Adaptive Multi-cue 3D Tracking of Arbitrary Objects

Germán Martín García, Dominik A. Klein, Jörg Stückler, Simone Frintrop, Armin B. Cremers

Department of Computer Science III
Rheinische Friedrich-Wilhelms-Universität Bonn

Abstract. We present a general method for RGB-D data that is able to track arbitrary objects in real-time in challenging real-world scenarios. The method is based on the Condensation algorithm. The observation model consists of a target/background classifier that is boosted from a pool of grayscale, color, and depth features. The training set of the observation model is updated with new examples from tracking and the classifier is re-trained to cope with the new appearances of the target. A mechanism maintains a small set of specialized candidate features in the pool, thus decreasing the computational time, while keeping the performance stable. Depth measurements are integrated into the prediction of the 3D state of the particles. We evaluate our approach with a new benchmark for RGB-D tracking algorithms; the results prove our method to be robust under real-world settings, being able to keep track of the targets over 96% of the time.

1 Introduction

Visual object tracking is the task of estimating the state of a target of interest among consecutive images. Its applications are well known and among them we find automatic surveillance [18], sports events video analysis [11], or autonomous navigation [4]; in the robotics community it is also a key ability in tasks such as visual servoing [12]. Visual tracking is a difficult task, since trackers usually have to deal with the problem that the target's appearance and dimensions change constantly, illumination varies in the scene, etc. A recent survey [23] gives a good overview and taxonomy of solutions. More recently, visual tracking has also been surveyed from the point of view of the robotics community in chapter 3 of [17].

Some of the proposed solutions are focused on a specific task or a type of object. Wu and Nevatia [22] proposed a system that is able to detect parts of humans independently and merge them to form solid hypotheses and thus deal with partial occlusions. Giebel et al. [8] use a set of training data to learn a representation of the object in the form of Dynamic Point Distribution Models before the tracking starts. Three visual cues, shape, texture, and stereo disparity measurements are integrated in the observation model of a particle filter. Our approach to the tracking problem, however, is not specific to a task, nor to any specific type of object.

When no previous knowledge of the object is available, the representation of the target needs to be updated to be able to cope with new appearances. The adaptation of the model brings along the drifting problem: the tracker can adapt to an object that is not the target. Using several cues is a common way to improve the robustness of the tracker. From the field of visual attention, a component-based tracker was introduced [7]; high contrast components in intensity and color channels are found and integrated in a descriptor. This descriptor is then used as part of the observation model of a particle filter. Recently, the tracking-by-detection paradigm has attracted more attention. Santner et al. [20] propose a combination of three trackers that adapt at increasing rates: template-based, on-line random forest, and optical flow. To solve the drifting problem, the trackers are disposed on a cascade so that updates can be inhibited. Avidan [2] treats tracking as a binary classification problem. An ensemble of weak classifiers is calculated using AdaBoost and is adapted to new appearances of the object. Grabner et al. [9] also used Adaboost to form a strong classifier of weak classifiers that are trained on-line with incoming frames. They propose a feature pool update mechanism where the worst weak classifiers are removed and new ones are sampled. We followed this idea to keep a reduced set of candidate features as well as specialized to the object modalities. Very related is the work of Klein et al. [14, 15]: Haar-like [15] and Gradient features [14] are boosted into a strong classifier. It keeps a training set of examples that is updated over time, on which the classifier is re-trained to cope with new appearances of target and background. This adaptive observation model is integrated in a particle filter [13].

The present work follows that of Klein et al. [14, 15]. We make use of the Kinect sensor as a source of RGB and depth data instead of a monocular camera. Therefore, the variety of objects that can be tracked is only limited by the measurement principle of the sensor. To improve the robustness of the algorithm, we propose the use of several cues: not only grayscale but also color and depth features. To deal with the increasing size of the feature space, we define a mechanism that keeps its cardinality low, thus saving computational time, as well as specialized to the target appearance modalities: the feature space is now composed of three different kinds in number proportional to their discriminative power. Furthermore, depth information allows us to improve the accuracy of the predictions of the particle filter, through the definition of a 3D state space for the particles. To test the convenience of the proposed improvements we developed an RGB-D tracking benchmark, where the benefit of each of the contributions is evaluated with respect to the old approach. The success of the applicable enhancements is also evaluated in two existing RGB benchmarks and compared with other state of the art algorithms.

2 Adaptive Tracking

The tracking system is based on the Condensation algorithm [13]: a set of N weighted particles $S_t = \{\pi_i, x_i\}_{i=1}^{N}$, where π_i is the weight and x_i is the state of the i^{th} particle, approximates the conditional pdf of the state given the mea-

surements $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$ at time t, and thus, estimates the location and appearance of the target over time. In the original tracker, the state space of the particles is two-dimensional: $\boldsymbol{x}_{\mathbf{2d},\boldsymbol{t}} = \{p_u, p_v, \dot{p}_u, \dot{p}_v, w, h\}$. Particles have position p_u, p_v, velocity $\dot{p}_u, \dot{p}_v$, and dimensions w, h defined in the image plane. The observation model is constructed by boosting a strong classifier of center surround Haar-like [15] or grayscale gradient features [14]. The feature space, or feature pool, consists of features of different sizes and positions that cover the entire object sub-window. The Gentle Boost algorithm [6] builds a strong classifier c as the combination of M features/classifiers weak that best discriminates the target from the background: $c(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} \text{weak}_i(\boldsymbol{x})$. A key aspect of the tracker is that the observation model is continuously adapted to the new appearances of the target and background. At every frame, to prevent from drifting, if the confidence on the estimate is high enough and the particles have enough overlap, the update is performed: the target estimate of the particle with the highest weight is added to the training set as a new positive example, and new negative examples are randomly drawn from the current background; finally the classifier is re-trained on the new training set. The weights of the particles are set by exponentiating the result of the classifier; more details can be found in [15, 14].

2.1 3D State Space

As opposed to the previous approach, where the particles "live" in the image plane, we define the particles' state space by the position $\boldsymbol{r}_t$ and velocity $\boldsymbol{v}_t$ of a bounding cylinder around the object in 3D world coordinates: $\boldsymbol{x}_{\mathbf{3d},\boldsymbol{t}} = \{\boldsymbol{r}_t, \boldsymbol{v}_t\}$. The pose of the cylinder is assumed to be upright with respect to the camera axis. Its height H and diameter D are fixed in the initialization of the tracking process. This is different from the previous approach, where the width and height were adapted. A first order autoregressive motion model is applied to the particles. Every time step, the positions are updated with the velocities and the value of the time interval. The velocities are also updated by adding Gaussian white noise with a variance of $550\,\mathrm{mm}^2/\mathrm{s}^2$. Equation 1 summarizes this step:

$$\boldsymbol{r}'_t = \boldsymbol{r}_{t-1} + \boldsymbol{v}_{t-1}\,\Delta t\,, \qquad \boldsymbol{v}'_t = \boldsymbol{v}_{t-1} + \mathcal{N}(0;\Sigma)\,, \tag{1}$$

The state of the particles is now corrected by incorporating the depth information of the sensor. At the predicted position of the particle $\boldsymbol{r}'_t$, its projection onto the image plane is found. The projection is a rectangle in the image plane; the average of depth measurements is computed in an inner rectangle defined by 10% of the projection's dimensions. This depth is used to correct the particles' positions as well as the speeds, Equation 2.

$$\boldsymbol{r}_t = \phi(\boldsymbol{r}'_t)\,, \qquad \boldsymbol{v}_t = \boldsymbol{v}'_t + \frac{\boldsymbol{r}_t - \boldsymbol{r}'_t}{\Delta t}\,. \tag{2}$$

where ϕ finds the corrected position as discussed above. By making this correction we ensure that the particles' positions are restricted to existing points

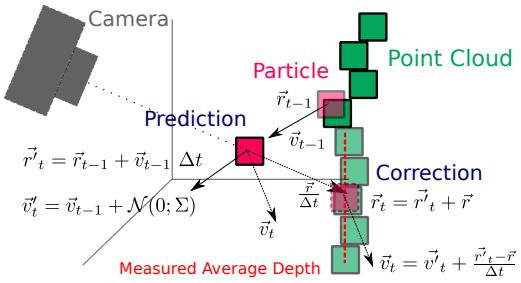

Fig. 1. At time $t-1$, a particle at position $\boldsymbol{r}_{t-1}$ moving with velocity $\boldsymbol{v}_{t-1}$. Based on the motion model, the predicted position is $\boldsymbol{r}'_t$ and the velocity $\boldsymbol{v}'_t$. They are corrected according to the sensor data giving $\boldsymbol{r}_t$ and $\boldsymbol{v}_t$.

in the point cloud. At the corrected position we project the particle onto the image plane to apply the observation model. Since the bounding cylinder is assumed to be upright with respect to the camera axis, its image projection is a rectangle. The process of prediction and correction is represented in Figure 1.

As a side effect of using a 3D state of the particles, the generation of negative examples for the observation model improves. Having the dimensions of the target in 3D, we are able to determine the size that a negative example should have at a given position in the image with a given depth measurement. This improves the accuracy of the observation model, since we are generating negative examples of the sizes that the target would have in the image if it occupied that position in 3D space.

2.2 Feature Kinds

In [14], the features are intensity gradients defined over rectangular regions. The features are computed in constant time with the help of integral images [21]. The weak classifier consists of a central bin that captures weak gradients and outer bins that incorporate strong gradients. It interpolates the gradient feature responses to the training examples, to the closest bins according to their orientation and magnitude. The gradient magnitude multiplied by the weight coming from the boosting algorithm is stored. When binning is finished, a log-normal distribution is fitted over the training examples. The prediction of a weak classifier is given by the ratio of maximum-likelihood estimators for the positive and negative example distributions: $\frac{p}{p+n}$. For further details about the weak classifier one can refer to the original paper [14]. Here, we extend the previous work based on gray value intensity gradients work by additionally using depth gradients and color averages. They are represented in Figure 2.

Depth Gradient Features The grayscale scalable gradient features of [14] can be applied to the depth layer. In the classifier, depicted in Figure 3, the thresholds that define the center and outer bins need to be adjusted to a meaningful value in the depth layer. We considered gradient values below 5 cm to

Fig. 2. Left: depth gradient features. Right: color average features.

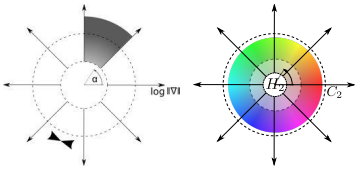

Fig. 3. Left: depth feature classifier. Right: color feature classifier.

correspond entirely to the center bin. Values between 5 and 20 cm are interpolated between center and outer bins, and those stronger than 20 cm fall entirely into the outer bins.

Color Features Color averages are measured over rectangular regions in the image. An approximation of the HSV color space [10] is used to allow for real-time processing. The V value, corresponding to brightness, is here discarded since it is already captured by the grayscale features. The feature computation happens as follows. Given the RGB image, the HSV Cartesian representation approximation, given by $\alpha = \frac{1}{2}(2R - G - B)$ and $\beta = \frac{\sqrt{3}}{2}(G - B)$, is calculated.

Integral images [21] let us compute α and β averages over feature regions in constant time. The Cartesian averages can then be transformed to the polar representation of $H_2 = atan2(\beta, \alpha)$ and $C_2 = \sqrt{(\alpha^2 + \beta^2)}$. The color feature result is formed by the tuple (H_2, C_2). The features compute color averages represented by two-dimensional vectors. To classify them, we use the same strategy of binning and regression of the gradient features classifier. The inner and outer bin thresholds were adjusted empirically to 1 and 3 respectively.

2.3 Feature Sampling and Dynamic Feature Pool

We define two mechanisms to reduce the number of candidate features in the pool, as well as to let them adapt to the object modalities. First, for each feature kind, 50 dynamic features are added to the pool. For the color ones, the positions are drawn from a truncated Gaussian distribution in the interval $[0, 1] \times [0, 1]$ with mean at $(0.5, 0.5)$, the center of the object window, and covariance a diagonal matrix with elements 0.15^2. This is done to position the color features closer to the center of the object window where the target is more likely to be contained. The depth and grayscale gradient features' positions are uniformly randomly sampled. Additionally, the pool contains a small fixed set of equally distributed static features[1]. Second, to adapt to the object modalities the dynamic feature pool set is updated concurrently with the observation model: at each iteration one feature kind is chosen for removal and one for addition; this decision is taken

[1] Features are here three times less densely sampled as in the previous approach [14], the actual number depends on the size of the object in the image. The ratio of dynamic:static features was between 1:1.5 and 1:2 in the tests performed.

proportionally to the mean training error of the feature kinds. Of the kind chosen for removal, the feature with the highest error is removed; of the kind chosen for addition, a new feature is sampled using the strategy defined before. The pool update does not affect the fixed set of static features.

3 Evaluation

The proposed enhancements are fully evaluated in a new RGB-D benchmark, and on two available RGB benchmarks, namely BoBoT [16] and PROST [20]. The metrics used, as in [15], are 1) the overlap between the estimated target and the ground truth, and 2) the hit rate; a hit happens when the overlap in one frame is bigger than $\frac{1}{3}$. In each sequence of the benchmarks, we run our algorithm ten times and compute the average metrics.

3.1 RGB Benchmarks

On the following two benchmarks, we evaluate the performance of the color features and the new feature pool mechanisms. The same parameters were used throughout the tests, namely: 2000 particles, 32 classifiers and 140 training examples. These parameters let the tracker run at a rate of 25 fps[2]. Four sets of tests were performed; in all of them the 2D state space was used:

- Test 1.1: grayscale and color features; new sampling approach, dynamic pool behavior enabled
- Test 1.2: grayscale and color features; new sampling approach, dynamic pool behavior disabled
- Test 1.3: grayscale and color features; old feature sampling approach
- Test 1.4 (as in [14] approach):grayscale features; old feature sampling approach

BoBoT Benchmark The BoBoT benchmark [16] contains thirteen RGB sequences suitable for evaluating tracking algorithms. As it was shown in [15], the algorithm of [15] proved to be superior to the particle filter based approaches of [19] and [7]. The results, in Table 1, show that the use of color features together with the new feature pool improves both overlap and hit rates without significant additional cost in computational time. In sequences A and Jb the use of color features gives a considerable benefit.

PROST Benchmark PROST [20] introduced an RGB benchmark of four sequences and compared their results with other state-of-the-art tracking approaches. The metrics used are the percentage of frames correctly tracked (based on the PASCAL score [5]) and the mean distance error to the ground truth. The results, in Table 2, show that our approach is on average better than three other state-of-the-art tracking algorithms.

[2] The experiments were executed on an Intel ® Xeon(R) CPU W3565 @ 3.20GHz x 4.

Table 1. BoBoT Benchmark Results: overlap and hit rate for test sequences A to L. The column *cpu* shows the average time in ms consumed for each frame. The best result is displayed in bold; the second best is underlined.

	A		B		C		D		E		F		G		
	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	
1.1	76.29	99.95	80.55	**100.0**	92.33	**100.0**	**81.07**	**100.0**	**87.26**	**100.0**	65.19	**91.85**	73.68	**100.0**	
1.2	75.06	**99.98**	80.22	**100.0**	91.85	**100.0**	80.48	**100.0**	87.08	**100.0**	**65.47**	91.74	72.60	**100.0**	
1.3	**76.78**	99.8	81.59	**100.0**	92.95	**100.0**	80.48	**100.0**	87.03	**100.0**	65.40	91.83	72.64	**100.0**	
1.4	62.09	94.0	**81.71**	**100.0**	**93.20**	**100.0**	79.94	**100.0**	**87.26**	**100.0**	65.00	91.78	**76.01**	**100.0**	

	H		I		Ja		Jb		K		L		Averages		
	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	cpu	Over.	Hit
1.1	**97.25**	**100.0**	86.96	96.12	83.75	**98.61**	80.20	96.92	**85.84**	**100.0**	**68.76**	**90.83**	**30.89**	80.00	**97.51**
1.2	96.52	**100.0**	87.91	96.58	82.74	98.53	**82.16**	**98.05**	85.20	**100.0**	62.86	78.85	31.13	**80.01**	97.44
1.3	96.12	**100.0**	87.99	**97.02**	82.71	98.53	78.14	97.46	85.37	**100.0**	48.33	62.58	37.80	78.70	94.93
1.4	95.80	**100.0**	**88.02**	97.01	**83.91**	**98.61**	62.38	80.75	85.37	**100.0**	57.36	84.66	31.12	77.74	94.37

Table 2. Pascal score and mean distance error for the PROST Benchmark [20]. The cpu column shows the average time in ms consumed for each frame. The best result is displayed in bold; the second best is underlined.

	board		box		lemming		liquor		Averages		
	pascal	distance	pascal	distance	pascal	distance	pascal	distance	cpu	pascal	distance
GRAD [14]	**94.3**	**14.7**	91.8	13.2	78.0	28.4	91.4	11.9		**88.9**	17.05
PROST	75.0	37.0	91.4	**12.1**	70.5	25.4	83.7	21.6	-	80.15	24.02
MILTrack [3]	67.9	51.2	24.5	104.6	**83.6**	14.9	20.6	165.5	-	49.15	84.05
FragTrack [1]	67.9	90.1	61.4	57.4	54.9	82.8	79.9	30.7	-	66.02	65.25
Test 1.1	81.25	26.78	89.54	12.95	77.38	**12.57**	95.20	8.57	36.65	85.84	15.22
Test 1.2	78.43	28.85	86.52	14.10	79.92	12.62	95.76	8.83	37.40	85.16	16.10
Test 1.3	78.03	22.88	**92.33**	12.15	83.21	12.71	**96.49**	**7.60**	42.02	87.52	**13.84**
Test 1.4 [5]	87.86	21.59	92.62	12.49	67.16	60.95	92.83	12.06	41.30	85.12	26.77

3.2 BoBoT-D Benchmark

The new benchmark consists of five RGB-D video sequences recorded with the Kinect sensor; it is also available at [16]. The ground truth data was manually labeled for each of them as the smallest rectangle that contains the target at each frame. Figure 4 shows a preview of the sequences. Sequence 1 shows a breakfast table. The target is a milk tetra-pack that is lifted, opened and from which some milk is poured into a coffee cup. This sequence attempts to test the performance of the algorithm on object rotations around the view point axis. In sequence 2, we find two persons passing a ball that is the target of the sequence. A radio control tank is the target of sequence 3; it moves around a scenario with batteries and a carton bridge. The next sequence contains a person walking down a corridor; the recording platform moves along while several people get in the way producing occlusions. The last of the sequences shows a white lunch box carried in front of an untextured background. The results are depicted in

[5] The results of the first line are those reported in [14]. They slightly differ to the ones we obtained with the current configuration in the last entry of the table (Test 1.4). Ours were obtained with a real-time configuration and the same set of parameters for arbitrary RGB sequences.

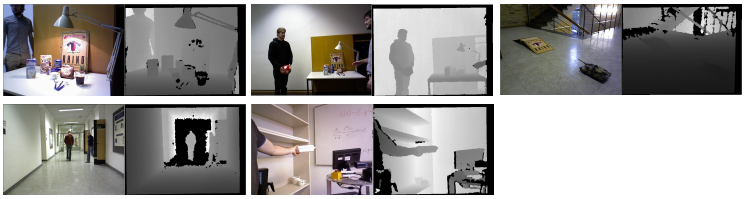

Fig. 4. BoBoT-D Benchmark: RGB and depth data for the five sequences: 'Milk', 'Ball', 'Tank', 'Person', and 'Lunch Box'.

Table 3; videos with the results are included as supplemental material. For the following experiments, 2000 particles and 23 classifiers were used:

- Test 2.1: 3D state space; three feature kinds; new sampling of features and pool update mechanism.
- Test 2.2: 3D state space; three feature kinds; new sampling of features but no update of the feature pool.
- Test 2.3: 3D state space; three feature kinds; old feature pool.
- Test 2.4: 2D state space; three feature kinds; old feature pool.
- Test 2.5: 2D state space, grayscale features and old feature pool.

Sequence 1 (Milk) The results were similar in the five experiments,with an overlap above 70% and hit rate around 95% in tests 2.1, 2.2, and 2.3.

Sequence 2 (Ball) The use of color features is crucial to the success of this experiment. This can be seen when comparing test 2.4 with 2.5: only by adding the new features, the hit rate raised from 19.07% to 83.68%. When the 3D model was used, the hit rate went up to over 95%.

Sequence 3 (Tank) The 3D state space played an important role in this sequence, with hit rates higher than 93%. The relatively low score of the overlap average, around 55% with the 3D model, can be explained by the fact that once the dimensions of the bounding cylinder are learned from the first frame they are never updated. The tank's appearance is learned from a side view in the first frame, and since the dimensions of the bounding cylinder are fixed, when it moves to a frontal view less overlap occurs.

Table 3. BoBoT-D Benchmark Results: overlap and hit rate for test sequences 'Milk', 'Ball', 'Tank', 'Person', and 'Lunch Box'. The cpu column shows the average time in ms consumed for each frame. The best result is displayed in bold; the second best is underlined.

	1 (Milk)		2 (Ball)		3 (Tank)		4 (Person)		5 (Box)		Averages		
	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	Over.	Hit	cpu	Over.	Hit
T. 2.1	73.47	**96.77**	**69.80**	96.91	**55.33**	94.09	70.67	95.32	73.10	99.81	30.69	68.47	**96.58**
T. 2.2	**74.61**	95.29	66.39	95.31	55.01	**94.27**	70.55	95.31	75.20	**100.00**	31.4	68.35	96.04
T. 2.3	73.21	96.39	68.63	**96.94**	54.46	93.80	**71.92**	95.17	**75.57**	**100.00**	33.52	**68.76**	96.46
T. 2.4	73.45	94.48	55.60	83.68	32.23	40.72	70.72	**95.92**	70.88	99.76	36.97	60.58	82.91
T. 2.5	69.01	89.14	14.08	19.07	28.92	32.30	67.07	91.70	47.42	70.66	**27.90**	45.3	60.57

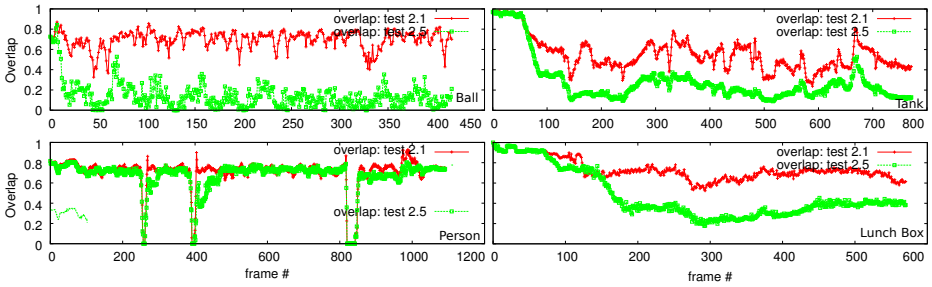

Fig. 5. Overlap plots in sequences 'Ball', 'Tank', 'Person', and 'Lunch Box'.

Sequence 4 (Person) The use of the 3D state space didn't result as crucial in this sequence because the size of the target remains almost constant. The occlusions occurring in this sequence can be observed in the three down peaks of the overlap plot of Figure 5.

Sequence 5 (Lunch Box) The first three experiments gave very close results. The depth features were essential in this sequence: comparing 2.4 and 2.5, there is a considerable difference in both metrics; and not that much between 2.4 and the tests with the 3D state space.

Figure 5 compares the overlap of the tracker with all the improvements against the old approach. The results show that the 3D state space and the two new feature kinds give a considerable improvement on the performance of the tracking algorithm. Both metrics, overlap and hit rate, are significantly higher as compared to the old approach (Test 2.5). As reflected in Test 1.1: the hit rate of 96.58% on average, shows that targets were very successfully tracked; the overlap average of 68.47% shows an adequate estimation of the position and dimensions of the target. In sequences 2 and 3, the old approach could not keep track of the target. The speed gain can be seen when comparing the results of tests 2.1 and 2.2 against 2.3; having similar overlap and hit rates, the time consumed differs on about 10% in favor of the reduced feature pool.

4 Conclusion

We have presented several improvements to the existing tracking algorithm of [14]. First, we use a Kinect sensor that provides RGB-D data. To achieve robustness we propose the use of several cues instead of one: intensity, color and depth. The precision of the predictions is increased by extending the particles' state space to 3D world coordinates, and by integrating depth measurements in it. We proposed a mechanism that reduces the size of the feature space and specializes the content to the object modalities, reducing execution time. We compared our approach with several other state of the art trackers in three different benchmarks. When depth measurements are available, the tracker is highly robust and precise, being able to track the targets over 96% of the frames in difficult real world scenarios.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR. pp. 798–805 (2006)
2. Avidan, S.: Ensemble tracking. IEEE Trans. Pattern Anal. Mach. Intell. 29, 494–501 (2007)
3. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR. pp. 983 –990 (2009)
4. Ess, A., Schindler, K., Leibe, B., Van Gool, L.: Object detection and tracking for autonomous navigation in dynamic environments. Int. J. Rob. Res. pp. 1707–1725 (2010)
5. Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vision pp. 303–338 (2010)
6. Freund, Y., Schapire, R.E.: Special invited paper. additive logistic regression: A statistical view of boosting: Discussion. The Annals of Statistics 28(2) (2000)
7. Frintrop, S., Koenigs, A., Hoeller, F., Schulz, D.: A component-based approach to visual person tracking from a mobile platform. Int. J. of Social Robotics pp. 4531–4536 (2010)
8. Giebel, J., Gavrila, D., Schnoerr, C.: A bayesian framework for multi-cue 3D object tracking. In: ECCV. pp. 241–252 (2004)
9. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. Proc BMVC pp. 6.1–6.10 (2006)
10. Hanbury, A.: Constructing cylindrical coordinate colour spaces. Pattern Recogn. Lett. 29(4), 494–500 (Mar 2008)
11. Hess, R., Fern, A.: Discriminatively trained particle filters for complex multi-object tracking. In: CVPR, 2009. pp. 240 –247. IEEE (2009)
12. Hutchinson, S., Hager, G., Corke, P.: A tutorial on visual servo control. Robotics and Automation, IEEE Transactions on pp. 651 –670 (1996)
13. Isard, M., Blake, A.: Condensation-conditional density propagation for visual tracking. Int. J. of Computer Vision 29, 5–28 (1998)
14. Klein, D.A., Cremers, A.B.: Boosting scalable gradient features for adaptive real-time tracking. In: ICRA. pp. 4411 –4416 (2011)
15. Klein, D.A., Schulz, D., Frintrop, S., Cremers, A.B.: Adaptive real-time video-tracking for arbitrary objects. In: IROS. pp. 772 –777 (2010)
16. Klein, D.A.: BoBoT - Bonn Benchmark on Tracking, `http://www.iai.uni-bonn.de/~kleind/tracking/index.htm`
17. Kragic, D., Vincze, M.: Vision for robotics. Foundations and Trends in Robotics (2009)
18. Liem, M., Gavrila, D.M.: Multi-person localization and track assignment in overlapping camera views. In: DAGM. pp. 173–183 (2011)
19. Prez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. In: Proc. ECCV. pp. 661–675 (2002)
20. Santner, J., Leistner, C., Saffari, A., Pock, T., Bischof, H.: Prost: Parallel robust online simple tracking. In: CVPR. pp. 723 –730 (2010)
21. Viola, P., Jones, M.: Robust real-time object detection. In: Int. J. of Computer Vision (2001)
22. Wu, B., Nevatia, R.: Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. Int. J. Comput. Vision pp. 247–266 (2007)
23. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. (2006)