STAR3D: Simultaneous Tracking And Reconstruction of 3D Objects Using RGB-D Data

Carl Yuheng Ren†, Victor Prisacariu†, David Murray† and Ian Reid‡

†Department of Engineering Science, University of Oxford
‡School of Computer Science, University of Adelaide
†{carl, victor, dwm}@robots.ox.ac.uk
‡ian.reid@adelaide.edu.au

Abstract

We introduce a probabilistic framework for simultaneous tracking and reconstruction of 3D rigid objects using an RGB-D camera. The tracking problem is handled using a bag-of-pixels representation and a back-projection scheme. Surface and background appearance models are learned online, leading to robust tracking in the presence of heavy occlusion and outliers. In both our tracking and reconstruction modules, the 3D object is implicitly embedded using a 3D level-set function. The framework is initialized with a simple shape primitive model (e.g. a sphere or a cube), and the real 3D object shape is tracked and reconstructed online. Unlike existing depth-based 3D reconstruction works, which either rely on calibrated/fixed camera set up or use the observed world map to track the depth camera, our framework can simultaneously track and reconstruct small moving objects. We use both qualitative and quantitative results to demonstrate the superior performance of both tracking and reconstruction of our method.

1. Introduction

Many applications need an accurate 3D model of a rigid object, but without access to a predefined CAD model or similar, the standard acquisition method involves 3D scanning using a precisely calibrated multi-camera or range sensor system. This is usually extremely costly and difficult to setup, and often slow. In this paper we introduce a framework for simultaneous tracking and reconstruction of unknown 3D rigid objects that is simple, fast and effective. The system is initialized with a simple primitive 3D shape (e.g. a sphere or a cube), then the 3D shape of the object being tracked is reconstructed incrementally online. This flexible framework for 3D reconstruction and tracking has many real-world applications. For example, it allows users to pick a random rigid object from their home, scan it and then use it as a controller to interact with a computer. The proposed framework comprises two modules: a tracking module and a reconstruction module. Next we review recent works related to each module.

3D Tracking. Most existing research work for 3D tracking with depth data uses a model-based approach, which generates pose hypotheses and evaluates them on the observed depth/RGB-D data. To find the best pose hypothesis, such methods define and minimise an objective function measuring the discrepancy between expected (from the model hypothesis) and observed visual cues. A common approach is Iterative Closest Point (ICP) [2]. For example, in [4], the authors use Kinect input to track hand-held 3D puppets (rigid objects). The system yields robust and real-time performance for tracking rigid objects, but accurate 3D models of puppets with color and textures need to be built off-line, in advance. The occlusion from the hand is handled by a color-based segmenter.

More general is the work KinectFusion [7], where the whole scene structure and camera pose are estimated simultaneously. Ray-casting is used to establish point correspondences between the observed point cloud and the reconstructed world map, and alignment achieved using ICP. However, the ICP-based tracking in KinectFusion relies on a static world map, which makes the method unable to track small moving objects in a static scene. Camera motion and (inverse) object motion of course induce identical image changes and the authors note in [5] that KinectFusion could be to reconstruct a moving 3D rigid object from a static Kinect, but in this case the object needs to be large enough to occupy the majority of the depth map.

Another school of the RGB-D based trackers [8, 11] uses sampling-based methods. These rely on the many evaluations of the objective function at arbitrary points in the pose hypothesis space. In [8], the authors use Particle Swarm Optimization (PSO) to solve the articulated hand tracking problem. The system is implemented efficiently on the GPU, yielding real-time performance and fast-recovery from tracking failure. Another similar work that tracks rigid ob-

ject is [11], which uses a particle filter to solve the pose optimization. Both works measure the discrepancy between the expected depth generated by the pose hypothesis and the observed one and are still very computational expensive (even with a GPU implementation), as they require a large number of energy function evaluations.

In [10], the authors use a gradient-based optimization method to solve the tracking problem but do not explicitly establish point correspondences between the model and the observation. Instead, they use 3D level-set embedding function to encode the 3D object model and, by back-projecting the observed depth image into object coordinates, they are able to take advantage of the gradient of the level-set function to guide the search for the pose. This method is both very computationally efficient and robust to missing data. However, the energy function only considers the fitting of the depth data to the surface of the object, making the tracker very sensitive to close-to-surface outliers (e.g. a hand held object). The tracker module presented in this work extends the back-projection scheme of [10] by formulating a probabilistic model to use both color and depth information, resulting in more robust and accurate tracking.

3D Reconstruction. Most traditional 3D reconstruction methods require a calibrated multiple camera setup and are based on space carving. The introduction of customizable, frame-rate RGB-D cameras has made 3D reconstruction using a single depth camera possible. KinectFusion [7] is among the most successful systems for real-time 3D reconstruction. With a single hand-held Kinect device, KinectFusion can incrementally reconstruct the surface of the physical world that the camera sees, in real-time. However, as discussed in previous subsection, KinectFusion relies heavily on the world map to track the camera, thus it can not reconstruct small moving objects in static scenes. Another related recent work is [12] where the authors use a single, fixed, un-calibrated Kinect to scan human body in a home environment. Accurate 3D human shapes are obtained by combining multiple monocular views of a person moving in front of the sensor. The SCAPE model [1] is used to constrain the alignment of the multiple depth maps from the various views. In [9], the authors use monocular 2D image cues to reconstruct 3D shapes. The reconstruction is constrained by a learnt low-dimensional 3D shape space. Both [12] and [9] are capable of reconstructing 3D objects with a single camera. However, they both rely heavily on a learnt shape space to constrain the reconstruction. This forces the reconstructed objects to be within a fixed, prelearned category.

Contributions. In this paper, we present a probabilistic framework for simultaneous tracking and reconstruction of an unknown rigid object using a single RGB-D camera. We highlight our contributions below.

1. We introduce a probabilistic model for model-based 3D tracking using RGB-D images. The proposed probabilistic model leads to a differentiable energy function, which can be efficiently solved by gradient-based optimization method. Our method yields real-time performance on the GPU and is robust to missing data, occlusion and outliers.

2. We extend the space carving approach by introducing a novel probabilistic framework for the reconstruction of unknown 3D objects. An inside/outside volumetric model of the object is learnt incrementally online, and the 3D shape is reconstructed by evolving a 3D level-set embedding function on this inside/outside model. The reconstruction method can be implemented in a massively parallel fashion, resulting in great computational efficiency.

2. Generative Model

Assuming calibrated color and depth cameras (i.e. aligned color-depth frames), let Ω_d be the depth image domain, and Ω_c be the color image domain. Ω is the RGB-D image domain, obtained by re-projecting the color image onto the depth image. A pixel $x \in \Omega$ at image coordinates (u, v) has depth value d and color value y. We use a dot notation to denote the homogeneous coordinate of x: $\dot{x} = (ud, vd, d)^\top \in \mathbb{R}^3$. A depth pixel $\dot{x}$ in the object region is projected from a 3D point $\dot{\mathbf{X}} = (X, Y, Z, 1)^\top$ on the object surface as:

$$\dot{x} = \mathtt{A}\mathtt{T}_{o,c}\dot{\mathbf{X}} \qquad \mathtt{T}_{o,c} = [R|t] \in \mathbb{SE}_3 \tag{1}$$

where the Euclidean group $\mathbb{SE}_3 := \{R, t | R \in \mathbb{SO}_3, t \in \mathbb{R}^3\}$, $\mathtt{T}_{o,c}$ is the 6 DoF pose parametrised by the pose parameter $\mathbf{p}$ that transforms from *object coordinates* to *camera coordinates*. $\mathtt{A}$ is the depth camera's intrinsic matrix.

We represent the shape of the 3D object by a 3D signed distance function (SDF) $\mathbf{\Phi}(\mathbf{X})$ defined in an object coordinate frame. The surface of the 3D shape is recovered as the zero level, $\mathbf{\Phi}(\mathbf{X}) = 0$. The domain outside maps to positive values, and the domain inside the object maps to negative values. Although our objective is to recover $\mathbf{\Phi}$ it is also useful to introduce a representation of the shape as a "bag-of-voxels", $\{\mathbf{X}, V\}$ indexed by i, in which $\mathbf{X}_i$ is the location of a voxel in object coordinates, and V_i an indicator variable that can take on values *in*, *on* or *out* for inside the shape, on the surface, or outside the shape, respectively.

Two appearance models are used to describe the color statistics of the scene: one for the object surface, which generates the foreground region in the image; and one for the background. These are represented by their likelihoods, $P(y|V)$, where V can take on values on or out, because a pixel inside the volume can never generate a pixel in Ω. The two appearance models are represented with RGB color histograms using 32 bins per channel. The histogram can be initialized either from a detection module or from a user-selected bounding box on the RGB image, in which the

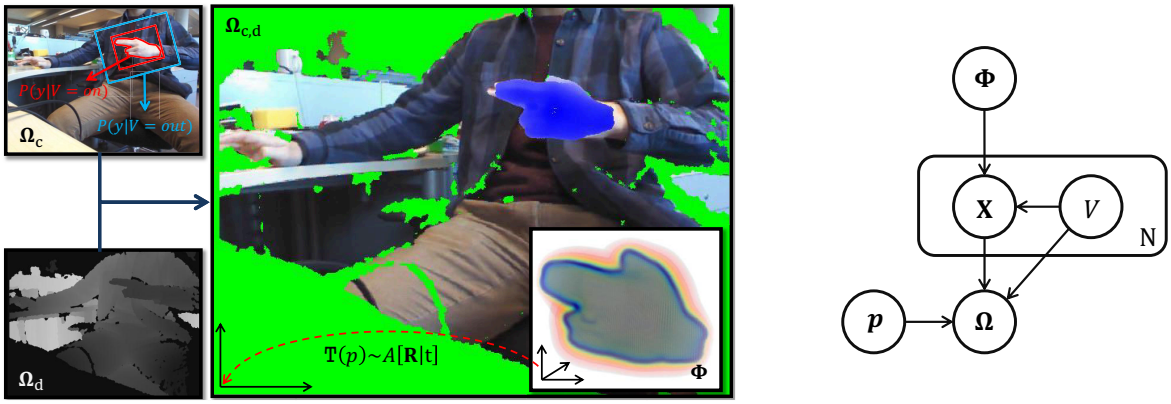

Figure 1. (Left): Representation of the 3D model $\mathbf{\Phi}$, the RGB-D image domain $\mathbf{\Omega}_{c,d}$, the surface / background models $P(y|V = on)$ / $P(y|V = out)$ and the pose $\mathtt{T}(\mathbf{p})$. (Right): Graphical model of our generative model for tracking and reconstruction.

foreground model is built from the interior of the bounding box and the background from the immediate region outside the bounding box. These initial color likelihoods are used in conjunction with the local depth information both for tracking and reconstruction, and are refined over time. An illustration of all notations are demonstrated in Fig. 1(left).

Fig. 1(right) shows the graphical model for our system. We use this model both for tracking and for reconstruction, but these two aspects make use of the information in different ways. In this model the shape $\mathbf{\Phi}$ generates a set of voxels $\{\mathbf{X}, V\}$ (indexed by i). This volumetric model, combined with the object pose $\mathbf{p}$, in turn generates the observed RGB-D images Ω comprising pixels $\{x, y\}$ (indexed by j).

The full joint distribution corresponding to the model is:

$$P(\Omega_0 \dots \Omega_T, \mathbf{p}_0 \dots \mathbf{p}_T, \mathbf{\Phi}, \{\mathbf{X}, V\}) = P(\mathbf{\Phi}) \prod_i P(\mathbf{X}_i, V_i|\mathbf{\Phi}) \prod_t P(\Omega_t|\{\mathbf{X}, V\}, \mathbf{p}_t) P(\mathbf{p}_t) \quad (2)$$

and our objective is to find the optimal sequence of poses and the shape, given the observed RGB-D images:

$$\max_{\mathbf{\Phi}, \mathbf{p}_0 \dots \mathbf{p}_t} P(\mathbf{\Phi}, \mathbf{p}_0 \dots \mathbf{p}_t|\Omega_0 \dots \Omega_t) \quad (3)$$

Note that there are further justifiable simplifications that can be made to 2. First, we assume that the observations are pixel-wise independent meaning that $P(\Omega_t|\{\mathbf{X}, V\}, \mathbf{p}_t)$ decomposes into a product of terms $P(x_j|\dots)$. Second, note that the form of the equation permits a recursive update of $\mathbf{\Phi}$ and $\{\mathbf{X}, V\}$ which are the constant terms, so all the evidence from frames 0 up to $t-1$ can be fused easily with that at the current frame. Finally, note that in this model, the locations of voxels $\mathbf{X}$ are treated as generated randomly from the shape $\mathbf{\Phi}$. Under this model all voxels locations have the same probability of being generated but in practice the situation is more certain, with every voxel being generated exactly once. The variables $\mathbf{X}$ are maintained in the model for convenience, but it is the indicator variable of each voxel V that carries the important information about the volumetric model.

Even with these simplifications, full inference is extremely difficult. In the remainder of the paper we perform approximate inference by finding MAP or maximum likelihood estimates of $\mathbf{\Phi}$ and $\mathbf{p}_t$, alternating steps that estimate the current pose given the (current estimate of) shape, and estimating the fixed shape by assuming knowledge of the current and past poses. We show in Section 5 that this yields good results in practice.

3. Tracking

For tracking, we assume known shape $\mathbf{\Phi}$ and optimise the pose at time t (dropping the subscript on the pose $\mathbf{p}$ henceforth) by maximising the likelihood $P(\Omega|\mathbf{\Phi}, \mathbf{p})$ as a function of $\mathbf{p}$. To optimise this conditional distribution we treat the RGB-D image Ω as a bag-of-independent-pixels $\{x, y\}$. Though not all voxels generate a pixel, each pixel x is generated by a unique voxel $\mathbf{X}$, where $\mathbf{X}$ is sampled from $\mathbf{\Phi}$, and x is its (deterministic) projection into the image. Its color is sampled from the appropriate model conditioned on V. The likelihood is the product over all pixel likelihoods:

$$\mathcal{L}(\mathbf{p}) = \prod_j P(x_j, y_j, V_{i(j)}|\mathbf{\Phi}, \mathbf{p}) \quad (4)$$

where $i(j)$ indicates that voxel i projects to pixel j.

This generative model is very similar to [3], which uses level-sets to track 2D deformable objects. In [3] the image and the level-set embedding function are in the same 2D domain, and each value in the level-set function is associated with a pixel in the image domain. The tracking is done by maximizing the discrepancy between the foreground/background region with the Heaviside function of the level-set embedding selecting either foreground or background. However, in our case, the level-set function is defined in 3D space, and all pixels in the RGB-D image domain are generated either from the object surface or from

outside the object. No pixel is generated from the interior of the model, and there is not a one-to-one mapping between pixels in the RGB-D image and voxels. These differences lead to subtle but important differences in the formulation.

In our work, the per-pixel likelihood of the pose (in which have marginalised V) is:

$$P(x, y|\mathbf{\Phi}, \mathbf{p}) = \sum_{k=on,out} \{P(x|\mathbf{\Phi}, \mathbf{p}, V{=}k)P(V{=}k|y)\} \tag{5}$$

The pixel location likelihoods for the foreground and background are simply uniform distributions:

$$P(x|\mathbf{\Phi}, \mathbf{p}, V{=}on) = \frac{\delta_\epsilon(\mathbf{\Phi}(\mathbf{X}))}{\eta_f} \tag{6}$$

$$P(x|\mathbf{\Phi}, \mathbf{p}, V{=}out) = \frac{H_\epsilon(\mathbf{\Phi}(\mathbf{X}))}{\eta_b} \tag{7}$$

$$\eta_f = \sum_\Omega \delta_\epsilon(\mathbf{\Phi}(\mathbf{X}_i)) \qquad \eta_b = \sum_\Omega H_\epsilon(\mathbf{\Phi}(\mathbf{X}_i)) \tag{8}$$

where $\mathbf{X} = \mathtt{T}_{o,c}^{-1}\mathtt{A}^{-1}\dot{x}$ is the back-projection into object coordinates of the RGB-D pixel x. H_ϵ and δ_ϵ are the smoothed Heaviside and Dirac delta functions, and thus select the outside of the object and the surface of the object respectively.

Substituting the likelihoods for x into Eqn. 5 and assuming pixel-wise independence, we obtain pose likelihood as:

$$P(\Omega|\mathbf{\Phi}, \mathbf{p}) \sim \prod_\Omega \{P_f \delta_\epsilon(\mathbf{\Phi}(\mathbf{X}_i)) + P_b H_\epsilon(\mathbf{\Phi}(\mathbf{X}_i))\} \tag{9}$$

where $P_f = P(y|V{=}on)$ and $P_b = P(y|V{=}out)$. This can be written as an energy summation by taking logs. We differentiate the energy function w.r.t the pose parameter $\mathbf{p}$:

$$\frac{\partial E}{\partial \mathbf{p}} = \sum_{\Omega_{c,d}} \left\{ \frac{\left(P_f \frac{\partial \delta_\epsilon}{\partial \mathbf{\Phi}} + P_b \frac{\partial H_\epsilon}{\partial \mathbf{\Phi}}\right)}{P(x_i, y_i|\mathbf{\Phi}, \mathbf{p})} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{X}_i} \frac{\partial \mathbf{X}_i}{\partial \mathbf{p}} \right\} \tag{10}$$

and optimise pose $\mathbf{p}$ using Levenberg-Marquardt.

4. Reconstruction

For the purposes of reconstruction, we initialize the tracker with a simple initial model (e.g. a sphere), and iterate the tracker until it converges to a pose that projects the initial model close to the object region in the RGB-D image domain. With the pixel-wise foreground/background posterior P_f/P_b in Section 3, we remove all the background pixels (where $P_f < P_b$) in the RGB-D image. We use $\hat{\Omega}$ to denote this new foreground-only RGB-D image domain. The reconstruction runs on each $\hat{\Omega}$ and the reconstructed 3D model is used for tracking in the next frame. Our approach is similar to 2D level-set based image segmentation methods but operating over a 3D volume. More specifically, we evolve a 3D level-set embedding function over an inside/outside probability volume to maximize the per-voxel posterior probability of the 3D level-set function, given the shape prior and all previously observed depth and poses.

In the reconstruction step, we assume the pose of object given by the tracker is fixed and we optimize

$$\begin{aligned} &P(\mathbf{\Phi}|\hat{\Omega}_{0\ldots t}, \mathbf{p}_{0\ldots t}) \propto P(\hat{\Omega}_{0\ldots t}|\mathbf{\Phi}, \mathbf{p}_{0\ldots t})P(\mathbf{\Phi}) \\ &= \sum_{k=in,out} P(\hat{\Omega}_{0\ldots t}|V{=}k, \mathbf{p}_{0\ldots t})P(\mathbf{\Phi}|V{=}k)P(V{=}k) \end{aligned} \tag{11}$$

Note that the equation above applies per voxel $\mathbf{X}$, with V as the corresponding inside/outside membership of $\mathbf{X}$.

Taking the two terms in the summation in turn, first we develop the likelihood of generating an RGB-D image Ω given the pose and voxel memberships. We decompose this into the per-voxel likelihood i.e. the likelihood that the single voxel $(\mathbf{X}, V)$ generated the RGB-D pixel x as follows:

$$\begin{aligned} L^{in}(V) &= P(x|\mathbf{X}, V, \mathbf{p}) \\ &= \begin{cases} \delta_u(\mathcal{D}(\mathbf{X})) & \mathcal{D}(\mathbf{X}) > 0 \\ 1 - \delta_u(\mathcal{D}(\mathbf{X})) & \mathcal{D}(\mathbf{X}) < 0 \\ 0.5 & otherwise \end{cases} \end{aligned} \tag{12}$$

$$L^{out}(V) = 1 - L^{in}(V) \tag{13}$$

$$\mathcal{D}(\mathbf{X}) = \frac{\|\mathtt{AT}_{o,c}\mathbf{X}\|_2}{\|\mathtt{A}^{-1}\mathtt{x}\|_2} - \Omega(x) \tag{14}$$

$$x = \pi(\mathtt{AT}_{o,c}\mathbf{X}) \tag{15}$$

$$\delta_u(z) = \frac{2e^{z/\sigma}}{(e^{z/\sigma}+1)^2} + 0.5 \tag{16}$$

where $\pi()$ is the dehomogenisation function. Here $\mathcal{D}(\mathbf{X})$ is the signed distance of the voxel to the observed surface measured along the projection ray. Note also that we assume that the volume of the voxels on the object surface is negligible, so the case $V{=}on$ is not considered. δ_u is plotted in Fig. 3(right): we ascribe no confidence to voxels that are distant to the measured surface ($L^{in}(V) = 0.5$), a low probability of being inside if the voxel is immediately in front of the surface, and a high probability of being inside the object if the voxel is immediately behind the surface. Note that this form of likelihood does not increase $L^{out}(V)$ if the surface is visible behind the voxel, and so misses the opportunity to take advantage of the extra information, but this has not been detrimental in our experiments.

Evaluation of this likelihood for all voxels yields a maximum likelihood estimate of the volume, which can be easily accumulated over time via the recurrence relation

$$P(\hat{\Omega}_{0\ldots t}|\mathbf{X}_i, V_i{=}in, \mathbf{p}_{0\ldots t}) = P_t^{in}(i) = L_t^{in}(i)P_{t-1}^{in}(i) \tag{17}$$

Second, we consider the term $P(\mathbf{\Phi}|V{=}k)$:

$$P(\mathbf{\Phi}(\mathbf{X})|V{=}in) = 1 - H_\epsilon(\mathbf{\Phi}(\mathbf{X})) \tag{18}$$

$$P(\mathbf{\Phi}(\mathbf{X})|V{=}out) = H_\epsilon(\mathbf{\Phi}(\mathbf{X})) \tag{19}$$

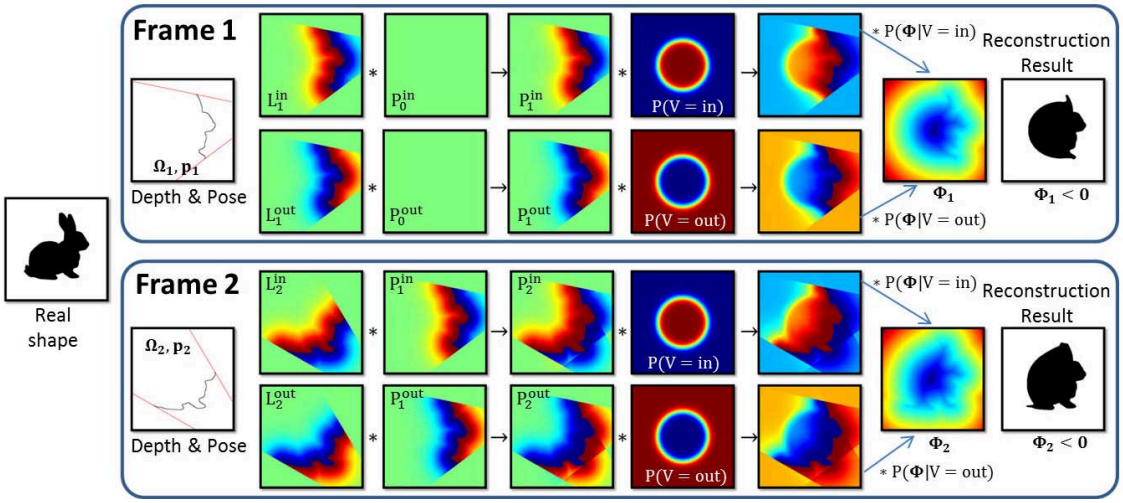

Figure 2. Work flow of our reconstruction framework.

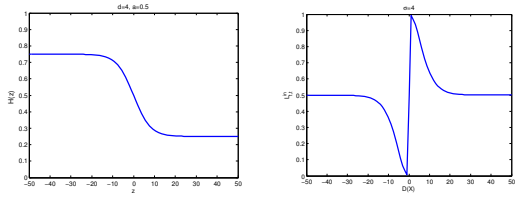

Figure 3. Plot of $\mathcal{H}(z)$ (left) and L^{in} (right)

Finally we consider the prior $P(V)$. This plays an important role in stabilising the estimation because any given RGB-D image only yields information about voxels between the camera and the object, and nothing behind the surface of the object. The prior is defined with respect to an arbitrary shape $\mathbf{\Psi}$, a SDF in which the zero level implicitly defines the shape. The prior in our algorithm $P(V)$ is then:

$$P(V = in) = P(\{\mathbf{X}_i, V_i = in\}) = \mathcal{H}(\mathbf{\Psi}) \quad (20)$$

$$P(V = out) = P(\{\mathbf{X}_i, V_i = out\}) = 1 - \mathcal{H}(\mathbf{\Psi}) \quad (21)$$

$$\mathcal{H} = 0.5 + a\left(\frac{1}{e^{z/\sigma}+1} - 0.5\right), \quad z = \mathbf{\Psi}(\mathbf{X}_i) \quad (22)$$

The influence of this prior is controlled by $a \in (0, 1)$ and its smoothness by d. With $a = 0$ there is no prior, while $a = 1$ yields a prior that is very confident away from the boundary. The form of $\mathcal{H}$ is shown in Fig. 3(left) with $d = 4, a = 0.5$. We use the same values for $\mathcal{H}(z)$ in all our experiments. Fig. 2 illustrates the formulation of our reconstruction framework in 2D. Note that observing a 3D surface is equivalent to observing part of the object contour along the projection ray.

Substituting Eqn. (17) to (22) into (11) yields (dropping the per-voxel indicator $\mathbf{X}$ here):

$$P(\mathbf{\Phi}|\Omega_{0\ldots t}, \mathbf{p}_{0\ldots t}) = \mathcal{H}(\mathbf{\Psi})(1 - H_\epsilon(\mathbf{\Phi}))P_t^{in} + (1 - \mathcal{H}(\mathbf{\Psi}))H_\epsilon(\mathbf{\Phi})P_t^{out} \quad (23)$$

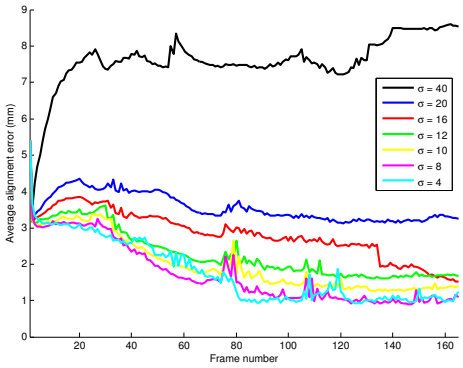

Figure 8. Quantitative evaluation of the accuracy of our method for 3D reconstruction with different σ.

We optimise $\mathbf{\Phi}$ using gradient flow methods as is standard for level sets. To preserve the SDF property of $\mathbf{\Phi}$, we additionally add a regularisation "prior" that encourages the gradient of the level set to have magnitude one [6].

5. Experimental evaluation

We have implemented the algorithm in C++ and Matlab and performed a variety of evaluations, both qualitative and quantitative. The tracking module has been ported for GPU and this yields real-time performance (20ms/frame on an NVIDIA GTX480 graphic card). Combined "online" tracking and reconstruction is currently implemented in Matlab, though its massively parallel nature means it too is amendable to GPU implementation. Qualitative examples of tracking in real-time and under significant occlusion from the hand holding a reconstructed object are given in supplementary materials , along with videos showing online tracking and reconstruction.

We begin with several real-world tracking-reconstruction sequences, which show that our method

Figure 4. Film strip showing our algorithm tracking and reconstructing a sponge. For each frame, left shows the color image while right shows the reconstruction result overlayed with the color image reprojected onto the depth image. Black indicates missing depth data.

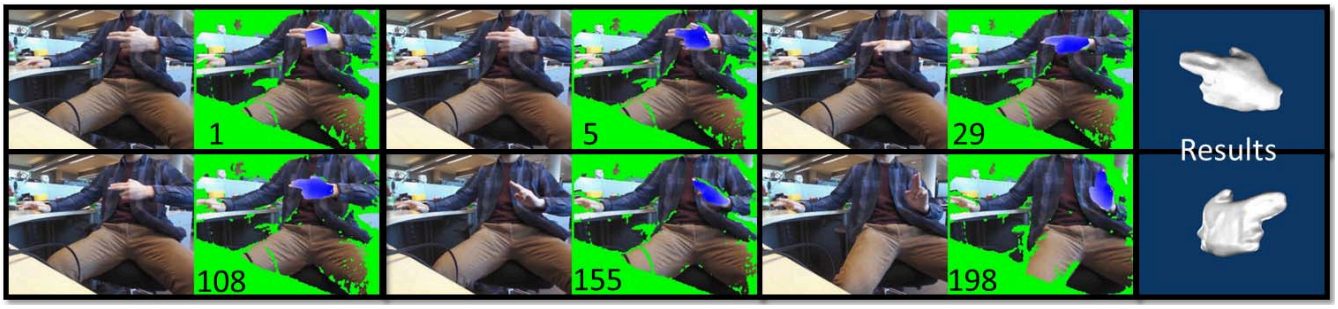

Figure 5. Film strip showing our algorithm tracking and reconstructing a hand with fixed articulation pose. For each frame, left shows the color image while right shows the reconstruction result overlayed with the color image reprojected onto the depth image. Green indicates missing depth data.

is robust to initialization and outliers and can work in unconstrained environments. Next we use generated ground truth data to evaluate the accuracy of both tracking and reconstruction. Finally, we compare both tracking and reconstruction results with KinectFusion [7], arguably the current state-of-the-art.

Figures 4, 5 and 6 show examples of our method simultaneously tracking and reconstructing different objects: a piece of sponge, a hand with fixed articulation pose and a shoe (see also supplementary videos). The sponge reconstruction is initialized using a sphere while the other two using a cube. The last column of each sequence shows the reconstruction result. All three objects are successfully reconstructed within a few hundred frames. Note that the sequences are filmed in an uncalibrated environment and the objects are small and moving and could not be reconstructed by KinectFusion. We used pixels that have $P_f > P_b$ for reconstruction, but, since in Fig. 5 and Fig. 6 the object is adjacent to close outliers (the sleeve in Fig. 5 and the hand in Fig. 6), we only use pixels from the foreground region which are at least 2 pixels away from any background pixel. This makes reconstruction results slightly smaller than the real object, by a fixed margin.

Next we evaluate our tracking and reconstruction performance using ground truth data. We use synthetic RGB-D sequences for this evaluation because of the difficulty of acquiring real video with accurate ground truth. We move a virtual RGB-D camera around an object of volume $\sim 100\times100\times100\text{mm}^3$ and generate RGB-D frames. The surface of the object has been fully observed across the frames. We present two evaluations: first, we consider a perfectly known model and perform tracking only, measuring the pose accuracy. Second, we initialize with a spherical model of radius 50mm and run tracking and reconstruction on each frame. After each frame we align the reconstructed 3D shape with the ground truth shape using ICP. We measure the reconstruction accuracy as the average Euclidean distance between all corresponding point pairs in the aligned 3D models, and pose accuracy as the difference between the aligned pose and the ground truth.

As shown in Fig. 7 our tracking accuracy in these "ideal" conditions with a known object is $< 1^\circ$ in rotation and $<$ 2mm in translation. In the case of previously unknown object with reconstructionand tracking, we can still recover reasonably accurate poses ($< 3^\circ$ in rotation and $<$ 8mm in translation), while reconstructing the 3D shape simultaneously. Examples showing tracking with reconstructed data, even under significant occlusion by the hand holding the object, are given in supplementary videos.

The constant parameter σ in Eqn. 22 controls the thickness of the reconstructed model, relative to the volume quantisation. We use a volume of $200\times200\times200$ for all experiments, with large objects scaled and reconstructed in the same fixed sized volume. To show the sensitivity of reconstruction accuracy to σ we vary it from 4 to 40 (see Fig. 8). The initial average alignment error is 6mm. For values

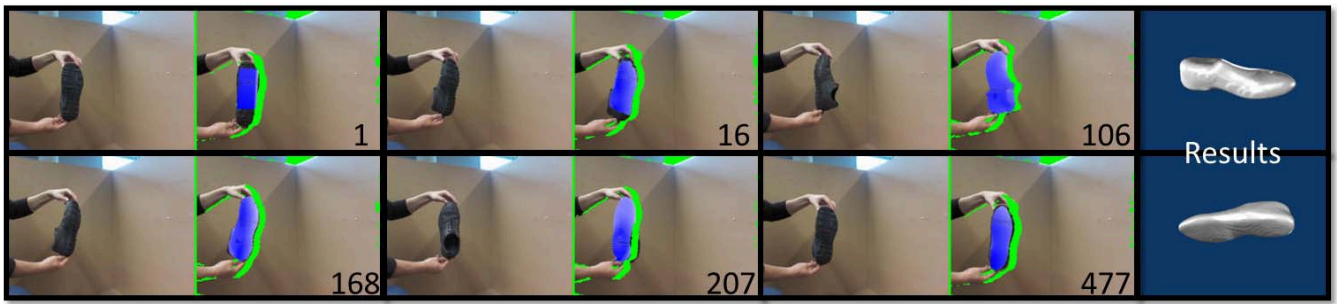

Figure 6. Film strip showing our algorithm tracking and reconstructing a shoe. For each frame, left shows the color image while right shows the reconstruction result overlayed with the color image reprojected onto the depth image. Green indicates missing depth data.

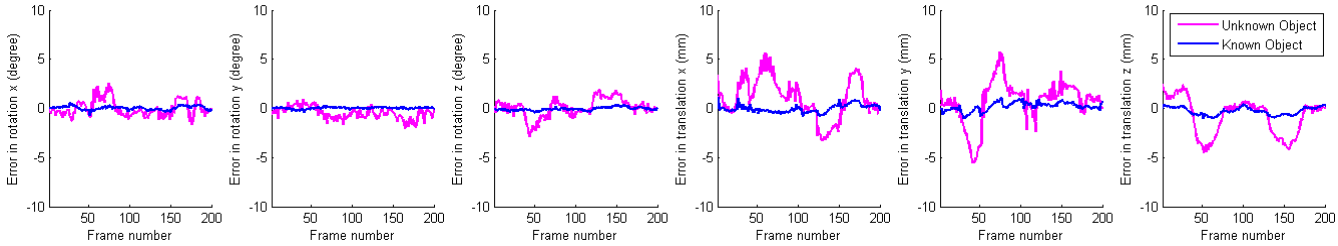

Figure 7. Quantitative evaluation of the precision our method for tracking 3D rigid object on synthetic data. The error in translation is measured in mm while rotation is measured in degree.

of $\sigma < 20$ the error decreases as more frames are observed and quickly converges at around frame 150. When $\sigma = 20$, even though the average alignment error does not converge to $<$ 2mm, the reconstructed shape is still visually correct, but it is larger than the real object. When $\sigma = 40$, the reconstructed shapes become incorrect (i.e. too thick) after the first few frames, resulting in tracking failure in all following frames, and the shape is not correctly reconstructed. We used $\sigma = 8$ for all our other tests.

In the last experiment, we compare both our tracking and the reconstruction with KinectFusion[7]. Since KinectFusion requires a static scene to fulfil reconstruction, we place the object (a piece of sponge) in the centre of a random scene and use Kinect SDK to record a sequence, in which we move the Kinect around the object to obtain most views of the object. Some sample frames are shown in Fig. 9. The first row shows the color frame, the second row visualize our reconstruction result on re-projected color image (aligned with depth frame). The third shows the 'Fusion frame' from Kinect SDK, which is the KinectFusion reconstruction result up to current frame. The last two columns show the reconstruction results of our method and KinectFusion. Both methods produce a visually correct result. The 3D model produced by KinectFusion has an incorrect lip on the top surface, while that part is correctly reconstructed by our algorithm. Our method does however produce a more noisy surface below the reconstructed 3D shape, in the areas that have never been observed by the camera. This is because noisy outlier depth pixels can propagate incorrect membership probabilities to areas in the 3D volume where the related views of object has never be observed.

In Fig. 10, we compare the camera poses produced by KinectFusion and our method. The KinectFusion camera pose is directly obtained from the Kinect SDK, using a $384 \times 384 \times 384$ volume and 384 voxels/meter. There is a fixed Euclidean transform between the two sets of poses, so we align the two camera poses using the trajectories of two camera centres. As shown in Fig. 10 our tracking result is very close to the output of KinectFusion despite relying only on the local geometry of the reconstructed object.

6. Conclusion and future work

In this paper, we have introduced a novel probabilistic framework for simultaneous tracking and reconstruction using RGB-D data. Our method is able to track and reconstruct a small moving object in unconstrained environment, and is robust to occlusion and missing data in RGB-D frames. The reconstruction module in our method evolves a 3D level-set embedding function on a per-voxel inside/outside posterior volume, which is learned incrementally online. The probabilistic formulation of our reconstruction allows us to introduce a shape prior into the 3D shape evolution, and this allows us to initialize the whole tracking-reconstruction framework with very simple 3D models, while improving the 3D model over time. The shape optimization comprises independent per-voxel operations and can be implemented in a massively parallel fashion, leading to great computational efficiency.

In future work we plan to extend the reconstruction module to take greater advantage of the color information. Cur-

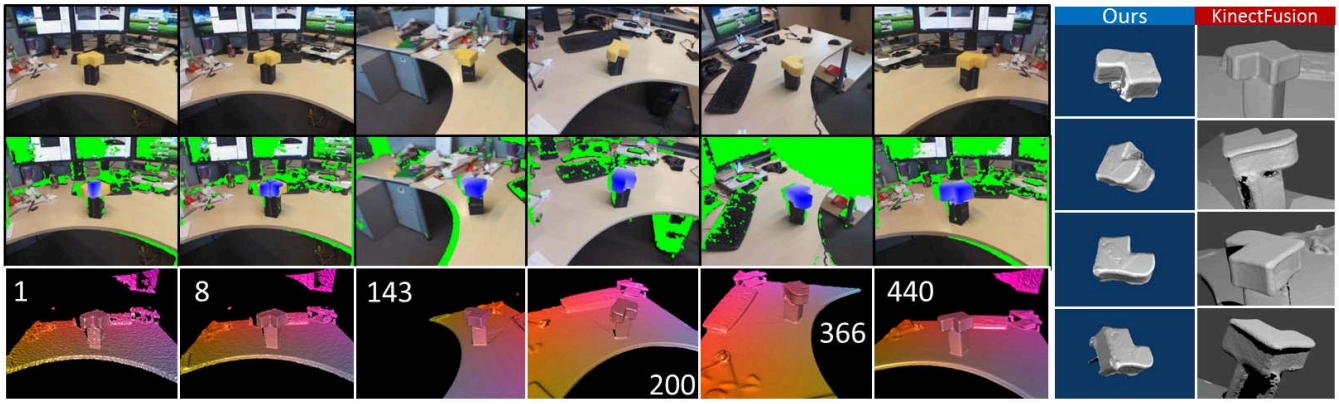

Figure 9. Film strip showing a comparison between our method and KinectFusion for 3D reconstruction. Each column shows a frame, the first row shows the color frame, the second row visualized our reconstruction result on re-projected color image (aligned with depth image), the third row shows the KinectFusion fusion frame. Last two columns shows the final reconstruction result of both methods.

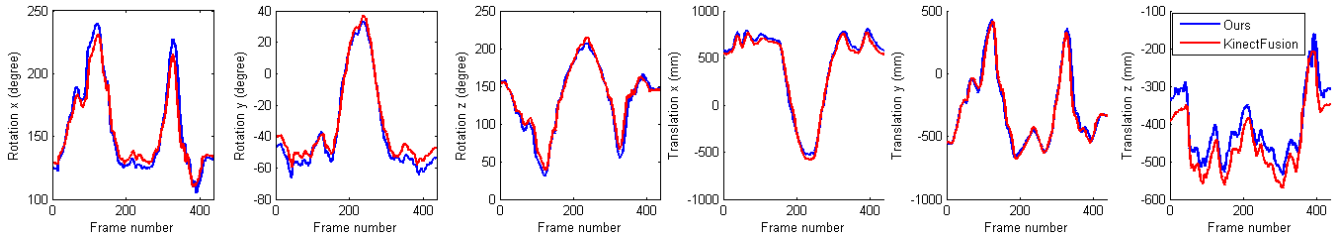

Figure 10. Quantitative comparison between camera pose output when using KinectFusion[7] and our method. Translation is measured in mm while rotation is measured in degree.

rently, we are using the color information to select foreground pixels for reconstruction, however, the background pixels can be used to improve the reconstruction as well.

Acknowledgments. This work is funded by the European Commission under the 7th Framework Programme from project 'REWIRE' (Grant No. 287713) and the Australian Research Council (Laureate Fellowship FL130100102 to IDR).

References

[1] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, 2005. 2

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

[3] C. Bibby and I. D. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *ECCV (2)*, pages 831–844, 2008. 3

[4] R. Held, A. Gupta, B. Curless, and M. Agrawala. 3d puppetry: a kinect-based interface for 3d animation. In *UIST*, pages 423–434, 2012. 1

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*, pages 559–568, 2011. 1

[6] C. Li, C. Xu, C. Gui, and M. D. Fox. Level set evolution without re-initialization: A new variational formulation. In *CVPR (1)*, pages 430–436, 2005. 5

[7] 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 *ISMAR*, pages 127–136, 2011. 1, 2, 6, 7, 8

[8] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, pages 1–11, 2011. 1

[9] V. A. Prisacariu, A. V. Segal, and I. Reid. Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *ACCV (1)*, pages 593–606, 2012. 2

[10] C. Y. Ren and I. Reid. A unified energy minimization framework for model fitting in depth. In *ECCV Workshops (2)*, pages 72–82, 2012. 2

[11] R. Ueda. Tracking 3d objects with point cloud library, 2012. `pointclouds.org`. 1, 2

[12] A. Weiss, D. A. Hirshberg, and M. J. Black. Home 3d body scans from noisy image and range data. In *ICCV*, pages 1951–1958, 2011. 2