HAL
open science

Deeply optimized Hough transform: Application to action segmentation

A. Chan-Hon-Tong, C. Achard, L. Lucat

To cite this version:

A. Chan-Hon-Tong, C. Achard, L. Lucat. Deeply optimized Hough transform: Application to action segmentation. 17th International Conference on Image Analysis and Processing, ICIAP 2013, Sep 2013, Naples, Italy. pp.51-60, 10.1007/978-3-642-41181-6_6 . cea-01813733

HAL Id: cea-01813733

https://cea.hal.science/cea-01813733v1

Submitted on 18 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Deeply Optimized Hough Transform: Application to Action Segmentation

Adrien Chan-Hon-Tong[1], Catherine Achard[2], and Laurent Lucat[1]

[1] CEA, LIST, DIASI, Laboratoire Vision et Ingénierie des Contenus, France
{adrien.chan-hon-tong,laurent.lucat}@cea.fr
[2] Institut des Systèmes Intelligents et Robotique, UPMC, France
catherine.achard@upmc.fr

Abstract. Hough-like methods like *Implicit Shape Model (ISM)* and Hough forest have been successfully applied in multiple computer vision fields like object detection, tracking, skeleton extraction or human action detection. However, these methods are known to generate false positives. To handle this issue, several works like *Max-Margin Hough Transform* (*MMHT*) or *Implicit Shape Kernel* (*ISK*) have reported significant performance improvements by adding discriminative parameters to the generative ones introduced by *ISM*. In this paper, we offer to use only discriminative parameters that are globally optimized according to all the variables of the Hough transform. To this end, we abstract the common vote process of all Hough methods into linear equations, leading to a training formulation that can be solved using linear programming solvers. Our new Hough Transform significantly outperforms the previous ones on *HoneyBee* and *TUM* datasets, two public databases of action and behaviour segmentation.

Keywords: Hough Transform, Learning, Action Segmentation.

1 Introduction

The Hough Transform [9] has first been introduced to detect lines in picture. The main idea of this method is to perform the detection not directly in the picture space but in the line parameter space (Hough space) where each line in the image is mapped into a single point. This method has subsequently been extended to parametric objects [1], and non-parametric objects [11] (e.g. car, pedestrian, sport activities, ...). For non parametric objects, the Hough Transform first learns a probabilistic-like parametrization of the objects on a training database, and, then performs the detections as a local problem in the corresponding Hough space.

Due to this property of local detection, Hough Transform is a very fast process both in theory (time complexity theory) and practice. For this reason, it has been applied in context of real-time systems like [7] for skeleton extraction and more generally in multiple computer vision fields like tracking [6], object detection [5], human action detection [19]. Actually, this method can be used to perform

multi-class segmentation in multi-dimensional data e.g. pixel segmentation in image [12], temporal human action segmentation [20].

In this paper, we focus on temporal action segmentation in video: for each frame, the goal is to determine automatically which action is performed by an actor among a set of actions. The Hough Transform, used in this context, is composed of three main steps:

1: feature extraction and quantization to form codewords.
2: each of these codewords ω extracted at frame t votes for the action l at frame $t+\Delta_t$ with a specific weight $\theta(\omega, l, \Delta_t)$. During this step, the same codeword ω votes for all actions l and all relative time displacements Δ_t. The weights have been learned previously, during a learning step.
3: All votes, for all codewords and all frames, are agglomerated to build the Hough score which is the basis for segmentation decisions.

More formally, the Hough Transform (step 2-3) is based on the function $\theta()$ (traditionally positive) that links codewords, time displacements and actions to vote weights. Thus, a codeword w extracted at time t votes with a weight $\theta(w, l, \Delta_t)$ for the hypothesis that an action l is present at time $t + \Delta_t$ (this weight does not depend on the time t but only on l, Δ_t and w). Hence, given a set of localized codewords $W = \{w, t\}$ (w for codewords, t for the extraction time), the Hough score $\mathcal{H}$ for the action l at the time $\overline{t}$ is:

$$\mathcal{H}\left(\overline{t}, l\right) = \sum_{(w,t)\in W} \theta\left(w, l, \overline{t} - t\right) \tag{1}$$

and, the decision about the action at time $\overline{t}$ is given by:

$$\widehat{l}\left(\overline{t}\right) = \arg\max_{l} \left(\mathcal{H}\left(\overline{t}, l\right)\right) \tag{2}$$

Thus an action is decided for each frame of the video.

Hence, all the purpose of the training is to select values for $\theta(w, l, \Delta_t)$ that will provide correct decisions with respect to the equations (1) (2) at testing time. Several works, recalled in section 2, offer to improve the generative votes introduced by the *ISM* method by introducing a partial discriminative optimization process during the vote estimation step. In section 3, we offer to extend these methods by optimizing globally all the votes in a discriminative way. With this new learning process, our Hough method significantly outperforms previous ones on two public datasets of action segmentation (the *Honeybee* dataset [14] and the *TUM* dataset [16]) as reported in section 4, before the conclusion in section 5.

2 State of the Art

In this section, we present the different published methods to select the vote weight during the training step of Hough Transform.

2.1 Implicit Shape Model

In the *ISM* [11], the Hough Transform (the values of $\theta()$) is based on generative weights. Let $\mathcal{P}(l, \Delta_t|w)$ be the probability that the action at time $t + \Delta_t$ is l, knowing that a codeword w has been extracted at time t. This probability is estimated with statistics on the training dataset and is supposed to be independent of t (it just depends on l, Δ_t and w). Then, the weights are given by:

$$\theta_{ISM}(w, l, \Delta_t) = \mathcal{P}(l, \Delta_t|w) \tag{3}$$

In practice, the probability $\mathcal{P}(l, \Delta_t|w)$ is estimated by:

$$\mathcal{P}(l, \Delta_t|w) \approx \frac{N(l, \Delta_t, w)}{N(w)} \tag{4}$$

where $N(l, \Delta_t, w)$ is the number of occurrences of an action l observed with a displacement Δ_t from a codeword w and $N(w)$ is the number of occurrences of the codeword w.

These *ISM*-based weights have several advantages (e.g. parameter-free training, robustness to over-training), but they suffer from several drawbacks. In particular, all codewords and training examples have the same importance and are considered independently from each other. Two methods, *MMHT* [13] and *ISK* [21] have been introduced to address these drawbacks.

2.2 Max-Margin Hough Transform

In *MMHT* [13], a coefficient is introduced for each codeword to ponderate the *ISM* values, resulting in:

$$\theta_{MMHT}(w, l, \Delta_t) = \lambda_w \times \theta_{ISM}(w, l, \Delta_t) = \lambda_w \times \mathcal{P}(l, \Delta_t|w) \tag{5}$$

The weights λ_w give more or less importance to the different codewords w according to their discriminative power. They are learnt simultaneously in a discriminative way through an optimisation process similar to *support vector machine* (SVM) training [4].

2.3 Implicit Shape Kernel

In *ISK* [21], 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:

$$\theta_{ISK}(w, l, \Delta_t) = \sum_i \lambda_i \times \mathcal{P}_i(l, \Delta_t|w) \tag{6}$$

where $\mathcal{P}_i(l, \Delta_t|w)$ is an estimation of the probability $\mathcal{P}(l, \Delta_t|w)$ based only on the training example i. The weights λ_i are learnt simultaneously in a discriminative way using a specific *kernel-SVM* training [21].

MMHT and *ISK* report experimental improvements over *ISM* by adding discriminative parameters. This trend is also supported by [18].

2.4 *ISM + SVM*

In [18], it is highlighted that learning directly the Hough map $\mathcal{H}\left(\overline{t}, l\right)$ with a *SVM* is equivalent to the *ISM* with the introduction of a weighting coefficient for each displacement. This results in:

$$\theta_{ISM+SVM}\left(w, l, \Delta_t\right) = \lambda_{\Delta_t} \times \mathcal{P}\left(l, \Delta_t | w\right) = \lambda_{\Delta_t} \times \theta_{ISM}\left(w, l, \Delta_t\right) \qquad (7)$$

2.5 Hough Forest

To our knowledge *ISM* [11] and the presented extensions [13,18,21] 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 [5] methods which are major methods of the state of the art. Hough forests use *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. Several works, like [5,20], report that this automatic feature mapping process associated with *ISM* votes leads to significant experimental improvements against codewords obtained without learning, by K-means algorithm for example.

However, in this paper, we focus on the optimisation of the weights used during the vote process and so to the link between codewords and votes which is generic whatever the features and codewords used. Thus, the offered method can be employed in the Hough forest context by substituting the weights estimated by *ISM* by the weights optimized by our offered method.

The common point between *MMHT*, *ISK* and *ISM+SVM* is that they add discriminative parameters to the generative ones introduced by the *ISM*. In this paper, we offer to use only discriminative votes strongly optimized. We call this method Deeply Optimized Hough Transform (*DOHT*).

3 Deeply Optimized Hough Transform

The goal of the training process is to establish a mapping between codewords and weights. While *ISM* method only uses generative weights, *MMHT*, *ISK* and *ISM+SVM* introduce discriminative parameters optimized according to codewords, training examples or displacements. We offer in this paper to optimize all these weights in a global way, according to all parameters of $\theta\left(w, l, \Delta_t\right)$ in multi-class context. Hence, our set of variables is indexed by codewords w (as in *HHMT*), displacements Δ_t (as in *ISM+SVM*) and also by actions l. These differences are summarized in table 1. In this way, we do not use *ISM* values and the method becomes deeply discriminative.

The goal is to define a function $\theta()$ such that for all training examples W and all times $\overline{t}$, the predicted action $\widehat{l}$ is the right one l^* (known on training data).

Considering the definition of the predicted action $\widehat{l}$ (eq. (2)), our problem formulation is equivalent to: $\forall W, \overline{t}, l \neq l^* \left(\overline{t}\right)$

$$\mathcal{H}\left(\overline{t}, l\right) < \mathcal{H}\left(\overline{t}, l^* \left(\overline{t}\right)\right) \tag{8}$$

By dividing $\theta()$ by the minimal gap in eq. (8), this is equivalent to

$$\mathcal{H}\left(\overline{t}, l\right) + 1 \leq \mathcal{H}\left(\overline{t}, l^* \left(\overline{t}\right)\right) \tag{9}$$

and, using equation (1), to $\forall W, \overline{t}, l \neq l^* \left(\overline{t}\right)$

$$\left(\sum_{(w,t)\in W} \theta\left(w, l, \overline{t} - t\right)\right) + 1 \leq \left(\sum_{(w,t)\in W} \theta\left(w, l^* \left(\overline{t}\right), \overline{t} - t\right)\right) \tag{10}$$

In addition, as a codeword extracted at time t should not provide more information about the time $t \pm 1$ than about the time t, we constraint θ to be decreasing with the absolute value of Δ_t.

However, it is not sure that a function can satisfy these constraints. Hence, we introduce a soft margin as in [4]: some variables ξ are introduced in eq. (10) leading to: $\forall \overline{t}, l \neq l^* \left(\overline{t}\right), W$

$$\sum_{(w,t)\in W} \theta\left(w, l, \overline{t} - t\right) + 1 - \xi\left(\overline{t}\right) \leq \sum_{(w,t)\in W} \theta\left(w, l^* \left(\overline{t}\right), \overline{t} - t\right) \tag{11}$$

and these variables are minimized to reduce the number of not-satisfied constraints, leading to the objective function: $\min_{\theta\geq 0, \xi\geq 0} \left(\sum_{\overline{t}} \xi\left(\overline{t}\right)\right)$.

To prevent over-fitting, a regularity term in added to the objective function as in [2]. A coefficient Υ regulates the trade-off between the attachment to data and the regularity as in [4,2].

Finally, the training problem is formulated as:

$$\begin{gathered} \min_{\theta\geq 0, \xi\geq 0} \left(\sum_{(w,l,\Delta_t)} \theta(w, l, \Delta_t) + \Upsilon \sum_{\overline{t}} \xi\left(\overline{t}\right)\right) \\ \text{under constraints: } \forall W, \overline{t}, l \neq l^* \left(\overline{t}\right), \\ \sum_{(w,t)\in W} \left(\theta\left(w, l^* \left(\overline{t}\right), \overline{t} - t\right) - \theta\left(w, l, \overline{t} - t\right)\right) + \xi\left(\overline{t}\right) \geq 1 \\ \text{and: } \forall w, l, \Delta_t, \theta\left(w, a, \Delta_t + \operatorname{sign}\left(\Delta_t\right)\right) \leq \theta\left(w, a, \Delta_t\right) \end{gathered} \tag{12}$$

where sign is the sign function.

These formulation is a linear program which which is a well studied problem in literature (e.g. [10]), and, which can be solved efficiency (for example using the solver CPLEX[1], freely available for academic purposes).

In the next section, we evaluate *ISM, HHMT, ISM+SVM* and *DOHT* in action segmentation or behavior segmentation contexts. As *ISK* is only intended

[1] `www-01.ibm.com/software/websphere/products/optimization/academic-initiative/`

Table 1. The different learning methods of the Hough Transform

methods	θ	variables
ISM [11]	$\theta_{ISM}(w,l,\Delta_t) = \mathcal{P}(l,\Delta_t\|w)$	-
MMHT [13]	$\theta_{MMHT}(w,l,\Delta_t) = \lambda_w \times \mathcal{P}(l,\Delta_t\|w)$	λ_w
ISK [21]	$\theta_{ISK}(w,l,\Delta_t) = \sum_i (\lambda_i \times \mathcal{P}_i(l,\Delta_t\|w))$	λ_i
ISM+SVM [18]	$\theta_{ISM+SVM}(w,l,\Delta_t) = \lambda_{\Delta_t} \times \mathcal{P}(l,\Delta_t\|w)$	λ_{Δ_t}
DOHT (our)	$\theta_{DOHT}(w,l,\Delta_t) = \lambda_{w,l,\Delta_t}$	λ_{w,l,Δ_t}

$P(l,\Delta_t|w)$ is the probability that the action at time $t+\Delta_t$ is l knowing that a codeword w has been extracted at time t. $P_i(l,\Delta_t|w)$ is the same probability estimated using only the training example i.

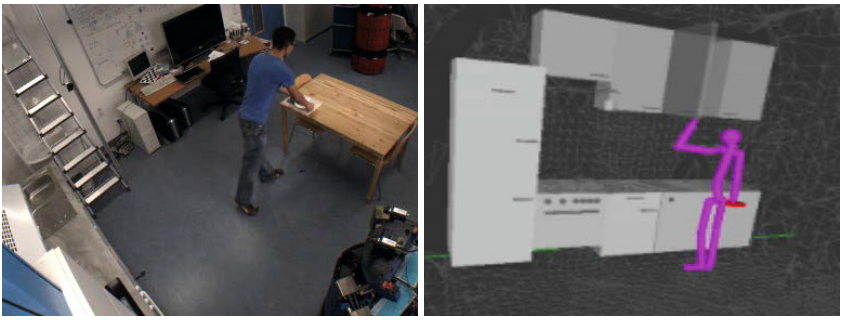
for detection and can not be straightforwardly extended to segmentation, we can not compare it to the others methods.

4 Experimental Results

Experiments have been conducted on the *TUM* [16] and *Honeybee* [14] datasets. These datasets are well designed for segmentation as each frame is associated to an action.

4.1 Application to Human Action Segmentation

TUM is a multi-sensor dataset and in particular it contains skeleton streams (fig. 1). It is composed of 19 sequences about 2 minutes each, containing 9 kinds of actions like *Lowering an object*, *Opening a drawer*, performed by 5 subjects. To provide results comparable to [20], the dataset is separated in training and testing sequences, all algorithms decide an action for each frame of each testing sequence and the performance is measured by the ratio of correctly decided actions.



example of action: *Lowering an object* Provided skeleton

Fig. 1. TUM dataset [16]

As [20] reports better performances using skeleton features (than visual or visual plus skeleton ones), we decide to consider only skeleton based features. Hence, the input signal of our algorithm is the 3D positions of each articulation at each time.

We use the same preprocessing (features and codewords) than the bag-of-gestures from [3] which achieves the best published performance on this dataset (with a manual segmentation). First, the positions are normalized (positions are expressed in a system of coordinates linked to the subject in order to be invariant to camera point of view, global body position, rotation and size). Then, we consider short temporal series of 3D positions of each articulation as features: let the vector $(p_1, ..., p_T)$ be the normalized trajectory of one articulation, then, we consider the vector $(p_{t-\tau}, ..., p_{t+\tau})$ as a feature extracted at time t. Similar features are also considered in [15,20,17] which emphasize the efficiency of interest points trajectories for human action recognition. Finally, all these features are clustered by K-means. The cluster centers define the codebook and features are mapped to their nearest codeword. We consider the 8 main articulations: feet, hands, knees, elbows. The quantization with K-means is performed independently for each articulation.

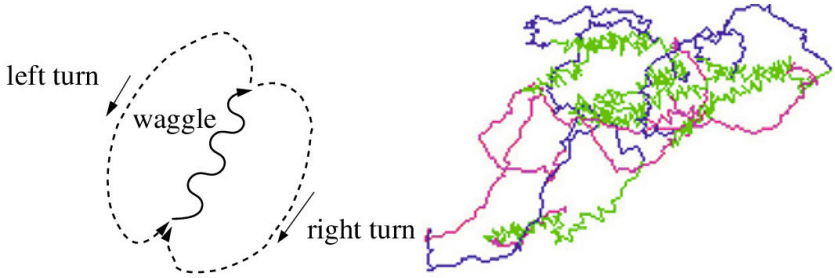
The parameters of this set of experiments are τ, K, Υ (for the learning process). On this dataset, the maximal performances achieved by *ISM, MMHT* and *ISM+SVM* when empirically varying the parameters values are less than the mean performances of *DOHT*. A typical run is obtained with $\tau = 6$, $K = 10$, and $C = 1$. Results of this experiment are presented in table 2.

In this set of experiments, *DOHT* significantly outperforms *ISM, MMHT* and *ISM+SVM* and achieves equivalent performance than a *SVM* based on the same features and codeword applied on the optimal segmentation (obtained from the ground truth) from [3]. Hence, for this dataset, we achieve equivalent performance than the best published (82.6% against 84.3%) without using any manual segmentation.

4.2 Application to Behaviour Segmentation

Experiments have also been conducted on the *Honeybee* dataset [14]. The Honeybee dataset provides tracking output of honey bees having 3 kinds of behaviour correlated with their trajectories (figure 2). It composed of 6 large sequences. To provide results comparable to [14], all algorithms decided an action for each frame in a leave-one-out cross validation setting and the performance is measured by the ratio of correctly decided actions.

The input signals in this dataset are the sequences of bee 2D positions and orientations (x_t, y_t, α_t). As in the previous experiment, normalized short temporal series of (2D here) positions are considered as features. Let us call $R(\beta)$ the matrix of the 2D rotation of angle β and $p(t) = (x_t, y_t)$, then we consider the vector $(R(-\alpha_t)(p_{t-\tau} - p_t), ..., R(-\alpha_t)(p_{t+\tau} - p_t))$ as the feature extracted at time t. All these features are clustered using K-means. The cluster centers define the codebook and features are mapped to their nearest codeword. K-means is performed independently for each τ.



(a) theoretic behaviour (b) tracking output

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

Fig. 2. Honeybee dataset [14]

The parameters of this set of experiments are τ, K, Υ. On this dataset, the maximal performances achieved by *ISM, MMHT* and *ISM+SVM* when empirically varying the parameters values are less than the mean performances of *DOHT*. A typical run is obtained with $\tau \in \{1, 3, 6\}$, $K = 10$, and $C = 1$. Results of this experiment are presented in table 2.

In this set of experiment, *DOHT* significantly outperforms *ISM, MMHT* and *ISM+SVM*. In addition, *DOHT* achieves equivalent performances than the latter best published results [8]. In [8], a multi-class *SVM* is applied on each temporal window (with similar kind of features and codewords). Then, segmentation is computed using dynamic programming. As scores are computed on each temporal window, this method is **quadratic** with respect to the maximal length of an activity while our is **linear**. This quadratic property is a common drawback caused by performing scoring as a global problem. Hence, for this dataset, we achieve equivalent performances (86.5% against 89.3%) than the best published results while being significantly faster.

Table 2. Global results on *TUM* [16] and *Honeybee* [14]

Method	Accuracy on *TUM*	Accuracy mean on *Honeybee*
ISM [11]	58.4	71.9
MMHT [13]	69.6	78.8
ISM+SVM [18]	68.5	77.5
DOHT (our)	**82.6**	**86.5**

5 Conclusion

In this paper, we offer to use Hough transform to segment temporal series. In a non parametric context, the training of Hough transform consists in properly selecting the weights used in the voting process. The simple way (Implicit Shape Model) consists in computing some probabilities on the training database,

leading to a generative model. Some methods (Max-Margin Hough Transform, Implicit Shape Kernel) offer to add some parameters optimized on a training database in a discriminative way. In this article, we offer to skip the first step based on a generative model, and, to globally learn all the parameters of the Hough transform on the training database, resulting a deeply discriminative model. This required to reformulate the voting process to express it in a linear form in order to use linear programming solvers.

We performed several experiments on public datasets where the Hough transform trained with our method significantly outperforms other Hough transform methods and provides equivalent results than best published results for these datasets while avoiding some limitations of the corresponding algorithms (e.g. manual segmentation).

In future works, we will adapt and apply our method on other contexts e.g. object segmentation in image, video spatio-temporal segmentation, automatic speech segmentation, sign language segmentation.

References

1. Ballard, D.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition (1981)
2. Chakrabartty, S., Cauwenberghs, G.: Gini support vector machine: Quadratic entropy based robust multi-class probability regression. Journal of Machine Learning (2007)
3. Chan-Hon-Tong, A., Ballas, N., Achard, C., Delezoide, B., Lucat, L., Sayd, P., Coise Prêteux, F.: Skeleton point trajectories for human daily activity recognition. In: International Conference on Computer Vision Theory and Application (2013)
4. Cortes, C., Vapnik, V.: Support-vector networks. Journal of Machine Learning (1995)
5. Gall, J., Lempitsky, V.: Class-specific Hough forests for object detection. In: Internationnal Conference on Computer Vision and Pattern Recognition (2009)
6. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. Pattern Analysis and Machine Intelligence (2011)
7. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: Internationnal Conference on Computer Vision and Pattern Recognition (2011)
8. Hoai, M., Lan, Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: Internationnal Conference on Computer Vision and Pattern Recognition (2011)
9. Hough, P.V.: Method and means for recognizing complex patterns. Tech. rep. (1962)
10. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: Theory of Computing (1984)
11. 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)
12. Sreedevi, M., Jeno Paul, P.: An efficient image segmentation using Hough transformation. Asian Journal of Information Technology (2011)

13. Maji, S., Malik, J.: Object detection using a max-margin Hough transform. In: Internationnal Conference on Computer Vision and Pattern Recognition (2009)
14. Oh, S., Rehg, J., Balch, T., Dellaert, F.: Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. Journal of Computer Vision (2008)
15. 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)
16. 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)
17. Wang, H., Kläser, A., Schmid, C., Cheng-Lin, L.: Action Recognition by Dense Trajectories. In: Internationnal Conference on Computer Vision and Pattern Recognition (2011)
18. Wohlhart, P., Schulter, S., Kostinger, M., Roth, P., Bischof, H.: Discriminative Hough forests for object detection. In: British Machine Vision Conference (2012)
19. Yao, A., Gall, J., Van Gool, L.: A Hough transform-based voting framework for action recognition. In: Internationnal Conference on Computer Vision and Pattern Recognition (2010)
20. Yao, A., Gall, J., Fanelli, G., Van Gool, L.: Does human action recognition benefit from pose estimation? In: British Machine Vision Conference (2011)
21. Zhang, Y., Chen, T.: Implicit shape kernel for discriminative learning of the Hough transform detector. In: British Machine Vision Conference (2010)