Unsupervised Dense Object Discovery, Detection, Tracking and Reconstruction

Lu Ma and Gabe Sibley

Autonomous Robotics and Perception Group
The George Washington University
Washington DC, USA
{luma,gsibley}@gwu.edu

Abstract. In this paper, we present an unsupervised framework for discovering, detecting, tracking, and reconstructing dense objects from a video sequence. The system simultaneously localizes a moving camera, and discovers a set of shape and appearance models for multiple objects, including the scene background. Each object model is represented by both a 2D and 3D level-set. This representation is used to improve detection, 2D-tracking, 3D-registration and importantly subsequent updates to the level-set itself. This single framework performs dense simultaneous localization and mapping as well as unsupervised object discovery. At each iteration portions of the scene that fail to track, such as bulk outliers on moving rigid bodies, are used to either seed models for new objects or to update models of known objects. For the latter, once an object is successfully tracked in 2D with aid from a 2D level-set segmentation, the level-set is updated and then used to aid registration and evolution of a 3D level-set that captures shape information. For a known object either learned by our system or introduced from a third-party library, our framework can detect similar appearances and geometries in the scene. The system is tested using single and multiple object data sets. Results demonstrate an improved method for discovering and reconstructing 2D and 3D object models, which aid tracking even under significant occlusion or rapid motion.

Keywords: Structure From Motion, SLAM, 3D Tracking, 3D Reconstruction, Dense Reconstruction, Learning, Level-Set Evolution.

1 Introduction

Object detection, tracking and 3D reconstruction are fundamental early-vision tasks that are often addressed independently. Here, we present a unified and unsupervised framework that places dense-object tracking and reconstruction in feedback with one another, allowing us to discover, detect, track and reconstruct multiple objects simultaneously. Rigid bodies are automatically and recursively detected and tracked based on appearance, geometry and motion information. A 2D level-set evolution is used to update the object's 2D contour. This contour and appearance model is useful for detection, and for image based 3D object tracking

D. Fleet et al. (Eds.): ECCV 2014, Part II, LNCS 8690, pp. 80–95, 2014.

estimation when no 3D depth data is available. Pixels inside a 2D contour are used to register and update an object's 3D model – this in itself is an object-oriented dense simultaneous localization and mapping (SLAM) system where the object models are fused and improved over time. However, the proposed system is also able to discover new objects based on coherent portions of the scene that that fail to match with any known object. Rigid bodies with distinct motion are hence natural candidates for new object creation.

In recent years, sparse structure from motion has been used for model inference from video [5]. Dense techniques have also been demonstrated[18], as well as level-set based fusion techniques[8], though not for multi-object simultaneous dense tracking and reconstruction in real-time.

KinectFusion, is a wonderful example of dense data fusion using signed distance functions (SDF) [14,10]. Using a single RGB-D camera, this system creates high-quality 3D models from live scenes. Similar results are also possible using monocular cameras alone [15,20]. The system presented in this paper extends 3D level-set fusion to model both the scene and also distinct objects. Further, we find that combining 3D level-set shape models with 2D level-set appearance models leads to a novel method for unsupervised object discovery tightly coupled with tracking, detection and reconstruction.

Normally, dense fusion techniques use the whole scene, and do not isolate objects for tracking or reconstruction purposes. There are notable exceptions which track 3D objects within a full-SLAM framework but do not automatically discover novel objects [16,19]. There have also been recent advances in fast detection and tracking, such as the Pixel-wise Posterior tracker (PWP)[3,4], which is a probabilistic 2D multiple-object tracking and segmentation method based on level-sets. We extend PWP to extract objects from the scene for 3D reconstruction purposes, and also more accurately represent changing contours in 3D [7].

Shape and semantic priors have recently been used to improve dense static object tracking and reconstruction [9,2]. Recently, PWP has also been extended to handle real-time 3D object tracking [16]. This system can track a moving object in a video sequence robustly and precisely even under rapid motion. There are also recent SLAM frameworks that detect and track multiple objects in real-time [19]. However, both [16] and [19] require known a priori 3D object models. Perhaps the most similar work to ours is [17]. They present a framework that simultaneously tracks a single object while reconstructing a 3D model using RGB-D data. There are, however, key methodological differences. First, while [17] selects the foreground (target) manually, we do this automatically by segmenting unexplained portions of the image using motion outliers. Second, our system can track and reconstruct multiple objects simultaneously. Third, they initialize an object's 3D model with a shape primitive model and update the model at each frame while we initialize and update the object's 3D model with its own geometry data. Finally, the presented framework can detect many objects simultaneously that are similar to other given objects (either learned by the system online or provided by a third-party library) in the scene.

2 Overview

The system aims to track and reconstruct $n+1$ objects (rigid bodies), $O = O_\emptyset, O_1, O_2, \cdots, O_n$, which are present in a given video sequence. For an object O_i, its corresponding camera pose in the reference frame (time $k-1$) and live frame (time k) are T^i_{wr} and T^i_{wl}, respectively. For a camera c with known calibration intrinsic matrix K, the transformation between its pose and world frame is $T_{wc} \in SE(3)$. A 3D point can be described as $x_c = (x, y, z)^\top$ and dehomogenisation by $\pi(x_c) = (x/z, y/z)^\top$. A pixel $\mathbf{u} = (u, v)^\top$ in the image domain can be back-projected to a 3D point as $\chi = \frac{1}{d} \cdot K^{-1} \cdot (\mathbf{u}; 1)$ with depth value d.

Figure 1 shows the representation of the 2D and 3D model of an object. An object's 2D model refers to its shape (contour) and appearance (RGB histogram), where the 2D shape is implicitly represented by the zero level-set of the object's 2D SDF Φ (Section 4.1). A contour C segments the image domain Ω into foreground Ω_f and background Ω_b, with appearance models M_f and M_b respectively. Specifically, we use Ω^Φ_f, Ω^Φ_b to denote the foreground and background region segments by the zero level-set of Φ.

An object's 3D model refers to its 3D shape (geometry), which is implicitly represented by the zero level-set surface of its 3D SDF $\mathcal{S}$ (more specifically, we use truncated signed distance function, TSDF). We also use $\mathcal{S}_i$ to denote the 3D SDF of an object O_i, where $\mathcal{S}_\emptyset$ refers to the 3D SDF of the dominant object. $\mathcal{S}$ is stored as a n^3 volume cube, where n is the number of voxels in one dimension. The size of each voxel is $v_m = \frac{2r}{n}$, where r is the radius of the volume. The volume is initialized and updated by SDF Fusion (Section 4.2). By setting a camera in different poses, we can ray cast the implicit surface of $\mathcal{S}$ and generate virtual images (Section 4.2).

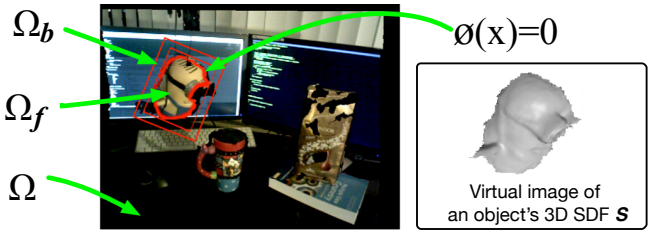


Fig. 1. Representation of an object's 2D and 3D model

Our system goes through the following stages (Figure 2):

Initialization. The system is initialized by creating a 3D model of the first frame (the scene). We call this initial 3D model the *dominant object,* $O_\emptyset$ (in our approach, the scene itself is considered an object). $\mathcal{S}_\emptyset$ is represented by a 512^3 resolution volume, where the radius of the volume is based on the scene size.

Discovery. The system tracks and reconstructs $O_\emptyset$, frame by frame, and discovers (Section 3) a contour set $C_D = C_{D1}, C_{D2}, \cdots, C_{Dp}$ of potential rigid bodies, by portions of the scene that fail to track with $O_\emptyset$.

Detection. Given object O_i with known 2D and 3D models, the system also detects (Section 3) objects similar to O_i in the scene and extracts contours $C_L = C_{L1}, C_{L2}, \cdots, C_{Lq}$. The final contour set is $C = C_D \cup C_L$.

Tracking. Given $C_j \in C$ with foreground domain Ω_f^j, if C_j fails to match with any known objects, the system will initialize a new object with C_j. If C_j matches a known object O_i, the 2D SDF $\Phi(x_i)$ of O_i is updated by Ω_f^j to $\Phi^{'}(x_i)$ via LSE producing $\Omega_f^{\Phi'}$. Tracking is achieved by estimating the relative camera pose between Ω_f^{Φ} and $\Omega_f^{\Phi'}$.

Reconstruction. The 3D SDF of O_i is then updated by fusing every pixel from $\Omega_f^{\Phi'}$ (instead of Ω_f^j) into the $\mathcal{S}_i$. This is done to reduce the noise sometimes associated with Ω_f^j.

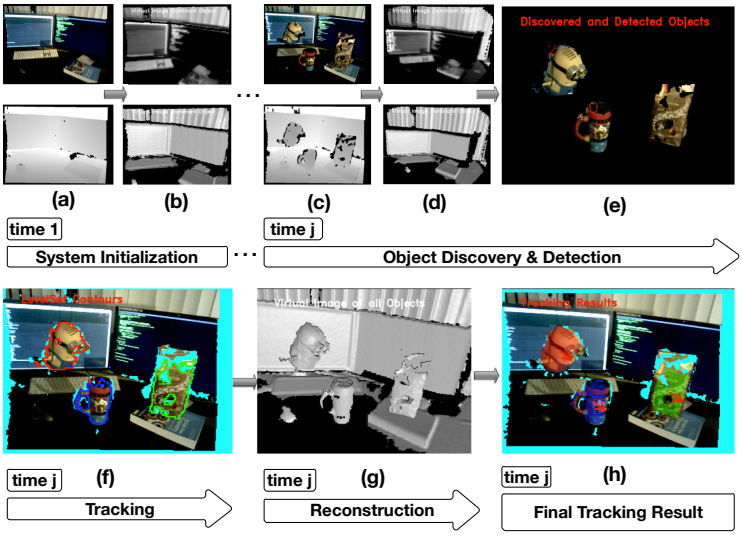


Fig. 2. System flow chart. Cyan represents missing depth data. (a) Initial RGB-D images. (b) Virtual images of $O_\emptyset$ (full scene) initialized by the first input frame. (c) Input RGB-D images at time j. (d) Virtual images of $\mathcal{S}_\emptyset$ at time j. (e) Discovered and detected (learned) objects at time j, with ID on their top left side. (f) Objects 2D SDF are updated and tracked (shown as red/green/blue contour). (g) Reconstruction results of objects after updating their 3D SDF by 2D SDF from (f). (h) Final tracking results are implicitly represented by the virtual images of the 3D SDF of all objects (shown as red/green/blue transparent masks, arrows show the objects estimated velocities).

3 Discovery and Detection

Discovery. The system offers an unsupervised object discovery method by extracting potential objects from portions of the scene that fail to track with $O_\emptyset$. Once $O_\emptyset$ is tracked by estimating the relative pose $T_{rl}^{\emptyset}$ of its camera between the

reference frame I_r and the live frame I_l via the ICP+RGB-D approach. (Section 4.1), we get $T_{wl}^{d} = T_{wr}^{\emptyset} \oplus T_{rl}^{\emptyset}$ where $\oplus$ indicate the composition of relative poses. Now, we can generate a virtual image $I_l^{'}$ of $\mathcal{S}_\emptyset$ from $T_{wl}^{\emptyset}$:

$$I_l^{'} = \Upsilon(S, T_{wl}^{\emptyset}) \tag{1}$$

here, $\Upsilon(\cdot)$ is a ray casting operator (Section 4.2). Equation (1) allows us to produce an outline image $I_o = I_l^{'} - I_l$. By searching for disjoint contours in I_o that have a minimum number of pixels d_1 inside of the contour boundary, a contour set C_D of potential objects can be obtained:

$$C_D = \Gamma(I_l^{'} - I_l) = \Gamma(\Upsilon(S, T_{wl}^{\emptyset}) - I_l), C_D = C_{D1}, C_{D2}, \cdots, C_{Dp} \tag{2}$$

where $\Gamma(\cdot)$ is a contour detection operator that detects disjoint contours in image domain [21]. For each $C_{Dj} \in C$, it defines foreground and background regions Ω_f^j, Ω_b^j . Here, Ω_f^j is either used to initialize a new object, or used to update a known object. We then compute $I_c = \Omega_f^1 + \Omega_f^2 + \cdots + \Omega_f^n$ and $I_e = I_o - I_c$, where I_e consists of two kinds of pixels: *(1)*, Pixels that should belong to I_c but were missed by a system error. *(2)*, Pixels of $O_\emptyset$ which are occluded in the reference frame. By comparing the depth value of each pixel in I_e, pixels that are very close (e.g. 1 mm) to $O_\emptyset$ will be compiled into $I_\emptyset$ and used to update $\mathcal{S}_\emptyset$. Pixels that are very close to Ω_f^j will be merged into Ω_f^j and refine its corresponding contour C_{Dj}.

Detection. For an object O_i which is either predefined in an object database or learned by our system, the system automatically searches for other instances in the live image I_l. We start with an image $I_l' = h(I_l, O_i)$, where $h(\cdot)$ extracts pixels in I_l that agree with the appearance model of object O_i. We then extract contours from I_l':

$$C_L = \Gamma(I_l' - I_l) = \Gamma(h(I_l, O_i) - I_l), C_L = C_{L1}, C_{L2}, \cdots, C_{Lq} \tag{3}$$

For each $C_{Lj} \in C_L$, we try to match Ω_f^{Lj} with the virtual image of $\mathcal{S}_i$ via the ICP+RGB-D approach. We then delete contours from C_L which correspondence Ω_f^{Lj} fail to match O_i.

Discovery and Detection Integration. Once our system finishes discovering and detecting contours in the live frame, the final contours set C based on equation (1) and (3) is:

$$C = C_D \cup C_L = \Gamma(\Upsilon(S, T_{wl}^{\emptyset}) - I_l) \cup \Gamma(h(I_l, O_i) - I_l), C = C_1, C_2, \cdots, C_n \tag{4}$$

4 Tracking and Reconstruction

4.1 Tracking

Objects and Contours Matching. A key problem of our framework is, given a set of contours $C = C_1, C_2, \cdots, C_n$ in the live frame, where C_j derives a

foreground region Ω_f^j, how to match Ω_f^j with a known object $O_i \in O$ in the reference frame? To address this question, we designed a simple score system to find a known object $O_i \in O$ that is best matched with Ω_f^j. The total score includes three different criteria: appearance (RGB Histogram), geometry (3D shape) and motion. The match score between O_i and Ω_f^j is represented by:

$$S(O_i, \Omega_f^j) = S_{Appearance_{i,j}} + S_{Motion_{i,j}} + S_{Geometry_{i,j}} \tag{5}$$

For the appearance score, $S_{Appearance_{i,j}} = r_a$ where r_a is the rate of pixels in Ω_f^j that agree with the appearance model of O_i. For the motion score, $S_{Motion_{i,j}} = p(P|P')$. Where $p(P|P') \sim \mathcal{N}(P', \Sigma)$ is the probability of $P^{'}$ appearing in P under Gaussian distribution with uncertainty Σ. Here, $P = (x, y, z)$ is the geometric center of Ω_f^j, $P' = (x', y', z')$ is the prediction of the geometric center of O_i in the live frame, which is estimated via a constant velocity motion model. For the geometry score, we match the depth image I_l of Ω_f^j and the virtual depth image I_r of $\mathcal{S}_i$ via the ICP+RGB-D approach (Section 4.1). Now, $S_{Geometry_{i,j}} = \frac{r_g}{rmse}$, where r_g is the percentage of pixels in I_r used in the ICP+RGB-D approach, $rmse$ is the error of the ICP+RGB-D approach.

The system computes score $S(O_i, \Omega_f^j)$ first by appearance, then motion, then geometry and will stop computing scores and set $S(O_i, \Omega_f^j)$ to 0 if any component is less than a threshold. Given a set of foreground regions $\Omega_f^k \in \Omega_f'$, we match O_i with $\Omega_f^k \in \Omega_f'$ if:

$$S(O_i, \Omega_f^k) > S(O_i, \Omega_f^l), \Omega_f^l \in \Omega_f', m, l \neq k \tag{6}$$

Notice that Ω_f^j may match several overlapping objects. However, each overlapping object can extract corresponding image regions from Ω_f^j via level-set evolution (LSE), which will be described in next Section.

Level Set Evolution. We use the level-set embedding function $\Phi(x_i)$, namely a 2D SDF, to implicitly represent an object's 2D contour (shape). Here $x_i = x_{i1}, x_{i2} \cdots, x_{in}$ is the set of pixel locations in the coordinate frame of O_i. We modify the Pixel-Wise Posteriors (PWP) segmentation method of [3] to use a new level-set evolution method described in [11] and propose $\mathcal{L}(\cdot)$:

$$\Phi(x_i)' = \mathcal{L}(\Phi(x_i), \Omega) \tag{7}$$

here, $\mathcal{L}(\cdot)$ is the level-set-evolution operation that evolves the object's 2D SDF $\Phi(x_i)$ to $\Phi(x_i)'$ in image domain Ω based on the 2D appearance and shape model of $\Phi(x_i)$. This operation can be achieved via:

$$\frac{\partial P(\Phi, p|\Omega)}{\partial \Phi} = \theta \cdot \Psi_p + \mu \cdot \Psi_d + \lambda \cdot \Psi_e + \alpha \cdot \Psi_a \tag{8}$$

The probability of updated $\Phi(x_i)$ appearing in location p (image warp parameters) depends on four terms. Ψ_p describes the probability of a pixel in the image domain belonging to the foreground or background under weight θ.

$$\Psi_p = \frac{(P_f - P_b)}{P_f H_\epsilon(\Phi) + P_b(1 - H_\epsilon(\Phi))} \tag{9}$$

where $H_\epsilon(\cdot)$ is a Heaviside step function, P_f, P_b are defined as:

$$P_f = \frac{P(y_i \mid M_f)}{\eta_f P(y_i \mid M_f) + \eta_b P(y_i \mid M_b)}, P_b = \frac{P(y_i \mid M_b)}{\eta_f P(y_i \mid M_f) + \eta_b P(y_i \mid M_b)} \tag{10}$$

here y_i is pixel value in input image, η_f and η_b is the area of foreground and background regions respectively:

$$\eta_f = \sum_{i=1}^{N} H_\epsilon(\Phi(x_i)), \eta_b = \sum_{i=1}^{N} (1 - H_\epsilon(\Phi(x_i)), \eta = \eta_f + \eta_b \tag{11}$$

Ψ_d is the distance regularization term that maintains the signed distance property of the level set function:

$$\Psi_d = div(d_p(|\nabla\Phi|\nabla\Phi)) \tag{12}$$

where $div(\cdot)$ is the divergence operator, $d_p(x) = \frac{p'(x)}{x}$.

Ψ_e is the edge term, which is minimized when the zero level set contour of $\Phi(x_i)$ reaches the object contour.

$$\Psi_e = \delta_\varepsilon(\Phi) div(g \frac{\nabla_\Phi}{|\nabla_\Phi|}) \tag{13}$$

here, $\delta_\epsilon(\cdot)$ is the derivative of the Heaviside step function. $g = \frac{1}{1+|\nabla G_\sigma \star I|^2}$ and ∇G_σ is a Gaussian kernel with standard deviation σ in image I.

Ψ_a is the area term, which speeds up the evolution of the level set. α is the weight of Ψ_a, where positive means shrink and negative means expand during the evolution of the level set function.

$$\Psi_a = g\delta_\epsilon(\Phi) \tag{14}$$

We seek $\frac{\partial log(P(\Phi,p|\Omega)}{\partial\Phi} = 0$ by using the gradient flow $\frac{\partial\Phi}{\partial t} = \frac{\partial log(P(\Phi,p|\Omega)}{\partial\Phi}$. In our implementation, we set weighting terms $\theta = 15$, $\mu = 0.015$, $\lambda = 5$, $\alpha = -4$, time step $t = 1$. level-set evolution (Equation 7) allows us update $\Phi(x_i)$ to $\Phi'(x_i)$ based on the appearance and shape of $\Phi(x_i)$, which provides a solution to obtain a more precise image domain $\Omega_f^{\Phi'}$ from noisy discovery or detection and overlapping objects for update the 3D SDF of an object (Section 4.2). Notice that LSE is not used for object's pose estimation. Instead, we use $\Phi'(x_i)$ to limit the pixels that are then used for shape and appearance based registration.

Pose Estimation. Once the 2D SDF of an object O_i is updated, object tracking is then achieved by estimating the relative camera pose T^i_{rl} between T^i_{wr} and T^i_{wl}, where $T^i_{wl} = T^i_{wr} \oplus T^i_{rl}$. Here, T^i_{rl} can be estimated by ICP based pose estimator [6] or RGB-D based (depth is only used for space warping) pose estimator [1]. However, both ICP and RGB-D approaches have drawbacks. The ICP approach highly depends on geometric information, and fails when tracking simple geometries (e.g. cups, dishes, etc). The RGB-D approach relies more heavily on intensity information, requires rich texture for good estimation. However, these shortcomings can be overcame by combining both approaches. We integrate the ICP approach with the RGB-D approach by using a weighting strategy [22]:

$$E = w_i \centerdot E_{icp}(T^i_{rl}) + (1 - w_i) \centerdot E_{rgbd}(T^i_{rl}) \tag{15}$$

here, $E_{icp}(T^i_{rl})$ and $E_{rgbd}(T^i_{rl})$ are the cost of ICP and RGB-D pose estimators, respectively. w_i is the weighting term, which is automatically set based on an object's color and geometric complexity during initialization. The pose is estimated by minimizing the $rmse$ of equation (15), which allows to have a ICP+RGB-D pose estimator $\mathcal{E}(\cdot)$:

$$T_{rl} = \mathcal{E}(I_r, I_l) \tag{16}$$

Where I_r and I_l are images in reference and live frame, T_{rl} is the relative pose we aim to estimate. This ICP+RGB-D pose estimator is also used to decide if two images match. Given images I_r and I_l for pose estimation, if the $rmse$ of ICP+RGB-D is higher than the threshold ε_1, or if the number of pixels used for estimation are fewer than the threshold ε_2, we assume that I_1 does not match with I_2. As I_r and I_l do not match in either color and geometric.

4.2 Reconstruction

SDF Fusion. Our system use SDF fusion [14] to initialize and update an object's 3D model. Given depth image I_D with world pose T_{wl}, SDF Fusion operation $\mathcal{S}(\cdot)$ fuses every valid point $\chi = (x, y, z)$ in I_D into the 3D SDF $\mathcal{S}$:

$$\mathcal{S}' = \mathcal{F}(\mathcal{S}, I_D, T_{wl}) \tag{17}$$

Given a contour $C_j \in C$, if C_j matches with a known object O_j, our framework uses the foreground domain Ω^{Φ}_f (derived by the updated 2D SDF Φ of O_j) to update $\mathcal{S}_j$. If C_j does not match with any known object, the system will initialize a new object O_k with C_j. The initialization of $\mathcal{S}_j$ depends on the 'type' of C_j. (1) If C_j belongs to object discovery result C_D, the system will initialize $\mathcal{S}_j$ with the foreground domain defined by C_j. (2) If C_j comes from object detection result C_L, which has a model similar to that of a known object O_k, the system will initialize $\mathcal{S}_j$ by using $\mathcal{S}_k$ directly.

Ray Casting. Having a 3D SDF $\mathcal{S}$, we can ray cast [14] the zero level set surface of $\mathcal{S}$ to generate virtual image I_v (can be grey or depth images. we also rendered

the 3D model with Phong shading) by setting the camera at pose T_{wc}, where the Ray Casting operation $\Upsilon(\cdot)$ is:

$$I_v = \Upsilon(\mathcal{S}, T_{wc}) \tag{18}$$

Since each object O_i has its own 3D SDF $\mathcal{S}_i$, the virtual image I_v contains the object O_i in the scene and the value of other pixels in I_v will be set to void.

2D and 3D SDF Interaction. For an object O_i, once the system updates its 3D SDF $\mathcal{S}_i$ and its corresponding camera pose T^i_{wl}, its 2D SDF $\Phi(x_i)$ can be refined by:

$$\Phi(x_i) = \nu(\Phi(x_i), \Upsilon(\mathcal{S}_i, T^i_{wr})) \tag{19}$$

$\nu(\cdot)$ sets the zero level-set of $\Phi(x_i) = 0$ to be the object contour of $\Upsilon(\mathcal{S}_i, T^i_{wr})$.

4.3 Tracking and Reconstruction Interaction

The 2D tracking result of an object O_i is implicitly represented by the matching result (Section, 4.1), which may be noisy or incomplete due to noisy input data or occlusion of its correspondence match contour $C_j \in C$. By ray casting the up-to-date 3D SDF of O_i, the system can generate a more complete and precise 2D tracking result with respect to the object's pose and shape, which can offer complete knowledge of objects even in the presence of significant occlusions.

In short, the input for updating an object's 3D SDF comes from the results of 2D tracking, while an object's 3D reconstruction results are used to inform the final 2D tracking results. This interaction between tracking and reconstruction improves tracking and reconstruction results simultaneously.

5 System Integration

Given a contour set C (Equation (4)), where $C_j \in C$ defines an image domain Ω^j that matches a known Object O_i, where O_i with camera pose T^i_{wr}, 2D SDF $\Phi(x_i)$ and 3D SDF $\mathcal{S}_i$ in the reference frame, $\Phi(x_i)$ can thus be updated to:

$$\Phi^{'}(x_i) = \mathcal{L}(\Phi(x_i), \Omega^j) \tag{20}$$

$\mathcal{L}(\cdot)$ is a level-set-evolution operator (Section 4.1). The relative pose T^i_{rl} between $\Phi_f(x_i)$ and $\Phi^{'}_f(x_i)$ can be estimated via ICP+RGB-D pose estimator $\mathcal{E}(\cdot)$:

$$T^i_{rl} = \mathcal{E}(\Phi_f(x_i), \Phi^{'}_f(x_i)) \tag{21}$$

Now $\mathcal{S}_i$ can be updated by SDF Fusion $\mathcal{F}(\cdot)$:

$$\mathcal{S}'_i = \mathcal{F}(\mathcal{S}_i, \Phi^{'}_f(x_i), T^i_{wl}) = \mathcal{F}(\mathcal{S}_i, \Phi^{'}_f(x_i), T^i_{wr} \oplus \mathcal{E}(\Phi_f(x_i), \Phi^{'}_f(x_i))) \tag{22}$$

here, The unknown parameters are the 2D SDF $\Phi(x_i)$, 3D SDF $\mathcal{S}_i$ and camera pose T^i_{wr}. Equation (22) shows how the 2D and 3D model of an object is tracked and updated with respect to the change of the camera pose, which links object discovery, detection, tracking, and reconstruction together.

6 Results and Discussion

We test the system with single and multiple-object data sets. Each data set, lasting between 300 and 550 frames, is captured in 30 FPS via a hand-held moving RGB-D camera.

To evaluate our method, we first generate ground truth 3D models of all testing objects using synthetic RGB-D sequences. We then use the pose estimation approach described in Section 4.1 to track each ground truth object model in the testing video sequences. This allows us to generate a ground truth depth image D_g for each object in each frame. We then run our system in the same sequences and generate a virtual depth image D_l from each object's learned 3D SDF. The performance of the system is evaluated at pixel level by comparing each corresponding object's pixel p in D_g and D_l. We define a good match (true positive) if $abs(D_l(p) - D_g(p)) < d_\varepsilon$. We set $d_\varepsilon = 10$mm in our experiments. We report per-frame precision and recall values for each learned object.

Figure 3 shows an examples of discovered objects and their corresponding matched object's 2D level-set after refine. It can be seen that LSE allows us to smooth noise from the raw detected contour (e.g. Fig. 3(c), 3(d)) and helps update the 2D SDF when objects overlap with each other (e.g. Fig. 3(r)). In totally, LSE is crucial as 1) it helps learn the unknown object based on its previous appearances and shape; 2) it helps track overlapping objects with significant occlusions; 3) without LSE, the reconstruction contour is often unusably noisy; 4) it allows object discovery and tracking when no depth is available. Once we update the 2D SDF, we are ready to track and reconstruct each object.

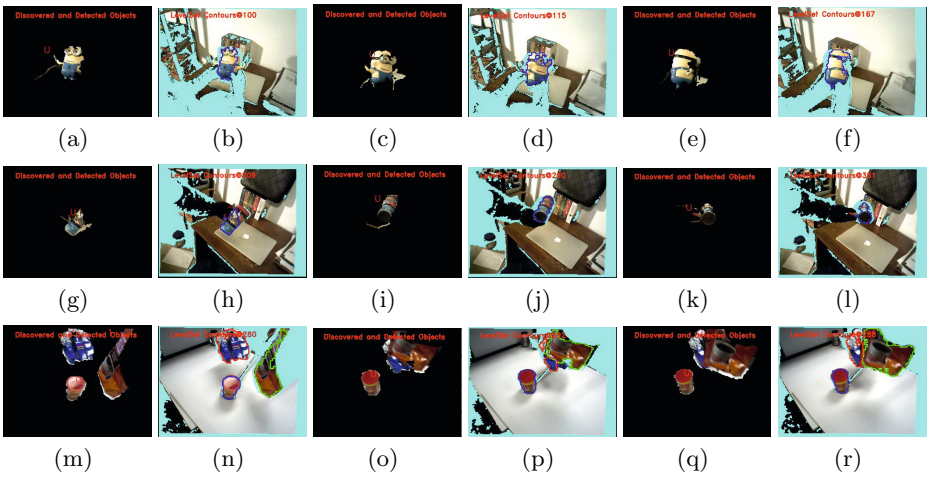
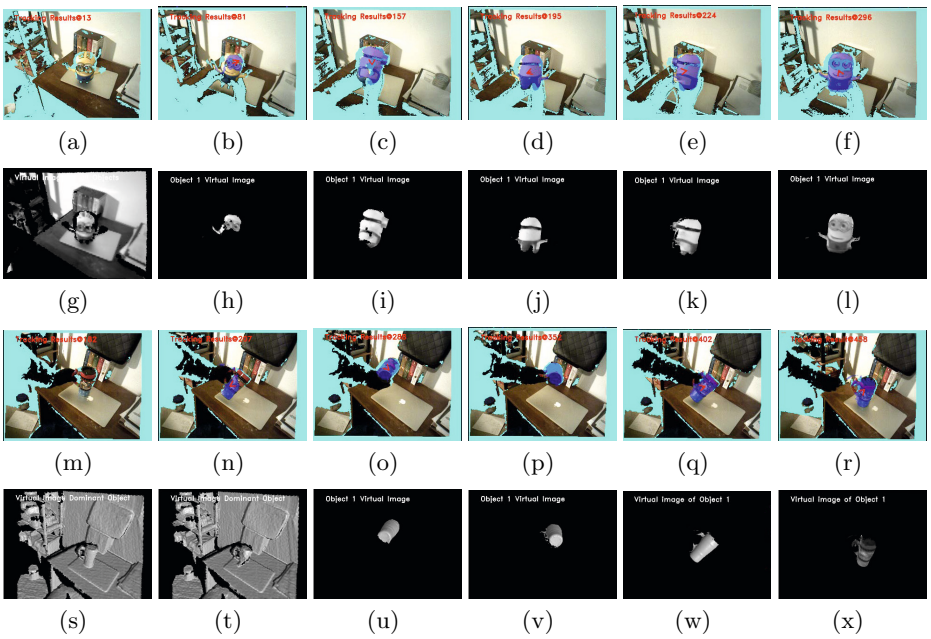


Fig. 3. Examples of raw discovered objects (odd columns, with ID on their top left side) and the zero level-sets refined by level-set-evolution (even columns), shown as red/green/blue outlines. Cyan represents missing depth data.

Figure 4 shows the tracking and reconstruction results for a cup and a yellow cartoon figure (Dave). As we move these objects, their shape and appearance is changed greatly to test the robustness of the system. The cup, which has a simple geometry, is used to test the ICP+RGB-D pose estimator. As motion is applied to the cup, it is slowly removed from the dominant object's 3D SDF $\mathcal{S}_\emptyset$ (Fig. 4(t)). This confirms that the system performs SLAM by simultaneously localization and mapping both the dominant object (desk and wall) and the singled out object (cup/Dave). It also shows that our system is able cope with limited input data, e.g. large portions of missing depth data (e.g. Fig. 4(f) 4(p)).



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

(m) (n) (o) (p) (q) (r)

(s) (t) (u) (v) (w) (x)

Fig. 4. Results of tracking (first and third rows, shown as blue transparent masks) and reconstruction (second and third rows) for single objects. Cyan represent missing depth data, red arrows show the objects estimated velocities.

Our multiple-object tracking and reconstruction tests are shown in Figure 5. For this experiment we focus on evaluating the performance of our system under rapid motions, overlapping and occlusions. Overall, we see that our system can discover, track and reconstruct multiple objects in a scene with high accuracy (Fig. 6(c)). Our system can recover the pose of an object even under severe occlusion (Fig. 5(m), the blue car is blocked by the donut bag).

Although our test sequences yield encouraging object tracking and reconstruction results, the precision and recall (Fig. 6) of our system changes with respect to an object's status. When our objects first appear (cup and Dave), the recall grows as the system continue to learn the objects' models.

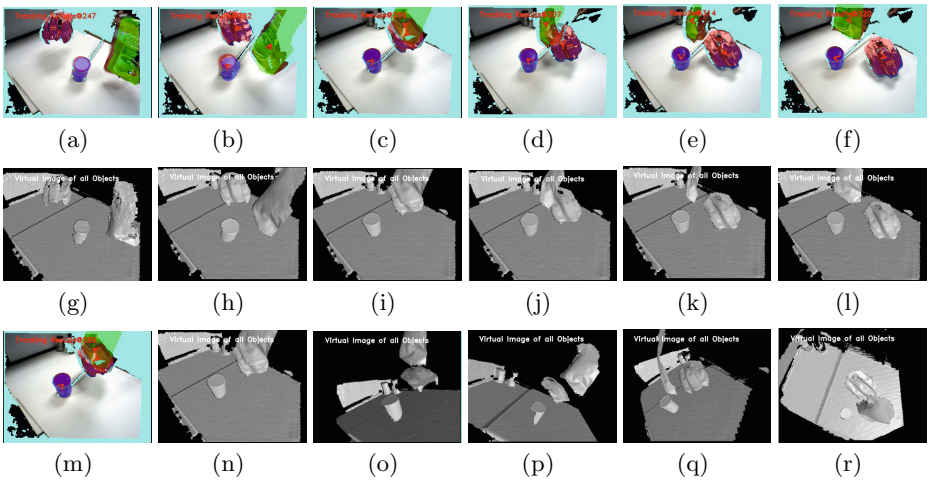


Fig. 5. Tracking (first row, shown as red/green/blue transparent masks) and reconstruction (second row) results of multiple objects. (m) Occluded car. (n)-(r) Multiple-view rendering of all objects during the occlusion shown in (m).

Objects without rapid motion − frame to frame translation in (x, y, z) less than 7 cm and rotation in $roll, pitch, yaw$ less than 12 degrees − can be tracked with high accuracy (precision over 90 % and recall over 80 %) for most frames with a true positive threshold of $d_\varepsilon = 10mm$ (cup/Dave/red can).

Drops in precision and recall that can be seen in Fig.(6) are due to rapid motions (e.g. the donut bag, frame to frame translation of more than 15cm in (x, y, z), and rotation of more than 15 degrees in $roll, pitch, yaw$) or occlusions (e.g. the blue car is 80 % occluded). In both cases, precision and recall recover to around 85% immediately after. When objects are highly occluded, (more than 90%, see blue car during frame 40 to 47 and frame 50 to 53 in Fig. 6(c) 6(d), corresponding tracking sequences in 5), both precision and recall are set to 0. While we don't yet have enough confidence to track occluded objects, our system proves adept at tracking objects that reappear after occlusion.

Over all, our system maintains good tracking and reconstruction performance (both precision and recall over 90%) when objects have less than $5cm$ frame to frame translations and 9 degrees frame to frame rotations. However, failure cases occur under significant motion blur, serious occlusions (more than 90% occluded), which predominantly hamper pose estimation.

Object Detection of Learned Objects. We demonstrate the learning and subsequent detection ability of the system with a coke can data set. Here, we grab a coke can in the left side of the scene (Fig. 7(a)). After the system has accrued enough 2D and 3D modeling information of the can (Fig. 7(b)), the system can detect other instances of the coke can, that will then be tracked and reconstructed (Fig 7(c)- 7(f)). This experiment shows that the system can discover new objects in the scene and learn similar objects without human intervention.

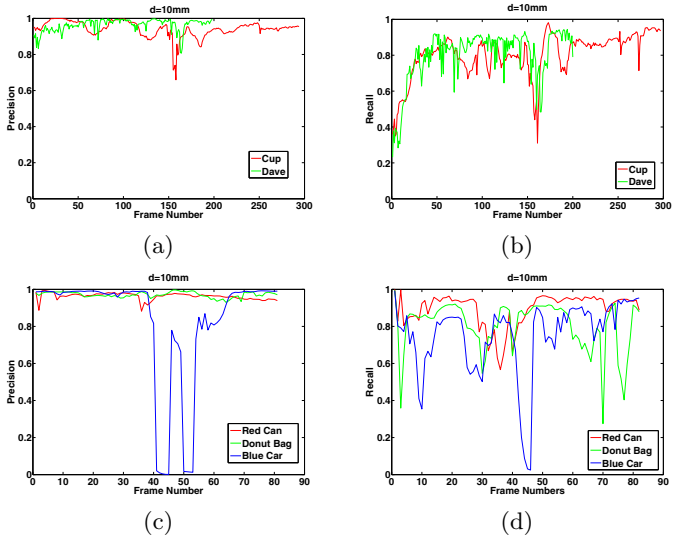


Fig. 6. Precision and Recall of system's performance under single (a), (b) and multiple-object (c),(d) data sets with true positive threshold of $d_\varepsilon = 10mm$. Frame number starts when motion is first applied to an object in test sequences.

System Run-Time. The system is implemented in C++. Both the reconstruction and the combined ICP and RGBD approach are GPU-based. We test our system with a single Nvidia TITAN GPU, Intel i7 CPU desktop, using 640×480 resolution of input images. Table 1 shows our system run-time in different stages.

Table 1. System Run-time of Different Stages, (*) Represent GPU-Based

	Discovery	Match Objects and Contours*	Single ICP+ RGB-D*	Single level-set evolution	Single Reconstruction*
Single Object	15 ms	24 ms	11 ms	31 ms	5 ms
Three Objects	15 ms	80 ms	11 ms	32 ms	5 ms

As table 1 shows, the most expensive stage comes when matching contours with objects. However, since computing the match score between each contour and a known object can be processed in parallel, we anticipate that running time can be significantly decreased. Discovery and the level-set-evolution are also amenable to GPU implementation where LSE is known to run in less than 6ms in [4]. Finally, tracking and reconstruction of each object can also be processed in parallel. With these enhancements and based on run-times reported in the literature [16,3], it is reasonable to conjecture that our system will run at 24Hz.

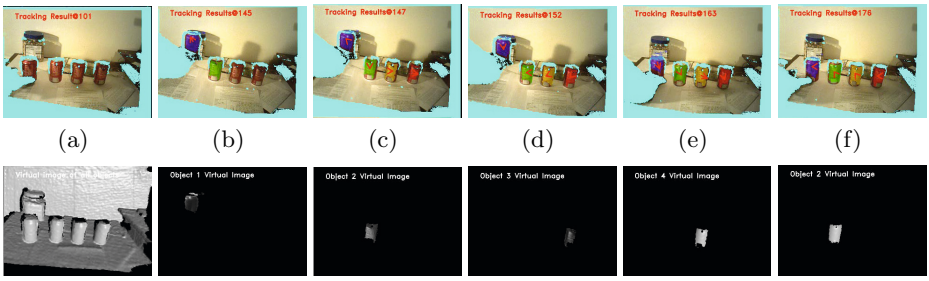


Fig. 7. Multiple instance detection. Above are shown tracking (first row, shown as different color transparent masks) and reconstruction (second row) results of detected (learned) objects (coke can) in the scene. After motion is applied to one coke can (a), all the other cans are detected.

7 Failure Cases and Future Work

Although the system is robust to many real-world operating conditions, it is not robust under significant motion blur, serious occlusions, etc., which predominantly hamper pose estimation. In future work, we aim to improve the resolution of the reconstructed models by [12,23] and enhanced the system with a single RGB camera techniques from [15,13]. We also see the potential applications in robot-object interaction and autonomous cars.

8 Conclusions

We present a novel unsupervised framework that is able to discover, detect, track and reconstruct multiple rigid bodies simultaneously. Instead of considering them independently, the presented system uses tracking results to refine reconstruction-models and reconstruction-models to aid detection and tracking. This allows objects tracking even under significant occlusion, missing data and rapid motion. Rigid bodies are automatically and recursively detected and tracked based on appearance, geometry and motion information. The system preforms dense SLAM on an object by object basis, and also discovers new objects it has not seen before. The system is tested using single and multiple object data sets. Results demonstrate an improved method for discovering and reconstructing 2D and 3D object models. To summarize, our primary contributions are: 1) a new framework, which unifies dense object discovery, detection, tracking and reconstruction; 2) an unsupervised approach to discover unknown objects; 3) a framework that not only learns objects but also detect similar objects based on learned models; and 4) present a new level-set evaluation method.

References

1. Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision 56(3), 221–255 (2004)
2. Bao, S.Y., Chandraker, M., Lin, Y., Savarese, S.: Dense object reconstruction with semantic priors. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1264–1271. IEEE (2013)
3. Bibby, C., Reid, I.D.: Robust real-time visual tracking using pixel-wise posteriors. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 831–844. Springer, Heidelberg (2008)
4. Bibby, C., Reid, I.: Real-time tracking of multiple occluding objects using level sets. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1307–1314. IEEE (2010)
5. Blais, G., Levine, M.D.: Registering multiview range data to create 3d computer objects. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(8), 820–824 (1995)
6. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and vision computing 10(3), 145–155 (1992)
7. Cremers, D., Rousson, M., Deriche, R.: A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. International Journal of Computer Vision 72(2), 195–215 (2007)
8. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 303–312. ACM (1996)
9. Dame, A., Prisacariu, V.A., Ren, C.Y., Reid, I.: Dense reconstruction using 3d object shape priors. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1288–1295. IEEE (2013)
10. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 559–568. ACM (2011)
11. Li, C., Xu, C., Gui, C., Fox, M.D.: Distance regularized level set evolution and its application to image segmentation. IEEE Transactions on Image Processin 19(12), 3243–3254 (2010)
12. Meilland, M., Comport, A.I.: Super-resolution 3d tracking and mapping. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 5717–5723. IEEE (2013)
13. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1498–1505. IEEE (2010)
14. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136. IEEE (2011)
15. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: Dtam: Dense tracking and mapping in real-time. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2320–2327. IEEE (2011)
16. Prisacariu, V.A., Reid, I.D.: Pwp3d: Real-time segmentation and tracking of 3d objects. International Journal of Computer Vision 98(3), 335–354 (2012)

17. Ren, C.Y., Prisacariu, V., Murray, D., Reid, I.: Star3d: Simultaneous tracking and reconstruction of 3d objects using rgb-d data
18. Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M.: Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009. pp. 1–6. IEEE (2009)
19. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H., Davison, A.J.: Slam++: Simultaneous localisation and mapping at the level of objects. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1352–1359. IEEE (2013)
20. Sibley, G., Keivan, N., Patron-Perez, A., Murphy, L., Lovegrove, S., Mamo, V.: Scalable perception and planning based control. In: International Symposium on Robotics Research (2013)
21. Teh, C.H., Chin, R.T.: On the detection of dominant points on digital curves. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(8), 859–872 (1989)
22. Whelan, T., Johannsson, H., Kaess, M., Leonard, J.J., McDonald, J.: Robust tracking for real-time dense rgb-d mapping with kintinuous (2012)
23. Zhou, Q.Y., Miller, S., Koltun, V.: Elastic fragments for dense scene reconstruction. Environments 27(16), 7–35