Hough Forests Revisited: An Approach to Multiple Instance Tracking from Multiple Cameras

Georg Poier[†], Samuel Schulter[†], Sabine Sternig, Peter M. Roth, Horst Bischof[†]

[†]Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{poier,schulter,bischof}@icg.tugraz.at, sternig.sabine@gmail.com,
p.m.roth@ieee.org

Abstract. Tracking multiple objects in parallel is a difficult task, especially if instances are interacting and occluding each other. To alleviate the arising problems multiple camera views can be taken into account, which, however, increases the computational effort. Evoking the need for very efficient methods, often rather simple approaches such as background subtraction are applied, which tend to fail for more difficult scenarios. Thus, in this work, we introduce a powerful multi-instance tracking approach building on Hough Forests. By adequately refining the time consuming building blocks, we can drastically reduce their computational complexity without a significant loss in accuracy. In fact, we show that the test time can be reduced by one to two orders of magnitude, allowing to efficiently process the large amount of image data coming from multiple cameras. Furthermore, we adapt the pre-trained generic forest model in an online manner to train an instance-specific model, making it well suited for multi-instance tracking. Our experimental evaluations show the effectiveness of the proposed efficient Hough Forests for object detection as well as for the actual task of multi-camera tracking.

1 Introduction

Visual object tracking is one of the most important tasks in computer vision, building the basis for various applications such as surveillance, sports analysis, and industrial (quality) inspection [24,26,32]. Even though there are considerable advances in single object tracking (*e.g.*, [21, 22]), there are still open challenges for multi-object tracking [38, 40], in particular if multiple instances of the same class interact.

However, by conducting well elaborated reasoning relying on part-based approaches (*e.g.*, [4, 39]) even complex scenarios can be handled very well with a single camera. Nevertheless, such methods fail if instances are fully occluding each other. To overcome this problem, recent approaches additionally exploit temporal information and analyze the tracks of individual instances over time [9, 13]. However, such methods are condemned to fail if the assumptions (*e.g.*, constant velocity) are hurt. A natural way to resolve the problem of similar

objects occluding each other is to exploit the additional information provided by multiple cameras [15, 26].

The increased amount of data captured by multiple cameras, however, limits the computational complexity of the employed tracking algorithms. Thus, often simple techniques like background subtraction or color models are used in order to locate foreground objects [15,23,30]. However, the simplicity of these methods usually yields severe problems like mistaken instances or ghost detections. In contrast, computationally more complex object detection models, such as [35], are often lacking efficiency and cannot be applied for real-world scenarios.

Thus, the goal of this work is to provide stronger models for tracking multiple instances using multiple cameras, however, still ensuring short response times and thus real-time capabilities. In particular, we focus on Hough Forests (HFs) [19] as underlying object model in a tracking-by-detection setup. In general, HFs are a powerful and versatile extension of Random Forests (RFs) [8] and have been successfully applied to various tasks including object detection [27, 36], tracking [17, 33], and pose estimation [34, 37].

For real-time tracking, however, their original formulation bears several drawbacks. For example, the runtime is strictly correlated with the number of training samples, as we will point out later. Hence, a major advantage of Random Forests, where the test time is independent from the amount of training data, is lost. This is especially a problem as the amount of training data is a crucial parameter for the accuracy of Random Forest based models [11, 18, 34]. Furthermore, as we also demonstrate, redundant and unnecessary image information is extracted and processed, indicating an *over-determined* object description.

Thus, the contribution of this work is twofold. First, we provide an efficient but still effective object detector based on Hough Forests. Second, this detector is adopted for online learning to allow real-time tracking from multiple cameras. We first, provide a theoretical as well as empirical analysis of Hough Forest based object detection, with special emphasis on computational efficiency. Investigating different aspects such as the classifier complexity, the underlying data complexity, and the final prediction reveals that the runtime can be drastically reduced, without a loss in accuracy. In fact, the performance of the proposed method is on par with the baseline approach, while being one to two orders of magnitude faster.

Once, having identified and eliminated the critical bottlenecks, Hough Forests can straightforwardly be applied for tracking. To this end, we first introduce a novel online learning strategy, where a pre-trained generic object classifier is adapted to discriminate specific instances on the fly. Second, this strategy – in contrast to similar approaches – allows integration of information from different views into one single classifier. This, on the one hand, reduces the complexity of the classifier as redundant information need not be modeled in parallel. On the other hand, it provides an elegant way to share features from different views in a single RF. Finally, the evaluations confirm that in this way an efficient and effective multi-camera tracking system can be built.

2 Hough Forests

Hough Forests [19, 27] combine the flexibility of Implicit Shape Models (ISMs) [25] with the efficiency of Random Forests [1, 8, 11], by integrating part-based classification and regression into a single model. Given a small image patch, the goal is to classify if it originates from an object as well as to predict the possible object location. In the following, we will briefly review the main ideas of HFs [19]. For more details we would like to refer to, *e.g.*, [18, 19].

The prediction process of HFs is based on the generalized Hough transform [3], where the Random Forest represents a codebook of local appearances (*i.e.*, small image patches). More specifically, the predictor model at the leafs of the trees includes a class histogram as well as offset vectors representing the relation of the observed image patches to a specified position on the object (*e.g.*, the object center). These offsets are used to cast votes for the object's location and scale. The codebooks are optimized such that the cast votes exhibit small uncertainty, which is crucial for accurate predictions [19]. In contrast to the codebook used in [25], the efficiency of the forest framework also permits dense sampling of local image patches, yielding an additional gain in accuracy.

In order to learn the codebook, a Hough Forest is given a labeled training set $\mathcal{L} = \{(\mathbf{x}_i, y_i, \mathbf{o}_i)\}$, where $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^D$ and $y_i \in \mathcal{Y} = \{0, \dots, (C-1)\}$ [1] represent the features and the corresponding class label, respectively, and $\mathbf{o}_i \in \mathcal{O} = \mathbb{R}^2$ is the offset vector. Each tree in the forest aims at splitting the given data into small subsets by simultaneously minimizing the uncertainty in the class labels and the offset vectors. For that purpose, a binary splitting test $\phi(\mathbf{x})$ is employed to split the data into two subsets $\mathcal{L}_R$ and $\mathcal{L}_L$. This test is found by generating a number of possible tests $\{\phi_k(\mathbf{x})\}$ at random for each node of a tree and selecting the test $\phi_*(\mathbf{x})$ which minimizes either a classification or a regression objective.

The classification objective $U_\mathcal{Y}(\mathcal{L}, \phi(\mathbf{x}))$ is designed to minimize the class uncertainty in the resulting sets $\mathcal{L}_R$ and $\mathcal{L}_L$. It is based on the entropy $H(\mathcal{L}) = -\sum_{y=0}^{C-1} p(y|\mathcal{L}) \ln p(y|\mathcal{L})$ computed for each set and weighted according to the number of samples contained in the respective set. Here $p(y|\mathcal{L})$ denotes the empirical class probability for class y, estimated from dataset $\mathcal{L}$. The regression objective $U_\mathcal{O}(\mathcal{L}, \phi(\mathbf{x}))$, on the other hand, aims at minimizing the uncertainty of the offset vectors and is based on their deviation from the mean $\bar{\mathbf{o}}$: $V(\mathcal{L}) = \sum_{y \in \mathcal{Y} \setminus \{0\}} \sum_{\mathbf{o} \in \mathcal{O}_P^{(y)}} \|\mathbf{o} - \bar{\mathbf{o}}\|_2^2$. The value of the objective function for a given test $\phi(\mathbf{x})$ is obtained by simply summing up the uncertainties in both sets, *i.e.*, $U_\mathcal{O}(\mathcal{L}, \phi(\mathbf{x})) = V(\mathcal{L}_L) + V(\mathcal{L}_R)$.

In terms of object detection with Hough Forests, a data sample $\mathbf{x}$ represents the appearance of a small image patch. To detect an object, all patches $\mathbf{x}_\mathbf{v}$ of a given image are associated to a leaf l_t of each tree by evaluating the splitting functions. The class probability $p_t(y|\mathbf{x}_\mathbf{v})$ and the offset vectors $\mathbf{o} \in \mathcal{O}_{l_t}^{(y)}$ of class y stored at the leaf are then used to cast votes for an object at position $\mathbf{u}$ and scale s. From the votes accumulated over a number of scales, finally the object positions and scales are identified.

[1] Note that $y = 0$ for the background class.

3 Efficient Hough Forests

In the following, we investigate the efficiency of Hough Forests for the task of object detection. For that purpose, we first determine the relevant parameters for the different parts of an ISM based detection approach, *i.e.*, feature computation, matching local image patches to the codebook, and casting votes into the Hough space. Second, we analyze the effectiveness of these parameters with respect to run-time and accuracy. We run different experiments on a standard benchmark dataset, namely *TUD-Pedestrian*[2] [2]. For performance evaluation we employ the standard PASCAL overlap criterion [16]. To ensure statistically valid results, each experiment was repeated 10 times and the averaged results are reported. As we are solely interested in the relative performance changes according to the specific parameters, the results are shown with respect to the original reference implementation.

3.1 Complexity Analysis

The first relevant parameter is the run-time for feature computation $\mathcal{C}_\mathcal{F}$. As we use a dense sampling strategy, which has shown to be crucial for accurate detection [19], the features have to be computed for each image pixel. Thus, the feature computation is independent from n, the number of actually used patches $\mathbf{x}_i$. Thus, for a single scale, $\mathcal{C}_\mathcal{F}$ only depends on the number of feature channels F and image pixels I: $\mathcal{C}_\mathcal{F} = O(FI)$.

While for the original ISM [25] the cost $\mathcal{C}_\mathcal{M}$ of matching a single data sample to the codebook linearly scales with the size of the codebook, for codebooks based on a Random Forest it logarithmically depends on the codebook size. In fact, $\mathcal{C}_\mathcal{M}$ is estimated as the sum of the cost of matching a sample to a leaf of each tree within the forest: $\mathcal{C}_\mathcal{M} = \sum_{t=1}^{T} O\big(\kappa \log(|l_t|)\big)$, where $|l_t|$ specifies the number of leafs of tree $t \in \{1, \dots, T\}$, and κ denotes the complexity of a single splitting test. Ignoring that $|l_t|$ may slightly differ from tree to tree, we get $\mathcal{C}_\mathcal{M} = O\big(T\kappa \log(|l|)\big)$.

Hence, assuming balanced trees, we finally derive the following overall detection costs:

$$\mathcal{C} \;=\; \mathcal{C}_\mathcal{F} + n\Big(\mathcal{C}_\mathcal{M} \;+\; \frac{T\,|\mathcal{O}_\mathcal{L}|}{|l|}\Big) \;=\; O\left(FI + nT\Big(\kappa \log\left(|l|\right) + \frac{|\mathcal{O}_\mathcal{L}|}{|l|}\Big)\right), \qquad (1)$$

where $|\mathcal{O}_\mathcal{L}|$ is the number of all offsets in the training set.

As our final target application is tracking, where the scale is roughly known, parameters regarding the scale space, which are essential for the efficiency of modern object detection systems (see, *e.g.*, [6, 14]), are not considered in the following. On the other hand, Eq. (1) reveals that the test time directly depends on the amount of training data, as for ISM based object detection each training sample is assigned one offset vector, *i.e.*, $|\mathcal{O}_\mathcal{L}| = |\mathcal{L}|$.

[2] `http://www.d2.mpi-inf.mpg.de/andriluka_cvpr08`

In the following, we analyze the parameters identified in Eq. (1). We structure them into those related to the complexity of the data representation and those related to the complexity of the classifier.

3.2 Data Complexity

Data complexity is manifested in three parameters, which are discussed in the following: the number of data samples n extracted from a test image, the number of feature channels F, and the number of image pixels I at which the features need to be computed:

Amount of Data Samples: Gall *et al.* showed [19] that sampling patches densely results in more accurate detections compared to sampling from interest points only [25]. On the other hand, *nearly dense* sampling strategies have proven to be almost as effective in terms of accuracy, whereas the number of samples and, thus, the runtime is significantly reduced. Increasing the sampling distance from one (dense) to two, already results in a speed-up of 40% at the same level of accuracy (see Fig. 1a).

Feature Channels: Originally, it was proposed to use 32 feature channels, however, different features may be highly correlated, representing similar information (see *e.g.*, [8]). To this end, evaluations using different subsets of feature channels reveal that more than half of the channels may be simply omitted without any loss in accuracy. Especially by omitting the min- and max-filtration [19] the runtime can be reduced by approx. 20% (see next paragraph and Fig. 1b, respectively).

Object Size: The smaller the scale at which an object can be detected, the less pixels have to be considered for feature computation. Hence, size matters. An according evaluation, starting with a unit object height of 100 pixels [19] is given in Fig. 1b. Moreover, filtration of the feature channels was found to be unnecessary for smaller scales since noise suppression is implicitly achieved by down-scaling. In fact, an additional filter step even decreases the discriminability at smaller scales. Thus – similar to the experiments regarding the number of data samples and feature channels – this points out that the model is strong enough to work well on a more compact object description. Please note that all results in Fig. 1b were generated without filtration of the feature channels but are relative to the standard setting, *i.e.*, with filtration.

3.3 Classifier Complexity

The complexity of the Hough Forest classifier is given by the number of trees, their depth, the complexity of a split tests, and by the complexity of the offset distribution. Due to space limitations we omit discussions on the depth and split complexity since these parameters did not yield a speed up without sacrificing accuracy for our experiments:

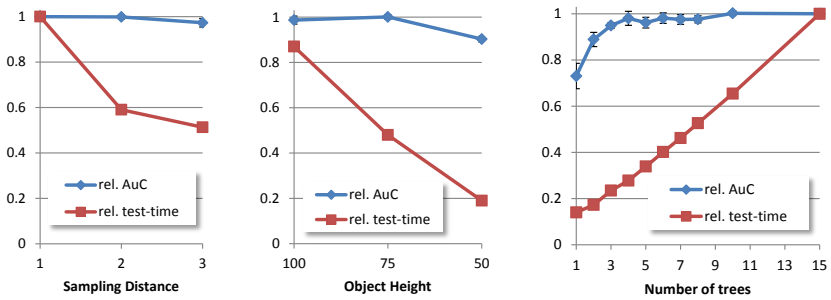


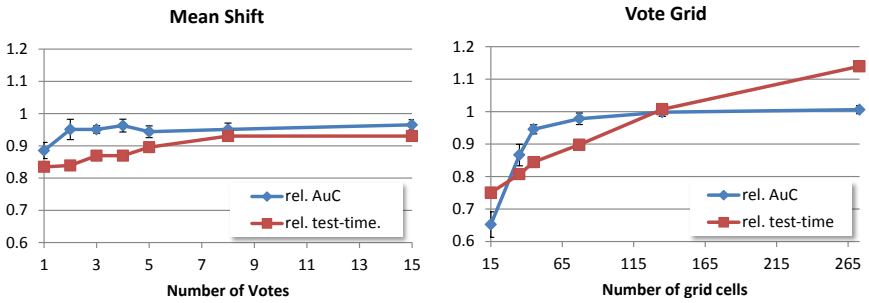
Fig. 1. Relative accuracy (in terms of *Area under the Curve* (AuC)) and relative test time per image for (a) different sampling distances during detection, (b) models trained on different object resolutions, and (c) using different forest sizes. For the results in (b) no min- and max-filtration were used, *i.e.*, 16 feature channels. The values are relative to the result obtained using the standard setting, *i.e.*, using dense sampling, object height: 100, patch size: 16×16, and using 32 channels (including min- and max-filtration). The error bars give the standard deviation for AuC.

Number of Trees: The number of trees T of a Random Forest provides a simple way to trade off accuracy versus runtime. In fact, the runtime scales linearly with the number of trees in the forest, while the accuracy usually levels up at some point (*c.f.* [28]). In our setup, the accuracy shows only a slight increase for $T > 4$ (see Fig. 1c).

Complexity of the offset distribution: A major drawback of Hough Forests, limiting their applicability, is that their test time depends on the amount of offset vectors stored at the leafs, and thus, on the amount of training data. While in [34] this drawback is addressed by summarizing voting information at the leaf nodes off-line, the authors of [21] employ a *grid-like* quantization of the offset distribution in order to update the distribution incrementally during online learning. Considering applications which require online learning as well as computational efficiency, we would like to take advantages of both ideas. Thus, we apply a grid-like approximation of the offset distribution on a coarse scale. The corresponding results over different grid resolutions compared to the results for vote compression using mean-shift as in [20] are shown in Fig. 2.

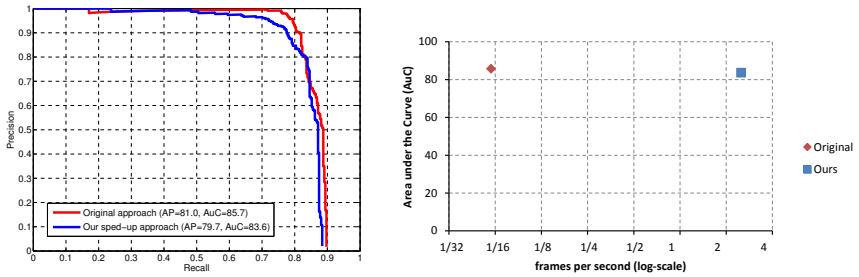
3.4 Discussion

The results presented in this section clearly show that we can enhance the efficiency of a Hough based object detector without negatively affecting the accuracy. In fact, integrating all findings gives an accuracy comparable to the original approach [19], while the computation is sped up by a factor of 40 (see Fig. 3). In particular, when combining the adaptions for all described parts the best results were obtained by using the following setup: two pixel sampling distance



(a) (b)

Fig. 2. Relative accuracy (in terms of *Area under the Curve* (AuC)) and relative test time per image: (a) As a function of the number of votes for vote compression by mean shift, and (b) as a function of the number of cells for vote compression by vote grid. Values are relative to voting with all offsets arrived at a leaf. The error bars give the standard deviation for AuC.



(a) (b)

Fig. 3. Precision-Recall curves (a), and accuracy plotted vs. runtime (b) for the original [19], and our sped-up approach. It can be seen that the loss in accuracy is small, while the system is sped up by a factor of 40. (image size: 720x576)

in x-/y-direction, an unit object height of 75 pixels, discarding the min-, and max-filtration of the feature channels, reducing the number of trees from 15 to 5 and summarizing the offset distributions by utilizing grids of 77 cells.

Note that the findings of recent works regarding efficiency of scale space analysis [6, 14] can be easily integrated, which bears the potential for an additional speed-up. Moreover, these adaptions are quite general and straightforwardly applicable to tracking in a tracking-by-detection manner.

4 Multi-Camera Multi-Instance Tracking

In the following, we first introduce a new forest-based multi-camera tracking method exploiting the findings from the previous section and then demonstrate the method in comparison to existing approaches for a realistic scenario.

4.1 Adaptive, Instance-Specific Models

In general, a multi-camera setup consists of several cameras observing the same scene, typically assuming that the objects are moving on a common ground plane. In our case, we build on a tracking-by-detection technique using a Hough Forest as underlying classifier. Thus, our approach is similar to [35], also using Hough Forest based person detectors, which are then updated online according to [17]. However, our approach differs in several ways, as [35] uses a ground plane voting and a separate model for each camera. Furthermore, their approach does not exploit any visual information for discriminating different instances but solely relies on temporal information based on a particle filter.

In contrast, we apply the original center voting [19]. However, at test time the votes are offset, such that they actually point to the ground plane. For that purpose, the offset from center to foot-point is computed based on the scale of the detected object. In this way, we limit the length of the learned vote vectors $\mathbf{o}_i$, since it was shown that the performance of Hough voting schemes decrease with the length of those [10, 12, 34]. Hence, we can now vote for the foot-points of the persons, without increasing the uncertainty in the voting process due to artificially elongated votes.

To further reduce the model's complexity, we exploit that an object's appearance may be very similar in different views at different points in time. Thus, instead of learning different models in parallel (*i.e.*, for each camera) we accumulate all information into a single comprehensive classifier, simultaneously updated from all views. To this end, we also exploit the ability of Random Forests to handle a huge amount of data and a significant amount of noise [5, 18, 29], which could be introduced by the large variability in the data from different views.

However, we are not interested in detecting generic persons but in distinguishing between specific ones, which can be realized by adapting the splitting tests. Obviously, the splitting tests of a generic person detector are not very discriminative when aiming at separation of specific instances. For example, color information is not intensively used by a generic forest classifier, as it is not well suited for distinguishing between foreground and background. However, for instance separation this is apparently one of the most important cues. Hence, a generic person detector is not an appropriate choice for separation of specific instances. A Hough Forest – being a part-based detector – trained for person detection is able to segregate the individual parts of an object. For example, for person detection, parts like feet, the head, or the torso are clustered together. The color of each part is highly informative when it comes to discrimination between object instances. Thus, in order to differentiate between individual instances we extend the pre-trained model online by learning additional splitting tests, which are specifically optimized to separate the instances occurring in the scene.

To do so, the data samples extracted from the detected instances are traversed down the trees reaching the respective leafs. Similar to [33] we collect the samples together with their instance specific offsets at the leafs until a fixed

number of samples n arrived. Subsequently, a split is optimized on a balanced sub-sample of $m\,|E|$ instance samples, where $|E|$ denotes the number of instances and $m < {}^{n}/_{|E|}$. In our experiments, this has shown to be preferable over sampling from the posterior distribution at the node. (For this work we set $m = 20$ and $n = 1.5m\,|E|$.) The process is repeated for three additional depth levels. Furthermore, in order to maintain the good generalization performance, we keep the samples from the pre-trained classifier, split them according to the selected splitting tests, and use them to build reliable statistics at the newly created leafs. Since only very few additional splits are generated, the samples need not be kept forever, which would obviously require infinite memory and hamper runtime. Instead, after a (pre-)defined number of splits, we keep the tree depth fixed and discard all collected samples. Subsequently, only the (class, instance, and offset) statistics are updated.

4.2 Experimental Results

To demonstrate the benefits of our approach, we pre-trained a Hough Forest on the publicly available INRIA person dataset[3] and compared our approach to three different methods. First, to a simple baseline based on mapping foreground masks obtained from a background subtraction onto the ground plane. Second, to Berlcaz *et al.* [7], which uses K-Shortest Patch (KSP) to link the detections obtained from probabilistic occupancy maps (POM). Third, to the most related approach of Sternig *et al.* [35], which also builds on Hough Forests.

All approaches were evaluated on *Set 1* from the publicly available dataset [31], which shows an indoor scene captured by three different cameras. Consisting of more than 2500 frames (where every tenth frame is annotated and used for evaluation), the scene shows three persons walking around, regularly occluding each other. The thus obtained results are presented in Table 1, where we show the averaged error (localization) on the ground plane. It can be seen that our single comprehensive classifier (*MultiInstanceHF*) outperforms not only the simple baseline but also the more sophisticated approaches of Berclaz *et al.* and Sternig *et al.* The latter one is of particular interest, as we do not use an additional particle filter and ensure a much lower computational effort. Furthermore, illustrative results are given in Fig. 4.

5 Conclusion & Outlook

In this work, we revisited Hough Forests, a prominent approach to object detection, which has been successfully applied to numerous tasks within the field of computer vision. We pointed out that – using simple means – their runtime can be reduced by one to two orders of magnitude, while scoring in the same range of accuracy. This enables their use for applications with limited computational budget. The gathered insights were then exploited for tracking multiple

[3] `http://pascal.inrialpes.fr/data/human/`

Table 1. Mean pixel errors on ground plane for different approaches on the multi-camera sequence *Set 1* [31].

Method	Error (in pixel)
background subtraction	75.7
Berclaz *et al.* [7]	106.3
Sternig *et al.* [35]	23.9
MultiInstanceHF	**18.8**

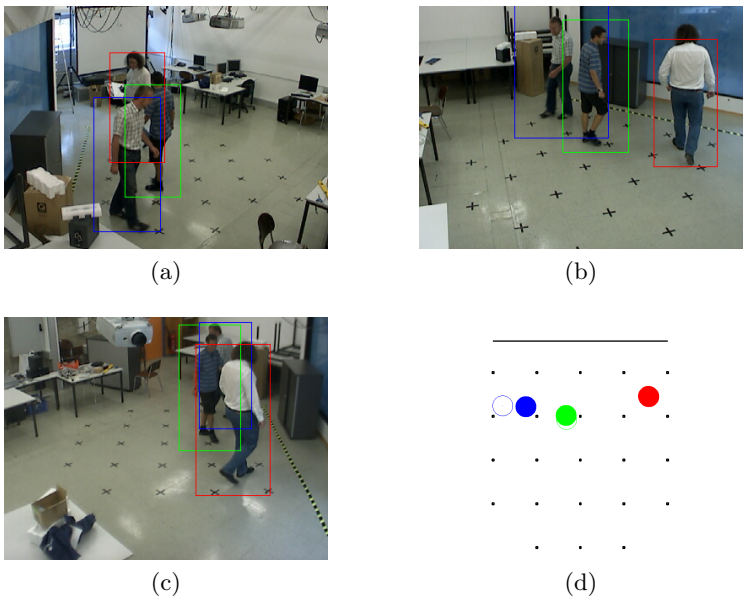


Fig. 4. Illustrative tracking results overlaid on the camera views (a-c) and the ground plane (d), where the filled circles represent the tracking result and the unfilled the corresponding ground truth annotations (best viewed in color and high definition).

instances from multiple cameras, where we showed that visual information can be as effective for instance discrimination as temporal information. This points out that instance discrimination should not be fully handed over to methods that only take temporal consistency into account. Instead, our work motivates an approach where both visual and temporal cues are incorporated.

Acknowledgment This work was supported by the Austrian Science Foundation (FWF) project Advanced Learning for Tracking and Detection in Medical Workow Analysis (I535-N23).

References

1. Amit, Y., Geman, D.: Randomized inquiries about shape; an application to handwritten digit recognition. Tech. Rep. 401, Department of Statistics, University of Chicago, IL (1994)
2. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: Proc. CVPR (2008)
3. Ballard, D.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition 13(2), 111 – 122 (1981)
4. Barinova, O., Lempitsky, V.S., Kohli, P.: On detection of multiple object instances using hough transforms. IEEE Trans. PAMI 34(9), 1773–1784 (2012)
5. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
6. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian detection at 100 frames per second. In: Proc. CVPR (2012)
7. Berclaz, J., Fleuret, F., Fua, P.: Multiple object tracking using k-shortest path optimization. IEEE Trans. PAMI 9(33), 1806–1819 (2011)
8. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
9. Butt, A.A., Collins, R.T.: Multi-target tracking by lagrangian relaxation to min-cost network flow. In: Proc. CVPR (2013)
10. Cootes, T.F., Ionita, M.C., Lindner, C., Sauer, P.: Robust and accurate shape model fitting using random forest regression voting. In: Proc. ECCV (2012)
11. Criminisi, A., Shotton, J., Konukoglu, E.: Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Foundations and Trends in Computer Graphics and Vision 7(2-3), 81–227 (2012)
12. Dantone, M., Gall, J., Fanelli, G., Van Gool, L.: Real-time facial feature detection using conditional regression forests. In: Proc. CVPR (2012)
13. Dicle, C., Camps, O., Sznaier, M.: The way they move: Tracking multiple targets with similar appearance. In: Proc. ICCV (2013)
14. Dollár, P., Belongie, S., Perona, P.: The fastest pedestrian detector in the west. In: Proc. BMVC (2010)
15. Eshel, R., Moses, Y.: Tracking in a dense crowd using multiple cameras. IJCV 88(1), 129–143 (2010)
16. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. IJCV 88(2), 303–338 (2010)
17. Gall, J., Razavi, N., Van Gool, L.: On-line adaption of class-specific codebooks for instance tracking. In: Proc. BMVC (2010)
18. Gall, J., Razavi, N., Van Gool, L.: An introduction to random forests for multi-class object detection. In: Theoretical Foundations of Computer Vision (2011)
19. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.S.: Hough forests for object detection, tracking, and action recognition. IEEE Trans. PAMI 33(11), 2188–2202 (2011)
20. Girshick, R.B., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.W.: Efficient regression of general-activity human poses from depth images. In: Proc. ICCV (2011)
21. Godec, M., Roth, P.M., Bischof, H.: Hough-based tracking of non-rigid objects. Computer Vision and Image Understanding 117(10), 1245–1256 (2013)
22. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. PAMI 34(7), 1409–1422 (2012)

23. Khan, S.M., Shah, M.: Tracking multiple occluding people by localizing on multiple scene planes. IEEE Trans. PAMI 31(3), 505–519 (2009)
24. Küttel, D., Breitenstein, M.D., Van Gool, L., Ferrari, V.: What's going on? discovering spatio-temporal dependencies in dynamic scenes. In: Proc. CVPR (2010)
25. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. IJCV 77(1-3), 259–289 (2008)
26. Liu, J., Carr, P., Collins, R.T., Liu, Y.: Tracking sports players with context-conditioned motion models. In: Proc. CVPR (2013)
27. Okada, R.: Discriminative generalized hough transform for object dectection. In: Proc. ICCV (2009)
28. Özuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast keypoint recognition using random ferns. IEEE Trans. PAMI 32(3), 448–461 (2010)
29. Perlich, C., Provost, F.J., Simonoff, J.S.: Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research 4, 211–255 (2003)
30. Possegger, H., Sternig, S., Mauthner, T., Roth, P.M., Bischof, H.: Robust real-time tracking of multiple objects by volumetric mass densities. In: Proc. CVPR (2013)
31. Roth, P.M., Leistner, C., Berger, A., Bischof, H.: Multiple instance learning from multiple cameras. In: IEEE Workshop on Camera Networks (CVPR) (2010)
32. Schreiber, D., Cambrini, L., Biber, J., Sardy, B.: Online visual quality inspection for weld seams. Int'l Journal of Advanced Manufacturing Technology 42(5-6), 497–504 (2008)
33. Schulter, S., Leistner, C., Roth, P.M., Van Gool, L., Bischof, H.: On-line hough forests. In: Proc. BMVC (2011)
34. Shotton, J., Girshick, R.B., Fitzgibbon, A.W., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., Blake, A.: Efficient human pose estimation from single depth images. IEEE Trans. PAMI 35(12), 2821–2840 (2013)
35. Sternig, S., Mauthner, T., Irschara, A., Roth, P.M., Bischof, H.: Multi-camera multi-object tracking by robust hough-based homography projections. In: IEEE Workshop on Visual Surveillance (ICCV) (2011)
36. Tang, D., Liu, Y., Kim, T.K.: Fast pedestrian detection by cascaded random forest with dominant orientation templates. In: Proc. BMVC (2012)
37. Tang, D., Yu, T.H., Kim, T.K.: Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In: Proc. ICCV (2013)
38. Tang, S., Andriluka, M., Milan, A., Schindler, K., Roth, S., Schiele, B.: Learning people detectors for tracking in crowded scenes. In: Proc. ICCV (2013)
39. Wohlhart, P., Donoser, M., Roth, P.M., Bischof, H.: Detecting partially occluded objects with an implicit shape model random field. In: Proc. ACCV (2012)
40. Zamir, A.R., Dehghan, A., Shah, M.: GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs. In: Proc. ECCV (2012)