2013 IEEE International Conference on Computer Vision

Towards understanding action recognition

Hueihan Jhuang[1] Juergen Gall[2] Silvia Zuffi[3] Cordelia Schmid[4] Michael J. Black[1]

[1]MPI for Intelligent Systems, Germany [2]University of Bonn, Germany, [3] Brown University, USA, [4]LEAR, INRIA, France

Abstract

*Although action recognition in videos is widely studied, current methods often fail on real-world datasets. Many recent approaches improve accuracy and robustness to cope with challenging video sequences, but it is often unclear what affects the results most. This paper attempts to provide insights based on a systematic performance evaluation using thoroughly-annotated data of human actions. We annotate human **J**oints for the **HMDB** dataset (**J-HMDB**). This annotation can be used to derive ground truth optical flow and segmentation. We evaluate current methods using this dataset and systematically replace the output of various algorithms with ground truth. This enables us to discover what is important – for example, should we work on improving flow algorithms, estimating human bounding boxes, or enabling pose estimation? In summary, we find that high-level pose features greatly outperform low/mid level features; in particular, pose over time is critical. While current pose estimation algorithms are far from perfect, features extracted from estimated pose on a subset of **J-HMDB**, in which the full body is visible, outperform low/mid-level features. We also find that the accuracy of the action recognition framework can be greatly increased by refining the underlying low/mid level features; this suggests it is important to improve optical flow and human detection algorithms. Our analysis and **J-HMDB** dataset should facilitate a deeper understanding of action recognition algorithms.*

1. Introduction

Current computer vision algorithms fall far below human performance on activity recognition tasks. While most computer vision algorithms perform very well on simple lab-recorded datasets [31], state-of-the-art approaches still struggle to recognize actions in more complex videos taken from public sources like movies [14, 17]. According to [30], the HMDB51 dataset [14] is the most challenging dataset for vision algorithms, with the best method achieving only 48% accuracy. Many things might be limiting current methods: weak visual cues or lack of high-level cues for example. Without a clear understanding of what makes a method perform well, it is difficult for the field to make progress.

Our goal is twofold. First, towards understanding algorithms for human action recognition, we systematically analyze a recognition algorithm to better understand the limitations and to identify components where an algorithmic improvement would most likely increase the overall accuracy. Second, towards understanding intermediate data that would support recognition, we present insights on how much low- to high-level reasoning about the human is needed to recognize actions.

Such an analysis requires ground truth for a challenging dataset. We focus on one of the most challenging datasets for action recognition (HMDB51 [14]) and on the approach that achieves the best performance on this dataset (Dense Trajectories [30]). From HMDB51, we extract 928 clips comprising 21 action categories and annotate each frame using a 2D articulated human puppet model [36] that provides scale, pose, segmentation, coarse viewpoint, and dense optical flow for the humans in action. An example annotation is shown in Fig. 1 (a-d). We refer to this dataset as **J-HMDB** for "joint-annotated HMDB".

J-HMDB is valuable in terms of linking low-to-mid-level features with high-level poses; see Fig. 1 (e-h) for an illustration. Holistic approaches like [30] rely on low-level cues that are sampled from the entire video (e). Dense optical flow within the mask of the person (f) provides more detailed low-level information. Also, by identifying the person in action and their size, the sampling of the features can be concentrated on the region of interest (g). Higher-level pose features require the knowledge of joints (h) but can be semantically interpreted. Relations between joints (h) provide richer information and enable more complex models.

Pose has been used in early work on action recognition [3, 32]. For a complex dataset such as ours however, typically low- to mid-level features are used instead of pose because pose estimation is hard. Recently, human pose as a feature for action recognition has been revisited [10, 22, 26, 29, 34]. In [34], it is shown that current ap-

1550-5499/13 $31.00 © 2013 IEEE
DOI 10.1109/ICCV.2013.396

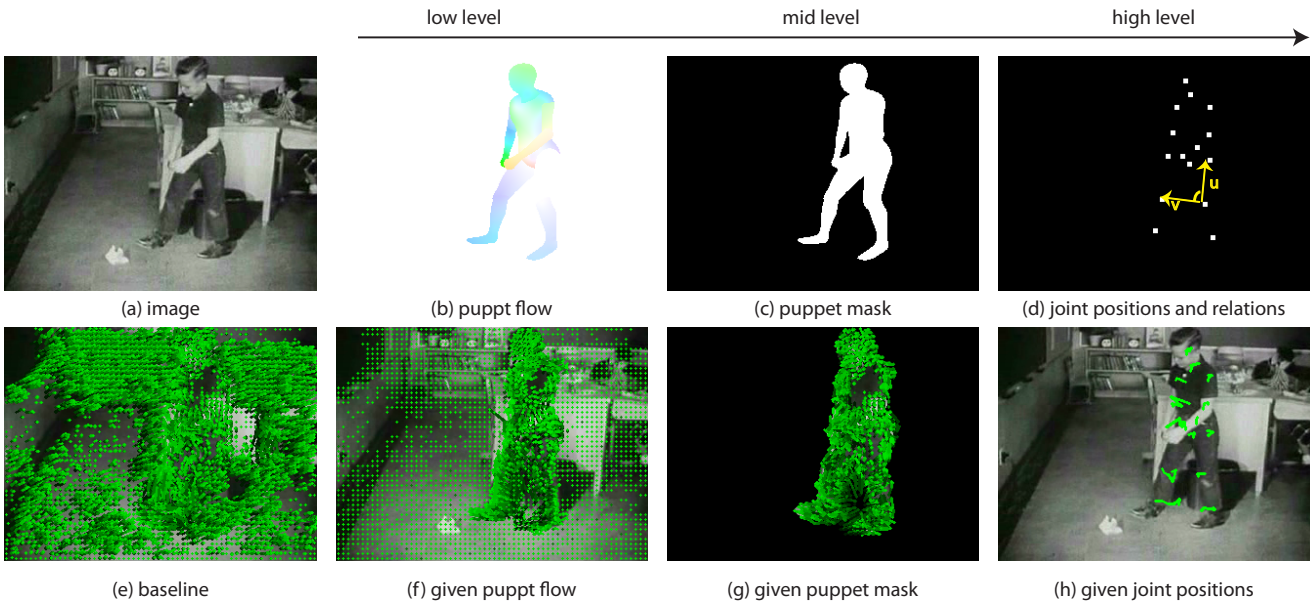

Figure 1. Overview of our annotation and evaluation. (a-d) A video frame annotated by a puppet model [36]. (a) image frame, (b) puppet flow [35], (c) puppet mask, (d) joint positions and relations. Three types of joint relations are used: 1) distance and 2) orientation of the vector connecting pairs of joints; *i.e.* the magnitude and the direction of the vector **u**. 3) Inner angle spanned by two vectors connecting triples of joints; *i.e.* the angle between the two vectors **u** and **v**. (e-h) From left to right, we gradually provide the baseline algorithm (e) with different levels of ground truth from (b) to (d). The trajectories are displayed in green.

proaches for human pose estimation from multiple camera views are accurate enough for reliable action recognition. For monocular videos, several works show that current pose estimation algorithms are reliable enough to recognize actions on relatively simple datasets [10, 26, 29], however [22] shows that they are not good enough to classify fine-grained activities. Using **J-HMDB**, we show that ground truth pose information enables action recognition performance beyond current state-of-the-art methods.

While our main focus is to analyze the potential impact of different cues, the dataset is also valuable for evaluating human pose estimation and human detection in videos. Our preliminary results show that pose features estimated from [33] perform much worse than the ground truth pose features, but they outperform low/mid level features for action recognition on clips where the full body is visible. We also show that human bounding boxes estimated by [2] and optical flow estimated by [27] do not improve the performance of current action recognition algorithms.

2. Related Studies and Datasets

Previous work has analyzed data in detail to understand algorithm performance in the context of object detection and image classification. In [20], a human study of visual recognition tasks is performed to identify the role of algorithms, data, and features. In [11], issues like occlusion, object size, or aspect ratio are examined for two classes of object detectors. Our work shares with these studies the idea that analyzing and understanding data is important to advance the state-of-the-art.

Previous datasets used to benchmark pose estimation or action recognition algorithms are summarized in Tab. 1. Existing datasets that contain action labels and pose annotations are typically recorded in a laboratory or static environment with actors performing specific actions. These are often unrealistic, resulting in lower intra-class variation than in real-world videos. While marker-based motion capture systems provide accurate 3D ground-truth pose data [12, 15, 19, 25], they are impractical for recording realistic video data. Other datasets focus on narrow scenarios [22, 28]. More realistic datasets for pose estimation and action recognition have been collected from TV or movie footage. Commonly considered sources for action recognition are sport activities [18], YouTube videos [21], or movie scenes [14, 16]. In comparison to sport videos, actions annotated from movies are much more challenging as they present real-world background variation, exhibit more intra-class variation, and have more appearance variation due to viewpoint, scale, and occlusion. Since HMDB51 [14] is the most challenging dataset among the current movie datasets [30], we build on it to create **J-HMDB**.

J-HMDB is, however, more than a dataset of human actions; it could also serve as a benchmark for pose estimation and human detection. Most pose datasets contain images of a single non-occluded person in the center of the image and

benchmark	examples	videos	actions	wild	pose
pose estimation	Buffy stickman [10]			y	y
	ETHZ PASCAL [8]			y	y
	H3D [2]			y	y
	Leeds Sports [13]			y	y
	VideoPose [24]	y		y	y
action recognition	UCF50 [21]	y	y	y	
	HMDB51 [14]	y	y	y	
	Hollywood2 [17]	y	y	y	
	Olympics [18]	y	y	y	
pose and action	HumanEvaII [25]	y	y		y
	CMU-MMAC [15]	y	y		y
	Human 3.6M [12]	y	y		y
	Berkeley MHAD [19]	y	y		y
	MPII Cooking [22]	y	y		y
	TUM kitchen [28]	y	y		y
	J-HMDB	y	y	y	y

Table 1. Related datasets.

the approximate scale of the person is known [8, 10, 13]. These image-based datasets constitute a very small subset of all the possible variations of human poses and sizes because the subjects are not performing actions, with the exception of the Leeds Sports Pose Dataset [13]. The VideoPose2 dataset [24] contains a number of annotated video clips taken from two TV series in order to evaluate pose estimation approaches on realistic data. The dataset is, however, limited to upper body pose estimation and contains very few clips. Our dataset presents a new challenge to the field of human pose estimation and tracking since it contains more variation in poses, humans sizes, camera motions, motion blur, and partial- or full-body visibility.

3. The Dataset

3.1. Selection

The HMDB51 database [14] contains more than 5,100 clips of 51 different human actions collected from movies or the Internet. Annotating this entire dataset is impractical so **J-HMDB** is a subset with fewer categories. We excluded categories that contain mainly facial expressions like smiling, interactions with others such as shaking hands, and actions that can only be done in a specific way such as a cartwheel. The result contains 21 categories involving a single person in action: *brush hair, catch, clap, climb stairs, golf, jump, kick ball, pick, pour, pull-up, push, run, shoot ball, shoot bow, shoot gun, sit, stand, swing baseball, throw, walk, wave*. Since we focus on and annotate the person in action in each clip, we remove clips in which the actor is not obvious. For the remaining clips, we further crop them in time such that the first and last frame roughly correspond to the beginning and end of an action. This selection-and-cleaning process results in 36-55 clips per action class with each clip containing 15-40 frames. In summary, there are 31,838 annotated frames in total. **J-HMDB** is available at http://jhmdb.is.tue.mpg.de.

3.2. Annotation

For annotation, we use a 2D puppet model [36] in which the human body is represented as a set of 10 body parts connected by 13 joints (shoulder, elbow, wrist, hip, knee, ankle, neck) and two landmarks (face and belly). We construct puppets in 16 viewpoints across the 360 degree radial space in the transverse plane. We built a graphical user interface to control the viewpoint and scale and in which the joints can be selected and moved in the image plane. The annotation involves adjusting the joint position so that the contours of the puppet align with image information [36]. In contrast to simple joint or limb annotations, the puppet model guarantees realistic limb size proportions, in particular in the context of occlusions, and also provides an approximate 2D shape of the human body. The annotated shapes are then used to compute the 2D optical flow corresponding to the human motion, which we call "puppet flow" [35]. The puppet mask (*i.e*. the region contained within the puppet) is also used to initialize GrabCut [23] to obtain a segmentation mask. Fig. 1 (b-d) shows a sample annotation.

The annotation is done using Amazon Mechanical Turk. To aid annotators, we provide the posed puppet on the first frame of each video clip. For each subsequent frame the interface initializes the joint positions and the scale with those of the previous frame. We manually correct annotation errors during a post-annotation screening process.

In summary, the person performing the action in each frame is annotated with his/her **2D joint positions**, **scale**, **viewpoint**, **segmentation**, **puppet mask** and **puppet flow**. Details about the annotation interface and the distribution of joint locations, viewpoints, and scales of the annotations are provided on the website.

3.3. Training and testing set generation

Training and testing splits are generated as in [14]. For each action category, clips are randomly grouped into two sets with the constraint that the clips from the same video belong to the same set. We iterate the grouping until the ratio of the number of clips in the two sets and the ratio of the number of distinct video sources in the two sets are both close to 7:3. The 70% set is used for training and the 30% set for testing. Three splits are randomly generated and the performance reported here is the average of the three splits. Note that the number of training/testing clips is similar across categories and we report the per-video accuracy, which does not differ much from the per-class accuracy.

4. Study of low-level features

We focus our evaluation on the Dense Trajectories (DT) algorithm [30] since it is currently the best performing

Figure 2. Comparison of various flow settings. The flow is numbered according to Tab. 2. See Sec. 4.2 and Sec. 5 for details.

method on the HMDB51 database [14] and because it relies on video feature descriptors that are also used by other methods. We first review DT in Sec. 4.1, and then we replace pieces of the algorithm with the ground truth data to provide low, mid, and high level information in Sec. 4.2, Sec. 5 and Sec. 6.2 respectively.

4.1. DT features

The DT algorithm [30] represents video data by dense trajectories along with motion and shape features around the trajectories. The feature points are densely sampled on each frame using a grid with a spacing of 5 pixels and at each of the 8 spatial scales which increase by a factor of $\frac{1}{\sqrt{2}}$. Feature points are further pruned to keep the ones whose eigenvalues of the auto-correlation matrix are larger than some threshold. For each frame, a dense optical flow field is computed w.r.t. the next frame using the OpenCV implementation of Gunnar Farnebäck's algorithm [9]. A 3×3 median filter is applied to the flow field and this denoised flow is used to compute the trajectories of selected points through the 15 frames of the clip.

For each trajectory, $L = 5$ types of descriptors are computed, where each descriptor is normalized to have unit L_2 norm: **Traj**: Given a trajectory of length $T = 15$, the shape of the trajectory is described by a sequence of displacement vectors, corresponding to the translation along the x- and y-coordinate across the trajectory. It is further normalized by the sum of displacement vector magnitudes, *i.e.* $\frac{(\Delta P_t, \ldots, \Delta P_{t+T-1})}{\sum_{j=t}^{t+T-1} ||\Delta P_j||}$, where $\Delta P_t = (x_{t+1} - x_t, y_{t+1} - y_t)$. **HOG**: Histograms of oriented gradients [5] of 8 bins are computed in a 32-pixels $\times$ 32-pixels $\times$ 15-frames spatio-temporal volume surrounding the trajectory. The volume is further subdivided into a spatio-temporal grid of size 2-pixels $\times$ 2-pixels $\times$ 3-frames. **HOF**: Histograms of optical flow [16] are computed similarly as HOG except that there are 9 bins with the additional one corresponding to pixels with optical flow magnitude lower than a threshold. **MBH**: Motion boundary histograms [6] are computed separately for the horizontal and vertical gradients of the optical flow (giving two descriptors).

For each descriptor type, a codebook of size $N = 4,000$ is formed by running k-means 8 times on a random selection of $M = 100,000$ descriptors and taking the codebook with the lowest error. The features are computed using the publicly available source code of Dense Trajectories [30] with one modification. While in the original implementation, optical flow is computed for each scale of the spatial pyramid, we compute the flow at the full resolution and build a spatial pyramid of the flow. While this decreases the performance on our dataset by less than 1%, it is necessary to fairly evaluate the impact of the flow accuracy using the puppet flow, which is generated at the original video scale.

For classification, a non-linear SVM with RBF-χ^2 kernel, $k(x, y)$, is used and L types of descriptors are combined in a multi-channel setup as $K(i, j) = \exp\left(-\frac{1}{L}\sum_{c=1}^{L}\frac{k(x_i^c, x_j^c)}{A^c}\right)$. Here, x_i^c is the c-th descriptor for the i-th video, A^c is the mean of the χ^2 distance between the training examples for the c-th channel. The multi-class classification is done by LIBSVM [4] using a one-vs-all approach. The performance is denoted as "baseline" in Tab. 2 (1), and the flow is shown in Fig. 2 (1).

4.2. DT given puppet flow

We can not evaluate the gain of having perfect dense optical flow, and therefore perfect trajectories. Instead, we use the puppet flow as the ground truth motion in the foreground, *i.e.* within the puppet mask (*pmask*). When the body parts move only slightly from one frame to the next, the puppets do not always move correspondingly because small translations are not easily observed and annotated. To address this, we replace the puppet flow for each body part that does not move with the flow from the baseline.

To evaluate the quality of the foreground flow, we set the flow outside *pmask* to zero to disable tracks outside the foreground. We compare optical flow (*of*) computed by Farnebäck's method and puppet flow (*pf*), as shown in Fig. 2 (2-3). Masking optical flow results in a 4 percentage points (*pp*) gain over the baseline, and masking puppet flow gives a 6 *pp* gain (Tab. 2 (2-3)). The gain mainly comes from *HOF* and *MBH*.

We dilate the puppet mask to include the narrow strip surrounding the person's contour, called *Dmask*. The width is scale dependent, ranging from 1 to 10 pixels with an average width of 6 pixels. Since the puppet flow is not defined outside the puppet mask, *of* is used on the narrow strip, as shown in Fig. 2 (4). Using *Dmask* increases the performance of (3) by 2.3 *pp* (Tab. 2 (4) *vs.* (3)). Comparing Fig. 2 (3) and (4), the latter has clear flow discontinuities caused

DT given low level features in Sec. 4

	Traj	HOG	HOF	MBH	ALL
1) baseline	40.0	32.9	40.1	51.1	**56.6**
2) *of pmask*	38.5	31.9	46.0	58.7	60.4
3) *pf pmask*	36.4	32.8	48.0	58.3	62.4
4) *pf Dmask*	38.0	32.2	46.4	60.8	64.7
5) *pf pmask* *of* outside *pmask*	43.0	36.1	44.1	63.6	65.3
6) 4) + 5)	**46.2**	35.2	**51.7**	**67.0**	**67.2**
7) 1) w. [27]	32.8	30.4	36.1	47.8	54.7
DT given mid level features in Sec. 5					
8) *bbox F*	38.5	34.9	42.2	51.1	58.5
9) *bbox Im*	42.7	46.9	44.5	57.0	62.2
10) *Dmask Im*	41.4	47.0	45.6	58.3	64.6
11) unit scale +*Dmask Im*	45.3	**52.1**	48.2	60.9	**66.0**
12) 8) w. [1]	37.7	33.9	39.0	52.2	56.7
DT given low + mid level features in Sec. 5					
13) 4) + 5) + 11)	51.3	49.4	54.4	68.7	**69.0**

Table 2. The impact of low and mid level feature modifications on **J-HMDB**. *of* and *pf* denote the optical flow computed by Farnebäck's method and puppet flow, respectively. *pmask* denotes the puppet mask and *Dmask* the dilated *pmask*. *F* and *Im* corresponds to masking in the feature space and in the image space, respectively. *bbox* is 20% larger in the x and y dimensions than the tightest box enclosing *pmask*.

by the difference of the motion around the person's contour and that of the surrounding background, suggesting that the motion boundary might be important for action recognition.

We further use *of* on the whole region outside *pmask* and *pf* within *pmask*, as shown in Fig. 2 (5), and use features within a bounding box that is 20% larger in the x and y directions relative to the tightest bounding box enclosing the puppet mask (*bbox*). This does not bring much overall gain over (4) but increases the performance of *Traj*, *HOG* and *MBH* (Tab. 2 (5) *vs*. (4)). We use features within *bbox* so that the result is comparable to Tab. 2 (2-4); *i.e*. only consider tracks/features in a subregion surrounding the foreground person. We also try to compute (5) with features from the whole frame. This results in a 5 *pp* gain over the baseline, with the main improvement coming from *MBH*.

Combing the kernel of Tab. 2 (4) and (5) results in a further boost of overall gain as well as a gain for each individual descriptor over both (4) and (5) (Tab. 2 (6)). It is now clear that the flow-related descriptors, *Traj*, *HOF* and *MBH* have a large gain (6.2-16 *pp*) over the baseline. This shows that the DT descriptors can indeed be improved with the ground-truth puppet flow.

At last, we replace the Farnebäck's flow with Classic+NL flow [27]. The flow is visually smoother than the baseline flow, as shown in Fig. 2 (7), but it is not clear whether this explains the slight drop of performance over the baseline (Tab. 2 (7)).

5. Study of mid-level features

Estimating the location and size of the human in action might be an easier task than estimating accurate pixel-wise flow. We therefore ask, without using the puppet flow, how helpful it is to know the region of interest, *i.e*. the image region in which the human in action occupies, and its size? In the section below, we only use Farnebäck's flow (*of*).

5.1. DT given foreground mask

We consider two types of regions of interest: the dilated puppet mask *Dmask* and *bbox* described above. We consider two ways of masking, one is in the feature space (*F*); *i.e*. compute flow/descriptors on the whole frame then only use those from within the mask. The other is to mask in the image space (*Im*) by setting the pixel values outside the mask to zero at every frame and then compute flow/descriptors, as shown in Fig. 2 (10). Masking features results in a slight 1.9 *pp* gain over the baseline for *bbox* (Tab. 2 (8)); using *Dmask* instead of *bbox* results in similar performance. Masking images results in a much higher gain: 5.6 and 8 *pp* for *bbox* and *Dmask* respectively (Tab. 2 (9-10)); in particular, it results in a much higher gain for *HOG* than masking features (Tab. 2 (9) *vs*. (8)). The reason that masking images performs better than masking features could be that the boundary of the image mask guides the optical flow algorithm to be more accurate around the contour of the person (Fig. 2 (10) *vs*. (1)). Not surprisingly, applying masks in all the cases boosts the performance of *HOG* because it only represents the texture of the foreground person. Note that when masking frames with *bbox*, flow has artifacts around boundaries, but this does not seem to decrease the performance much compared to masking with *Dmask*.

We also consider *bbox* from a human detector [1]. In 50% of the images, the overlap between the predicted box and the ground truth box exceeds 50%. Using the predicted boxes as above for the features does not improve the baseline (Tab. 2 (12)) and masking frames gives much worse results (34.7%). This suggests that the human detector in [1] is not accurate enough to help action recognition.

5.2. DT given scale

We resize all the frames as well as the corresponding *Dmask* such that all persons are around 200 pixels in height, and repeat the analysis in (10). This causes a slight 1.4 *pp* gain over (10), and the *HOG* alone has a 5 *pp* gain, suggesting that DT features are not perfectly scale invariant (Tab. 2 (11) *vs*. (10)). Finally, combining kernels of features relying on different low/mid level features results in a 12.4 *pp* gain over the baseline (Tab. 2 (13)).

It is interesting to see that for many paired comparisons, such as (5) *vs*. (6), (1) *vs*. (7), (10) *vs*. (11), the amount of performance change for an individual descriptor does not

always result in a similar amount of overall performance change, indicating that the features are not very complimentary, but have different error characteristics.

6. Study of high-level features

6.1. Pose features

For action recognition with pose features, we use various types of descriptors derived from joint annotations.

NTraj: For each frame, we have the x- and y-coordinates of 15 joints. We first normalize joint positions w.r.t the scale of the underlying puppet. We then use as features the translation of the normalized joint positions along the x and y-coordinates (dx, dy), the direction of the translational vector $(\arctan(\frac{dy}{dx}))$, and the relative positions of normalized joint positions w.r.t the puppet center in a sequence of T frames. Here T is the trajectory length described in Sec. 4.1. Note that due to the nature of the puppet annotation tool, all 15 joint positions are available even if they are not annotated when they are occluded or outside the frame. In this case, the joints are in the neutral puppet positions. Unless otherwise specified, we use all 15 joints regardless of their visibility. There are totally 75 descriptor types (30 for positions, 30 for translations, and 15 for directions). Note that unlike *Traj* in Sec. 4.1, we consider features along the x- and y- coordinate as separate descriptors, and this results in better performance than treating them as one descriptor. For *Traj*, translation is considered as the difference of positions between two adjacent frames along the trajectory. Here we use the differences between frame t and $t+s$; *i.e.* the feature of type f is a sequence $(f_{t+s} - f_t, .., .f_{t+ks} - f_{t+(k-1)s})$, $k = \frac{T-t}{s}$. The idea is that, for a small s, the trajectories might have jitter caused by imperfect annotation, and a larger s would reveal "true" motions; we compare $s = 1$ and 3.

NTraj+: Since it has been shown in [34] that relational features describing geometric relations between joints perform better than using normalized joint positions, we also extract a set of relational features: $C_2^{15} = 105$ distances between all the pairs of joints, 105 orientations of the vector connecting two joints, and $3 \times C_3^{15} = 1365$ inner angles spanned by two vectors connecting all the triples of joints, as shown in Fig. 1 (d). All possible relational features are computed for each frame, yielding 1575 descriptor types. In addition to using relational features, we also use the differences of relations between frame t and $t+s$ as described in **NTraj**. There are in total 3225 descriptor types (75 for *NTraj*, 1575 for relations, and 1575 for their difference).

For each descriptor type, all the training samples are used to generate a codebook. We compare several small codebook sizes, $N = 10$, 20 and 50, because each descriptor has a small dimensionality. The performance is similar and, hereafter, we report results of $N = 20$.

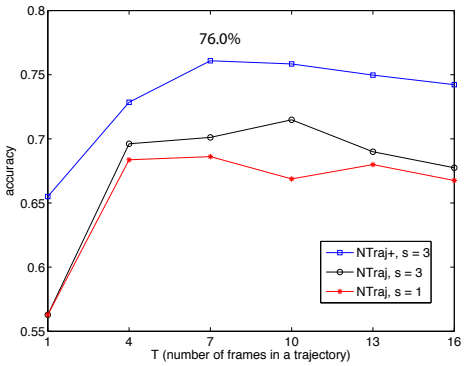

Figure 3. Performance of pose features as a function of the trajectory length T and the frame step size s, see Sec. 6.1.

Figure 3 shows the performance of the position-based *NTraj* and the position-and-relation-based *NTraj+* with respect to the trajectory length T and the frame step size s. It shows that a large step size ($s = 3$) results in higher accuracy and that having temporal information ($T > 1$) is very important although the trajectory length is not critical beyond $T = 7$ frames. It also shows that using relation features in addition to position-based features is key to increasing accuracy. Hereafter we report the performance of $s = 3$ and $T = 7$ for *NTraj+*; *i.e.* 76.0%.

To evaluate the sensitivity of the performance to the variance of joint positions, we add Gaussian noise to every joint in every frame; the noise has zero mean and the variance is $x \times$ (the distance to the closest joint), with $x \leq 1$. The rationale is that a joint, even not perfectly estimated or annotated, is unlikely to be confused with its nearby joints because of the limbs and torso connecting them. With the noise, the performance drop is less than 2 *pp*.

6.2. DT given joints

We also consider a sparse version of DT that tracks the 15 joint positions instead of tracking dense points. We use a smaller codebook size ($N = 100$) because here there are only 15 trajectories per frame. The trajectories are ordered to encode high-level pose information; *i.e.* there are 75 descriptor types (15 joints $\times$ 5 types in Sec. 4.1).

Since not all the joints are visible within a frame, we use a subset of **J-HMDB** that has all the joints inside the frame, denoted as sub-**J-HMDB**. The subset contains 316 clips distributed over 12 categories. The baseline performance on the subset is 10.6 *pp* lower than on the full set although the chance level of the former is lower (Tab. 3 (1) *vs*. Tab. 2 (1)). This suggests that the subset is more challenging, which could be because it contains only full body actions (*e.g.* kicking); these might exhibit richer variation in terms of appearance and optical flow than partial body actions (*e.g.* pour). Note that here we combine the texture features *HOG*, *HOF* and *MBH* into *HOX*. We also evaluate

	Traj	HOX	ALL	NTraj+	ALL+
1) baseline	36.4	45.2	**46.0**		
2) baseline w. low	37.5	54.4	54.0		
3) baseline w. low/mid	46.0	64.8	**63.2**		
4) baseline w. joints +*NTraj+*	51.0	59.4	**63.2**	75.1	75.5
5) 4) w. [33]	19.9	45.6	49.8	54.1	52.9

Table 3. The impact of high-level feature modifications on sub-**J-HMDB**. *ALL* is the combination of *HOX*/*Traj*. *ALL+* is the combination of *ALL*/*NTraj+*, see Sec. 6.2 for details.

DT given low and low/mid-level information as in Tab. 2 (6) and (13) respectively. The gain over the baseline is 8.0 and 17.2 *pp* respectively (Tab. 3 (2) and (3)).

We then compute the sparse version of DT with given joint positions. We firstly recognize that the overall accuracy is the same as DT given low/mid level information (*ALL* in Tab. 3 (4) *vs*. (3)). A closer look at the performance of individual descriptors reveals that the texture-based *HOX* benefits more given low/mid-level than high-level information, while the position-based *Traj* shows the opposite. This is consistent with the intuition that *HOX* relates more to low/mid level cues while *Traj* to high-level cues. We also observe that, using the same flow setting, the sparse *HOX* performs better than the dense *HOX* by 5 *pp* (Tab. 3 (4) *vs*. (2)). This suggests that representing texture around joints is not only more effective but also more discriminative than representing the texture in the whole frame.

We then evaluate the position-and-relation-based *NTraj+* on this subset (Tab. 3 (4)), the performance is similar to that on the full set (75.1% *vs*. 76.0%). It dramatically outperforms *Traj* by 24.1 *pp*, as well as the combination of *HOX* and *Traj* (*i.e*. *ALL*), showing that *the high-level pose feature derived from normalized joints positions and their relations is the best feature for action recognition.* While combining *HOX* and *Traj* improves performance, combining them with *NTraj+* does not increase the performance of the latter (*NTraj+ vs*. *ALL+* in Tab. 3 (4)), suggesting that *texture features do not add much additional information when the pose features are already thoroughly extracted.*

The subset sub-**J-HMDB** also allows us to evaluate the pose estimation algorithm from [33], which assumes the full body is visible. Using the error measurement in [7] with threshold 0.15, the pose estimation accuracy is 22.4%. There is no strong correlation between scale and accuracy but the correctly detected images mostly have people with non-occluded frontal views of upright poses. Dense Trajectories given estimated joints results in a 3.8 *pp* gain over the baseline, and *NTraj+* computed from the 15 estimated joint positions results in a 8.1 *pp* gain over the baseline (Tab. 3 (5)). This suggests that *while the estimated joint positions are not accurate compared to the ground truth, the derived pose features already outperform low/mid level features for action recognition.*

data	**J-HMDB**	sub-**J-HMDB**
baseline	56.6%	46.0%
baseline w. low/mid	69.0%	63.2%
baseline w. joints + *NTraj+*	NA	75.5%
high level pose (*NTraj+*)	76.0%	75.1%

Table 4. Overview of the recognition rate for both datasets.

6.3. Summary

Table 4 summarizes the improvements to Dense Trajectories realized by providing low/mid-level and high-level features on the full dataset **J-HMDB** and the subset sub-**J-HMDB**. Overall, the two sets show a 12-17 *pp* improvement over the baseline with ground truth low/mid features and a 19-29 *pp* improvement with high-level features.

7. Discussion

We have presented a complex, annotated, video dataset in order to analyze action recognition algorithms. Starting with a state-of-the-art method [30], we supply the algorithm with a range of low-to-high-level ground truth information. Our experiments show that there are several ways to improve action recognition without changing the existing framework. This includes improving low-level flow to improve the motion-based *HOF* and *MBH* and integrating mid-level information such a bounding box surrounding the person to improve the frame-based *HOG*. A surprising result is that the motion boundaries around a person's body contour seem to contain information for action recognition that is as important as the optical flow within the region of the body. It is also surprising that, with a good bounding box, which is probably easier to achieve than estimating accurate flow, one can obtain a large improvement over the baseline. Unfortunately, the human detector we evaluated is not accurate enough to predict such bounding boxes.

Despite all the modifications to the Dense Trajectories algorithm using low-to-mid ground truth data, we find that the best features for action recognition (of those tested) are high-level pose features. While this might not be surprising, our contribution here is threefold. First, we point out that pose over time is the best representation for action recognition; we also point out several factors that are important to make good pose features, such as the use of relations, the number of frames, and the step size between frames in a trajectory. Second, the sparse version of Dense Trajectories as well as sub-**J-HMDB** allows a fair comparison between joint-wise low/mid-level texture features and high-level pose features. We observe that the texture around joints is more discriminative and effective than dense texture on the whole frame, but the low-level texture around joints performs worse than the high-level position-and-relation-based features derived directly from joint positions. Third, for sub-**J-HMDB**, where the full body is

visible, a recent pose estimation algorithm computes poses that are more reliable than low/mid level features for action recognition of complex actions in realistic videos.

Beyond understanding algorithms for action recognition, **J-HMDB** can serve as a challenge to the fields of pose estimation, flow estimation, and human detection.

Acknowledgements: JG was supported in part by the DFG Emmy Noether program (GA 1927/1-1) and CS by the ERC advanced grant Allegro.

References

[1] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. *ECCV*, pp. 168–181, 2010. 5

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. *ICCV*, pp. 1365–1372, 2009. 2, 3

[3] L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. *ICCV*, pp. 624 – 630, 1995. 1

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. ACM TIST, 2:27:1–27:27, 2011. 4

[5] N. Dalal, , and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, pp. 886–893, 2005. 4

[6] N. Dalal, , B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *ECCV*, pp. 428–441, 2006. 4

[7] M. Dantone, J. Gall, C. Leistner, and L. Van Gool. Human pose estimation using body parts dependent joint regressors. *CVPR*, pp. 3041–3048, 2013. 7

[8] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. *BMVC*, pp. 3.1– 3.11, 2009. 3

[9] G. Farneback. Two-frame motion estimation based on polynomial expansion. *Image Analysis*, pp. 363–370, 2003. 4

[10] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. *CVPR*, pp. 1–8, 2008. 1, 2, 3

[11] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. *ECCV*, pp. 340–353, 2012. 2

[12] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human 3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *PAMI*, 2014. 2, 3

[13] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. *BMVC*, pp. 12.1–12.11, 2010. 3

[14] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. *ICCV*, pp. 2556 – 2563, 2011. 1, 2, 3, 4

[15] F. D. la Torre, J. Hodgins, J. Montano, S. Valcarcel, R. Forcada, and J. Macey. Guide to the Carnegie Mellon University multimodal activity (CMU-MMAC) database. Tech. Report CMU-RI-TR-08-22, July 2009. 2, 3

[16] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. *CVPR*, pp. 1–8, 2008. 2, 4

[17] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. *CVPR*, pp. 2929–2936, 2009. 1, 3

[18] J. Niebles, C. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. *ECCV*, pp. 392–405, 2010. 2, 3

[19] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Berkeley MHAD: A comprehensive multimodal human action database. *WACV*, pp. 53–60, 2013. 2, 3

[20] D. Parikh and C. L. Zitnick. The role of features, algorithms and data in visual recognition. *CVPR*, pp. 2328–2335, 2010. 2

[21] K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *MVA*, 24(5):971–981, 2013. 2, 3

[22] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. *CVPR*, pp. 1194–1201, 2012. 1, 2, 3

[23] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, pp. 309–314, 2004. 3

[24] B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. *CVPR*, pp. 1281–1288, 2011. 3

[25] L. Sigal, A. Balan, and M. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *IJCV*, 87(1):4–27, 2010. 2, 3

[26] V. K. Singh and R. Nevatia. Action recognition in cluttered dynamic scenes using pose-specific part models. *ICCV*, pp. 113–120, 2011. 1, 2

[27] D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, to appear, 2013. 2, 5

[28] M. Tenorth, J. Bandouch, and M. Beetz. The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. *THEMIS*, pp. 1089–1096, 2009. 2, 3

[29] K. Tran, I. Kakadiaris, and S. Shah. Modeling motion of body parts for action recognition. *BMVC*, pp. 64.1–64.12, 2011. 1, 2

[30] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013. 1, 2, 3, 4, 7

[31] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *CVIU*, 115(2):224–241, 2010. 1

[32] Y. Yacoob and M. Black. Parameterized modeling and recognition of activities. *CVIU*, 73(2):232–247, 1999. 1

[33] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *PAMI*, to appear. 2, 7

[34] A. Yao, J. Gall, and L. Van Gool. Coupled action recognition and pose estimation from multiple views. *IJCV*, 100(1):16–37, 2012. 1, 6

[35] S. Zuffi and M. Black. Puppet flow. Technical Report TR-IS-MPI-007, MPI for Intelligent Systems, 2013. 2, 3

[36] S. Zuffi, O. Freifeld, and M. Black. From pictorial structures to deformable structures. *CVPR*, pp. 3546–3553, 2012. 1, 2, 3