Real-Time Tracking of Single and Multiple Objects from Depth-Colour Imagery Using 3D Signed Distance Functions

C. Y. Ren[1] · V. A. Prisacariu[1] · O. Kähler[1] · I. D. Reid[2] · D. W. Murray[1]

Received: 22 May 2015 / Accepted: 29 November 2016 / Published online: 11 January 2017

Abstract We describe a novel probabilistic framework for real-time tracking of multiple objects from combined depth-colour imagery. Object shape is represented implicitly using 3D signed distance functions. Probabilistic generative models based on these functions are developed to account for the observed RGB-D imagery, and tracking is posed as a maximum a posteriori problem. We present first a method suited to tracking a single rigid 3D object, and then generalise this to multiple objects by combining distance functions into a shape union in the frame of the camera. This second model accounts for similarity and proximity between objects, and leads to robust real-time tracking without recourse to bolt-on or ad-hoc collision detection.

Keywords Multi-object tracking · Depth tracking · RGB-D imagery · Signed distance functions · Real-time

Communicated by Lourdes Agapito, Hiroshi Kawasaki, Katsushi Ikeuchi, Martial Hebert.

✉ C. Y. Ren
carl@robots.ox.ac.uk

V. A. Prisacariu
victor@robots.ox.ac.uk

O. Kähler
olaf@robots.ox.ac.uk

I. D. Reid
ian.reid@adelaide.edu.au

D. W. Murray
dwm@robots.ox.ac.uk

[1] Department of Engineering Science, University of Oxford, Oxford, UK

[2] School of Computer Science, University of Adelaide, Adelaide, Australia

1 Introduction

Tracking object pose in 3D is a core task in computer vision, and has been a focus of research for many years. For much of that time, model-based methods were concerned with rigid objects having simple geometrical descriptions in 3D and projecting to a set of sparse and equally simple features in 2D. The last few years have seen fundamental changes in every aspect, from the use of learnt, geometrically complex, and sometimes non-rigid objects, to the use of dense and rich representations computed from conventional image and depth cameras.

In this paper we focus on very fast tracking of multiple rigid objects, without placing arbitary constraints upon their geometry or appearance. We first present a revision of our earlier 3D object tracking method using RGB-D imagery (Ren et al. 2013). Like many current 3D trackers, this was developed for single object tracking only. An extension to multiple objects could be formulated by replicating multiple independent object trackers, but such a naïve approach would ignore two common pitfalls. The first is similarity in appearance: multiple objects frequently have similar colour and shape (hands come in pairs; cars are usually followed by more cars, not by elephants; and so on). The second is the hard physical constraint that multiple rigid bodies may touch but may not occupy the same 3D space. These two issues are addressed here in an RGB-D tracker that we originally proposed in Ren et al. (2014). This tracker can recover the 3D pose of multiple objects with *identical* appearance, while preventing them from intersecting. The present paper summarizes our previous work and places the single and multiple object trackers in a common framework. We also extend the discussion of related work, and present additional experimental evaluations.

The paper is structured as follows. Section 2 gives an overview of related work. Sections 3 and 4 detail the probabilistic formulation of the single object tracker and the extensions to the multiple object tracking problem. Section 5 discusses the implementation and performance of our method and Sect. 6 provides experimental insight into its operation. Conclusions are drawn in Sect. 7.

2 Related Work

We begin our discussion by covering the general theme of model-based 3D tracking, then consider more specialised works that use distance transforms, and detail methods that aim to impose physical constraints for multi object tracking.

Most existing research on 3D tracking, with or without depth data, uses a model-based approach, estimating pose by minimising an objective function which captures the discrepancy between the expected and observed image cues. While limited computing power forced early authors (e.g. Harris and Stennett 1990; Gennery 1992; Lowe 1992) to exploit highly sparse data such as points and edges, the use of dense data is now routine.

An algorithm commonly deployed to align dense data is Iterative Closest Point (Besl and McKay 1992). ICP is used by Held et al. (2012) who input RGB-D imagery from a Kinect sensor to track hand-held rigid 3D puppets. They achieve robust and real-time performance, though occlusion introduced by the hand has to be carefully managed through a colour-based pre-segmentation phase. Rather awkwardly, a different appearance model is required to achieve this pre-segmentation when tracking multiple objects. A more general work is KinectFusion (Newcombe et al. 2011), where the entire scene structure along with camera poses 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 that the scene moves rigidly with respect to the camera, a condition which is obviously violated when generalising tracking to multiple independently moving objects.

Kim et al. (2010) perform simultaneous camera and multi-object pose estimation in real-time using only colour imagery as input. First, all objects are placed statically in the scene, and a 3D point cloud recovered and camera pose initialized by triangulating matched SIFT features (Lowe 2004) in a monocular keyframe reconstruction (Klein and Murray 2007). Second, the user delineates each object by drawing a 3D box on a keyframe, and the object model is associated with the set of 3D points lying close to the surfaces of the 3D boxes. Then, at each frame, the features are used for object re-detection, and a pose estimator best fits the detected object's model to the SIFT features. The bottom-up nature of the work rather limits overall robustness and extensibility. With the planar model representation used, only cuboid-shaped objects can be tracked.

A number of related tracking methods—and ones which appear much more readily generalisable to multiple objects—use sampling to optimise pose. In each the objective function involves rendering the model at some hypothesised pose into the observation domain and evaluating the differences between the generated and the observed visual cues; but in each the cost is deemed too non-convex, or its partial derivatives too expensive or awkward to compute, for gradient-based methods to succeed. Particle Swarm Optimization was used by Oikonomidis et al. (2011a) to track an articulated hand, and by Kyriazis and Argyros (2013) to follow the interaction between a hand and an object. Both achieve real-time performance by exploiting the power of GPUs, but the level of accuracy that can be achieved by PSO is not thoroughly understood either empirically or theoretically. Particle filtering has also been used, and with a variety visual features. Recalling much earlier methods, Azad et al. (2011) match 2D image edges with those rendered from the model, while Choi and Christensen (2010) add 2D landmark points to the edges. Turning to depth data, the objective function of Ueda (2012) compares the rendered and the observed depth map, while Wuthrich et al. (2013) also model the per-pixel occlusion and win more robust tracking in presence of occlusion. Adding RGB to depth, Choi and Christensen (2013) fold in photometric, 3D edge and 3D surface normal measures into their likelihood function for each particle state. Real-time performance is achieved using GPUs, but nonetheless careful limits have to be placed on the number of particles deployed.

An alternative to ICP is the use of the signed distance function (SDF). It was first shown by Fitzgibbon (2001) that distance transforms could be used to register 2D/3D point sets efficiently. Prisacariu and Reid (2012) project a 3D model into the image domain to generate an SDF-like embedding function, and the 3D pose of a rigid object is

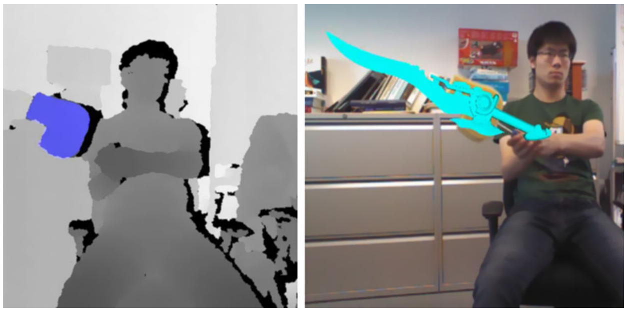


Fig. 1 Illustration of our method tracking an arbitrary object and enabling its use as a game controller. On the *left* we show the depth image overlaid with the tracking result and on the *right* we visualise a virtual sword with the corresponding 3D pose overlaid on the RGB image

recovered by evolving this embedding function. A faster approach has been linked with a 3D reconstruction stage, both without depth data by Prisacariu et al. (2012, 2013) and with depth by Ren et al. (2013). The SDF was used by Ren and Reid (2012) to formulate different embedding functions for robust real-time 3D tracking of rigid objects using only depth data, an approach extended by Ren et al. (2013) to leverage RGB data in addition. A similar idea is described by Sturm et al. (2013), who use the gradient of the SDF directly to track camera pose. KinectFusion (Newcombe et al. 2011) and most of its variants use a truncated SDF for shape representation, but, as noted earlier, KinectFusion uses ICP for camera tracking rather than directly exploiting the SDF. As shown by Sturm et al. (2013), ICP is less effective for this task.

Physical constraints in 3D object tracking 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 by Drummond and Cipolla (2002). However, when tracking multiple independently moving objects, physical constraints are introduced suddenly and intermittently by the collision of objects, and cannot be conveniently enforced by dof reduction. Indeed, rather few works explicitly model the physical collision between objects. Oikonomidis (2012) tracks two interacting hands with Kinect input, introducing a penalty term measuring the inter-penetration of fingers to invalidate impossible articulated poses. Both Oikonomidis et al. (2011b) and Kyriazis and Argyros (2013) track a hand and moving object simultaneously, and invalid configurations similarly penalized. In both cases the measure used is the minimum magnitude of 3D translation required to eliminate intersection of the two objects, a measure computed using the Open Dynamic Engine library (Smith 2006). In contrast, in the method presented here the collision constraint is more naturally enforced through a probabilistic generative model, without the need of an additional physics simulation engine (Fig.1).

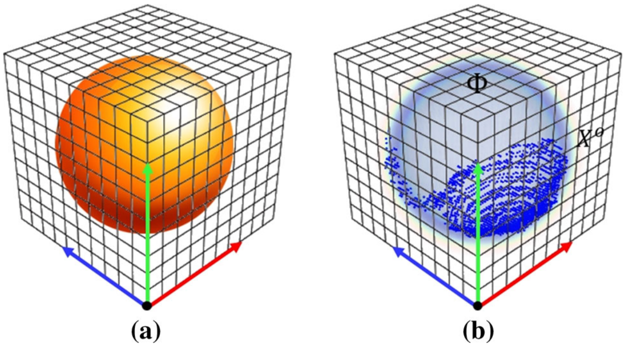


Fig. 3 **a** An object defined in a voxelised space. **b** Its signed distance embedding function is also defined in object coordinates with the same voxelisation

3 Single Object Tracking

Sections 3.2 and 3.3 introduce the graphical model and develop the maximum a posterior estimation underpinning our 3D tracker; and in Sect. 3.4 we discuss the online learning of the appearance model. First though we describe the basic geometry of the scene and image, sketched in Fig. 2, and establish notation.

3.1 Scene and Image Geometry

Using calibrated values of the intrinsic parameters of the depth and colour cameras, and of the extrinsics between them, the colour image is reprojected into the depth image. We denote the aligned RGB-D image as

$$\Omega = \left\{ \{\mathbf{X}^{\mathsf{i}}_1, c_1\}, \{\mathbf{X}^{\mathsf{i}}_2, c_2\} \ldots \{\mathbf{X}^{\mathsf{i}}_{N_\Omega}, c_{N_\Omega}\} \right\} , \tag{1}$$

where $\mathbf{X}^{\mathsf{i}} = Z\,\mathbf{x} = [Zu, Zv, Z]^\top$ is the homogeneous coordinate of a pixel with depth Z located at image coordinates $[u, v]$, and c is its RGB value. (The superscripts i, c and o will distinguish image, camera and object frame coordinates).

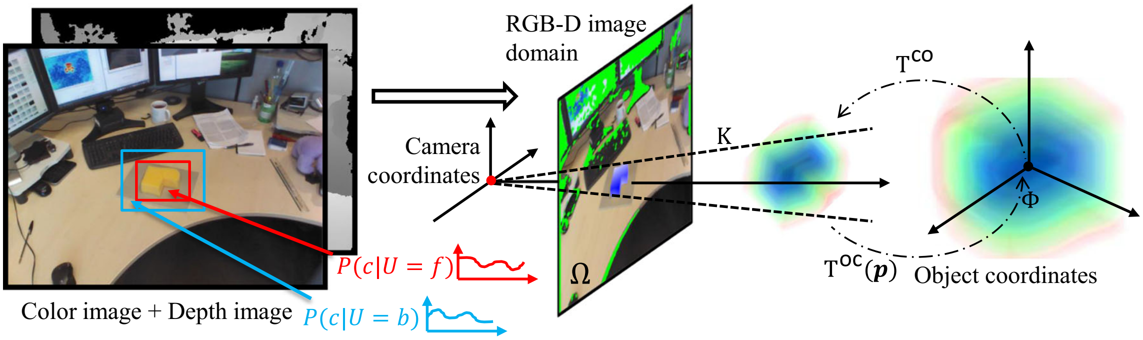

RGB-D image domain
Camera coordinates
T^{co}
K
Φ
Ω
$\mathrm{T}^{\mathsf{oc}}(\boldsymbol{p})$
Object coordinates
$P(c|U = f)$
$P(c|U = b)$
Color image + Depth image

Fig. 2 Representation of the 3D model Φ, the RGB-D image domain Ω, the foreground/background models $P(c|U{=}f)$, $P(c|U = b)$ and the pose $\mathrm{T}^{\mathsf{co}}(\mathbf{p})$

As illustrated in Fig. 3, we represent an object model by a 3D signed distance function (SDF), Φ, in object space. The space is discretised into voxels on a local grid surrounding the object. Voxel locations with negative signed distance map to the inside of the object and positive values to the outside. The surface of the 3D shape is defined by the zero-crossing $\Phi = 0$ of the SDF.

A point $\mathbf{X}^{\mathrm{o}} = [X^{\mathrm{o}}, Y^{\mathrm{o}}, Z^{\mathrm{o}}, 1]^{\top}$ on an object with pose $\mathbf{p}$, composed of a rotation and translation $\{\mathtt{R}, \mathbf{t}\}$, is transformed into the camera frame as $\mathbf{X}^{\mathrm{c}} = \mathtt{T}^{\mathrm{co}}(\mathbf{p})\mathbf{X}^{\mathrm{o}}$ by the 4×4 Euclidean transformation $\mathtt{T}^{\mathrm{co}}(\mathbf{p})$, and projected into the image under perspective as $\mathbf{X}^{\mathrm{i}} = \mathtt{K}[\mathtt{I}_{3\times3}|\mathbf{0}]\mathbf{X}^{\mathrm{c}}$, where $\mathtt{K}$ is the depth camera's matrix of intrinsic parameters.

We introduce a co-representation $\left\{\mathbf{X}^{\mathrm{i}}, c, U\right\}$ for each pixel, where the label $U \in \{f, b\}$ is set depending on whether the pixel is deemed to originate from the foreground object or from the background. Two appearance models describe the colour statistics of the scene: that for the foreground is generated by the object surface, while that for the background is generated by voxels outside the object. The models are represented by the likelihoods $P(c|U = f)$ and $P(c|U = b)$ which are stored as normalised RGB histograms using 16 bins per colour channel. The histograms can be initialised either from a detection module or from a user-selected bounding box on the RGB 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.

3.2 Generative Model and Tracking

The generative model motivating our approach is depicted in Fig. 4. We assume that each pixel is independent, and sample the observed RGB-D image Ω as a bag-of-pixels $\{\mathbf{X}^{\mathrm{i}}_j, c_j\}_{1\ldots N_\Omega}$. Each pixel depends on the shape Φ and pose $\mathbf{p}$ the object, and on the per-pixel latent variable U_j. Strictly, it is the depth $Z(\mathbf{x}_j)$ and colour c_j that are randomly drawn for each pixel location $\mathbf{x}_j$, but we use $\mathbf{X}^{\mathrm{i}}_j$ as a convenient proxy for $Z(\mathbf{x}_j)$.

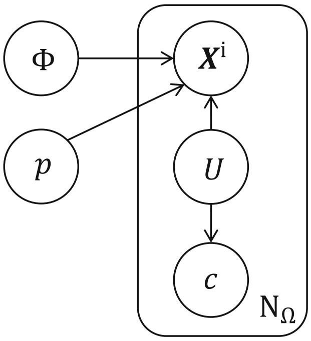


Fig. 4 The graphical model underpinning the single-object tracker

Omitting the index j, the joint distribution for a single pixel is

$$P(\mathbf{X}^{\mathrm{i}}, c, U, \Phi, \mathbf{p}) = P(\Phi)\, P(\mathbf{p})\, P(\mathbf{X}^{\mathrm{i}}|U, \Phi, \mathbf{p})\, P(c|U)\, P(U) \tag{2}$$

and marginalising over the label U gives

$$P(\mathbf{X}^{\mathrm{i}}, c, \Phi, \mathbf{p}) = P(\Phi)P(\mathbf{p}) \sum_{u\in\{f,b\}} P(\mathbf{X}^{\mathrm{i}}|U = u, \Phi, \mathbf{p})P(c|U = u)P(U = u). \tag{3}$$

Given the pose, $\mathbf{X}^{\mathrm{o}}$ can be found immediately as the back-projection of $\mathbf{X}^{\mathrm{i}}$ into object coordinates

$$\mathbf{X}^{\mathrm{o}} = \mathtt{T}^{\mathrm{oc}}(\mathbf{p})[(\mathtt{K}^{-1}\mathbf{X}^{\mathrm{i}})^{\top}\ 1]^{\top}\ , \tag{4}$$

so that $P(\mathbf{X}^{\mathrm{i}}|U = u, \Phi, \mathbf{p}) \equiv P(\mathbf{X}^{\mathrm{o}}|U = u, \Phi, \mathbf{p})$. This allows us to define the per-pixel likelihoods as functions of $\Phi(\mathbf{X}^{\mathrm{o}})$: we use a normalised smoothed delta function and a smoothed, shifted Heaviside function

$$P(\mathbf{X}^{\mathrm{i}}|U{=}f, \Phi, \mathbf{p}) = \delta^{\mathrm{on}}(\Phi(\mathbf{X}^{\mathrm{o}}))/\eta_f \tag{5}$$

$$P(\mathbf{X}^{\mathrm{i}}|U{=}b, \Phi, \mathbf{p}) = H^{\mathrm{out}}(\Phi(\mathbf{X}^{\mathrm{o}}))/\eta_b, \tag{6}$$

with $\eta_f = \sum_{j=1}^{N_\Phi} \delta^{\mathrm{on}}(\Phi(\mathbf{X}^{\mathrm{o}}_j))$, and $\eta_b = \sum_{j=1}^{N_\Phi} H^{\mathrm{out}}(\Phi(\mathbf{X}^{\mathrm{o}}_j))$. The functions themselves, plotted in Fig. 5, are

$$\delta^{\mathrm{on}}(\Phi) = \mathrm{sech}^2(\Phi/2\sigma) \tag{7}$$

$$H^{\mathrm{out}}(\Phi) = \begin{cases} 1 - \delta^{\mathrm{on}}(\Phi) & \text{if } \Phi \geq 0 \\ 0 & \text{if } \Phi < 0. \end{cases} \tag{8}$$

The constant parameter σ determines the width of the basin of attraction—a larger σ gives a wider basin of convergence to the energy function, while a smaller σ leads to faster convergence. In our experiments we use $\sigma = 2$.

The prior probabilities of observing foreground and background models $P(U = f)$ and $P(U = b)$ in Eq. (3) are assumed uniform:

$$P(U{=}f){=}\eta_f/\eta, \quad P(U{=}b){=}\eta_b/\eta, \quad \eta{=}\eta_f + \eta_b\ . \tag{9}$$

Substituting Eqs. (5)–(9) into Eq. (3), the joint distribution for an individual pixel becomes

$$P(\mathbf{X}^{\mathrm{i}}, c, \Phi, \mathbf{p}) = P(\Phi)P(\mathbf{p}) \left(P^f \delta^{\mathrm{on}}(\Phi(\mathbf{X}^{\mathrm{o}})) + P^b H^{\mathrm{out}}(\Phi(\mathbf{X}^{\mathrm{o}}))\right)\ , \tag{10}$$

where $P^f{=}P(c|U{=}f)$ and $P^b{=}P(c|U{=}b)$ are developed in Sect. 3.4 below.

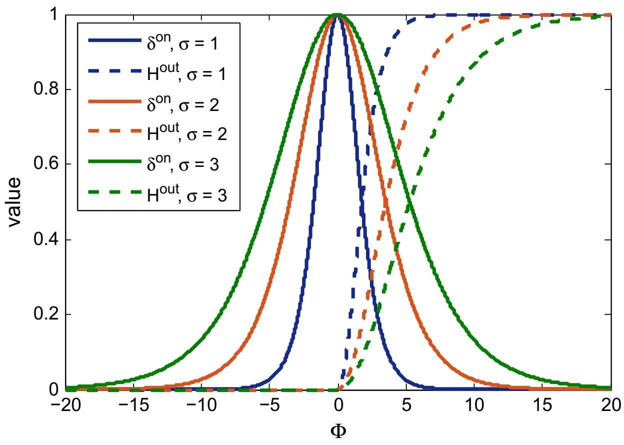


Fig. 5 The smoothed delta δ^{on} and Heaviside H^{out} functions

3.3 Pose Optimisation

Tracking involves determining the MAP estimate of the poses given their observed RGB-D images and the object shape Φ. We consider the pose at each time step t to be independent, and seek

$$\operatorname{argmax}_{\mathbf{p}_t} P(\mathbf{p}_t|\Phi, \Omega_t) = \operatorname{argmax}_{\mathbf{p}_t} \frac{P(\mathbf{p}_t, \Phi, \Omega_t)}{P(\Phi, \Omega_t)} . \qquad (11)$$

Were the pose optimisation guaranteed to find the "correct" pose no matter what the starting state, this notion of independence would be exact. In practice it is an approximation. Assuming that tracking is healthy, to increase the chance of *maintaining* a correct pose we start the current optimization at the pose delivered at the previous time step, and accept that if tracking is failing this introduces bias. We note that the starting pose is not a prior, and we do not maintain a motion model.

The denominator in Eq. (11) is independent of $\mathbf{p}$ and can be ignored. (We drop the index t to avoid clutter). Because the image Ω is sampled as a bag of pixels, we exploit pixel-wise independence and write the numerator as

$$P(\mathbf{p}, \Phi, \Omega) = \prod_{j=1}^{N_\Omega} P(\mathbf{X}^{\mathsf{i}}_j, c_j, \Phi, \mathbf{p}) . \qquad (12)$$

Substituting $P(\mathbf{X}^{\mathsf{i}}_j, c_j, \Phi, \mathbf{p})$ from Eq. (10), and noting that $P(\Phi)$ is independent of $\mathbf{p}$, and $P(\mathbf{p})$ will be uniform in the absence of prior information about likely poses,

$$P(\mathbf{p}|\Phi, \Omega) \sim \prod_{j=1}^{N_\Omega} \left\{ P^f_j \delta^{\mathsf{on}}(\Phi(\mathbf{X}^{\mathsf{o}}_j)) + P^b_j H^{\mathsf{out}}(\Phi(\mathbf{X}^{\mathsf{o}}_j)) \right\} . \qquad (13)$$

The negative logarithm of Eq. (13) provides the cost

$$\mathscr{E} = -\sum_{j=1}^{N_\Omega} \log \left\{ P^f_j \delta^{\mathsf{on}}(\Phi(\mathbf{X}^{\mathsf{o}}_j)) + P^b_j H^{\mathsf{out}}(\Phi(\mathbf{X}^{\mathsf{o}}_j)) \right\} \qquad (14)$$

to be minimised using Levenberg–Marquardt. In the minimisation, pose $\mathbf{p}$ is always set in a local coordinate frame, and the cost is therefore parametrised in the *change* in pose, $\mathbf{p}^*$. The derivatives required are

$$\frac{\partial \mathscr{E}}{\partial \mathbf{p}^*} = \sum_{j=1}^{N_\Omega} \left\{ \left[\frac{P^f_j \frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} + P^b_j \frac{\partial H^{\mathsf{out}}}{\partial \Phi}}{P(\mathbf{X}^{\mathsf{i}}_j, c_j|\Phi, \mathbf{p})} \frac{\partial \Phi}{\partial \mathbf{X}^{\mathsf{o}}_j} \right] \frac{\partial \mathbf{X}^{\mathsf{o}}_j}{\partial \mathbf{p}^*} \right\} \qquad (15)$$

where $\mathbf{X}^{\mathsf{o}}$ is treated as a 3-vector. The derivatives involving δ^{on} and H^{out} are

$$\frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} = -\frac{1}{\sigma} \tanh(\Phi/2\sigma) \operatorname{sech}^2(\Phi/2\sigma) \qquad (16)$$

and

$$\frac{\partial H^{\mathsf{out}}}{\partial \Phi} = \begin{cases} -\frac{\partial \delta^{\mathsf{on}}}{\partial \Phi} & \text{if } \Phi \geq 0 \\ 0 & \text{if } \Phi < 0. \end{cases} \qquad (17)$$

The derivatives $(\partial \Phi / \partial \mathbf{X}^{\mathsf{o}})$ of the SDF are computed using finite central differences. We use modified Rodrigues parameters for the pose $\mathbf{p}$ (c.f. Shuster (1993)). Using the local frame, the derivatives of $\mathbf{X}^{\mathsf{o}}$ with respect to the pose update $\mathbf{p}^* = \left[t^*_x, t^*_y, t^*_z, r^*_1, r^*_2, r^*_3 \right]^\top$ are always evaluated at identity so that

$$\frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial t^*_x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial t^*_y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial t^*_z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
$$\frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial r^*_1} = \begin{bmatrix} 0 \\ -4Z^{\mathsf{o}} \\ 4Y^{\mathsf{o}} \end{bmatrix} \quad \frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial r^*_2} = \begin{bmatrix} 4Z^{\mathsf{o}} \\ 0 \\ -4X^{\mathsf{o}} \end{bmatrix} \quad \frac{\partial \mathbf{X}^{\mathsf{o}}}{\partial r^*_3} = \begin{bmatrix} -4Y^{\mathsf{o}} \\ 4X^{\mathsf{o}} \\ 0 \end{bmatrix} . \qquad (18)$$

The pose change is found from the Levenberg–Marquardt update as

$$\mathbf{p}^* = \left\{ - \left[\mathtt{J}^\top \mathtt{J} + \lambda \operatorname{diag}\left[\mathtt{J}^\top \mathtt{J} \right] \right]^{-1} \frac{\partial \mathscr{E}}{\partial \mathbf{p}^*} \right\} , \qquad (19)$$

where $\mathtt{J}$ is the Jacobian matrix of the cost function, and λ is the non-negative damping factor adjusted at each iteration. Interpreting the solution vector $\mathbf{p}^*$ as an element in $\mathbb{SE}(3)$, and re-expressing as a 4×4 matrix, we apply the incremental transformation at iteration $n+1$ onto the estimated transformation at the previous iteration n as $\mathtt{T}^{n+1} \leftarrow \mathtt{T}(\mathbf{p}^*)\mathtt{T}^n$. The estimated object pose $\mathtt{T}^{\mathsf{oc}}$ results from composing the

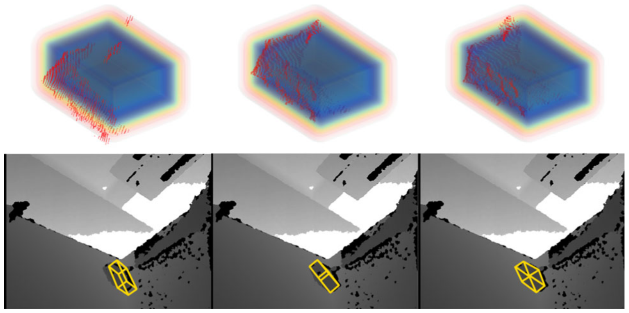


Fig. 6 Typical process of convergence for one frame. The *top row* shows the back-projected points and the SDF in the object coordinates. The *bottom row* visualises the object outline on depth image with corresponding poses

final incremental transformation $\mathtt{T}^N$ onto the previous pose as $\mathtt{T}^{\mathsf{oc}}_{t+1} \leftarrow \mathtt{T}^N \mathtt{T}^{\mathsf{oc}}_t$.

Figure 6 illustrates outputs from the tracking process during minimization. At each iteration the gradients of the cost function guide the back-projected points with $P^f > P^b$ towards the zero-level of the SDF and also force points with $P^f < P^b$ to move outside the object. At convergence, the points with $P^f > P^b$ will lie on the surface of the object.

The initial pose for the optimisation is specified manually or, in the case of live tracking, by placing the object in a prespecified position. An automatic technique, for example one based on regressing pose, could readily be incorporated to bootstrap the tracker.

3.4 Online Learning of the Appearance Model

The foreground/background appearance model $P(c|U)$ is important for the robustness of the tracking, and we adapt the appearance model online after tracking is completed on each frame. We use the pixels that have $|\Phi(\mathbf{X}^{\mathsf{o}})| \leq 3$ (that is, points that best fit the surface of the object) to compute the foreground appearance model and the pixels in the immediate surrounding region of the objects to compute the background model. The online update of the appearance model is achieved using a linear opinion pool

$$P_t(c|U=u) = (1-\rho^u)P_{t-1}(c|U) + \rho^u P_t(c|U) \qquad (20)$$

where ρ^u with $u \in \{f, b\}$ are the learning rates, set to $\rho^f = 0.05$ and $\rho^b = 0.3$. The background appearance model has a higher learning rate because we assume that the object is moving in an uncontrolled environment, where the change of appearance of the background is much faster than that of the foreground.

4 Generalisation for Multiple Object Tracking

One straightforward approach to tracking multiple objects would be to replicate several single object trackers. However, as argued in the introduction and as shown below, a more careful approach is warranted. In Sect. 4.2 we will find a probabilistic way of resolving ambiguities in case of identical appearance models. Then in Sect. 4.3 we show how physical constraints such as collision avoidance can be incorporated in the formulation. First though we extend our notation and graphical model.

4.1 Multi-Object Generative Model

The scene geometry and additional notation for simultaneous tracking of M objects is illustrated in Fig. 7(a), and the graphical generative model for the RGB-D image is shown in Fig. 7 (b). When tracking multiple objects in the scene, Ω is conditionally dependent on the set of 3D object shapes $\{\Phi_1 \ldots \Phi_M\}$ and their corresponding poses $\{\mathbf{p}_1 \ldots \mathbf{p}_M\}$.

Given the shapes and poses at any particular time, we transform the shapes into the camera frame and fuse them into a single 'shape union' $\boldsymbol{\Phi}^{\mathsf{c}}$. Then, for each pixel location, the depth is drawn from the foreground/background model U and the shape union $\boldsymbol{\Phi}^{\mathsf{c}}$, following the same structure as in Sect. 3. The colour is drawn from the appearance model $P(c|U)$, as before. We stress that although each object has a separate shape model in the set, two or more might be identical both in shape and appearance. This is the case later in the experiment of Fig. 14. We also note that when the number of objects drops

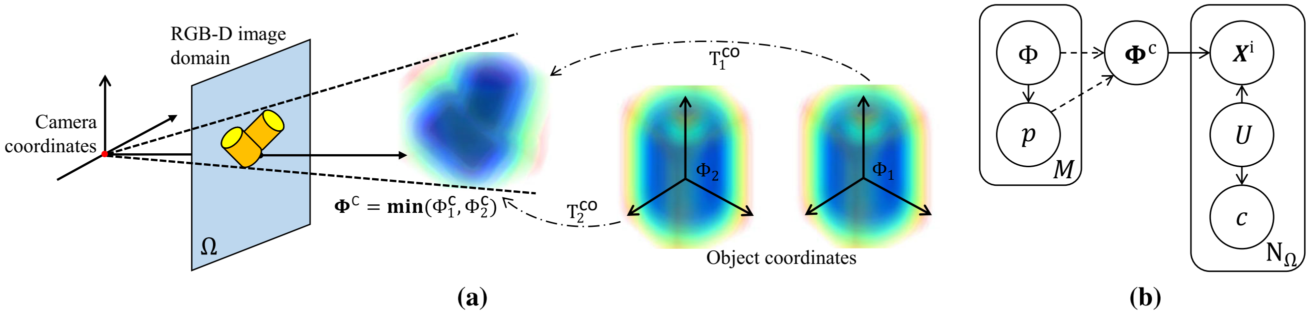

RGB-D image domain
Camera coordinates
Ω
$\boldsymbol{\Phi}^{\mathsf{C}} = \mathbf{min}(\Phi_1^{\mathsf{C}}, \Phi_2^{\mathsf{C}})$
$\mathrm{T}_1^{\mathsf{CO}}$
$\mathrm{T}_2^{\mathsf{CO}}$
Φ_2
Φ_1
Object coordinates
(a)
Φ
p
M
$\boldsymbol{\Phi}^{\mathsf{c}}$
X^{i}
U
c
N_Ω
(b)

Fig. 7 **a** Illustration of the fusion of multiple object SDFs in the shape union in the camera frame. SDFs are first transformed into camera coordinates then fused together by a minimum function. The observed RGB-D image domain is generated by projecting the fused SDF. **b** The extended graphical model to $M{=}1$ the generative model deflates gracefully to the single object case.

From the graphical model, the joint probability is

$$\begin{aligned} &P(\Phi_1 \dots \Phi_M, \mathbf{p}_1 \dots \mathbf{p}_M, \boldsymbol{\Phi}^{\mathsf{c}}, \mathbf{X}^{\mathsf{i}}, U, c) \\ &\quad = P(\Phi_1 \dots \Phi_M) P(\boldsymbol{\Phi}^{\mathsf{c}} | \Phi_1 \dots \Phi_M, \mathbf{p}_1 \dots \mathbf{p}_M) \\ &\qquad P(\mathbf{X}^{\mathsf{i}}, U, c | \boldsymbol{\Phi}^{\mathsf{c}}) P(\mathbf{p}_1 \dots \mathbf{p}_M | \Phi_1 \dots \Phi_M) \end{aligned} \tag{21}$$

where

$$P(\mathbf{X}^{\mathsf{i}}, U, c | \boldsymbol{\Phi}^{\mathsf{c}}) = P(\mathbf{X}^{\mathsf{i}} | U, \boldsymbol{\Phi}^{\mathsf{c}}) P(c|U) P(U)\,. \tag{22}$$

Because the shape union is completely determined given the sets of shapes and poses, $P(\boldsymbol{\Phi}^{\mathsf{c}} | \Phi_1 \dots \Phi_M, \mathbf{p}_1 \dots \mathbf{p}_M)$ is unity. As in the single object case, the posterior distribution of the set of poses given all object shapes can be obtained by marginalising over the latent variable U

$$\begin{aligned} &P(\mathbf{p}_1 \dots \mathbf{p}_M | \mathbf{X}^{\mathsf{i}}, c, \Phi_1 \dots \Phi_M) \sim \\ &\quad P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}}) P(\mathbf{p}_1 \dots \mathbf{p}_M | \Phi_1 \dots \Phi_M)\,, \end{aligned} \tag{23}$$

where

$$\begin{aligned} &P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}}) \\ &\quad = \sum_{u \in \{f,b\}} P(\mathbf{X}^{\mathsf{i}} | U = u, \boldsymbol{\Phi}^{\mathsf{c}}) P(c | U = u) P(U = u). \end{aligned} \tag{24}$$

The first term in Eq. (23), $P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}})$, describes how likely a pixel is to be generated by the current shape union, in terms of both the colour value and the 3D location, and is referred to as the data term. The second term, $P(\mathbf{p}_1 \dots \mathbf{p}_M | \Phi_1 \dots \Phi_M)$, puts a prior on the set of poses given the set of shapes and provides a physical constraint term.

4.2 The Data Term

Echoing Sect. 3, the per-pixel likelihoods $P(\mathbf{X}^{\mathsf{i}} | U = u, \boldsymbol{\Phi}^{\mathsf{c}})$ are defined by smoothed delta and Heaviside functions

$$P(\mathbf{X}^{\mathsf{i}} | U = f, \boldsymbol{\Phi}^{\mathsf{c}}) = \delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})) / \eta_f^{\mathsf{c}} \tag{25}$$

$$P(\mathbf{X}^{\mathsf{i}} | U = b, \boldsymbol{\Phi}^{\mathsf{c}}) = H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})) / \eta_b^{\mathsf{c}} \tag{26}$$

where $\eta_f^{\mathsf{c}} {=} \sum_{j=1}^{N_\Omega} \delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}_j^{\mathsf{c}}))$, $\eta_b^{\mathsf{c}} {=} \sum_{j=1}^{N_\Omega} H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}_j^{\mathsf{c}}))$, and where $\mathbf{X}^{\mathsf{c}}$ is the back-projection $\mathbf{X}^{\mathsf{i}}$ into the camera frame (note, not the object frame). The per-pixel labellings again follow uniform distributions

$$P(U{=}f){=}\frac{\eta_f^{\mathsf{c}}}{\eta^{\mathsf{c}}}, \quad P(U = \quad b){=}\frac{\eta_b^{\mathsf{c}}}{\eta^{\mathsf{c}}}, \quad \eta^{\mathsf{c}} = \eta_f^{\mathsf{c}} {+} \eta_b^{\mathsf{c}}. \tag{27}$$

Substituting Eqs. (25–27) into Eq. (24) we obtain the likelihood of the shape union for a single pixel

$$P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}}) = P^f \delta^{\mathsf{on}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})) + P^b H^{\mathsf{out}}(\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})), \tag{28}$$

where P^f and P^b are the appearance models of Sect. 3.

To form the shape union $\boldsymbol{\Phi}^{\mathsf{c}}$ we transform each object shape Φ_m into camera coordinates as Φ_m^{c} using $\mathtt{T}^{\mathsf{co}}(\mathbf{p}_m)$, and fuse them into a single SDF with the minimum function approximated by an analytical relaxation

$$\boldsymbol{\Phi}^{\mathsf{c}} = \min\left(\Phi_1^{\mathsf{c}}, \dots, \Phi_M^{\mathsf{c}}\right) \approx -\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp\{-\alpha \Phi_m^{\mathsf{c}}\} \tag{29}$$

in which α controls the smoothness of the approximation. Larger α gives a better approximation of the minimum function, but we find empirically that choosing a smaller α gives a wider basin of convergence for the tracker. We use $\alpha{=}2$ in this work. The per-voxel values of Φ_m^{c} are calculated using

$$\Phi_m^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}}) = \Phi_m(\mathbf{X}_m^{\mathsf{o}}) \tag{30}$$

where $\mathbf{X}_m^{\mathsf{o}} = \mathtt{T}^{\mathsf{oc}}(\mathbf{p}_m)\mathbf{X}^{\mathsf{c}}$ is the transformation of $\mathbf{X}^{\mathsf{c}}$ into the m-th object's frame. The likelihood for a pixel then becomes

$$\begin{aligned} &P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}}) \\ &\quad = P^f \delta^{\mathsf{on}} \left(-\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp\{-\alpha \Phi_m(\mathbf{X}_m^{\mathsf{o}})\} \right) \\ &\qquad + P^b H^{\mathsf{out}} \left(-\frac{1}{\alpha} \log \sum_{m=1}^{M} \exp\{-\alpha \Phi_m(\mathbf{X}_m^{\mathsf{o}})\} \right)\,. \end{aligned} \tag{31}$$

Assuming pixel-wise independence, the negative log likelihood across the RGB-D image provides a data term

$$\mathscr{E}_{\text{data}} = -\log P(\Omega | \boldsymbol{\Phi}^{\mathsf{c}}) = -\sum_{j=1}^{N_\Omega} \log P(\mathbf{X}_j^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}}) \tag{32}$$

in the overall energy function.

We will require the derivatives of this term w.r.t. the change of the set of pose parameters $\boldsymbol{\Theta}^*{=}\{\mathbf{p}_1^* \dots \mathbf{p}_M^*\}$. Dropping the pixel index j, we write

$$\frac{\partial \mathscr{E}_{\text{data}}}{\partial \boldsymbol{\Theta}^*} = -\sum_{\mathbf{X}^{\mathsf{i}} \in \Omega} \left\{ \frac{P^f \frac{\partial \delta^{\mathsf{on}}}{\partial \boldsymbol{\Phi}^{\mathsf{c}}} + P^b \frac{\partial H^{\mathsf{out}}}{\partial \boldsymbol{\Phi}^{\mathsf{c}}}}{P(\mathbf{X}^{\mathsf{i}}, c | \boldsymbol{\Phi}^{\mathsf{c}})} \frac{\partial \boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})}{\partial \boldsymbol{\Theta}^*} \right\} \tag{33}$$

where

$$\frac{\partial \boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})}{\partial \boldsymbol{\Theta}^*} = -\frac{1}{\alpha} \sum_{m=1}^{M} w_m \frac{\partial \Phi_m}{\partial \mathbf{X}_m^{\mathsf{o}}} \frac{\partial \mathbf{X}_m^{\mathsf{o}}}{\partial \boldsymbol{\Theta}^*}\,, \tag{34}$$

$$w_m = \frac{\exp\{-\alpha \Phi_m(\mathbf{X}_m^{\mathsf{o}})\}}{\sum_{k=1}^{M} \exp\{-\alpha \Phi_k(\mathbf{X}_k^{\mathsf{o}})\}}, \tag{35}$$

and

$$\frac{\partial \mathbf{X}^{\mathsf{o}}_m}{\partial \boldsymbol{\Theta}^*} = \left[\frac{\partial \mathbf{X}^{\mathsf{o}}_m}{\partial \mathbf{p}^*_1} \dots \frac{\partial \mathbf{X}^{\mathsf{o}}_m}{\partial \mathbf{p}^*_M} \right]. \tag{36}$$

The remaining pose and SDF derivatives ($\partial \mathbf{X}^{\mathsf{o}}_m / \partial \mathbf{p}^*_k$ and $\partial \Phi_m / \partial \mathbf{X}^{\mathsf{o}}_m$) are as in Sect. 3.

Note that instead of assigning a pixel $\mathbf{X}^{\mathsf{i}}$ in the RGB-D image domain deterministically to one object, we back-project $\mathbf{X}^{\mathsf{i}}$ (i.e. $\mathbf{X}^{\mathsf{c}}$ in camera coordinates) into all objects' frames with the current set of poses. The weights w_m are then computed according to Eq. (35), giving a smoothly varying pixel to object association weight. This can also be interpreted as the probability that a pixel is projected from the m-th object. If the back-projection $\mathbf{X}^{\mathsf{o}}_m$ of $\mathbf{X}^{\mathsf{c}}$ is close to the m-th object's surface ($\Phi(\mathbf{X}^{\mathsf{o}}_m) \approx 0$) and other back-projections $\mathbf{X}^{\mathsf{o}}_k$ are further away from the surfaces ($\Phi(\mathbf{X}^{\mathsf{o}}_k) \gg 0$), then we will find $w_m \to 1$ and the other $w_k \to 0$.

4.3 Physical Constraint Term

Consider $P(\mathbf{p}_1 \dots \mathbf{p}_M | \Phi_1 \dots \Phi_M)$ in Eq. (24). We decompose the joint probability of all object poses given all 3D object shapes into a product of per-pose probabilities:

$$P(\mathbf{p}_1 \dots \mathbf{p}_M | \Phi_1 \dots \Phi_M) = P(\mathbf{p}_1 | \Phi_1 \dots \Phi_M) \prod_{m=2}^{M} P(\mathbf{p}_m | \{\mathbf{p}\}_{-m}, \Phi_1 \dots \Phi_M) \tag{37}$$

where $\{\mathbf{p}\}_{-m} = \{\mathbf{p}_1 \dots \mathbf{p}_M\} \setminus \{\mathbf{p}_m\}$ is the set of poses excluding $\mathbf{p}_m$. We do not place any pose priors on any single objects, so we can ignore the factor $P(\mathbf{p}_1 | \Phi_1 \dots \Phi_M)$. The remaining factors can be used to enforce pose-related constraints.

Here we use them to avoid object collisions by discouraging objects from penetrating each other. The probability $P(\mathbf{p}_m | \{\mathbf{p}\}_{-m}, \Phi_1 \dots \Phi_M)$ is defined such that a surface point on one object should not move inside any other object. For each object m we uniformly and sparsely sample a set of K "collision points" $\mathscr{C}_m = \{\mathbf{C}^{\mathsf{o}}_{m,1} \dots \mathbf{C}^{\mathsf{o}}_{m,K}\}$ from its surface in object coordinates. K needs to be high enough to account for the complexity of the tracked shape, and not undersample parts of the model. We found throughout our experiments that $K = 1000$ insures sufficient coverage of the object to produce an effective collision constraint.

At each timestep the collision points are transformed into the camera frame as $\{\mathbf{C}^{\mathsf{c}}_{m,1} \dots \mathbf{C}^{\mathsf{c}}_{m,K}\}$ using the current pose $\mathbf{p}_m$. Denoting the partial union of SDFs $\{\Phi^{\mathsf{c}}_1 \dots \Phi^{\mathsf{c}}_M\} \setminus \{\Phi^{\mathsf{c}}_m\}$ by $\boldsymbol{\Phi}^{\mathsf{c}}_{-m}$ we write

$$P(\mathbf{p}_m | \{\mathbf{p}\}_{-m}, \Phi_1 \dots \Phi_M) \sim \frac{1}{K} \sum_{k=1}^{K} H^{\mathsf{out}} \left(\boldsymbol{\Phi}^{\mathsf{c}}_{-m}(\mathbf{C}^{\mathsf{c}}_{m,k}) \right) \tag{38}$$

where H^{out} is the offset smoothed Heaviside function already defined. If all the collision points on object m lie outside the shape union of objects excluding m this quantity asymptotically approaches 1. If progressively more of the collision points lie inside the partial shape union, the quantity asymptotically approaches 0.

The negative log-likelihood of Eq. (38) gives us the second part of the overall cost

$$\mathscr{E}_{\text{coll}} = - \sum_{m=1}^{M} \log \left(\frac{1}{K} \sum_{k=1}^{K} H^{\mathsf{out}} (\boldsymbol{\Phi}^{\mathsf{c}}_{m-}(\mathbf{C}^{\mathsf{c}}_{m,k})) \right). \tag{39}$$

The derivatives of this energy are computed analogously to those used for the data term (Eqs. 33 and 34), but with $\boldsymbol{\Phi}^{\mathsf{c}}(\mathbf{X}^{\mathsf{c}})$ replaced by $\boldsymbol{\Phi}^{\mathsf{c}}_{m-}(\mathbf{C}^{\mathsf{c}}_{m,k})$.

4.4 Optimisation

The overall cost is the sum of the data term and the collision constraint term

$$\mathscr{E} = \mathscr{E}_{\text{data}} + \mathscr{E}_{\text{coll}}. \tag{40}$$

To optimise the set of poses $\{\mathbf{p}_1 \dots \mathbf{p}_M\}$, we use the same Levenberg-Marquardt iterations and local frame pose updates as given in Sect. 3.

5 Implementation

We have coded separate CPU and GPU versions of our generalised multi-object tracker. Figure 8 shows the processing time per frame for the CPU implementation executing on an Intel Core i7 3.5 GHz processor with OpenMP support as the number of objects tracked is increased. As expected, the time rises linearly with the number of objects. With two objects the CPU version runs at around 60 Hz, but above five

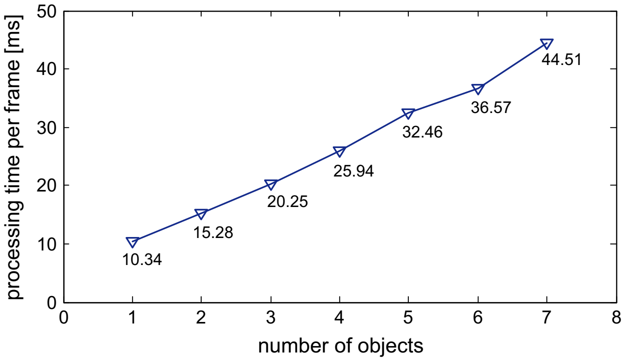


Fig. 8 The processing time per frame in milliseconds of the multi-object tracker implemented on the CPU rises linearly with the the number of objects tracked

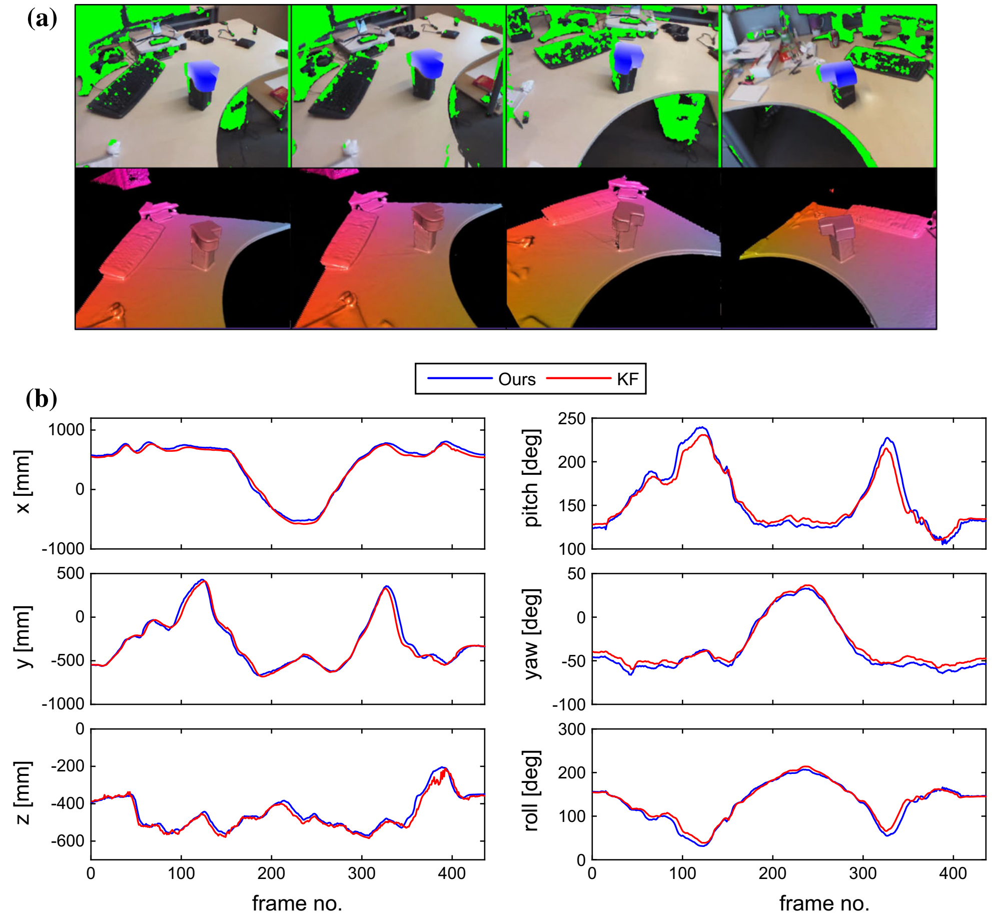


Fig. 9 A quantitative comparison of camera pose output obtained using the present method on a single object and from using KinectFusion on the entire scene. **a** Frames from the two approaches. *Top row* the tracked object using our method. *Bottom row* camera track from KinectFusion. **b** The 6 degrees of freedom in pose compared. Translation is measured in mm and rotation is measured in degrees.

objects the process is at risk of falling below frame rate. The accelerated version, running on an Nvidia GTX Titan Black GPU and same CPU, typically yields a 30% speed-up in the experiments reported below. The rate is not greatly increased because the GPU only applies full leverage to image pixels that backproject into the 3D voxelised volumes around objects. In the experiments here, the tracked objects typically occupy a very small fraction (i.e. just a few %) of the RGB-D image, involving only a few thousands of pixels, insufficient to exploit massive parallelism.

6 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 can be found in the video at https://youtu.be/BSkUee3UdJY. (NB: to be replaced by an official archival site).

6.1 Quantitative Experiments

We ran three sets of experiments to benchmark the tracking accuracy of our algorithms. First we compare the camera trajectory obtained by our algorithm tracking a single stationary object against that obtained by the KinectFusion algorithm of Newcombe et al. (2011) tracking the entire world map. Several frames from the sequence used are shown in Fig. 9a and the degrees of freedom in translation and rotation are compared in Fig. 9b. Despite using only the depth pixels corresponding to the object (an area of the depth image considerably smaller than that employed by KinectFusion) our algorithm obtains comparable accuracy. It should be noted that this is not a measure of ground truth accuracy: the trajectory obtained by the KinectFusion is itself just an estimate.

In our second experiment, we follow a standard benchmarking strategy from the markerless tracking literature and evaluate our tracking results on synthetic data to provide ground truth. We move two objects of known shape in front of a virtual camera and generate RGB-D frames. The objects

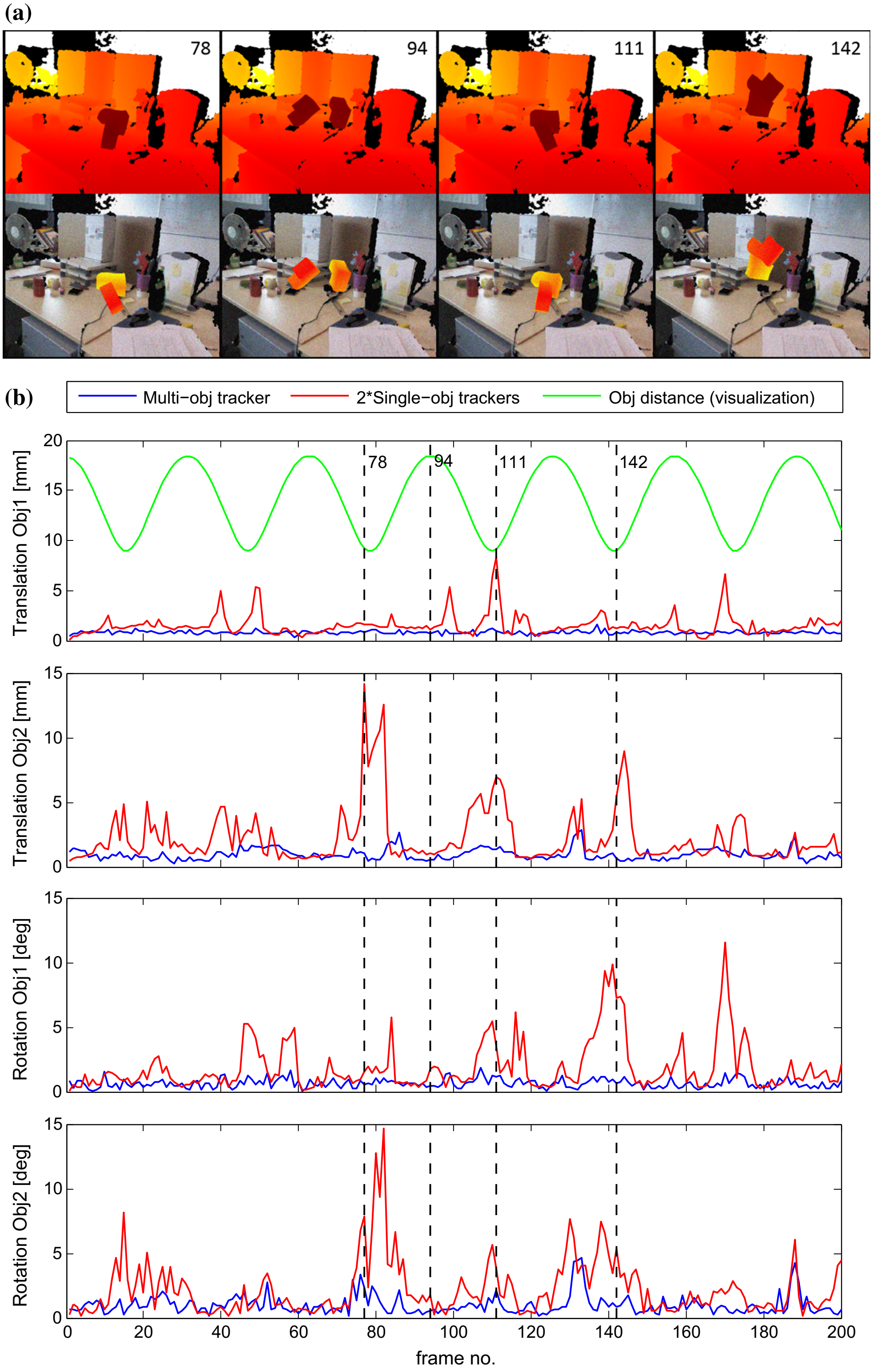


Fig. 10 A comparison of pose estimation error between our generalised multi-object tracker and two instances of our single object method. **a** Four examples of the synthetic RGB-D frames with the frame number corresponding to the marks on the pose graphs in **b**. **b** As the objects are periodically brought closer, so the pose error (*red*) of the two independent trackers increases (Color figure online)

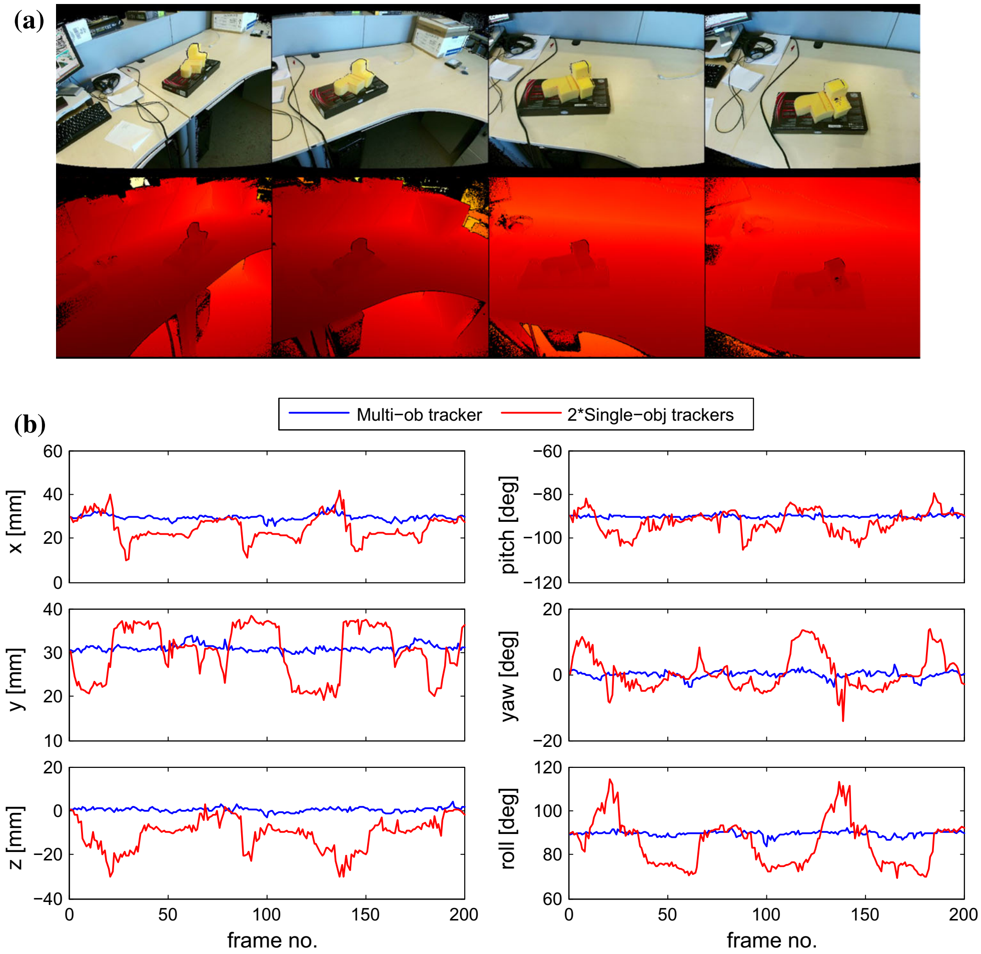


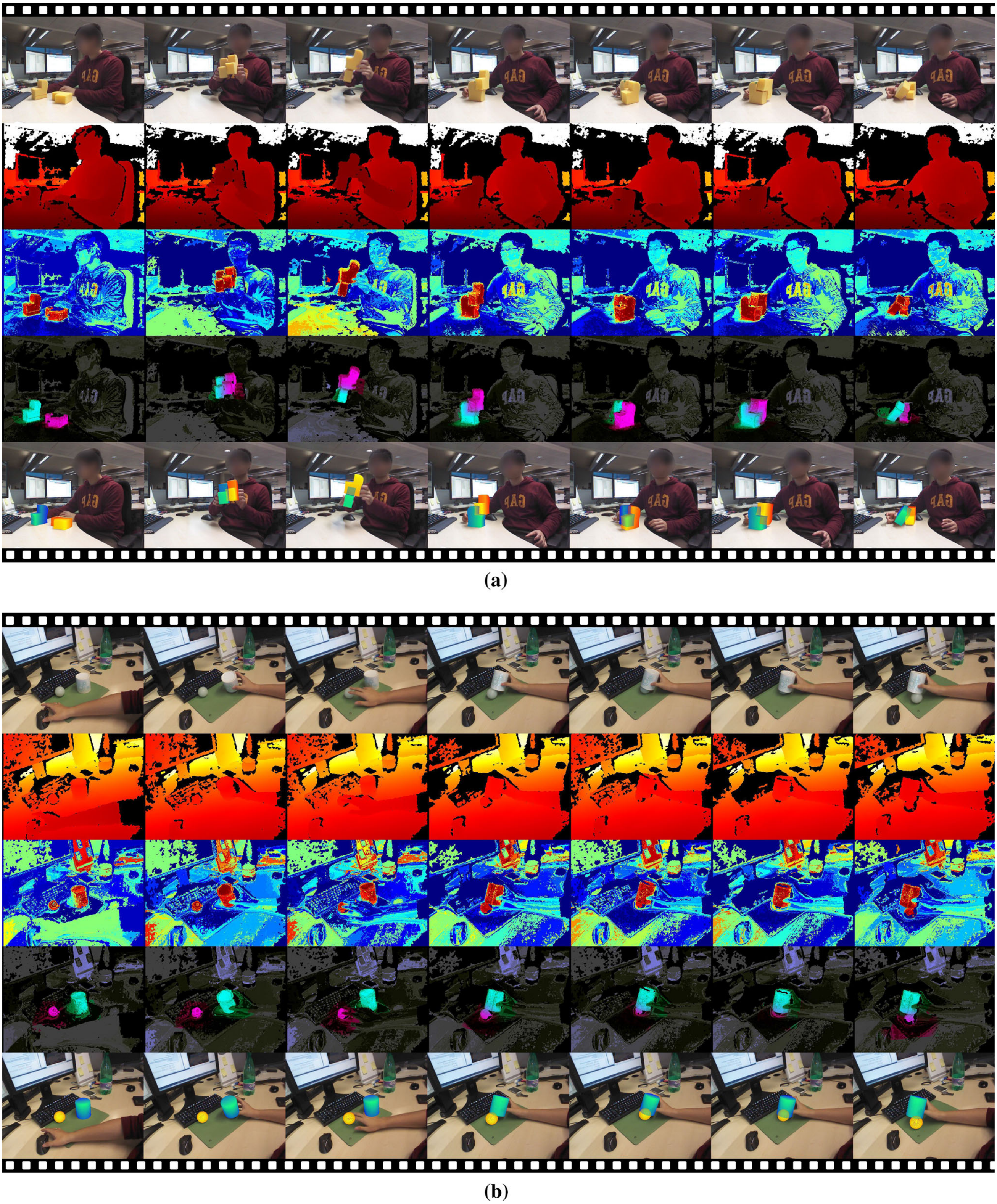
Fig. 11 Comparison of the difference in relative pose estimation between our multi-object tracker and two instances of our single-object tracker using real data. **a** Sample frames **b** Pose recovery compared: the multiobject tracker (*blue*) is stable (Color figure online)

periodically move further apart then closer to each other. Realistic levels of Gaussian noise are added to both the rendered colour and the depth images. Four sample frames from the test sequence are shown in Fig. 10a. Using this sequence we compare the tracking accuracy of our generalised multi-object tracker with two instances of our single object tracker. 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 R_g and we estimate the rotation matrix R_e. The error value is averaged over the three including angles of the resulting vectors:

$$r_{err} = \frac{1}{3} \sum_{i \in x,y,z} \cos^{-1} \left((\mathrm{R}_e \mathbf{e}_i)^\top \mathrm{R}_g \mathbf{e}_i) \right) \tag{41}$$

In the graphical results of Fig. 10b the green line shows the relative distance between the two objects. Note that this value has been scaled and offset for visualisation. 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 multi-object tracker provide accurate results. However, once the two objects move close together, the two separate single object trackers produce 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.

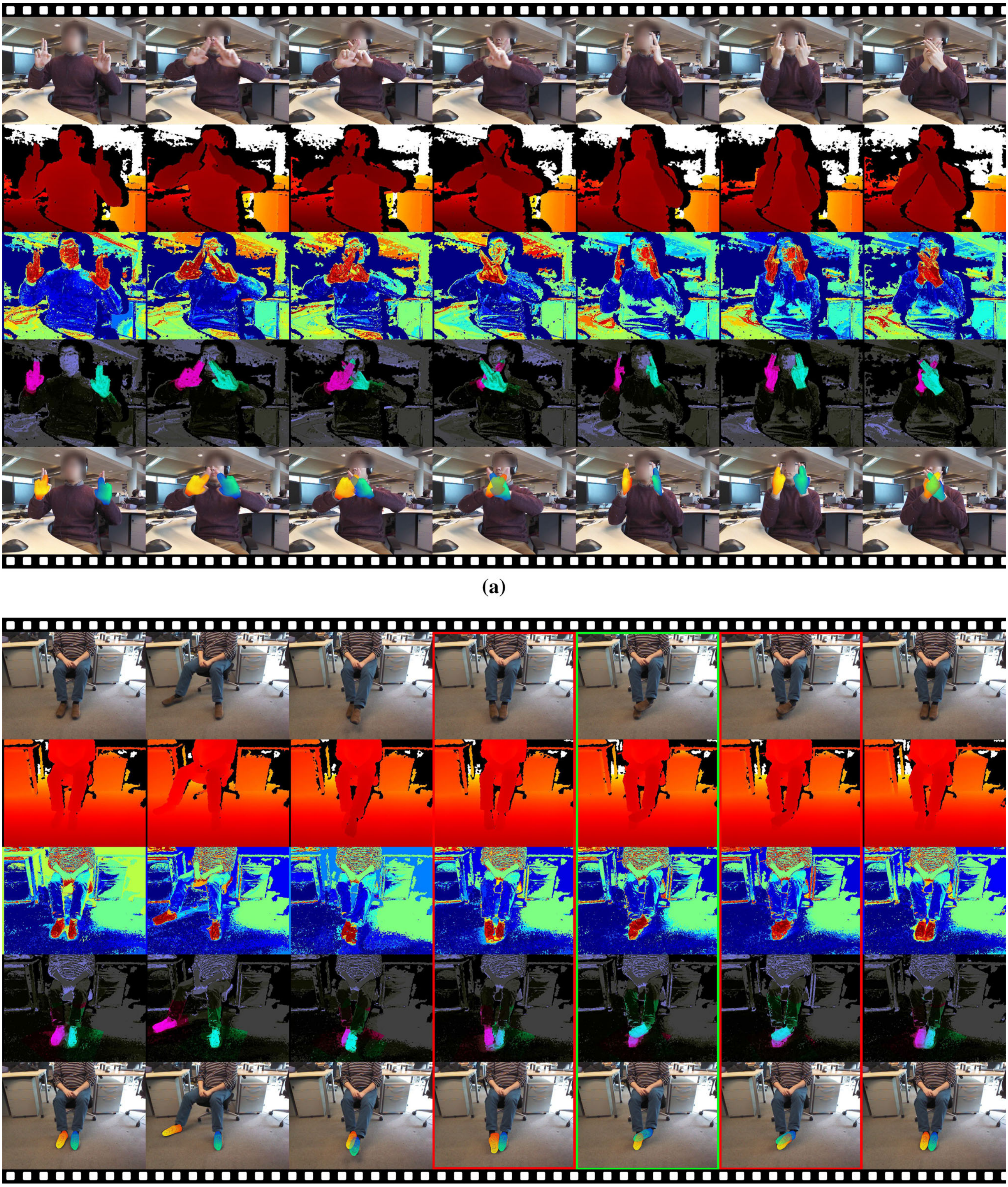
The third quantitative experiment (Fig. 11) makes a similar comparison, but with real imagery. As before, it is difficult to obtain the absolute ground truth pose of the objects, and instead we measure the consistency of the relative pose between two static objects by moving the camera around while looking towards the two objects. Example frames are shown in Fig. 11a. If the two recovered poses are accurate we would expect consistent relative translation and rotation through the whole sequence. As shown in Fig. 11b, our multi-object tracker is able to recover much more consistent relative translation and rotation than two independent instances of our single object tracker.



(a)

(b)

Fig. 12 Film strips showing our algorithm tracking accurately known object models: **a** two pieces of formed sponge, and **b** a ball and a cup. In each, *Rows 1&2* show the *colour* and the depth image inputs. *Row 3* is per-pixel foreground probability P^f. *Row 4* is the per-pixel membership weight w_i, *magenta and cyan* olour correspond to the two objects, the *blue coloured* pixels are with ambiguous membership. *Row 5* shows the tracking result (Color figure online)



(a)

(b)

Fig. 13 Film strip showing our algorithm tracking two cases where the models are inaccurate: **a** two hands and **b** two feet. The rows are as in Fig. 12

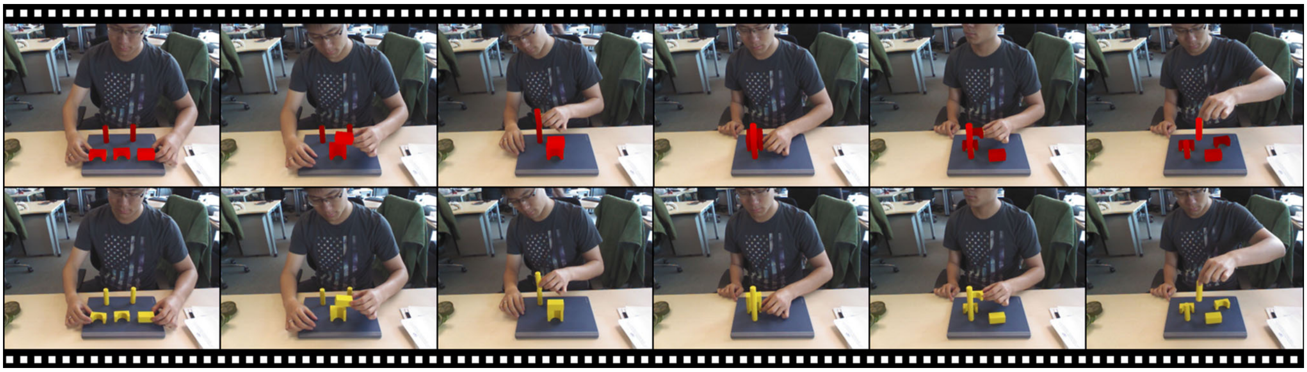


Fig. 14 A film strip showing a very challenging sequence where 5 pieces of toy bricks with identical *colour* are tracked. The top sequence shows the tracking result rendered on the *colour* image and the sequence below shows the original *colour* images. Our multi-object tracker manages to track through the whole sequence without tracking failure

6.2 Qualitative Experiments

We use five challenging real sequences to illustrate the robust performance of our multi-object tracker.

In Fig. 12 we use *accurate*, hand crafted models for tracking. Figure 12a shows the tracking of two pieces of sponge with identical shape and appearance models. Rows 1 and 2 of the figure show the colour and the depth image inputs, and Row 3 shows the per-pixel foreground probability P^f. Row 4 shows the per-pixel membership weight w_m. The two objects, one with $w_1 \gg 0.5$, $w_2 \ll 0.5$ and the other with $w_2 \gg 0.5$, $w_1 \ll 0.5$, are highlighted in magenta and cyan respectively. The blue highlighted pixels have ambiguous membership ($w_1 \approx w_2 \approx 0.5$). The darkened pixels are background, as obvious from Row 3. The final tracking result is shown as Row 5.

The tracker is able to track through heavy occlusions and handle challenging motions. This is a result of the region based nature of our approach, which makes it robust to missing or occluded parts of the tracked target, as long as these do not introduce extra ambiguity in the shape to pose mapping.

In Fig. 12b we simultaneously track a white cup and a white ball to demonstrate the effectiveness of the physical collision constraint. Even though there is no depth observation from the ball owing to significant occlusion from the cup, our algorithm can still estimate the location of the ball. This happens because (i) the physical constraint prevents the ball from intersecting with the cup and (ii) the table is a different colour from the ball, which prevents the ball from overlapping with the table.

As a contrast, Fig. 13 illustrates our tracker using previously reconstructed and hence somewhat *inaccurate* 3D shapes. First in Fig. 13a we track two interacting hands (fixed hand articulation pose). Even though the hand models do not fit the observation perfectly—indeed they are models of hands from a different person obtained using the algorithm (Ren et al. 2013)—the tracker still recovers the poses of both by finding the local minimum that best explains the colour and depth observations.

In Fig. 13b we track two interacting feet with a pair of approximate 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 shown in column 4 of Fig. 13b, where the shoe is incorrectly rotated. This happens because the 3D model is somewhat 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, leading to failure. We note that the tracker does automatically recover from both failure cases. As soon as the feet move out of the ambiguous position, the multi-object tracker uses the previous incorrect pose as initialization and converges to the correct pose at the current frame.

In Fig. 14 we show a challenging sequence where five toy bricks are tracked, illustrating that the proposed tracker is able to handle larger number of objects. All the objects in the toy set have the same colour and some also have identical shapes. The top sequence shows the tracking result and the bottom sequence shows the original colour input. In spite of the heavy self-occlusion and the occlusion introduced by hands, the multi-object tracker is able to track robustly. Importantly, there is no bleeding of one object into another when blocks are placed together then separated.

7 Conclusions

In this paper we presented a novel framework for tracking single and multiple 3D objects from a sequence of RGB-D images. Our method is particularly well suited to 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, yielding weights that indicate the probability of individual image observations being generated by each of the tracked objects, thus implicitly solving the data association problem. Furthermore, in the multi-object case, the formulation leads to a natural imposition of a physical constraint term, allowing 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, the formulation would allow for generic interaction forces between objects to be modelled.

We validate our claims with several experiments, showing both robustness and accuracy. For this evaluation we used an implementation that can easily track multiple objects in real-time without the use of any GPU acceleration.

Since the tracker is region-based and currently uses simple histograms as appearance models, it is particularly well suited to objects where the texture is uninformative. A possible direction of research is to transfer our tracking framework to 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 area to explore further, as tracking individual fingers might greatly benefit from a method that can handle near-identical appearance and imposes collision constraints.

Acknowledgements This work was funded by the Project REWIRE (Grant No. 287713) under the EU 7th Framework Programme, by Grants EP/H050795 and EP/J014990 from the UK's Engineering and Physical Science Research Council, and by a Laureate Fellowship (FL130100102) to IDR from the Australian Research Council.

References

Azad, P., Münch, D., Asfour, T., & Dillmann, R. (2011). 6-dof model-based tracking of arbitrarily shaped 3D objects. In *Proceedings 2011 IEEE International Conference on Robotics and Automation*, (pp. 5204–5209), doi:10.1109/ICRA.2011.5979950.

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256. doi:10.1109/34.121791.

Choi, C., & Christensen, H. (2010). Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *Proceedings 2010 IEEE International Conference on Robotics and Automation*, (pp. 4048–4055), doi:10.1109/ROBOT.2010.5509171.

Choi, C., & Christensen, H. I. (2013), RGB-D object tracking: a particle filter approach on GPU. In *Proceedings IEEE/RSJ Conference on Intelligent Robots and Systems*, (pp. 1084–1091), doi:10.1109/IROS.2013.6696485.

Drummond, T., & Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 932–946. doi:10.1109/TPAMI.2002.1017620.

Fitzgibbon, A. W. (2001). Robust registration of 2D and 3D point sets. In *Proceedings 12th British Machine Vision Conference*, (pp. 662–670).

Gennery, D. B. (1992). Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, *7*(3), 243–270.

Harris, C., & Stennett, C. (1990). RAPiD—a video rate object tracker. In *Proceedings 1st British Machine Vision Conference*.

Held, R.T., Gupta, A., Curless, B., & Agrawala, M. (2012). 3D puppetry: A Kinect-based interface for 3D animation. In *Proceedings ACM Symposium on User Interface Software and Technology*, (pp. 423–434), doi:10.1145/2380116.2380170.

Kim, K., Lepetit, V., & Woo, W. (2010). Keyframe-based modeling and tracking of multiple 3d objects. In *Proceedings 9th IEEE/ACM International Symposium on Mixed and Augmented Reality*, (pp. 193–198), doi:10.1109/ISMAR.2010.5643569.

Klein, G., & Murray, D. W. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings 6th IEEE/ACM International Symposium on Mixed and Augmented Reality*, (pp. 225–234), doi:10.1109/ISMAR.2007.4538852.

Kyriazis, N., & Argyros, A. A. (2013). Physically plausible 3D scene tracking: The single actor hypothesis. In *Proceedings 26th IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 9–16), doi:10.1109/CVPR.2013.9.

Lowe, D. G. (1992). Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, *8*(2), 113–122.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. doi:10.1023/B:VISI.0000029664.99615.94.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., & Fitzgibbon, A. W. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings 10th IEEE/ACM International Symposium on Mixed and Augmented Reality*, (pp. 127–136), doi:10.1109/ISMAR.2011.6092378.

Oikonomidis, I. (2012) Tracking the articulated motion of two strongly interacting hands. In *Proceedings 25th IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 1862–1869), doi:10.1109/CVPR.2012.6247885.

Oikonomidis, I., Kyriazis, N., & Argyros, A. A. (2011a). Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings 21st British Machine Vision Conference*, (pp. 1–11), doi:10.5244/C.25.101.

Oikonomidis, I., Kyriazis, N., & Argyros, A.A. (2011b) Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proceedings 13th IEEE International Conference on Computer Vision*, (pp. 2088–2095), doi:10.1109/ICCV.2011.6126483.

Prisacariu, V. A., & Reid, I. D. (2012). PWP3D: Real-time segmentation and tracking of 3D objects. *International Journal of Computer Vision*, *98*, 335–354. doi:10.1007/s11263-011-0514-3.

Prisacariu, V. A., Segal, A. V., & Reid, I. D. (2012). Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. In *Proceedings 11th Asian Conference on Computer Vision*, (pp. 593–606), doi:10.1007/978-3-642-37331-2_45.

Prisacariu, V. A., Kähler, O., Murray, D. W., & Reid, I. D. (2013). Simultaneous 3D tracking and reconstruction on a mobile phone. In *Proceedings 12th IEEE/ACM International Symposium on Mixed and Augmented Reality*, (pp. 89–98), doi:10.1109/ISMAR.2013.6671768.

Ren, C. Y., & Reid, I. D. (2012). A unified energy minimization framework for model fitting in depth. In *Proceedings ECCV Workshops (2)*, (pp. 72–82), doi:10.1007/978-3-642-33868-7_8.

Ren, C. Y., Prisacariu, V. A., Murray, D. W., & Reid, I. D. (2013). STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data. In *Proceedings 14th IEEE International Conference on Computer Vision*, doi:10.1109/ICCV.2013.197.

Ren, C. Y., Prisacariu, V. A., Kähler, O., Murray, D. W., & Reid, I. D. (2014). 3D tracking of multiple objects with identical appearance using RGB-D input. In *Proceedings International Conference on 3D Vision*, (pp. 47–54), doi:10.1109/3DV.2014.39.

Shuster, M. D. (1993). A survey of attitude representations. *Journal of the Astronautical Sciences*, *41*(4), 439–517.

Smith, R. (2006). Open dynamics engine. www.ode.org.

Sturm, J., Bylow, E., Kahl, F., & Cremers, D. (2013). CopyMe3D: Scanning and printing persons in 3D. In *Proceedings 35th German Conference on Pattern Recognition*, (pp. 405–414), doi:10.1007/978-3-642-40602-7_43.

Ueda, R. (2012). Tracking 3D objects with Point Cloud Library. www.pointclouds.org.

Wuthrich, M., Pastor, P., Kalakrishnan, M., Bohg, J., & Schaal, S. (2013). Probabilistic object tracking using a range camera. In *Proceedings IEEE/RSJ Conference on Intelligent Robots and Systems*, (pp. 3195–3202), doi:10.1109/IROS.2013.6696810.