmarks are available at: `http://www.openu.ac.il/home/hassner/projects/MIP/`

Optical-Flow Based Methods. These methods first estimate the optical-flow between successive frames [13, 14], sub-volumes of the whole video [15], or surrounding the central motion [16, 17]. Optical-flow, filtered or otherwise, provides a computationally efficient means of capturing the local dynamics in the scene. Aggregated either locally (e.g, [16]) or for whole video volumes as in [13] such information has been shown to provide strong cues to Action Recognition. As we later discuss (Section 4) optical-flow based methods commit early-on to a particular motion estimate at each pixel. Unreliable or wrong flow estimates would therefore provide misleading information to any subsequent processing.

Dynamic-Texture Representations. These methods evolved from techniques originally designed for recognizing textures in 2D images, by extending them to time-varying "dynamic textures" (e.g., [18] and recently also [19]). The Local Binary Patterns (LBP) [20], for example, use short binary strings to encode the micro-texture centered around each pixel. A whole 2D image is represented by the frequencies of these binary strings. In [18, 21] the LBP descriptor was extended to 3D video data and successfully applied to facial expression recognition tasks.

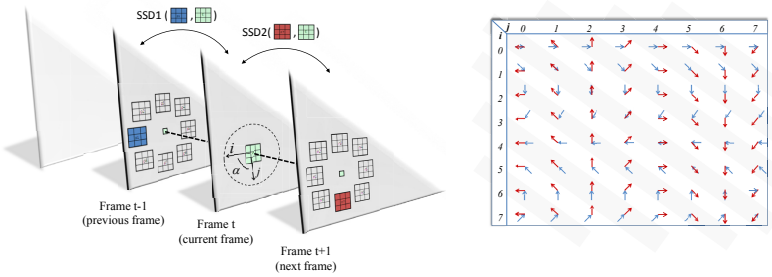
Another LBP extension to videos, related to our own work here, is the Local Trinary Patterns (LTP) descriptor of [22]. To compute a pixel's LTP code, the 2D patch centered on it is compared with 2D patches uniformly distributed on two circles, both centered on its spatial coordinates: one in the previous frame, and one in the succeeding frame. Three values are used to represent whether the central patch is more similar to one in the proceeding frame, the succeeding frame or neither one. A string of such values represents the similarities computed for the central patch with the patches lying on its two corresponding circles. A video is partitioned into a regular grid of non-overlapping cells and the frequencies of the LTP codes in each cell are then concatenated to represent the entire video.

3 Overview of the Method

Our method encodes a video clip as a single vector. While most real-world scenarios require the detection and recognition of human action in an unsegmented video, we adhere to the conventions of the community and assume known starting and end frames. Note that since our encoding, processing and pooling are all local, our system can be applied to unsegmented videos with few re-computations.

Given an input video, we encode every pixel on every frame by eight strings of eight trinary digits each. Each digit compares the compatibility of two motions with the local patch similarity pattern: one motion in a specific direction from the previous frame to the current frame, and one motion in a different direction from the current frame to the next one. A value of -1 indicates that the former motion is more likely, 1 indicates that the latter is more likely. A values of 0 indicates that both are compatible in approximately the same degree.

Going in multiple directions, we provide a complete characterization of the change from one motion to the next, hence we call our representations Motion Interchange Patterns (MIP). We provide details of our encoding in Sec. 4.



(a) (b)

Fig. 1. (a) Our encoding is based on comparing two SSD scores computed between three patches from three consecutive frames. Relative to the location of the patch in the current frame, the location of the patch in the previous (next) frame is said to be in direction i (j); The angle between directions i and j is denoted α. (b) Illustrating the different motion patterns captured by different i and j values. Blue arrows represent motion *from* a patch in position i in the previous frame; red for the motion *to* the patch j in the next frame. Shaded diagonal strips indicate same α values.

Some motion patterns are implausible, i.e., it should not happen that motion i from the previous frame to the current one is more likely than motion j from the current frame to the next one, and at the same time, using other patch comparison, motion j is more likely than motion i. However, since inferring motion locally from patch similarities is inherently ambiguous, this situation is common. We analyze this scenario and suggest a remedy in Section 5.

Another source of ambiguity stems from the camera motion which introduces image motion even in motionless parts of the scene. Recent Action Recognition benchmarks, which we aim to address, have significant amounts of such motion. Attempts to employ independent video stabilization showed only little improvement for our descriptors. Instead, we propose in Section 6 finding the alignment parameters that maximize the number of zero encoded pixels in the video.

Finally, in Section 7, we describe the pipeline we use in order to pool pixel-wise signatures from the entire video clip and represent the clip as a single vector. Specifically, we combine conventional bag-of-words techniques with suitable learning techniques to obtain discriminative action models.

4 Basic Encoding

Attempts to encode motion by first computing flow and then, at a separate stage, analyzing the motion seem intuitively appealing; in practice, however, they suffer greatly from an early commitment to an estimated flow. This situation resembles the one in object recognition, where gradient based methods and other "implicit-coding" techniques currently dominate, despite the appeal of methods that explicitly compute edges as a preliminary step.

The sum of squared differences (SSD) patch-comparison operator has long been used in estimating motion. The self-similarity technique [23] has demonstrated its usage in encoding both structure and motion, while being appearance invariant. Combining this line of reasoning with the effectiveness of LBP [20] in encoding texture for recognition tasks, LTP [22], and also patch-LBP [24, 25], use SSDs as a basic building block for defining local patterns. Here, we generalize this idea and fully encode both motion and motion interchange. In addition, we add layers of processing, which overcome the inherent limitations of earlier encoding schemes.

Comparing two differently-located patches in a pair of nearby frames (obtaining one SSD score) provides a datum that is unscaled and inherently ambiguous. Adding a third patch in order to obtain another SSD score enables the comparison of two similarly scaled data points. Our encoding scheme is therefore based on three patches as depicted in Fig. 1(a).

Encoding is performed for every pixel in every triplet of frames (i.e., previous, current, and next frame). We consider 3×3 pixel patches[2], where the location of the center of the patch in the current frame is denoted $(0, 0)$, and eight possible locations in each of the previous and the next frames are denoted i and j (respectively) and numbered from 0 to 7. The eight index values correspond to center pixel locations of $(-4, 0)$, $(-3, 3)$, $(0, 4)$, $(3, 3)$, $(4, 0)$, $(3, -3)$, $(0, -4)$, and $(-3, -3)$. The angle between i and j is denoted as $\alpha = 0°, 45°, ..., 315°$.

We consider all combinations of i and j. Each pixel $p = (x, y, t)$, in the current frame t, is therefore represented by a 64-trit (trinary digit) code denoted by $S(p)$. Each trit $S_{i,j}(p)$ corresponds to a different combination of i and j patch location indices in the previous and next frames (respectively). We consider gray value images (discarding color information), in which the intensities are scaled between 0 and 255, and use a threshold of $\theta = 1296$. So, for example, a 3 gray-value difference matrix of a constant value 12 is on the verge of being discernible from a zero difference matrix. Denote by SSD1 (SSD2) the sum of squared differences between the patch in the previous (next) frame and the patch in the current frame. Each trit, $S_{i,j}(p)$, is computed as follows:

$$S_{i,j}(p) = \begin{cases} 1 & if \quad SSD1 - \theta > SSD2 \\ 0 & if \quad |SSD2 - SSD1| \leq \theta \\ -1 & if \quad SSD1 < SSD2 - \theta \end{cases} \tag{1}$$

In a complex enough scene a low SSD value can help track the motion of a patch. For example, a low value of SSD1, is interpreted as indicating that a patch moved from location i in the previous frame to the central location in the middle frame. Therefore, for specific values of i and j we can infer whether the motion from location i to the central location is more likely than the subsequent motion from the central location to location j.

Taken for all values of i and j, MIP compare all eight motions to the eight subsequent motions, obtaining a comprehensive characterization of the change

[2] Since we did not optimize for the encoding parameters, and to improve readability, we provide the explicit values of the various parameters whenever possible.

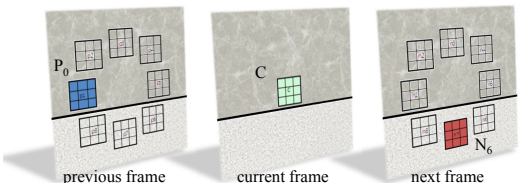


Fig. 2. Suppressing codes on static edges: A value of $S_{0,6}(p) = -1$ resulting from a static, horizontal edge in the scene. In this case, $S_{6,0}(p) = 1$. See text for more details.

in motion at each video pixel. This is obtained while refraining from deciding on the local motion between every two frames.

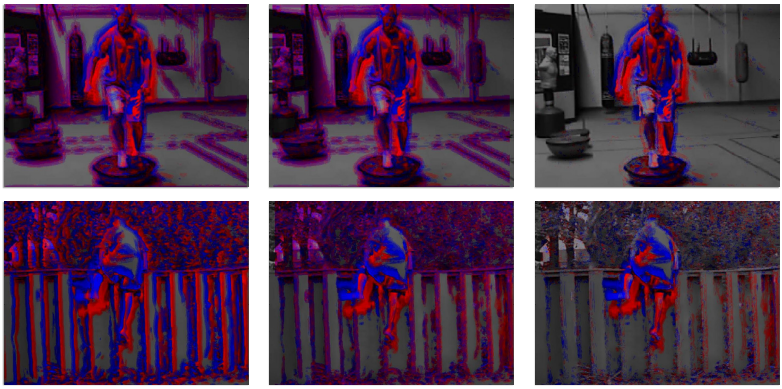
The 64 different motion combinations are illustrated in Fig. 1(b). For computational reasons they are stratified into 8 channels, each corresponding to a different value of α. In various stages of our pipeline, each channel is processed separately, considering each time only the relevant substring of 8 trits.

There is a number of significant differences between our method and LTP [22], however, at the encoding level they both share the use of three patches and two SSDs. Comparing the two methods, we note that LTP is restricted to the single channel of $\alpha = 0$ (the case of $i = j$ in Fig. 1(b)). Therefore, each LTP trit votes for motion in one of two opposite directions. MIP votes separately for motions *from* the previous frame and motions *to* the next frame by considering 8 $j's$ for each location i and vice versa. MIP therefore encodes local changes in motion in addition to the motion itself. As can be seen in our experiments, despite the success of LTP in encoding motion, its limited motion characterization is outperformed by the complete characterization employed by our method.

5 Distinguishing between Motion and Image Edges

It is instructional to consider the different trinary codes assigned for different motions and different choices of i and j. The value $S_{0,6}(p) = 0$, for example, would be obtained when a patch changed its position from frame $t-1$ to t by moving horizontally to the right and then from frame t to frame $t+1$ vertically downwards (Fig. 1(a,b)). Of course, a zero value will also be assigned in homogeneous regions where no motion can be detected. A value $S_{0,6}(p) = -1$, on the other hand, would be produced by a horizontal motion to the right followed by an unknown motion which is *not* vertically downwards. The value -1 could, however, alternatively indicate the presence of a static scene edge (see Fig. 2). These examples raise the obvious question of how to distinguish codes produced by meaningful motion from those caused by the scene structure?

To answer this question, assume that the value $S_{0,6}(p) = -1$ was obtained due to an edge in the scene. In other words, the patch in the previous frame, denoted P_0, resembles the current frame's patch C more closely than the patch in the next frame N_6. Now switch the locations of the patches in the previous and next frames, comparing C to P_6 and N_0 to compute $S_{6,0}(p)$. If no motion



(a) (b) (c)

Fig. 3. The effects of motion stabilization (Sec. 6) and suppression (Sec. 5), illustrated by color coding pixels by their 8-trit values. Top: stable camera scene; Bottom: moving camera. (a) **Raw codes:** In the top row, non-zero codes are produced along static edges in the scene despite there being no motion. In the bottom, codes are produced throughout the image due to both, image edges and camera motion. (b) **Stabilization:** Once the codes are stabilized, the image in the bottom shows codes along image edges. This step has no effect on the top image. (c) **Suppression:** Following suppression, non-zero codes concentrate almost entirely on the foreground action.

occurred, than P_6 will now contain the same intensities N_6 previously did, and vice versa. Consequently, the value of $S_{6,0}(p)$ will be 1.

The reasoning above suggests the following means for distinguishing between motion and still scene-edges: Once all 64 trits are computed, for each i and j, if $sign(S_{i,j}(p)) \neq sign(S_{j,i}(p))$ and both are non-zero, then suppress both trits by assigning $S_{i,j}(p) = S_{j,i}(p) = 0$. This, to indicate a high likelihood of no-motion. Following this suppression, homogeneous regions are also apparent: these are simply pixels with zero in all 64 trits. Code suppression is demonstrated in Fig. 3(c).

6 Overcoming Camera Motion

Camera motion is one of the most challenging aspects of performing Action Recognition "in-the-wild". We seek to compensate for any apparent motion caused by camera motion in a way that is compatible with- and suitable for-our encoding scheme. The compensation mechanism works by performing the following steps for each of the eight channels separately: (1) basic MIP encoding (Sections 4 and 5) (2) recovering image translation from the next frame to the previous frame through the current frame, using entire images; (3) a second MIP encoding using the warped next and previous frames (4) estimating an affine transformation on zero-coded pixels from steps 1 and 3; and (5) computing the final MIP codes on the warped frames.

In the second step, global image translations between the previous frame and the current frame, and between the current frame and the next frame, are computed using the direct method of [26]. Then, the next frame is warped to the previous frame using the recovered motion vectors. This step does not depend on the value of α and is common to all channels.

For each channel (fixed α value), we encode the obtained warped frames using our encoding scheme (including the suppression step of Section 5) and consider the zero encoded pixels of that channel, both before, and after warping by global translation. These pixels are part of image regions in which for the threshold θ used, no discernible motion was detected, and are therefore suitable candidates for background pixels.

We then compute an affine transformation between the current frame and the previous frame, and an affine transformation between the current frame and the next frame, considering only the zero encoded pixels defined above. The direct method of [26] is again used, this time with six degrees of freedom. Lastly, the next frame is warped to the previous frame and the final encoding is computed. Our stabilization is demonstrated in Fig. 3(b), bottom row.

7 Computing Similarity

Following the suppression step and the camera motion compensation step, we are left with 8 channel maps, where each pixel in the video is encoded by one 8-trit string per channel. Following the efficient representation presented in [22], we treat the positive and negative parts of the strings separately obtaining 2 UINT8 per pixel, for each of the 8 channels. These 16 values represent the complete motion interchange pattern for that pixel. For each channel, the frequencies of these MIP codes are collected in small 16×16 patches in the image to create 512-dimensional code words. Unlike [22], who represent the video by concatenating cells' histograms, we use instead a bag-of-words approach and by that we avoid the disadvantages of a grid (e.g., it *not* being translation invariant).

To this end, we employ k-means clustering on the small patches' histograms obtained for the training images. This is done separately per-channel, and we set each dictionary's size to be $k = 5000$ following [12]. Each train or test video clip is then represented by eight histograms (one per channel), in which each local string is assigned to the closest single dictionary element.

Each such histogram is normalized to have a sum of one, and then the square root is taken from each histogram element. In this manner, the L_2 distance between the processed histograms becomes the Hellinger distance between approximated distributions [24]. Needless to say, the design choices of our pooling scheme described here favor simplicity over performance, and the Computer Vision community has come up with more elaborate alternatives to each of these steps.

From each video clip, we thus obtain eight vectors of length $5,000$, which are denoted below as u_α. Depending on the actual task to be performed, these are further processed by employing a pipeline that combines existing learning

algorithms. These algorithms are chosen with accordance to the available training information and the required output of each task, as described next.

Action labeling. For multi-class Action Recognition, i.e., the standard task where during training multiple video clips per action label are provided, and during testing the task is to label each test sample, we use a one-vs-all linear Support Vector Machine (SVM). The linear SVM is learned with a parameter value of $C = 1$ on vectors u, each such u is a concatenation of the 8 u_α of all channels. For N labels there are N binary classifiers obtained in the one-vs-all scheme. Given a test sample, classification is obtained by choosing the class which classifies the test datum with greatest margin.

Action Pair-Matching. We additionally perform experiments in a pair-matching ("same/not-same") setup, which has recently gained popularity following the success of the Labeled Faces in the Wild benchmark [27]. Here, the input is given in the form of pairs of video clips, each labeled as portraying the same action or dissimilar (not-same) actions. The advantages of this setup include the fact that one does not have to explicitly define labels for actions and that the system trained in this manner can potentially identify new actions that were not seen during training.

In the training step we are given pairs of video clips, each with one binary label. We first encode each video clip as eight vectors u_α of length $5,000$ by employing the bag-of-words pipeline describe above. Then, using the training data, we learn eight transformations T_α by employing the CSML algorithm [28].

The CSML algorithm takes as input n pairs of vectors (v_i, v_i'), $i = 1..n$, and the corresponding same, $l_i = 1$, or not same, $l_i = -1$, labels. It minimizes the following cost function over the transformation matrix T:

$$\begin{array}{l} CSML(T, \{(v_i, v_i')\}, \{l_i\}) = \\ \sum_{\{i|l_i=1\}} CS(T, v_i, v_i') - \beta_1 \sum_{\{i|l_i=-1\}} CS(T, v_i, v_i') - \beta_2 ||T - I|| \ , \end{array} \quad (2)$$

with I the identity matrix, and the transformed cosine similarity is defined as: $CS(T, v, v') = \frac{(Tv)^\top (Tv')}{||Tv||\ ||Tv'||}$. The regularization parameter β_1 is set to one, and the parameter β_2 is optimized using the coarse to fine scheme suggested in [28].

Similarly to [28, 29], we first employ PCA (trained on a subset of the training data, and for each channel separately) to reduce the dimension of the input data from $5,000$ to 50. Let the PCA matrices be denoted as R_α. CSML is trained on the training data using pairs of vectors $(R_\alpha u_\alpha, R_\alpha u_\alpha')$ and the corresponding labels. The resulting transformation of size 30×50 is then applied, and each video is represented by $8 \times 30 = 240$ features vectors v obtained by concatenating $T_\alpha R_\alpha u_\alpha$ over all channels. A binary, linear SVM is then learned on the training data to map the 240 features of each pair of video clips to a binary decision of same/not-same. This is done by transforming each training pair representations v and v' to a single vector by performing point-wise multiplications of the two vectors ($v. * v'$, where $.*$ denotes point-wise multiplication). Given a test pair, PCA and then the CSML transformations are applied to each video clip, followed

by the application of the obtained SVM to the point-wise product of the resulting pair of feature vectors.

8 Experiments

We demonstrate the effectiveness of our method on a variety of Action Recognition benchmarks, focusing on recently published benchmarks of challenging, "in-the-wild" videos. As mentioned earlier, the various parameters are fixed and no real effort was made to optimize them; however, to demonstrate the contribution of the various parts of our method we also compare with partial variants of it.

ASLAN. The Action Similarity Labeling (ASLAN) collection [12] is a recent Action Recognition benchmark, which is modeled similarly to the Labeled Faces in the Wild face identification dataset [27]. It includes thousands of video clips collected from YouTube, depicting over 400 complex action classes. A "same/not-same" challenge is provided, which transforms the Action Recognition problem from a multi-class labeling task to a binary decision one. The goal is to answer the question of whether a pair of video clips presents the same action or not.

Results are reported as the average performance of ten separate experiments in a leave-one-split-out cross validation fashion. Each of the ten splits contains 300 pairs of same action videos and 300 not-same pairs. In each experiment, nine of the splits are used for training, and the tenth for testing. The ASLAN splits are mutually exclusive in the action labels they contain; if videos of a certain action appear in one split, no videos of that same action will appear in any other split. Note that we make sure to train all aspects of our system on the training data only, and so the dictionary is built (k-means) ten times, and similarly new PCA matrices and CSML transformations are learned per test split.

We compare multiple algorithms, including LTP [22] which is the most relevant to our own, and the STIP descriptors – HOG, HOF, and HNF [30], each one separately, and combined.

For the proposed MIP method, we show several variants in which some of the method's components were muted. First there are the single channel variants, in which the entire pipeline was applied to only one channel out of the eight. We display results for the $\alpha = 0$ channel, which is the one closest to LTP, to allow a direct comparison, as well as to the $\alpha = 1$ channel which is the best performing out of the eight. In addition, we also compare to variants in which the suppression mechanism of Section 5 is removed, a variant in which the global motion compensation mechanism of Section 6 is removed, and a variant in which both are removed.

Lastly, to demonstrate the need for a specially suited motion compensation mechanism we compute MIP, without the motion compensation component, on stabilized videos. The stabilized videos were produced using SSD based interest-point matching combined with RANSAC, as provided by Matlab's video stabilization routine.

Table 1 lists the resulting performance measures. Both aggregated Area Under the ROC Curve (AUC) and the average accuracy $\pm$ standard errors for the ten

Table 1. Comparison to previous results on the ASLAN benchmark. The average accuracy and standard error on the ASLAN benchmark is given for a list of methods (see text for details). All HOG, HOF, and HNF results are taken from [12, 29].

	No CSML		With CSML	
System	Accuracy	AUC	Accuracy	AUC
LTP [22]	55.45 ± 0.6 %	57.2	58.50 ± 0.7 %	62.4
HOG [30]	58.55 ± 0.8 %	61.59	60.15 ± 0.6 %	64.2
HOF [30]	56.82 ± 0.6 %	58.56	58.62 ± 1.0 %	61.8
HNF [30]	58.67 ± 0.9 %	62.16	57.20 ± 0.8 %	60.5
MIP single channel $\alpha = 0$	58.27 ± 0.6 %	61.7	61.52 ± 0.8 %	66.5
MIP single best channel $\alpha = 1$	61.45 ± 0.8 %	66.1	63.55 ± 0.8 %	69.0
MIP w/o suppression	61.67 ± 0.9 %	66.4	63.17 ± 1.1 %	68.4
MIP w/o motion compensation	62.27 ± 0.8 %	66.4	63.57 ± 1.0 %	69.5
MIP w/o both	60.43 ± 1.0 %	64.8	63.08 ± 0.9 %	68.2
MIP on stabilized clips	59.73 ± 0.77 %	62.9	62.30 ± 0.77%	66.4
MIP	62.23 ± 0.8 %	67.5	64.62 ± 0.8 %	70.4
HOG+HOF+HNF	60.88 ± 0.8 %	65.3	63.12 ± 0.9 %	68.0
HOG+HOF+HNF with OSSML [29]	62.52 ± 0.8 %	66.6	64.25 ± 0.7 %	69.1
MIP+HOG+HOF+HNF	64.27 ± 1.0 %	69.2	**65.45 ± 0.8 %**	**71.92**

splits are reported. As can be seen the proposed MIP method considerably outperforms all single feature methods, and even outperforms the application of the elaborate OSSML algorithm [29] to the three methods HOG, HOF, and HNF combined. Furthermore, MIP seems to be complimentary to HOG, HOF, and HNF, and combining MIP with these descriptors improves performance even further. The contribution of each component of the method, including the multiplicity of channels, the suppression mechanism, and the motion compensation mechanisms is also clear from the results. Specifically, for the motion compensation mechanism, it can be seen that standard alignment techniques perform poorly in improving recognition rates when applied to unconstrained videos; presumably due to misleading cues from multiple scene motions, low video quality etc.

HMDB51. The recent HMDB51 dataset [31] contains 51 distinct action categories, each represented by at least 101 video clips. The total 6,766 video clips were extracted from a wide range of varied sources. It is said to be one of the most challenging datasets of its kind, with performance levels in the low twenties.

The benchmark results on this dataset are evaluated using three distinct training and testing splits, each containing 70 training and 30 testing clips per action category. The splits are provided by the authors of [31] and were selected to display a representative mix of video quality and camera motion attributes. The dataset comes in two flavors: original and stabilized. Since our method contains its own motion compensation mechanism we test our method on the more challenging, original video collection.

The results are depicted in Table 2. Baseline results, taken from [31], include the combination of the HOG and HOF STIP systems [30], as well as the biologically-motivated Action Recognition system [32] based on a model of the human visual

Table 2. Comparison to previous results on the HMDB51 database. Since our method contains a motion compensation component, we tested our method on the more challenging unstabilized videos. Our method significantly outperforms all results obtained by previous work.

System	Original clips	Stabilized clips
HOG/HOF [30]	20.44%	21.96%
C2 [32]	22.83%	23.18%
Action Bank [6]	26.90%	N/A
MIP	**29.17%**	N/A

Table 3. Comparison to previous results on the UCF50 database. Our method significantly outperforms all reported methods.

System	splits	LOgO
HOG/HOF [30]	47.9%	N/A
Action Bank [6]	57.9%	N/A
MIP	**68.51%**	**72.68%**

cortex. The last was recently shown to perform on-par with humans in recognizing rodent behaviors [33]. We also compare to a recent contribution by [6]. As seen in this table, our system outperforms all reported results on this set.

UCF50. Another recent real-world dataset is the UCF50 data set[3], which includes videos of 50 actions collected from YouTube. It contains at least 100 videos per action category. With regards to action classes, there are 10 classes which overlap the HMDB51 dataset; however, HMDB51 spans beyond YouTube, including, e.g., motion pictures, and is therefore claimed to be more varied.

Following the recommendation of the UCF50 authors, we use a Leave-One-group-Out (LOgO) cross validation scheme, since each class in this dataset is divided into 25 homogeneous groups. In this setting, our system, without any tuning or modification, achieves 72.68% accuracy (chance level 2%). To be able to compare our results to the only two previously reported results on these set, we have conducted also a 10-fold cross validation test, where each time 9 random groups out of the 25 were used as test and the remaining 16 as training data, in this setting we achieved accuracy of 68.51%, which significantly outperform other methods on this set. See Table 3 for comparison.

KTH. The KTH dataset is an older dataset, on which many results have been reported, many of which using different testing protocols. It contains sequences from six classes: walking, jogging, running, boxing, clapping, and hand-waving. The camera in this dataset is static, and several previous contributions have used an attention mechanism in order to focus on the moving parts. We employ no such mechanism.

To report results we follow the protocol of [34], where the sequences of eight subjects are used for training, eight other subjects reserved for validation are not used in our system, and the actions of nine subjects are used for testing. On this older, artificial, and perhaps more limited benchmark our system achieves 93% accuracy which is on par with results reported by leading contributions that apply similar setting (for example, [9, 14, 22]).

[3] Available from `http://vision.eecs.ucf.edu/data/UCF50.rar`

9 Conclusion

Action Recognition in unconstrained videos remains a challenging problem in spite of much research. Here, based on the observation that local patterns of patch similarities can encode motion direction and motion change, we propose a novel Action Recognition pipeline, which also includes components that enable the decoupling of shape from motion, and compensate for camera motion in a manner tailored for the encoding scheme. Tested on the most realistic and challenging Action Recognition benchmarks, our method outperforms all reported methods. The modular structure of our system enables further improvements, such as the addition of learning-based hierarchical encoding layers, using recently proposed methods [9, 35].

References

1. Poppe, R.: A survey on vision-based human action recognition. IVC 28 (2010)
2. Turaga, P., Chellappa, R., Subrahmanian, V., Udrea, O.: Machine recognition of human activities: A survey. CSVT 18 (2008)
3. Yamato, J., Ohya, J., Ishii, K.: Recognizing human action in time-sequential images using hidden markov model. In: CVPR (1992)
4. Cheung, K., Baker, S., Kanade, T.: Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In: CVPR (2003)
5. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. TPAMI 29 (2007)
6. Sadanand, S., Corso, J.: Action bank: A high-level representation of activity in video. In: CVPR (2012)
7. Laptev, I.: On space-time interest points. IJCV 64 (2005)
8. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
9. Kovashka, A., Grauman, K.: Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In: CVPR (2010)
10. Wang, H., Klaser, A., Schmid, C., Liu, C.: Action recognition by dense trajectories. In: CVPR (2011)
11. Liu, J., Yang, Y., Saleemi, I., Shah, M.: Learning semantic features for action recognition via diffusion maps. CVIU 116 (2012)
12. Kliper-Gross, O., Hassner, T., Wolf, L.: The action similarity labeling challenge. TPAMI 34 (2012)
13. Ali, S., Shah, M.: Human action recognition in videos using kinematic features and multiple instance learning. TPAMI 32 (2010)
14. Schindler, K., Van Gool, L.: Action snippets: How many frames does human action recognition require? In: CVPR (2008)
15. Ke, Y., Sukthankar, R., Hebert, M.: Efficient visual event detection using volumetric features. In: ICCV (2005)
16. Efros, A., Berg, A., Mori, G., Malik, J.: Recognizing action at a distance. In: ICCV (2003)
17. Fathi, A., Mori, G.: Action recognition by learning mid-level motion features. In: CVPR (2008)

18. Kellokumpu, V., Zhao, G., Pietikäinen, M.: Human activity recognition using a dynamic texture based method. In: BMVC (2008)
19. Hassner, T., Itcher, Y., Kliper-Gross, O.: Violent flows: Real-time detection of violent crowd behavior. In: CVPRW (2012)
20. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. TPAMI 24 (2002)
21. Zhao, G., Pietikäinen, M.: Dynamic texture recognition using local binary patterns with an application to facial expressions. TPAMI 29 (2007)
22. Yeffet, L., Wolf, L.: Local trinary patterns for human action recognition. In: ICCV (2009)
23. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: CVPR (2007)
24. Wolf, L., Hassner, T., Taigman, Y.: Descriptor based methods in the wild. In: ECCVW (2008)
25. Wolf, L., Hassner, T., Taigman, Y.: Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. TPAMI 33 (2011)
26. Irani, M., Anandan, P.: About direct methods. In: ICCVW (1999)
27. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, UMASS, TR 07-49 (2007)
28. Nguyen, H.V., Bai, L.: Cosine Similarity Metric Learning for Face Verification. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part II. LNCS, vol. 6493, pp. 709–720. Springer, Heidelberg (2011)
29. Kliper-Gross, O., Hassner, T., Wolf, L.: The one shot similarity metric learning for action recognition. In: SIMBAD (2011)
30. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)
31. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV (2011)
32. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: ICCV (2007)
33. Jhuang, H., Garrote, E., Mutch, J., Yu, X., Khilnani, V., Poggio, T., Steele, A., Serre, T.: Automated home-cage behavioral phenotyping of mice. Nature Communications 1 (2010)
34. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: ICPR (2004)
35. Le, Q., Zou, W., Yeung, S., Ng, A.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: CVPR (2011)