SDF-Net: Real-time Rigid Object Tracking Using a Deep Signed Distance Network*

Prayook Jatesiktat[1], Ming Jeat Foo[1], Guan Ming Lim[1], and Wei Tech Ang[1]

[1]School of Mechanical and Aerospace Engineering, Nanyang Technological University
50 Nanyang Avenue, Singapore 639798
{prayook001, foom0009, guanming001}@e.ntu.edu.sg, wtang@ntu.edu.sg

Abstract. *In this paper, a deep neural network is used to model the signed distance function (SDF) of a rigid object for real-time tracking using a single depth camera. By leveraging the generalization capability of the neural network, we could better represent the model of the object implicitly. With the training stage done off-line, our proposed methods are capable of real-time performance and running as fast as 1.29 ms per frame on one CPU core, which is suitable for applications with limited hardware capabilities. Furthermore, the memory footprint of our trained SDF-Net for an object is less than 10 kilobytes. A quantitative comparison using public dataset is being carried out and our approach is comparable with the state-of-the-arts. The methods are also tested on actual depth records to evaluate their performance in real-life scenarios.*

Keywords: deep neural network, signed distance, object representation, object tracking, depth image

1 Introduction

The tracking of a rigid object in 3D can be characterized as a problem of estimating the six degrees of freedom (6-DOF) trajectory of an object as it moves around a scene. Rigid object tracking is a useful tool in various applications such as augmented reality in which a user interacts with an object, or industrial automation in which a robot manipulates an assembly part. There are multiple approaches to achieve 3D tracking such as the use of inertial sensors or fiducial markers attached to the object. However, the readings of inertial sensors may drift with time and the marker-based approach can be intrusive. To overcome those limitations, vision-based tracking offers solutions that are non-invasive, practical, and cheap [15].

Since the introduction of commodity depth cameras, the availability of depth data extends RGB tracking methods by utilizing depth information in particle filter algorithm [5, 14], Gaussian filter algorithm [8], and also the well-established Iterative Closest Point (ICP) algorithm [3, 30]. The ICP aims to find the best pose estimate to minimize the distance between two sets of depth data and while

* Supported by Rehabilitation Research Institute of Singapore (Ref.No. RRG2/16001)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

there are several variants of ICP with more efficient and robust solutions [23], the main process of searching for point correspondences can be computationally expensive and error-prone.

Thus, to avoid the time-consuming point-to-point correspondence search, depth data could be modelled implicitly to allow point-to-model distance minimization [21]. For example, a minimal set of primitive shapes is used to model a simple industrial part [26], but for a more complex model, implicit functions such as implicit B-Spline can be employed to provide a richer data representation for a better registration [22]. The signed distance function (SDF) is another representation that implicitly encodes 3D surfaces and can be used directly to define a cost function for accurate registration [4]. As such, SDF has been applied in recent works on scene reconstruction [12, 25] and rigid object tracking [19]. The SDF of basic geometric shapes such as spheres, cubes, and ellipsoids can be represented implicitly. For example, the SDF of a sphere of radius r centred at the origin can be written using the implicit expression $\sqrt{x^2 + y^2 + z^2} - r$.

However, the SDF of more complex shapes are much more difficult to define and are thus often represented as sampled volumes [6]. To obtain a continuous representation of a complex object surface, there are a few works that incorporate machine learning techniques. For instance, Radial Basis Function (RBF) neural network can be used to classify a 3D point into three classes, namely internal, on-surface, and external [17]. To date, the use of a multi-layer neural network to model the SDF of an object has only been done for brain structures segmentation [9]. To the best of our knowledge, our work is the first to model the SDF of an object using a deep neural network for object tracking purpose.

Recently, learning-based methods have revolutionized many areas of computer vision including object tracking. Since Tan and Ilic [27] have proposed a random-forest-based method to regress the movement of the object from the change in the observed point cloud, several improvements [29, 1, 28] have been made to advance the state-of-the-arts in temporal 3D object tracking without GPU. On the other hand, Garon and Lalonde [7] claimed to develop the first end-to-end deep learning for temporal object tracking. However, it needs GPU for real-time tracking due to the large network size.

The contribution of this paper is the adoption of a learning-based method to train a deep neural network, which we term as SDF-Net, to approximate the SDF of an object. In addition, we also propose two methods to utilise the trained SDF-Net for rigid object tracking. Furthermore, a quantitative comparison on a public dataset is carried out to compare our approach against the state-of-the-arts. Our methods are also tested with real depth data from two different commodity depth cameras to demonstrate the real-time object tracking capability in different scenarios.

This paper is organized as follows. Section 2 details the methodology to train a deep neural network that models the SDF of an object and the two different ways to use the network for object tracking. Section 3 outlines the evaluation methods and Section 4 discusses the results. Finally, Section 5 presents the conclusion, as well as the limitation and future work.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

2 Method

Our goal is to estimate the current 6-DOF pose θ_t, which contains an orientation R_t and a translation $\boldsymbol{t}_t$, of the given object in the camera reference frame $\{C\}$.

A depth camera is used to provide a sequence of depth images. For each depth frame, the following inputs are used in our method:

1. The depth image of the current frame t and the previous frame $t-1$.
2. Pose estimation results from the previous frames $(\theta_{t-h}, ..., \theta_{t-1})$ where $h > 1$

Also, the method has access to the camera intrinsic parameters and the triangular mesh model of the target object in the object reference frame $\{H\}$.

2.1 Method Overview

The SDF of an object is simply a function that takes a 3D point $\boldsymbol{p}$ and returns a signed Euclidean distance to the closest point on the object surface. In this paper, the signed distance is defined to be negative when $\boldsymbol{p}$ is inside the object, and positive when $\boldsymbol{p}$ is outside the object. The intuition behind our fitting approaches is to use SDF in a form of trained neural network to guide the pose update in moving sampled points from the observation towards zero signed distance value.

Our method can be divided into two stages. The first stage is to prepare an approximation of the SDF of the object by training a neural network, while the second stage utilizes the network for pixel sampling and pose tracking. Two different methods of pose tracking are proposed. The first method is based on the conventional ICP approach with an adaptation at the correspondence search. The second method is based on an optimization approach. Both methods share the same sparse sampling mechanism which is designed to increase the robustness of tracking when there is an occlusion. Nonetheless, both methods depend heavily on the quality of learned SDF-Net.

2.2 Building a Signed Distance Network

Training Data Preparation. The 3D model of the object to be tracked is translated to make its centroid stays at the origin. To facilitate the learning process, the model is scaled with a scaling factor $s = 1/d$, where d is the maximum diameter of the object; one unit in this scaled world $\{S\}$ is equivalent to d.

Three sets of 100,000 points, namely A_I, A_S and A_O, are sampled from the region inside the object, on the surface, and outside the object respectively. The points in A_O are only sampled from the space outside object up to 1.5 units from the surface. The space is divided equally into 100 sections with a thickness of 0.015 units each. One thousand points are then randomly sampled from each section to ensure that the training data obtained in A_O are evenly distributed.

Let $K : \mathbb{R}^3 \mapsto \mathbb{R}^3$ maps a 3D position to the closest point on the object surface and $\Phi' : \mathbb{R}^3 \mapsto \mathbb{R}^3$ be the expected gradient that returns a unit direction, which points towards the closest point on the surface when the input lies inside

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

the object. In contrast, the direction will point away from the closest point on the surface if the input falls outside the object. All the associated expected gradients and closest surface mapping at all points in A_I and A_O are pre-calculated.

Network Structure. Our neural network has three input nodes for a 3D position and one output node for the signed distance. The hidden layers are fully-connected. All the hidden nodes use tanh activation function, except for the output node which uses linear activation function.

In this study, two different diamond-shaped networks are tested. Network $\mathcal{A}$ is a smaller network with 10 hidden layers, while network $\mathcal{B}$ is a bigger network with 12 hidden layers. The number of nodes in all layers are 3-6-9-12-15-18-15-12-9-6-3-1 for network $\mathcal{A}$ and 3-6-9-12-15-18-21-18-15-12-9-6-3-1 for network $\mathcal{B}$, with total tunable parameters of 1,369 and 2,162 respectively. The diamond structure allows a gradual projection of the 3D data to a higher dimensional space before being slowly reduced to one dimension. Given the same number of tunable parameters, shallower networks with equal numbers of hidden nodes for all layers seems to be less effective in learning.

Cost Function. Let Θ represent all weights and biases of a target network. Function $N_\Theta : \mathbb{R}^3 \mapsto \mathbb{R}$ does a feed-forwarding that maps a 3D position (input layer) to a single number at the output node. Function $N_\Theta{}' : \mathbb{R}^3 \mapsto \mathbb{R}^3$ maps a 3D position to the gradient of N_Θ at that spot. The cost function is defined as:

$$\begin{aligned} Cost(\Theta) = &\Big(\frac{1000}{s^2|A_S|}\sum_{\boldsymbol{p}_i \in A_S} N_\Theta(\boldsymbol{p}_i)^2\Big) \\ &+\Big(\frac{1000}{s|A_O|}\sum_{\boldsymbol{p}_i \in A_O} \max(0, -N_\Theta(\boldsymbol{p}_i))\Big) + \Big(\frac{1000}{s|A_I|}\sum_{\boldsymbol{p}_i \in A_I} \max(0, N_\Theta(\boldsymbol{p}_i))\Big) \\ &+\Big(\frac{1}{|A_I|+|A_O|}\sum_{\boldsymbol{p}_i \in A_I \cup A_O} \|N_\Theta{}'(\boldsymbol{p}_i) - \Phi'(\boldsymbol{p}_i)\| + \|N_\Theta{}'(K(\boldsymbol{p}_i)) - \Phi'(\boldsymbol{p}_i)\|\Big) \end{aligned} \tag{1}$$

The first (*SurfaceDistancePenalty*) term penalizes the network when some outputs at surface points deviate from zero. The second term penalizes the network when some outputs at outsider points are negative. The third term penalizes the network when some outputs at insider points are positive. Since the gradient is applied directly in our tracking methods, it is also considered in the training. Thus, the last (*GradientPenalty*) term is introduced to penalize the deviation of gradient from the expectation. This term also applies gradient constraints on the object surface using the same expected gradient from its correspondence.

All the weights and biases (Θ) are trained to minimize the cost function in Eq.1. The training is done on TensorFlow using ADAM optimizer [13] with a learning rate of 0.001 for 100,000 iterations.

Notation. After the training, the network is ready to be used in the scaled reference frame $\{S\}$. To make the notation more compact, the network is augmented so that it can work in the object reference frame $\{H\}$ and output the

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

signed distance that relates to the real world scale. We define our learned signed distance function $D : \mathbb{R}^3 \mapsto \mathbb{R}$ and its gradient function $\boldsymbol{G} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ as

$$D({}_H\boldsymbol{p}) = N_\Theta(s \cdot {}_H\boldsymbol{p})/s \qquad \text{and} \qquad \boldsymbol{G}({}_H\boldsymbol{p}) = N'_\Theta(s \cdot {}_H\boldsymbol{p}) \tag{2}$$

given that ${}_H\boldsymbol{p}$ is the input position in the object reference frame $\{H\}$. The function D and $\boldsymbol{G}$ are implemented in a closed-form using the standard feed-forwarding and back-propagation algorithm. This technique allows the computation to be vectorised and to run efficiently on a CPU with SIMD capability.

The state of orientation R_t and translation $\boldsymbol{t}_t$ represents the transformation of the object with respect to the camera reference frame at time t:

$${}_C\boldsymbol{p} = R_t \, {}_H\boldsymbol{p} + \boldsymbol{t}_t \tag{3}$$

2.3 Current Frame Pose Prediction

At the current frame t, when the θ_t is not yet calculated, the predicted pose $\hat{\theta}_t$ will be calculated using a short series of pose estimation results from the previous frames (θ_{t-h}, ..., θ_{t-1}). The predicted translation $\hat{\boldsymbol{t}}_t$ and the predicted orientation $\hat{R}_t$ are calculated independently. For translation, a weighted linear regression is used to extrapolate $\hat{\boldsymbol{t}}_t$ with weights $v_1, ..., v_h$. For orientation, all the past orientations are expressed in quaternions. Then, we predict the current orientation $\hat{q}_t$ with the following equation:

$$\hat{q}_t = \vartheta\Big(\sum_{i=1}^{h-1} w_i \cdot (q_{t-1} \times (\sigma_{t-2}(t-1-i, t-1))^{-1} \times q_{t-1})\Big) \tag{4}$$

given that $\sigma_c(a, b)$ is a quaternion spherical linear interpolation (Slerp) between q_a and q_b at frame c. The function $\vartheta(\cdot)$ represents quaternion normalization. The weights $v_1, ..., v_h$ and $w_1, ..., w_{h-1}$ allow the adjustment of the responsiveness of the prediction. An incremental geometric series 2^{i-1}, with $i = 1, 2, \ldots, h$, is used for both weight series.

2.4 Object Pixel Sampling

Among all the depth pixels in the current frame, a number of pixels will be sampled and used in object tracking. If a few non-object pixels are sampled to fit with the object, it could reduce the tracking accuracy and lead to a loss of tracking eventually. Therefore, it is important to ensure that all the pixels used are sampled from the object surface.

As the pose estimation from the previous frame ($\theta_{t-1} = [R_{t-1}, \boldsymbol{t}_{t-1}]$) is known, we can transform the 3D associated position of every pixel (from the previous frame) into the object reference frame and use the learned SDF to classify whether the pixel belongs to the tracked object. If $D(R_{t-1}^T({}_C\boldsymbol{p}_i - \boldsymbol{t}_{t-1}))$ is less than a small positive value (e.g. 0.004 m), the pixel i of previous frame will be classified as object surface. Otherwise, it will be classified as non-object.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

To accelerate this process, we introduce a classification interval k. Then, we only consider pixels every k-th row and k-th column with distance from $\boldsymbol{t}_{t-1}$ less than the maximum object radius r_o. All the classified object and non-object points are kept in the object reference frame as ${}_H\boldsymbol{p}_i = R_{t-1}^T({}_C\boldsymbol{p}_i - \boldsymbol{t}_{t-1})$.

In the current frame t, the collected non-object points ${}_H\boldsymbol{p}_i$ from the previous frame will be transformed by the previous pose θ_{t-1} and the predicted pose $\hat{\theta}_t$ separately to represent both non-object points from the previous frame and predicted non-object points in the current frame. The transformed points are then projected onto the image plane. When a transformed non-object point is projected to a pixel, that pixel and its neighbor within Chebyshev distance of k will record the shallowest depth from all non-object projections. All these records can be considered as potential occluders, meaning that any point in the volume behind them should not be sampled.

Then, the collected object points from the previous frame will be transformed by θ_{t-1} and $\hat{\theta}_t$ and then projected to the depth image. If the projected depth is at least 10 cm shallower than the occluder at that pixel and if the current observed depth is within 5 cm from the projected depth, we consider that pixel to be safe to sample. Among those survivals, m pixels will be sampled randomly.

The rationale of this method is to cover both static occlusions and those which move together with the object such as hand and fingers. Therefore, θ_{t-1} and $\hat{\theta}_t$ are used to represent the two kinds of occlusions. Moreover, the expansion of the occlusions after projection will give some margins for the error in pose prediction and unpredicted movements of those occlusions.

2.5 ICP-based Fitting Approach

This approach is similar to the well-established ICP except for the correspondence searching step. Instead of finding the exact closest point on the object surface, the trained SDF is used to infer the correspondence. The latest state of the pose $\tilde{\theta} = [\tilde{R}, \tilde{\boldsymbol{t}}]$, initialized as $\tilde{\theta} := [R_{t-1}, \hat{\boldsymbol{t}}_t]$, will be updated iteratively until a stopping condition is met.

Pseudo-correspondence Inference. Given ${}_H\boldsymbol{p}_i = \tilde{R}^T({}_C\boldsymbol{p}_i - \tilde{\boldsymbol{t}})$, with $i = 1, 2, \ldots, m$, as the transformed sampled 3D points from Section 2.4 using the latest state of the pose $\tilde{\theta}$, their correspondences will be

$$ {}_H\boldsymbol{e}_i = {}_H\boldsymbol{p}_i - D({}_H\boldsymbol{p}_i) \cdot \mathcal{N}(\boldsymbol{G}({}_H\boldsymbol{p}_i)) \tag{5} $$

where $\mathcal{N}(\cdot)$ is the vector normalization.

Pose Update. Let ${}_H\boldsymbol{c}_p$ be the centroid location calculated from all ${}_H\boldsymbol{p}_i$ and ${}_H\boldsymbol{c}_e$ be the centroid location calculated from all ${}_H\boldsymbol{e}_i$. The optimal orientation ΔR can be found by minimizing the sum of pair-wise distances:

$$ E(\Delta R) = \sum_{i=1}^{m} \|({}_H\boldsymbol{c}_e + \Delta R({}_H\boldsymbol{p}_i - {}_H\boldsymbol{c}_p)) - {}_H\boldsymbol{e}_i\|^2. \tag{6} $$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

Let P be a matrix whose i-th column is a vector ${}_H\boldsymbol{p}_i - {}_H\boldsymbol{c}_p$ and Q be a matrix whose i-th column is a vector ${}_H\boldsymbol{e}_i - {}_H\boldsymbol{c}_e$. E is minimized by performing a singular value decomposition (SVD) to the cross-covariance matrix $M = PQ^\top$ which results in $M = UWV^\top$. The rotation matrix is obtained from $\Delta R = VU^\top$. As a consequence, the optimal translation will be $\Delta\boldsymbol{t} = {}_H\boldsymbol{c}_e - \Delta R\,{}_H\boldsymbol{c}_p$.

While ΔR and $\Delta\boldsymbol{t}$ moves the sampled observation points to fit with the signed distance field in the object reference frame, the state $\tilde{R}$ and $\tilde{\boldsymbol{t}}$ perform the opposite. Therefore, an inversion will be applied to update the parameters in the following order:

$$\tilde{\boldsymbol{t}} := \tilde{\boldsymbol{t}} - \tilde{R}\Delta\boldsymbol{t} \qquad \text{then} \qquad \tilde{R} := \tilde{R}(\Delta R)^T \tag{7}$$

Stopping Condition. The iteration will stop when the number of iterations has reached n_{max} or the update is small. For the latter criterion, we set the condition to be $(3 - \text{Trace}(\Delta R))/3 < 10^{-6}$ and $\Delta\boldsymbol{t} < 10^{-5}$. Then, the latest state will be assigned to the final pose estimation $\theta_t := \tilde{\theta}$.

2.6 Optimization-based Fitting Approach

Cost Function. This approach makes use of an optimization algorithm to perform the fitting for current frame t. In our application, the Levenberg-Marquardt algorithm (LM) is used to minimize a cost function defined as

$$F(\Delta R, \Delta\boldsymbol{t}) = \sum_{i=1}^{m} D\Big(\Delta R\tilde{R}^\top\big[{}_C\boldsymbol{p}_i - (\tilde{\boldsymbol{t}} + \Delta\boldsymbol{t})\big]\Big)^2 \tag{8}$$

where ΔR and $\Delta\boldsymbol{t}$ are the changes of orientation and translation respectively, and m is the number of sampled points used in the optimization.

In contrast to Section 2.5, the current state $\tilde{\theta} = [\tilde{R}, \tilde{\boldsymbol{t}}]$ is initialized as $\tilde{\theta} := [R_{t-1}, \boldsymbol{t}_{t-1}]$, while ΔR and $\Delta\boldsymbol{t}$ are initialized to be the identity rotation and the zero vector respectively.

Formulation of Jacobian. For the cost function in Eq.8, the rotation ΔR is represented using a quaternion $\boldsymbol{q} = w + x\hat{i} + y\hat{j} + z\hat{k}$.

Given that $\boldsymbol{\omega} = [x\ y\ z]^\top$, a point $\boldsymbol{p}$ rotated by the quaternion $\boldsymbol{q}$ is

$$\boldsymbol{p}_{\boldsymbol{rotated}} = \boldsymbol{p} + 2w(\boldsymbol{\omega} \times \boldsymbol{p}) + 2[\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{p})] \tag{9}$$

For each point ${}_C\boldsymbol{p}_i$, we define $\boldsymbol{p}'_i = \tilde{R}^\top\big[{}_C\boldsymbol{p}_i - (\tilde{\boldsymbol{t}} + \Delta\boldsymbol{t})\big]$, $D_i = D(\Delta R\boldsymbol{p}'_i)$ and $\boldsymbol{G}_i = \boldsymbol{G}(\Delta R\boldsymbol{p}'_i)$. The Jacobian J can be derived as follow:

$$\left(\frac{\partial F}{\partial \Delta\boldsymbol{t}}\right)_i = -\boldsymbol{G}_i^\top(\Delta R\tilde{R}^\top) \tag{10}$$

$$\left(\frac{\partial F}{\partial w}\right)_i = 2\boldsymbol{G}_i^\top\big([\boldsymbol{\omega}]_\times \boldsymbol{p}'_i\big) \tag{11}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

$$\left(\frac{\partial F}{\partial v}\right)_i = 2\boldsymbol{G}_i^\top \left[\left(w[\boldsymbol{e}_v]_\times + [\boldsymbol{e}_v]_\times[\boldsymbol{\omega}]_\times + [\boldsymbol{\omega}]_\times[\boldsymbol{e}_v]_\times\right)\boldsymbol{p}_i'\right] \tag{12}$$

where $v \in \{x, y, z\}$, $\boldsymbol{e_v}$ are the respective standard basis vectors, and $[\cdot]_\times$ represents the skew symmetric matrix.

When ΔR and $\Delta \boldsymbol{t}$ are the identity rotation and the zero vector respectively, we have $\boldsymbol{p}_i' = \tilde{R}({}_C\boldsymbol{p}_i - \tilde{\boldsymbol{t}})$ and the Jacobian calculations are simplified to

$$\left(\frac{\partial F}{\partial \Delta \boldsymbol{t}}\right)_i = -\boldsymbol{G}_i^\top \tilde{R}^\top, \qquad \left(\frac{\partial F}{\partial w}\right)_i = 0, \qquad \left(\frac{\partial F}{\partial v}\right)_i = 2\boldsymbol{G}_i^\top \left([\boldsymbol{e}_v]_\times \boldsymbol{p}_i'\right) \tag{13}$$

Pose Update. The change in pose parameters $\boldsymbol{\delta}$ is computed from

$$\left[J^\top J + \lambda K\right]\boldsymbol{\delta} = J^\top \boldsymbol{r} \tag{14}$$

where $\boldsymbol{r} = -[D_1\ D_2\ \ldots\ D_m]^\top$, λ is the damping factor of the optimization. K can be chosen as the identity matrix or diag $\left(J^\top J\right)$.

Adding the respective components of $\boldsymbol{\delta}$ to the zero translation and the identity quaternion, we have $\Delta \boldsymbol{t} = [\Delta t_x\ \Delta t_y\ \Delta t_z]^\top$ and $\Delta q = 1 + \Delta x\hat{i} + \Delta y\hat{j} + \Delta z\hat{k}$, where the latter is normalized before being converted to the rotation matrix ΔR.

The current pose $\tilde{R}$ and $\tilde{\boldsymbol{t}}$ are updated with $\tilde{R} := \tilde{R}(\Delta R)^\top$ and $\tilde{\boldsymbol{t}} := \tilde{\boldsymbol{t}} + \Delta \boldsymbol{t}$. The terms ΔR and $\Delta \boldsymbol{t}$ are always reset to the identity rotation and the zero vector for the next iteration.

The optimization process is iterated until the cost function (Eq. 8), its change, or the magnitude of $\boldsymbol{\delta}$ falls below their respective thresholds, or the number of iterations has reached n_{max}.

3 Evaluation

3.1 Quantitative Evaluation

Our proposed method is evaluated using synthetic dataset from Choi and Christensen [5] which consists of four RGB-D sequences with ground truth object trajectories. The parameters used in the ICP-based method are $m = 50$, $k = 5$, and $n_{max} = 20$. The parameters used in the LM-based method are $m = 30$, $k = 11$, $\lambda_{init} = 0.1$, $\lambda_{scaling} = 2$, $n_{max} = 20$, and $K = I$. The pose is initialized using the ground truth in the first frame for each sequence. The algorithms are run on a single CPU thread on Intel Core i7-4770 @ 3.4 GHz.

3.2 Qualitative Evaluation

The tracking methods are tested on two different real objects, namely the Stanford Bunny and a detergent bottle. The bunny is 3D-printed using the mesh model obtained from the Stanford 3D Scanning Repository [16]. Meanwhile, the detergent bottle is scanned using Microsoft Kinect V2 and *3D Scan* software. In the evaluation, we demonstrate our dynamic tracking with occlusions by nearby objects and by the hand holding the object. We have also tested the algorithms on two types of depth cameras: Intel RealSense SR300 that uses a structured-light sensor and Microsoft Kinect V2 that uses a time-of-flight (ToF) sensor.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

4 Results & Discussion

4.1 The Trained Networks

Multiple Stages in the Training. All the networks, despite the randomized weight initializations, have evolved automatically through a few stages in a surprisingly systematic way. During the initial few thousands of iterations, the network would flatten the value of the output node to a very small value as it tries to satisfy the *SurfaceDistancePenalty* term in Eq.1. As a side effect, all the gradient magnitudes in the whole volume would be almost close to zero. Most of the gradient errors in the *GradientPenalty* term will stay slightly below one as we expect all the gradients to have a magnitude of one. Next, the network will slowly shape the terrain to satisfy the *GradientPenalty* term while the surface points are already loosely nailed at the near zero output value.

Single Point Convergence. To verify that the SDF-Net and its gradient can be used to guide any point in the surrounding region towards the object surface, we try to reconstruct the surface using random points sampled from the region. Each point will get updated by $\boldsymbol{p} := \boldsymbol{p} - 0.01\boldsymbol{G}(\boldsymbol{p}) \cdot \text{sign}(D(\boldsymbol{p}))$ iteratively until $D(\boldsymbol{p})$ passes a zero-crossing point. Then, the zero-crossing position is linearly interpolated from the two latest positions. The estimated zero-crossing points are collectively shown in Fig.1. All the random points are able to converge on the surface in a virtually straight trajectory without being trapped in any local minimum. Refer to the interactive results in the supplementary materials [10].

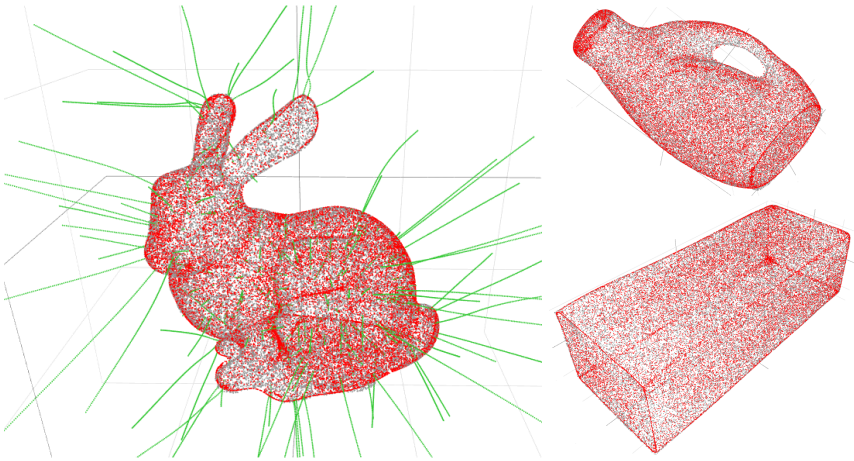

Fig. 1. Convergence of random points. Grey cloud: points on the actual model surface. Red cloud: points converged from random positions. Green lines: some examples of point travelling paths from a random position to the surface. 3D interactive interface of all trained network can be found in the supplementary repository [10].

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

Surface Smoothing Error. One inherent nature of the neural network is to smooth out noises in its training data. Thus, some small regions on the object surface may not be modelled accurately, especially protruding volumes such as the corner of a box or a deep concave surface that is too small. As they are small relative to the whole surface area, it should not largely affect the tracking.

4.2 Object Tracking

Quantitative Results. For the ICP-based method, we obtain the best average accuracy with a 0.85 mm translation error and a 0.27° orientation error with an average processing time of 5.11 ms. The accuracy is better than the LM-based method, which has average accuracies of 0.93 mm and 0.48° for translation and rotation errors respectively. However, the LM-based method runs at a faster rate of 1.29 ms. In terms of memory usage, our method requires the least memory footprint among the state-of-the-arts at 9 kB or lower if a smaller network is used.

Among the methods that can run at sub-degree accuracy with a real-time speed on CPU, most of them [29, 1, 28] are developed on a decision-forest-based regression method [27]. Some methods are learning-free [11, 20] but developed on a well-established PWP3D method [18] which relies heavily on the colour data. Our new branch of methods has reached the similar level of accuracy and calculation time on one CPU core without relying on any decision forest or colour-based method. Refer to Table 1 for detailed results.

According to Table 1, the translation errors perpendicular to the optical axis (t_x and t_y) are 2-3 times larger than the translation errors along the depth axis (t_z) in our methods and in Tan et al.'s method [29]. However, the drifts (t_x and t_y) are minimized in Tan et al.'s later method [28] as they utilise colour to extract edge points, which constrain the whole model from sliding along the plane perpendicular to the optical axis; hence, the overall error is reduced significantly.

The comparison between the ICP-based and the LM-based methods is presented in Table 2. The LM-based method is found to take much lesser time when compared with the ICP-method. This is because of the fast convergence rate due to the low damping factor λ, which results in lower iteration count. In addition, a sparser classification interval k and a lower number of sampled points m are used. This configuration greatly reduces the computational time without compromising much accuracy.

Table 2 also compares the result from the small and the large neural network. As expected, network $\mathcal{B}$ with more memory capacity has a better overall accuracy but requires additional 20% of calculation time when compared with network $\mathcal{A}$.

In terms of memory footprint, our representation is the most compact, with 6 kB and 9 kB for the network $\mathcal{A}$ and $\mathcal{B}$ respectively. With this size, a discrete representation of SDF can only represent a crude model of the surface.

Qualitative Results. The results show that the object remains tracked even when more than half of it is occluded (Fig. 2b,2f). Also, our methods are able to

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

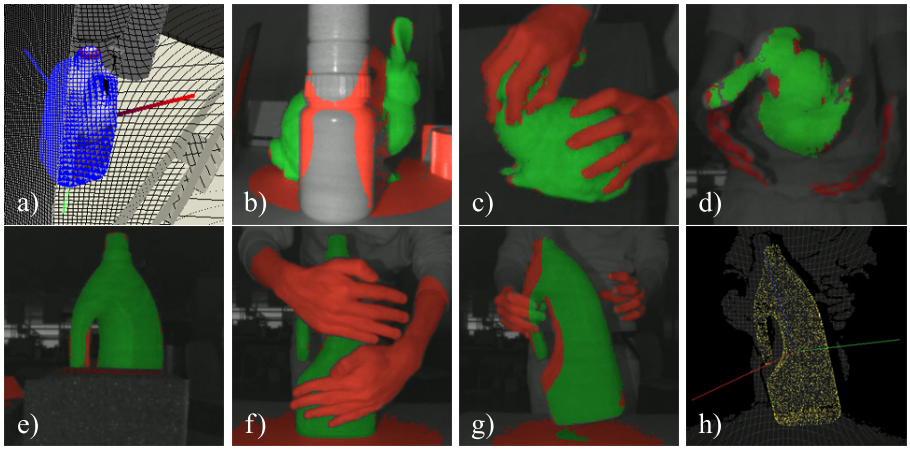

Fig. 2. a) An example of fitting results from the dataset [5] with partial occlusion. **b-g)** Examples of post-fitting segmentation of objects with occlusions; green and red regions represent pixels classified as object and non-object points respectively. **g-h)** When the object is lifted abruptly, the tracking fails as the predicted pose $\hat{\theta}$ lags behind the actual observation, causing the failure in sampling some key features, e.g. the handle of the detergent, that are critical in determining the object pose.

track the object when it is toppled and thrown (Fig. 2d). Videos of our tracking results can be found in the supplementary repository [10].

When the object is static, the fitted model generally shows slight jitter when the ICP-based method is used; this is less likely to be seen for the LM-based method. This is because the LM algorithm always verifies if a particular update will lower the cost function (Eq.8), making it tend to stay at its pose from the previous frame; this feature is not present in the ICP-based method.

The comparison between the tracking results on both short-ranged and long-ranged cameras shows that the latter generally yields sub-par performance. This is mainly due to a larger depth error at far distance together with the *multi-path effect* which only occurs in ToF camera.

After the fitting process is done in each frame, as a by-product, SDF-Net can be used to differentiate object and non-object pixels without the use of colour. Hands and fingers on the object are also segmented out (Fig. 2c,2f). This feature could be useful in hand tracking applications with object interaction.

5 Conclusion & Future Work

We have shown that a deep neural network trained with our proposed method could be used to learn an approximation of an object SDF which is accurate enough for object tracking purpose. Our results have been shown to reach up to sub-millimeter and sub-degree accuracy when evaluated on a public dataset. The real-time capability of our rigid object tracking method has been demonstrated

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

Table 1. Benchmark comparison of the RMS errors in translation (mm), orientation (degree) and the runtime (ms) on a synthetic dataset [5]. The required memory footprints used to store pre-calculated content are approximated. The results from our two methods, ICP-based and LM-based, are from the bigger network $\mathcal{B}$.

	Errors	PCL [24]	C&C [5]	Krull [14]	Tan'15 [29]	Kehl [11]	S.A [2]	Tan'17 [28]	Ours ICP	Ours LM
Kinect Box	t_x	43.99	1.84	0.83	1.54	0.76	0.30	0.15	1.27	1.12
	t_y	42.51	2.23	1.67	1.90	1.09	0.49	0.19	1.22	1.81
	t_z	55.89	1.36	0.79	0.34	0.38	0.31	0.09	0.50	0.56
	Roll	7.62	6.41	1.11	0.42	0.17	0.21	0.09	0.16	0.16
	Pitch	1.87	0.76	0.55	0.22	0.18	0.27	0.06	0.17	0.30
	Yaw	8.31	6.32	1.04	0.68	0.20	0.23	0.04	0.10	0.17
	Time	4539	166	143	1.5	8.10	1.71	2.2	5.62	1.37
Milk	t_x	13.38	0.93	0.51	1.23	0.64	0.63	0.09	0.98	1.01
	t_y	31.45	1.94	1.27	0.74	0.59	1.19	0.11	0.80	0.76
	t_z	26.09	1.09	0.62	0.24	0.24	0.48	0.08	0.32	0.36
	Roll	59.37	3.83	2.19	0.50	0.41	0.19	0.07	0.20	0.34
	Pitch	19.58	1.41	1.44	0.28	0.29	0.28	0.09	0.27	0.76
	Yaw	75.03	3.26	1.90	0.46	0.42	0.27	0.06	0.15	0.22
	Time	2205	134	135	1.5	8.54	1.70	2.1	4.99	1.26
Orange Juice	t_x	2.53	0.96	0.52	1.10	0.50	0.39	0.11	1.12	1.15
	t_y	2.20	1.44	0.74	0.94	0.69	0.37	0.09	0.88	0.96
	t_z	1.91	1.17	0.63	0.18	0.17	0.37	0.09	0.67	0.67
	Roll	85.81	1.32	1.28	0.35	0.12	0.12	0.08	0.41	0.44
	Pitch	42.12	0.75	1.08	0.24	0.20	0.17	0.08	0.25	0.65
	Yaw	46.37	1.39	1.20	0.37	0.19	0.15	0.08	0.39	0.65
	Time	1637	117	129	1.5	8.79	1.69	2.2	4.79	1.25
Tide	t_x	1.46	0.83	0.69	0.73	0.34	0.42	0.08	0.99	1.01
	t_y	2.25	1.37	0.81	0.56	0.49	0.51	0.09	1.03	1.05
	t_z	0.92	1.20	0.81	0.24	0.18	0.64	0.07	0.28	0.30
	Roll	5.15	1.78	2.10	0.31	0.15	0.22	0.05	0.09	0.24
	Pitch	2.13	1.09	1.38	0.25	0.39	0.29	0.12	0.54	1.03
	Yaw	2.98	1.13	1.27	0.34	0.37	0.30	0.05	0.13	0.34
	Time	2762	111	116	1.5	9.42	1.70	2.2	5.04	1.30
Mean	Transl.	18.72	1.36	0.82	0.81	0.51	0.50	0.10	0.85	0.93
	Rot.	29.70	2.45	1.38	0.37	0.26	0.22	0.07	0.27	0.48
	Time	2786	132	131	1.5	8.71	1.7	2.2	5.11	1.29
Requires	Memory	-	-	-	7.4MB	10MB	123kB	13MB	9 kB	
	GPU	-	✓	✓	-	-	-	-	-	
	Colour	✓	✓	✓	-	✓	-	✓	-	
	Depth	✓	✓	✓	✓	✓	✓	✓	✓	

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

Table 2. Comparison of our methods using two different networks on the synthetic dataset [5]. Note that the translation and orientation errors in this table are different from those in Table 1; we use the Euclidean distance and the angle difference between the estimated and the ground truth orientation to compute the errors as these measures are invariant under coordinate transformation.

Network	ICP-based method		LM-based method	
	$\mathcal{A}$ (smaller)	$\mathcal{B}$ (bigger)	$\mathcal{A}$ (smaller)	$\mathcal{B}$ (bigger)
Average translation error (mm)	1.62	1.49	1.70	1.55
Maximum translation error (mm)	15.75	4.82	10.02	8.27
Average orientation error (deg)	0.43	0.33	0.77	0.74
Maximum orientation error (deg)	3.59	3.06	6.80	4.30
Average number of iterations	13.84	12.89	2.32	2.05
Calculation time per frame (ms)	4.03	5.11	1.11	1.29

using depth data from commodity depth cameras and the algorithm could run on a single CPU thread.

As the proposed tracking method works by finding the transformation parameters between consecutive frames, the initial pose of the object must be provided. Also, in case of tracking loss, the object pose has to be reinitialized manually. Hence, initialization and detection of the object in real-time are to be investigated in the future.

References

1. Akkaladevi, S., Ankerl, M., Heindl, C., Pichler, A.: Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In: IEEE Int. Conf. on Robotics and Autom. pp. 5644–5649 (2016)
2. Akkaladevi, S.C., Ankerl, M., Fritz, G., Pichler, A.: Real-time tracking of rigid objects using depth data. In: Proc. of OAGM&ARW Jt. Workshop (2016)
3. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992). https://doi.org/10.1109/34.121791
4. Canelhas, D.R., Stoyanov, T., Lilienthal, A.J.: SDF tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In: IEEE/RSJ Int. Conf. on Intell. Robots and Syst. pp. 3671–3676 (2013)
5. Choi, C., Christensen, H.I.: RGB-D object tracking: A particle filter approach on GPU. In: IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (2013)
6. Frisken, S.F.: Designing with distance fields. In: Int. Conf. on Shape Model. and Appl. pp. 58–59 (2005). https://doi.org/10.1109/SMI.2005.16
7. Garon, M., Lalonde, J.F.: Deep 6-DOF tracking. IEEE Trans. Vis. Comput. Graphics **23**(11), 2410–2418 (2017). https://doi.org/10.1109/TVCG.2017.2734599
8. Issac, J., Wthrich, M., Cifuentes, C.G., Bohg, J., Trimpe, S., Schaal, S.: Depth-based object tracking using a robust gaussian filter. In: 2016 IEEE Int. Conf. on Robotics and Autom. (2016). https://doi.org/10.1109/ICRA.2016.7487184
9. Jabarouti Moghaddam, M., Soltanian-Zadeh, H.: Automatic segmentation of brain structures using geometric moment invariants and artificial neural networks. In: Inf. Process. in Med. Imaging. pp. 326–337 (2009)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3

10. Jatesiktat, P., Foo, M.J., Lim, G.M.: SDF-Net: Supplementary materials repository. `https://koonyook.github.io/SDF-Net-materials/`
11. Kehl, W., Tombari, F., Ilic, S., Navab, N.: Real-time 3D model tracking in color and depth on a single CPU core. In: IEEE Conf. on Comput. Vis. and Pattern Recognit. pp. 465–473 (2017). https://doi.org/10.1109/CVPR.2017.57
12. Kehl, W., Navab, N., Ilic, S.: Coloured signed distance fields for full 3D object reconstruction. In: Proc. of the Br. Mach. Vis. Conf. (2014)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. Comput. Res. Repos. **abs/1412.6980** (2014), `http://arxiv.org/abs/1412.6980`
14. Krull, A., Michel, F., Brachmann, E., Gumhold, S., Ihrke, S., Rother, C.: 6-DOF model based tracking via object coordinate regression. In: Asian Conf. on Comput. Vis. LNCS, vol. 9006, pp. 384–399 (2014)
15. Lepetit, V., Fua, P.: Monocular model-based 3D tracking of rigid objects. Found. Trends. Comput. Graph. Vis. **1**(1), 1–89 (2005)
16. Levoy, M.: The stanford 3d scanning repository. `http://graphics.stanford.edu/data/3Dscanrep/`
17. Lu, G., Ren, L., Kolagunda, A., Wang, X., Turkbey, I.B., Choyke, P.L., Kambhamettu, C.: Representing 3D shapes based on implicit surface functions learned from RBF neural networks. J. Vis. Commun. Image Represent. **40**, 852 – 860 (2016). https://doi.org/10.1016/j.jvcir.2016.08.014
18. Prisacariu, V., Reid, I.: PWP3D: Real-time segmentation and tracking of 3D objects. Int. J. of Comput. Vis. pp. 1–20 (2012)
19. Ren, C.Y., Prisacariu, V., Murray, D., Reid, I.: STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data. In: IEEE Int. Conf. on Comput. Vis. pp. 1561–1568 (2013). https://doi.org/10.1109/ICCV.2013.197
20. Ren, C.Y., Prisacariu, V.A., Kähler, O., Reid, I.D., Murray, D.W.: Real-time tracking of single and multiple objects from depth-colour imagery using 3D signed distance functions. Int. J. of Comput. Vis. **124**(1), 80–95 (2017)
21. Rouhani, M., Sappa, A.D.: Correspondence free registration through a point-to-model distance minimization. In: Int. Conf. on Comput. Vis. pp. 2150–2157 (2011)
22. Rouhani, M., Sappa, A.D.: The richer representation the better registration. IEEE Trans. on Image Process. **22**(12), 5036–5049 (2013)
23. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proc. Third Int. Conf. on 3-D Digit. Imaging and Model. pp. 145–152 (2001)
24. Rusu, R.B., Cousins, S.: 3D is here: Point cloud library (PCL). In: IEEE Int. Conf. on Robotics and Autom. pp. 1–4 (2011)
25. Slavcheva, M., Kehl, W., Navab, N., Ilic, S.: SDF-2-SDF: Highly accurate 3D object reconstruction. In: Eur. Conf. on Comput. Vis. Springer (2016)
26. Somani, N., Cai, C., Perzylo, A., Rickert, M., Knoll, A.: Object recognition using constraints fromprimitive shape matching. In: Adv. in Vis. Comput. pp. 783–792 (2014). https://doi.org/10.1007/978-3-319-14249-4_75
27. Tan, D.J., Ilic, S.: Multi-forest tracker: A chameleon in tracking. In: IEEE Conf. on Comput. Vis. and Pattern Recognit. pp. 1202–1209 (2014)
28. Tan, D.J., Navab, N., Tombari, F.: Looking beyond the simple scenarios: Combining learners and optimizers in 3D temporal tracking. IEEE Trans. Vis. Comput. Graph. **23**(11), 2399–2409 (2017). https://doi.org/10.1109/TVCG.2017.2734539
29. Tan, D.J., Tombari, F., Ilic, S., Navab, N.: A versatile learning-based 3D temporal tracker: Scalable, robust, online. In: IEEE Int. Conf. on Comput. Vis. (2015)
30. Zhang, Z.: On local matching of free-form curves. In: Proc. of the Br. Mach. Vis. Conf. pp. 347–356. Springer London (1992)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_3