# Tracking Revisited using RGBD Camera: Unified Benchmark and Baselines

Shuran Song Jianxiong Xiao
Princeton University

## Abstract

*Despite significant progress, tracking is still considered to be a very challenging task. Recently, the increasing popularity of depth sensors has made it possible to obtain reliable depth easily. This may be a game changer for tracking, since depth can be used to prevent model drift and handle occlusion. We also observe that current tracking algorithms are mostly evaluated on a very small number of videos collected and annotated by different groups. The lack of a reasonable size and consistently constructed benchmark has prevented a persuasive comparison among different algorithms. In this paper, we construct a unified benchmark dataset of 100 RGBD videos with high diversity, propose different kinds of RGBD tracking algorithms using 2D or 3D model, and present a quantitative comparison of various algorithms with RGB or RGBD input. We aim to lay the foundation for further research in both RGB and RGBD tracking, and our benchmark is available at http://tracking.cs.princeton.edu.*

## 1. Introduction

Visual object tracking is an important but challenging task. For example, in the standard tracking-by-detection pipeline [2], a slight offset in one frame may be reinforced after an online learning step, resulting in the so-called model drift problem. Besides, occlusion of target objects occurs quite often in real world scenarios. To address these issues, over the last decade, object tracking algorithms have evolved significantly in both their sophistication and quality of results. In particular, many new learning theories are introduced into tracking[1]. However, all these approaches are evaluated on a very small number of videos collected and annotated by different groups over the years (*e.g.* 8 videos used for evaluating [3, 12]). A consistent collection and annotation protocol is hard to guarantee, and the small size of the dataset induces significant bias [23]. There is no consistent evaluation metric, especially when occlusion happens. Furthermore, all ground truth annotation is publicly available, which makes it even worse in terms of parameter overfitting. Many practitioners in the field find that it is hard to generalize some of these approaches to other videos because of parameter sensitivity. The lack of a reasonable size and consistently constructed benchmark for tracking has been preventing persuasive comparisons.

Figure 1. Examples from our Princeton Tracking Benchmark.

Meanwhile, great popularity of affordable depth sensors, such as Microsoft Kinect, Asus Xtion and PrimeSense, make depth acquisition very easy. Reliable depth maps can provide valuable additional information to significantly improve tracking results with robust occlusion and model drift handling. How much does depth information help in tracking? Will the availability of depth significantly change the design of the standard tracking pipeline? What is a reasonable baseline algorithm for tracking with RGBD data? And how do the state-of-the-art RGB tracking algorithms perform compared with these new RGBD algorithms?

This paper seeks to answer these questions by conducting a quantitative benchmark evaluation, and proposing various simple but powerful baseline algorithms. To establish a unified benchmark, we construct a RGBD dataset of 100 videos, named as Princeton Tracking Benchmark (PTB), which includes deformable objects, various occlusion con-

[1] *e.g.* Multiple Instance Learning [3], Compressive Sensing [30], Kernelized Structured Support Vector Machine [12], Semi-supervised Boosting [11], Eigenbasis and Adaptive Particle Filter [18], Sparse Principal Component Analysis and Interactive Markov Chain Monte Carlo [16] *etc.*

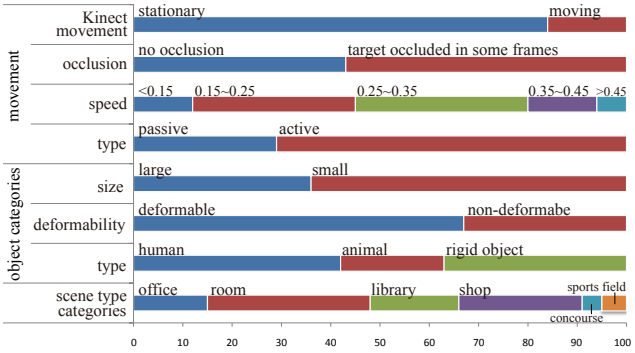



Figure 2. Statistics of our RGBD tracking benchmark dataset.

ditions, moving camera, and different scenes (Figure 1). To build a set of diverse baseline algorithms, we design several tracking algorithms incorporating depth information to reduce model drift, and propose a simple scheme for occlusion handling.

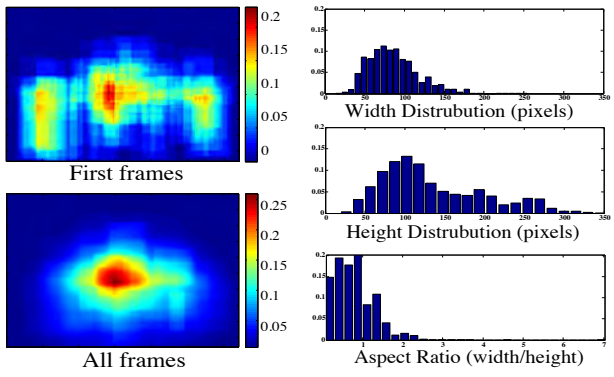
The goal of this paper is four-fold: 1. To construct one unified benchmark dataset with analysis of bias statistics; 2. To standardize uniform evaluation criteria for comparing different kinds of algorithms, including the protocol for occlusion evaluation; 3. To carefully design various kinds of baseline algorithms, including traditional 2D image patch based tracker, new 3D point cloud based tracker, low-level flow based tracker, and trivial algorithms without even using the video; 4. To open up new research direction for RGBD tracking, and provide basic insights by quantifying the importance of depth and 3D information.

In our PTB, We withhold ground truth for 95 videos, open source all baseline algorithms, and host an online evaluation server to allow new result submissions for comparison.

### 1.1. Related works

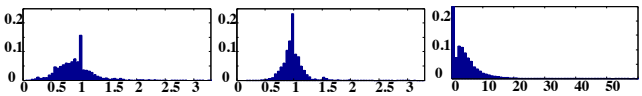
Many noteworthy tracking algorithms have been proposed in the last decade. Here we briefly summarize only a partial list of them, due to space constraints. [3] proposes a very robust system with online multiple instance learning. [15] designs a framework to integrate tracking, learning, and detection using P-N loops. [11] uses semi-supervised online boosting to increase tracking robustness, while [1] handles it using a fragments-based model. To address target appearance and motion changes, [16] uses visual tracking decomposition scheme to integrate multiple observation and motion trackers, and [18] presents an incremental subspace learning algorithm. More recently, [30] proposes using compressive sensing for real-time tracking, and [12] adopts structured output prediction to avoid intermediate classification.

In comparison, dataset as well as benchmarks for RGBD tracking evaluation are less comprehensive. The publicly available RGBD People Dataset [22, 17] contains only one sequence with 1,132 frames captured with static cameras




(a) Box location distribution. (b) Box size distribution.

Figure 3. Bounding box distribution over all sequences.



(a) Relative area to the first frame box. (b) Relative aspect ratio to the first frame box. (c) Distance between consecutive frames.

Figure 4. Bounding box variation over time.

with only people moving, which is obviously not enough to evaluate tracking algorithms for general objects. In concurrent work, Wu *et al*. [26] evaluate 29 2D tracking algorithms on 50 RGB videos combining from different sources captured and annotated in different settings. In contrast, our benchmark evaluates both RGB and RGBD tracking algorithms, together with our proposed 3D tracking algorithms, on 100 RGBD videos consistently captured and annotated by us. Furthermore, to separate the effect of various assumptions, we calculate upper bounds and lower bounds for the algorithms, and categorize error into three different types to analyze occlusion handling.

There have been several great benchmarks for various computer vision tasks that help to advance the field and shape computer vision as a rigorous experimental science, *e.g*. two-view stereo matching [19], multi-view stereo reconstruction [20], optical flow [4], image segmentation [8], and object classification, detection, and segmentation [9], scene classification [27] and our PTB is an addition to the list to provide a benchmark for tracking algorithms using either RGB or RGBD data.

Apart from tracking, various efforts have been made on constructing RGBD datasets for other computer vision tasks. In particular, the NYU dataset [21] is comprised of 1449 fully labeled RGBD images with unlabeled short videos. The SUN3D dataset [28] has a large collection of big spaces in 3D reconstructed from long RGBD videos, using structure from motion and object annotations. [13, 29] contain 3D bounding box annotations for cuboid-like objects in RGBD images.

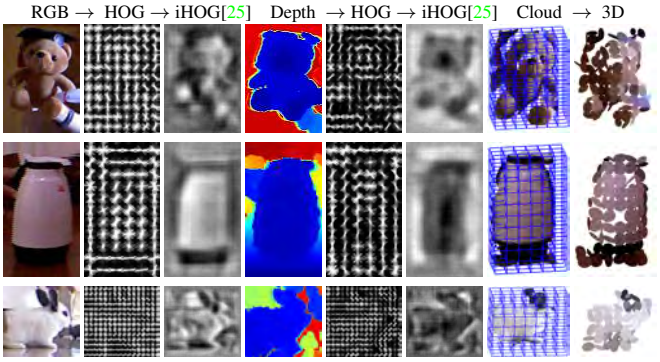



Figure 5. Visualization of features in the baseline algorithms. iHOG[25] is just to visualize the HOG feature and not used in the actual algorithms. Column 7 shows the cell division of point cloud. Column 8 visualizes the 3D feature, in which color of the ellipsoids is the average color of points in each cell, and the axes of the ellipsoids are the principal components of those points.

# 2. Unified tracking benchmark

## 2.1. Dataset construction

To construct one unified benchmark dataset for different kinds of tracking algorithms, we recorded 100 video clips with both RGB and depth data, and manually annotated ground truth bounding boxes.

**Hardware setup** Our dataset is captured using a standard Microsoft Kinect 1.0. It uses a paired infrared projector and camera to calculate depth value, thus its performance is severely impaired in outdoor environment under direct sunlight. Also, Kinect requires a minimum and a maximum distance from objects to the cameras in order to obtain accurate depth values. Due to the above constraints, our videos are captured indoors, with object depth values ranging from 0.5 to 10 meters.

**Annotation** We manually annotate the ground truth (the target location) of the dataset by drawing a bounding box on each frame as follows: A minimum bounding box covering the target is initialized on the first frame. In the next frame, if the target moves or its shape changes, the bounding box will be adjusted accordingly; otherwise, it remains the same. All frames are manually annotated by an author to ensure high consistency. Because we manually annotate each frame, there is no interpolation or key frames. When occlusion occurs, the ground truth is defined as the minimum bounding box covering only the visible portion of the target. When the target is completely occluded there will be no bounding box for this frame. We annotate all following frames in this way.

## 2.2. Dataset statistics

As a benchmark dataset, high diversity and low bias are important. Figure 2, 3, 4 summarize the statistics of our dataset, which presents varieties in the following aspects:

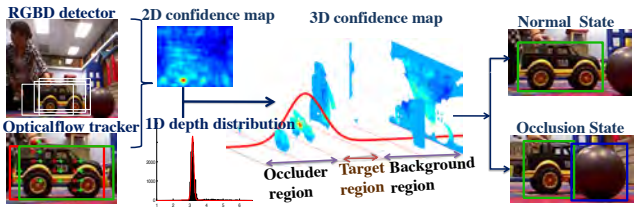



Figure 6. RGBD tracking algorithm based on 2D image patch. The 2D confidence map shows the combined confidence from detector and optical flow tracker. The 1D depth distribution is a Gaussian estimated from target depth histogram. The 3D confidence map is computed by applying a threshold from the 1D Gaussian on the 2D confidence map. In the output, the target location (the green bounding boxes) is the position of the highest confidence. Occluder (the blue bounding box) is recognized from its depth value.

**Target type** We divide targets into three types: human, animal and relatively rigid object. Rigid objects, such as toys and human faces, can only translate or rotate. Animals include dogs, rabbits and turtles, whose movement usually consists of out-of-plane rotation and some deformation. The degree of freedom for human body motion is very high, which may increase the difficulty in tracking.

**Scene type** Each scene type in our dataset has a different level of background clutter. The living room, for example, has a simple and mostly static background, while the background of a cafe is complex, with many people passing by.

**Presence of occlusion** Our videos cover several aspects of occlusion, *e.g.* how long the target is occluded, whether the target moves or undergoes appearance change during occlusion, and similarity between the occluder and target.

**Bounding box distribution over all sequences** Figure 3 shows the location and size distribution of ground truth boxes across all sequences. The location distribution is computed as a normalized histogram in $640 \times 480$ image space, where value on each pixel represents the possibility for a bounding box to cover that pixel. The box size distribution is also over all sequences, which shows our dataset covers long and short, wide and narrow objects.

**Bounding box variation over time** Apart from overall statistics, we also provide average bounding box statistics within a single video sequence. For each sequence, we compute a histogram of relative area and aspect ratio of the ground truth bounding boxes to the one in the first frame, as well as box center distance between consecutive frames. Afterwards, the histogram is normalized and averaged over all sequences. The resultant average histogram is shown in Figure 4, which illustrates how much bounding boxes may deform or shift in a single video.

## 2.3. Evaluation metric

We use two metrics to evaluate the performance. The first one is center position error (CPE), which is the Euclidean distance between centers of output bounding boxes and the ground truth. This metric shows how close the

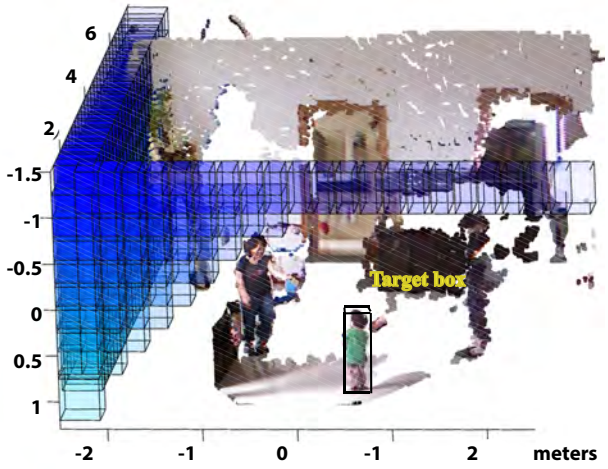



Figure 7. 3D point cloud with 3D sliding window detection.

tracking results are to the ground truth in each frame. However, the overall performance of trackers cannot be measured by averaging this distance, especially when trackers are misled by background clutter and produce faraway outliers. Besides, this distance is undefined when trackers fail to output a bounding box or there is no ground truth bounding box (the target is totally occluded).

To evaluate the overall performance, we employ the criterion used in the PASCAL VOC challenge [9], the ratio of overlap $r_i$ between the outputs and true bounding boxes:

$$r_i = \begin{cases} \frac{\text{area}(\text{ROI}_{T_i} \cap \text{ROI}_{G_i})}{\text{area}(\text{ROI}_{T_i} \cup \text{ROI}_{G_i})} & \text{if both ROI}_{T_i} \text{ and ROI}_{G_i} \text{ exist} \\ 1 & \text{if neither ROI}_{T_i} \text{ and ROI}_{G_i} \text{ exist} \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where $\text{ROI}_{T_i}$ is the target bounding box in the $i$-th frame and $\text{ROI}_{G_i}$ is the ground truth bounding box. By setting a minimum overlapping area $r_t$, we can calculate the average success rate $R$ of each tracker as follows:

$$R = \frac{1}{N}\sum_{i=1}^{N} u_i, \quad \text{where} \quad u_i = \begin{cases} 1 & \text{if } r_i > r_t \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

where $u_i$ is an indicator denoting whether the output bounding box of the $i$-th frame is acceptable, $N$ is the number of frames, and $r_t$ is the minimum overlap ratio deciding whether an output is correct. Since some trackers may produce outputs that have small overlap ratio over all frames while others give large overlap on some frames and fail completely on the rest, $r_t$ must be treated as a variable to conduct a fair comparison. Furthermore, we can divide tracking failures into three types:

Type I :$\text{ROI}_{T_i} \neq null$ and $\text{ROI}_{G_i} \neq null$ and $r_i < r_t$

Type II :$\text{ROI}_{T_i} \neq null$ and $\text{ROI}_{G_i} = null$

Type III :$\text{ROI}_{T_i} = null$ and $\text{ROI}_{G_i} \neq null$

Type I error occurs when the target is visible, but the tracker's output is far away from the target. Type II error occurs when the target is invisible but tracker outputs a bounding box. Type III error occurs when the target is visible but the tracker fails to give any output.

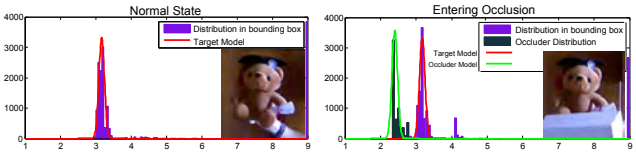



Figure 8. Depth distribution inside target bounding box. Left: distribution in normal state. Right: distribution when occlusion occurs. The red Gaussians denote the target model, and the green Gaussian denotes the occluder model.

## 3. Baseline algorithms

Now that depth data is available, we design two types of approaches to utilize it. The first one adopts traditional 2D image patch tracking with additional depth features; the second one is based on 3D point cloud and outputs 3D target bounding box in space, which is a more natural way of handling 3D data. An occlusion handling mechanism is also added to both of the systems. Overview of the two algorithms is shown in Figure 6 and 7 respectively. Here are basic building blocks of the two types of approaches.

### 3.1. Detection based tracking

Tracking by detection is done by building a discriminative model of the tracking target, and using it to classify potential targets in subsequent frames. Candidate with highest confidence is regarded as the tracking result. Following paragraphs describe two features we used to build the discriminative model, and the procedure of classification using support vector machine (SVM [6])

**RGBD HOG feature** RGBD HOG feature is the histogram of oriented gradients (HOG[7, 10]) computed from both RGB and depth images (Figure 5 columns 1 to 6). HOG for depth is obtained by treating depth data as a grayscale image. This RGBD HOG feature describes local color textures as well as 3D shapes, improving robustness against target illumination variation, lack of texture and similarity in color to the background.

**Point cloud feature** Point cloud feature is designed to capture the color and shape of cells of 3D points. We first divide 3D space into cubic cells, then for each cell we compute three features: (1) a color histogram in color names space [24], (2) the number of points in the cell and (3) 3D shape feature [14]. Color name is a linguistic label assigned to color spectrum space. The color name feature is computed by first mapping the RGB value of each point to one of the 11 basic color names and then building a histogram of these color names within each cell. The 3D shape feature is composed of scatter-ness, linear-ness and surface-ness of the point distribution inside each cell, obtained from its principal components: Denote the eigenvalues of the covariance matrix of point coordinates as $\lambda_1 > \lambda_2 > \lambda_3$, then

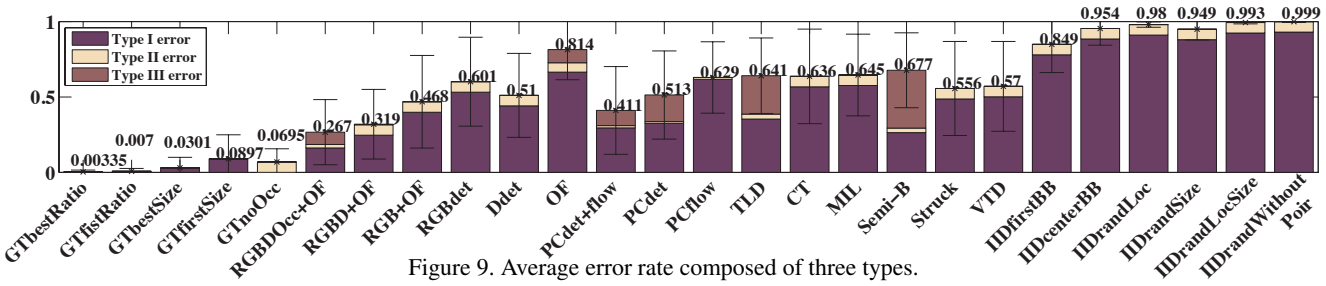


Figure 9. Average error rate composed of three types.

scatter-ness is $\lambda_1$, linear-ness is $\lambda_1 - \lambda_2$, surface-ness is $\lambda_2 - \lambda_3$. Figure 5 column 7 and 8 visualize the cell division of point cloud and its corresponding 3D feature.

**SVM training and detection** Both RGBD HOG and point cloud detectors train a linear SVM classifier. In the first frame, the SVM is trained by using user's input bounding box as the positive example and randomly picked bounding boxes that do not overlap with the target as negative examples. In the subsequent frame, computed features are convolved with SVM weights, and several possible target locations whose confidence is high are returned. Point cloud detector preforms 3D convolution for each feature dimension and then sums them up. Afterwards, the SVM is retrained during non-occlusion state using the resulting bounding box and the positive support vectors in the previous frames with hard negative mining.

### 3.2. Point tracking

**2D optical flow tracking** The 2D optical flow tracker adopts large displacement optical flow [5] on RGB data from consecutive frames, then generates the bounding box of points validated by forward-backward checking.

**3D iterative closest point tracking** For 3D tracking, we adopt Iterative Closest Point (ICP) algorithm [31], which iteratively computes a rigid transformation that minimize the sum of mean square error between two set of points in real time. The rigidity assumption holds in most cases, but it might fail when the target deforms and produces a large error $E(R, t, s)$. In this case, according to the small motion assumption, our tracker looks for the target near its previous position. We treat the biggest connected component in the neighborhood of the previous position as the new target position, and return a 3D bounding box that encompasses it.

### 3.3. Integration of detection and point tracking

Both detection and point tracking are initialized by the input bounding box in the first frame and updated online. In each frame, they run independently, and after their results are available, the confidence of detection result is adjusted as: $c = c_d + \alpha r_{(t,d)}$, where $c_d$ is the confidence of the detection, and $r_{(t,d)}$ is the overlap ratio between the detection and point tracker's resulting bounding boxes, *i.e.* an indication of their consistency. $\alpha$ denotes the weight of the overlap ratio ($\alpha = 0.5$ in our experiment). After thresholding on the adjusted confidence, the bounding box with highest confidence, if exists, is passed to occlusion checking, then output as the final result when occlusion is not detected. Under occlusion, the tracker outputs the best valid detection or segmentation result, as described in the section below.

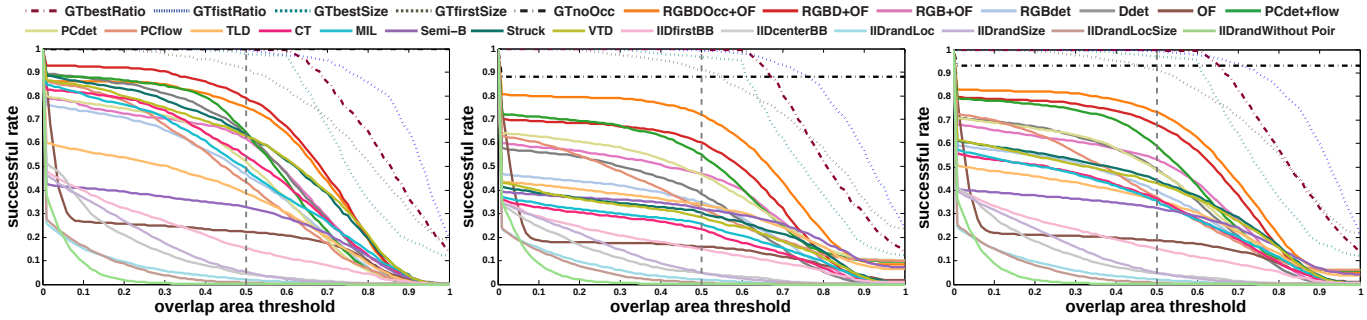
### 3.4. Occlusion handling

To handle occlusion, some traditional RGB trackers like [15] use forward-backward error to indicate tracking failure, and some like [1, 3] use a fragment-based model to reduce sensitivity to partial occlusion. However, with depth information the solution becomes more straight-forward. Here we propose a simple yet effective occlusion handling mechanism which actively detects occlusion and recovery.

**Occlusion detection** Occlusion handling is based on 2D bounding boxes (3D bounding boxes are projected back to 2D space). To detect the occlusion, we assume that the target is the closest object that dominates the bounding box when not occluded. A new object in front of the target inside the bounding box indicates the beginning of occlusion state. Therefore, depth histogram inside bounding box is expected to have a newly rising peak with a smaller depth value than target, and/or a reduction in the size of bins around the target depth, as illustrated in Figure 8.

In the $i$-th frame, the depth histogram $h_i$ of all pixels inside a bounding box can be approximated as a Gaussian distribution: $h_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. We define the likelihood of occlusion in this frame as: $O_i = \sum_{d=0}^{\mu_i - \sigma_i} h_i(d) / \sum_d h_i(d)$, where $h_i(d)$ is the value of the $d$-th bin in the $i$-th frame, and $d = 0$ is the depth of the camera. $\mu_i - \sigma_i$ is a threshold for a point to be considered as occluder. The number of pixels in the bounding box that have smaller depth value than target depth are considered the area of the occluder. Hence, a larger $O_i$ indicates that an occlusion is more likely. The search also includes the neighborhood of the target bounding box, which has a size of 0.1 times of the bounding box size. The target depth value is updated online, so a target moving towards the camera will not be treated as an occlusion.

**Recovery from occlusion** The occluder's model, *i.e.* its depth and color distribution, is initialized when entering the occlusion state, and its position is updated by an optical flow tracker. A list of possible target candidates are identified either by the detection or a local search around the oc-




(a) Test cases without occlusion (b) Test cases with occlusion (c) All test cases

Figure 10. Average success rate vs. threshold of overlap ratio ($r_t$) evaluated on different categories of test cases.

cluder. With depth and color distributions of target and occluder, the local search is done by performing segmentation on RGB and depth data respectively and combining their results. The combined segmentation produces a list of target candidates, whose validity is then judged by the SVM classifier. By examining the list of possible target candidates, the tracker interprets target recovery when at least one candidate's score evaluated by the SVM classifier is high, and its visible area is large enough compared to the target area before entering occlusion. The occlusion subroutine ends if the target is recovered from occlusion.

## 4. Evaluation

In our evaluation system, 5 videos out of 100 are used for parameter tuning, and the remaining 95 are used for evaluation. 3D bounding boxes are projected back to 2D for evaluation. To understand how much depth information improves the performance and evaluate the contribution of each building blocks, we tested nine variations of our proposed RGBD tracker listed in Table 1. We also evaluated six state-of-the-art RGB trackers: TLD[15], CT[30], MIL[3], semi-B[11], Struck[12] and VTD[16], since currently there is no other public available RGBD tracking algorithms for general objects. To understand the impact of model assumptions, we use the ground truth to design several performance upper bounds under different model assumptions, such as fixed box size, aspect ratio, or target being always visible (Table 2). To understand the impact of dataset bias to the evaluation, we also obtained performance lower bounds from several trivial image-independent algorithms (Table 3).

The performance measured by CPE and the corresponding snapshots are shown in Figure 11. The success rates measured by overlap ratio are shown in Figure 10. Error decomposition of each tracker is shown in Figure 9. Furthermore, in Table 4 we compute an average ranking of algorithms by averaging the individual rankings under different categorizations.

The effect of using depth data can be seen by comparing the tracker with depth input (RGBD+OF) and without (RGB+OF). With depth data, error is reduced by 14.9%. After enabling the occlusion handler, the tracker (RGB-DOcc+OF) error rate further decreases by 5.2%. The point cloud based tracker (PC) also achieves at least a 5.7% reduction in error rate when compared with other RGB trackers.

Table 1. Strong RGBD baseline algorithms

| | |
|---|---|
| OF | Uses only optical flow tracking. |
| Ddet | Uses Depth HOG detection tracking. |
| RGBdet | Uses RGB HOG detection tracking. |
| RGB +OF | Uses RGB HOG detection with optical flow. |
| RGBD +OF | Uses RGBD HOG detection with optical flow. |
| RGBDOcc +OF | Uses RGBD HOG detection and optical flow with occlusion handling. |
| PCflow | Uses 3D point tracker. |
| PCdet | Uses point cloud detection. |
| PC(det+flow) | Uses point cloud detection and 3D point tracking with occlusion handling. |

Table 2. Performance upper-bounds (GT:ground truth)

| | |
|---|---|
| GTfirstSize | Uses the GT location and first frame box size. |
| GTbestsize | Uses the GT location and fixed box size that optimize the successful rate. |
| GTfirstRatio | Uses the GT location and first frame box aspect ratio. |
| GTbestRatio | Uses the GT location and fixed box aspect ratio that optimize the successful rate. |
| GTnoOcc | Outputs the GT box, if exists, and a random box otherwise. |

Table 3. Performance lower-bound algorithms

| | |
|---|---|
| IIDfirstBB | Always outputs the first frame bounding box for all frames. |
| IIDcenterBB | Always outputs the box locate at center of image, with first frame box size. |
| IIDrandSize | Outputs bounding boxes with the first frame box location and a random size based on dataset statistics. |
| IIDrandLoc | Outputs bounding boxes with the first frame box size and a random location based on dataset statistics |
| IIDrandLoc Size | Outputs bounding boxes with random location and size based on dataset statistics |
| IIDrand WithoutPrior | Outputs bounding boxes with random location and size without any prior knowledge of dataset |

Figure 10 measures the success rate $R$ while varying the threshold $r_t$ in equation 2. A reasonable algorithm should have a curve lying between the lower bound and the upper bound under the same assumption. Figure 9 distinguishes sources of error to achieve a fair comparison and analysis between different algorithms. For some algorithms, performance difference may be partially attributed to the assumptions they are based on, which disagree with our dataset. For example, MIL, CT, Struck, and TVD do not have an occlusion handling mechanism which naturally lead to high

Table 4. Evaluation results: successful rate (SR) % and corresponding rankings (in parentheses) under different categorizations.

| algorithm | avg. rank | all SR | target type | | | target size | | movement | | occlussion | | motion type | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | human | animal | rigid | large | small | slow | fast | yes | no | active | passive |
| GTbestRatio | - | 99.665 | 99.9(1) | 100.0(1) | 99.3(1) | 99.8(1) | 99.6(1) | 99.5(1) | 99.7(1) | 99.6(1) | 99.7(2) | 99.7(1) | 99.5(1) |
| GTfirstRatio | - | 99.3 | 99.6(2) | 99.8(2) | 98.6(2) | 99.4(2) | 99.2(2) | 99.0(2) | 99.4(2) | 99.2(2) | 99.5(3) | 99.4(2) | 99.0(2) |
| GTbestSize | - | 97 | 95.0(3) | 99.3(3) | 98.1(3) | 96.2(3) | 97.6(3) | 98.4(3) | 96.5(3) | 96.4(3) | 97.8(4) | 96.3(3) | 98.8(3) |
| GTonOcc | - | 93.05 | 91.0(4) | 94.3(4) | 94.8(5) | 92.6(4) | 93.4(4) | 94.3(4) | 92.5(4) | 88.0(5) | 100.0(1) | 93.2(4) | 92.5(5) |
| GTfirstSize | - | 91.03 | 86.7(5) | 90.3(5) | 96.5(4) | 88.7(5) | 92.8(5) | 93.2(5) | 90.1(5) | 90.8(4) | 91.3(5) | 88.6(5) | 97.5(4) |
| RGBDOcc+OF | 1.18 | 73.3 | **74.0(1)** | 62.6(2) | **78.4(1)** | **78.1(1)** | **69.7(1)** | **76.3(1)** | **72.2(1)** | **72.0(1)** | 75.2(2) | **70.0(1)** | **82.3(1)** |
| RGBD+OF | 1.91 | 68.1 | 63.9(2) | **65.3(1)** | 74.5(2) | 71.5(2) | 65.5(2) | 73.4(2) | 65.9(2) | 60.1(2) | **79.0(1)** | 65.8(2) | 74.0(3) |
| PC(det+flow) | 3.09 | 58.9 | 50.5(3) | 51.6(3) | 72.7(3) | 63.4(3) | 55.5(4) | 73.9(2) | 53.0(3) | 55.0(3) | 64.4(5) | 75.5(2) | 52.7(3) |
| RGB+OF | 4.82 | 53.2 | 47.1(4) | 47.0(6) | 63.6(4) | 47.4(6) | 57.5(3) | 56.7(6) | 51.8(4) | 46.9(4) | 61.9(7) | 63.4(5) | 49.3(4) |
| Ddet | 5.73 | 49 | 43.3(5) | 48.3(5) | 55.9(6) | 47.2(5) | 50.3(5) | 52.7(8) | 47.5(5) | 38.4(6) | 63.5(4) | 54.3(9) | 46.9(5) |
| PCdet | 6 | 48.7 | 40.6(6) | 42.1(9) | 61.7(5) | 55.4(4) | 43.6(8) | 58.5(4) | 44.8(6) | 46.3(5) | 52.0(9) | 64.9(4) | 42.6(6) |
| Struck[12] | 7.18 | 44.4 | 35.4(7) | 47.0(7) | 53.4(9) | 45.0(7) | 43.9(9) | 58.0(5) | 39.0(7) | 30.4(10) | 63.5(3) | 54.4(8) | 40.6(7) |
| VTD[16] | 7.64 | 43 | 30.9(10) | 48.8(4) | 53.9(8) | 38.6(9) | 46.2(6) | 57.3(7) | 37.2(8) | 28.3(11) | 63.1(6) | 54.9(7) | 38.5(8) |
| RGBdet | 9.36 | 39.9 | 26.7(13) | 40.9(10) | 54.7(7) | 31.9(12) | 46.0(7) | 50.5(10) | 35.7(9) | 34.8(7) | 46.8(11) | 56.2(6) | 33.7(11) |
| PCflow | 10.82 | 37.1 | 35.2(8) | 29.1(14) | 43.6(11) | 42.2(8) | 33.2(13) | 47.2(12) | 33.1(10) | 32.4(9) | 43.5(12) | 41.3(13) | 35.5(9) |
| CT[30] | 10.91 | 36.4 | 31.1(11) | 46.7(8) | 36.9(13) | 39.0(10) | 34.4(12) | 48.6(11) | 31.5(11) | 23.3(14) | 54.3(8) | 42.1(12) | 34.2(10) |
| MIL[3] | 11.18 | 35.5 | 29.0(12) | 35.1(12) | 44.4(10) | 32.5(13) | 38.5(10) | 51.6(9) | 29.7(13) | 33.8(8) | 38.7(13) | 50.2(10) | 30.5(13) |
| TLD[15] | 11.55 | 35.9 | 32.2(9) | 37.2(11) | 38.3(12) | 36.6(11) | 34.6(11) | 45.5(13) | 31.5(12) | 25.6(12) | 49.0(10) | 40.4(14) | 33.6(12) |
| SemiB[11] | 13.55 | 32.3 | 22.5(14) | 33.0(13) | 32.7(14) | 24.0(14) | 31.6(14) | 38.2(14) | 24.4(14) | 25.1(13) | 32.7(14) | 41.9(11) | 23.2(14) |
| OF | 15.27 | 18.6 | 17.9(15) | 11.4(16) | 23.4(15) | 20.1(15) | 17.5(16) | 18.1(16) | 18.8(15) | 15.9(15) | 22.3(15) | 23.4(15) | 16.8(15) |
| IIDfirstBB | 15.27 | 15.1 | 9.7(16) | 20.9(14) | 18.3(15) | 13.3(16) | 16.5(14) | 31.2(14) | 8.7(16) | 15.0(16) | 15.3(16) | 13.6(16) | 19.2(15) |
| IIDrandSize | 17.18 | 5.1 | 2.8(17) | 7.0(18) | 6.8(17) | 4.0(18) | 6.0(17) | 10.9(17) | 2.8(17) | 5.0(17) | 5.2(17) | 5.1(17) | 5.2(17) |
| IIDcenterBB | 17.91 | 4.6 | 0.7(19) | 9.4(17) | 6.5(18) | 4.6(17) | 4.6(18) | 10.0(18) | 2.4(18) | 4.8(18) | 4.3(18) | 5.1(18) | 3.3(18) |
| IIDrandLoc | 19 | 2 | 2.3(18) | 1.2(19) | 2.1(19) | 3.0(19) | 1.2(20) | 2.0(19) | 2.0(19) | 2.0(19) | 2.0(19) | 2.0(19) | 2.2(19) |
| IIDrandLocSize | 20 | 0.7 | 0.6(20) | 0.4(21) | 0.8(20) | 0.8(20) | 0.6(19) | 0.9(20) | 0.6(20) | 0.7(20) | 0.7(20) | 0.7(20) | 0.6(20) |
| IIDrandWithoutPoir | 20.91 | 0.1 | 0.1(21) | 0.0(20) | 0.1(21) | 0.1(21) | 0.0(21) | 0.1(21) | 0.1(21) | 0.1(21) | 0.1(21) | 0.1(21) | 0.1(21) |

Type II error. For those algorithms, GTnoOcc is the appropriate performance upper bound they should be compared to, which includes the total amount of Type II error caused by this assumption. For algorithms that assume size fixed bounding boxes, GTfirstSize is the upper bound they should be compared with. Error decomposition also reveals where performance difference come from. TLD and SemiB have a relatively high Type III error, suggesting that their models are sensitive to target appearance change or partial occlusion.

Our proposed baseline algorithms use very powerful but computationally expensive features, classifiers, and a state-of-the-art optical flow algorithm, while some other trackers mainly focus on real-time performance. Thus our tracker is expected to have higher accuracy at the cost of longer running time. The current median frame rate is 0.26 FPS for RGBDOcc+OF, implemented in Matlab without optimization on speed.

## 5. Discussion

**Advantage from depth** From the evaluation results, trackers that utilize depth have advantages especially when the target rotates, deforms or is under occlusion. Target appearance can change significantly after rotation or deformation, making recognition difficult, which are the main causes of model drifting for traditional RGB trackers. However, the depth or 3D features are still distinguishable when the similarity in RGB vanishes. When the target is partially occluded (video "face", Figure 11 Row 3), fragment based trackers (*e.g.* [3, 11]) can locate the target but sometimes mistake background clutter for the target. Conservative approaches, which do not produce output with low confidence, often lose track of the target at this point. However, from depth data, trackers are able to identify the occluder and raise the confidence in its neighboring 3D region, compensating for the confidence loss due to partial occlusion, and thus identifies the target more accurately. When the occluder gradually grows inside the target bounding box, if not excluded, will finally dominates the bounding box (video "sign" "walking people", Figure 11 Row 2, 4). Traditional trackers are often misled to track or detect the occluder. It is difficult to make corrections afterwards because their models are already updated incorrectly. With a reliable occlusion detection mechanism, the occluder can be recognized and hence will not be output as the result or used to update models.

**2D image patch and 3D point cloud** The results above show that between the two methods that utilize depth data, the 2D image patch based tracker slightly outperforms the one based on 3D point cloud. Considering that image-patch based tracker, such as detection with HOG feature and point tracking with optical flow, are very well studied while our approach in 3D point cloud tracking are relatively new, we believe that the small performance gap indicates the great potential for future development in 3D trackers, such as 3D HOG and 3D optical flow.

## 6. Conclusions

We propose a unified tracking benchmark for both RGB and RGBD tracking, and present the evaluation of sev-

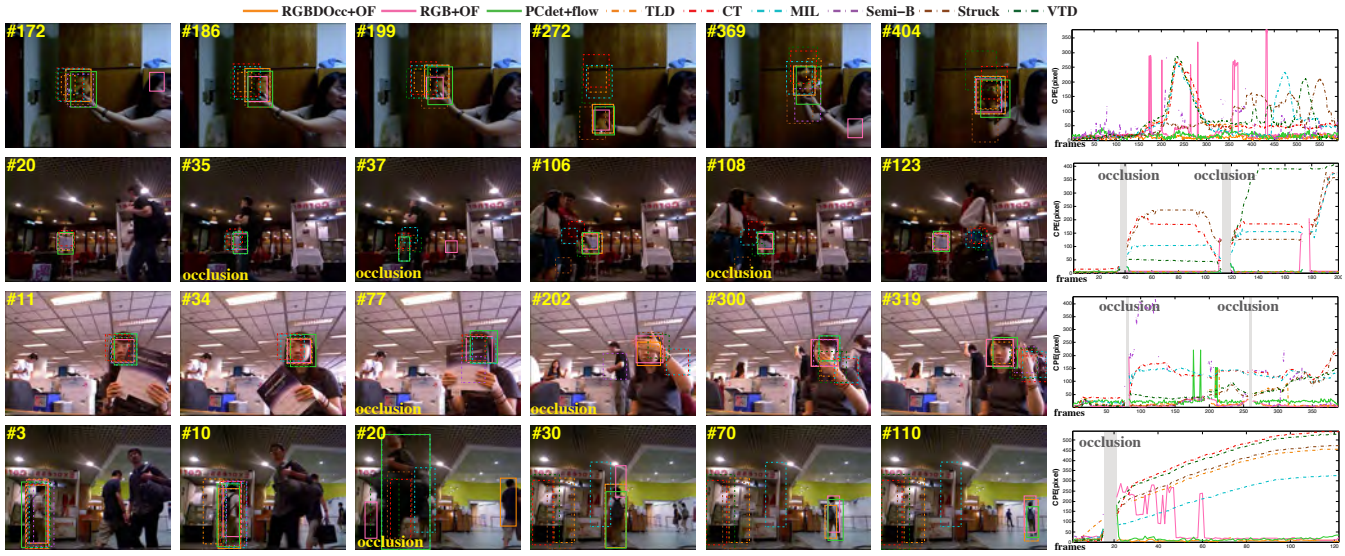



Figure 11. Output bounding boxes and their center position error (CPE). For the sake of clarity we only show results of RGBDOcc+OF, RGB+OF, PC(det+flow), TLD, CT, MIL, Semi-B, Struck, VTD. The CPE is undefined when trackers fail to output a bounding box or there is no ground truth bounding box (the target is totally occluded).

eral baseline algorithms using 2D detector, 3D detector, optical flow and ICP. We design a simple occlusion handling algorithm based on the depth map, and also evaluate several state-of-the-art RGB tracking algorithms. The results demonstrate that by incorporating depth data, trackers can achieve better performance and handle occlusion much more reliably. We hope that our unified benchmark provides new insights to the field, by making experimental evaluation more standardized and easily accessible.

## References


[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 2, 5

[2] S. Avidan. Support vector tracking. *PAMI*, 2004. 1

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *CVPR*, 2009. 1, 2, 5, 6, 7

[4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 2

[5] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 2011. 5

[6] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 2007. 4

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 4

[8] M. Everingham, H. Muller, and B. Thomas. Evaluating image segmentation algorithms using the pareto front. *ECCV*, 2002. 2

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 2, 4

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010. 4

[11] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008. 1, 2, 6, 7

[12] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 1, 2, 6, 7

[13] H. Jiang and J. Xiao. A linear approach to matching cuboids in RGBD images. In *CVPR*, 2013. 2

[14] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, 1997. 4

[15] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 2012. 2, 5, 6, 7

[16] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, 2010. 1, 2, 6, 7

[17] M. Luber, L. Spinello, and K. O. Arras. People tracking in rgb-d data with on-line boosted target models. In *IROS*, 2011. 2

[18] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 2008. 1, 2

[19] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 2

[20] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006. 2

[21] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[22] L. Spinello and K. O. Arras. People detection in rgb-d data. In *IROS*, 2011. 2

[23] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 1

[24] J. van de Weijer, C. Schmid, and J. Verbeek. Learning color names from real-world images. In *CVPR*, 2007. 4

[25] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Inverting and visualizing features for object detection. *ICCV*, 2013. 3

[26] Y. Wu, J. Lim, and M. hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 2

[27] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 2

[28] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 2

[29] J. Xiao, B. Russell, and A. Torralba. Localizing 3D cuboids in single-view images. In *NIPS*, 2012. 2

[30] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, 2012. 1, 2, 6, 7

[31] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994. 5