Hough-based Tracking of Non-Rigid Objects *

Martin Godec Peter M. Roth Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
[godec|pmroth|bischof]@icg.tugraz.at

Abstract

Online learning has shown to be successful in tracking of previously unknown objects. However, most approaches are limited to a bounding-box representation with fixed aspect ratio. Thus, they provide a less accurate foreground/background separation and cannot handle highly non-rigid and articulated objects. This, in turn, increases the amount of noise introduced during online self-training.

In this paper, we present a novel tracking-by-detection approach to overcome this limitation based on the ***generalized Hough-transform****. We extend the idea of Hough Forests to the online domain and couple the voting-based detection and back-projection with a rough segmentation based on GrabCut. This significantly reduces the amount of noisy training samples during online learning and thus effectively prevents the tracker from drifting.*

In the experiments, we demonstrate that our method successfully tracks a variety of previously unknown objects even under heavy non-rigid transformations, partial occlusions, scale changes and rotations. Moreover, we compare our tracker to state-of-the-art methods (both bounding-box-based as well as part-based) and show robust and accurate tracking results on various challenging sequences.

1. Introduction

In computer vision, visual object tracking is a major component for a wide range of applications. Domains such as surveillance, driving assistant systems, interactive games, or augmented reality require robust and reliable object detection through video sequences. While for many tasks the object is known in advance, there are numerous applications where the object to track is unknown (*e.g.*, collision avoidance, video re-targeting, or image stabilization).

A recent and popular way to address this lack of knowledge is to apply online learning to train a discriminative object detector during tracking and to adapt the detector to changes of the object over time (*e.g.*, [2, 3, 13]). These approaches formulate tracking as alternation between object detection and online learning where the current prediction is used to update the classifier. In this setting a detector for an arbitrary object can be trained from scratch. Only the initial bounding-box containing the object is needed and the type of object is not restricted to specific classes or categories.

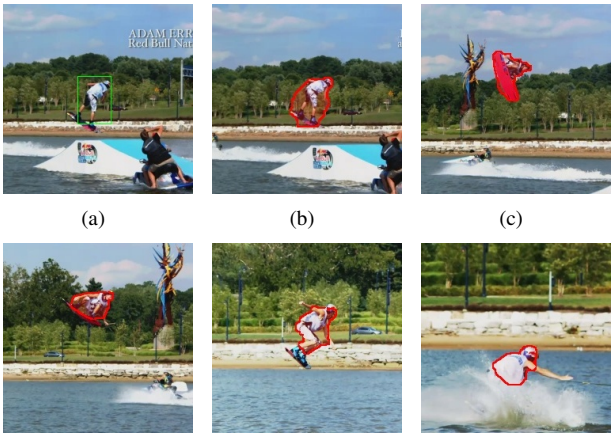


Figure 1. **Tracking of non-rigid objects**: simple bounding-box initialization in the first frame (a) and continuous tracking and segmentation of the object (b) to (f) (green: initialization; red: tracking result).

While having many advantages (*e.g.*, improving performance over time, online adaption of the object and the background model, ability to derive a discriminative object detector at any time, ...), the main challenge is to avoid drifting while still being adaptive to changes in the object's appearance. Thus, we have to distinguish between allowed transformations (*e.g.*, non-rigid deformations, rotations, appearance changes) and invalid ones (*e.g.*, occlusions, drifting). This problem is well-known as the *template update problem* [24] or the *stability-plasticity dilemma* [15] and has been addressed more or less successfully by using *e.g.*, more robust learning algorithms (*e.g.*, [28]), a different learning paradigm (*e.g.*, [3, 14]), multiple different classifiers (*e.g.*,

*This work has been supported by the Austrian FFG project MobiTrick (8258408) under the FIT-IT program.

[29, 19]), or coupling a conservative learning framework to a very adaptive tracking approach [18].

However, most of the existing approaches are limited to a rectangular bounding-box. Therefore, they have to cope with a rather inaccurate object description (*e.g.*, part of the bounding-box may consist of background). To avoid this problem, non-rigid (*e.g.*, articulated) objects can be represented by a part-based representation, such as the very prominent approach of Felzenszwalb *et al.* [9]. Even more detailed models are obtained using the generalized Hough-transform [10, 23, 25], where a large number of small object parts are combined in a voting-based framework. However, these methods need a very large amount of labeled training data. Thus, such approaches are well suited for detection/tracking of object classes that are known in advance (*e.g.*, pedestrians [11]) but are infeasible for tracking of unknown objects.

In this work, we address two major limitations of previous approaches in the tracking-by-detection domain: First, we get rid of the bounding-box limitation by transferring the Hough-based classification framework to the online domain. This allows us to robustly detect non-rigid objects. Second, we use back-projection to locate the support of our detection and to guide a segmentation process which roughly separates the object from the background. The rough segmentation delivers a more precise description of the object and is used to decrease the noise in the online learning stage which would be introduced by a simple bounding-box. Thereby, our approach allows tracking of objects with changing aspect ratio, scale, and orientation.

The overall principle is depicted in Figure 1. Starting from a bounding-box initialization, the Hough-based detector is continuously trained with the current object appearance and guides the segmentation process. Our approach robustly tracks the object during non-rigid transformations, appearance changes and partial occlusions.

1.1. Related Work

Javed *et al.* [17] and Avidan [2] initiated the use of online learning for object detection and tracking. Subsequently, Grabner *et al.* [13] transfer *Boosting for feature selection* [33] to the online domain and apply it, among other tasks, to visual tracking of objects. From that time, there is a reasonable interest in online learning for the task of visual tracking.

By defining tracking as an unsupervised online learning problem, Grabner *et al.* [14] successfully apply semi-supervised online boosting. To avoid drifting, only the training samples defined in the very first frame are considered as correctly labeled, while all samples generated by the classifier during runtime are considered as unlabeled. Alternatively, Babenko *et al.* [3] define tracking as a multiple-instance learning problem, where the current tracking position is considered uncertain and several positive samples are selected close to the current object position, arranged in a so-called bag. These learning concepts shift the problem of the sample selection from the tracking application towards the learning algorithm. Saffari *et al.* [28] transform Random Forests to the online learning domain. They use it for tracking and interactive segmentation because Random Forests have high robustness to noise. Learning of a stable object detector during tracking is the main motivation for the approach of Kalal *et al.* [18]. They combine an adaptive Lukas-Kanade-tracker and several restrictive learning constraints to establish an incremental classifier and to avoid drifting over time. Thus, the detector is quite robust and enables reliable object re-detection if the used short-term tracker fails. However, the final object detector is still based on a bounding-box.

To avoid the limitations of existing bounding-box trackers, Nejhum *et al.* [30] propose a tracker for articulated objects. They use blocks of appearance and shape descriptions but assume stationary foreground appearance. Kwon and Lee [20] define a fixed number of object parts that are automatically renewed during tracking and track the geometric relations of these parts over time. Additionally, they apply Basin Hopping Monte Carlo (BHMC) sampling to reduce the computational complexity. Bibby and Reid [5] overcome the tracking problem within a probabilistic framework. However, the high complexity of their framework makes if computationally infeasible. Thus, they separate the tracking of non-rigid objects into registration, segmentation based on level-sets and online appearance learning for continuous refinement of both object and background models. Beside that, there exist various trackers based on segmentation, but they either need prior knowledge (*e.g.*, [8]), use only very simple object appearance models (*e.g.*, color histograms [5]), or perform offline processing of the sequence (*e.g.*, [16, 32]).

In the domain of generic object detection, part-based representations are able to improve detection results for various categories of objects that contain intra-class variations. Felzenszwalb *et al.*demonstrate that the *deformable parts model* [9] allows to reliably detect objects even under heavy non-rigid transformations and partial occlusions. Using a latent SVM, they train a discriminative part-based object detector which is able to handle a small number of parts selected automatically during training phase. However, due to the complexity of the approach it is infeasible for real-time applications and online learning so far.

A different, recently revisited approach overcoming the limitations of bounding-boxes is the *Generalized Hough Transform* [4, 21]. In several recent publications, it is successfully applied to object detection [10, 23, 25], action recognition [34] and tracking [11]. Though, in the case of tracking, the detector is only able to track objects from a

certain class and learns to distinguish a specific instance from all other instances in an online manner. In addition to the detection of objects, the Hough-based classification framework also allows to determine the support of a detectors decision (*i.e.*, which positions in the image voted for the assumed object center position). This issue has been addressed in detail by Razavi *et al.* [27] and is of major interest for our work.

2. Hough Forests

Random Forests (RF) [6] are of great interest for the computer vision domain because of their speed, parallelization characteristics and robustness to noisy training data. They are used for various tasks, such as key-point recognition [22] or semantic segmentation [31]. Before we introduce our approach, we will give a brief introduction to Random Forests and the integration of the Hough votings.

2.1. Random Forests

Considering an F-channel feature space (the channels may encode *e.g.*, colors, derivatives, histograms of gradients, ...), we draw samples $\langle \boldsymbol{x}, y \rangle$, where $\boldsymbol{x}$ is a feature vector of size $F \times n \times n$ (corresponding to a rectangular image patch of size $n \times n$) and $y \in \mathcal{Y}$ denotes the corresponding class label ($\mathcal{Y} = \{-1, +1\}$ for binary classification). We access values in the samples by $\boldsymbol{x}(f, p)$, where $f \in F$ corresponds to a specific feature channel and $p \in \mathbb{N}^2$ defines a position within the rectangular image patch.

To build a classifier, tree-like structures are used, where binary splitting tests are applied at all nodes of the tree (except the leaves). The used binary tests are defined as

$$t_{f,p_a,p_b,\theta}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}(f,p_a) - \boldsymbol{x}(f,p_b) > \theta \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

where θ denotes a random threshold.

Since each tree node splits the training samples into two subsets according to its left and right child node, the basic idea of RFs is to perform this splitting recursively until the subsets are internally consistent (*i.e.*, belonging to the same class y) or a maximum depth d_T has been reached. Random Forests [6] perform an extensive selection of binary tests that optimize a certain criterion (*e.g.*, Gini-index, information gain) for each node during training.

The statistics within the leaf nodes model the probability $P_y(\boldsymbol{x})$ of the sample subset ending up in this node and being of class y. The overall classifier H is constructed by an ensemble of T classification trees and can be written as

$$H(\boldsymbol{x}) = \arg\max_y \sum_{t=1}^{T} P_y^t(\boldsymbol{x}), \quad (2)$$

where P_y^t is the probability of class y in the specific tree t.

2.2. Hough Voting

While the leaf nodes in RFs only store the probability of a sample ending up in this node being of class y, a Hough Forest additionally stores votes $\boldsymbol{d} \in \mathbb{R}^2$ that point toward the expected object center. Thus, a training sample for a Hough Forest consists of the triplet $\langle \boldsymbol{x}, y, \boldsymbol{d} \rangle$, where $\boldsymbol{d}$ is the displacement vector to the objects center position (only contained in positive (*i.e.*, $y = +1$) samples). The distribution of these votes v within each leaf node can be modeled by a sum of Dirac measures according to the displacement vectors $\boldsymbol{d}$ from all samples $\langle \boldsymbol{x}, +1, \boldsymbol{d} \rangle$ that ended up in this specific leaf node. While training a tree node of a Hough Forest, either the information gain or the uncertainty of the displacement vectors of the given training set is optimized while selecting the best test [11].

During evaluation the voting map can be generated by accumulating the voting vectors v, weighted by the foreground probability $P_{+1}(x)$ of the corresponding leaf node, for all possible locations in the image. The intensity of the voting map on a specific position corresponds to the probability of an object being centered there.

2.3. Support

Beside the detection capabilities, the voting mechanism of Hough Forests can also be applied backwards to detect the *support* of a specific position. Given a local maximum at position m, we define the support of this maximum as the sample set $S(m, \rho)$ containing all samples $\boldsymbol{x}$ that have voted to the center position m with maximum position deviation ρ. By using the corresponding voting vectors v, we can back-project the original position of samples $\boldsymbol{x}$ onto the image space. In this way, we obtain a sparse point-set of positions supposable belonging to the object that voted for the center position m.

3. Hough-based Online Tracking

Based on the work of Gall *et al.* [11], we use a Hough-voting based classification framework for tracking, which we transfer to the online learning domain. Therefore, we have to model foreground probabilities and voting maps within each leaf node online. Since the online training sample generation is a very crucial part, we have to ensure high accuracy and low label noise. Therefore, we propose to use a rough segmentation of our object, initialized by the support set S of the detected object center. We then use this segmentation to accurately update our classifier, which allows learning of highly non-rigid object deformations during tracking. Figure 2 illustrates the application flow and all parts of our tracking system.

3.1. Extremely Randomized Tree Structure

Before the Hough-based classifier can be applied for detection, the tree structure and the statistics in the leaf nodes

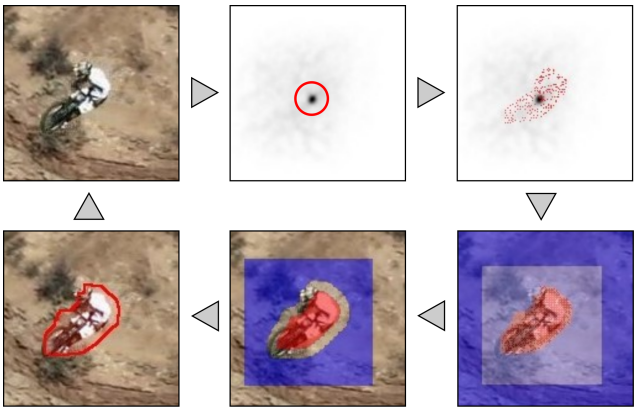


Figure 2. **Tracking Loop**: Hough-based object detection, back-projection and supporting image positions, guided segmentation, robust updating and tracking result (red: foreground support, segmentation and updates; blue: background segmentation and updates)

have to be established. In the Hough Forest approach of Gall *et al.* [11], the tree is trained using a large amount of training data. They use two types of splitting nodes: (a) classification nodes, that try to separate positive and negative training samples, and (b) regression nodes, that try to cluster voting vectors within the positive training samples presented to a specific tree node. This results in homogeneous statistics within the leaf nodes where samples have similar appearance and similar displacement from the object center. However, to generate this structure, all training data has to be available in advance. In our case, given only a single labeled sample, we perform random initialization of the tree structures. This avoids over-fitting to the single sample which would lead to bad generalization of the classifier over time since the object may change it's appearance. Following the approach of Geurts *et al.* [12], we completely randomize our tests and thresholds and abstain on optimizing of splitting tests or thresholds in the tree nodes. This is also highly related to the work of Ozuysal *et al.* [26]. To initially populate the statistics within the leaf nodes we apply online learning, as described subsequently, using the data provided by the very first frame.

Due to the complete randomization, we cannot guarantee the expressiveness of the trees which may lead to a very sparse population of the leaf nodes. Therefore, we use a larger number of trees within our classifier and measure the amount of populated leaf nodes per tree as

$$\xi_t = \sum_{n=1}^{2^{d_T}} \delta(\eta_n^+ + \eta_n^-), \tag{3}$$

with $\delta(\cdot)$ being the step function, η_n^+ and η_n^- being the number of positive and negative samples in leaf node n and d_T being the depth of the tree. This measure is used to select T trees which have the highest population ξ_t within our ensemble, which we combine using equal weights. Due to the ongoing training of our trees, the configuration of our ensemble may change during tracking.

3.2. Online Leaf Node Statistics

To establish a Hough-based classifier, we have to model (a) the foreground probability of the leaf node and (b) the corresponding voting map during online training. We model the foreground probability incrementally by counting positive and negative samples arriving at a specific leaf node during runtime. Since this would limit the adaptivity of the classifier due to saturation effects, we apply an *iir-filtering like* forgetting function,

$$\eta_n^+ = \sum_{t=-\infty}^{t_0} N_t^+ \cdot \tau^{t_0 - t}, \tag{4}$$

where η_n^+ is the weight of positive training data in leaf node n and N_t^+ is the amount of positive samples at time t. t_0 denotes the current time and $\tau \in (0...1)$ the forgetting speed. The same function is applied to the weight of negative data η_n^-, respectively. Additionally, we normalize the foreground probability P_n^+ in each leaf node n to simulate an equal amount of positive and negative training data for each tree.

Since the tests in our classification tree are not trained to cluster similar voting directions, we have to handle a very diverse set of voting vectors within a single leaf node. Therefore, we discretize the voting space into small rectangular cells and measure the weight of each of these cells incrementally. When the classifier is applied to a certain image position, we retrieve the corresponding leaf node n and select a subset of strong voting vectors from the collected voting map (see Figure 3). This is done by picking v voting cells with the largest weights ω_{cell} and setting their vote strength to

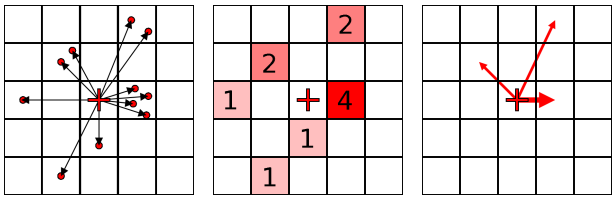
$$\omega_{vote} = P_n^+ \cdot \omega_{cell}. \tag{5}$$

To adapt the vote map to the current object configuration, we apply the same forgetting scheme as described in Eq. (4) to each cell in our voting map individually.

The described adaptations allow online training of Hough Forests using the current frame and to detect the object in the subsequent frame. Therefore, we apply the classifier to all positions in the image and accumulate the responding votes and their weights. After performing a non-maxima suppression, we assume the maximum to be the current object position.

3.3. Back-projection and Support

As denoted in Section 2.3, the Hough-voting framework can also be used to detect the support S of a given image position m using an uncertainty ρ. This is especially interesting for tracking of non-rigid objects, because the resulting



(a) (b) (c)

Figure 3. **Voting Map:** (a) training / input votes, (b) weighted voting cells, (c) weighted output votes for $v = 3$.

point set shares a stable displacement according to the object's center position. Our basic observation is that even if the object undergoes heavy deformations from one frame to another, there are still several parts of the object that keep a stable geometric configuration. Since the Hough-based voting framework works with a very large number of small patches, we have a high probability that the majority, or at least the stable parts of the object, will be covered by such supporting points. Since we adapt the voting map over time, the stable parts are not static but may change over time. Similar to the appearance model, the voting directions just have to fit the current object configuration.

3.4. Closing the Tracking-Loop

Up to now, we have defined all parts that are necessary to perform online learning in a Hough-voting based classification framework. However, there is a crucial part missing to close the tracking loop: *online training sample selection*. Selecting the right update strategy is an important part for online tracking-by-detection. The major problem is that the correctness of the tracking result is not guaranteed (due to misalignments, occlusions or cluttered background) but the learning algorithm has to generate training samples including as little noise as possible. To avoid the problem of a coarse bounding-box annotation, we want to separate the object from the background on a more fine-grained level to get more accurate training data. Therefore, we use the support S of the detected object position (*i.e.*, the parts having a stable geometric relation to the object center) to guide a rough segmentation process that extracts the object. Even if this segmentation is not very precise, it lowers the amount of noise which is introduced to the online learning. We apply the well-known *GrabCut* [7][1] algorithm to establish a reasonable binary segmentation using the color channels, initialized by the support $S(m, \rho)$ of our object position as foreground and a maximum-object-sized rectangle as background. This rough segmentation separates our image into two regions: positive samples, located on the object and negative ones, located in the background. Since we do not rely on an exact segmentation (due to *e.g.*, missing parts, over-segmentations) we consider a narrow band in-between these two sets as uncertain and do not use this region for training.

To be adaptive to geometric reconfiguration of the object, we shift the object's center position to the current center-of-mass in the foreground segment. Thus, the object center detected by our classifier represents the center of the currently visible part of the tracked object, because we do not distinguish between occlusions and geometric reconfiguration of the object. However, this simple but efficient strategy delivers accurate training data which is used to update our classifier during tracking. If the segmentation fails, our tracker acts like a bounding-box-based tracker, but Random Forests are known to be robust to noise and are able to handle a notable amount of incorrectly labeled samples.

4. Evaluation

To demonstrate the performance of our tracking approach denoted as *HoughTrack* (HT), we evaluate and compare it to existing approaches using two different datasets. The first dataset was originally defined to evaluate bounding-box trackers and is used extensively in many publications. We evaluate our approach on this dataset to show the general applicability of our tracking approach. Additionally, we collected a set of very diverse and challenging sequences including highly non-rigid object transformations.

We use the same settings for all sequences: the classifier pool consists of 20 trees and we pick the $T = 10$ trees with the highest population for detection. Our trees are fully grown to a maximum depth of $d_T = 8$ and we are using Lab-color space (3 channels), first and second derivatives in x and y direction (4 channels) and a 9-bin histogram of gradients (9 channel) as feature channels $\boldsymbol{x}^f$. The used patch size of our samples is 12×12 and we return $v = 10$ strong votes from a leaf node if a sample ends up there. Our forgetting constant τ is set to 0.9 and the maximum support deviation ρ is 0.5.

4.1. Standard Dataset

For quantitative analysis, we use the publicly available tracking dataset by Babenko *et al.* [3], which consists of 8 sequences collected from several different publications and an overall size of more than 5000 frames. We compare to *MILTrack* [3] using the original configuration of 50 weak classifiers and *Online Random Forests* [28] using 50 trees and standard settings provided by the implementation. Since the compared trackers only report bounding-boxes, we also convert our result to bounding-boxes of original size, centered around the center-of-mass of our segmentation. Table 1 clearly shows that our approach delivers competitive results, even not considering partial or full occlusions in the evaluation due to the lack of annotations.

[1] We used the implementation from http://opencv.willowgarage.com.

Sequence	**HT**	MIL [3]	ORF [28]
David	**100**	84	95
Sylvester	**99**	93	71
Girl	86	85	**99**
Face Occlusion 1	**100**	91	**100**
Face Occlusion 2	**100**	94	70
Coke	24	**46**	17
Tiger 1	45	78	27
Tiger 2	71	**78**	21
Average	78	81	63

Table 1. **Babenko Sequences**: Percentage of correctly tracked frames ($score > 0.5$) for all sequences and average.

Based on the ground-truth annotation included in the dataset of Babenko *et al.* [3], which is represented by a simple bounding-box of the same size as the initialization, our tracker cannot be compared fairly with other bound-box-based trackers because object occlusions are ignored completely. To alleviate the influence of occlusions to the overall performance, we measure the tracking accuracy using the Agarwal-criterion [1], which is defined as $score = \frac{R_T \cap R_{GT}}{R_T}$, where R_T is the tracking rectangle and R_{GT} the ground truth. We report the amount of successfully tracked frames ($score > 0.5$), since this value is less sensitive to the effect described above. Figure 4 shows some selected frames from the dataset and demonstrates that the raw accuracy values from Table 1 fail to meet the true performance of our tracking approach.

4.2. Tracking of Non-Rigid Objects

Since the intended purpose of our tracking approach is the tracking of objects that may deform during runtime, we want to demonstrate the performance on several challenging sequences. Therefore, we have collected several videos showing different ranges of complexity and non-rigid deformations, consisting of about 2500 frames. We compare to *Basin Hopping Monte Carlo Tracking* (BHMC)[2], because this tracker solves a similar task. We also include the sequences provided by the authors in our comparison (*Transformer*, *Diving*, *High Jump*, and *Gymnastics*).

We also list the results of *Online Random Forests* (ORF)[3], a bounding-box-based tracker that is not designed to cope with the amount of transformation presented in this videos. Since this tracker cannot adapt the aspect ratio of the bounding-box, we accept the tracking result to be correct if the center position of the tracked bounding-box is roughly correct, although the result is much more inaccurate than using the two other approaches.

Table 2 depicts tracking results of the selected approaches evaluated on our sequences. We have denoted the percentage of frames for each sequence until the tracking approach fails by visual inspection. Figure 5 shows some selected frames of our sequences and our tracking results.

Sequence	**HT**	BHMC [20]	ORF [28]
Cliff-dive 1	**100**	**100**	**100**
Motocross 1	**100**	5	15
Skiing	60	0	5
Mountain-bike	**100**	50	**100**
Cliff-dive 2	**100**	30	50
Volleyball	**100**	60	45
Motocross 2	**100**	25	10
Transformer	**100**	**100**	**100**
Diving	35	**100**	30
High Jump	**100**	**100**	5
Gymnastics	10	**100**	65
Average	82	61	48

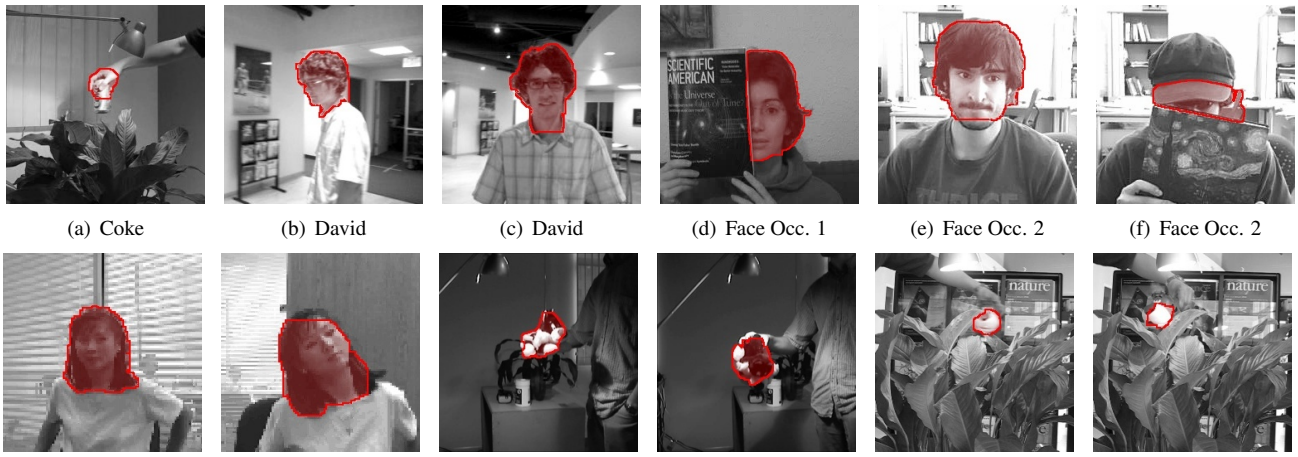
Table 2. **Tracking of non-rigid objects**: Percentage of frames correctly tracked until failure.

4.3. Discussion

We have defined a maximum object size that is used for background initialization of our segmentation algorithm. If the segmentation fails, it is not allowed to grow beyond this maximum scale. This prevents leakage of the object segmentation. This effect can be seen in sequence *Cliff-dive 2* (3^{rd} row in Figure 5). It is clearly visible that the segmentation changes over time and it gets more accurate during tracking.

Since we use a rectangular initialization of our tracker in the first frame, the support of our detection in the subsequent frames may also include background positions because they have also been included in the initial training set. This may end up in confusion of the classifier. However, these votes disappear as soon as the object moves and the according votes do not match the support criterion any more. Only if the majority of the support originates from the near background of the object, the recognized object center will be supported by the background and the tracker is not able to follow the object any longer. This effect may occur when there is very cluttered background (Sequence *Diving*) or the segmentation algorithm fails due to similar colors in the background (Sequence *Gymnastics*). Since our classifier's structure is created without any training data, we have used the same binary test within a whole tree level. Thus, our implementation is very related to Random Ferns [26]. For comparison to future approaches, our reference implementation and the used sequences are available online[4].

[2]Implementation from http://cv.snu.ac.kr/research/~bhmctracker/.
[3]Implementation from http://lrs.icg.tugraz.at/download/.
[4]http://lrs.icg.tugraz.at/research/houghtrack/.



(a) Coke (b) David (c) David (d) Face Occ. 1 (e) Face Occ. 2 (f) Face Occ. 2

(g) Girl (h) Girl (i) Sylvester (j) Sylvester (k) Tiger 1 (l) Tiger 2

Figure 4. **Illustrative Tracking Results**: Selected frames from the *Babenko Sequences*.

5. Conclusion

In this work, we present a tracking approach that is able to handle non-rigid object deformations, part- and self-occlusions and transformations such as rotation or scaling quite naturally. By the combination of a Hough-voting based classification framework, online learning techniques, and an out-of-the-box segmentation algorithm we are able to track various objects in several challenging sequences. We use the classification framework for both detection of the object and finding the support of the currently estimated object position to guide the segmentation process. The segmentation allows for a rough per-pixel separation of object and background which itself enables more robust training of the classifier. If the segmentation fails and background is included in the training process, our learner is able to handle a large amount of noise.

In future work, we plan to investigate the robustness of our learner and the appropriate integration of prior information. To further enhance the tracking, the incorporation of a motion model and a more advanced occlusion handling that also supports re-detection of the object would be beneficial. Issues regarding the speed can be addressed by parallelization of all modules of our tracker, especially because Random Forests/Ferns can be parallelized easily. Due to the high parallelization capabilities, implementation on a Graphics Processing Unit (GPU) for the whole approach (including segmentation) would be beneficial. That would also allow for the use of more complex segmentation algorithms which would further improve the overall performance of our approach.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. PAMI*, 2004. 6

[2] S. Avidan. Ensemble tracking. In *Proc. CVPR*, 2005. 1, 2

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Proc. CVPR*, 2009. 1, 2, 5, 6

[4] D. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 1981. 2

[5] C. Bibby and I. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proc. ECCV*, 2008. 2

[6] L. Breiman. Random forests. *Machine Learning*, 2001. 3

[7] A. B. Carsten Rother, V. Kolmogorov. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 2004. 5

[8] D. Cremers and G. Funka-lea. Dynamical statistical shape priors for level set based tracking. *IEEE Trans. PAMI*, 2006. 2

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 2010. 2

[10] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Proc. CVPR*, 2009. 2

[11] J. Gall, N. Razavi, and L. van Gool. On-line adaption of class-specific codebooks for instance tracking. In *Proc. BMVC*, 2010. 2, 3, 4

[12] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 2006. 4

[13] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, 2006. 1, 2

[14] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. ECCV*, 2008. 1, 2

[15] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 1987. 1

[16] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *Proc. CVPR*, 2010. 2

[17] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. CVPR*, 2005. 2

Figure 5. **Illustrative Tracking Results**: Initialization (first column) and selected frames (column 2 to 6) (green: initialization; red: tracking result; *best viewed in color*). More results can be found at http://lrs.icg.tugraz.at/research/houghtrack/.

[18] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *Proc. CVPR*, 2010. 2

[19] T. K. Kim, B. Stenger, T. Woodley, and R. Cipolla. Online multiple classifier boosting for object tracking. In *Proc. Online Learning for Computer Vision Workshop*, 2010. 2

[20] J. Kwon and K. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *Proc. CVPR*, 2009. 2, 6

[21] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *In ECCV workshop on statistical learning in computer vision*, 2004. 2

[22] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Trans. PAMI*, 2006. 3

[23] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Proc. CVPR*, 2009. 2

[24] L. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. PAMI*, 2004. 1

[25] R. Okada. Discriminative generalized hough transform for object dectection. In *Proc. ICCV*, 2009. 2

[26] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Trans. PAMI*, 2010. 4, 6

[27] N. Razavi, J. Gall, and L. van Gool. Backprojection revisited: Scalable multi-view object detection and similarity metrics for detections. In *Proc. ECCV*, 2010. 3

[28] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *Proc. On-line Learning for Computer Vision Workshop*, 2009. 1, 2, 5, 6

[29] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST Parallel Robust Online Simple Tracking. In *Proc. CVPR*, 2010. 2

[30] S. Shahed Nejhum, J. Ho, and M.-H. Yang. Visual tracking with histograms and articulating blocks. In *Proc. CVPR*, 2008. 2

[31] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Proc. CVPR*, 2008. 3

[32] D. Tsai, M. Flagg, and J. M.Rehg. Motion coherent tracking with multi-label MRF optimization. In *Proc. BMVC*, 2010. 2

[33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001. 2

[34] A. Yao, J. Gall, and L. van Gool. A hough transform-based voting framework for action recognition. In *Proc. CVPR*, 2010. 2