3D Part-Based Sparse Tracker with Automatic Synchronization and Registration

Adel Bibi, Tianzhu Zhang, and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
adel.bibi@kaust.edu.sa, tianzhu.zhang@kaust.edu.sa, bernard.ghanem@kaust.edu.sa

Abstract

In this paper, we present a part-based sparse tracker in a particle filter framework where both the motion and appearance model are formulated in 3D. The motion model is adaptive and directed according to a simple yet powerful occlusion handling paradigm, which is intrinsically fused in the motion model. Also, since 3D trackers are sensitive to synchronization and registration noise in the RGB and depth streams, we propose automated methods to solve these two issues. Extensive experiments are conducted on a popular RGBD tracking benchmark, which demonstrate that our tracker can achieve superior results, outperforming many other recent and state-of-the-art RGBD trackers.

1. Introduction

Visual object tracking is a classical and very popular problem in computer vision with a plethora of applications such as vehicle navigation, human computer interface, human motion analysis, surveillance, and many more. The problem involves estimating the location of an initialized visual target in each frame of a video. Despite numerous object tracking methods that have been proposed in recent years [31, 25, 29, 26], most of these trackers suffer a degradation in performance mainly because of several challenges that include illumination changes, motion blur, complex motion, out of plane rotation, and partial or full occlusion, while occlusion is usually the most contributing factor in degrading the majority of trackers, if not all of them.

Fortunately, there is a recent surge in the availability of affordable and increasingly reliable RGBD sensors that provide image and depth data such as the Microsoft Kinect, Asus Xtion, and PrimeSense. When depth data is available, there exist more visual cues that can help resolve the nuisances of object tracking, especially occlusion. Recently, a relatively large RGBD dataset for visual tracking (100 videos) was compiled and released to the public [27] in the form of an online competition, where the ground truth object tracks are mostly suppressed. Since then, much deserved attention has been brought towards developing robust RGBD visual trackers. Although only a few of these

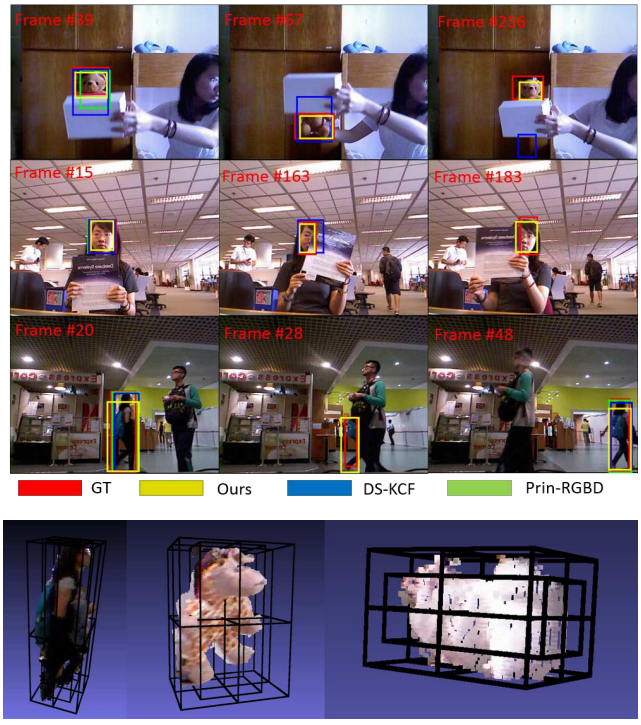

Figure 1. **Top**: Shows the tracking results on the videos "bear_front", "face_occ_5", and "new_ex_occ_4" respectively comparing our method, DS-KCF [9], and Princeton RGBD tracker [27]. **Bottom**: Shows 3D cuboids and their parts for different sequences as proposed in this paper.

trackers exist, they easily and with a big margin outperform state-of-art RGB trackers on the same videos. This is clear evidence of how depth information can be useful to visual tracking, especially regarding early and robust detection of occlusion and proper model update, both of which can significantly boost the performance of any tracker.

Along with providing an RGBD benchmark [27], Song *et al.* proposed an RGBD tracker that performs quite well on the benchmark. They adopt an exhaustive search paradigm coupled with an SVM classifier trained on image/depth HOG and point cloud features, which reduces the tracker's runtime to only 0.26 FPS. An occlusion handling method is also adopted, where the tracked object is assumed to be contained in the plane closest to the camera. This

method of handling occlusions makes it difficult to properly update the target's depth throughout the frames.

Recently, the work in [9] incorporated depth information into the popular Kernelized Correlation Filter tracker [14, 13]. This method demonstrates very promising performance with real-time speeds of up to 40 FPS. However, tracking is still done in the 2D image plane, which might not make use fully of the depth information. Also, an occlusion handling method was proposed that is very similar in spirit to [27], thus, suffering from similar difficulties.

Part-based RGB trackers have been prevalent for a while now. They demonstrate very desirable performance as compared to RGB trackers that use only one part, owing to the fact that an object can still be tracked as long as some (not necessarily) all its parts are visible in the next frame. This is most helpful in handling partial occlusions and recovering from complete occlusions. Inspired by this line of work [39, 38, 30], we propose a 3D part based-tracker, whereby the parts not only help in representing the target but also support the detection of partial or full occlusion.

When testing our tracker and other 3D methods on the RGBD Princeton benchmark [27], we realized that the provided RGB and depth sequences contain many synchronization and registration errors. A synchronization error occurs when an RGB frame is assigned to an incorrect depth frame in the sequence. This happens because the RGB and depth video streams are usually generated independently and although the frame rate of both cameras is similar, it might fluctuate slightly over time and there might also be dropped frames. This can cause substantial errors when tracking is done in 3D, since the projection to the image plane is affected. For example, even if the 3D tracking results are accurate, their resulting 2D bounding boxes, which are in turn used to evaluate the RGBD tracker, will not be since they are with respect to the depth image plane while the ground truth is in the RGB image plane. On the other hand, a registration error usually occurs due to imprecise calibration between the RGB and depth cameras. This imprecision manifests itself in erroneous assignment of depth values to RGB pixels. For example, background RGB pixels are assigned the target's depth values and vice versa. Due to both types of errors, the tracking performance of methods, which perform representation, sampling, and tracking in 3D, can be significantly affected. In fact, we suspect that this is the reason why the authors of a very recent RGBD tracker [9] mention that they mandate these errors be rectified (mostly probably manually) before tracking can be performed. In this paper, we suggest a simple yet effective method to automatically alleviate many of these errors that plague 3D tracking and possibly other applications that make use of RGBD stream data.

Contributions This work makes two main contributions. **(i)** To the best of our knowledge, we propose the first generic 3D part-based tracker with a very effective part-based occlusion handling method with desirable performance. At the time of submission, outranks all other trackers in the online Princeton RGBD benchmark [27]. **(ii)** We propose a simple method to synchronize and register RGBD videos for the purpose of single object tracking. We also provide to the vision community the manual synchronization and registration of the videos provided in the benchmark.

2. Related Work

RGB Trackers. In general, they can be divided into two main categories: discriminative and generative methods. Discriminative trackers formulate visual object tracking as a binary classification problem that searches for the target location that is the most distinctive from the background. Examples of discriminative trackers include multiple instance learning tracking (MIL) [5], ensemble tracking [4], support vector tracking [3], correlation filter based trackers [14, 13, 7, 6, 20, 18], multiple experts entropy minimization (MEEM) tracker [32] and other multi-object based trackers like the tracklet association with identity constraints [28]. On the other hand, a generative tracker searches for a candidate target that is best represented by its current appearance model. As such, the major contribution of this type of tracker is in developing suitable and versatile representative models that can reliably describe the object even when it undergoes different appearance changes. Examples of generative models include the mean shift [11], incremental (IVT) [23], fragment-based (Frag) [1], L1-min [21], multi-task (MTT) [35, 34], low-rank sparse [33], exclusive context modelling based tracker [36], occlusion detection based structural sparse learning based tracker [37] and structural sparse tracker [39].

RGBD Trackers. As for the RGBD domain, only a limited number of generic methods exist in the literature, owing to the novelty of the problem and the only recent availability of RGBD tracking data. In [27], a computationally expensive tracker is proposed that combines the SVM scores of two tracking-by-detection instances: one based on RGB and depth HOG features and the other is based on point cloud features. This scoring function is used to evaluate a dense sampling of 3D non-overlapping cells, thus, incurring a large computational cost. In our proposed tracker, we avoid this unnecessary computation induced by naive 3D sampling by exploiting the object's part-based structure as well as its previous motion. In doing so, only a very small number of 3D samples need to be evaluated at any given time. Also, the authors of [27] propose an expensive depth based histogram segmentation method (based on a strict assumption that the object lies in the nearest plane) to detect occlusion, unlike our tracker that infuses the occlusion handling scheme directly in the tracking process without any additional complexity.

Moreover, the conventional and high-speed KCF tracker

Figure 2. Overall pipeline for the proposed method, including the three major modules synchronization, registration, and 3D tracking.

[14, 13] was adapted to the RGBD domain in [9]. Similar to [27], occlusion is claimed to be handled by a segmentation of the depth histogram followed by a connected component analysis to incorporate spatial features of the object. As mentioned in the paper, the results of [9] on the Princeton benchmark relied on an apriori synchronization and realigning of the provided RGB and depth sequences. To the best of our knowledge, this process was most probably done manually, since no elaboration was given on the method used. In this paper, we show how this issue can be alleviated automatically. In [17], a 2D particle filter was adopted in which the sampling variance of each individual particle changes according to its occlusion state. Despite its good performance on the benchmark, its representation and motion models are both constructed in 2D, unlike our method that natively treats the problem in 3D, from both representation and motion perspectives. In [12], authors build an adaptive boosting classifier from a pool of features (color, depth, and grayscale) that is incrementally re-trained from frame-to-frame. Similar to the previous method, this tracker operates solely in 2D representations with no clear method of handling occlusion.

Other 3D trackers usually rely on a target-specific 3D object model, which is known apriori [30]. For example, the trackers of [22, 24] assume a 3D model of a hand is provided before tracking begins. Other methods like [19, 10] are category-specific RGBD trackers, which are mainly used for human detection and tracking.

In this paper, we propose a 3D particle filter tracker that exploits sparse representation, object structure (parts), as well as, adaptive particle sampling and pruning, all in a unified framework. We also present a simple yet effective method to automatically re-synchronize and re-register RGB and depth pairs for better tracking performance. Through extensive experiments, we show that our tracker outperforms all the state-of-the-art methods in Princeton RGBD benchmark by ranking first and third on the manually and automatically synchronized and registered data respectively in the online evaluation.

3. Part-Based Sparse 3D Tracker

In this paper, we propose a 3D part-based particle-filter tracker, where both representation and motion models are constructed entirely in 3D. We also propose a simple yet effective method for handling structural information provided by the object parts and how it is used for both representation and particle pruning. We represent target candidates (particles) and their parts using a linear sparse representation. Moveover, an occlusion handling scheme supported by the part representation is integrated with the motion model so as to adaptively guide/direct how particles are sampled in the next frame. As compared to tracking-by-detection methods that densely sample the 3D space, our strategy produces only a small number of particles needed for 3D tracking. As mentioned earlier, synchronized and registered data is exceptionally important for 3D trackers; therefore, we propose an automatic method to synchronize and register RGB-D sequences and apply it to those in the Princeton RGBD benchmark [27]. The overall pipeline of our method is illustrated in Figure 2.

3.1. Representation Model

Generative RGB trackers [39, 15, 21] and most RGBD trackers [17, 9] define their target candidates (e.g. particles) to be 2D bounding boxes in the RGB or depth image plane, from which features for representation are extracted. In our formulation, we use a particle filter to sample 3D cuboid candidates. To limit the number of particles to a practical number, we propose an adaptive data-driven sampling approach. Each particle cuboid is divided into a pre-defined number of overlapping parts, each of which is defined also as a 3D cuboid. For convenience, all particles have the same part layout. One of these parts covers 65% volume of the entire particle (cuboid) and is located at its center as seen in Figure 1. This part captures holistic object information, while the other smaller parts capture information from different sides of the object (refer to Figure 1). Clearly, other (possibly hierarchical) part layouts can be used here too. Each part is represented using a $m = 13$ dimensional feature: ten color names and three 3D shape features, as proposed in [27]. We model each particle part as a sparse linear combination of a set of dictionary elements. To do this, we adopt a similar approach as in [39, 15, 21]. At the first frame, we collect several observations of the object (and its parts) by sampling multiple cuboids around its ground truth location. We build $K = 2$ sparsifying dictionaries (using KSVD [2], one for each type of feature and for each part. Therefore, the representation of each particle can be described mathematically as follows:

$$\min_{\mathbf{X}_k} \sum_{k}^{K} \|\mathbf{D}_k\mathbf{X}_k - \mathbf{Y}_k\|_F^2 + \lambda\|\mathbf{X}_k\|_{1,1}, \qquad (1)$$

where $\mathbf{D}_k$ denotes the dictionary corresponding to the k^{th} feature type, such that $\mathbf{D}_k = [\hat{\mathbf{D}}_{1k}|\hat{\mathbf{D}}_{2k}|...|\hat{\mathbf{D}}_{Nk}|\mathbf{I}_{m_k}]$, where $\hat{\mathbf{D}}_{ik} \in \mathbb{R}^{m_k \times n_k}$, where m_k and n_k are the dimensionality of the k^{th} feature space and the number of atoms in dictionary k of the part i respectively. $\mathbf{I}_{m_k}$ is an identity matrix that encodes sparse error (e.g. partial occlusion); therefore, $\mathbf{D}_k \in \mathbb{R}^{m_k \times (Nn_k + m_k)}$. For computational reasons, these dictionaries are not updated with time. We concatenate the k^{th} feature type for all the parts of a particle in the observation matrix $\mathbf{Y}_k \in \mathbb{R}^{m_k \times N}$, where N is the total number of parts ($N = 9$ in our experiments). The resulting sparse code matrix is denoted as $\mathbf{X}_k \in \mathbb{R}^{(Nn_k + m_k) \times N}$. The optimization in Eq (1) is non-smooth but convex. It is the matrix form of the popular Lasso problem. It can be efficiently solved using a number of methods, including Alternating Direction Method of Multipliers (ADMM).

3.1.1 Temporal Coherence on Part Structure

Unlike sampling in 2D where the image plane is dense, most of the 3D point cloud scene is in fact empty. This may result in empty particle parts or completely empty particles. Scoring particles that are partially or completely empty using the objective in Eq (1) is not appropriate because the objective is not representative of these instances. That is, if one of the parts has disappeared, one of the two very different scenarios have occurred. Either the part is occluded, or the particle having that part is not representative enough. The first has to be associated with a low cost, while the latter with a high one. In Eq (1), an empty part (one of the columns of $\mathbf{Y}_k$ is the zero vector) will result in a the trivial solution of Eq (1) for both scenarios. That means the objective will favor parts that are empty; thus, leading the tracker to drift into completely empty regions in 3D space. Similarly, setting the cost too high for empty parts within a cuboid would favor dense regions and will lead to favoring any nearby object, even if it were the occluder.

To address this issue, we make use of the temporal coherence of the object's part structure, which is modeled using the distribution of 3D points within each part of the current target. For simplicity, each part is described with a single binary value (1 or 0), which depicts whether that part is empty or not. As such, each particle is described using an N-dimensional binary vector, denoted as the part-based structure feature. We expect that this feature changes gradually with time, i.e. many parts tend not to abruptly disappear or re-appear in the next frame. To preserve structural information in between consecutive frames and to determine occluded parts, we use the hamming distance between the binary structure feature of the current target and that of each particle sampled in the next frame. Only those particles with the minimum hamming distance are selected and represented using Eq (1). This strategy also helps prune unlikely particles, thus, substantially reducing the overall computation time needed for representation.

3.2. Motion Model

In this part, we give a detailed explanation of how we use particle filters in our tracker. But first, we give a brief of particle filtering.

3.2.1 Particle Filter

The particle filter is a Bayesian sequential important sampling technique for estimating a posterior distribution of state variables $\mathbf{x}_t$ that characterize a dynamic mode. It consists of two major steps, the prediction and the update for re-sampling. At the new instance t, a new observation $\mathbf{z}_t$ is available and the new probability distribution given all observations and the previous state is given by:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \qquad (2)$$

As for the update step, it is based on $p(\mathbf{z}_t|\mathbf{x}_t^i)$. The higher the probability is, the higher the weight is. As such, more particles will be sampled from a state that has been observed with a higher probability. It is computed as follows:

$$w_t^i = w_{t-1}^i p(\mathbf{z}_t|\mathbf{x}_t) \qquad (3)$$

3.2.2 Adaptive Directed Sampling

Owing to the way objects move in 3D and since our particles are modeled as cuboids, we take the state vector $\mathbf{x}_t$ to represent a 3D rigid body transformation, i.e. $\mathbf{x}_t \in \mathbb{R}^6$, which is characterized by a 3D rotation and 3D translation. Note that scale could be incorporated in this setup; however, in most cases and unlike the 2D scale, the 3D scale of a target does not change dramatically from its initial value. Depending on the state of the object (whether it is occluded or not), the motion model changes accordingly.

No occlusion. To not sample particles unnecessarily, we do *not* use a zero-order motion model. Instead, we use 2D optical flow on the current target to the next RGB frame to compute a crude estimate of its new 3D location. Since only a crude estimate is needed, most optical flow methods can be used here. So, for mainly computational reasons, the basic Horn-Schunck optical flow [16] is sufficient. In fact, we experimented with more sophisticated large-displacement methods (e.g. the work in [8]), only to find that the tracking performance is only subtly affected. Given the pixel correspondences from optical flow, we can get a crude estimate of the rigid body transformation (rotation $\mathbf{R}$ and translation $\mathbf{t}$) of the current target. Then, we sample the particle states

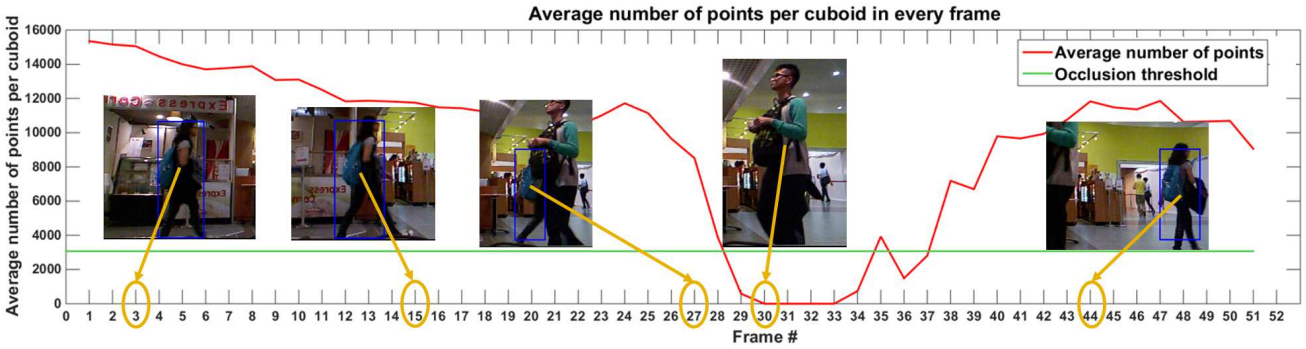

Figure 3. Shows the proposed occlusion handling scheme on video "new_ex_occ_4 ", where the number of points in all the particles decreases significantly in 3D space when the tracked object is occluded.

using a Gaussian distribution centered at the previous estimate $(\mathbf{R}, \mathbf{t})$ and with a diagonal covariance. Since the orientation of a target cuboid does not change much over time (e.g. a human walking on a planar surface), we set the variances of the translation parameters to be much larger than those of the rotation.

Occlusion. When the target is determined to be in an occlusion state (as we will describe in the next section), optical flow is no longer a valid measure. Therefore, we resort back to a zero-mean motion model with large translational variance, so as to recapture the target when it re-appears.

3.2.3 Occlusion Handling

We integrate the occlusion handling scheme with the particle filter formulation. As discussed earlier, each sampled particle represents a cuboid, which contains a certain number of 3D points. In case of occlusion, we observe that this number tends to decrease significantly in all particles all at once. By monitoring this change, we can determine if an object is being occluded or not. Since image resolution is inversely proportional to the distance from the camera, the number of points in a cuboid is also inversely proportional with depth. Therefore, we need to compensate for the number of points in depth by computing a depth-normalized measure: $\tilde{t}^j = \left(\frac{z^j}{z^1}\right)^2 t^j$. Here, z^j and z^1 are the average depth values in the 3D cuboids in frame j and the first frame respectively. t^j and $\tilde{t}^j$ are the original and the depth adjusted average number of points in the current 3D cuboid in frame j. In case of an occlusion at frame j, the depth-normalized number of points among all particles will be very low. If the average depth adjusted falls below some threshold compared to the average number of points in the first frame, the object is identified to be in an occlusion state as illustrated in Figure 3.

3.3. Synchronizing and Registering RGBD Images

There are several videos in Princeton RGBD benchmark [27] with registration and synchronization issues that can substantially affect 3D tracking performance. We consider this as a fundamental problem that needs to be addressed because it not only affects 3D trackers but many other computer vision applications that involve operations on 3D data. In fact, it is clearly stated in previous work [9] that the RGB and depth sequences need to be synchronized and realigned properly before tracking can be applied. In the following, we propose a method to solve both problems.

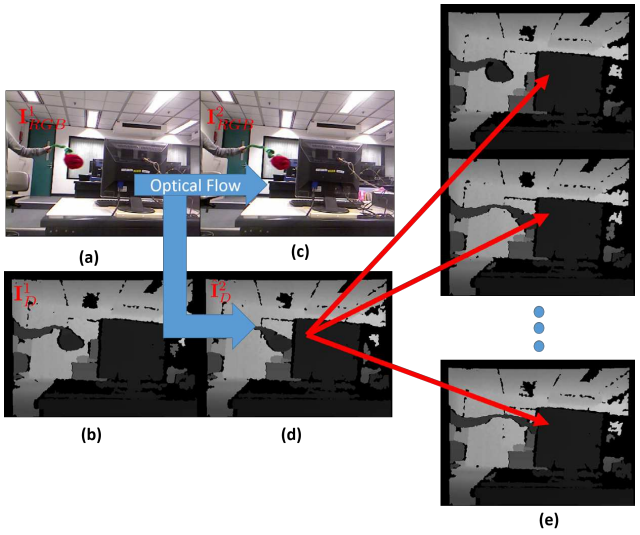

Figure 4. (a)&(b) is a previously synchronized RGBD image pair. (c)&(d) is an RGBD pair formed of the next RGB image in the sequence and its corresponding synthetic depth image generated using optical flow. (e) shows a set of possible matches to (c) most similar to (d).

3.3.1 Synchronization

As explained earlier, synchronization problems arise in RGBD videos because the RGB and depth streams tend to be acquired independently, with different frame rates, and sometimes frames are dropped. A commonly used method to resolve this issue is to assign a depth image to every RGB image that has the closest time stamp. This is what is done

in the benchmark [27]. Obviously, this issue can also be resolved from a hardware perspective by simply increasing the frame rate of both cameras, thus, reducing the effect of time stamp offset. In this work, we formulate the problem as a matching task to alleviate the synchronization issues.

We take the RGB sequence as reference. We seek to match a depth image to every image in the RGB sequence. An RGB image is allowed to match to any depth image, whose time stamp is close enough to that of the RGB image. We start with a manually synchronized pair of RGB and depth images in the first frame, denoted as $\mathbf{I}^1_{RGB}$ and $\mathbf{I}^1_D$ respectively. Under the assumption that the depth values of a scene change smoothly between consecutive frames, we can compute a large-displacement optical flow [8] between the current synchronized RGB image $\mathbf{I}^1_{RGB}$ and the next one in the sequence $\mathbf{I}^2_{RGB}$. Only point correspondences $\{(\mathbf{x}^1_i, \mathbf{y}^2_i)\}_{i=1}^p$ with a large enough flow magnitude are maintained, since moving points give clear indication of when synchronization errors occur. Then, we generate a synthetic depth image $\hat{\mathbf{I}}^2_D$ by transferring over depth values, i.e. $\hat{\mathbf{I}}^2_D(\mathbf{y}^2_i) = \mathbf{I}^1_D(\mathbf{x}^1_i)$. If $\mathcal{C}$ is the set of frame indices identifying the depth images that are close in time stamp to $\mathbf{I}^2_{RGB}$, then we can synchronize $\mathbf{I}^2_{RGB}$ to the depth image $\mathbf{I}^{j^*}_D$, whose depth values at $\{\mathbf{y}^2_i\}_{i=1}^p$ are closest to those in $\hat{\mathbf{I}}^2_D$. We formulate this mathematically as follows:

$$j^* = \underset{j \in \mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{p} \left(\hat{\mathbf{I}}^2_D(\mathbf{y}^2_i) - \mathbf{I}^j_D(\mathbf{y}^2_i) \right)^2 \tag{4}$$

This strategy can be applied iteratively (refer to Fig. 4) until all the images in the RGB sequence are synchronized.

3.3.2 Registration

Several videos in the Princeton RGBD benchmark [27] also suffer from incorrect registration between synchronized RGB and depth image pairs. Being able to correctly register these pairs is very important, since this registration defines the correspondences between pixels in the RGB image and those in the depth image, which in turn enable the generation of the 3D point cloud of the scene.

The usual strategy for registering these image pairs is to stereo calibrate both the depth and RGB cameras together. This calibration can be used to map pixels from one of them to the other. This strategy is used in the RGBD benchmark [27]; however, in many cases, registration errors do occur and they could be as large as 30-40 pixels. Such an offset can negatively affect any RGBD based tracker, especially one that tracks the object in 3D. This offset is not uniformly random at each pixel, since the source of the error arises from perturbations in the extrinsic calibration parameters, i.e. the rotation and translation that transforms the coordinate system of the RGB camera to that of the depth. By assuming that the perturbation in the rotation is negligible w.r.t. the perturbation in the translation, it can be shown that the per-pixel offset in registration varies with the depth and image location of the pixel (refer to **supplementary material**). However, this variation is structured. For example, neighboring pixels with similar depth values tend to have very similar offsets. This is why some of the registration errors in the benchmark (corresponding to videos showing a predominant foreground object infront of a far away background) can be easily fixed by simply translating the whole depth image in a single direction. To estimate the registration offsets from one frame to the next, we formulate a structured selection problem as shown in Eq (5). We only require the first RGBD image pair to be correctly registered.

$$\begin{aligned} \min_{\mathbf{z}_i \forall i} \quad & \frac{1}{p} \sum_{i}^{p} (d^i_1 - \mathbf{d}^{i\,T}_2 \mathbf{z}_i)^2 \\ s.t \quad & \mathbf{z}_i \in \{0,1\}^N \; \forall i, \;\; \mathbf{1}^T \mathbf{z}_i = 1 \; \forall i, \; \text{rank}(\mathbf{Z}) \leq r \end{aligned} \tag{5}$$

Here, d^i_1 is the depth of pixel i in the previous RGBD image pair, which is assumed to be correctly registered. We assume that the offset of this pixel in the next frame can be one of N possible offsets. The vector $\mathbf{d}^i_2 \in \mathbb{R}^N$ contains the depth values in the next depth image corresponding to the N possible offsets. Moreover, the binary vector $\mathbf{z}_i$ can be viewed as a selection vector that chooses only one of the offset depth values in the next frame for pixel i. The selection vectors $\mathbf{Z} = [\mathbf{z}_1|...|\mathbf{z}_p]$ for the p pixels (i.e. matched pixels using optical flow) must be constrained to follow the perturbation model described above. Since neighboring pixels with similar depths tend to have the same offset, we only expect a small number of distinct offsets to be selected among the p pixels. We formulate this as a low-rank constraint on the binary matrix $\mathbf{Z}$.

Computing the global solution of this binary problem cannot be done in polynomial time, so we seek a tradeoff between solution quality and computational efficiency. We observe that when $r = 1$, any feasible matrix $\mathbf{Z}$ is all zeros except for an entire row. We can exhaustively evaluate all N feasible matrices and return the one leading to the smallest objective value. This is equivalent to selecting a single offset for all p pixels. We can exploit this observation to efficiently compute an approximate solution when $r > 1$. This can be done in two ways. **(i)** We can pre-cluster the p pixels (e.g. according to their depth values d^i_1 and their image locations) and then find the best rank-1 solution for each cluster independently. **(ii)** We first find the best rank-1 solution and remove all pixels whose offset has been found (i.e. their contribution to the overall objective is zero). Then, we reiterate this process on the remaining pixels at most $(r-1)$ times or until no pixels are left.

4. Experimental Results

To evaluate the performance of our proposed tracker, we conduct extensive experiments on the well-known Prince-

Algorithm	Target type			Target Size		Movement		Occlusion		Motion Type	
	Human	Animal	Rigid	Large	Small	Slow	Fast	Yes	No	Passive	Active
Ours_Manual	0.81	0.64	0.73	0.80	0.71	0.75	0.75	0.73	0.78	0.79	0.73
Ours_Sync_Reg	0.74	0.66	0.70	0.77	0.65	0.76	0.68	0.67	0.76	0.75	0.69
Ours_Sync_Raw_Reg	0.68	0.58	0.71	0.76	0.61	0.74	0.64	0.62	0.74	0.75	0.64
Ours_Raw_Sync_Raw_Reg	0.64	0.57	0.67	0.71	0.58	0.73	0.60	0.57	0.73	0.72	0.61
Ours_One_Part_Manual	0.60	0.56	0.46	0.58	0.50	0.58	0.52	0.48	0.62	0.54	0.54

Table 1. Comparison between our proposed tracker in different design variations.

Algorithm	Avg. Rank	Target Type			Target Size		Movement		Occlusion		Motion Type	
		Human	Animal	Rigid	Large	Small	Slow	Fast	Yes	No	Passive	Active
Ours_Manual	2.27	0.81	0.64	0.73	0.80	0.71	0.75	0.75	0.73	0.78	0.79	0.73
OAPF[17]	2.63	0.64	0.85	0.77	0.73	0.73	0.85	0.68	0.64	0.85	0.78	0.71
RGBDOcc+OF[27]	2.81	0.74	0.63	0.78	0.78	0.70	0.76	0.72	0.72	0.75	0.82	0.70
Ours_Sync_Reg	3.72	0.74	0.66	0.70	0.77	0.65	0.76	0.68	0.67	0.76	0.75	0.69
DS-KCF[9]	4.54	0.67	0.61	0.76	0.69	0.70	0.75	0.67	0.63	0.78	0.79	0.66
RGBD+OF[27]	5.27	0.64	0.65	0.75	0.72	0.65	0.73	0.66	0.60	0.79	0.74	0.66
PCdet_flow[27]	7.27	0.51	0.52	0.73	0.63	0.56	0.74	0.53	0.55	0.64	0.75	0.53

Table 2. Tracking results from the online evaluation for our tracker on both the manually and automatically synchronized and registered data compared with the top 5 trackers.

ton RGBD benchmark [27], comprising a total of 100 video sequences only five of which have publicly available ground truth tracks. The videos contain many challenges including partial and full occlusion, fast motion, out of plane rotation, background clutter, moving camera, and shape deformation and distortion. For comparison, the authors of [27] provide an online evaluation system that compares a tracker's performance with that of 20 others, 12 of which use depth information while the rest are popular state-of-the-art RGB trackers. The evaluation for the comparisons among trackers was based on the intersection over union criterion (IOU). For the details of the tracking criterion used for evaluation, reader is referred to [27]. To evaluate the merits of our synchronization and registration method, we compare our tracker on both the RGBD data provided in the benchmark and the same data after both methods are applied. Moreover, we empirically validate the benefits of using multiple parts instead of a holistic representation for tracking.

4.1. Implementation Details

All our experiments are done using MATLAB R2014b on a 3.07GHz Intel(R) Xeon(R) with 48GB RAM. The number of parts $N = 9$, with $K = 2$ dictionaries learnt for each part. Only a total of 20 particles were used in our method. As for particle sampling, we set the standard deviations of the translation component to $\{0.05, 0.05, 0.03\}$ for when the object is not in a state of occlusion and $\{0.35, 0.10, 0.2\}$ when it is. To reduce computational cost, we do not sample the rotation obtained from point correspondences. Moreover, we set the Lasso parameter $\lambda = 0.05$ as a decent tradeoff between sparsity and meaningful reconstruction. The target is in an occlusion state, when the depth-normalized average number of points in a frame for all particles falls below $0.2\bar{\mathbf{t}}^1$, where $\bar{\mathbf{t}}^1$ is the average number of points inside the cuboid in the first frame.

As for synchronization, any RGB image is allowed to match to one depth image that is within 5 frames from it.

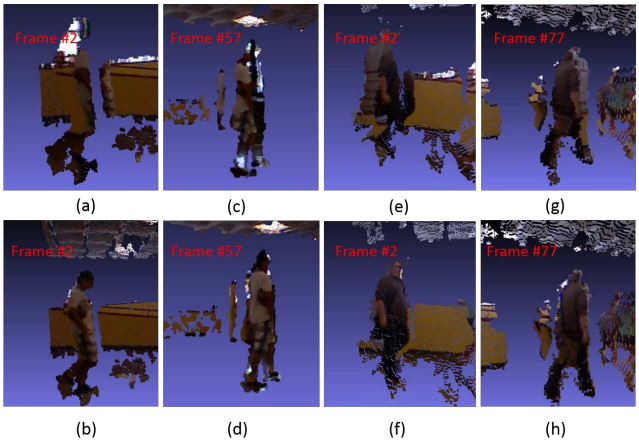

Figure 5. Pairs (a)&(b) and (c)&(d) denote the registered RGB and depth pairs as provided in the benchmark [27], compared with our method of registration on the video "new_student_center_no_occ" at frames 2 and 57 respectively. Similarly, we show pairs (e)&(f) and (g)&(h) for the video "new_student_center_3 " at frames 2 and 77 respectively.

For the registration problem, we use the offset in the previous frame as an initialization for the next one.

4.2. Evaluation of Design Choices

In the following, we show the impact on tracking performance of using the automatically synchronized and registered data and compare it to the provided benchmark data [27]. In the defense of parts, we see a significant performance improvement when multiple parts are considered.

Synchronization results. As discussed earlier, around 14% of Princeton RGBD benchmark [27] videos have synchronization error, while an additional 8% also require re-registration. Since this is very important for 3D based trackers, Table 1 shows our proposed tracker results on the RGB-D ground truth data provided by [27] and the same data after

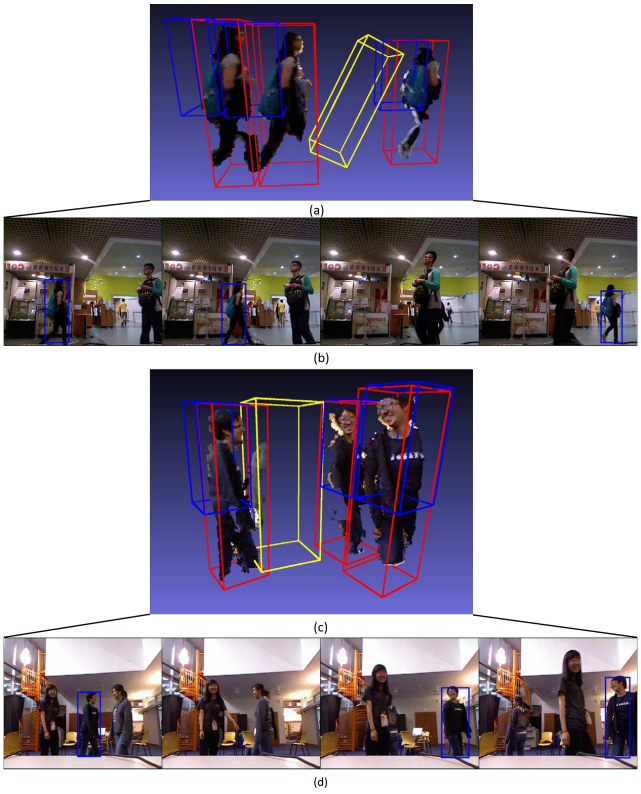

Figure 6. Images (a)&(c) show the tracking results in 3D where the red and blue cuboids are two different parts tracked. The yellow cuboid is an occlusion detection. Images (b)&(d) show the corresponding 2D tracking results. The videos from top to bottom are "new_ex_occ _4 "and "three_people "respectively.

our proposed method of synchronization is used. For both the fast motion and occlusion categories, we observe a notable improvement in performance (4% and 5% respectively), since the un-synchronized RGB and depth image pairs in these categories tend to be significantly different.

Registration results. Figure 5 illustrates the problem of registration with a qualitative example. Around 12% of the testing videos provided by [27] suffer from severe registration errors. Table 1 shows tracking results on the videos as given by [27] and on the data registered using our proposed method. Significant improvement in performance is noted in almost all categories Again, this is mainly because 3D based trackers project the points back into the depth frame's reference, while the evaluation ground truth is constructed in the RGB image. Moreover, unregistered data contaminates the representation of the target because color attributes of the background appear in the target and vice versa.

In the defense of parts. Table 1 summarizes the performance of our method when only one holistic part is used. Clearly, adding multiple parts substantially improves performance across *all* tracking categories. It is evident that parts help provide a more robust representation of the target, as well as, structural information that is important to prune unnecessary and possibly confusing particles.

4.3. Results on the RGBD Tracking Benchmark

In Table 2, we summarize the performance of our tracker on both the manually and automatically synchronized and registered data compared to the top-5 closest trackers on a total of 95 videos. We use the same category breakdown as the online benchmark. The list of competing trackers was shortened to show only the best 5 methods (refer to the **supplementary material** for the entire table). Our part-based sparse tracker ranks first among all other methods on the manually synchronized and registered data while ranking third when the automatically synchronized and registered data is used. In some categories (e.g. Human), it registers an improvement of 7% over the second best tracker. This is attributed to the use of parts and the temporal coherence in their structure, which is a reasonable assumption for humans. Other categories (e.g. Animal and Fast Motion) show the target interacting in very close distance with other objects that have similar cues and/or exhibiting complex motion, thus, restricting tracking performance.

Qualitative results. Figure (6) shows the tracking results in 3D of sample frames taken from two benchmark videos, namely "new_ex_occ_4 "and "three_people". Both of these videos include examples of full occlusion. For visualization purposes, we only show two of the nine constituent parts, shown as red and blue cuboids. We also backproject these cuboids into the image plane as upright bounding boxes. The yellow cuboids are instances when the object is determined to be in an occlusion state. Notice how our tracker is able to easily recover from this full occlusion.

5. Conclusion

In this paper, we proposed a 3D part-based sparse tracker, which exploits parts to preserve temporal structural information and help in particle pruning. A fast yet powerful method was proposed to embed occlusion detection in the motion model framework. Since 3D trackers are sensitive to synchronization and registration noise, we proposed methods to correct for both, especially since no less than 30% of the videos on the popular RGBD tracking benchmark [27] suffer from these issues. Extensive experiments demonstrate the positive impact of each module of the proposed tracker. In fact, our tracker currently ranks first on the benchmark, as compared to many state-of-the-art trackers. For future work, we aim to exploit part-to-part spatial and appearance relationships to encode particles and to detect occlusion. Moreover, we plan to develop a strategy to incrementally build and maintain a prototypical 3D model of the target by registering its tracking results with time.

Acknowledgments. Research in this publication was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE Computer Vision and Pattern Recognition (CVPR), 2006.*, volume 1, pages 798–805. IEEE, 2006. 2

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing (TSP), 2006.*, 54(11):4311–4322, 2006. 3

[3] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2004.*, 26(8):1064–1072, 2004. 2

[4] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2007.*, 29(2):261–271, 2007. 2

[5] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR), 2009.*, pages 983–990. IEEE, 2009. 2

[6] A. Bibi and B. Ghanem. Multi-template scale-adaptive kernelized correlation filters. In *IEEE International Conference on Computer Vision Workshops (ICCVW), 2015.*, pages 50–57, 2015. 2

[7] D. S. Bolme, J. R. Beveridge, B. Draper, Y. M. Lui, et al. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.*, pages 2544–2550. IEEE, 2010. 2

[8] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.*, pages 41–48. IEEE, 2009. 4, 6

[9] M. Camplani, S. Hannuna, M. Mirmehdi, D. Damen, A. Paiement, L. Tao, T. Burghardt, and U. Bristol. Real-time rgb-d tracking with depth scaling kernelised correlation filters and occlusion handling. In *IEEE British Machine Vision conference (BMVC), 2015.*, 2015. 1, 2, 3, 5, 7

[10] W. Choi, C. Pantofaru, and S. Savarese. Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *IEEE International Conference on Computer Vision Workshops (ICCVW), 2011.*, pages 1076–1083. IEEE, 2011. 3

[11] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2003.*, 25(5):564–577, 2003. 2

[12] G. M. García, D. A. Klein, J. Stückler, S. Frintrop, and A. B. Cremers. *Adaptive multi-cue 3D tracking of arbitrary objects*. Springer, 2012. 3

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *booktitle=IEEE European Conference on Computer Vision (ECCV), 2012., pages=702–715, year=2012, publisher=Springer*. 2, 3

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015.*, 37(3):583–596, 2015. 2, 3

[15] Z. Hong, X. Mei, D. Prokhorov, and D. Tao. Tracking via robust multi-task multi-view joint sparse representation. In *IEEE International Conference on Computer Vision (ICCV), 2013.*, pages 649–656. IEEE, 2013. 3

[16] B. K. Horn and B. G. Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981. 4

[17] S. O. H. S. Y. L. Kourosh Meshgi, Shin-ichi Maeda and S. Ishii. Occlusion aware particle filter tracker to handle complex and persistent occlusions. *Computer Vision and Image Understanding (CVIU)*, 2015 [Under Review]. 3, 7

[18] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *IEEE European Conference on Computer Vision Workshops (ECCVW), 2014.*, pages 254–265. Springer, 2014. 2

[19] M. Luber, L. Spinello, and K. O. Arras. People tracking in rgb-d data with on-line boosted target models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.*, pages 3844–3849. IEEE, 2011. 3

[20] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.*, pages 5388–5396, 2015. 2

[21] X. Mei and H. Ling. Robust visual tracking using &# x2113; 1 minimization. In *IEEE International Conference on Computer Vision (ICCV), 2009.*, pages 1436–1443. IEEE, 2009. 2, 3

[22] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *IEEE British Machine Vision conference (BMVC), 2011.*, volume 1, page 3, 2011. 3

[23] T. Poggio and G. Cauwenberghs. Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems (NIPS), 2001.*, 13:409, 2001. 2

[24] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.*, pages 1106–1113. IEEE, 2014. 3

[25] S. Salti, A. Cavallaro, and L. D. Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Transactions on Image Processing (TIP), 2012.*, 21(10):4334–4348, 2012. 1

[26] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2014.*, 36(7):1442–1468, July 2014. 1

[27] S. Song and J. Xiao. Tracking revisited using rgbd camera: Unified benchmark and baselines. In *IEEE International Conference on Computer Vision (ICCV), 2013.*, pages 233–240. IEEE, 2013. 1, 2, 3, 5, 6, 7, 8

[28] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous clustering and tracklet linking for multi-face tracking in videos. In *IEEE International Conference on Computer Vision (ICCV), 2013.*, pages 2856–2863, 2013. 2

[29] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.*, pages 2411–2418. IEEE, 2013. 1

[30] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese. Monocular multiview object tracking with 3d aspect parts. In *IEEE European Conference on Computer Vision (ECCV), 2014.*, pages 220–235. Springer, 2014. 2, 3

[31] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM computing surveys (CSUR)*, 38(4):13, 2006. 1

[32] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *IEEE European Conference on Computer Vision (ECCV), 2014.*, pages 188–203. Springer, 2014. 2

[33] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *IEEE European Conference on Computer Vision (ECCV), 2012.*, pages 470–484. Springer, 2012. 2

[34] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.*, pages 2042–2049. IEEE, 2012. 2

[35] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *International Journal of Computer Vision (IJCV), 2013.*, 101(2):367–383, 2013. 2

[36] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja. Robust visual tracking via exclusive context modeling. *IEEE Transactions on Cybernetics, 2016.*, 46(1):51–63, 2016. 2

[37] T. Zhang, B. Ghanem, C. Xu, and N. Ahuja. Object tracking by occlusion detection via structured sparse learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2013.*, pages 1033–1040, 2013. 2

[38] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja. Partial occlusion handling for visual tracking via robust part matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.*, pages 1258–1265. IEEE, 2014. 2

[39] T. Zhang, S. Liu, C. Xu, S. Yan, B. Ghanem, N. Ahuja, and M.-H. Yang. Structural sparse tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015*, pages 150–158, 2015. 2, 3