Robust information fusion in the DOHT paradigm for real-time action detection

Geoffrey Vaquette[1] · **Catherine Achard**[2] · **Laurent Lucat**[1]

Abstract In the increasingly explored domain of action analysis, our work focuses on action detection—i.e., segmentation and classification—in the context of real applications. Hough transform paradigm fits well for such applications. In this paper, we extend *deeply optimized Hough transform* paradigm to handle various feature types and to merge information provided by multiple sensors—e.g., RBG sensors, depth sensors and skeleton data. To this end, we propose and compare three fusion methods applied at different levels of the algorithm, one being robust to data losses and, thus, to sensor failure. We deeply study the influence of merged features on the algorithm's accuracy. Finally, since we consider real-time applications such as human interactions, we investigate the latency and computation time of our proposed method.

Keywords Hough transform · Action detection · Feature fusion · Activity detection · Deeply optimized hough transform

✉ Geoffrey Vaquette
geoffrey.vaquette@cea.fr

Catherine Achard
catherine.achard@upmc.fr

Laurent Lucat
laurent.lucat@cea.fr

[1] CEA, LIST, Vision and Content Engineering Laboratory, Point Courrier 173, 91191 Gif-sur-yvette, France

[2] UPMC Univ Paris 06, CNRS, UMR 7222, ISIR, Sorbonne University, 75005 Paris, France

1 Introduction

Action recognition is an increasingly explored field in computer science with many applications such as video surveillance, video games, augmented reality, automatic annotation, smart homes and health monitoring.

Our work focuses on real-life applications in the context of human action analysis. In this framework, input videos are not segmented—relatively to actions. More and more methods are proposed to analyze human action or activity but most of them consider the problem of short segmented video classification while only a few ones deal with large unsegmented videos. This task is a more difficult one task since the algorithm needs to perform both segmentation and classification. Thus, in this paper, we present an action detection—segmentation and recognition—algorithm.

In the context of a real application, multiple sensors are likely to be used to capture as much information as possible—e.g., kinect sensors capture both RGB and depth information—or to cover the largest possible area. The number of involved sensors depends on the monitored area. Real-life application-oriented algorithms have to handle successfully a variable number of sensors and, ideally, to merge information coming from different feature types. Furthermore, since some extracted features can be unavailable at some times—e.g., skeleton data extracted from classical depth sensors—algorithms are expected to profit from multi-sensor data when available, while staying robust to data failure.

In applications involving human interaction, such as robotic ones, low latency is a crucial issue to be addressed. It involves both low computation time and low reaction time, i.e., the algorithm has to detect an action from as little frames as possible.

In this paper, we present an action recognition method performing both segmentation and recognition of actions in video streams. We chose *Hough transform* paradigm for its short latency and computation time properties. We adapt *deeply optimized Hough transform (DOHT)* to exploit various feature types and manage a variable number of sensors. More precisely, we propose and compare three information fusion methods, one of them aims to be robust to data loss. A major effort is also made on the choice of low-level features, either in terms of modality (skeleton or video) or in terms of descriptors [HOG, HOF, Trajectory Shape (TS)]. Again, the performance is studied both for accuracy and computation time points of view. Lastly, we study the trade-off between latency and accuracy in order to find a good compromise. The proposed features, fusion strategies and models are evaluated on the TUM Kitchen dataset [47] since it combines several views, several modalities and video streams.

Our main contribution, guided by the implementation of a real-time action recognition system, is threefold. First, we propose a method to merge information coming from various sensors that is able to manage sensor failure or momentary lack of data since these are frequent situations occurring in real-life applications. Second, we introduce a comparative study on the most suitable characteristics to be employed in this context. Third, computation times are studied according to all parameters of the method, i.e., features, number of views and latency. These elements are essential to get the best trade-off between performance, speed and latency.

The remainder of this paper is organized as follows. Section 2 describes the works related to action detection and recognition by focusing on two main issues: characterization of actions on one hand and their modeling on the other hand. Section 3 presents the detection principle based on the Hough transform paradigm. Section 4 discusses in details the proposed strategies to merge information coming from different sensors. This fusion is performed at three different levels: descriptor level, vote level and score level. Section 5 explains the experimental protocol, describes the database and reports the results obtained with fusion of views and fusion of descriptors. Section 6 deals with computation times and latency. Section 7 concludes the paper and discusses potential future works.

2 Related works

At first, motion-based recognition was developed for cyclic motion [1, 2] or gesture interpretation [14–16], for instance. Darell and Pentland used normalized correlation between an input gesture and constructed models, along with Dynamic Time Warping to detect gestures. In the context of hand gestures, Davis et al. [16] tracked the finger trajectories (direction of motion and displacement) in order to build a *motion code* used to recognize the hand gesture. For action recognition, work was done to recognize human motion as *walking, running and skipping actions* [17, 21, 27] by describing the human body motion with a combination of line segments for both upper and lower body parts. These segments were then used in a *scenario hierarchy* [6, 17] to describe and recognize the target actions. Yamato et al. [56] made an effort to recognize tennis strokes using Hidden Markov Models (HMM). For a more detailed review of motion-based recognition, see [6, 29, 44] for surveys.

Since, action analysis has been a widely explored domain and can be categorized into two major topics: action characterization and action temporal modeling. While the first one extracts features independently on the actions in order to describe a video, the last one aims to build a temporal model of actions and is supervised. This section presents some work in action recognition and is organized according to this categorization.

2.1 Features

To capture information about the real world and more precisely about human motion, several low-cost sensors are available. We report below some features extracted from the most current sensors, i.e., camera, motion capture and depth sensors.

2.1.1 Video-based features

In order to extract and exploit information contained in video streams, most of the existant methods adopt *local features*, an extension of interest points [22] introduced for object recognition applications. For instance, Laptev [31] proposed Space–Time Interest Points (STIPs), aiming to extract salient local structures: pixel groups with high variation in both space and time. Laptev et al. [32] then used these interest points in a multi-scale approach and combined them with space–time pyramids and nonlinear multi-channel SVMs to recognize segmented actions.

An intuitive approach, when dealing with video streams, is to extract salient trajectories and process them as descriptor to recognize actions. Trajectories can be extracted with the well-known Kanade–Lucas–Tomasi tracker (KLT) or with interest points matching across temporal dimension [11, 36, 37, 46].

Sun et al. [46] proposed a hierarchical spatio-temporal context modeling. The idea is to describe context around interest points at different levels, namely *point-level context, intra-trajectory context, inter-trajectory context* and then to combine feature channels through a multi-channel nonlinear SVM as in [32]. More precisely, once SIFT

features have been extracted and trajectories computed with KLT tracker, the authors model the point-level context averaging the SIFT descriptor over all points from the extracted trajectory. Intra-trajectory is described by an ergodic Markov chain, and inter-trajectory context is described by a *trajectory proximity descriptor* based on a Markov Stationary Distribution calculated from closed trajectories.

To exploit the discriminative ability of trajectories and in order to benefit from dense sampling, which has shown to improve results in the image classification field [41], Wang et al. [49] extract trajectories over a dense grid using optical flow for tracking. Irrelevant trajectories (e.g., immobile points) are then ignored. The authors proposed 4 features to describe the videos: Trajectory Shape (TS), Histogram of Gradient (HoG) [12], Histogram of Optical Flow (HOF) [13] and Motion Boundary Histogram (MBH). MBH showed to outperform previous descriptors, particularly on videos with camera movement since it removes constant camera motion. Later, Wang et al. [50] improved this work considering camera motion when extracting trajectories and integrating a human detector as a filter. These dense trajectory descriptors have shown to be effective and have been widely used for action recognition [39].

2.1.2 Motion capture (MoCap)-based features

Another explored field for action recognition is associated with motion capture (*MoCap*). Tenorth et al. [47] proposed the *TUM Kitchen Dataset*, presented in Sect. 5.1, providing sequences of skeleton joint coordinates suitable for action recognition. Barnachon et al. [5] proposed an exemplar-paradigm-based method to recognize actions from MoCap data. More precisely, they extract local histogram of action poses as features and compare actions through dynamic programming. Recently, Amor et al. [3] used Kendall's shape framework to characterize shapes of individual skeletons.

2.2 Depth-based features

More recently, with the emergence of low-cost depth sensors, efforts were done to profit from information contained in 3D space. First, work was done to create or adapt features from RGB domain to depth or RGB-D (RGB and depth combined) domain [42, 48, 55].

Oreife and Liu [42] proposed a descriptor, named Histogram of Oriented 4D Normals (HON4D), which is "analogous to the histogram of gradients in color sequences". It encodes both orientation and magnitude of the surface in 4D space. Xia and Aggarwal [55] proposed a method called *DSTIP* which allows the extraction of STIPs from depth videos. They introduced a feature designed to describe 3D depth cuboids called *Depth Cuboid Similarity Feature (DCSF)*. Wang et al. [51] used depth information to describe 3D context around areas of interest through *Local Occupancy Patterns (LOP)* describing how the area is filled in a 3D grid.

Other works [24, 40, 45] combined RGB and depth information since these two channels are complementary, encoding both color appearance and 3D shape. For instance, in order to compute 3D *Trajectories of Surface Patch (ToSP)*, Song et al. [45] compute STIPs and apply the KLT tracker in RGB domain to extract trajectories before extending them to 3D space since interest points are much more easily extractable from RGB images than from depth maps.

Finally, depth sensors paved the way for MoCap methods in realistic context for action recognition since they can be used for skeleton extraction without heavy and invasive equipment. Wang et al. [51] defined *actionlet ensemble* being a discriminative subset of joints for a particular action and combined it with the *Local Occupancy Patterns* they introduced.

Wang et al. [52] proposed a graph model which benefits from skeleton data during training process, avoiding heavy manual annotation of body parts. However, at testing time, this model only uses 2D information. This way, their *Multiview Spatio-Temporal AND-OR graph (MST-AOG)* can be applied on data provided by simple cameras.

2.3 Modeling of action

Several methods exist to model actions in the action recognition literature. The first one is a simple extension of the bag-of-word feature, used in object recognition, to spatio-temporal bag of word [32]. It consists in characterizing spatio-temporal volume around Space–Time Interest Points (STIPs) and representing them by words. Recognition is then obtained using a SVM on the histogram of words. The use of histograms in this method has the disadvantage of loosing spatial and temporal dimensions. This is a major disadvantage since these dimensions contain abundant information about occurring actions.

A simple method to compare features exploiting temporal aspect is to align them temporally before estimating distances. This alignment and comparison can be performed using Dynamic Time Warping (DTW) [38]. That is what Barnachon et al. [5] proposed in order to resolve the action recognition problem, looking for the nearest neighbor of the tested exemplar in a temporal alignment independent way.

Generative methods can also be used for modeling temporal information. In such a paradigm, actions can be modeled by Hidden Markov Model (HMM) [18] or grammars [30] for example. In [18], each sequence is first characterized by a sequence of movelet codewords—a

movelet being a collection of shape motion and occlusion of image patches corresponding to main parts of the body. Then, each action is modeled using HMM. The approach proposed in [30] takes advantage of similitude between speech and human action. Thus, authors first modelled action units using Hidden Markov Models—as done for words in speech analysis— and model action as sentences using an action grammar.

Finally, Conditional Random Field CRF [60] used by Wang et al. [53] is an extension of HMM to a discriminative context.

Deep learning [33] is a recent and popular approach of machine learning that automatically learns a hierarchy of features based on low-level descriptors. It has been employed in [25] in the context of action recognition to extract features from both spatial and temporal dimensions, capturing motion information encoded in multiple adjacent frames. Baccouche et al. [4] extend the convolutional neural networks to 3D and thus, automatically learn spatio-temporal features. In a second step, a recurrent neural network is trained to classify each sequence of the learned features.

The above modeling methods all consider an action recognition problem. Since videos extracted from real applications are not segmented, we need to handle action analysis for detection (segmentation and recognition).

Adapting methods initially designed for action recognition to action detection is a first way to go. For instance, Hoai et al. [23] proposed a discriminative temporal extension of the spatial bag-of-words model. Classification is performed robustly within a multi-class SVM framework applied on a large number of segments. Inference over segments is done with dynamic programming.

Yu et al. [59] proposed to build action proposals based on human and motion detectors. These action proposals can then be used to detect actions in unsegmented videos.

Another approach, proposed in [58], classifies and localizes human actions using a Hough transform voting framework. Mapping between densely sampled feature patches and their corresponding votes in a spatio-temporal-action Hough space is learned with random trees. Contrary to the previous presented methods, the latter does not need the use of a classifier applied on an important amount of segments to perform the segmentation and has low latency. For these reasons, our work focuses on Hough transform paradigm, presented in next section.

3 Hough transform paradigm

3.1 Introduction

In this section, we describe Hough paradigm and its usage for action recognition. Hough paradigm has proven to be an efficient way to detect object [19], track them [20] and has been recently used for action recognition [8, 28, 57].

The action detection process using Hough transform can be simply described as a two step model:

1. Feature extraction and quantization resulting in a set of words $(c_1, t_1), (c_2, t_2), \ldots, (c_N, t_N)$ where c is the label of the word—named codeword—and t is the time where it was extracted and N is the total number of extracted words in a sequence. In the following, when there is no ambiguity, we note them (c, t) for simplicity.
2. Voting process: each word (c, t) votes with a weight $\theta(a, \delta_t, c)$ for an action a, centered on time $t + \delta_t$.

Hough score is then computed as:

$$\mathcal{H}(t', a) = \sum_{(c,t)} \theta(a, t' - t, c), \tag{1}$$

and detection at each time t is done by pooling the maximal value of $\mathcal{H}(t, a)$:

$$\widehat{a}(t) = \underset{a}{\operatorname{argmax}} \mathcal{H}(t, a) \tag{2}$$

Classification accuracy of Hough transform algorithms highly depends on weights $\theta(a, \delta_t, c)$, learned on the training database. Their number depends on the number of action a to detect, the maximal displacement time considered for voting M $(-M < \delta_t < M)$ and the number of extracted codeword c. Action detection methods based on Hough transform paradigm mostly differ on the estimation of these weights, more precisely on the parameters considered for optimization.

A very intuitive way to estimate these weights is to compute statistics on the learning database. Leibe et al. [34] proposed the Implicit Shape Model (ISM) where the weights are proportional to the number of detections of a codeword c at the relative time δ_t to the center of the action a.

Following this work, some authors proposed to weight this expression in order to take benefit from the discriminative power of different elements. Thus, Maji et al. [35] put emphasis on the discriminative power of each codeword, with a weight w_c learned during the training step. For Zhang et al. [61], not all training examples represent actions with the same accuracy. They introduced a weight to model how important each training example should be. Finally, Wohlhart et al. [54] revealed that learning these weights with a linear SVM can be viewed as learning a different weight for each time displacement δ_t.

These previous methods optimizes weights θ considering only one parameter. None of them takes into account the influence of all parameters at the same time. This issue was addressed by [8] that defined the *deeply optimized Hough transform (DOHT)*, aiming to learn weights

considering all three parameters a, δ_t and c. As this global optimization gives better results than the previously described ones, we chose to follow this paradigm.

3.2 Deeply optimized Hough transform (DOHT)

With the goal to find actions occurring at time t through a Hough transform process, Chan-Hon-Tong et al. [8] proposed a new formulation to optimize the Hough weight $\theta(a, \delta_t, c)$. Among others, this model makes the assumption that a codeword extracted at time t should provide less and less information when time displacement δ_t increases. Then, the training process to solve consists of:

$$\min_{\theta \geq 0, \xi \geq 0} \left(L_{\text{reg}}(\theta) + C \times L_{\text{data}}(\xi)\right)$$

under constraints:

$$\begin{cases} \forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t'): \\ \quad \sum\limits_{(c,t)\in\mathcal{V}} \begin{pmatrix} \theta\left(a^*_{\mathcal{V}}(t'), t'-t, c\right) \\ -\theta(a, t'-t, c) \end{pmatrix} + \xi(t') \geq 1 \\ \forall a, c, \delta_{t1}, \delta_{t2}: \\ \quad \begin{cases} \delta_{t1} \leq \delta_{t2} \leq 0 \Rightarrow \theta(a, \delta_{t1}, c) \leq \theta(a, \delta_{t2}, c) \\ 0 \leq \delta_{t1} \leq \delta_{t2} \Rightarrow \theta(a, \delta_{t1}, c) \geq \theta(a, \delta_{t2}, c) \end{cases} \end{cases} \tag{3}$$

where $\mathcal{V}$ is the set of training examples, $a^*_{\mathcal{V}}(t)$ is the actual action occurring, $L_{\text{data}}(\xi)$ is a loss function introduced to allow some data constraints to be unsatisfied and $L_{\text{reg}}(\theta)$ is added to prevent over-fitting. Coefficient C mitigates the trade-off between attachment to data and regularity as in [7, 10]. Note that in DOHT formulation, extracted features vote for an action presence rather than for the action center. More details on this formulation, expressed as a linear problem that can be efficiently solved, will be found in [8].

As the number of constraints is very large and can be not tractable, Chan-Hon-Tong et al. [9] introduce a new formulation, which can be solved using SVM and which is based on a set $\mathcal{J}^{\text{full}}$ of intervals I. With this new formulation, the problem is equivalent to find a function $w() > 0$ satisfying:

$$\theta(a, \delta_t, c) \approx \sum_{I \in \mathcal{J}^{\text{full}}} w(a, I, c) \times \chi_I(\delta_t), \tag{4}$$

where $\mathcal{J}^{\text{full}}$ is the set of all temporal intervals included in [−M, M] containing 0 and M is the maximal time displacement considered during voting step and is set to the maximal length of actions by the authors. $\chi_I(t)$ is defined as:

$$\forall t: \ \chi_I(t) = \begin{cases} 1 & t \in I \\ 0 & t \notin I \end{cases}. \tag{5}$$

With this new formulation, L_{reg} and L_{data} are defined as $L_{\text{reg}} = \|w\|_2^2$ and $L_{\text{data}} = \|\xi\|_1$. The formulation (Eq. 3) is then expressed as a standard multi-class SVM [10, 26]:

$$\begin{aligned} &\min_{w,\xi}\left(\|w\|_2^2 + C \times \|\xi\|\right) \\ &\text{underconstraints:} \\ &\forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t'): \\ &\textstyle\sum_{(c,t)\in\mathcal{V}, I\in\mathcal{J}^{\text{full}}} \left((w(a^*_{\mathcal{V}}(t'), I, c) - w(a, I, c))\chi_I(t'-t)\right) \\ &\quad + \xi(t') \geq 1. \end{aligned} \tag{6}$$

In order to speed up the training step, the authors replaced $\mathcal{J}^{\text{full}}$ with a subset of intervals J, reducing the number of constraints. This subset is designed in such a way that the more interval boundaries are near 0 the higher granularity is. More formally, these intervals are defined as:

$$J = \left\{[-2^{-\alpha}M, 2^{-\beta}M]\right\} \text{ with } \alpha,\ \beta \in \{0, \ldots, \mathrm{n_I}, +\infty\}. \tag{7}$$

Thus, for example, using $n_I = 4$, leads to 25 intervals.

The whole training process is summarized in Fig. 1 and testing process in Fig. 2.

4 Fusing information in the DOHT paradigm

In the original paper [9], DOHT was only applied to skeleton joint data. Yet, in deployed applications, skeleton data is not available at all time as, for example, for video-surveillance systems based on camera sensors. Besides,

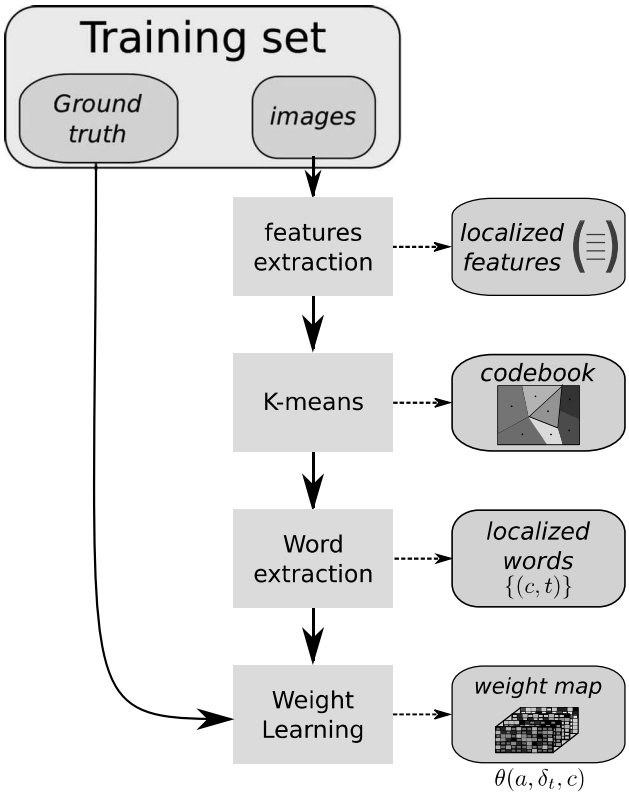

Fig. 1 Learning process

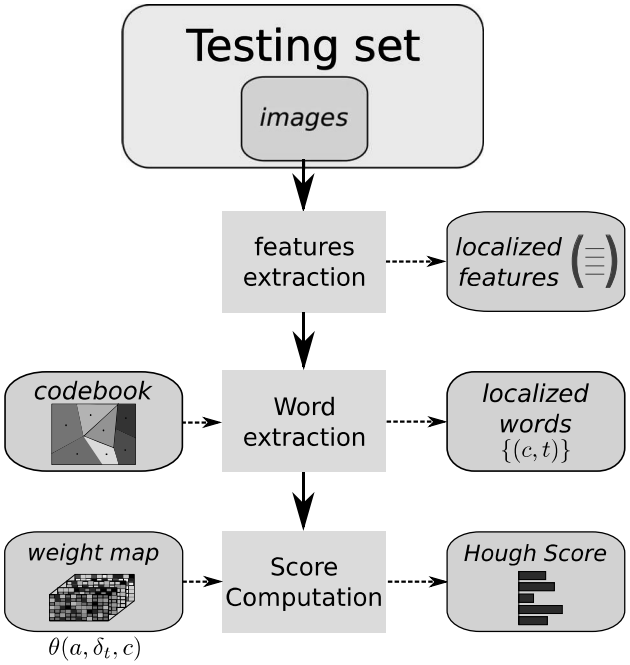

Fig. 2 Testing process

multiple sensors are generally required to cover an area and the sensor number highly depends on the application and on the shape of the monitored area. We are therefore faced with a twofold problem: the problem of managing variable nature of data and the problem of managing an unknown and variable number of sensors.

To handle these issues, this paper introduces a data fusion template using Hough paradigm. It aims to benefit from as much information as possible, whatever the nature of data or the number of sensors. As proposed in [43], this fusion can be performed at three different levels, namely *low level*, *middle level* and *high level*, corresponding to *descriptor-level*, *vote-level* and *score-level* fusion in the DOHT paradigm. In the following, we refer to them with the latter formulation. These levels will be described in next sections as well as their pro and cons.

4.1 Descriptor-level fusion

Descriptor fusion is the lowest proposed fusion level. This fusion consists in a concatenation of feature descriptors right after extraction, producing a larger feature as shown in Fig. 3a. Quantization (codebook computation and representation) is then computed using k-means on these concatenated features, and the rest of the algorithm is executed. In other words, these features are the new input of the DOHT algorithm. At this level, the main goal is to compute a descriptor containing more information. In our case, with dense trajectories, descriptor fusion is applied on features

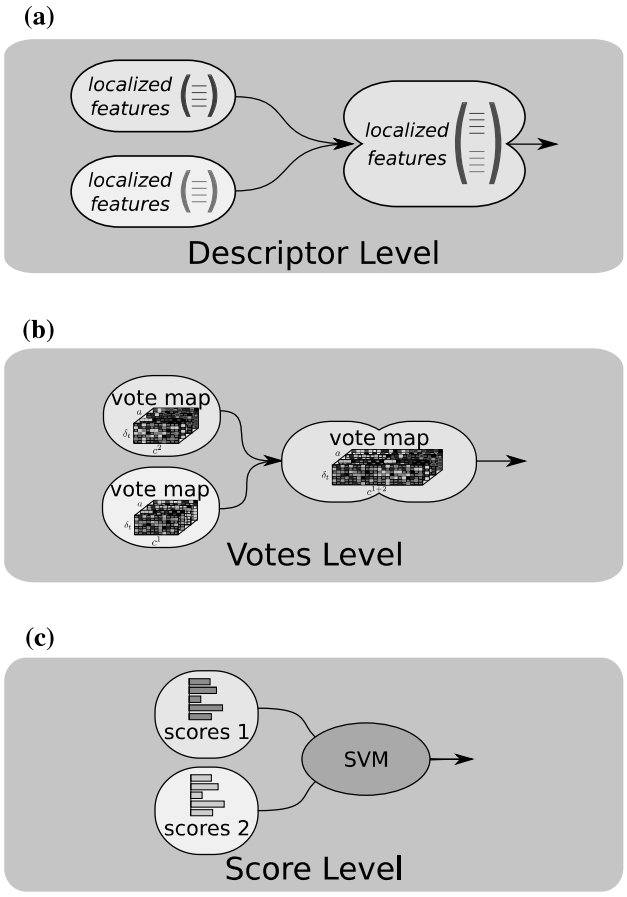

Fig. 3 Three fusion levels in the DOHT Paradigm. **a** *Descriptor-level* or *low-level* fusion. **b** *Vote-level* or *middle-level* fusion. **c** *Score-level* or *high-level* fusion

extracted on the same track leading to a more complete descriptor of extracted data. This new descriptor is richer than simple ones since it may contain temporal information from HOF and spatial information from HOG, for instance. Note that this feature level fusion can only be used for data coming from a single sensor or, otherwise, requires to match data across sensors. In this last case, a disadvantage is the high sensitivity of this fusion to sensor failure.

4.2 Vote-level fusion

To combine features at the vote level, codebooks are first generated for each feature independently. Then, during the learning step, the dimensions of vote map $\theta(a, \delta_t, c)$ are increased in order to learn weights in a jointly way. The third dimension size is now the summation of the extracted codebook number from each feature as illustrated in Fig. 3b. Note that since weight learning is not done independently for each feature but in a jointly way, more importance is given to high representative descriptor instances for each action.

As a result, a sensor failure during the testing step at time t will simply lead to a lack of descriptor for this sensor at this instant. Other sensors will still generate votes so that the algorithm can output a score; since the predicted action is the one associated with the highest score value, ranking is not highly affected and the output is, in most cases, unaffected. Thus, this vote level can be computed with a variable number of sensors. Finally, as codeword generation is done independently for each sensor, middle-level data fusion can be performed with different modalities (RGB, depth, skeleton, etc.).

4.3 Score-level fusion

Finally, fusion can be computed at the highest level, namely *score level* as illustrated in Fig. 3c. Here, it consists in computing the score $H_s(t,a)$ at time t for each action a and sensor or descriptor s independently, then balancing sensor importance in a global way rather than in a local way like in the *vote-level fusion*. To do so, we run a SVM on Hough scores estimated independently for each sensor/descriptor $H_s(t,a)$. After learning, a global score $H(t, a)$ is obtained and used to classify actions. By principle, this level of fusion cannot be employed with missing sensors as one of the sensor scores $H_s(t,a)$ will be empty and thus it perturbs the estimation of the global score. An advantage is that, as for the vote-level fusion, the fusion can be done for different sensors and different modalities.

5 Results

In order to evaluate information fusion, we chose dense trajectory [49] and skeleton trajectory [8] as input feature, with 15 and 10 frames length, respectively. We evaluated DOHT algorithm with a 3000 words codebook for visual features and 5 words for each skeleton feature. These parameters are those used in the original papers [8, 49]. Note that contrary to descriptors extracted from dense image trajectories, the tracklets from skeleton joints are very constrained by human movements. Moreover, dense trajectory descriptors contains more variable information based on appearance and shape (as HoG and TS for example). They are also more numerous as they are extracted on a dense grid. The maximal displacement time is set to $M = 50$ frames and $n_I = 4$ is employed to define intervals (Eq. 7).

Moreover, we chose to use the TUM Kitchen Dataset since it is well adapted to action detection (most of the databases are dedicated to recognition and provide only segmented actions) and contains multiple points of view and modalities (RGB captured with 4 camera views and skeleton data captured with motion capture).

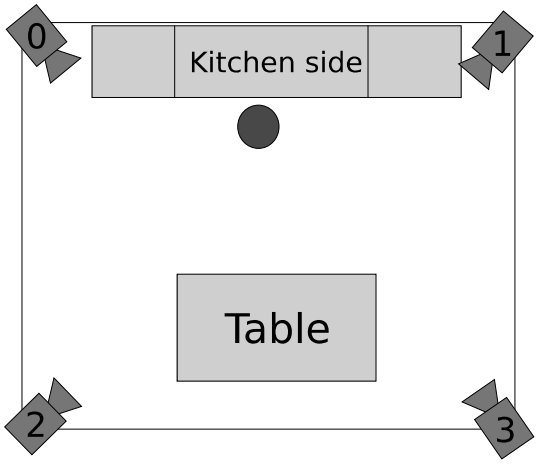

Fig. 4 Approximate position of each camera (*red*), with the actor (*blue circle*) being on kitchen side

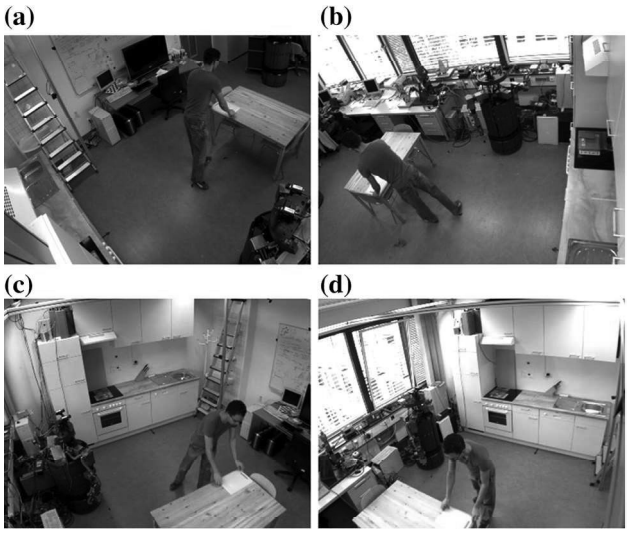

Fig. 5 Different camera views in TUM Kitchen Database. **a** Example from View 0. **b** Example from View 1. **c** Example from View 2. **d** Example from View 3

5.1 TUM kitchen dataset

TUM Kitchen Dataset proposed by [47] is composed of 19 videos in which 4 subjects are setting the table in either a natural or a more robot like way. Videos have been recorded with four cameras homogeneously located around the scene as represented in Fig. 4. An example of camera views is presented in Fig. 5. Depending on where the performed action takes place in the scene, specific cameras will be more suitable than the others. Thus, in the illustrated example, since the person is facing backwards in views 0 and 1, corresponding cameras capture a limited information on the performed action. This database also includes motion capture data, RFID and magnetic sensors.

Table 1 Accuracy on TUM when considering each descriptor independently

	TS	HoG	HOF	Skeleton
View 0	75.0	**81.8**	79.6	81.5
View 1	72.6	**81.3**	76.5	
View 2	70.5	**80.5**	74.7	
View 3	73.5	**77**	74.5	

Bold values indicate the highest results

Videos are labeled according to 9 actions being *Carrying while locomoting*, *Reaching*, *taking something*, *Lowering an object*, *Releasing/Grab something*, *Opening door*, *Closing door*, *Opening a drawer* and *Closing a drawer*.

The database is split into training and testing parts following the scheme defined in [57], and algorithm was trained to estimate the action associated with the left hand motion in each frame. Accuracy is defined as the number of correct predictions divided by the total number of frames in the testing videos.

5.2 Results with independent descriptors

For comparison, we first evaluate each descriptor independently, i.e., dense descriptors in images [49] [(Trajectory Shape (TS), Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF)] and skeleton descriptors [8]. We obtained the results presented in Table 1.

We observe that, in the context of the TUM Kitchen Dataset, visual aspect is more informative than other features. For example, since we do not use spatial localization of extracted features, Trajectory Shape of points around the hand for the action *Taking something* will be similar to other displacements as *Carrying While Locomoting* so this descriptor will not be very discriminative. On the contrary, visual aspect (encoded by HOG feature) will probably extract information as presence of an object or drawer, for instance. For this particular example of *Taking something*, TS gives an accuracy of 26% while HOG gives 72% on view 0. Figure 6 shows confusion matrices of these descriptors on view 0.

5.3 Results on feature fusion

We compute feature fusion by computing all possible combinations of visual features being *Trajectory Shape*, *Histogram of Gradients* and *Histogram of Optical Flow*.

We observed in Table 2 that best results are likely obtained when combining Trajectory Shape and Histogram of Gradient. These two features are complementary since the first one encodes movement in the image, while the second one encodes visual aspect. On the contrary, features as TS and HOF are both extracted from optical flow and merging them does not improve results.

We also observe that the lowest fusion level leads to the more accurate results. More generally: the lower the fusion level is, the greater the performance is.

To evaluate the stability of DOHT algorithm according to the learning parameter, we ran it for various values of parameter C of the SVM (the parameter that controls the trade-off between the attachment to data and the regularity, see Eq. 3). Results are reported in Fig. 7 for the low-level fusion. For all configurations, results are quite stable when $C > 2$ which shows that our method does not highly depend on this parameter and does not need fine tuning. To avoid over-fitting, we chose to keep the lowest stable value, being $C = 2$ in the following.

We can also notice in Table 2 that views 0 and 1 lead to the best performance. Actually, as they focus on the kitchen, they are really reliable to recognize actions such as

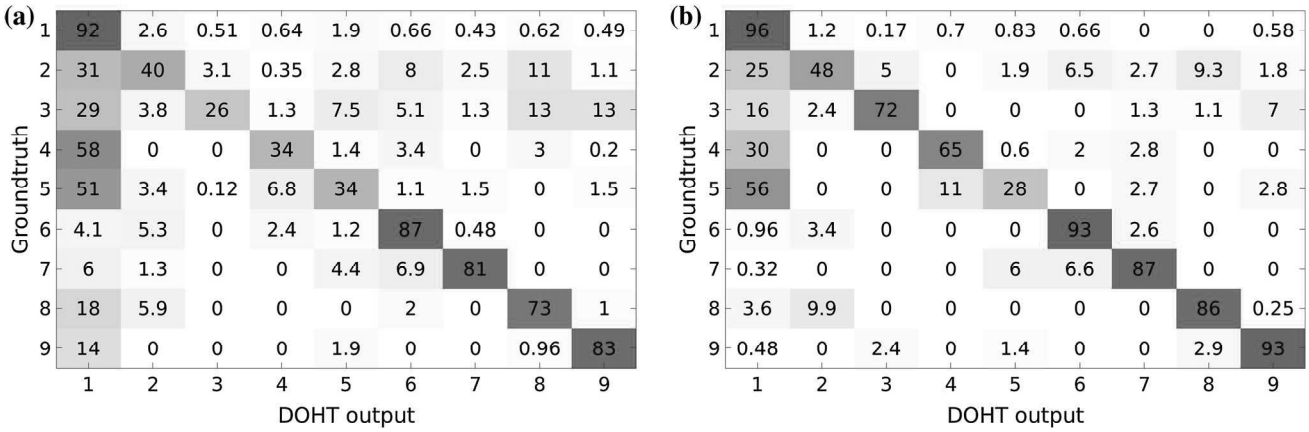

Fig. 6 Confusion matrices on View 0 of TUM Kitchen dataset. 1: carrying while locomotion, 2: reaching, 3: taking something, 4: lowering object, 5: releasing/grab something, 6: opening door, 7: closing door, 8: opening drawer, 9: closing drawer. **a** Trajectory shape (TS). **b** Histogram of gradient (HOG)

Table 2 Accuracy on TUM dataset for different combinations of visual descriptors

	TS + HOG	TS + HOF	HOG + HOF	TS + HOG + HOF
View 0				
Low level	*82.5*	78.6	80.5	80.6
Medium level	80.3	79.3	*81.9*	81.2
High level	79.2	78.9	81.4	80.0
View 1				
Low level	*81.7*	76.5	78.2	78.0
Medium level	80.1	76.6	80.4	80.0
High level	78.2	79.4	79.4	78.6
View 2				
Low level	*80.7*	74.1	79.7	78.3
Medium level	78.1	73.4	80.0	77.6
High level	78.1	73.9	78.5	77.3
View 3				
Low level	*77.9*	74.5	*77.5*	77.9
Medium level	77	75.0	*77.2*	77.2
High level	76.9	75.4	76.4	76.8

Italic values outperform corresponding single descriptor performance

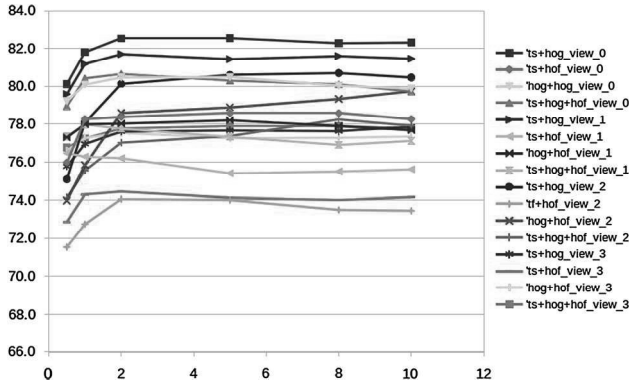

Fig. 7 Influence of parameter C on accuracy with feature fusion at the low-level fusion, for each configuration

Opening door, *Closing door*, *Opening a drawer* and *Closing a drawer*. The two other views are more appropriate to recognize actions performed around the table. As we do not know *a priori* the actions carried out, merging the views is a natural approach.

5.4 Results on view fusion

In order to benefit from information extracted from different points of view in the scene, we evaluated our algorithm for all possible combinations of view pairs. Since it gave best results at low-level fusion, we kept the new TS + HOG feature introduced in Sect. 5.3.

Information coming from different views is combined at middle and high fusion level—they cannot be fused at the lowest level as described in Sect. 4. Results are resumed in Table 3.

Table 3 Views fusion accuracy in DOHT paradigm according to the fusion level

Views	Medium level	High level
Fusion of 2 views		
(0 + 1)	83.1 (+0.6)	83.0 (+0.5)
(1 + 2)	82.1 (+0.4)	82.1 (+0.4)
(0 + 2)	83.9 (+1.4)	82.5 (+0.0)
(1 + 3)	81.7 (+0)	81.5 (−0.2)
(0 + 3)	83.4 (+0.9)	82.0 (−0.5)
(2 + 3)	80.1 (−0.6)	79.5 (−1.2)
Fusion of all views		
All	83.1 (+0.6)	83.2 (+0.7)

In brackets, the difference between the fusion score and the best view score used in the fusion

Depending on the configuration of the room, different views can be unequally informative. For instance, since view 0 and 1 are located on the same side of the kitchen relatively to the direction of movements, they are less complementary than view 0 and 2 which can help each other when occlusions occur—see Fig. 5 for camera positions.

Merging information from different views outperforms single view results, and best results are obtained when combining views 0 and 2. Once again, concatenating information at the lowest possible level gives best results.

Combining all four views does not increase results, probably due to the information redundancy and to the large number of weights the SVM has to deal with.

5.5 Results on modalities fusion

In this section, we investigate the performance when fusing modalities, i.e., RGB data and skeleton data. Benefitting from of the previous section results, we use a combination of TS + HOG at the lowest level of fusion as image descriptor. This descriptor is then combined with the skeleton data. This new fusion can only be achieved at medium and high fusion level, giving an accuracy of 85% which outperforms single modality results.

5.6 Robustness to missing data or sensor failure

In this section, we study the influence of loss of information or sensor failure on accuracy. This can be a critical issue, particularly for real-life applications where sensors can be sporadically unavailable for some reason. We evaluate this effect for two configurations. In the first one, the learning is done with (TS + HOG) features combined at the low level of fusion, then the 4 views are merged at the middle level of fusion. In the second case, (TS + HOG) features are combined at the low fusion level and then the views and the skeleton data are merged at the middle level of fusion. Please remember that only the intermediate level can manage the lack of data (Sect. 4). For both configurations, at testing time, we ran the algorithm while ignoring data from one sensor.

Table 4 shows results obtained with a sensor failure. Results emphasize that the method still performs well when data from one view are lost at testing time. Indeed, whichever view is missing, results do not vary significantly, showing the method's ability to exploit all available data. Worst results are obtained when skeleton data were available at training time and dropped at testing time and, still, the algorithm shows an accuracy of 72.2% which is in the range of results obtained with only one feature.

These results show a good robustness of the algorithm to sensor failure or missing data, which is a crucial characteristic of real systems.

Table 4 Average accuracy (%) on TUM dataset when a particular sensor is unavailable at testing time

Learning data	4 views	4 views + skeleton
All features	83.1	85.0
View 0 out	77.7	83.1
View 1 out	80.2	83.3
View 2 out	82.2	84.5
View 3 out	81.3	85.0
Skeleton out	X	72.2

Data fusion of views or skeleton is done at the middle level of fusion

6 Computation time and latency

In this section, we evaluate two relevant performance criteria in the context of real applications: computation time and latency. The first one depends on the detection method and on input features. The second one is defined as the delay between input frame and the corresponding output decision.

6.1 Latency

In applications such as health care or robot interaction, low latency can be very important to guaranty a safe usage. In this section, we investigate the influence of latency on detection performances.

In the DOHT Paradigm, a frame f_t captured at time t receives votes from features extracted at time $t+\delta_t$, $\delta_t \in [-M;M]$. As a result, action prediction for the frame f_t is fully completed after time $t+M$. In other words, the DOHT paradigm has a maximal theoretical prediction latency value equal to M, which is half the size of the voting window.

However, since Hough score is a progressive sum of received votes, it can be calculated frame after frame. This way, we can compute a partial Hough score at each feature extraction time. Let us denote $\mathcal{H}^{[T_1,T_2]}(t,a)$ the Hough score for frame f_t extracted at time t, regarding action a and computed from votes extracted on $[T_1,T_2]$.

In order to investigate the influence of latency $-M<l<M$ on detection rates, we computed a partial Hough score $\mathcal{H}^{[-M,l]}(t,a)$ rather than a completed score. Note that when l is negative, prediction is done using only frames extracted before time t, and that this is predicting the action of frame t before this frame has actually been observed.

Figure 8 shows accuracy of the algorithm depending on M and latency l. We observe a strong improvement around $l=0$—and less variations after $l=10$ for all configurations. This is consistent with the fact that DOHT method mainly votes around the central frame t. The figure also shows the impact of parameter M on accuracy results. One could think that accuracy would increase with M since we exploit more and more information, but we observe that results are higher when $M=20$—corresponding to a vote window of 40 frames/1.6 s—with a score of 86.1% which is our best result, and slightly decrease when M grows. This is a consequence of the implementation of the decreasing constraint (Eqs. 4, 7) in approximated DOHT. Indeed, this constraint is handled by a fixed number of intervals which individual size increases with M. Some weights θ become thus less discriminant since they receive predictions from numerous frames. For comparison, Fig. 10 shows action length distribution in the TUM Kitchen Dataset. This

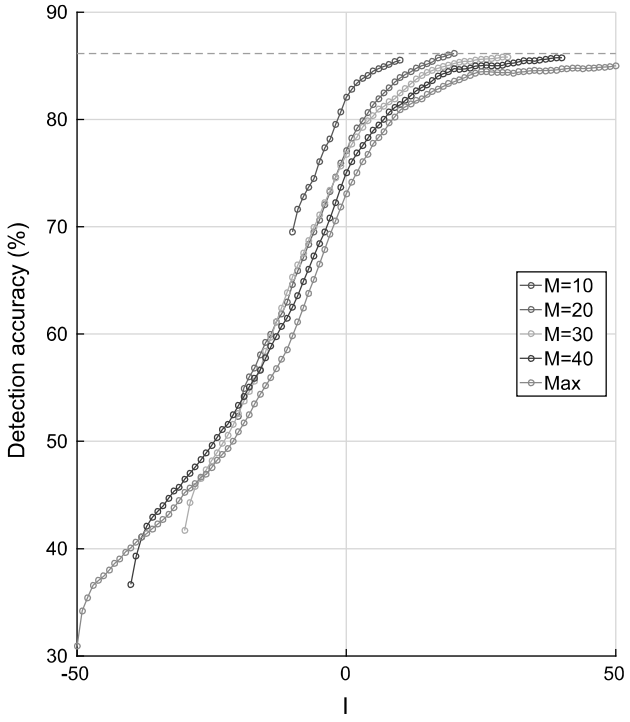

Fig. 8 Prediction accuracy relatively to latency. Algorithm had been performed on all features, namely 4 views fusion with TS + HOG feature and skeleton

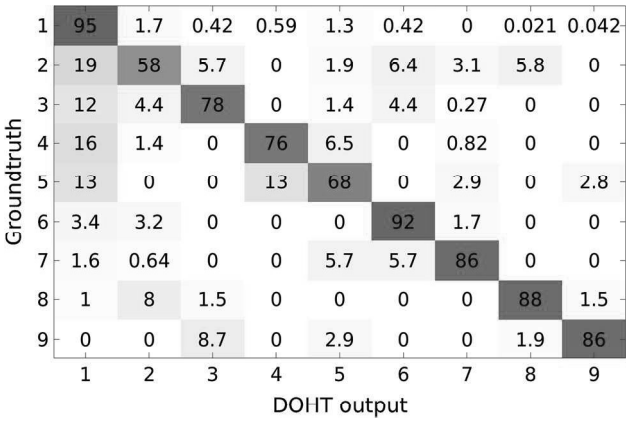

Fig. 9 Confusion matrices when combining information from all views and skeleton data. For class correspondences, report to Fig. 6

length has to be compared to 2 M since codewords localized in the interval $[t - M, t + M]$ are used to predict action at time t. We provide the confusion matrix for this configuration (Fig. 9), showing that fusion of descriptors reduces confusion between classes (compared to Fig. 6 for instance).

These results emphasize the fact that the algorithm gives pretty good results with a short latency, even when computed with a large voting window, since it computes a result evolving frame after frame. Another interpretation

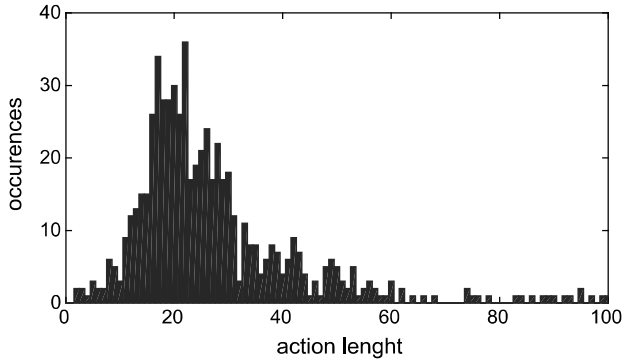

Fig. 10 Action lengths histogram in the TUM Kitchen database

could be that the algorithm efficiently exploit information from action beginning. This property suits well for real-life applications.

6.2 Computation time

As emphasized in [9], DOHT detector's complexity for each frame extracted at time $t \in [0, T]$ is $O(M, A)$ since votes are computed on a time window of size $2M$ for each action $a \in \mathcal{A}$ with $A = \text{card}(\mathcal{A})$. As a result, computation time for each frame varies linearly with M, A and the number of features extracted in this frame.

Figure 11 shows computation time of our algorithm in relation with M, half the size of the considered time window during voting step. Note that we did not optimize the extraction step and that we used available code from [49] for visual feature extraction. This code uses a heavy grid sampling of trajectories which could be enhanced.

As expected, vote computation time grows linearly with M. For $M = 20$, which gives best results according to Sect. 6.1 (accuracy of 86.1%), the overall process considering all four views and skeleton data requires 0.127 s, with vote process only requiring 0.029 s. This corresponds to a 8 fps frame rate, without time optimization (task parallelization for instance). This confirms that DOHT is suitable for real-time applications.

When using only skeleton data, as in [9], the number of extracted features at each frame is lower. With $K = 5$ centers—for K-means—we observe a maximum computation time of 1.02 ms (Fig. 12). Note that this result does not consider extraction time since skeleton features are directly provided in the TUM dataset. For comparison, when using the same $M = 50$ with dense trajectories, computing time is 98 ms. Note that this time can be reduced with optimization and dimension reduction.

All time were measured on an "Intel(R) Xeon(R) CPU E5-2687W 0 @ 3.10GHz", using 16 cores when computing votes, without heavy optimization.

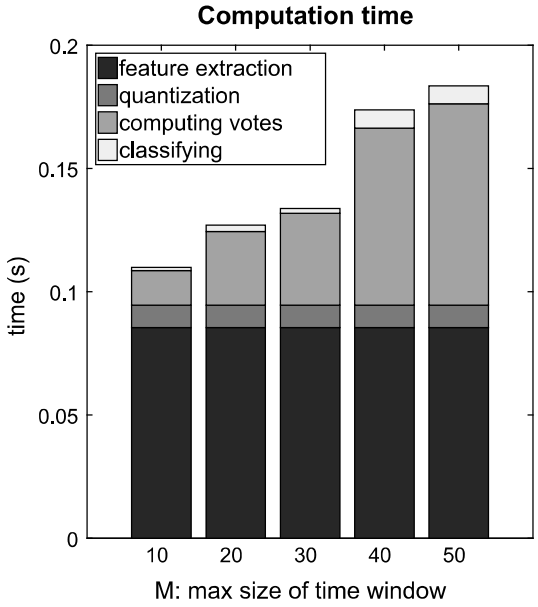

Fig. 11 Computation time using 4 views and skeleton according to M

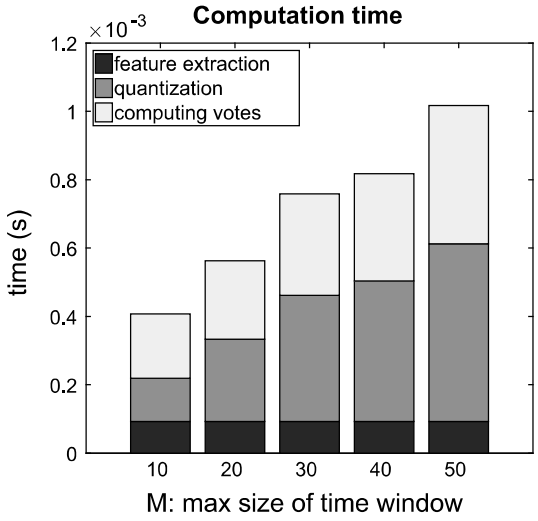

Fig. 12 Computation time using only skeleton data. Since skeleton data are given in the TUM dataset, extraction time is not available

6.3 Comparison with state-of-the-art methods

In this section, we compare our results to state-of-the-art algorithms. Results are presented in Table 5 according to original labels proposed by [47] or after splitting the first class (*Carrying while locomoting*) into two classes namely *Walking* and *Standing*, following [57].

Our method outperforms previous works presented in [8], showing that the DOHT algorithm can benefit from information fusion. Without further optimization and with the same parameters, we compare our result on modified labels proposed in [57] and our method achieves comparable results. Note that we did not include results published in [5] since this work uses segmented data.

Table 5 Comparison with published results on TUM datasets

Method	Result
Original TUM labels	
DOHT [8]	81.5
DOHT (27 joint skeleton)	83.0
ours (HOG + HOF, one view, M=50)	82.5
ours (all Views, M=20)	84.6
ours (all Views + Skeleton, M=20)	**86.1**
Modified label (following[57])	
All features + HF [57]	81.5
ours (all Views + Skeleton)	81.6

Results are just extracted from the corresponding papers and do not come from reimplementation

7 Conclusion

In this article focusing on action detection, we presented an information fusion method aiming to benefit from as much information as possible. This information fusion has been performed at three different levels in the deeply optimized Hough transform paradigm which fits well for real-time systems thanks to its ability to deal with unsegmented data and its low latency property. The *medium-level fusion* enables to merge effectively information from completely different sensors as RGB and skeleton for instance.

We evaluated this method on the *TUM Kitchen Dataset* since it is composed of various views and allows combining skeleton and RGB data targeting human action detection. We used the well-known dense trajectories features as input to our detection process and more precisely Trajectory Shape, Histogram of Gradient and Histogram of Optical Flow.

Feature fusion has first been evaluated at three different levels. Then, we evaluated the DOHT ability to successfully combine information provided by different views from the dataset. Finally, we merge all views, relevant features and skeleton data.

Our results emphasized that the method successfully merges available information in the context of action recognition with detection accuracy close to state-of-the-art results.

Furthermore, it has proven its robustness to information loss which is a relevant issue for real-world applications where sensors can sporadically fail to deliver a signal.

Finally, we evaluated the latency of the DOHT algorithm—induced by the voting time window—and obtained best results, 86.1%, when outputting action with a delay of 20

frames and a computation time of 0.127 s. This includes dense trajectories extraction, which could be accelerated. When using skeleton data only, overall computation time is 1 ms. Theses results show that the DOHT algorithm is suited for real-time applications dealing with human action detection.

References

1. Allmen, M., Dyer, C.R.: Cyclic motion detection using spatiotemporal surfaces and curves. In: Pattern Recognition, 1990. Proceedings., 10th International Conference on, vol. 1, pp. 365–370. IEEE (1990)
2. Allmen, M.C.: Image sequence description using spatiotemporal flow curves: toward motion-based recognition. Ph.D. thesis, Citeseer (1991)
3. Amor, B.B., Su, J., Srivastava, A.: Action recognition using rate-invariant analysis of skeletal shape trajectories. IEEE Trans. Pattern Anal. Mach. Intell. **38**(1), 1–13 (2016)
4. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: Human Behavior Understanding, pp. 29–39. Springer (2011)
5. Barnachon, M., Bouakaz, S., Boufama, B., Guillou, E.: Ongoing human action recognition with motion capture. Pattern Recognit. **47**(1), 238–247 (2014)
6. Cedras, C., Shah, M.: Motion-based recognition a survey. Image Vis. Comput. **13**(2), 129–155 (1995)
7. Chakrabartty, S., Cauwenberghs, G.: Gini support vector machine: quadratic entropy based robust multi-class probability regression. J. Mach. Learn. Res. **8**(Apr), 813–839 (2007)
8. Chan-Hon-Tong, A., Achard, C., Lucat, L.: Deeply optimized hough transform: Application to action segmentation. In: Image Analysis and Processing–ICIAP 2013, pp. 51–60. Springer (2013)
9. Chan-Hon-Tong, A., Achard, C., Lucat, L.: Simultaneous segmentation and classification of human actions in video streams using deeply optimized hough transform. Pattern Recognit. **47**(12), 3807–3818 (2014)
10. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
11. Cuntoor, N.P., Chellappa, R.: Epitomic representation of human activities. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pp. 1–8. IEEE (2007)
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886–893. IEEE (2005)
13. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Computer Vision–ECCV 2006, pp. 428–441. Springer (2006)
14. Darrell, T., Pentland, A.: Recognition of space-time gestures using a distributed representation. In: Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology (1993)
15. Darrell, T., Pentland, A.: Space-time gestures. In: Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on, pp. 335–340. IEEE (1993)
16. Davis, J., Shah, M.: Visual gesture recognition. IEE Proc. Vis. Image Signal Process. **141**(2), 101–106 (1994)
17. Feldman, J.A.: Four frames suffice: a provisional model of vision and space. Behav. Brain Sci. **8**(02), 265–289 (1985)
18. Feng, X., Perona, P.: Human action recognition by sequence of movelet codewords. In: 3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on, pp. 717–721. IEEE (2002)
19. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: Internationnal Conference on Computer Vision and Pattern Recognition (2009)
20. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE. Trans. Pattern. Anal. Mach. Intell. **33**(11), 2188–2202 (2011)
21. Goddard, N.H.: The perception of articulated motion: recognizing moving light displays. Tech. rep, DTIC Document (1992)
22. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the fourth alvey vision conference. pp. 147–151 (1988)
23. Hoai, M., Lan, Z.Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3265–3272. IEEE (2011)
24. Hu, J.F., Zheng, W.S., Lai, J., Zhang, J.: Jointly learning heterogeneous features for rgb-d activity recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5344–5352 (2015)
25. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 221–231 (2013)
26. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural SVMs. Mach. Learn. **77**(1), 27–59 (2009)
27. Johansson, G.: Visual perception of biological motion and a model for its analysis. Percept. Psychophys. **14**(2), 201–211 (1973)
28. Kosmopoulos, D.I., Papoutsakis, K., Argyros, A.A.: Online segmentation and classification of modeled actions performed in the context of unmodeled ones. Trans. PAMI **33**(11), 2188–2202 (2011)
29. Krüger, V., Kragic, D., Ude, A., Geib, C.: The meaning of action: a review on action recognition and mapping. Adv. Robot. **21**(13), 1473–1501 (2007)
30. Kuehne, H., Arslan, A., Serre, T.: The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 780–787 (2014)
31. Laptev, I.: On space-time interest points. Int. J. Comput. Vis. **64**(2–3), 107–123 (2005)
32. Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1–8. IEEE (2008)
33. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
34. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: Workshop on Statistical Learning in Computer Vision (2004)
35. Maji, S., Malik, J.: Object detection using a max-margin hough transform. In: Internationnal Conference on Computer Vision and Pattern Recognition (2009)
36. Matikainen, P., Hebert, M., Sukthankar, R.: Trajectons: Action recognition through the motion analysis of tracked features. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pp. 514–521. IEEE (2009)
37. Messing, R., Pal, C., Kautz, H.: Activity recognition using the velocity histories of tracked keypoints. In: Computer Vision, 2009 IEEE 12th International Conference on, pp. 104–111. IEEE (2009)
38. Müller, M.: Dynamic time warping. In: Information retrieval for music and motion, pp. 69–84. Springer, Heidelberg (2007)
39. Ni, B., Moulin, P., Yang, X., Yan, S.: Motion part regularization: Improving action recognition via trajectory selection. In:

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3698–3706 (2015)
40. Ni, B., Pei, Y., Moulin, P., Yan, S.: Multilevel depth and image fusion for human activity detection. IEEE Trans. Cybern. **43**(5), 1383–1394 (2013)
41. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: Computer Vision–ECCV 2006, pp. 490–503. Springer (2006)
42. Oreifej, O., Liu, Z.: Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 716–723. IEEE (2013)
43. Peng, X., Wang, L., Wang, X., Qiao, Y.: Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. arXiv preprint arXiv:1405.4506 (2014)
44. Poppe, R.: A survey on vision-based human action recognition. Image Vis. Comput. **28**(6), 976–990 (2010)
45. Song, Y., Liu, S., Tang, J.: Describing trajectory of surface patch for human action recognition on rgb and depth videos. IEEE Signal Process. Lett. **22**(4), 426–429 (2015)
46. Sun, J., Wu, X., Yan, S., Cheong, L., Chua, T., Li, J.: Hierarchical spatio-temporal context modeling for action recognition. In: Internationnal Conference on Computer Vision and Pattern Recognition (2009)
47. Tenorth, M., Bandouch, J., Beetz, M.: The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In: International Conference on Computer Vision Workshops (2009)
48. Vieira, A.W., Nascimento, E.R., Oliveira, G.L., Liu, Z., Campos, M.F.: Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 252–259. Springer (2012)
49. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action Recognition by Dense Trajectories. In: IEEE Conference on Computer Vision & Pattern Recognition, pp. 3169–3176. Colorado Springs, United States (2011). http://hal.inria.fr/inria-00583818/en
50. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3551–3558. IEEE (2013)
51. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 1290–1297. IEEE (2012)
52. Wang, J., Nie, X., Xia, Y., Wu, Y., Zhu, S.C.: Cross-view action modeling, learning, and recognition. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pp. 2649–2656. IEEE (2014)
53. Wang, Y., Mori, G.: Learning a discriminative hidden part model for human action recognition. In: Advances in Neural Information Processing Systems, pp. 1721–1728 (2009)
54. Wohlhart, P., Schulter, S., Kostinger, M., Roth, P., Bischof, H.: Discriminative hough forests for object detection. In: Conference of British Machine Vision Conference (2012)
55. Xia, L., Aggarwal, J.: Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 2834–2841. IEEE (2013)
56. Yamato, J., Ohya, J., Ishii, K.: Recognizing human action in time-sequential images using hidden markov model. In: Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on, pp. 379–385. IEEE (1992)
57. Yao, A., Gall, J., Fanelli, G., Van Gool, L.: Does human action recognition benefit from pose estimation? In: Conference of British Machine Vision Conference (2011)
58. Yao, A., Gall, J., Van Gool, L.: A hough transform-based voting framework for action recognition. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 2061–2068. IEEE (2010)
59. Yu, G., Yuan, J.: Fast action proposals for human action detection and search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1302–1311 (2015)
60. Zhang, J., Gong, S.: Action categorization with modified hidden conditional random field. Pattern Recognit. **43**(1), 197–203 (2010)
61. Zhang, Y., Chen, T.: Implicit shape kernel for discriminative learning of the hough transform detector. In: Conference of British Machine Vision Conference (2010)

Geoffrey Vaquette was born in France in 1990. He received the M.S. degree from Ecole Normale Superieure de Cachan, in 2014, and is currently a Ph.D. candidate at the Pierre and Marie Curie University in Paris, France. His current research interests are human activity detection, machine learning, data mining.

Catherine Achard was born in France in 1970. She received her Ph.D. in computer vision from Blaise Pascal University, France in 1996. Since 1997, she is an Associate Professor at the Pierre and Marie Curie University in Paris, France. Her current research interests include, image and video processing, and specifically the study of human behavior and natural interaction.

Laurent Lucat received his Ms.D. from Paris VI University/IRCAM (1995) and Ph.D. in computer science from Nantes University (1998). From 1999 to 2007, he worked with Philips Mobile Phones, mostly involved in audio compression and synthesis as well as image enhancement. He joined the Vision Lab from CEA LIST (Paris-Saclay), in 2008, where he focuses on people detection and human activity analysis.