Multi-modal Point-wise Generic Object Tracking with RGB-D Data

Mehmet Kemal Kocamaz[1]

Robotics Institute, Carnegie Mellon University

Christopher Rasmussen[2]

Department of Computer and Information Sciences, University of Delaware

Abstract

Object tracking aims to estimate the state of the object representation in consecutive frames. Representing the object in the low-dimensional space, such as a *bounding box*, is a common way followed by the tracking algorithms. However, a more accurate and detailed point-wise representation is useful to obtain better object descriptors which could be more beneficial for action recognition and pose estimation tasks. Hence, point-wise tracking of the object could be more beneficial for these applications. In this paper, we propose a novel and accurate multi-modal, point-wise, and generic object tracker which uses RGB-D data. It does not make any assumptions about the shape of the object, so it is generic. The presented method builds a point-wise descriptor which combines the color and the shape related cues. The descriptors are trained by Random Decision Forests online. The confidence scores achieved from the classification stage are used in a graph cut step to have the final mask of the object. The displacement of the object is computed by a keypoint matching method between the frames. The proposed method was experimented with several datasets and outperformed suitable point-wise trackers.

Keywords: RGB-D Tracker; Multi-modal tracking; Point-wise tracker; Random Decision Forests; RGB-D; Graph Cut

1. Introduction

Object tracking is an essential task for wide range of applications, such as robot vision [1] [2] [3] and surveillance [4] [5] [6]. Many tracking algorithms represent the object in the low-dimensional space as a *bounding box* [7] [8] [9] [10] [11] [12] [13] [14]. The purpose of these methods is to track the object by estimating the position and scale of the *bounding box* in the subsequent images. Even though this representation is enough for some tasks, a more accurate and detailed point-wise representation is useful to obtain better object descriptors which could be more beneficial for action recognition [15] [16] and pose estimation tasks [17] [18]. This is possible in two ways. First, the point-wise representation of the object can be obtained by a refining the *bounding box* [19]. Or in another way, directly its point-wise representation is tracked [20] [21] [22] [23] [24].

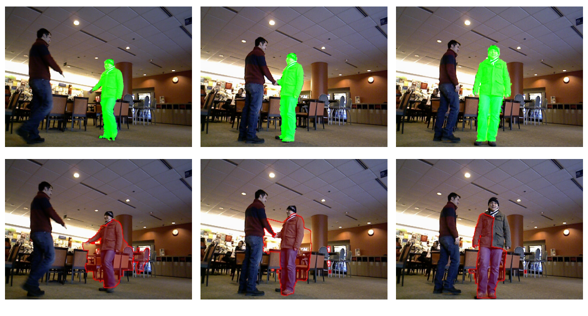


Figure 1: Top row illustrates the result of our proposed point-wise multi-modal tracker for a dataset. Bottom row displays the result of a single-model tracker [20]. The proposed tracker uses RGB-D data to achieve more accurate results.

[1]kocamaz@cmu.edu
[2]ras@udel.edu

In this paper, we propose a novel multi-modal point-wise tracker which utilizes RGB-D data. It does not make any assumptions and restrictions about the shape of the object, so it is generic for different object types, such as rigid and deformable. It learns point-wise shape related cues and the color of the object online. A point-wise descriptor is built to employ the color and the shape information of the point neighborhood. This powerful descriptor uses the depth data of the scene by generating three cues that are (1) relative geodesic and (2) vectorial spatial distances of a point to the center of the mass of the object, and (3) the local structure information encoded as the normal. The descriptors are used to train a Random Decision Forests machine learning algorithm [25] online which favors the most important cues in the descriptor. The confidence scores obtained from the classifier are passed to a graph cut step to have final more smooth results. The displacement of the object mask between the frames is estimated by computing the shift of the center of the mass. This estimation process employs a keypoint matching process. The proposed tracker was experimented with several datasets which include rigid and deformable objects. It outperformed the compared algorithms which aim to track point-wise representations of the objects. An illustration of the results of the proposed method can be seen in Figure 1

Related work is reviewed and summarized in the next section. The details of the multi-modal tracker, point-wise descriptor, training, classification and graph cut steps are described in Section 3. The results of the proposed tracker are compared and analyzed in Section 4. Finally, the proposed work is summarized and possible future directions of this work are drawn in the last section.

2. Related Work

Object tracking has been studied for decades. The tracking methods differ according to the types of the object representations. Different object representations require different solutions. Categorization of object representations is summarized by [26] and recently by [27]. One way of representing the object is as a point as explained in [28]. To track a non-rigid object, it can be formed by primitive geometric shapes as in [29]. If the object has a complex shape and its detailed border has to be output, an active contour based method [30] can be the solution.

Tracking the objects by representing them as point-wise is possible. Optical flow is one method to estimate the motion of some sparse points in the images [31] [32] [33]. Also, dense optical flow methods are available to estimate the movements of all points in the object [34]. [35] incorporates a distance penalty to the graph cut energy to eliminate non-object regions. This work is one of the earliest attempt to convert the graph cut to a point-wise tracker.

The methods described in [23] [24] introduce a novel way to incorporate the previous locations of the tracked objects as high-level observations to the graph cut. A multi-layer graph is constructed to combine low and high level observations. Background subtraction is performed and a set of candidate blobs are obtained in the preprocessing step. High level observations are included as nodes in another layer in the graph. These nodes provides temporal and spatial consistency for the object tracking between the frames. The pixel displacements between the frames are estimated by Lucas-Kanade tracker. In the graph cut based tracker presented in [24], occluded parts of the objects are tracked by adding a new penalty term to the graph cut energy function. The points in the object which do not move as the average displacement of the object are penalized to determine the occluded parts.

A point-wise tracking method which combines the graph cut and optical flow techniques is described for augmented reality applications in [36]. Since augmented reality applications are so depended on the computational time, this algorithm works in real-time. A similar approach is explained in [37] to track the people in videos by modeling them as ellipsoids. [38] proposes an object tracker method for live videos. A 3-D graph is built and location probability information of the object is employed in the graph.

[20] introduces a point-wise tracking method which learns object appearance online. The learning step includes generalized Hough-Transform which provides a rough estimation of the object. It is provided to a final GrabCut [19] step. [21] uses Hough transform by building point-wise descriptors. [39] proposes a level-set formulation to track non-rigid objects. [40] presents a joint method which fuses multi-part and segmentation in a RANSAC-style iterative energy optimization framework. The method explained in [40] aims to track superpixels. [41] uses superpixels to build a dynamic graph and generates rough borders of the

object.

The methods proposed in [10] [11] [12] [9] [42] use RGB-D data to track the objects. [10] describes a particle filter based object tracker. This algorithm takes a 3-D mesh model of the object as the prior. Its particle filter framework is parallelized. A model based and hypothesize-and-test approach is proposed in [11] to track interacting objects. In [12], 3-D level set functions are employed to represent the object. A 3D chamfer matching based energy function is minimized during tracking. A fast people RGB-D tracker algorithm is proposed in [9]. A joint likelihood function consisting of appearance and depth models are learned online. [42] proposes a method to track deformable objects without requiring prior object model. It learns how to weight local image appearance, depth discontinuities, and surface normals from a set of ground truth data.

3. Multi-modal Tracker Details

The proposed point-wise multi-modal tracker, $MM - Tracker$, is initialized with the ground truth mask of the object at the first frame. It learns point-wise color and shape related cues online from RGB-D data to track the object. Lastly, a graph cut step is applied to obtain more smooth results. The displacement of the object between the frames is estimated by computing the center of mass shift. This estimation process involves a keypoint matching step. The overall process diagram of $MM - Tracker$ is illustrated in Figure 2.

3.1. Estimation of the Object Displacement

Detecting the keypoints in a 3-D point cloud is computationally more expensive than detecting them in a color image. Hence, $MM - Tracker$ prefers to compute them in the color images instead of in the corresponding point clouds. SIFT [43] feature descriptor is used to estimate roughly the displacement of the object from time t to time $t + 1$. There are several reasons why SIFT is chosen for this purpose. SIFT feature descriptor is invariant to uniform scaling and orientation. Also, it is partially invariant to the distortion and illumination changes. In addition, it is fast to compute, so it is suitable for $MM - Tracker$. However, corresponding 3-D point cloud of the image obtained from the depth image is used for projecting the keypoints to 3-D space in $MM - Tracker$. The displacement of the keypoints between the frames are computed in this 3-D space to achieve more accurate estimation. The location of the object at time $t + 1$ is estimated by the following steps:

1) For given two color images, I_t and I_{t+1}, whose time stamps are t and $t+1$ respectively in the image sequences, their SIFT keypoints are computed by:

$$K_t = \mathscr{S}(I_t) \ \ , \ \ K_{t+1} = \mathscr{S}(I_{t+1}) \tag{1}$$

where K_t is the set of the keypoints, and $\mathscr{S}$ is the operator to obtain SIFT keypoints from the image I_t.

2) The keypoints which are outside of the object region are excluded only from K_t:

$$K_t^- = \{k(x, y) \in K_t \ : \ k(x, y) \notin M_t\} \tag{2}$$

$$K_t = K_t - K_t^- \tag{3}$$

where $k(x, y)$ is a keypoint in the image, M_t is the object region in the image.

3) The keypoints in K_t and K_{t+1} are matched:

$$K_{t,\ t+1} = \mathscr{M}(K_t,\ K_{t+1}) \tag{4}$$

$$\begin{aligned} K_{t,\ t+1} = \{k_t(x_1, y_1), k_{t+1}(x_1, y_1)\}&, \\ ....&, \\ \{k_t(x_n, y_n), k_{t+1}(x_n, y_n)\}& \end{aligned} \tag{5}$$

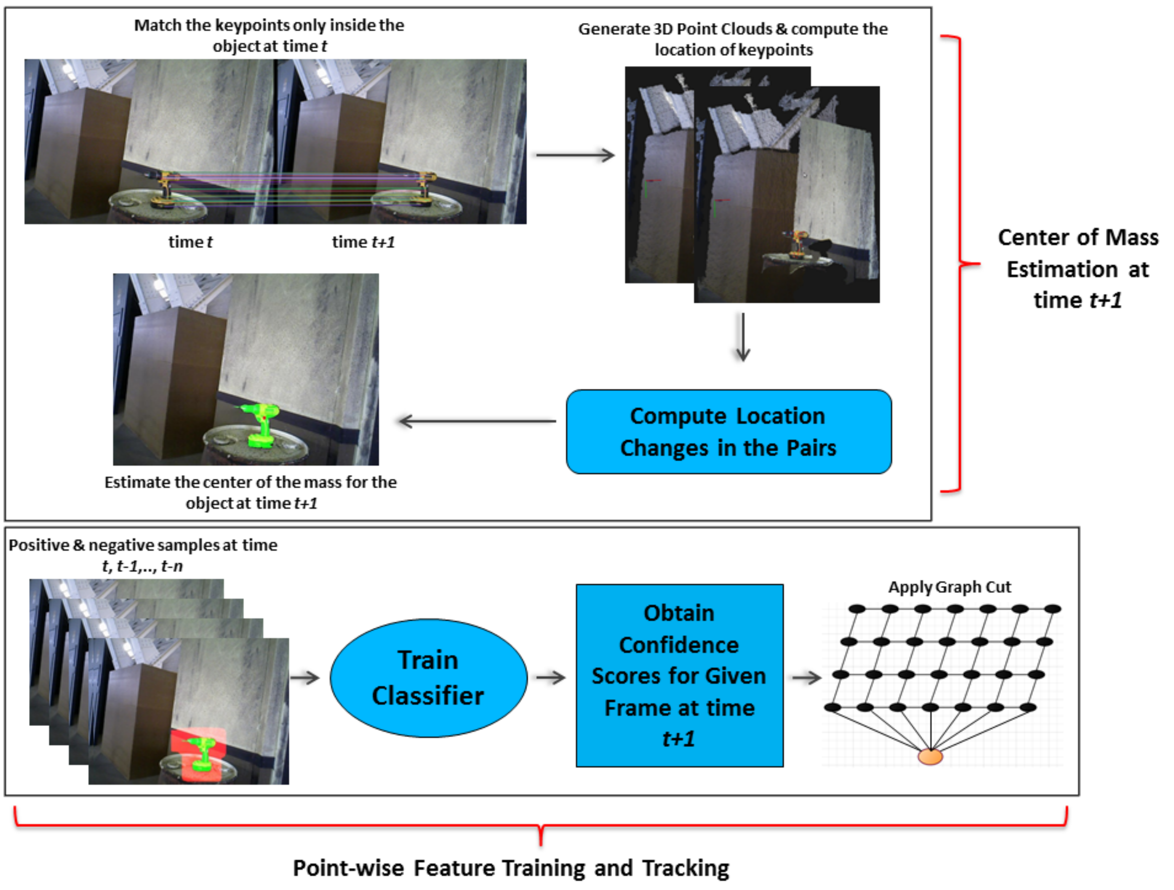
where $K_{t,\ t+1}$ is the set of pairs of keypoints matched from K_t to K_{t+1}, and $\mathscr{M}$ is the keypoint matching function.

4) Convert the depth images to 3-D point clouds for timestamps of t and $t+1$ in the image sequences. Then, compute the corresponding 3-D points of all keypoints in $K_{t,\ t+1}$:

$$k^{3D}(x, y, z) = T(\mathscr{P},\ k(x, y)) \tag{6}$$

$$\begin{aligned} K^{3D}_{t,\ t+1} = \{k^{3D}_t(x_1, y_1, z_1), k^{3D}_{t+1}(x_1, y_1, z_1)\}&, \\ ....&, \\ \{k^{3D}_t(x_n, y_n, z_n), k^{3D}_{t+1}(x_n, y_n, z_n)\}& \end{aligned} \tag{7}$$

where T is the function to compute the corresponding point in the point cloud $\mathscr{P}$ for a given keypoint $k(x, y)$ in the image. As an example, Figure 3 shows the keypoints inside of the object in both color image and its corresponding 3-D point cloud after this step is applied.



Match the keypoints only inside the object at time *t*
time *t*
time *t+1*
Generate 3D Point Clouds & compute the location of keypoints
Compute Location Changes in the Pairs
Estimate the center of the mass for the object at time *t+1*
Center of Mass Estimation at time *t+1*
Positive & negative samples at time *t, t-1,.., t-n*
Train Classifier
Obtain Confidence Scores for Given Frame at time *t+1*
Apply Graph Cut
Point-wise Feature Training and Tracking

Figure 2: Overall process diagram of the proposed $MM-Tracker$.

5) The rough displacement of the object from the time t to the time $t+1$ is calculated as the average displacement of the matched keypoints:

$$p^{3D}_{dis} = \frac{1}{n}\sum_{i=1}^{n} k^{3D}_{t+1}(x_i, y_i, z_i) - k^{3D}_{t}(x_i, y_i, z_i) \quad (8)$$

where p^{3D}_{dis} is the displacement of the object.

6) All object points, M_t, at time t are converted to their corresponding in 3-D, M^{3D}_t, by using function T as in the Equation 6. Then, the estimated location of the object at time $t+1$, M^{3D}_{t+1} is computed by adding average displacement, $p^{3D}_d is$, to the each point of M^{3D}_t:

$$p^{3D}_i = T(\mathscr{P},\ p_i) \quad where\ p_i \in M_t,\ \ p^{3D}_i \in M^{3D}_t \quad (9)$$

$$M^{3D}_{t+1} = M^{3D}_t\ +\ p^{3D}_{dis} \quad (10)$$

7) Assuming that the camera calibration parameters are known, also the estimated object points in the image, M_{t+1} can be achieved:

$$p_i = T^{-}(p^{3D}_i) \quad (11)$$

where $p_i \in M_{t+1}$, $p^{3D}_i \in M^{3D}_{t+1}$, and T^{-} is the transformation function from the 3-D world space to the image space.

3.2. Point-wise Descriptor

A point-wise descriptor, f_p is formed by incorporating several shape related and color cues in the following way:

1) Normals: A cue about the local shape information surrounding the point, p_i, can be encoded in the descriptor, f_p, by calculating the normal, η_i, of the point, p_i. It is computed for all three dimensions of the point cloud space, $\eta_i = (\eta_x, \eta_y, \eta_z)$. The neighborhood search of the points is performed

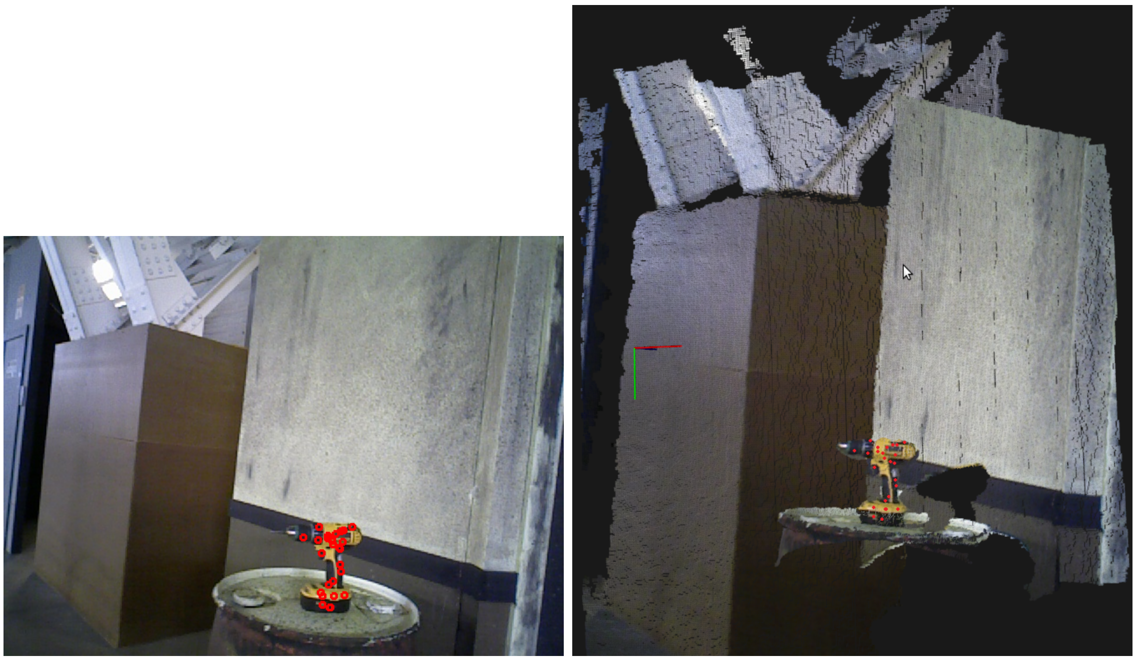


Figure 3: Detected keypoints inside of the object in the color image and 3D point cloud. (a) displays the image at *time t* with the keypoints drawn as red, (b) shows the corresponding point cloud and the keypoints colored as red. The scene is rotated in the Point Cloud Viewer for better display.

by building a FLANN-based Kd-tree [44] to reduce the computation time.

2) Vectorial Spatial Distance: First, the center of the mass, $mid_B = (mid_x, mid_y, mid_z)$, of the object region is calculated as formulated in the following equations:

$$mid_B = \frac{1}{m}\sum_{i=1}^{m} p_i^{3D} \tag{12}$$

where m is the number of the points in M_t^{3D} and $p_i^{3D} \in M_t^{3D}$. The vectorial distance relative to the center of the mass of the object, $\Delta_v = (\Delta_x, \Delta_y, \Delta_z)$, is computed for the point $p_i = (p_x, p_y, p_z)$. This computation can be formulated as:

$$\Delta_v = (p_x - mid_x, p_y - mid_y, p_z - mid_z) \tag{13}$$

3) Geodesic Distance: The geodesic distance between two points on the object is constant in different poses. This cue is incorporated into our descriptor, f_p. The relative geodesic distance, GD_i, to the center of the mass of the object, mid_B, of the point, p_i, is computed by Dijkstra's Shortest Path Algorithm. The image is converted to a graph, $G(V, E)$, where V is the graph nodes, and E is the edges between the nodes. Each point, p_i, in the image is represented as a node in the graph, $G(V, E)$. The neighbors of each node in the graph are restricted to 4 pixels. The edge weight, w_{ij}, between two points is set to the Euclidean distance between p_i and p_j in the corresponding point cloud of the scene as in Equation 14.

$$w_{ij} = \sqrt{|p_{i_x} - p_{j_x}|^2 + |p_{i_y} - p_{j_y}|^2 + |p_{i_z} - p_{j_z}|^2} \tag{14}$$

If there is no depth data is available for the neighbor, the edge weight, w_{ij} is assigned a large distance. A sample geodesic distance map for the given image can be seen in Figure 4.

4) Color: In addition to the shape related cues, also the color of the point is added to the pointwise descriptor, f_p, to make $MM-Tracker$ more robust. Finally, f_p becomes:

$$f_p = [\eta_x\ \eta_y\ \eta_z\ \Delta_x\ \Delta_y\ \Delta_z\ GD_i\ r\ g\ b]^T \tag{15}$$

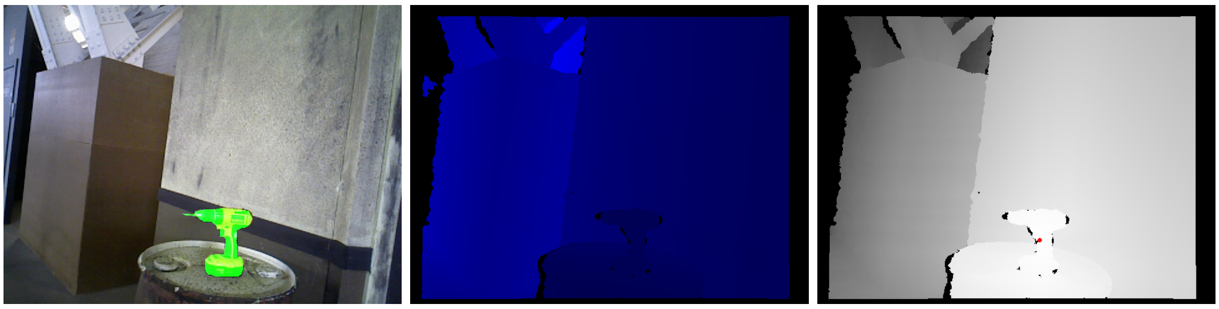

(a) (b) (c)

Figure 4: Geodesic distance calculation. (a) Color image with overlaid object mask, (b) Depth image, (c) Colored geodesic distance map of the given image. The red point corresponds to the object center of the mass. The points which have darker intensity in the map are farther from the object and lighter points are closer to it.

where r, g, and b are red, green, and blue channels of the point, p_i, in the color image of a sequence.

3.3. Training Online Classifier and Classification

In order to classify the points in next image whose time stamp is $t+1$, a classifier is trained with the samples obtained from a image set, S_{img}, online. This image set, S_{img}, includes the images whose timestamps are t, $t-1$, ..., $t-n$. This mechanism can be considered as sliding a tracking window, w_t, on the image sequences and the size of the w_t is n. Randomized Decision Forests, RDF, are a state of the art, fast, and effective machine learning technique [45] [25] [46] [47] which are suitable and applicable for wide range of different tasks and problems [48] [49] [50]. Therefore, it is used to train our online classifier.

Positive training samples, f_p^+, are taken from the inside of the object region, M_t. Negative samples, f_p^-, are obtained from the outside of the object. In the classification process of the points in next image whose time stamp is $t+1$, first the object location is estimated by SIFT based method explained in the previous section. The center of the mass of the object, mid_B, is computed according to the estimated object location. Then, the points of this frame are classified. RDF provides a confidence score, $\mathcal{C}_i$, of being in the object region for a point, p_i, in the image. In our case, $\mathcal{C}_i$ is the average decision distribution at the leaf nodes of RDF trees. The classification confidence scores produced by RDF are feeded to a graph cut procedure to obtain the final result.

3.4. Graph Cut Smoothing

Graph cut [51] [52] [53] [54] is a powerful method to achieve more smooth results or to eliminate noises in the final result mask of the object, M_{t+1}. The confidence scores, $\mathcal{C}_i$, can be used to set edge weights in the graph cut. Therefore, $MM-Tracker$ applies the graph cut method as the last step in the tracking process by utilizing $\mathcal{C}_i$.

The energy function of the graph cut consists of two terms, namely the regional term, R, and the boundary term, B:

$$E(\widetilde{L}) = \sum_{i \in V} R(l_i) \; + \; \alpha \sum_{\{i,j\} \in V} B_{i,j}(l_i, l_j) \qquad (16)$$

where i and j are the nodes of any edge, $e_{i,j}$, in the graph. α sets the relative influence between the terms.

The regional term of the graph cut is formulated as:

$$R(l_i) = -\ln(p_c(p_i, l_i)) \qquad (17)$$

where l_i is the label of the point, p_i, in the image. $p_c(i, l_i)$ is the likelihood of the point as it is defined in the following Equation:

$$p_c(l_i) = \begin{cases} \mathcal{C}_i & \text{if } \; l = \text{"}Object\text{"} \\ 1 - \mathcal{C}_i & \text{if } \; l = \text{"}Background\text{"} \end{cases} . \qquad (18)$$

The boundary term is the absolute difference between the regional term scores of the points:

$$B_{i,j}(l_i, l_j) = |R_i(\text{"}Object\text{"}) - R_j(\text{"}Object\text{"})| \qquad (19)$$

4. Experiments

Several experiments were conducted to analyze and measure the performance of the proposed

$MM - Tracker$. Darpa Robotics Challenge [55], *DRC*, became our inspiration to create first experiment to test the described method. In one of the tasks in *DRC*, the robot must accomplish using a drill to open a hole on the wall. A similar a scenario was established as a test case.

In this scenario, the robot was far away from a drill. It walks toward the drill to grab it by avoiding from the obstacles on its path. The robot might make different movements because of the obstacles, such as left/right turns, while walking. Tracking the drill properly is important for several reasons in this case. First of all, it shows the path/way for it. Also, in order to be able grab the drill for a specific task, it must recognize and know the orientation/position of the object relative to its arms to execute a correct motion plan. Tracking and finally grabbing an object a common problem for the robots. The object might be any kind of hand tool, house or kitchen equipments.

A similar test environment as in *DRC* was created to record a dataset, called as *DRC-Track*. The recording setup moved around the object. Different types of movements that the robot can make while walking were captured, such as left/right turns, approaching to the target or walking away from it. The frame rate was set to 20 per second during the recording. Total of 1900 frames, depth and color images, were saved. 76 ground-truth masks (one for every 25 frames) of the object were manually generated.

Another dataset was collected to analyze and compare the performance of $MM - Tracker$ for the articulated and deformable objects. A short video of two people in which they approach each other, shake their hands, and then walk away in front of a complex background was recorded. This dataset, called as *HandShaking*, includes 220 frames of the color and depth images. Its video frame rate is about 20 per second. One ground truth per 25 images was labeled manually.

4.1. Other Methods for Comparison

The result of $MM - Tracker$ was compared to several other methods whose goals are to track the point-wise representations of the objects. These methods are:

***Godec*[20]:** This method includes an online appearance learning step. Its learning process is based on generalized Hough-Transform which provides a rough estimation of the object. The rough segmentation of the object is given to the *GrabCut* [19] to obtain final point-wise mask of the tracked object. This method outperforms some of the state-of-the-art methods to track non-rigid objects in several challenging videos.

***GC-Tracker-3D*:** The graph cut based tracking method which is explained in [56] has been extended to compare to $MM - Tracker$. [56] is a single frame point-wise object tracker which does not estimate the location of the object in next frames. Firstly, the same SIFT based location estimation approach, described in the previous section, has been employed into it. In this way, the produced object mask by the method at time *t-1*, is shifted by the location estimation method at time *t*.

[56] includes a distance map which is computed in the image space to penalize the points far from the object. The computation of this map was modified and it was computed in 3-D space to obtain more accurate penalties. Firstly, 3-D point cloud of the scene was generated. Then, Euclidean distances of each point to the object were computed. The distance penalty maps computed in two ways are showed for a given image in Figure 4. As it can be noticed in (d) of this figure, even though some points are close to the border of the object in the image space, they might have large penalties due to their distances to the object in 3-D space.

***Farneback*[34]:** It is a dense optical flow method whose details is explained in [34]. In contrast to detecting and matching a set of keypoints, this method estimates the displacement of all points inside a mask in next frames. Its performance comparing to other types of the optical flow methods showed in [34]. It outperforms other dense optical flow algorithms as showed in [34].

4.2. Analyzing the Performance of the Methods

The tracking results of the methods were saved when they hit a ground-truth frame to analyze their performances. The following polygon area overlap formula was used to measure the overlap between the ground-truth and the result of the tracker suggested by [57]:

$$O(\mathcal{R}_1, \mathcal{R}_2) = A(\mathcal{R}_1 \cap \mathcal{R}_2)^2 / (A(\mathcal{R}_1)A(\mathcal{R}_2)) \tag{20}$$

where $\mathcal{R}_1$ and $\mathcal{R}_2$ are the two regions to calculate the overlap between. In the experiments, sliding window size of $MM - Tracker$, w_t, is set to 4. The max depth of the *RDF* tree was set to 10.

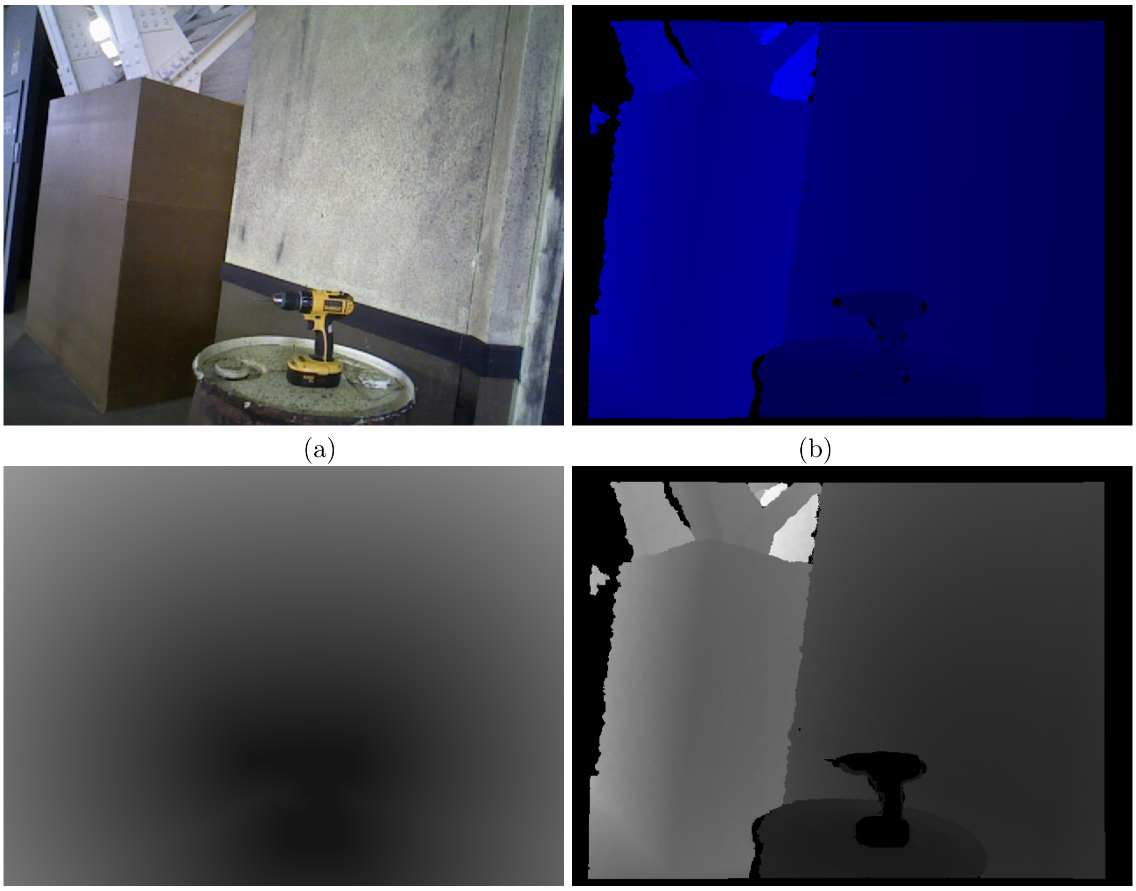


Figure 5: Examples of 2-D and 3-D distance penalty maps. (a) displays given color image, and (b) its depth image. (c) shows its 2-D distance penalty map computed in 2-D image space, and (d) its distance penalty map computed in 3-D. Please note that no penalty is assigned for missing depth data and inside the object. Darker points in the maps have less penalty.

Method Name	Overlap Score for *DRC-Track*	Overlap Score for *HandShaking*
$MM-Tracker$ (Only Color)	0.16	0.18
$MM-Tracker$ (Only Shape)	0.74	0.82
$MM-Tracker$ (No Location Estimation)	0.41	0.73
$MM-Tracker$ (Inludes All)	0.96	0.93

Table 1: Median overlap scores of $MM-Tracker$ for *DRC-Track* and *HandShaking* datasets. Each row specifies an experiment in which different combinations of the cues were used in the method. For example, $MM-Tracker$ (Only Shape) means that color cue is removed from the point-wise descriptor, f_p, of $MM-Tracker$. $MM-Tracker$ (No Location Estimation) means that the displacement of the object between the frames was not computed that experiment.

In order to see the effect of using different cues in the proposed point-wise descriptor, f_p, two tests were conducted by removing the shape or color information from f_p. Also, in another test, all cues were kept in f_p, but the displacement of the object between the frames was not computed for $MM-Tracker$. Simply, it was assumed that the object or camera does not move in the next frame. Median overlap scores of the methods for these tests can be seen in Table 1. The best performance was

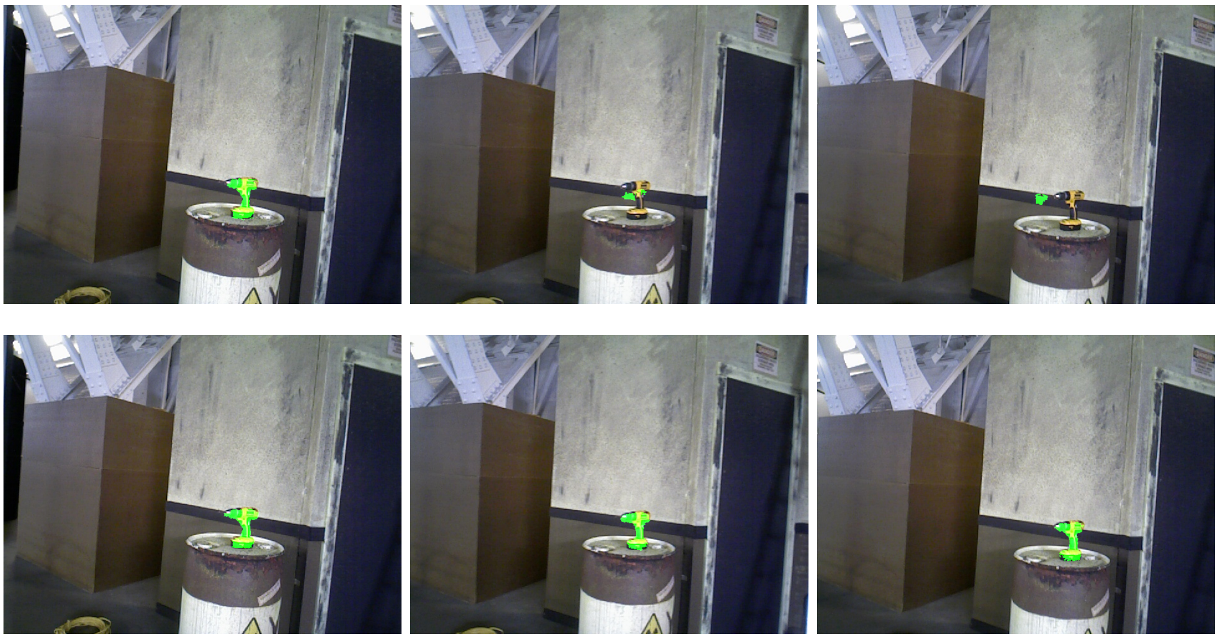


Figure 6: Sample results of $MM - Tracker$ that demonstrate the effect of the location estimation step at different time-stamps. First row shows the outcome of $MM - Tracker$ without estimating the object displacement. Second row displays the results of $MM - Tracker$ if it includes the proposed SIFT based location estimation method. In this example, the camera makes first a right and then a left movement.

Method Name	Overlap Score for *DRC-Track*	Overlap Score for *HandShaking*
$GC - Tracker - 3D$*	0.23	0.37
$GC - Tracker - 3D$	0.48	0.43
Farneback [34]	0.28	0.45
Godec [20]	0.46	0.61
$MM - Tracker$	0.96	0.93

Table 2: Median overlap scores of the methods for *DRC-Track* and *HandShaking* datasets. $GC - Tracker - 3D$* does not include the SIFT based location estimation between the frames for $GC - Tracker - 3D$.

achieved when all cues and location estimation step were included in $MM - Tracker$.

In the absence of the location estimation step, the performance dropped from 0.95 to 0.41. Figure 6 demonstrates a case, when the location estimation was removed. In this example, the camera first moved to the right. It caused the tracker to loose the object location and stick on the wall. Then, the camera made a left movement. It sticked on tracking a part of the wall which has similar color cue. However, SIFT based location estimation employed in $MM - Tracker$ helped to continue correctly to track the object as can be seen in the second row of Figure 6. Figure 7 shows a case when the color information helped $MM - Tracker$ to track the object properly. In this case, some part of the object was lost by the tracker because of the pose changes. Only shape related cues were not enough when the camera was rotated over time. First row displays that $MM - Tracker$ sticked at the middle part of the object. However, the color information which was associated with the point-wise shape related cues helped $MM - Tracker$ to work more robust in noticable pose changes.

Median overlap scores of the methods can be seen in Table 2. $MM - Tracker$ outperformed other

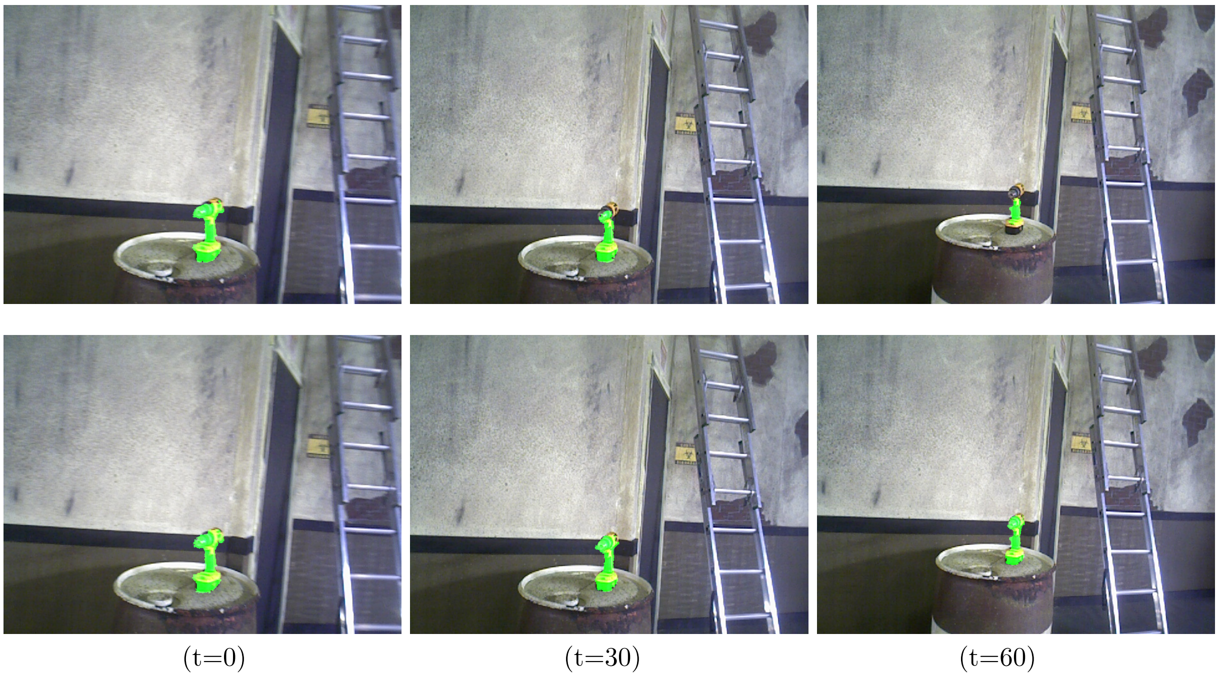


Figure 7: Sample results of $MM - Tracker$ that demonstrates the effect of incorporating the color information at different time-stamps. First row shows the results of $MM - Tracker$ without fusing the color in its point-wise descriptor, f_p. Second row displays the results of $MM - Tracker$ when all cues are incorporated.

trackers by achieving highest scores of 0.96 and 0.93, for *DRC-Track* and *HandShaking* datasets, respectively. $MM - Tracker$ can figure out that which cue is more important and distinctive in which part of the scene around the object. This ability is provided to it by the shape related cues in its point-wise descriptor, f_p. *Godec* [20] performed better than *Farneback* and $GC - Tracker - 3D$ in *HandShaking*. It can be said that *Godec*'s online learning step to build the object model works better than $GC - Tracker - 3D$. *Godec* [20] performed slightly worse than $GC - Tracker - 3D$ for *DRC-Track*. Since the background has similar color information as the object, and *Godec* does not use any 3-D related features, this performance of *Godec* can be expected. $Farneback$ [34] uses only the visual cues. Therefore, $MM - Tracker$ outperformed its results by a factor of 2.6. It can be noticed in Figure 8 that the results of $Farneback$ do not consist of dense points. The displacements of some points inside of the object hit one point in the next frame, so this situation causes the aggregation of the points to only one point over time. Some results of the trackers when they hit a ground-truth can be seen in Figure 8 and 9.

Figure 10 displays the point-wise confidence scores produced by the classifier and final results obtained from graph cut step employed in $MM - Tracker$. It can be seen in the final results that some points which have weak scores that are inside of the objects were included into the result masks. Moreover, some non-connected weak points were eliminated by graph cut. These improvements can be explained by the ability of graph cut which can combine smoothness and data terms in one framework.

4.3. Discussion About a Special Case

A rare case was encountered during recording of *HandShaking* dataset. At one point of recording, the hardware stalled due to so rare buffer problem in the sensor setup. This kind of conditions can always be expected in real life robotics systems which require to process high bandwidth data. It is so useful, if the employed method can handle these kind of issues in software level. In this case, there had been large change in the location and pose of the human between two frames. Therefore, it was decided to analyze this case separately. Figure 11 demonstrates the results of $MM - Tracker$

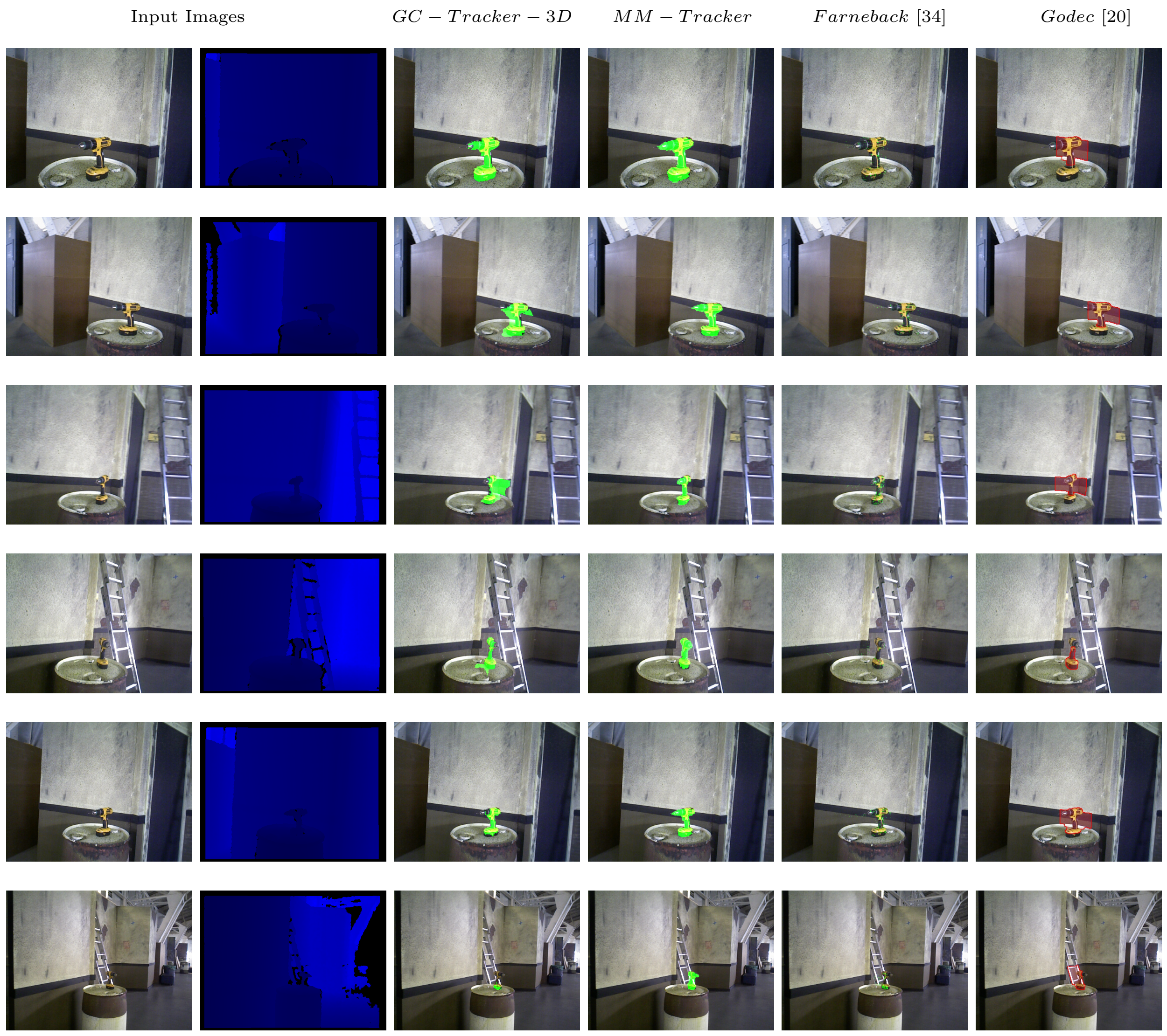


Figure 8: Sample results of different tracking methods whenever they hit a ground-truth in *DRC-Track* dataset. Column headings show the name of the methods. If the object was lost by the tracker, there is no green colored mask.

with and without having proposed location estimation method. The stall occurs between time=0 and time=1. As it can be seen in the last row of Figure 11, $MM-Tracker$ was be able to capture most part of the person at time=1, even though the pose of the foot and arms are different. It grasped the other parts of the body at subsequent frames, after time=1.

4.4. Computational Load

A machine which has 32GB RAM and Intel i7-2760QM quad processor was used in the experiments. The methods were implemented in C++.

Method Name	*Time (in sec)*
$MM-Tracker$	0.41
$GC-Tracker-3D$	0.12
$Farneback$ [34]	0.05
$Godec$ [20]	0.71

Table 3: Average running time of the trackers per image.

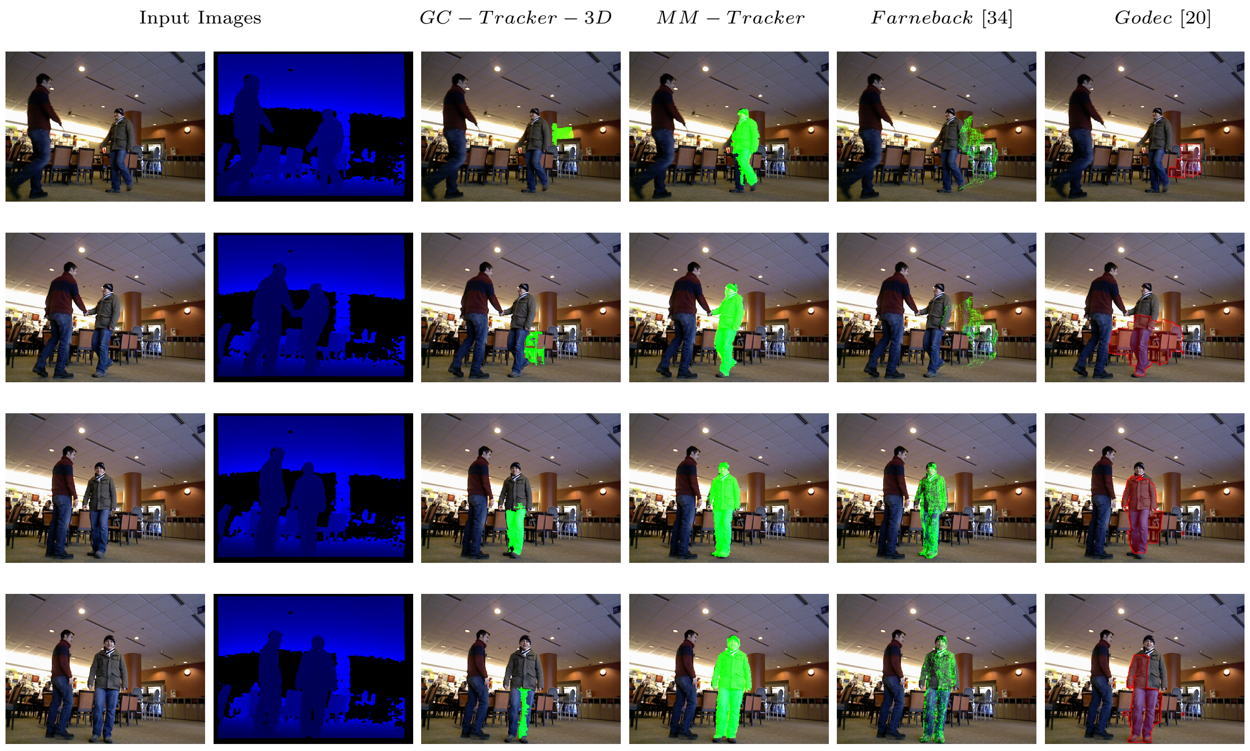


Figure 9: Sample results of different tracking methods whenever they hit a ground-truth in *HandShaking* dataset. Column headings show the name of the methods.

The implementations do not contain thread-level parallel processing. The image resolution was 640 x 480 for all experiments. 3 RDF trees was trained for $MM-Tracker$ in the experiments. As it can be seen in Figure 12, in order to reduce the online training time during the tracking, only some points which are around the object were used as the background samples.

The average running time of the trackers per image can be seen in Table 3. $MM-Tracker$ took the average of 0.41 seconds processing time per image for the experimented datasets. This time also includes the steps for the computation of the features. As expected, *Farneback* [34] was the fastest tracker. We believe in that implementation of our proposed tracker on Graphics Processing Unit (GPU) would reduce its computational time. Also, scaling the input image can help to decrease the computational load.

5. Conclusion

We describe a novel point-wise and multi-modal object tracker, $MM-Tracker$, which uses RGB-D data. Our method estimates the displacement of the object in a generic way from one frame to another by utilizing a keypoint matching process. $MM-Tracker$ forms a powerful point-wise descriptor which consists of the color and shape related cues. Positive and negative point-wise descriptors are trained by Random Decision Forests in $MM-Tracker$. A final graph cut step is applied to produce better smooth results.

The performance of $MM-Tracker$ was compared with other suitable point-wise trackers. One of the previously published method [56] was modified for this purpose. Two different sets of experiments were performed to quantify and analyze the results of $MM-Tracker$. One of the dataset includes a complex shaped hand tool, which is a drill, and the other dataset consists of two people who approach each other, shake their hands and then walk away. $MM-Tracker$ outperformed the other methods in these experiments. As future work, the observed locations of the object can be included as the high level observations into the graph cut step.

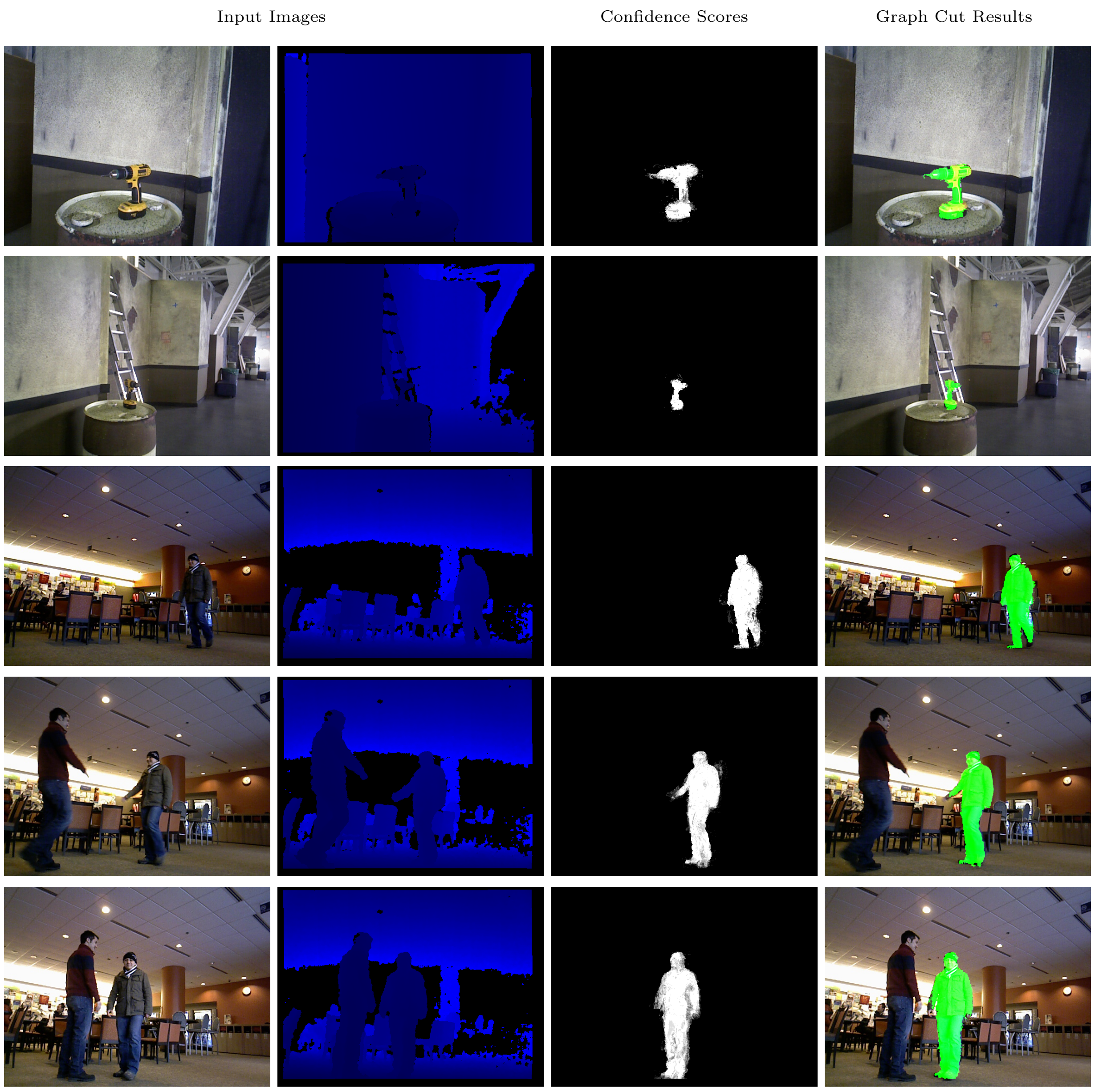


Figure 10: Sample confidence scores and their result masks produced by graph cut in $MM-Tracker$.

References

[1] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, G. Bradski, Self-supervised monocular road detection in desert terrain, in: Proceedings of Robotics: Science and Systems, Philadelphia, USA, 2006.

[2] C. Hu, X. Ma, X. Dai, K. Qian, Reliable people tracking approach for mobile robot in indoor environments, Journal of Robotics and Computer-Integrated Manufacturing 26 (2) (2010) 174–179.

[3] C. Pantofaru, L. Takayama, T. Foote, B. Soto, Exploring the role of robots in home organization, in: Proc. of Human-Robot Interaction, 2012, pp. 327–334.

[4] B. U. Toreyin, Y. Dedeoglu, U. Gudukbay, A. E. Cetin, Computer vision based method for real-time fire and flame detection, Pattern Recognition Letters 27 (1) (2006) 49 – 58.

[5] O. Javed, Z. Rasheed, O. Alatas, M. Shah, Knight/spl trade/: a real time surveillance system for multiple and non-overlapping cameras, in: Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 2, ICME '03, 2003, pp. 649–652.

[6] C. J. Costello, C. P. Diehl, A. Banerjee, H. Fisher,

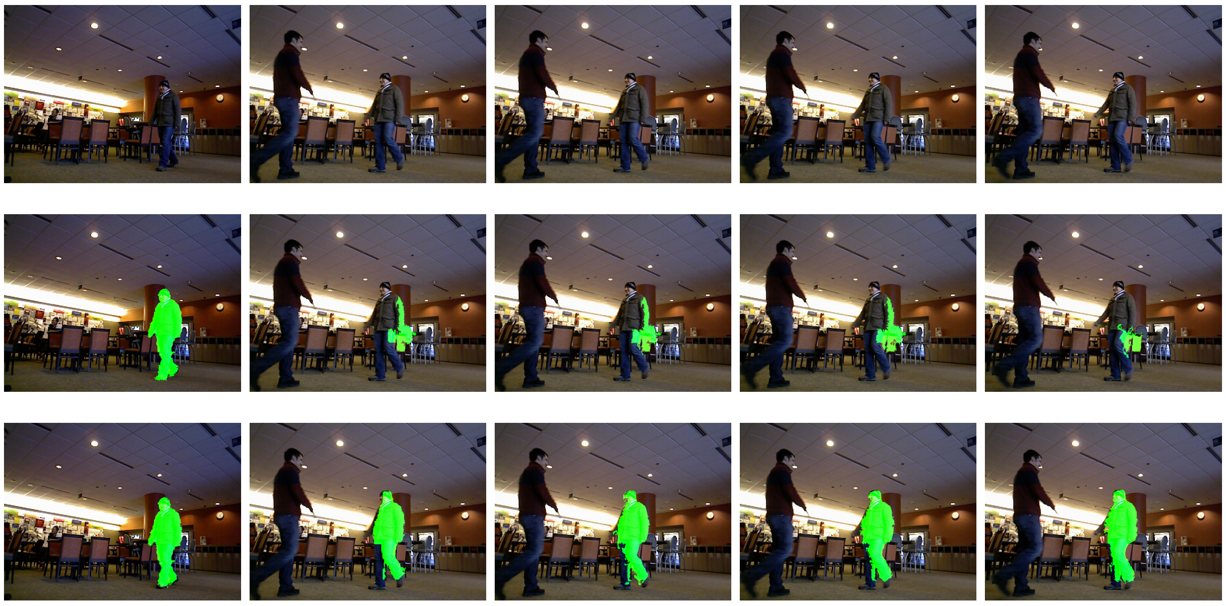


Figure 11: Analyzing a special case in which the hardware system stalled between time t=0 and t=1. The stall caused a large skip in the object location and pose difference. First row displays the raw images. Second row shows the result of $MM-Tracker$ without employing SIFT based location estimation method. Last row shows the results of $MM-Tracker$.

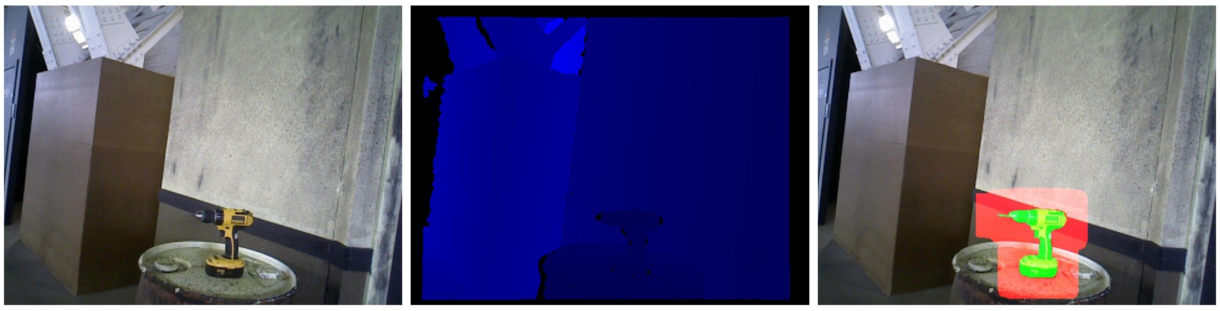


Figure 12: The object and background samples for $MM-Tracker$ during the training step. (a) Color image, (b) Depth image, (c) Overlaid object and background points in the image. The red points are used as the background samples for the training process during tracking and green points for the object. Background points are obtained by dilating the object region by some factor and then removing the object points.

Scheduling an active camera to observe people, in: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks, VSSN '04, 2004, pp. 39–45.

[7] W. Dong, L. Huchuan, Y. Ming-Hsuan, Online object tracking with sparse prototypes, IEEE Transactions Image Processing 22 (1) (2013) 314–325.

[8] S. Hare, A. Saffari, P. H. S. Torr, Struck: Structured output tracking with kernels, in: Proceedings of the International Conference on Computer Vision (ICCV), 2011, pp. 263–270.

[9] M. Munaro, E. Menegatti, Fast RGB-D people tracking for service robots, Journal of Autonomous Robots 37 (3) (2014) 227–242.

[10] C. Changhyun, H. Christensen, RGB-D object tracking: A particle filter approach on gpu, in: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 1084–1091.

[11] N. Kyriazis, A. Argyros, Scalable 3D tracking of multiple interacting objects, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 3430–3437.

[12] C. Y. Ren, I. Reid, A unified energy minimization framework for model fitting in depth, in: Proceedings of

the European Conference on Computer Vision (ECCV), 2012, pp. 72–82.

[13] B. Babenko, M.-H. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1619–1632.

[14] J. Henriques, R. Caseiro, P. Martins, J. Batista, Exploiting the circulant structure of tracking-by-detection with kernels, in: Proceedings of the European Conference on Computer Vision (ECCV), 2012, pp. 702–715.

[15] L. Chen, H. Wei, J. Ferryman, A survey of human motion analysis using depth imagery, Pattern Recognition Letters 34 (15) (2013) 1995–2006.

[16] R. Vemulapalli, F. Arrate, R. Chellappa, Human action recognition by representing 3-D skeletons as points in a lie group, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[17] M. Wai, R. Nevetia, Body part detection for human pose estimation and tracking, in: Proceedings of IEEE Workshop on Motion and Video Computing, 2007.

[18] T.-H. Yu, T.-K. Kim, R. Cipolla, Unconstrained monocular 3-D human pose estimation by action detection and cross-modality regression forest, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[19] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics 23 (2004) 309–314.

[20] M. Godec, P. M. Roth, H. Bischof, Hough-based tracking of non-rigid objects, in: Proceedings of the International Conference on Computer Vision (ICCV), 2011.

[21] S. Duffner, C. Garcia, PixelTrack: a fast adaptive algorithm for tracking non-rigid objects , in: Proceedings of the International Conference on Computer Vision (ICCV), 2013, pp. 2480–2487.

[22] A. Nelson, J. Neubert, Object tracking via graph cuts, in: SPIE Applications of Digital Image Processing, 2009.

[23] A. Bugeau, P. Prez, Track and cut: Simultaneous tracking and segmentation of multiple objects with graph cuts., in: VISAPP (2)'08, 2008, pp. 447–454.

[24] N. Papadakis, A. Bugeau, Tracking with occlusions via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (1) (2011) 144–157.

[25] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[26] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, ACM Comput. Surv. 38.

[27] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, IEEE Transactions on Pattern Analysis and Machine Intelligence PP (99) (2015) 1–1.

[28] D. Serby, L. V. Gool, Probabilistic object tracking using multiple features, in: In IEEE International Conference of Pattern Recognition (ICPR), 2004, pp. 184–187.

[29] D. Comaniciu, V. Ramesh, P. Meer, S. Member, S. Member, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 564–577.

[30] A. Yilmaz, X. Li, M. Shah, Contour based object tracking with occlusion handling in video acquired using mobile cameras, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1531–1536.

[31] B. D. Lucas, T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, in: Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2, 1981, pp. 674–679.

[32] J.-Y. Bouguet, Pyramidal implementation of the lucas kanade feature tracker description of the algorithm (2000).
URL `http://robots.stanford.edu/cs223b04/algo_tracking.pdf`

[33] M. Lucena, J. Fuertes, N. Blanca, Using optical flow for tracking, in: Progress in Pattern Recognition, Speech and Image Analysis, Vol. 2905 of Lecture Notes in Computer Science, 2003, pp. 87–94.

[34] G. Farneback, Two-frame motion estimation based on polynomial expansion, in: Image Analysis, 13th Scandinavian Conference, Vol. 2749 of Lecture Notes in Computer Science, 2003, pp. 363–370.

[35] J. Malcolm, Y. Rathi, A. Tannenbaum, Multi-object tracking through clutter using graph cuts, in: Non-Rigid Registration and Tracking Through Learning (in ICCV), 2007.

[36] J. Mooser, S. You, U. Neumann, Real-time object tracking for augmented reality combining graph cuts and optical flow, in: Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, 2007, pp. 145–152.

[37] A. Soudani, E. Zagrouba, People tracking based on predictions and graph-cuts segmentation, in: Advances in Visual Computing, Vol. 8034 of Lecture Notes in Computer Science, 2013, pp. 158–167.

[38] Z. Garrett, H. Saito, Live video object tracking and segmentation using graph cuts, in: IEEE International Conference on Image Processing, 2008, pp. 1576–1579.

[39] C. Prakash, P. Nalin, B. Stan, Adaptive fragments-based tracking of non-rigid objects using level sets, in: Proceedings of the International Conference on Computer Vision (ICCV), 2009, pp. 1530–1537.

[40] W. Shu, L. Huchuan, Y. Fan, Y. Ming-Hsuan, Superpixel tracking, in: Proceedings of the International Conference on Computer Vision (ICCV), 2011, pp. 1323–1330.

[41] C. Zhaowei, W. Longyin, L. Zhen, N. Vasconcelos, S. Li, Robust deformable and occluded object tracking with dynamic graph, IEEE Transactions Image Processing 23 (12) (2014) 5497–5509.

[42] A. Teichman, J. Lussier, S. Thrun, Learning to segment and track in rgbd, IEEE Transactions on Automation Science and Engineering 10 (4) (2013) 841–852.

[43] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[44] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: International Conference on Computer Vision Theory and Application, 2009, pp. 331–340.

[45] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[46] A. Yali, G. Donald, Shape quantization and recognition with randomized trees, Neural Computation 9 (1997) 1545–1588.

[47] B. A. Shepherd, An appraisal of a decision tree approach to image classification, in: Proceedings of International Joint Conference on Artificial Intelligence, 1983, pp. 473–475.

[48] F. Moosmann, B. Triggs, F. Jurie, Fast discriminative visual codebooks using randomized clustering forests, in: Proceeding of Advances in Neural Information Processing Systems, 2007.

[49] V. Lepetit, P. Lagger, P. Fua, Randomized trees for real-time keypoint recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 775–781.

[50] J. Shotton, M. Johnson, R. Cipolla, Semantic texton forests for image categorization and segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[51] Y. Boykov, O. Veksler, R. Zabih, A new algorithm for energy minimization with discontinuities, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops, 1999, pp. 26–29.

[52] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, in: Proceedings of the International Conference on Computer Vision (ICCV), 1999.

[53] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239.

[54] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 1124–1137.

[55] http:// darparoboticschallenge. org, Darpa robotics challenge website (February 2014).

[56] M. K. Kocamaz, Y. Lu, C. Rasmussen, Deformable object shape refinement and tracking using graph cuts and support vector machines, in: International Symposium on Visual Computing, 2011, pp. 506–515.

[57] S. Sclaroff, L. Liu, Deformable shape detection and description via model-based region grouping, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (5) (2001) 475–489.