Simultaneous segmentation and classification of human actions in video streams using *deeply optimized Hough transform*

Adrien CHAN-HON-TONG[a], Catherine ACHARD[c,b], Laurent LUCAT[a]

[a] *CEA, LIST, Laboratoire Vision et Ingénierie des Contenus, Centre d'études de Saclay, Route Nationale, 91400 Gif-sur-Yvette, FRANCE*
{*adrien.chan-hon-tong,laurent.lucat*}*@cea.fr*
[b] *CNRS, UMR 7222, ISIR, F-75005, FRANCE*
[c] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, FRANCE*
catherine.achard@upmc.fr

Abstract

Most researches on human activity recognition do not take into account the temporal localization of actions. In this paper, a new method is designed to model both actions and their temporal domains. This method is based on a new *Hough* method which outperforms previous published ones on *honeybee* dataset thanks to a deeper optimization of the *Hough* variables. Experiments are performed to select skeleton features adapted to this method and relevant to capture human actions. With these features, our pipeline improves state-of-the-art performances on *TUM* dataset and outperforms baselines on several public datasets.

Key words: human actions, segmentation, classification, video streams, Hough

1. Introduction

Human activity recognition is becoming a major research topic (see [1, 30] for reviews). The ability to recognize human activities would enable the development of several applications like gestures based interfaces/interactions or multimedia data mining on web data. However, requirements for human activity recognition systems depends on the targeted applications. For example, on one hand, human action recognition systems for data mining on web data [23] deal with heterogeneous contents, heterogeneous resolutions and severe camera motions, but are not expected to be real time or to deal correctly with all the web database content. On the other hand, systems for gestures based interaction [31] deal with homogeneous videos acquired under controlled setting but need the highest possible accuracy and real time capacity.

Action recognition in natural and unconstrained videos has known a turning point from [33], driven by progresses in object recognition in image. For this task, most of the state-of-the-art approaches [42] are based on *Bag Of Words*

[35] (*BOW*) and *Support Vector Machine* [10, 8] (*SVM*) pipeline that captures the link between low-level features e.g. *spatiotemporal interested points* [33] (*STIP*) or trajectories of interested points [37, 42] and actions through a learning process.

Gestures based interaction is independently reaching a turning point with the release of active camera, such as the *Kinect*, providing suitable information for human action recognition like depth map [27], and even, under controlled setting, the skeleton of the subject [31] (see [14] for a review).

Beside gestures based interaction and offline web data annotation, a lot of applications are emerging like ambient intelligence [29] or medical video protection [47]. A typical example is the detection of daily actions to monitor people with a limited autonomy (elderly or disabled person) at home. Indeed, the distribution of their daily actions during a day or a week has a medical relevancy.

This last kind of applications introduces new challenges which are addressed in this paper. In particular, the temporal location of an action is as relevant as the action itself in video based monitoring. It is not the case in human action recognition in web databases, where the goal is to annotate a whole (i.e. segmented) video. For such new applications, the aim is simultaneous classification and temporal localizations of actions. Thus, approaches like *BOW+SVM* which discard temporal information can not deal with such a challenge. As there is no clear consensus in literature about how this localization should be evaluated, we focus in this paper, on determining **at each time** what action is performed by an actor among a set of actions. This task is called *segmentation* in this paper.

Moreover, vision based monitoring systems need to take into account the temporal context. In this way, they oppose gesture based interaction systems where features extracted in a specific time are enough to decide the corresponding action. In particular, the duration of an action is about two orders of magnitude larger in vision based monitoring data than in interface/interaction ones: the maximal size of an action is less than a couple of second for gestures based mouse (e.g. [19]) or 4s in [31] (dance move) against around 30s in [39, 43] (setting a table) and around 2 minutes for [38] (cooking, working at computer).

Among the introduced challenges, daily motions present a higher variability than interface ones where subjects try to perform predefine motions. Thus, direct template matching [24, 31] may not be efficient for daily actions. Also, skeleton extraction is not as accurate for daily action recognition as for gestures based interface. However, considering the large academic efforts to deal with general pose estimation [12, 34], we expect the skeleton to be available in the future in medical surveillance field, and thus, we still rely on skeleton features in this paper.

To tackle these challenges, we introduce two contributions which are presented after related works (section 2). First, in section 3 and 4, we introduce a new optimized *Hough Transform* method and present experiments where this new method outperforms previous *Hough Transform* ones. Then, in section 5, we perform experiments which highlight the relevancy of this method for human

action segmentation in video streams, as it improves state-of-the-art results on several public datasets using skeleton based features. Perspectives opened by these contributions are presented together with conclusions (section 6)

2. Related works

Most related works are based on graphical model like [38, 48, 22, 26]. In [38], a two-layered *Maximum Entropy Markov Model* (*MEMM*) is built on pose based features such as proximity of hands with other articulations, positions of articulations, speeds. In [22], recognition is performed by an *Action Graph* based on velocity features. In [48], *Hidden Markov Model* (*HMM*) is learnt on skeleton representations. The ability of latent variables to extract middle level features by clustering similar successive features is currently a major improvement axis [44, 36] for such *Conditional Random Field* [18] (*CRF*) pipelines. Such middle level features are also considered in [40] in a naive Bayesian classifier context and lead to performance improvements.

Besides graphical models, works like [15] deal with the segmentation challenge by padding the video with windows on which a *BOW+SVM* pipeline is applied. By dynamic programming, an optimal padding of the video is computed, thus we call this method *OP* (for optimal padding). *Hough Forest* can also perform such segmentation as in [49], where skeleton stream patches are densely extracted and clustered by trees formed with weak binary classifiers based on distances between articulations. Then, each patch votes, and vote agglomeration provides a segmentation after smoothing.

In addition, an important point to mention is that the rising of such methods is slowed down by the few public databases available.

2.1. Available public databases

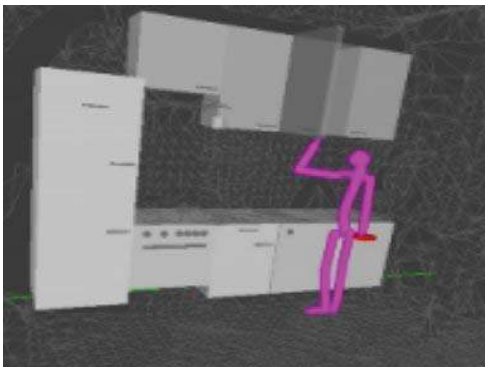
In [15], a segmentation method is presented but mainly applied on *Weizmann* [13] and *Hollywood* [20] datasets where actions are associated with whole (or segmented) videos. This is also the case in [38] which introduces a typical dataset for classification and presents a method for segmentation. The *honeybee* dataset [28] considered in [15] is not relevant for human action recognition but is relevant for general segmentation: [28] presents an algorithm for segmentation based on *Linear Dynamic System* (*LDS*) [32] (a graphical model with *continuous state*).

This is the opposite situation in [48] where unsegmented videos are provided but the presented algorithm, based on *HMM*, performs classification but not segmentation.

Considering [7], it appears that there are only few datasets providing unsegmented videos i.e. with different actions in different parts of the video **and** where the ground truth provides the temporal localization of actions (e.g. one action by frame). This is even worse, if we consider in these datasets only those containing skeleton data and/or focusing on daily actions.

As a result, to evaluate our algorithm, we mainly rely in this paper on *UTKAD* for *TUM* [39] and *University of Texas Kinect-Action Dataset* [48]

example of action: *Lowering an object*



Provided skeleton

Figure 1: TUM dataset [39].

which are relatively similar datasets: *TUM* is a multi-sensor dataset that contains especially skeleton streams (fig. 1). Actions are performed by 5 subjects setting the table (e.g. *Lowering an object*, *Opening a drawer*). *UTKAD* is a set of videos acquired with a Kinect that provides skeleton streams. The actions are performed by 10 subjects in a human-machine interaction setting (e.g. *clap hands*, *wave hands*). Both datasets are composed of around 20 sequences, about 2 minutes each, containing around 10 kinds of actions.

These datasets contain middle length videos. However, our objective is to offer a pipeline that can be used on stream context for daily action segmentation (e.g. 24 hours of video surveillance).

2.2. Stream constraints

In stream context, segmentation algorithms can not store nor process the entire video to take a decision concerning the human action performed at a specific time. Hence, all algorithms with an unbounded latency like [26] or [28] (*LDS*) can not be applied. Also, for the same stream constraints, algorithms with a low computational complexity are required. This discards approach like [15] where a *SVM* classifier [10] is applied to a number of windows which is quadratic in the maximal size of an action, which may be too large for daily actions. Finally, *LDS*-like algorithms can be degraded: decisions can be taken with bounded latency. However, these degraded versions have also degraded complexity.

For these two reasons, we focus on *Hough Transform* (*HT*) which allows both bounded latency and linear complexity (see table 1).

3. *Hough transform* based segmentation

3.1. Voting process

The main idea of the probabilistic *HT* for non parametric object (e.g. human activities) introduced in [21] is to perform the detection as a local recognition problem in the voting space built by agglomerating feature votes. The bounded

Method	Bounded latency	small complexity
LDS e.g. [28]	×	✓
degraded *LDS*	✓	×
OP e.g. [15]	✓	×
HT e.g. [49]	✓	✓

Table 1: Computational ability for algorithms like *Linear Dynamic System based smoothing*, *Optimal Padding* and *Hough Transform*.

latency and linear complexity of *HT* are induced by this property of local detection.

In the context of temporal detection applied to human action in streams, the *HT* is composed of three steps:

1: Feature extraction and quantization form codewords

2: Each codeword votes for the possible temporal centers of the possible actions according to specific learned weights

3: All the votes are agglomerated to form the Hough score from which detection decisions are taken

In this article, we offer a new formalism for *HT* based methods. It consists to model the steps 2 and 3 by introducing a function $\theta()$ linking codewords, time displacements, and human actions, to vote weights. Let c be a codeword extracted at time t. It votes with a weight $\theta(a, \delta_t, c)$ for the hypothesis that the current action is an action of type a and is centered at time $t + \delta_t$ (this weight does not depend on the time t but only on a, δ_t and c). Then, given the set of localized codewords extracted from the video $\mathcal{V} = \{c, t\}$ (c for codewords and t for the corresponding extraction time), the Hough score $\mathcal{H}_\mathcal{V}$ that an action a is centered at the time t' is:

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

Salient peaks of $\mathcal{H}_\mathcal{V}$ form the detections in video $\mathcal{V}$ (for simplicity, both the video and the set of localized codewords are designed by $\mathcal{V}$).

3.2. State-of-the-art on training processes

All *HT* based methods can be described using the proposed formalism: they differ only in their estimation of the weights $\theta(a, \delta_t, c)$. Some of these methods consider only statistics on the training database [21], while the other ones introduce an optimization step [25, 46, 50]. In this section, we introduce the main methods of the state of the art and describe them with the proposed formalism.

Implicit Shape Model:

In the *Implicit Shape Model* (*ISM*) [21], the *HT* is based on generative weights. Let $\mathcal{P}(a, \delta_t|c)$ be the probability that the action occurring at time t is a and is centered at time $t + \delta_t$, knowing that a codeword c has been extracted at time t. This probability is estimated with statistics of the training dataset and is supposed to be independent of t (it depends only on a, δ_t and c). Then, the weights are given by:

$$\theta_{ISM}(a, \delta_t, c) = \mathcal{P}(a, \delta_t|c)$$

The probability $\mathcal{P}(a, \delta_t|c)$ is estimated by:

$$\mathcal{P}(a, \delta_t|c) \approx \frac{N(a, \delta_t, c)}{N(c)}$$

where $N(a, \delta_t, c)$ is the number of times a codeword c has been extracted at time t in an action a centered in $t + \delta_t$ and $N(c)$ is the number of occurrences of the codeword c.

This generative *ISM* approach has the advantage to be meta parameter-free and relatively robust to over-training. However, [25, 46, 50] report that this method produces false positives which can be partially reduced by introducing discriminative parameters.

Max-Margin Hough Transform:

In *Max-Margin Hough Transform* (*MMHT*) [25], a coefficient is introduced for each codeword to weight the *ISM* values, resulting in:

$$\theta_{MMHT}(a, \delta_t, c) = w_c \times \theta_{ISM}(a, \delta_t, c) = w_c \times \mathcal{P}(a, \delta_t|c)$$

The weights w_c give more or less importance to the different codewords c according to their discriminative power. They are learnt simultaneously in a discriminative way through an optimization process similar to a *SVM*.

Implicit Shape Kernel:

In *Implicit Shape Kernel* (*ISK*) [50], the votes are also based on the *ISM* generative ones, but some coefficients are introduced to weight the different training examples. Hence, *ISK* training leads to:

$$\theta_{ISK}(a, \delta_t, c) = \sum_i w_i \times \mathcal{P}_i(a, \delta_t|c)$$

where $\mathcal{P}_i(a, \delta_t|c)$ is an estimation of the probability $\mathcal{P}(a, \delta_t|c)$ based only on the training example i. The weights w_i are learnt simultaneously in a discriminative way using a specific *kernel-SVM* training [50].

MMHT and *ISK* report experimental improvements over *ISM* by adding discriminative parameters. This trend is also supported by [46].

ISM + SVM:

In [46], it is shown that learning directly the *ISM* Hough map $\mathcal{H}(t', a)$ with a linear *SVM* is equivalent to the *ISM* combined with a weighting coefficient for each displacement. This results in:

$$\theta_{ISM+SVM}(a, \delta_t, c) = w_{\delta_t} \times \mathcal{P}(a, \delta_t | c) = w_{\delta_t} \times \theta_{ISM}(a, \delta_t, c)$$

Hough forest:

To our knowledge, *ISM* [21] and the presented extensions [25, 46, 50] are the only published methods to estimate the weights of *Hough transform*. More precisely, these methods define links between codewords and votes. There are, of course, various ways to select the features and the codewords, like the *Hough forest* (*HF*) [49] which are major methods of the state of the art. *Hough forest* uses *ISM* votes, but the mapping between features (usually data patches) and codewords (a leaf in a weak binary classifier tree) is constructed such that all training features associated with the same codeword are expected to come from training examples with a same label (e.g. same action). Several works (e.g. [49]) report that this automatic feature mapping process associated with *ISM* votes leads to significant experimental improvements against codewords obtained without learning, e.g. by K means algorithm.

However, in this section, we focus on the optimization of the weights defining the voting process i.e. on the link between codewords and votes which is generic in terms of features and codewords. Thus, the proposed method can be employed with codewords provided by *Hough forest* approach.

One common point of [25, 46, 50] is that the addition of discriminative parameters to influence *ISM* votes increases performances of *HT*. Hence, a natural idea is to learn directly all votes. But before, let us introduce some standard *HT* post-processing.

3.3. From detection to segmentation

Post-processing for segmentation:

In temporal action **detection** context, the goal is to find the temporal bounding box of each action, or often simply the center of each action. In **segmentation** context: the goal is to determine which action is performed by the actor among a set of actions in each frame.

Most of the articles of the *HT* literature (and particularly those presented before) focus on detection, hence, each codeword naturally votes for **centers** (temporal center of each action). Thus, these methods need to be adapted to perform segmentation.

Smoothing based segmentation:

To our knowledge, in literature, this adaptation is done by propagating the votes from centers to all frames thank to a smoothing function. For example, in [49], this is done by applying a strong Gaussian smoothing on $\mathcal{H}$:

$$\mathcal{H}(t'', a) \leftarrow \sum_{t'} \mathcal{H}(t', a) \exp\left(-\frac{1}{2\sigma^2}(t' - t'')^2\right)$$

Such smoothing can be applied even in a detection context to group close peaks. However, in segmentation context this smoothing should be more intense as each frame must receive a piece of the votes from centers.

Then, the decision about the action at time t' is given by:

$$\widehat{a}\left(t'\right) = \arg\max_{a}\left(\mathcal{H}\left(t', a\right)\right) \tag{2}$$

As smoothing may have a significant impact on performances in segmentation context, it is interesting to learn an adapted smoothing e.g. to learn a specific variance for each action [49], or more generally, to learn completely some specific smoothing functions (especially not necessarily Gaussian).

Beyond smoothing:

However, this smoothing approach has a major drawback: as smoothing is performed separately from voting, it does not simultaneously learn how a codeword should vote and how the smoothing should be done.

Yet, this drawback can be solved by learning **already smoothed votes**. In this way, votes have just to be accumulated to perform segmentation following equation (2) (without any smoothing or other post-processes). Hence, the entire process, described by equation (1) and (2), is learnt in one step.

Let us point out that smoothed votes are **natural** for segmentation as they can be seen as votes for presence. In the detection context, each codeword votes for a position corresponding to the center of the action, while in the segmentation context, each codeword votes ideally for the interval where the action takes place (as illustrated in figure 2). This interval can be seen as a smoothed vote. Also, if the votes $\mathcal{P}\left(a \text{ is centered at } t'|c \text{ extracted at } t\right)$ are smoothed by the function $\mathcal{P}\left(t'' \text{ is part of } a|a \text{ is centered at } t'\right)$ then the result is the same as if votes are $\mathcal{P}\left(t'' \text{ is part of } a|c \text{ extracted at } t\right)$.

In addition, it is better to smooth individually the votes of each codeword and to accumulate the votes than to do the opposite (as mainly done in literature): *accumulate and smooth* is the particular case of *smooth and accumulate* when each codeword is smoothed in the same way. In contrary, *smooth and accumulate* allows to learn a specific smoothing shape for each couple codeword-action.

Hence, from now, codewords will not vote for the **center** of an action but for the **presence** of an action. Thus, a codeword c extracted at time t votes with a weight $\theta\left(a, \delta_t, c\right)$ for the hypothesis that an action a is **present** at time $t + \delta_t$. This does not change the form of the equation (1) but the semantic of θ and of $\mathcal{H}$ (which now represents the score that an action a is **present** at time t'). Then, equation (2) provides segmentation.

Decreasing constraints:

This semantic change leads to several consequences e.g. on perfectly structured data, all the codewords of an action vote equally for each time in the interval of the action and not only for the center (figure 2). These two ways are equivalent on perfect data as there is a clear correspondence between peaks

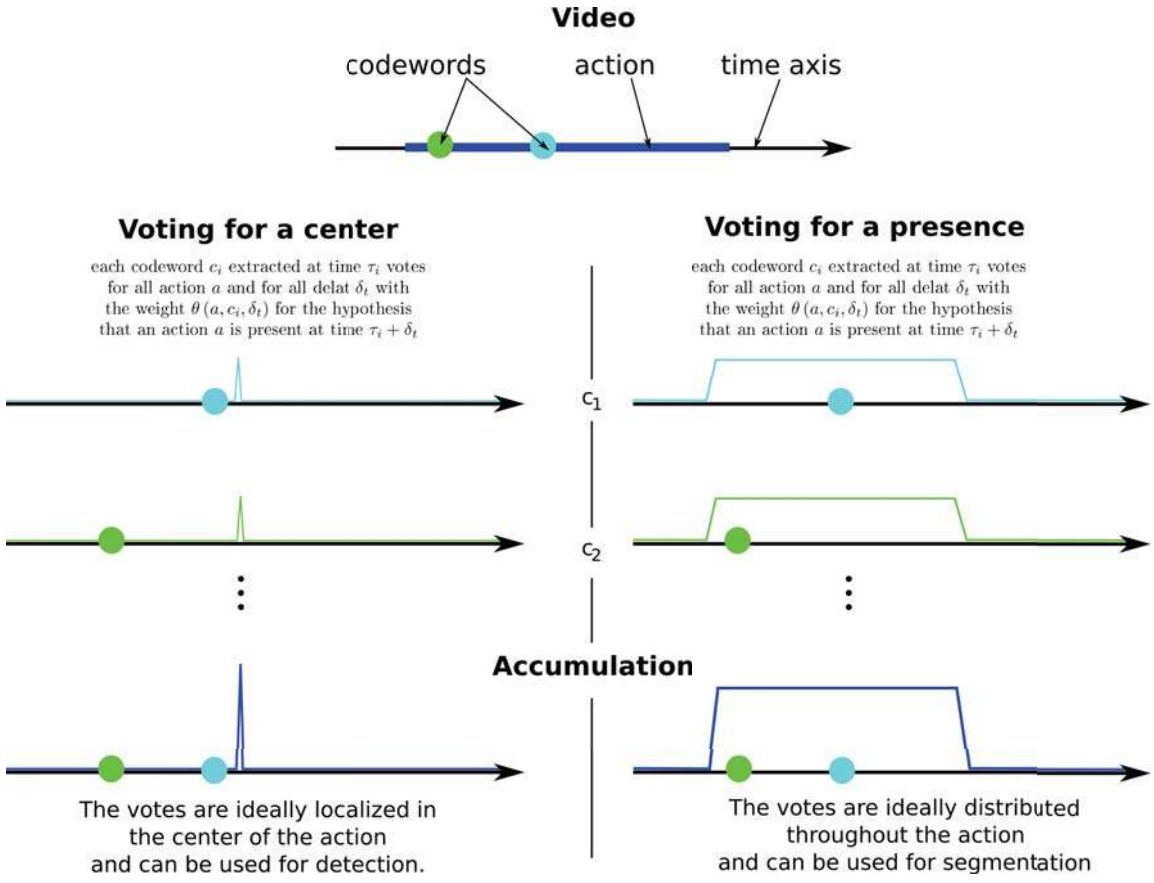


Figure 2: Voting for a center versus for a presence.

intensity and action duration, but the *presence* representation is more general (because going from interval to center is easy but not the opposite) and thus is more adapted to deal with real data.

Also, a smoothing function is naturally decreasing toward the distance to 0. In our context, the more t' is far from t the less a codeword extracted in t should provide information about t'. Later on in this paper, these constraints are called decreasing constraints and can be written mathematically as:

$$\forall a, c, \delta_{t1}, \delta_{t2} : \begin{cases} \delta_{t1} \leq \delta_{t2} \leq 0 \Rightarrow \theta(a, \delta_{t1}, c) \leq \theta(a, \delta_{t2}, c) \\ 0 \leq \delta_{t1} \leq \delta_{t2} \Rightarrow \theta(a, \delta_{t1}, c) \geq \theta(a, \delta_{t2}, c) \end{cases} \tag{3}$$

The advantage *to vote for presence and not centers* is the ability to learn an optimal already smoothed voting function $\theta()$ as presented in next section.

4. *Deeply optimized Hough Transform*

We offer a new training process for *Hough transform* where all votes namely all $\theta()$ values are directly optimized without using any generative values coming from *ISM*. Hence, the set of variables to optimize is indexed simultaneously by codewords, displacements and action and not only for one of them like in *MMHT* or *ISM + SVM* (see table 4). For this reason, this training process is called *Deeply Optimized Hough Transform* (*DOHT*).

methods	θ	variables
ISM [21]	$\theta_{ISM}(a,\delta_t,c) = \mathcal{P}(a,\delta_t\|w)$	-
MMHT [25]	$\theta_{MMHT}(a,\delta_t,c) = w_c \times \mathcal{P}(a,\delta_t\|c)$	w_a
ISK [50]	$\theta_{ISK}(a,\delta_t,c) = \sum_i (w_i \times \mathcal{P}_i(a,\delta_t\|c))$	w_i
ISM+SVM [46]	$\theta_{ISM+SVM}(a,\delta_t,c) = w_{\delta_t} \times \mathcal{P}(a,\delta_t\|c)$	w_{δ_t}
DOHT (our)	$\theta_{DOHT}(a,\delta_t,c) = w_{a,\delta_t,c}$	$w_{a,\delta_t,c}$

$P(a,\delta_t|c)$ is the probability that an action a occurring in t is centered at time $t+\delta_t$ knowing that a codeword c has been extracted at time t. $P_i(a,\delta_t|c)$ is the same probability estimated using only the training example i.

Table 2: The different learning methods of the *Hough transform*.

This process is straightforwardly defined by the goal of *selecting the optimal values for $\theta(a,\delta_t,c)$ that provide correct decisions using equations (1) (2) at testing time*. For all training examples $\mathcal{V}$ and all times t', the predicted action $\widehat{a}$ should be the real one a^* (known on the training data) i.e.

$$\forall \mathcal{V}, t' :\ \widehat{a}_{\mathcal{V}}(t') = a^*_{\mathcal{V}}(t')$$

Considering the definition (eq. (2)) of the predicted action $\widehat{a}$, we obtain:

$$\forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t') :\ \mathcal{H}_{\mathcal{V}}(t',a) < \mathcal{H}_{\mathcal{V}}(t', a^*_{\mathcal{V}}(t'))$$

However, dealing with strict inequality is not computationally feasible, thus we impose a margin of 1:

$$\forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t') :\ \mathcal{H}_{\mathcal{V}}(t',a) + 1 \leq \mathcal{H}_{\mathcal{V}}(t', a^*_{\mathcal{V}}(t'))$$

and by replacing $\mathcal{H}$ by the definition (eq. (1)), we obtain:

$$\forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t') : \sum_{(c,t)\in\mathcal{V}} \theta(a, t'-t, c) + 1 \leq \sum_{(c,t)\in\mathcal{V}} \theta(a^*_{\mathcal{V}}(t'), t'-t, c)$$

Hence, the optimal function θ is simply the solution of this set of data constraints to which the decreasing constraints (eq. (3)) are added.

However, to manage noisy training data, a soft margin framework is applied as in [8]: some variables ξ are introduced to allow some data constraints to be unsatisfied but these variables are minimized in order to reduce the number of unsatisfied constraints though a loss function L_{data}. Moreover, to prevent over-fitting, a regularity term L_{reg} is added to the objective function as in [4]. A coefficient C mitigates the trade off between the attachment to data and the regularity as in [8, 4]. Thus, our training process consists to solve:

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

under constraints :

$$\begin{cases} \forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t') : \sum\limits_{(c,t)\in\mathcal{V}} \begin{pmatrix} \theta(a^*_{\mathcal{V}}(t'), t'-t, c) \\ -\theta(a, t'-t, c) \end{pmatrix} + \xi(t') \geq 1 \\ \forall a, c, \delta_{t1}, \delta_{t2} : \begin{cases} \delta_{t1} \leq \delta_{t2} \leq 0 \Rightarrow \theta(a,\delta_{t1},c) \leq \theta(a,\delta_{t2},c) \\ 0 \leq \delta_{t1} \leq \delta_{t2} \Rightarrow \theta(a,\delta_{t1},c) \geq \theta(a,\delta_{t2},c) \end{cases} \end{cases} \quad (4)$$

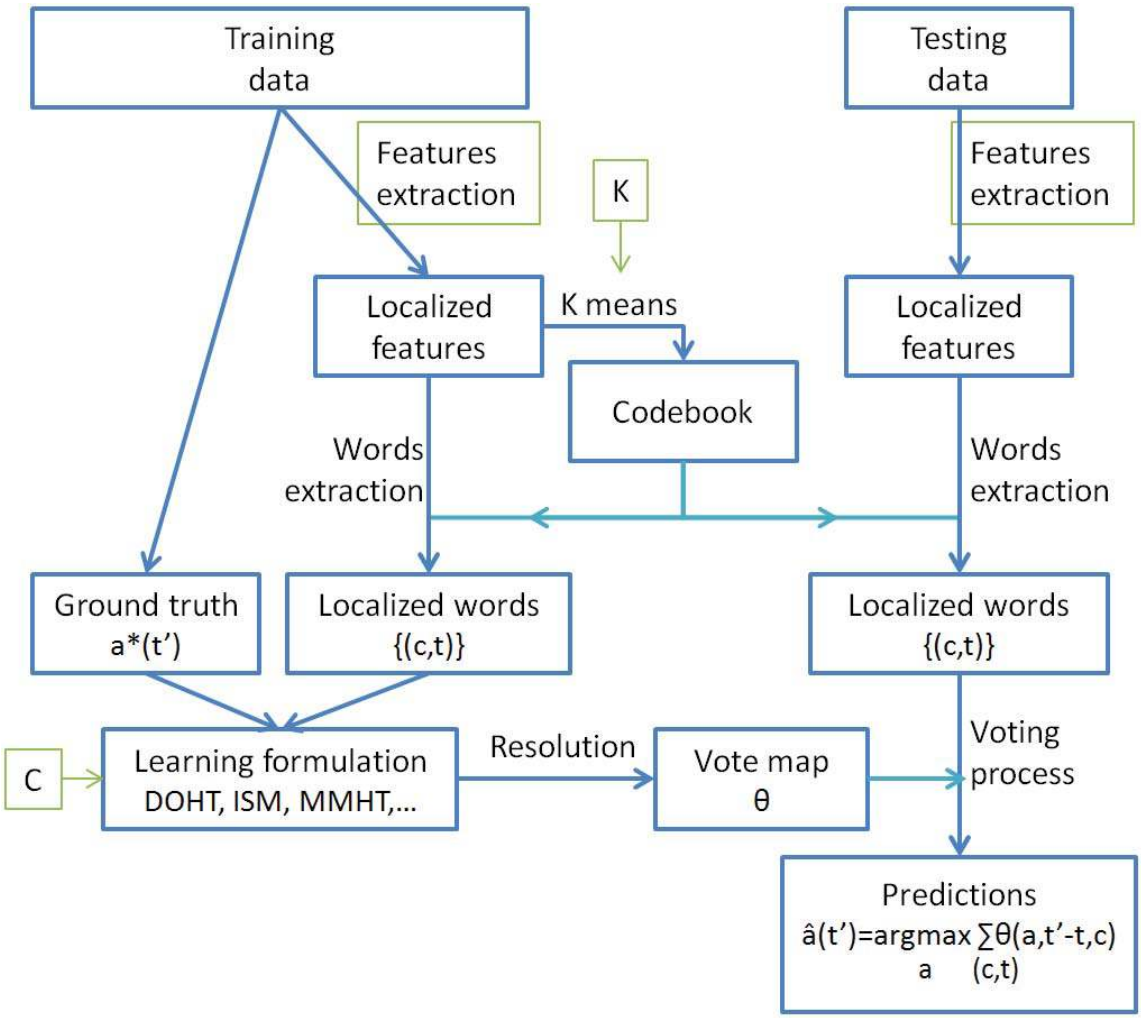


Figure 3: Overview of the *Hough* method.
Green indicated meta parameters namely what features are extracted and K and C for the training step. There is no meta parameter during testing time

There are to our knowledge no specific algorithm to deal with this formulation. The optimization framework eq. (4) seems very similar to *SVM*, but nearly all decreasing constraints may be saturated (whereas only a few data constraints, the *support vectors*, are usually expected to be saturated in *SVM*). This makes optimization challenging. Hence, we offer to use the L_1-norm for both L_{reg} and L_{data} as the problem becomes a linear program which is a well-studied problem in literature (eg. [17]) and which can be efficiency solved (for example using the solver CPLEX[1] available freely for academic purposes).

4.1. Overview

The figure 3 gives a complete overview of the pipeline.
This pipeline is the same for all methods from table 4: the only difference

[1] www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

is the cells *learning formulation*. If the *learning formulation* is the *DOHT* formulation (eq. (4)), then the pipeline is the *DOHT* pipeline. If the *learning formulation* consists to use the elementary probability as vote map, this is just *ISM* pipeline. In addition, prediction $\widehat{a}$ at frame t' during testing time in video V is simply, as indicated, equal to $\arg\max_a \left(\sum_{(c,t)} \theta\left(a, t'-t, c\right) \right)$. There is thus no difficulty to naively implement such pipeline.

For an efficient implementation, the voting process has to be done incrementally as the words are extracted. This can be done by the following pseudo-code:

```
//let T be the number of frame in the sequence
//let A be the number of action
//let M be the maximal size of a training action
//let c[t=1..T] be the word extracted at frame t
//let theta[a=1..A,dt=-M..M,c=1..max_words] be the voting weights
//then pred[t'=1..T] is the predicted action at frame t'
pred=prediction(T,A,M,theta,c)
  initialize pred[t'=1..T]
  initialize H[t'=1..T,a=1..A] to 0
  for t from 1 to T do
    for t' from t-M to t+M do
      for a from 1 to A do
        H[t',a]+=theta[a,t'-t,c[t]]
    pred[t-M] = arg max over a of H[t-M,a]
  return pred
```

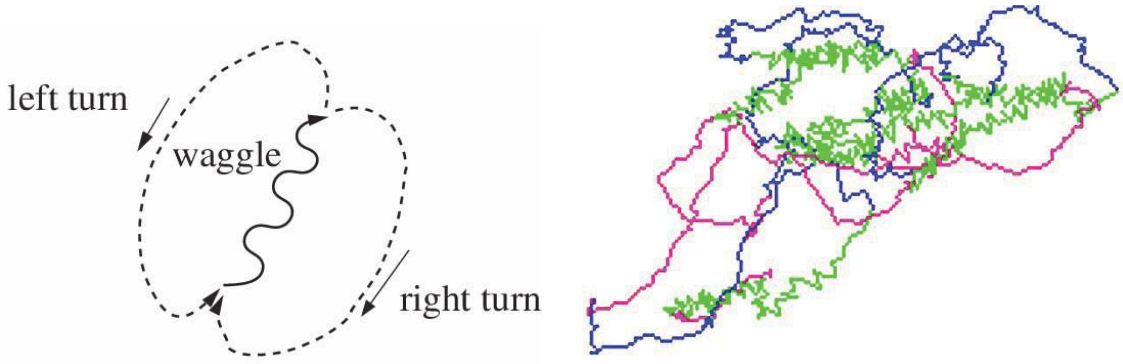
As illustrated by this pseudo code, the testing time complexity of the *Hough* pipeline is $O\left(T \times A \times M\right)$ and decision in time t' is taken with bounded latency M. For *OP*-like algorithm testing time complexity is $O\left(T \times A \times M^2\right)$ and for *degraded-LDS* it is $O\left(T \times A^2 \times M\right)$ (it is $O\left(T \times A^2\right)$ for standard *LDS* but with unbounded latency). So as discussed in table 1, the advantage of the *Hough* pipeline is the straightforward ability to deal with streams involving long action and large set of actions.

However, the selection of θ function is crucial: in the next sub-section, we evaluate the different *Hough* methods i.e. the different ways to select θ (*ISM*, *HHMT*, *ISM+SVM*, *DOHT*) on public datasets. As *ISK* is only intended for detection and not for segmentation, it is not compared to the others methods.

4.2. Evaluation on honeybee dataset

honeybee:

Experiments for evaluating *DOHT* have been conducted on the *honeybee* dataset [28]. The latter dataset provides tracking output of honey bees having 3 kinds of behaviours correlated with their trajectory (figure 4). It is composed of 6 large sequences. This dataset is well designed for action segmentation as each frame is associated with an action. To provide results comparable to [28], algorithms perform a leave-one-out cross validation setting (*LOOCV*): algorithms learn models using all videos from the dataset except one which is used



(a) theoretic behaviour (b) tracking output

Green correspond to waggle, magenta to right turn and blue to left turn.

Figure 4: *honeybee* dataset [28].

for actions segmentation (one decision at each frame). Performance for this setting is measured by the ratio of correctly decided actions in the testing video. Then, the final performance is measured by the mean of these ratios for all setting i.e. when each sequence is tested alternately. As explained in section 3.3, to provide segmentation results with *ISM*, *HHMT*, *ISM+SVM*, a probabilistic based smoothing is applied to the Hough score $\mathcal{H}$. *DOHT* provides directly segmentation results through equations (1) and (2).

Features and codewords:

The raw input signals in this dataset are the sequences of bee 2D positions and orientations (x_t, y_t, ϕ_t). As the goal of this set of experiments is to evaluate the voting process optimization, we do not focus on features, and thus, we just consider similar features from [15] which achieves the best published performances on this dataset. We use normalized short temporal series of 2D positions as features. Let us call $R(\mu)$ the matrix of the 2D rotation of angle μ and $p(t) = (x_t, y_t)$, we consider the vector $(R(-\phi_t)(p_{t-\tau} - p_t), ..., R(-\phi_t)(p_{t+\tau} - p_t))$ as the feature extracted at time t. Multiple τ are considered simultaneously. Same features are considered for all *HT*. All these features are clustered (independently for each τ) using standard K means algorithm. The cluster centres define the codebook and features are mapped to their nearest codeword. In order to provide deterministic results (random free), K means is initialised with deterministic seeds: the two first seeds are a diameter of the point clouds, then next seed is the point with highest distance to the current set of seeds until K seeds have been extracted.

Results:

The meta parameters of the algorithms before the voting process are the sizes τ of the extracted short series, the K of the K means algorithms. As we focus on the voting process, we simply make those meta parameters varying on a coarse empirical grid. The only meta parameter of the voting process is C (eq. 4) which is tuned on the $2^{-5}, ..., 2^{10}$ grid from [5] (independently for

Method	Accuracy mean on *honeybee*
ISM from [21]	71.9
MMHT from [25]	78.8
ISM + SVM from [46]	77.5
DOHT (our)	**86.5**
Approximated DOHT (our)	85.1
Inconsistent DOHT (our)	79.4
HDP-HMM [45]	83.3
SLDS [28]	85.9
PS-SLDS [28]	**87.7**
naive OP [15]	84.5
OP [15]	**89.3**

Table 3: Results on *honeybee* [28] for a common typical setting. *ISM*, *MMHT*, *ISM+SVM* are reimplemented. Other results are just extracted from the corresponding papers.

each algorithm). We want to stress that all these meta parameters only concern input data or training step: there is no testing meta parameter in the voting process.

The main result on this dataset is that the maximal performances achieved by *ISM, MMHT* and *ISM+SVM* over all meta parameters are lower than the means performances of *DOHT*. Results for a typical run corresponding to $\tau \in \{1, 3, 6\}$, $K = 10$, and $C = 1$ are presented in table 3.

For these meta parameters, *DOHT* achieves 86.5% mean accuracy and significantly outperforms *ISM*, *MMHT* and *ISM+SVM* (table 3). In addition, *DOHT* results are just a little behind state-of-the-art results published in [28, 15]: 86.5% against 89.3% and 87.7%. However, these last results are obtained with algorithms which would lead to overweening computation testing times on larger datasets as discussed in section 2.2 and table 1. This non-linear-testing-time would be especially problematic for *PS-SLDS* which is close to *CRF* with latent variables that cluster frames into intervals, leading to a testing time complexity containing both quadratic factor from *CRF* and *OP*.

Discussion:

These results on *honeybee* dataset support the efficiency of the proposed *DOHT* toward state-of-the-art algorithms with important testing computation times and other *HT* methods (notice that the testing time of *DOHT* is the same as the *ISM* one).

Unfortunately, there is a practical limitation for using *DOHT*: the training formulation (eq. (4)) can hardly be performed on large datasets due to the required training time. Of course, training is an offline process that can be done one for all, but, even with a 64 processors 2024 Go RAM server, training is not trivial on large datasets. Thus, we offer a practical way to make the learning

step tractable.

4.3. Training speed-up

Identified issues:

Three ways have been identified to reduce the training time: reducing the number of constraints, handling specifically the positivity constraints ($\theta \geq 0$) and handling specifically the decreasing constraints (eq. (3)).

The number of constraints is reduced by sampling the frame t' considered to define the constraints.

The positivity constraints are approximated by transforming them into virtual data constraints: for all c, δ_t, we add to the real data constraints, a data constraint corresponding to a virtual action a_{false} present at $t + \delta_t$ in a stream containing only $\Upsilon \gg 1$ *negative* codewords c at time t. These virtual data constraints are processed as real ones leading to:

$$\forall a, c, \delta_t : (-\Upsilon \times \theta\left(a_{false}, c, \delta_t\right)) - (-\Upsilon \times \theta\left(a, c, \delta_t\right)) \geq 1 - \xi' \tag{5}$$

These virtual data constraints are satisfied at lower cost if $\theta\left(a_{false}, c, \delta_t\right) \approx 0$ and for all a, c, δ_t, $\theta\left(a, c, \delta_t\right) \geq \frac{1}{\Upsilon} \approx 0$, and thus distort optimization toward positive $\theta\left(\right)$ (for real actions).

To our knowledge, there is no standard way to deal with the decreasing constraints. A very simple solution could be to discard these constraints, as in this way, the problem is similar to a multiclass *SVM*. However, this approach, called the *Inconsistent DOHT*, leads to poor results as shown in table 3. Indeed, as the temporal consistency of votes is completely lost, this lead to a harmful noise over fitting. So, we introduce a trick to make implicit these constraints in a *SVM* formulation by expressing θ using characteristic functions.

Handling decreasing constraints:

Let us introduce the characteristic function: if I is a time interval then

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

Let M be the maximal length of action in the training data and let $\mathcal{J}^{full}$ denote the set of all time intervals containing 0 included in $[-M, M]$. Then, the satisfaction of equation (3) by $\theta\left(\right)$ is equivalent to the existence of a function $w\left(\right) \geq 0$ satisfying:

$$\theta\left(a, \delta_t, c\right) = \sum_{I \in \mathcal{J}^{full}} w\left(a, I, c\right) \times \chi_I\left(\delta_t\right)$$

especially because equation (3) is satisfy by each χ_I if $0 \in I$. Hence, the decreasing constraints can be omitted during the optimization step by looking for $w\left(\right)$ instead of $\theta\left(\right)$. However, as there are M^2 intervals in $\mathcal{J}^{full}$ (leading to a computational dead-end), *DOHT* is approximated (an called *Approximated DOHT*) using a subset $J \subset \mathcal{J}^{full}$ of time intervals.

Final training formulation:

Considering our experiments, we recommend this *Approximated DOHT* with 1-norm for L_{data} and with squared 2-norm for L_{reg}. Thus, our final training problem is a standard multi-class *SVM* [6, 16]:

$$\begin{array}{c} \min\limits_{w,\xi} \left(\|w\|_2^2 + C \times \|\xi\|_1 \right) \\ \text{under constraints:} \quad \forall \mathcal{V}, t', a \neq a^*_{\mathcal{V}}(t') : \\ \sum\limits_{(c,t)\in\mathcal{V}, I\in\mathcal{J}} \left(\left(w\left(a^*_{\mathcal{V}}(t'), I, c\right) - w\left(a, I, c\right) \right) \chi_I\left(t' - t\right) \right) + \xi(t') \geq 1 \end{array} \tag{6}$$

where $\mathcal{V}$ can represent a real data or come from eq. (5). To highlight that this problem is an *SVM*, let us map the couples (I, c) into integers through a function ϕ, then let us introduce the vector Q such that $Q_{\phi(I,c)}$ is the number of codewords c extracted from the video during the interval I translated by t', and let the vectors W_a be such that $W_{a,\phi(I,c)} = w(a, I, c)$. Then

$$\sum_{(c,t)\in\mathcal{V}, I\in\mathcal{J}} \left(\left(w\left(a^*_{\mathcal{V}}(t'), I, c\right) - w\left(a, I, c\right) \right) \chi_I\left(t' - t\right) \right) = \left\langle W_{a^*_{\mathcal{V}}(t')} - W_a, Q \right\rangle$$

where $\langle , \rangle$ is the standard scalar product. Alternatively, the 1-norm for L_{reg} et L_{data} leads to a *lp-SVM* [2].

We observe experimentally on *honeybee* dataset that a small fixed-size subset $\mathcal{J}$ is sufficient for *Approximated DOHT* to perform as well as the *DOHT*: using $\mathcal{J} = \left\{ \left[-2^{-\alpha} M, 2^{-\beta} M \right], 0 \leq \alpha, \beta \leq 6 \right\}$ and the *SVM* solver from [16], *Approximated DOHT* speeds up the training time by a factor 50 and achieves equivalent performances than *DOHT* ($85,1\%$ against $86,5\%$ see tab. 3) on *honeybee*.

This *Approximated DOHT* combined with a weak sub-sampling of constraints (called *DOHT* for simplicity later on in this paper) can be trained on large dataset particularly on motion capture stream to segment human activities.

5. Human Action Segmentation in Video Streams

5.1. Normalization of the skeleton signal

In this paper, we consider videos whose skeleton can be extracted to focus on skeleton based features. The raw signal from the video is thus a set of positions for each articulation along the time, in a coordinate system linked to the camera. In order to make the posture independent of the camera configuration, the signals have to be *normalized*. The main normalization processes of the literature are based on distances, angles or on subjective view.

Angle based normalization. The angles between two vectors are invariant to global rotations and translations. A representative angle normalization has been introduced in [31]. First, the principal component analysis (*PCA*) is applied on the 3D positions of neck, spine, tail and left/right shoulders and hips providing

two vectors u and r. Then, the vector connecting shoulders to elbows, hips to knees and neck to head are decomposed in the spherical system u,r i.e. the vector is describe by the two angles with u (inclination) and r (azimuth). Then, the vector connecting respectively elbows to hands, knees to feet are decomposed in a local spherical system u',r' where u' is the vector connecting respectively shoulders to elbows and hips to knees and r' is the projection of r into the orthogonal plane of u'. Finally all angles are unfolded into a 16D vector that describes the 48D skeleton (16 articulations, each corresponding to a 3D point).

Subjective view based normalization. Given in a standard coordinate system linked to a skeleton, the positions of articulations become invariant to global rotations and translations of the skeleton. For example, the positions of all articulations can be expressed in the coordinate system linked to the torso introduced in [31].

In [48],the coordinate system linked to the torso is computed and the sphere containing the person is decomposed into solid angle. Each solid angle corresponds to a histogram bin and each articulation votes for its containing bin, leading to a histogram that characterizes the skeleton.

Distances based normalization. The distances between two articulations are invariant to global rotations and translations. In [43], the distances for all pairs of articulations are considered, introducing some redundancy: 128 distances are computed for a 48D skeleton. Relevant distances are then selected by a learning process. In [49], a *Hough forest* selects relevant distances throw a sampling into distance based weak binary classifier and entropy based rejection.

Evaluation. In order to evaluate the influence of the normalization, we extract some common features and use them either without normalization or with one of the three different normalizations. The simplest features are considered in this set of experiments: the feature extracted in a frame is simply the full skeleton unfolded into a vector. Each skeleton vector is then transformed into a vector of angles from [31] or a vector of distances from [43] or expressed into the coordinate system from [31] (to evaluate respectively angle, distance or subjective view normalization).

In all cases, after normalization, the skeleton extracted at a specific frame is represented by a fixed size vector. These vectors are then quantified using K means algorithm. Evaluations are performed on *UTKAD* and *TUM* datasets with *DOHT*.

For *TUM*, [49] being the most relevant paper associated to *TUM* dataset in our context, we follow the same experimental protocol. For this purpose, the dataset is split into training and testing parts following a predefine scheme (see [49]). Each algorithm provides a predicted action associated with the left hand motion in each frame of the testing sequences. Performances are measured by the total number of correct predictions over the total number of frames in testing videos. If not mentioned, the input of our pipeline is the reduced skeleton with *13-joints*.

Normalization of skeleton coupled with *DOHT*	Accuracy mean on *UTKAD*	Accuracy on *TUM*
raw skeleton	58.3	53.4
angle skeleton	**72.0**	61.7
distance skeleton	69.2	60.2
subjective skeleton	70.7	**62.7**

Table 4: Results of different normalizations on the skeleton features on *HTKAD* and *TUM* datasets.

In *UTKAD*, there are sometimes multiple actions in a same frame. In such frame, we remove all actions except the dominant. Then, performances are measured through a *LOOCV* as for *honeybee*.

Results and discussion. The training meta parameters of this experiment are the K and C (as for *honeybee*). Maximal performances of these different algorithms are presented in table 4.

All these normalizations achieve roughly equivalent performances but outperform not normalized skeleton. Also, when *DOHT* is applied on both 3 normalized skeletons, it does not achieve better results than on each normalized skeletons separately, highlighting that these normalizations are not complementary.

On the other side, angle based normalization is less generic than the others since the transformation applied is different for each articulation (normalization is performed incrementally from center articulations to body extremities). Also, distance normalization is not invariant to the global body size of the different subjects contrary to subjective view and angle normalization. So, we decide to apply this subjective view normalization on the feature extracted from the stream. We focus on them in the next sub-section.

5.2. Selected features

In literature, both single frame skeleton features [48] and multiple frames skeleton features [43] are proposed as well as both single articulation [43] or combined [31, 48, 49] approaches. The feature type has a significant impact on e.g. the discriminative power and robustness to occlusions. Typically, a grouped feature may be more discriminative than the corresponding part features but may not be extracted as soon as one part is occluded. In addition, articulation combination leads to a higher feature space and may lead to adverse effects e.g. for dictionary construction using *K means* algorithm. Finally, when relying on training step, the learning algorithm may be more efficient by selecting relevant combination from individual features.

In order to compete single articulation versus skeleton features, we evaluate *DOHT* fed by one or the other.

input features for *DOHT*	Accuracy mean on *UTKAD*	Accuracy on *TUM*
skeleton	70.7	62.7
articulations	**74.8**	**77.6**

Table 5: Results of skeleton versus articulation features on *UTKAD* and *TUM* datasets.

Single articulation versus skeleton features:

The skeleton feature is a vector s where $s_{3\times i+j}$ is the coordinate j (in 3D) of the articulation i (one vector for each skeleton). These skeleton features are quantified using K means algorithm.

The single articulation feature corresponds to the 3D position of one articulation i i.e. the vector $(s_{3\times i}, s_{3\times i+1}, s_{3\times i+2})$. These vectors are then quantified independently for each articulation, using K means algorithm. A unique K value is considered for all articulations.

The training meta parameters of this experiment are again K and C. The maximal performances of these pipelines are presented in table 5.

These experiments support that articulation based features are better than skeleton ones for learning based algorithms. This trend is yet not crucial on *UTKAD* where actions are correlated with global body motion, but on *TUM* where action is strongly correlated with the motion of the left hand, the articulation based features strongly outperform skeleton ones with *DOHT*.

These results support independent *articulation* features toward combined ones and invite to compete independent *frame* features toward combined ones.

Single frame versus multiple frame features:

Multiple frame approaches have the ability to capture some temporal property of the action unlike single frame ones, and, have provided an important breakthrough [37, 42] in human action recognition. Particularly, in a *BOW+SVM* framework [37, 42, 43], actions like *stand up* or *sit down* can hardly be distinguished using single frame features as no element of the pipeline takes into account temporal information. So such framework requires multiple frame features like *Fourier temporal pyramid* [43] or interest point trajectories [37, 42]. But, single frame skeleton features are more robust to skeleton extraction failure, speed variation and to irregular sub-sampling of the frames of the video. This invites to evaluate if multiple frame features are useful in *HMM* context [48] or *DOHT* context where the temporal information is already captured by the learning processes.

To evaluate the relevancy of multiple frame features in our context, we introduce several multiple frame features:

Speeds. The first multiple frame features are simply the displacement vectors between two successive positions of an articulation i.e. if q_t is the 3D position of a specific articulation in the normalized coordinate system then $q_{t+1} - q_{t-1}$ is

input features	Accuracy
positions	77.6
speeds	77.1
tracklets	**80.1**
position + speeds	79.8
positions + *tracklets*	80.6
speeds + *tracklets*	80.8
positions + speeds + *tracklets*	**81.5**

Table 6: Results of different features with *DOHT* on *TUM* dataset.

the feature extracted for this articulation at frame t. As for position, K means algorithm transforms these features into codeword.

Sub-trajectories. The second multiple frame features considered are short temporal series of 3D positions of each articulation: let the vector $(q_1, ..., q_T)$ be the normalized trajectory of one articulation, then, we consider the vector $(q_{t-\tau}, ..., q_{t+\tau})$ as a feature extracted at time t with size τ. These features called *tracklets* are densely extracted i.e. extracted at each frame where available. The hidden target of these features is to capture the gestures performed by the subject. Similar features are also considered in [37, 42] emphasizing the efficiency of such interest point trajectories for human action recognition.

Several sizes τ may be simultaneously considered. In this case, *tracklets* with different sizes may be clustered together to make codeword extraction robust to local speed of the action. Such clustering can be performed with K medoids algorithm based on *dynamic time wrapping* (DTW). DTW [3] provides a classical distance between different sizes series by matching several times in one sequence to one time in the other and vice-versa. K medoids is a variant of K means where only distances are needed and not means which require a common space. Alternatively clustering can be done independently for each *tracklet* size τ with K means algorithm. However, considering our experiment, it appears that using different sizes unlikely provides complementary information, so for simplicity we consider one size only.

Results. Experiments to compete single frame versus multiple frame features are performed only on *TUM* dataset as an irregular sub sampling does not allow using multiple frame features on *UTKAD*. The meta parameters of the experiment are C, K and τ for *tracklets* (τ concerns only input data and C, K only the training). Due to the size of parameters grid, we can not consider different K for different features, so in an experiment e.g. *positions + tracklets* a common K is considered for all clustering processes even if positions are 3-D features and tracklets $(3 \times \tau)$-D ones. Maximal performances of pipelines are presented in table 6.

These experiments support that multiple frame features performs better than single frame ones in context of *DOHT* algorithm. However, these different

$\tau = 4$

$K:$	5	10	15
$C = 0.25$	70.5	69.5	67.9
$C = 0.5$	72.4	72.1	70.4
$C = 1$	74.4	74.2	73.8
$C = 2$	75.1	76.3	75.3
$C = 4$	74.8	76.5	76.3
$C = 8$	74.8	77.2	75.7

$\tau = 6$

$K:$	5	10	15
$C = 0.25$	70.2	70.0	70.4
$C = 0.5$	72.2	73.2	73.7
$C = 1$	75.4	75.8	76.3
$C = 2$	77	78.0	77.4
$C = 4$	77.3	79.4	78.3
$C = 8$	76.4	78.9	77.6

$\tau = 8$

$K:$	5	10	15
$C = 0.25$	70.2	71.7	70.5
$C = 0.5$	72.8	76.0	73.2
$C = 1$	74.9	77.9	76.0
$C = 2$	76.6	**80.1**	78.1
$C = 4$	77.1	79.2	78.6
$C = 8$	77.3	78.3	77.7

$\tau = 10$

$K:$	5	10	15
$C = 0.25$	71.4	72.3	70.8
$C = 0.5$	73.7	76.1	74.3
$C = 1$	76.1	78.2	76.9
$C = 2$	77.6	79.3	78.2
$C = 4$	76.7	79.4	78.8
$C = 8$	76.3	79.0	78.9

Table 7: Results for *tracklet* features when varying the training meta parameters K, C, τ with *DOHT* on *TUM* dataset.

features are complementary as *DOHT* fed with *positions + speeds + tracklets* perform better than *DOHT* fed by one of the three.

Discussion:
Best selected features. To summarise these experiments, it appears that in our context single articulation features are better than combined ones and multiple frame features better than single frame ones. The best evaluated features alone are *tracklets*: the concatenation on a small time window of the 3D positions of one specific articulation densely extracted across videos. However, the best evaluated pipeline is the combination of simple positions + speeds + *tracklets*.

Stability toward meta parameters. In these experiments, we focus on maximal performance achieved when coarsely varying meta parameters e.g. K, C, τ. The stability toward these meta parameters is yet important in a system perspective. Hence, in table 7, we present the different raw results. We want to stress that all these meta parameters only concern input data (τ) or training step (K, C) but that there are no meta parameter specific to testing voting process.

The interested point is that none of the meta parameters is crucial: even if empirical maximal performance is achieved only with one specific setting, there are plenty of settings which lead to nearly maximal performances. For example, there are 6 setting providing more than the maximal performance minus 1.

One other interesting point is that K is small. An explanation is that in our training formulation (eq. (6)), the number of variables is $K \times A \times J$ where A is the number of activities (typically 10) and J the number of intervals considered

algorithm	*TUM*	*TUM 27-joints*	*UTKAD*
speed features + *HF* [49]	80.3	80.3	
all features + *HF* [49]	77.6	81.5	
ISM	58.3	58.3	62.7
DOHT	**81.5**	**83.0**	**74.8**

Table 8: Comparison with published results on *HTKAD* and *TUM* datasets. Results are just extracted from the corresponding papers and do not come from reimplementation.

for the approximation (typically 64). Thus, increasing K leads to more variables and may lead to over-fitting.

5.3. Comparison to the state-of-the-art

In table 8, our results on human actions are presented together with published comparable ones and with the *ISM* baseline.

However, considering the few number of comparable results (works dealing with segmentation on *TUM* and *UTKAD* datasets), we also offer to compare our algorithm to published *close* results. For this purpose, we present performance of our pipeline in classification context i.e. on already segmented data. As our algorithm is not designed for such context, these experiments lead to under estimate the performances of our algorithm. However, to our knowledge, the only alternative could have been to reimplement algorithms e.g. [49, 15, 28] and apply it on *TUM* and *UTKAD* datasets. But, these works need expert knowledge to select properly hyper parameters, and thus, the reimplementation may lead to under estimate performances of the state of the art. In addition, in classification context, we evaluate another baseline obtained by erasing location information from *DOHT*. This can be done by erasing the keyword locations (e.g. replacing them by a constant). Thus, the performance differences between this baseline and *DOHT* can be interpreted as the information provided by taking into account temporal location of extracted keywords from the videos. Alternatively, this can be done in practice by setting $\mathcal{J} = \{[-M, M]\}$ where $\mathcal{J}$ is the set of intervals from eq. (6). So this baseline corresponds to a *BOW+SVM* without any normalization (e.g. no l_1 normalization, no *tfidf* normalization). Thus, we call this baseline *unscaled SVM*. Nearest neighbours algorithm is call *NN*. Videos from **both** training and testing sets are segmented according to the ground truth and then given as input to *DOHT* as it was unsegmented data.

These results are presented in table 9 (to our knowledge no other results on these datasets have been published). Our algorithm achieves 82.4% means accuracy on *UTKAD* toward 91.5% for the state-of-the-art. This is a promising result considering that our algorithm is designed for another context.

At first sight, another comparison could have be done on *CAD-60* [38], especially because [38] is an algorithm which deals with segmentation. However there are three problems. First, even if [38] presents a segmentation algorithm,

algorithm	*TUM*	*UTKAD*
HMM [48]		90.9
NN [11]		**91.5**
SVM [9]		90.4
sparse SVM [41]		90.9
ISM	65.5	64.1
unscaled SVM	80.6	79.8
DOHT	**88.5**	82.4

Table 9: Results on already segmented data on several datasets.
Results are just extracted from the corresponding papers and do not come from reimplementation.

CAD-60 data are already segmented. Then, results on this dataset are reaching a saturation point: even *ISM* achieves **90.3%** of precision and **86.0%** of recall in *new person* setting (see [38]). Finally but more problematic, algorithm from [38] is designed to decide as *neutral* the frames which are not interesting for any action but the ground truth (yet provided by [38]) does not take this into account. So, algorithm from [38] only achieves only 69.0% precision and 57.3% recall that is much lower than the baseline *ISM*.

Considering these comparisons, our pipeline improves state-of-the-art performances in human action segmentation context and achieves promising results on human action classification task. This supports the pertinence of our pipeline for human action segmentation.

6. Conclusion

In this paper, we offer two contributions. First, we offer a new *Hough transform* method based on discriminative parameters only. This *Deeply Optimized Hough Transform* is designed for segmentation by performing a voting process based on smoothed weights. It significantly outperforms generative based *Hough transform* on public datasets.

The second contribution of this paper is the application of this method to human action segmentation. Combined with the selected skeleton based features, this method slightly improves state-of-the-art performances on unsegmented data and achieves honorable performances on segmented ones on public datasets highlighting the relevancy of this new method for human action segmentation.

In future works, we will adapt and apply our method to deal with human action spatiotemporal segmentation in classic videos. This aim introduces two important challenges: performing segmentation in 3D (picture space + time) and managing a large number of features.

References

[1] Aggarwal, J., Ryoo, M. S., 2011. Human activity analysis: A review. ACM Computing Surveys.

[2] Bennett, K. P., Mangasarian, O. L., 1992. Robust linear programming discrimination of two linearly inseparable sets. Optimization methods and software.

[3] Berndt, D. J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series. In: KDD workshop.

[4] Chakrabartty, S., Cauwenberghs, G., 2007. Gini support vector machine: Quadratic entropy based robust multiclass probability regression. Journal of Machine Learning.

[5] Chang, C.-C., Lin, C.-J., 2011. Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST).

[6] Chapelle, O., Keerthi, S. S., 2010. Efficient algorithms for ranking with svms. Information Retrieval.

[7] Chaquet, J. M., Carmona, E. J., Fernndez-Caballero, A., 2013. A survey of video datasets for human action and activity recognition. Computer Vision and Image Understanding.

[8] Cortes, C., Vapnik, V., 1995. Support-vector networks. Journal of Machine learning.

[9] Cottone, P., Re, G. L., Maida, G., Morana, M., 2013. Motion sensors for activity recognition in an ambient-intelligence scenario. In: Smart Environments and Ambient Intelligence Workshop.

[10] Crammer, K., Singer, Y., 2002. On the algorithmic implementation of multiclass kernel-based vector machines. The Journal of Machine Learning Research.

[11] Devanne, M., Wannous, H., Berretti, S., Pala, P., Daoudi, M., Del Bimbo, A., 2013. Space-time pose representation for 3d human action recognition. In: International Conference on Image Analysis and Processing Workshop.

[12] Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A., 2011. Efficient regression of general-activity human poses from depth images. In: International Conference on Computer Vision.

[13] Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R., 2007. Actions as space-time shapes. Transactions on Pattern Analysis and Machine Intelligence.

[14] Han, J., Shao, L., Xu, D., Shotton, J., 2013. Enhanced computer vision with microsoft kinect sensor: A review. Transactions on Cybernetics.

[15] Hoai, M., Lan, Z., De la Torre, F., 2011. Joint segmentation and classification of human actions in video. In: International Conference on Computer Vision and Pattern Recognition.

[16] Joachims, T., Finley, T., Yu, C.-N. J., 2009. Cutting-plane training of structural svms. Machine Learning.

[17] Karmarkar, N., 1984. A new polynomial-time algorithm for linear programming. In: symposium on Theory of computing.

[18] Lafferty, J., McCallum, A., Pereira, F. C., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: International Conference on Machine Learning.

[19] Lai, K., Konrad, J., Ishwar, P., 2012. A gesture-driven computer interface using kinect. In: Symposium Image Analysis and Interpretation.

[20] Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B., 2008. Learning realistic human actions from movies. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[21] Leibe, B., Leonardis, A., Schiele, B., 2004. Combined object categorization and segmentation with an implicit shape model. In: Europeen Conference on Computer Vision - workshop.

[22] Lin, S.-Y., Shie, C.-K., Chen, S.-C., Lee, M.-S., Hung, Y.-P., 2012. Human action recognition using action trait code. In: Internationnal Conference on Pattern Recognition.

[23] Liu, J., Luo, J., Shah, M., 2009. Recognizing realistic actions from videos in the wild. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[24] Lv, F., Nevatia, R., Lee, M., 2005. 3d human action recognition using spatio-temporal motion templates. Computer Vision in Human-Computer Interaction.

[25] Maji, S., Malik, J., 2009. Object detection using a max-margin hough transform. In: International Conference on Computer Vision and Pattern Recognition.

[26] Morency, L.-P., Quattoni, A., Darrell, T., 2007. Latent-dynamic discriminative models for continuous gesture recognition. In: CInternationnal Conference on Computer Vision and Pattern Recognition.

[27] Ni, B., Wang, G., Moulin, P., 2011. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In: International Conference on Computer Vision Workshop.

[28] Oh, S., Rehg, J., Balch, T., Dellaert, F., 2008. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. International Journal of Computer Vision.

[29] Popa, M., Koc, A. K., Rothkrantz, L. J., Shan, C., Wiggers, P., 2012. Kinect sensing of shopping related actions. In: Constructing Ambient Intelligence.

[30] Poppe, R., 2010. A survey on vision-based human action recognition. Image and vision computing.

[31] Raptis, M., Kirovski, D., Hoppe, H., 2011. Real-time classification of dance gestures from skeleton animation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation.

[32] Roweis, S., Ghahramani, Z., 1999. A unifying review of linear gaussian models. Journal of Neural computation.

[33] Schuldt, C., Laptev, I., Caputo, B., 2004. Recognizing human actions: a local svm approach. In: Internationnal Conference on Pattern Recognition.

[34] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., Moore, R., 2011. Real-time human pose recognition in parts from single depth images. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[35] Sivic, J., Zisserman, A., 2003. Video google: A text retrieval approach to object matching in videos. In: International Conference on Computer Vision.

[36] Song, Y., Morency, L.-P., Davis, R., 2013. Action recognition by hierarchical sequence summarization. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[37] Sun, J., Wu, X., Yan, S., Cheong, L., Chua, T., Li, J., 2009. Hierarchical spatio-temporal context modeling for action recognition. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[38] Sung, J., Ponce, C., Selman, B., Saxena, A., 2011. Human activity detection from rgbd images. In: AAAI Workshop.

[39] Tenorth, M., Bandouch, J., Beetz, M., 2009. The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In: International Conference on Computer Vision Workshops.

[40] Thanh, T. T., Chen, F., Kotani, K., Le, H.-B., 2012. Extraction of discriminative patterns from skeleton sequences for human action recognition. In: International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future.

[41] Theodorakopoulos, I., Kastaniotis, D., Economou, G., Fotopoulos, S., 2013. Pose-based human action recognition via sparse representation in dissimilarity space. Journal of Visual Communication and Image Representation.

[42] Wang, H., Kläser, A., Schmid, C., Cheng-Lin, L., 2011. Action recognition by dense trajectories. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[43] Wang, J., Liu, Z., Wu, Y., Yuan, J., 2012. Mining actionlet ensemble for action recognition with depth cameras. In: International Conference on Computer Vision and Pattern Recognition.

[44] Wang, Y., Mori, G., 2009. Max-margin hidden conditional random fields for human action recognition. In: Internationnal Conference on Computer Vision and Pattern Recognition.

[45] Willsky, A. S., Sudderth, E. B., Jordan, M. I., Fox, E. B., 2008. Nonparametric bayesian learning of switching linear dynamical systems. In: Advances in Neural Information Processing Systems.

[46] Wohlhart, P., Schulter, S., Kostinger, M., Roth, P., Bischof, H., 2012. Discriminative hough forests for object detection. In: British Machine Vision Conference.

[47] Wouter, Josemansand Gwenn, E., Ben, K., 2013. Fusion of color and depth camera data for robust fall detection. In: International Conference on Computer Vision Theory and Application.

[48] Xia, L., Chen, C.-C., Aggarwal, J., 2012. View invariant human action recognition using histograms of 3d joints. In: International Conference on Computer Vision and Pattern Recognition Workshops.

[49] Yao, A., Gall, J., Fanelli, G., Van Gool, L., 2011. Does human action recognition benefit from pose estimation? In: British Machine Vision Conference.

[50] Zhang, Y., Chen, T., 2010. Implicit shape kernel for discriminative learning of the hough transform detector. In: British Machine Vision Conference.