3D Tracking of Multiple Objects with Identical Appearance using RGB-D Input

Carl Yuheng Ren†, Victor Prisacariu†, Olaf Kaehler†, Ian Reid‡ and David Murray†
†Department of Engineering Science, University of Oxford
‡School of Computer Science, University of Adelaide
†{carl,victor,olaf,dwm}@robots.ox.ac.uk
‡ian.reid@adelaide.edu.au

Abstract

Most current approaches for 3D object tracking rely on distinctive object appearances. While several such trackers can be instantiated to track multiple objects independently, this not only neglects that objects should not occupy the same space in 3D, but also fails when objects have highly similar or identical appearances. In this paper we develop a probabilistic graphical model that accounts for similarity and proximity and leads to robust real-time tracking of multiple objects from RGB-D data, without recourse to bolt-on collision detection.

1. Introduction

Tracking 3D object pose over time is a core task in computer vision, and one which has been the matter of sustained research over three decades. For much of that time, 3D model-based vision was concerned with tracking rigid entities having a simple and predefined geometrical description and using sparse features computed from 2D imagery. However, work over the last few years has seen fundamental changes in every aspect: from the use of learned, geometrically complex, and sometimes non-rigid objects, to the use of dense and rich representations computed from combined conventional image and depth cameras.

A key limitation of current RGBD-based 3D trackers (e.g. [14]) is their focus on *single* objects, with the extension to multiple objects often being cast as a straightforward replication of independent object trackers. We believe "straightforward" to be a gross simplification: it ignores two pitfalls commonly encountered in real-world scenarios. First is similarity in appearance, where objects have similar colour and shape (cars are usually followed by more cars, not by elephants) and second is the physical constraint that exists between multiple rigid bodies, chiefly that they must not intersect one another. In this work we address these two problems and propose an RGBD-based tracker that can recover the 3D pose of multiple objects with *identical* appearance, while preventing them from intersecting.

The paper is structured as follows. Section 2 gives an overview of related work. Section 3 describes our probabilistic formulation of the multiple object tracking problem, and Section 4 provides experimental insight into its operation. Conclusions are drawn in Section 5.

2. Related Work

We begin this section by covering the general theme of model-based 3D tracking. Next, we consider more specialized works that use distance transforms. Finally, we detail methods that aim to impose physical constraints.

Model based 3D tracking. Most existing research on 3D tracking with or without depth data uses a model-based approach that is probabilistically estimating a state by minimizing some objective function measuring the discrepancy between the expected and observed image cues. While early works out of necessity exploited highly sparse data such as points and edges (e.g. [5, 4, 8]), a common algorithm deployed on denser data is Iterative Closest Point (ICP) [1]. In [6], the authors input RGB-D imagery from Kinect and use ICP to track hand-held 3D rigid puppets. The system yields robust and real-time performance, but occlusion introduced by the hand has to be carefully managed through a colour-based pre-segmentation phase. Awkwardly, a different appearance model is required to achieve pre-segmentation when tracking multiple objects. A more general work is KinectFusion [9], where the entire scene structure along with camera pose are estimated simultaneously. Ray-casting is used to establish point correspondences, after which estimation of alignment or pose is achieved with ICP. However, a key requirement when tracking with KinectFusion is a static scene, or in our context a single object, a condition which is obviously violated when tracking multiple independently moving objects. Another cluster of RGB-D based tracking works uses sampling, and relies on evaluations of the objective function at many positions in the state space. For example, in [11] the authors use Particle Swarm Optimization to track an articulated hand, whereas [18]

instead uses a particle filter to estimate pose. Relying on a large number of objective evaluations is computational taxing, even given parallel GPU implementations as in [11].

Signed distance functions for tracking. An alternative to ICP is the use of the signed distance function (SDF). It was first shown in [3] that distance transforms could be used efficiently to register 2D/3D point sets. The SDF was used in [15] to formulate different embedding functions for robust real-time 3D tracking of rigid objects using only depth data, an approach extended in [14] to leverage RGB data in addition. A similar idea is described in [17], where the authors directly use the gradient of the SDF for tracking camera pose. Using only image data, the authors in [13] project a 3D model into the image domain to generate a SDF-like embedding function, and the 3D pose of a rigid object is recovered by evolving this embedding function. KinectFusion [9] also uses a truncated SDF for shape representation, but, as noted earlier, it uses ICP for camera tracking, instead of directly exploiting the SDF. As shown in [17], ICP is less effective for this task.

Tracking multiple objects with physical collision constraints. Physical constraints between 3D objects are usually enforced by reducing the number of degrees of freedom (dof) in the state. An elegant example of tracking of connected objects (or sub-parts) in this way is given in [2]. However, when tracking multiple independently moving objects, physical plausibility is violated suddenly and intermittently by the collision of objects: these constraints cannot be conveniently enforced by reparameterization. Indeed, rather few works model the physical collision between objects. In [10], the authors track two interacting hands with Kinect input. A penalty term measuring the inter-penetration of fingers is introduced to invalidate impossible articulated poses. In [12, 7], a hand and a moving object are simultaneously tracked, and invalid configurations similarly penalized. In both cases, the used measure is the minimum magnitude of 3D translation required to eliminate intersection of the two objects, which is computed using the Open Dynamic Engine library [16]. In contrast, in our proposed method, the collision constraint is more naturally enforced through a probabilistic generative model, without the need of an additional constraint engine.

3. Methodology

The theoretical underpinning of our multi-object tracker is a probabilistic model. After establishing notation in §3.1, we describe this model in §3.2, and detail inference on it in §3.3 and §3.4. The optimization method and an additional online learning of appearance models are summarized in §3.5 and §3.6.

3.1. Notation

For clarity, we introduce the notation using single object tracking, as illustrated in Fig. 1. Let Ω_d and Ω_c be the depth and colour image domains, respectively, and assume known calibration so that the aligned and combined RGB-D image domain, Ω is readily derived. A pixel $x^{\mathsf{img}} \in \Omega$ at image coordinates (u, v) has depth value d and colour value y. We use a dot to denote the homogeneous coordinates of x^{img}: $\dot{x}^{\mathsf{img}} = (ud, vd, d)^\top \in \mathbb{R}^3$. A pixel x^{img} in the object region is projected from a 3D point $\dot{X}^{\mathsf{obj}} = (X, Y, Z, 1)^\top$ on the object surface in object coordinates as:

$$\dot{x}^{\mathsf{img}} = \mathtt{A} X^{\mathsf{cam}} , \qquad X^{\mathsf{cam}} = \mathtt{T}^{\mathsf{c,o}} \dot{X}^{\mathsf{obj}} \tag{1}$$

where $\mathtt{A}$ is the depth camera's intrinsic matrix, and $\mathtt{T}^{\mathsf{c,o}} = [R|t]_{3\times4} \in \mathbb{SE}_3$ is the Euclidean transformation taking a 3D point $\dot{X}^{\mathsf{obj}}$ from object coordinates to camera coordinates. By $\mathtt{T}^{\mathsf{o,c}} \in \mathbb{SE}_3$ we denote the inverse transformation. For computational efficiency, our implementation parametrizes the transformation $\mathtt{T}^{\mathsf{o,c}}$ by a 6-dimensional pose vector p, with rotation encoded by modified Rodrigues parameters.

We represent object models by a 3D SDF, Φ, defined in a local space around each object. The surface of the 3D shape is recovered as the zero level, $\Phi = 0$, and the regions outside and inside the object map to positive and negative values of Φ, respectively. We use the "bag-of-voxels" representation given in [14]: the i-th voxel $\{X, V\}_i$ comprises its 3D location X_i and an indicator V_i that can take on values *on* or *out* for *on the surface*, or *outside the shape*, respectively. Note that V_i never indicates *in*: a voxel inside the object cannot generate a pixel in Ω.

We focus here on the challenging task of tracking multiple similar or identical objects, and without loss of generality assume that all tracked objects share the same appearance model. Only two such appearance models are then needed to describe the colour statistics of the scene — one for the object surface, which generates the foreground region in the image; and one for the background. Both are represented by their likelihoods, $P(y|V)$, where V, as noted above, has values on or out only. The two appearance models are represented with RGB colour histograms. The histograms are initialized either from an object detector or from a user-selected bounding box on the RGB-D image, in which the foreground model is built from the interior of the bounding box and the background from the immediate region outside the bounding box. Once initialized, the histograms are updated continuously while tracking.

3.2. Generative Model

The graphical model motivating our multi-object tracking algorithm is shown in Fig. 1 (left). Generating a RGB-D image Ω should be conditionally dependant on the set of 3D object shapes $\{\Phi_1 ... \Phi_M\}$ and their corresponding set

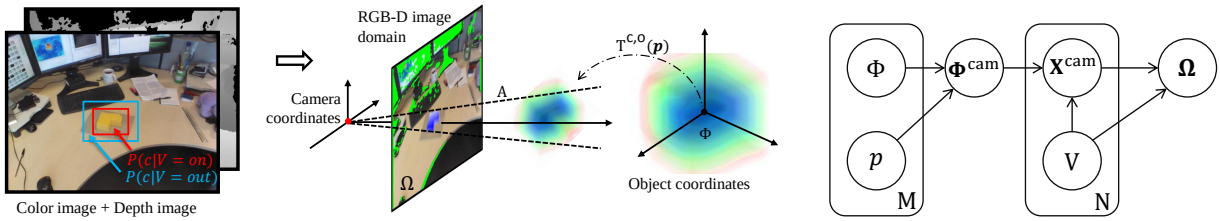

Figure 1. Left: representation of the 3D model $\mathbf{\Phi}$, the RGB-D image domain $\mathbf{\Omega}$, the surface/background models $P(y|V = on)$, $P(y|V = out)$ and the pose $\mathsf{T}^{\mathsf{c,o}}(p)$. Right: graphical model of our multi-object tracker.

of poses $\mathbf{p} = \{p_1...p_M\}$ at the time the image was taken. To make the generative process more intuitive, we introduce an intermediate variable $\mathbf{\Phi}^{\mathsf{cam}}$, which is the union of all 3D object shapes in *camera coordinates*. The generative process follows: the set of 3D shapes $\{\Phi_1...\Phi_M\}$ and their corresponding set of poses $\mathbf{p}$ first generate a 'shape union' $\mathbf{\Phi}^{\mathsf{cam}}$ in camera coordinates, then $\mathbf{\Phi}^{\mathsf{cam}}$ generates a set of voxels $\{X^{\mathsf{cam}}, V\}$, again in camera coordinates. Finally, the observed RGB-D image domain Ω is generated from $\{X^{\mathsf{cam}}, V\}$. Note that when the number of objects $M = 1$ the generative model deflates gracefully to the single object case given in [14]. In the model, the locations of voxels X^{cam} are treated as generated randomly from the shape $\mathbf{\Phi}^{\mathsf{cam}}$, and all voxel locations *in camera coordinates* have the same probability of being generated.

The objective of multi-object tracking is to find the optimal sequence of sets of poses $\{\mathbf{p}_0...\mathbf{p}_t\}$ given the set of object shapes $\Phi_1...\Phi_M$ and observed RGB-D images $\{\Omega_1...\Omega_t\}$:

$$\max_{\mathbf{p}_0...\mathbf{p}_t} P(\mathbf{p}_0 \dots \mathbf{p}_t | \Phi_1...\Phi_M, \Omega_0 \dots \Omega_t) \tag{2}$$

We do not assume a motion model, hence all poses $\mathbf{p}_0...\mathbf{p}_t$ in the sequence are independent and we can consider each time step independently dropping the index t. Using the Bayes rule, it follows that:

$$P(\mathbf{p}|\Phi_1...\Phi_M, \Omega) \sim P(\Omega|\Phi_1...\Phi_M, \mathbf{p})P(\mathbf{p}|\Phi_1...\Phi_M) \tag{3}$$

In the following subsections, we will refer to the first term $P(\Omega|\Phi_1...\Phi_M, \mathbf{p})$ as the data term or image likelihood and the second term $P(\mathbf{p}|\Phi_1...\Phi_M)$ as the physical constraint term.

3.3. Data term

According to our graphical model, the data term (or image likelihood) follows as:

$$P(\Omega|\Phi_1...\Phi_M, \mathbf{p}) = P(\Omega|\mathbf{\Phi}^{\mathsf{cam}})P(\mathbf{\Phi}^{\mathsf{cam}}|\Phi_1...\Phi_M, \mathbf{p}), \tag{4}$$

where we introduced the 'shape union' $\mathbf{\Phi}^{\mathsf{cam}}$. Assuming that the observations are pixel-wise independent, $P(\Omega|\mathbf{\Phi}^{\mathsf{cam}})$ can be decomposed into a product of per-pixel likelihoods:

$$P(\Omega|\mathbf{\Phi}^{\mathsf{cam}}) = \prod_i P(x_i^{\mathsf{img}}, y_i|\mathbf{\Phi}^{\mathsf{cam}}) \tag{5}$$

By marginalization over V the per-pixel likelihood becomes:

$$P(x^{\mathsf{img}}, y|\mathbf{\Phi}^{\mathsf{cam}}) \propto \sum_{k=\{on,out\}} \left\{P(x^{\mathsf{img}}|\mathbf{\Phi}^{\mathsf{cam}}, V{=}k)P(V{=}k|y)\right\} \tag{6}$$

The pixel location likelihoods for the foreground and background are distributed as:

$$P(x^{\mathsf{img}}|\mathbf{\Phi}^{\mathsf{cam}}, V{=}on) = \frac{\delta_\epsilon(\mathbf{\Phi}^{\mathsf{cam}}(X^{\mathsf{cam}}))}{\eta_f} \tag{7}$$

$$P(x^{\mathsf{img}}|\mathbf{\Phi}^{\mathsf{cam}}, V{=}out) = \frac{H_\epsilon(\mathbf{\Phi}^{\mathsf{cam}}(X^{\mathsf{cam}}))}{\eta_b} \tag{8}$$

$$\eta_f = \sum_{x_i^{\mathsf{img}} \in \Omega} \delta_\epsilon(\mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})) \tag{9}$$

$$\eta_b = \sum_{x_i^{\mathsf{img}} \in \Omega} H_\epsilon(\mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})) \tag{10}$$

where H_ϵ is the smoothed Heaviside step function and δ_ϵ is the smoothed unit impulse function defined as:

$$H_\epsilon(z) = \frac{1}{1 + e^{-z/2}} \qquad \delta_\epsilon(z) = \frac{2e^{-z/2}}{(1 + e^{-z/2})^2} \tag{11}$$

Substituting the per-pixel likelihoods for x^{img} into Eqn. 6, we obtain the image likelihood as:

$$P(\Omega|\mathbf{\Phi}^{\mathsf{cam}}) \propto \prod_{x_i^{\mathsf{img}} \in \Omega} \left\{P_f \delta_\epsilon\left(\mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})\right) + P_b H_\epsilon\left(\mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})\right)\right\} \tag{12}$$

where $P_f = P(y|V = on)$ and $P_b = P(y|V = out)$. Note that this likelihood function is very similar to the one used in [14], but our likelihood is no longer directly a function of the set of object poses $\mathbf{p}$. Instead, the 'shape union' $\mathbf{\Phi}^{\mathsf{cam}}$

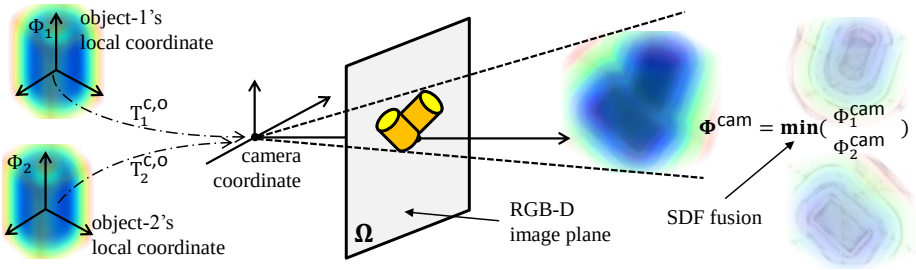

Figure 2. Illustration of the fusion of multiple object SDFs and the projection process. SDFs are first transformed into camera coordinates then fused together by a minimum function. The observed RGB-D image domain is generated by the fused SDF.

now is a function of the object shapes $\{\Phi_1 ... \Phi_M\}$ and poses $\mathbf{p}$. As mentioned earlier, if the number of objects $M = 1$ then Eqn. 12 reverts to the image likelihood in [14].

Because multiple SDFs can be fused together by taking the minimum value of all SDFs, we formulate $\mathbf{\Phi}^{\mathsf{cam}}$ as follows. Given a set of object shapes $\{\Phi_1 ... \Phi_M\}$ and their corresponding set of poses $\mathbf{p}$, we transform each object shape Φ_j into camera coordinates as Φ_j^{cam} using p_j. Then the object shapes in *camera coordinates* $\{\Phi_1^{\mathsf{cam}} ... \Phi_M^{\mathsf{cam}}\}$ are fused into a single SDF $\mathbf{\Phi}^{\mathsf{cam}}$ with a minimum function (see Eqn. 15 later). The geometry and notation are illustrated in Fig. 2.

The pose p_j of the j-th object parameterizes a transformation $\mathtt{T}_j^{\mathsf{o,c}}$ from object to camera coordinates. The j-th contribution to the shape union Φ_j^{cam} then follows as:

$$\Phi_j^{\mathsf{cam}}(X^{\mathsf{cam}}) = \Phi_j(\mathtt{T}^{\mathsf{o,c}} \dot{X}^{\mathsf{cam}}) = \Phi_j(X^{\mathsf{obj}}) \tag{13}$$

In the camera coordinate system, the contributions $\{\Phi_1^{\mathsf{cam}} ... \Phi_M^{\mathsf{cam}}\}$ are fused together using a 'soft' minimum function, so that we arrive at an analytical approximation of the shape union:

$$\begin{aligned} \mathbf{\Phi}^{\mathsf{cam}} &= \min\left(\Phi_1^{\mathsf{cam}}, \Phi_2^{\mathsf{cam}}, ..., \Phi_M^{\mathsf{cam}}\right) \\ &\approx -\frac{1}{\alpha} \log \sum_{j=1}^{M} e^{-\alpha \Phi_j^{\mathsf{cam}}} \end{aligned} \tag{14}$$

where α controls the smoothness of the approximation. Theoretically, larger α gives better approximation of the minimum function, but empirically, smaller α gives wider based of convergence for our tracker. Substituting Eqn. 13 into Eqn. 15:

$$\begin{aligned} \mathbf{\Phi}^{\mathsf{cam}}(X^{\mathsf{cam}}) &= -\frac{1}{\alpha} \log \sum_{j=1}^{M} e^{-\alpha \Phi_j(\mathtt{T}_j^{\mathsf{o,c}} \dot{X}^{\mathsf{cam}})} \\ &= -\frac{1}{\alpha} \log \sum_{j=1}^{M} e^{-\alpha \Phi_j(X_j^{\mathsf{obj}})} \end{aligned} \tag{15}$$

where X_j^{obj} is the back-projection of X^{cam} in the j-th object coordinate system. The probability of the shape union $P(\mathbf{\Phi}^{\mathsf{cam}} | \Phi_1 ... \Phi_M, \mathbf{p})$ in Eqn. 4 is just a Dirac distribution around this formulation, thus the image likelihood in Eqn. 4 becomes:

$$\begin{aligned} P(\Omega | \Phi_1 ... \Phi_M, \mathbf{p}) \propto \\ \prod_{x_i^{\mathsf{img}} \in \Omega} \Big\{ P_f \delta_\epsilon \left(-\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp\{-\alpha \Phi_m(X^{\mathsf{obj}} j')\} \right) \\ + P_b H_\epsilon \left(-\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp\{-\alpha \Phi_m(X^{\mathsf{obj}} j')\} \right) \Big\} \end{aligned} \tag{16}$$

The log-likelihood gives us the data-fitting term of the overall energy function: $E_{data} = -\log P(\Omega | \Phi_1 ... \Phi_M, \mathbf{p})$. We differentiate this energy term w.r.t. the set of pose parameters $\mathbf{p} = \{p_1, ..., p_M\}$ in which each p_j, recall, is a 6-vector:

$$\frac{\partial E_{data}}{\partial \mathbf{p}} = -\sum_{x_i^{\mathsf{img}} \in \Omega} \mathcal{L}_i \frac{\partial \mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})}{\partial \mathbf{p}}; \tag{17}$$

$$\mathcal{L}_i = \frac{P_f \delta_\epsilon^{'} + P_b H_\epsilon^{'}}{P_f \delta_\epsilon(\mathbf{\Phi}^{\mathsf{cam}}) + P_b H_\epsilon(\mathbf{\Phi}^{\mathsf{cam}})} \tag{18}$$

$$\frac{\partial \mathbf{\Phi}^{\mathsf{cam}}(X_i^{\mathsf{cam}})}{\partial \mathbf{p}} = -\frac{1}{\alpha} \sum_{j=1}^{M} w_j \nabla \Phi_j \frac{\partial X_{i,j}^{\mathsf{obj}}}{\partial \mathbf{p}}; \tag{19}$$

$$w_j = \frac{e^{-\alpha \Phi_j(X_{i,j}^{\mathsf{obj}})}}{\sum_{k=1}^{M} e^{-\alpha \Phi_k(X_{i,k}^{\mathsf{obj}})}} \tag{20}$$

where $\delta_\epsilon^{'}$ and $H_\epsilon^{'}$ are derivatives of δ_ϵ and H_ϵ respectively. $X_{i,j}^{\mathsf{obj}}$ is the back-projection of X_i^{cam} in the j-th object's coordinate frame with $\mathtt{T}_j^{\mathsf{o,c}}$. The gradients of the SDFs $\nabla \Phi_j = \left[\frac{\partial \Phi_j}{\partial x} \frac{\partial \Phi_j}{\partial y} \frac{\partial \Phi_j}{\partial z} \right]$ are computed using central finite differences, and, since the shape of the object is already known, these gradients for each object can be computed in advance.

Note that the derivative of this energy term has a very clear meaning: given a pixel x_i^{img} in the RGB-D image domain, instead of assigning this pixel deterministically to a certain object, we back-projected x_i^{img} (i.e. X_i^{cam} in camera coordinates, since $\dot{x}^{\mathsf{img}} = \mathtt{A} X^{\mathsf{cam}}$) into all objects' coordinates with the current set of poses $\mathbf{p}$. Then, a membership

weight w_j is automatically computed as in Eqn. 20. For example, consider a point in camera coordinates X_i^{cam}. If the back-projection $X_{i,m}^{\mathsf{obj}}$ is close to the m-th object's surface ($\Phi(X_{i,m}^{\mathsf{obj}}) \to 0$) and other back-projections $X_{i,j}^{\mathsf{obj}}$ are further away from the surfaces ($\Phi(X_{i,j}^{\mathsf{obj}}) > 0$), then we will find $w_m \to 1$ and the other $w_j \to 0$, with $\sum_{j=1}^{M} w_j = 1$. Thus w_j can be interpreted as the probability that a pixel X_i^{cam} belongs to the j-th object.

3.4. Physical constraint term

Now we come to the second likelihood term in Eqn. 3. We decompose the joint probability of all object poses given all 3D object shapes into a product of per-pose probabilities:

$$P(\mathbf{p}|\Phi_1...\Phi_M) \tag{21}$$
$$= P(p_1|\Phi_1...\Phi_M) \prod_{j=2}^{M} P(p_j|\{p_1...p_{j-1}\}, \Phi_1...\Phi_M)$$

This formulation can be used to enforce any physical priors on any poses, but in our case we only use it to avoid physical collisions between objects. Since we do not have any pose priors on any objects, we can drop the probability term $P(p_1|\Phi_1...\Phi_M)$.

The probability $P(p_j|\{p_1...p_{j-1}\}, \Phi_1...\Phi_M)$ is defined based on the fact that a surface point on one object should not move inside any other objects. For each object Φ_j, we uniformly and sparsely sample a set of collision points $\mathbf{C}_j = \{C_{j,1}, ..., C_{j,K}\}$ from its surface in object coordinates. We then project these collision points with its current pose $\mathtt{T}_j^{\mathsf{c,o}}(p_j)$ into the camera coordinates as $\mathbf{C}_j^{\mathsf{cam}} = \{C_{j,1}^{\mathsf{cam}}...C_{j,K}^{\mathsf{cam}}\} = \mathtt{T}_j^{\mathsf{c,o}}\mathbf{C}_j$. We use $\mathbf{\Phi}_{-j}^{\mathsf{cam}}$ to demote the union of SDFs $\{\Phi_1^{\mathsf{cam}}...\Phi_{j-1}^{\mathsf{cam}}\}$ in the camera coordinates. And $P(p_j|\{p_1...p_{j-1}\}, \Phi_1...\Phi_M)$ is defined as the average collision point agreement with $\mathbf{\Phi}_{-j}^{\mathsf{cam}}$

$$P(p_j|\{p_1...p_{j-1}\}, \Phi_1...\Phi_M) = \frac{1}{K}\sum_{k=1}^{K} H_\epsilon(\mathbf{\Phi}_{-j}^{\mathsf{cam}}(C_{j,k}^{\mathsf{cam}}) - \xi) \tag{22}$$

where H_ϵ is the smoothed Heaviside function from Eqn. 11. Since the partial shape union $\mathbf{\Phi}_{-j}^{\mathsf{cam}}$ also has negative values inside and positive values outside the surface, the per-collision point agreement $H_\epsilon(\mathbf{\Phi}_{-j}^{\mathsf{cam}}(C_{j,k}^{\mathsf{cam}}) - \xi)$ equals to 1 when $C_{j,k}^{\mathsf{cam}}$ is outside $\mathbf{\Phi}_{-j}^{\mathsf{cam}}$ and take on a decreasing value between 1 and 0 as $C_{j,k}^{\mathsf{cam}}$ moves deeper inside $\mathbf{\Phi}_{-j}^{\mathsf{cam}}$ (i.e. violating the collision constraint). ξ is a small positive offset to allow very close object surfaces. The log-likelihood of $P(\mathbf{p}|\Phi_1...\Phi_M)$ gives us the second energy term, the physical constraint term:

$$E_{phy} = -\sum_{j=1}^{M} \log\left(\frac{1}{K}\sum_{k=1}^{K} H_\epsilon(\mathbf{\Phi}_{j-}^{\mathsf{cam}}(C_{j,k}^{\mathsf{cam}}) - \xi)\right) \tag{23}$$

3.5. Optimization

The overall energy function is the sum of the data term and the physical constraint term $E = E_{data} + E_{phy}$. We use the local frame to evaluate the derivative at identity at each iteration, and use Levenberg-Marquardt (LM) to compute the incremental change in all poses jointly. The pose update are as follows:

$$\tilde{\mathbf{p}} = -(J_E^\top J_E + \lambda \mathbf{diag}(J_E^\top J_E))\frac{\partial E}{\partial \mathbf{p}} \tag{24}$$

$$\text{for each } \mathtt{T}_j^{\mathsf{o,c}}(p_j), \quad \mathtt{T}_j^{\mathsf{o,c}} \leftarrow \mathtt{T}(\tilde{p_j}) \begin{bmatrix} \mathtt{T}_j^{\mathsf{o,c}} & \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{25}$$

where λ is the non-negative LM damping factor that is adjusted at each iteration.

3.6. Online learning of appearance model

The surface and background appearance models $P(y|V)$ are very important for the robustness of the tracking, so we adapt the appearance model online in each frame after the tracking is completed. We use the pixels that have $|\mathbf{\Phi}^{\mathsf{cam}}(X^{\mathsf{cam}})| \leqslant 3$ (points that best fit the surfaces of multiple objects) to compute the surface appearance model and the pixels in the immediate surrounding region of the objects to compute the background model. The online update of the appearance models is achieved by using a linear opinion pool with learning rates $\{\alpha_{on,out}\}$:

$$P_t(y|V_i) = (1-\alpha_i)P_{t-1}(y|V_i) + \alpha_i P_t(y|V_i) \tag{26}$$

In all our experiments, we set $\alpha_{on} = 0.05$ and $\alpha_{out} = 0.3$.

4. Implementation and Experiments

We have performed a variety of experimental evaluations, both qualitative and quantitative. Qualitative examples of our algorithm tracking different types of objects in real-time and under significant occlusion and missing data are provided in supplementary material.

Implementation. We implemented both CPU and GPU versions of our multi-object tracker. With two objects, both implementations achieve real-time performance. The CPU version runs at some 30Hz on an Intel Core i7 3.4GHz processor, and the accelerated version at 35Hz using the same CPU and Nvidia GTX 680 GPU. The speed of the two implementations is similar in our experiments because (i) we only use pixels that are close to the projection of the object in the depth image and (ii) most of the tracked objects occupy a small region of the RGB-D image. This means we only leverage a few thousands points, which does not take full advantage of the computational power of the GPU. When a larger number of points is used (i.e when tracking objects that occupy larger image regions or when using denser depth), the GPU implementation will outperform the CPU implementation by a larger margin.

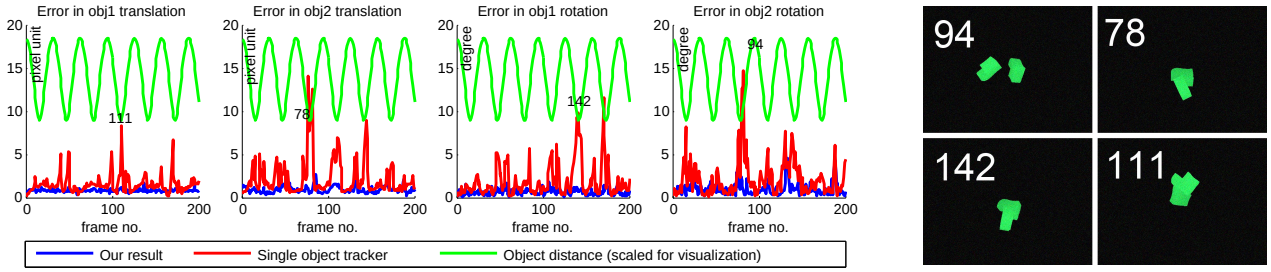

Figure 3. Pose estimation error comparison between our multi-object tracker and two instances of the single-object tracker [14]. Sample synthetic RGB-D frames used in our experiment are shown on the right with the frame number corresponding to the marks on the charts.

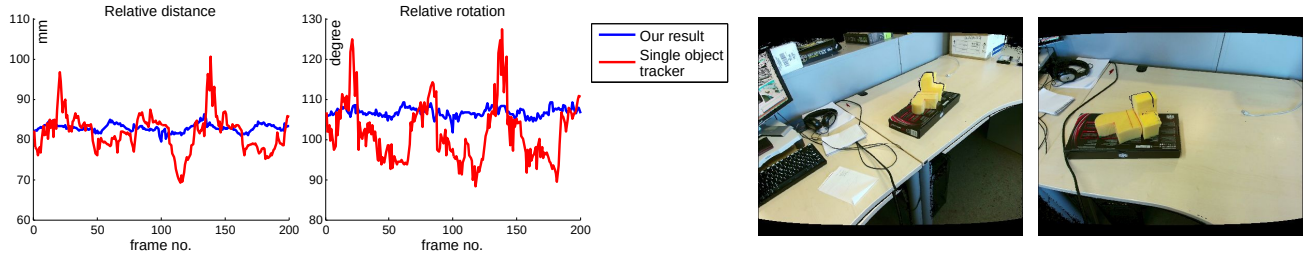

Figure 4. Comparison of the variance in relative pose estimation between our multi-object tracker and two instances of the single-object tracker in [14]. Sample RGB-D frames used in our experiment are shown on the right.

Quantitative Experiments. We begin with two sets of quantitative experiments to evaluate the proposed method. For the first (Fig. 3) we follow a standard benchmarking strategy from the markerless tracking literature and evaluate our tracking results on synthetic data, since ground truth information for real data is very difficult to obtain. We move two objects of known shape in front of a virtual camera and generate RGB-D frames. The objects periodically move further apart and closer from each other. We add Gaussian noise to both the rendered colour and the depth images. Four sample frames from the test sequence are shown in Fig. 3. Using this sequence we compare the tracking accuracy of our multi-object tracker with two instances of the single object tracker presented in [14]. To evaluate translation accuracy we use the Euclidean distance between the estimated and ground truth poses. To measure rotation accuracy, we rotate the unit vectors to the three axis directions $\mathbf{e}_x$,$\mathbf{e}_y$,$\mathbf{e}_z$ using the ground truth $\mathtt{R}_g$ and we estimate the rotation matrix $\mathtt{R}_e$. The error value is averaged over the three including angles of the resulting vectors: $r_{err} = \frac{1}{3}\sum_{i\in x,y,z} \cos^{-1}\left((\mathtt{R}_e\mathbf{e}_i)^\top \mathtt{R}_g\mathbf{e}_i)\right)$. In the graphical results of Fig. 3 the green line shows the relative distance between the two objects. Note that this value has been scaled and offset for visualization. It can be seen that when the two objects with similar appearance model are neither overlapping nor close (e.g. frame 94), both two single object trackers and our multi-object tracker provide accurate results. However, once the two objects move close together, the two separate single object trackers [14] produce very large errors. The single object tracker fails to model the pixel membership, leading to an incorrect pixel association when the two objects are close together. Our soft pixel membership solves this problem.

The second quantitative experiment (Fig. 4) makes a similar comparison, but with real imagery. As before, it is difficult to obtain the *absolute* ground truth pose of the objects, and instead we measure the consistency of the relative pose between two static objects by moving the camera around while looking towards the two objects. Example frames are shown in Fig. 4. If the two recovered poses are accurate we would expect consistent relative translation and rotation through the whole sequence. As shown in Fig. 4, our multi-object tracker is able to recover much more consistent relative translation and rotation than two independent instances of [14].

Qualitative Experiments. We use four challenging real sequences to demonstrate the robust performance of our multi-object tracker. The results are shown in Figs. 5,6 and 7. In Fig. 5, we show our multi-object algorithm tracking two objects: two pieces of cut foam (top) and a mug and a ball (bottom). The leftmost column shows snapshots of typical RGB and D images and the per-pixel foreground probability P_f. The right columns show the per-pixel membership weight w_j (top rows in each set), magenta and cyan corresponding to the two objects and blue pixels showing ambiguous membership. The tracking result is shown in the bottom rows. In the second sequence, the cup provides both occlusion and physical constraints to the ball. For this experiment, note that even though there is no depth observation from the ball (owing to significant occlusion from the

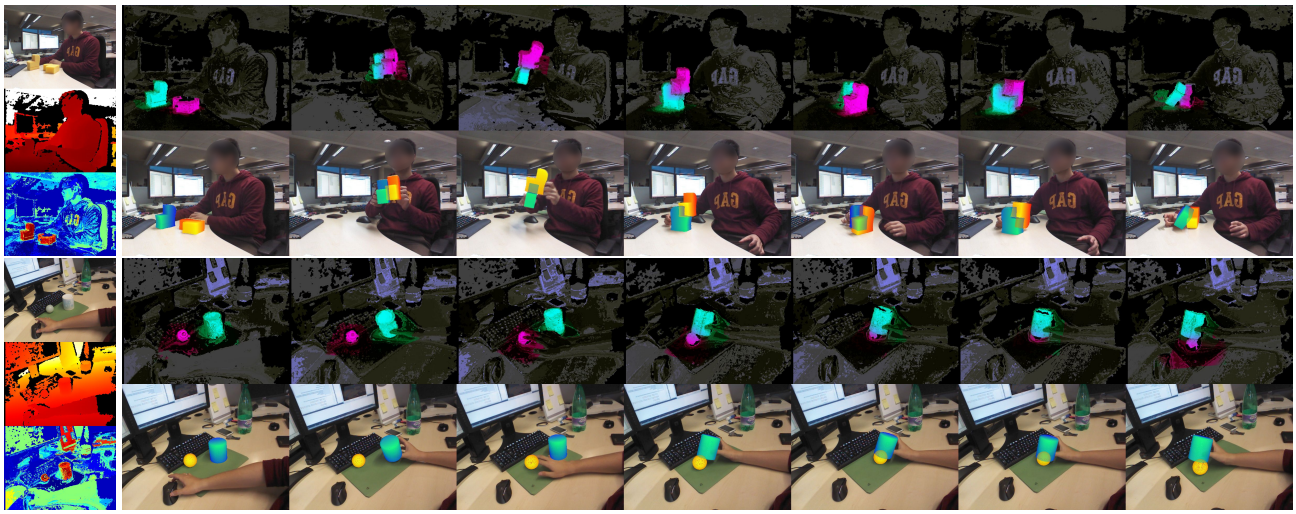

Figure 5. Film strips showing our multi-object algorithm tracking two objects: two pieces of cut foam (top) and a mug and a ball (bottom). The leftmost column shows snapshots of typical RGB and D images and the per-pixel foreground probability P_f. The right columns show the per-pixel membership weight w_j (top rows in each set) and the tracking result (bottom row in each set).

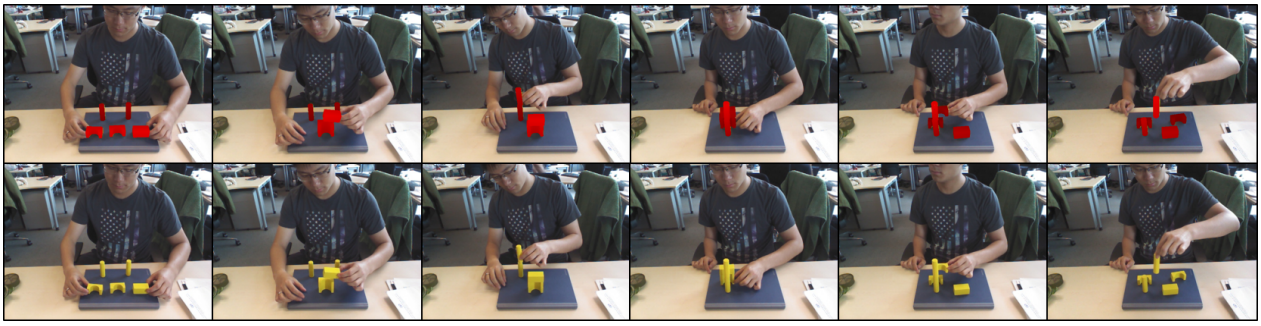

Figure 6. Film strips showing a challenging sequence where 5 pieces of toy bricks with identical colour are tracked. The top sequence shows the tracking result rendered on the colour image and the sequence below shows the original colour images.

cup), our algorithm can still accurately estimate the location of the ball using solely the physical constraint. In Fig. 6, we show a more challenging sequence where five pieces of toy bricks are tracked. The top sequence shows the tracking result and the bottom sequence shows the original colour input. The set of toy bricks has difference shapes but identical appearance. In spite of the heavy self-occlusion and the occlusion introduced by hands, our method can still track robustly and accurately.

When tracking in real-world scenarios, it is often difficult to obtain accurate 3D object models. Fig. 7 shows our tracker using *inaccurate* 3D shapes and still producing reasonable results. We track two interacting feet with a pair of very coarse shoe models. Throughout most of the sequence our tracker successfully recovers the two poses. However, we do also encounter two failure cases here. The first one is visible in column 4 of Fig. 7, where the shoe is incorrectly rotated. This happens because the 3D model is rotationally ambiguous around its long axis. The second failure case can be seen in column 6. Here, the ground pixels (i.e. the black shadow) have very high foreground probability, as can be clearly seen in row 3. With most of one foot occluded, the tracker incorrectly tries to fit the model to the pixels with high foreground probability, which leads to failure. As a testament to its robustness, our tracker does automatically recover from both failure cases.

5. Conclusions

We have presented a novel framework for tracking multiple 3D objects from a sequence of RGBD images. We show that our formulation has several advantages over instantiating multiple individual trackers, both from a theoretical point of view and from a practical one. First, our method is particularly well suited for tracking several objects with similar or identical appearance, which is a common case in many applications, such as tracking cars or pairs of hands or feet. Our method is grounded in a rigorous probabilistic framework, so we naturally obtain weights that tell us the probability of individual image observations being generat-

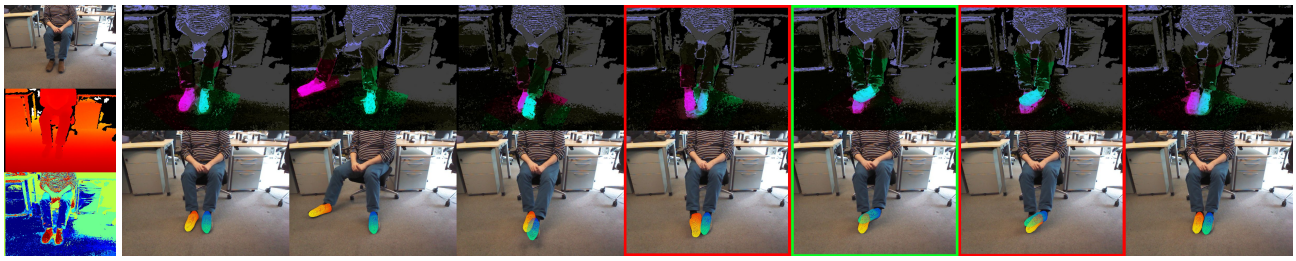

Figure 7. Film strips showing our algorithm tracking two interacting feet with two *inaccurate* models. The leftmost column shows snapshots of typical RGB and D images and the per-pixel foreground probability P_f. The right columns show the membership weight w_j (top row) and the tracking result (bottom row). The tracker failed on the frames outlined with red frames, but automatically recovers from failure.

ed by each of the tracked objects, thus implicitly solving the data association problem. Furthermore, our formulation naturally leads to what we call a physical constraint term, which allows us to specify prior knowledge about the world. We have used this term to indicate that it is unlikely that several objects occupy the same locations in 3D space. In addition to collision avoidance, our formulation allows for generic interaction forces between objects to be modelled.

We validate our claims with several experiments, showing that a combined tracking of multiple objects exhibits superior performance over instatiating the same tracker multiple times independently. For this evaluation we used an efficient implementation, that easily tracks multiple objects at 30 Hz without the use of any GPU acceleration. Our system is therefore well suited for real time applications, which is often an important criterion for tracking tasks.

Our tracker is region-based and currently uses simple histograms as appearance models, making in particularly well suited for untextured objects. Sometimes however other types of appearance models might be better suited, so a possible direction of research is to use different appearance models, such as texture-based models. In line with other model based 3D trackers our approach currently also requires 3D models of the tracked objects to be known and given to the algorithm. While we do explicitly show good performance even with crude and inaccurate models, this might be considered another shortcoming to be resolved in future work. In particular dynamic objects, such as hands, could be an interesting direction, since tracking individual fingers might greatly benefit from a tracker that can deal with near-identical appearance and nicely integrated physical constraints.

Acknowledgments. This work is funded by the 'REWIRE' project (Grant No. 287713) under the EU 7th Framework Programme, EPSRC grants EP/H050795 and EP/J014990, and the Australian Research Council (Laureate Fellowship FL130100102 to IDR).

References

[1] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans PAMI*, 14(2):239–256, 1992.

[2] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans PAMI*, 24(7):932–946, 2002.

[3] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *Proc. 12th BMVC*, 2001.

[4] D. B. Gennery. Visual tracking of known three-dimensional objects. *IJCV*, 7(3):243–270, 1992.

[5] C. Harris and C. Stennett. RAPiD – a video rate object tracker. In *Proc. 1st BMVC*, 1990.

[6] R. Held, A. Gupta, B. Curless, and M. Agrawala. 3D puppetry: a Kinect-based interface for 3D animation. In *Proc. ACM UIST*, 2012.

[7] N. Kyriazis and A. Argyros. Physically plausible 3D scene tracking: The single actor hypothesis. In *Proc. 26th CVPR*, 2013.

[8] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *IJCV*, 8(2):113–122, 1992.

[9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. 10th ISMAR*, 2011.

[10] I. Oikonomidis. Tracking the articulated motion of two strongly interacting hands. In *Proc. 25th CVPR*, 2012.

[11] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proc. 22th BMVC*, 2011.

[12] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proc. 13th ICCV*, 2011.

[13] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. In *Proc. 19th BMVC*, 2009.

[14] C. Y. Ren, V. Prisacariu, D. Murray, and I. Reid. STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data. In *Proc. 14th ICCV*, 2013.

[15] C. Y. Ren and I. D. Reid. A unified energy minimization framework for model fitting in depth. In *Proc. 12th ECCV Workshops*, 2012.

[16] R. Smith. Open Dynamics Engine, 2006. `http://www.ode.org/`.

[17] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. CopyMe3D: Scanning and printing persons in 3D. In *Proc. GCPR*, 2013.

[18] R. Ueda. Tracking 3D objects with Point Cloud Library, 2012. `pointclouds.org`.