sensors

MDPI

Article

CuFusion: Accurate Real-Time Camera Tracking and Volumetric Scene Reconstruction with a Cuboid

Chen Zhang * and Yu Hu

College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; yudeshui@zju.edu.cn

* Correspondence: zhangxaochen@163.com; Tel.: +86-135-7575-2056

Received: 6 August 2017; Accepted: 27 September 2017; Published: 1 October 2017

Abstract: Given a stream of depth images with a known cuboid reference object present in the scene, we propose a novel approach for accurate camera tracking and volumetric surface reconstruction in real-time. Our contribution in this paper is threefold: (a) utilizing a priori knowledge of the precisely manufactured cuboid reference object, we keep drift-free camera tracking without explicit global optimization; (b) we improve the fineness of the volumetric surface representation by proposing a prediction-corrected data fusion strategy rather than a simple moving average, which enables accurate reconstruction of high-frequency details such as the sharp edges of objects and geometries of high curvature; (c) we introduce a benchmark dataset CU3D that contains both synthetic and real-world scanning sequences with ground-truth camera trajectories and surface models for the quantitative evaluation of 3D reconstruction algorithms. We test our algorithm on our dataset and demonstrate its accuracy compared with other state-of-the-art algorithms. We release both our dataset and code as open-source (https://github.com/zhangxaochen/CuFusion) for other researchers to reproduce and verify our results.

Keywords: real-time reconstruction; SLAM; Kinect sensors; depth cameras; open source

1. Introduction

Real-time camera tracking and simultaneous dense scene reconstruction has been one of the most actively studied problems in computer vision over recent years. The advent of depth cameras based either on structured light (e.g., Asus Xtion, Kinect 1.0) or time-of-flight (ToF) (e.g., Kinect 2.0) sensing offers dense depth measurements directly in real-time as video streams. Such dense depth sensing technologies have drastically simplified the process of dense 3D modeling, which turns the widely available Kinect-style depth cameras into consumer-grade 3D scanners.

KinectFusion [1] is one of the most famous systems for registering each incoming frame of depth images captured during the scanning into one integrated volumetric representation of the scene. An iterative closest point (ICP) algorithm [2] is performed to align the current depth map to the reconstructed volumetric truncated signed distance function (TSDF) [3] surface model to get the camera pose estimation. Each depth measurement is fused into the TSDF model directly to update the reconstruction. A triangulated 3D mesh model could finally be extracted using a Marching Cubes type algorithm [4].

Existing geometric alignment approaches based on ICP and its variants [5] are prone to drift in the presence of structure-less surfaces. Drift might be accumulated and even cause the failure of camera tracking when scanning larger man-made environments. Meanwhile, the weighted moving average TSDF fusion strategy makes the assumption of a Gaussian noise model on the depth measurements with a naïve surface visibility predicate that every surface point is visible from all sensor viewpoints [6]. This predicate is only locally true and usually violated due to surface occlusions [1] when scanning

around the scene. Although truncation of the signed distance function (SDF) is performed to avoid surfaces interfering, surface blurring and the inflating problem (as shown in Figure 1c) may happen when scanning around tiny objects or sharp geometries in the scene.

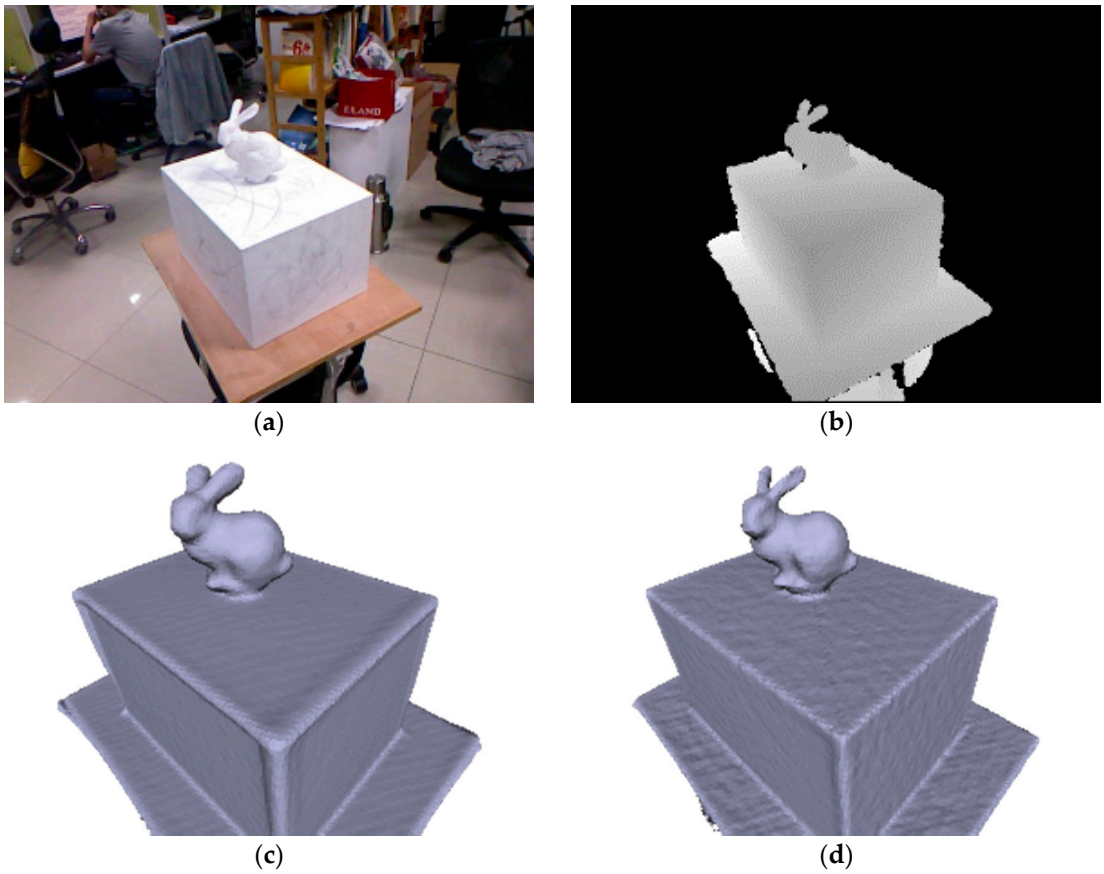


Figure 1. (**a**) Color (not used) and (**b**) depth image from our input sequence "lambunny"; (**c**) KinectFusion: mild accumulated camera drift and simple moving average truncated signed distance function (TSDF) fusion result in reconstruction inflation; (**d**) Our approach, CuFusion, keeps drift free camera tracking with additional constraints of a cuboid reference object and preserves the fidelity of the reconstructed objectives using our prediction-corrected TSDF fusion strategy. Note the sharpness of the cuboid edges and the thinness of the character's ears of our reconstruction.

Existing algorithms have been proposed to keep globally consistent camera trajectory estimation. Pose graphs are created and optimized when large loop closures are found [7], which may substantially reduce the odometry error accumulation. On the task of scanning small-sized scenes or objects, however, even small camera drift may cause deformation of the reconstruction. We propose a novel algorithm called CuFusion, which particularly focuses on the application of reconstructing small-sized scenes and objects precisely in real-time, with the accuracy of both camera tracking and data fusion significantly improved. With a priori knowledge of the planar faces and occluding contours of the cuboid reference object partly or totally present in the scene, each data frame is aligned against both the reconstructed scene and the localized cuboid model, and thus drift-free camera trajectories are maintained.

The predicate that every surface point is visible from all sensor viewpoints is only locally true due to surface occlusions [1]. In our work, we drop such assumptions and implement a "prediction-corrected" data fusion algorithm to integrate all incoming data into one geometrically consistent 3D model in the global reference frame. Instead of a simple moving average surface reconstruction, our work extends the TSDF representation by adding components storing the locally consistent TSDF value, the pixel ray and surface normal vector in each voxel grid for the detection of the camera view variation and correction

of the global TSDF value. Experimental results (Figure 1) show the ability of our fusion method to keep the structural details of surfaces, which is on par with, or better than, existing state-of-the-art reconstruction systems that focus mostly on camera tracking accuracy.

Many scanning and reconstruction systems use both RGB and depth images. Feature-based registration is combined with dense ICP shape matching to estimate the best alignment between consecutive frames. Our system exploits only depth information as input to maximize tracking accuracy for the following reasons: First, some depth cameras such as ASUS Xtion PRO are not accompanied by RGB cameras. Second, for RGB-D cameras which provide both color and depth streams, the spatiotemporal alignment of RGB and depth information in pixel level may not be perfect. Third, by using only depth data, our system enables scanning in complete darkness regardless of the ambient lighting conditions.

We evaluate our algorithm qualitatively and quantitatively using both noiseless synthetic and noisy real-world data captured by a hand-held Kinect. The synthesized data provide both ground-truth (GT) camera trajectories and GT mesh models enabling both the trajectories and reconstructions to be quantitatively evaluated. For real-world image sequences, unfortunately we do not have GT camera trajectories. We 3D printed several rigid models using a high precision 3D printer (http://www.dowell3d.com/3d/3.html) for scanning and evaluate the quality of our reconstructions directly compared with the GT models.

2. Related Work

The research into the real-time 3D model reconstruction problem has been extensively studied in recent decades. The advance of range sensing technology has facilitated the development of real-time interactive range scanners for dense 3D surface model acquisition. Such range sensors, particularly on active sensing technologies, could be categorized into different types including laser scanners [8,9], time-of-flight (ToF) [10,11] sensing and structured-light cameras [12]. The introduction of Microsoft's Kinect—based on structured-light sensing—has brought dense depth sensors to wide consumer-grade accessibility.

KinectFusion [1] of Newcombe et al. is one of the founding systems for real-time dense SLAM, taking a sequence of depth maps streamed from a Kinect-style sensor as the input to create a globally consistent 3D model of the scene. Despite its enlightenment, this algorithm has limitations in several aspects. First, pure geometric alignment of ICP is prone to drift in the presence structure-less surfaces. Second, the regular volumetric representation is memory consuming, which limits the size of the reconstructed model to medium sized rooms, also with limited resolution. Third, it cannot detect loop closures and therefore lacks the ability to recover from accumulating drift, leading to mesh artifacts.

Researchers have been making efforts to address the problems mentioned above. Henry et al. [13] were the first to combine texture feature matching with Generalized-ICP [14] using RGB-D data to reduce drift result from pure geometric alignment and increase the robustness of visual odometry [15]. Loop closure is detected when the previously seen region is revisited, and a pose graph is optimized to create a globally consistent map in [13], as well as in the work of Endres et al. [7,16], Whelan et al. [17,18] and Kerl et al. [19]. Whelan et al. [20] further proposed ElasticFusion, a novel algorithm for loop closure optimization without a pose graph. Moreover, higher-level primitives such as edges [21,22], occluding contours [23], curvature information [24], lines [25] and planes [26–29] are used as additional information to constrain the pose estimation process.

On dense scene representation, Whelan et al. [17] extended the KinectFusion algorithm spatially to support large unbounded scenes, with a cyclical buffer data structure. Endres et al. [7,16] used an octree-based mapping framework OctoMap [30] to generate a volumetric 3D map of the environment at scale, yet no mesh model is created. Other researchers have been using points and surfels [20,24,31–34] to represent the scene and render it with the surface-splatting technique [35]. Such point-based scene representation has significantly reduced computational complexity and lowered the memory overhead compared with the volumetric approaches and is therefore adequate for reconstructing large-scale

environments. Note that Lefloch et al. [24] use curvature information as an independent surface attribute for their real-time reconstruction, leading not only to camera drift reduction but also to improved scene reconstruction.

However, despite the efforts exerted, both the camera pose estimation and the reconstructed models are far from perfect. On small-sized scenes particularly, slight camera drift may lead to reconstruction deformation and sharp depth edges or highly concave scenes are problematic for these approaches [36]. We tackle these problems and focus on fidelity preservation in this paper.

3. Method

We base our work on an open-sourced implementation of the KinectFusion algorithm from the PCL library [37]. Our reconstruction pipeline is illustrated in Figure 2, which is described in detail in the following sections.

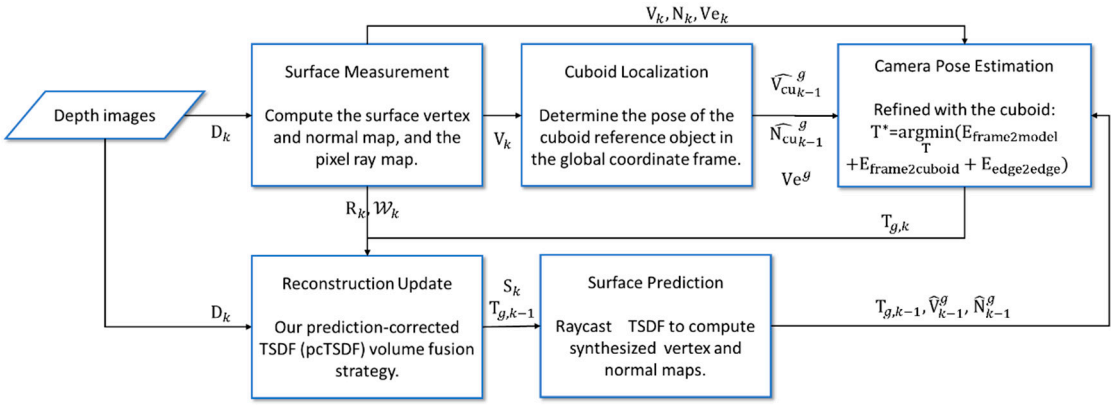


Figure 2. System overview.

3.1. Notation

We define the image domain as $\Omega \subset \mathbb{N}^2$, and a depth image $D_k : \Omega \to \mathbb{R}$ at time k. We represent the camera pose at time k in the global coordinate frame $\mathcal{F}_g$ by a rigid transformation matrix:

$$T_{g,k} = \begin{bmatrix} R_{g,k} & t_{g,k} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}(3), \tag{1}$$

with a 3×3 rotation matrix $R_{g,k} \in \mathbb{SO}(3)$ and a 3×1 translation vector $t_{g,k} \in \mathbb{R}^3$, which transforms a point $\mathbf{p}_k \in \mathbb{R}^3$ in the camera coordinate frame $\mathcal{F}_k$ to a global point $\mathbf{p}_g = R_{g,k}\mathbf{p}_k + t_{g,k} \in \mathbb{R}^3$. We model the depth camera by the simple pinhole model, and use a constant camera intrinsic matrix K to transform points on the sensor plane into image pixels:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where (f_x, f_y) are the horizontal and vertical focal lengths and (c_x, c_y) is the image coordinate of the principal point.

We define the 3D back-projection of an image pixel $\mathbf{u} \in \Omega$ as $\mathbf{p} = K^{-1}\dot{\mathbf{u}}D(\mathbf{u})$, where $\dot{\mathbf{u}} := (\mathbf{u}^T|1)^T$ is the homogeneous form of $\mathbf{u}$. And inversely, we define the perspective projection of point $\mathbf{p} = (x, y, z)^T$ as $\mathbf{u} = \pi(K\mathbf{p})$, where function $\pi(\mathbf{p}) = (x/z, y/z)^T$ performs perspective projection including de-homogenization process.

Prior to registration, an organized vertex map V_k is computed by bilateral-filtering and back-projecting the raw depth image D_k. The normal map N_k is computed using the PCA method. Given the camera pose $T_{g,k}$ at time k, we could transform both V_k, N_k to the global frame of coordinate:

$$\begin{cases} \dot{\mathrm{V}}_k^g(\mathbf{u}) = \mathrm{T}_{g,k}\dot{\mathrm{V}}_k(\mathbf{u}) \\ \mathrm{N}_k^g(\mathbf{u}) = \mathrm{R}_{g,k}\mathrm{N}_k(\mathbf{u}) \end{cases}, \tag{3}$$

3.2. Cuboid Localization

Given a depth image D_k and a rectangular cuboid with edge lengths $\mathcal{P}_{cu} = (\mathrm{a}, \mathrm{b}, \mathrm{c})$ present in the image, we localize the cuboid and calculate its pose in the global coordinate frame $\mathcal{F}_g$. Live depth frames will be latterly aligned against the reference cuboid when scanning around it to mitigate the accumulating camera drift.

We first perform plane segmentation using the Agglomerative Hierarchical Clustering (AHC) algorithm [38], as illustrated in Figure 3c. Then we check the orthogonality of the segmented planes. Two planes are considered to be orthogonal if the angle Θ_p between their normal vectors is approximately 90° (i.e., $|\Theta_p - 90°| < \varepsilon_\Theta$; $\varepsilon_\Theta = 5°$). Once we find three planes that are orthogonal to each other, we check the length of the intersecting line segments between the planes. If the three line segments' lengths match the cuboid edge length parameter $\mathcal{P}_{cu}$ approximately (differences below a threshold $\varepsilon_\mathcal{P} = 10$ mm), we claim to find the cuboid and mark the three planes as its adjacent planes.

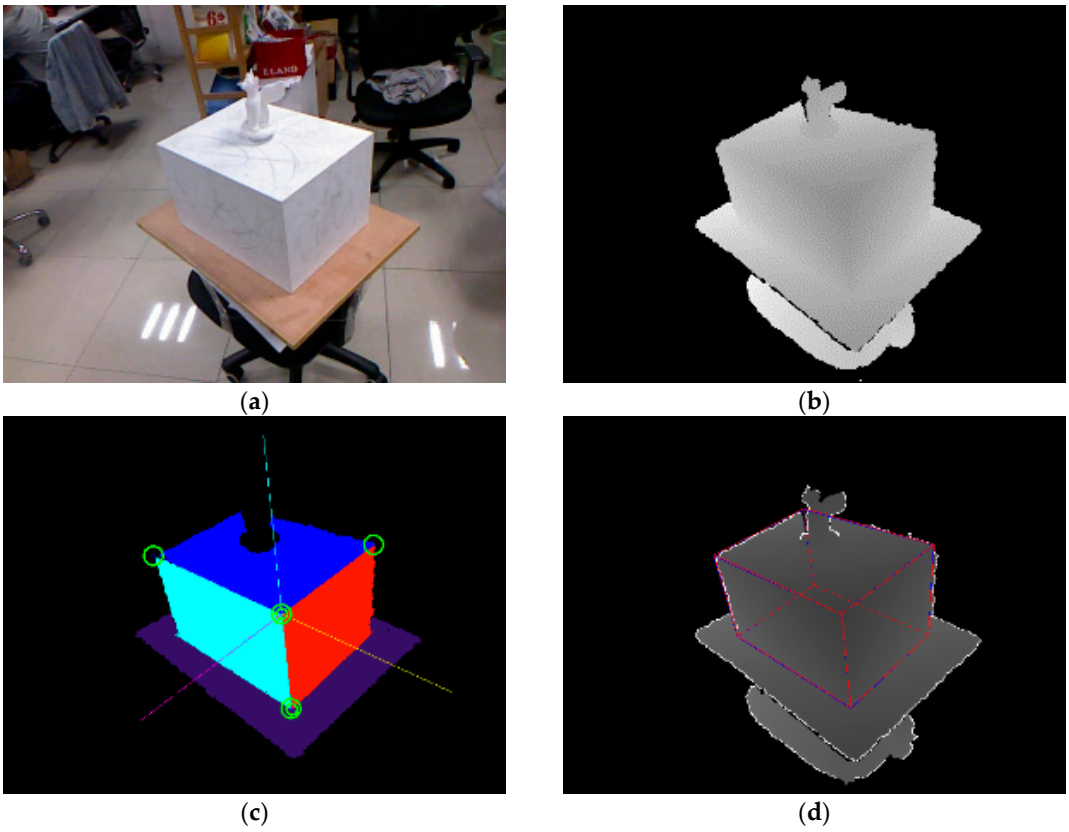


Figure 3. (**a**) Color (not used) and (**b**) depth image from our input sequence "wingedcat;" pixels with depth values larger than 1.5 m are truncated in the depth image; (**c**) Segmented planes obtained by the AHC algorithm [38] are labeled with random colors, the cuboid is localized with its vertices marked as green circles, and the axes of the cuboid frame are drawn in CMY colors; (**d**) The localized cuboid is drawn as a red wireframe in the depth image, and the "contour generators" proposed in [23] are drawn as white lines.

We consequently define the cuboid coordinate frame of reference. We set frame origin O_{cu} to the intersection point of the three orthogonal planes, and draw the system axes from the normal vectors. Due to the inaccuracy of the depth measurement and camera intrinsic calibration, orthogonality

between the normal vectors of the segmented adjacent planes are not guaranteed strictly. We obtain the nearest orthogonal axes $[X_{cu},\ Y_{cu},\ Z_{cu}]$ of the frame by solving the Orthogonal Procrustes Problem. The cuboid pose in the camera frame at time k is:

$$\mathrm{T}_{k,\,cu} = \begin{bmatrix} \mathrm{R}_{k,\,cu} & \mathrm{t}_{k,\,cu} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \in \mathbb{SE}(3), \tag{4}$$

$$\mathrm{R}_{k,\,cu} = [X_{cu},\ Y_{cu},\ Z_{cu}], \tag{5}$$

$$\mathrm{t}_{k,\,cu} = \mathrm{O}_{cu}^{\mathrm{T}}, \tag{6}$$

Assuming the camera pose $\mathrm{T}_{g,\,k}$ at time k is known, the cuboid pose $\mathrm{T}_{g,\,cu} = \begin{bmatrix} \mathrm{R}_{g,\,cu} & \mathrm{t}_{g,\,cu} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$ in the global frame of coordinate could then be derived: $\mathrm{T}_{g,\,cu} = \mathrm{T}_{g,\,k}\,\mathrm{T}_{k,\,cu}$. Figure 4 illustrates the notations used in the paper.

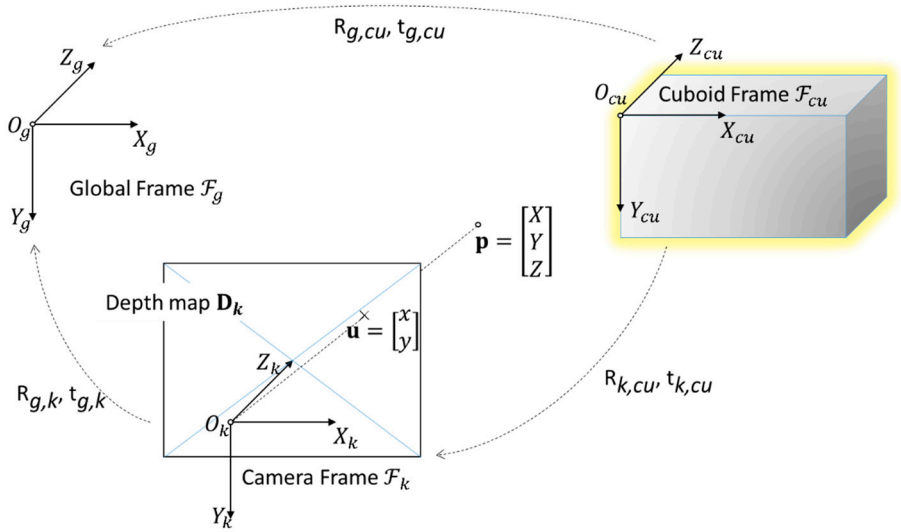


Figure 4. Illustration of the notations used in this paper.

3.3. Camera Pose Estimation

Since we use depth maps as input sequences, only geometric alignment is performed. For each input frame D_k at time k, we estimate the pose $\mathrm{T}_{g,\,k}$ of the depth camera frame $\mathcal{F}_k$ with respect to the global frame $\mathcal{F}_g$ by registering the live depth map to both the global reconstructed surface model and the cuboid reference object.

A. Frame to Model Registration

Given the implicit TSDF surface model $\mathbf{S}$, the surface prediction w.r.t. the camera pose $\mathrm{T}_{g,\,k-1}$ is obtained as an organized vertex and normal map $(\widehat{\mathrm{V}}_{k-1},\ \widehat{\mathrm{N}}_{k-1})$, and transformed into the global frame as $(\widehat{\mathrm{V}}^{g}_{k-1},\ \widehat{\mathrm{N}}^{g}_{k-1})$. For frame-to-model registration, a transformation $\mathrm{T}_{g,\,k}$ is pursued to minimize the point-to-plane error between $\mathrm{T}_{g,\,k}\mathrm{V}_k$ and $\widehat{\mathrm{V}}^{g}_{k-1}$:

$$\mathrm{E}_{frame2model}\left(\mathrm{T}_{g,\,k}\right) = \sum_{(\mathbf{u},\,\widehat{\mathbf{u}})\in\mathbb{K}_1} \left(\left(\mathrm{T}_{g,\,k}\dot{\mathrm{V}}_k(\mathbf{u}) - \widehat{\mathrm{V}}^{g}_{k-1}(\widehat{\mathbf{u}})\right)\widehat{\mathrm{N}}^{g}_{k-1}(\widehat{\mathbf{u}})\right)^2, \tag{7}$$

where $\mathbb{K}_1 = \{(\mathbf{u},\ \widehat{\mathbf{u}})\}$ is the set of correspondences obtained by projective data association [1]:

$$\widehat{\mathbf{u}} = \pi\left(\mathrm{K}\,\widetilde{\mathrm{T}}_{k-1,\,k}\dot{\mathrm{V}}_k(\mathbf{u})\right), \tag{8}$$

$\widetilde{\mathrm{T}}_{k-1,\,k}$ denotes the transformation from current time k to time $(k-1)$ during each ICP iteration.

B. Frame to Cuboid Registration

Assuming the cuboid pose w.r.t., the global coordinate frame is already known. For each camera pose $\mathrm{T}_{g,\,k}$, per-pixel ray casting is performed on the global cuboid to synthesize a proxy depth map $\widehat{\mathrm{D}}_k^{cu}$. An organized vertex and normal map in the global frame as $\left(\widehat{\mathrm{V}_{\mathrm{cu}}}_{k-1}^{g},\ \widehat{\mathrm{N}_{\mathrm{cu}}}_{k-1}^{g}\right)$ is then obtained using back projection of the depth map and local to global transformation. Similar to the frame-to-model registration, a frame is aligned against the cuboid surface in the global coordinate frame by minimizing the point-to-plane error:

$$\mathrm{E}_{frame2cuboid}\left(\mathrm{T}_{g,\,k}\right) = \sum_{(\mathbf{u},\,\widehat{\mathbf{u}})\in\mathbb{K}_2}\left(\left(\mathrm{T}_{g,\,k}\dot{\mathrm{V}}_k(\mathbf{u}) - \widehat{\mathrm{V}_{\mathrm{cu}}}_{k-1}^{g}(\widehat{\mathbf{u}})\right)\widehat{\mathrm{N}_{\mathrm{cu}}}_{k-1}^{g}(\widehat{\mathbf{u}})\right)^2, \tag{9}$$

In addition, we adopt the edge-to-edge error metric as a constraint to mitigate the potential camera drift. Given the inpainted depth map D'_k, we find the edge points (i.e., pixels at depth discontinuities) on the live depth map along the contour generator set $\mathbf{C}_k$ as proposed in [23]:

$$\mathbf{C}_k = \left\{\mathbf{s}\in\mathrm{D}_k\ :\exists \mathrm{t}\in\mathcal{N}_{\mathbf{s}}^8,\ \text{s.t. } \mathrm{D}'_k(\mathbf{s}) - \mathrm{D}'_k(\mathbf{t}) > \delta_c\right\}, \tag{10}$$

where $\mathcal{N}_{\mathbf{s}}^8$ is the 8-neighborhood of pixel $\mathbf{s}\in\mathrm{D}_k$ and δ_c is the depth discontinuity threshold, set to 50 mm according to the sensor noise magnitudes [39]. Figure 3d demonstrates the contour generators with white lines labeled on the depth map. Edge points set Ve_k of the live depth map is obtained by back-projection of $\mathbf{C}_k$.

On the other hand, the cuboid edges are discretized into a 3D point set Ve_g^{cu} in the global frame with an interval of 1 mm. Ve_g^{cu} is invariant to the camera pose, and is obtained once the cuboid is successfully localized, prior to the ICP registration procedure. We also set up a KD-tree over Ve_g^{cu} beforehand for fast correspondence search for each point in Ve_k. The edge-to-edge error to minimize is:

$$\mathrm{E}_{edge2edge}\left(\mathrm{T}_{g,\,k}\right) = \sum_{(\mathbf{s},\,\mathbf{t})\in\mathbb{K}_3}\left(\left(\mathrm{T}_{g,\,k}\dot{\mathrm{Ve}}_k(\mathbf{s}) - \mathrm{Ve}_g^{cu}(\mathbf{t})\right)\widehat{\mathrm{N}_{\mathrm{cu}}}_{k-1}^{g}(\mathbf{t})\right)^2, \tag{11}$$

where $\mathbb{K}_3 = \{(\mathbf{s},\ \mathbf{t})\}$ is the correspondence set obtained by nearest neighbor search with KD-tree.

C. Joint Optimization

We combine Equations (7), (9) and (11) to form a joint cost function:

$$\mathrm{E}_{track} = \mathrm{E}_{frame2model} + w_{f2c}\mathrm{E}_{frame2cuboid} + w_{e2e}\mathrm{E}_{edge2edge}, \tag{12}$$

where w_{f2c} and w_{e2e} are the weights that determine the influence of correspondences on the cuboid surfaces and edges. When setting $w_{f2c} = w_{e2e} = 0$, our optimization objective is equivalent to KinectFusion. We set $w_{f2c} = 1$ and $w_{e2e} = 4$ in our experiments empirically, enforcing the constraint of the edge correspondences.

The camera pose $\mathrm{T}_{g,\,k}$ is then obtained by minimizing the overall cost function E_{track} iteratively. A linear approximation is made to solve the system, assuming the orientation change between consecutive frames is very small [1,40]. Using the small angle assumption at each iteration, we approximate the incremental rotation matrix as:

$$\mathrm{R}_{inc} = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix}, \tag{13}$$

where α, β, and γ are the rotation in radians about the X, Y, and Z axis, respectively. Similar to KinectFusion, we compute and sum the linear system in parallel on the GPU, and solve it on the CPU using a Cholesky decomposition.

3.4. Improved Surface Reconstruction

Although we are trying to stabilize camera tracking, surface reconstruction is yet to be perfect. The TSDF volumetric representation allows for online surface extraction as a polygon mesh, while the simple moving average TSDF fusion strategy proposed in KinectFusion suffers from the inflation problem, and lower the reconstruction accuracy. Figure 5 illustrates one of our synthetic datasets "armadillo." Even with noiseless depth images and GT camera trajectory as input, surface reconstruction is smoothed and inflated, particularly at the cuboid edges, the claws and ears of the armadillo, which is far less satisfactory than the GT surface model.

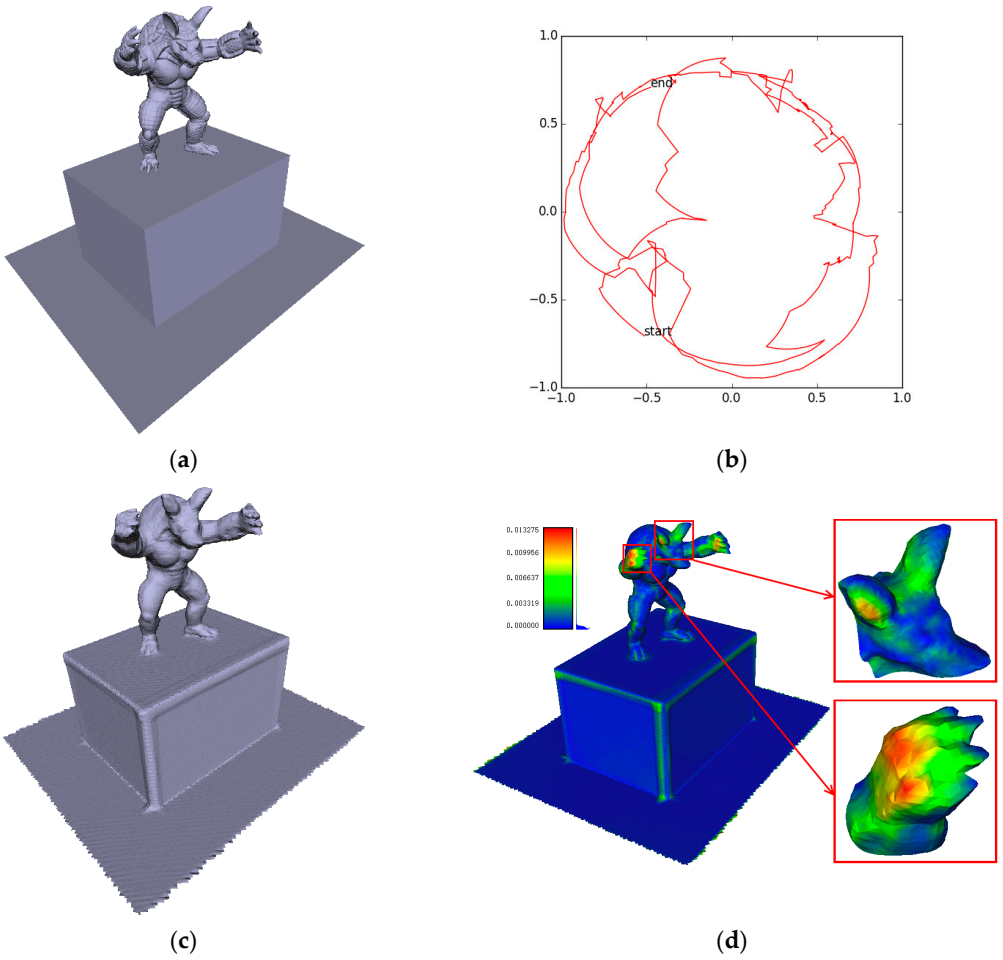


Figure 5. (**a**) Ground-truth (GT) mesh surface model and (**b**) GT camera trajectory which are used for depth image generation; (**c**) Surface reconstruction with GT camera trajectory as input using the simple moving average TSDF fusion strategy; (**d**) A heat map is used to visualize the cloud-to-mesh distances from reconstructed point cloud to the GT mesh. Note the inflation of the cuboid edges, the claws and ears of the armadillo character.

The reason for fusion inflation is illustrated in Figure 6. Due to the simple moving average TSDF fusion algorithm based on the predicate that every surface point is visible from all sensor viewpoints [6], voxel grids with negative TSDF values interfere with the positive ones. To tackle this problem, we extend the storage of TSDF $\mathbf{S}(\mathbf{p})$ from the truncated signed distance value $\mathrm{F}(\mathbf{p})$ and its weight $\mathrm{W}(\mathbf{p})$ to:

$$\mathbf{S}(\mathbf{p}) \mapsto \Big[\mathrm{F}(\mathbf{p}),\ \mathrm{W}(\mathbf{p}),\ \mathrm{F}\prime(\mathbf{p}),\ \mathrm{W}\prime(\mathbf{p}),\ \mathrm{R}_g(\mathbf{p}),\ \mathrm{N}_g(\mathbf{p}),\ \mathrm{Cv}\ (\mathbf{p}),\ \mathrm{Cn}\ (\mathbf{p})\Big], \tag{14}$$

where for each voxel grid $\mathbf{p}$:

1. $[\mathrm{F}(\mathbf{p}),\ \mathrm{W}(\mathbf{p})]$ are the original TSDF components, and $[\mathrm{F}\prime(\mathbf{p}),\ \mathrm{W}\prime(\mathbf{p})]$ are "ghost" distance value and weight for correction of the existing TSDF prediction;
2. $\mathrm{R}_g(\mathbf{p})$ and $\mathrm{N}_g(\mathbf{p})$ are the view ray and the normal vector in the global coordinate frame respectively, which are used to check if a new surface patch is observed from a different view;
3. $\mathrm{Cv}\ (\mathbf{p})$ and $\mathrm{Cn}\ (\mathbf{p})$ are two integer counters as the confidence indices of voxel $\mathbf{p}$ and its normal vector $\mathrm{N}_g(\mathbf{p})$. When $\mathrm{Cv}\ (\mathbf{p}) > \delta_v$, we think the distance value $\mathrm{F}(\mathbf{p})$ of voxel has been robustly estimated; when $\mathrm{Cn}\ (\mathbf{p}) > \delta_n$, the normal vector $\mathrm{N}_g(\mathbf{p})$ is believed to be stable enough against the measurement noise. A simple Boolean semantic function is defined to check if a new face is observed from a new view point:

$$\mathrm{IsNewFace}(\mathbf{p}) = \text{True iff} \begin{cases} \mathrm{Cn}\ (\mathbf{p}) > \delta_n, \text{and} \\ \mathrm{Angle}\left(\mathrm{R}_g(\mathbf{p}), \mathrm{R}_{\mathrm{D}_k}(\mathbf{p})\right) > \theta_r, \text{and}, \\ \mathrm{Angle}\left(\mathrm{N}_g(\mathbf{p}), \mathrm{N}_{\mathrm{D}_k}(\mathbf{p})\right) > \theta_n. \end{cases} \tag{15}$$

where the thresholds are set to $\delta_v = 15$, $\delta_n = 5$, $\theta_r = 15^\circ$, $\theta_n = 30^\circ$ empirically. We define a weight map $\mathcal{W}_k$ for each input frame D_k:

$$\mathcal{W}_k(\mathbf{u}) = cos(\theta_I) * \frac{\mathrm{L}_k(\mathbf{u})}{\mathrm{D}_k(\mathbf{u})}, \tag{16}$$

with $\theta_I = \mathrm{Angle}\big(\mathrm{R}_{\mathrm{D}_k}(\mathbf{p}),\ \mathrm{N}_{\mathrm{D}_k}(\mathbf{p})\big)$ denoting the incidence angle of the view ray to the surface, and L_k is a distance transform map obtained from the contour generator map $\mathbf{C}_k$. For each grid $\mathbf{p}$ in the TSDF volume, we obtain the adaptive fusion weight $\mathrm{W}_k(\mathbf{p})$ and the truncation distance threshold $\mu_k(\mathbf{p})$:

$$\begin{cases} \mathrm{W}_{\mathrm{D}_k}(\mathbf{p}) = \mathrm{W}_{base} * \mathcal{W}_k(\mathbf{u}) \\ \mu_{\mathrm{D}_k}(\mathbf{p}) = \mu_{base} * \mathcal{W}_k(\mathbf{u}) \end{cases}, \tag{17}$$

where $\mathbf{u}$ is the projection of $\mathbf{p}$ given the camera pose $\mathrm{T}_{g,\,k}$, and W_{base}, μ_{base} are the base weight and base truncation distance which are set empirically. Our prediction-corrected TSDF fusion algorithm is then detailed as a flowchart in Figure 7. We categorize the fusion procedure into three sub-strategies:

***Moving Average*:** Identical to the TSDF update procedure of KinectFusion, simple moving average TSDF fusion is performed when a voxel has high uncertainty (e.g., at glancing incidence angle or too close to the depth discontinuity edge):

$$\begin{cases} \mathrm{F}_k(\mathbf{p}) = \mathrm{W}_{k-1}(\mathbf{p})\mathrm{F}_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{D}_k}(\mathbf{p})\mathrm{F}_{\mathrm{D}_k}(\mathbf{p}) \\ \mathrm{W}_k(\mathbf{p}) = \mathrm{W}_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{D}_k}(\mathbf{p}) \end{cases}, \tag{18}$$

***Ignore Current*:** We ignore the TSDF value at the current time when a previously robustly estimated voxel is at glancing incidence angle along the view ray. This is also the case when the current TSDF value with higher uncertainty is observed from a new perspective.

***Fix Prediction*:** When a voxel with previously stable TSDF value $\mathrm{F}_{k-1}(\mathbf{p}) < 0$ is observed to increase from a new point of view—either with $\mathrm{F}_{\mathrm{D}_k}(\mathbf{p}) > 0$ or $\big(\mathrm{F}_{\mathrm{D}_k}(\mathbf{p})\langle 0 \text{ and } \mathrm{F}_{\mathrm{D}_k}(\mathbf{p})\rangle \mathrm{F}_{k-1}(\mathbf{p})\big)$—we believe the live TSDF estimation is more trustworthy as a correction of the previous prediction. In the case of measurement noise, we fuse the live estimation into the ghost storage:

$$\begin{cases} \mathrm{F}'_k(\mathbf{p}) = \mathrm{W}'_{k-1}(\mathbf{p})\mathrm{F}'_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{D}_k}(\mathbf{p})\mathrm{F}_{\mathrm{D}_k}(\mathbf{p}) \\ \mathrm{W}'_k(\mathbf{p}) = \mathrm{W}'_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{D}_k}(\mathbf{p}) \end{cases}, \tag{19}$$

and replace the global TSDF with the ghost storage when $W'_k(\mathbf{p})$ is above a threshold:

$$\begin{cases} F_k(\mathbf{p}) = F'_k(\mathbf{p}) \\ W_k(\mathbf{p}) = W'_k(\mathbf{p}) \end{cases}, \tag{20}$$

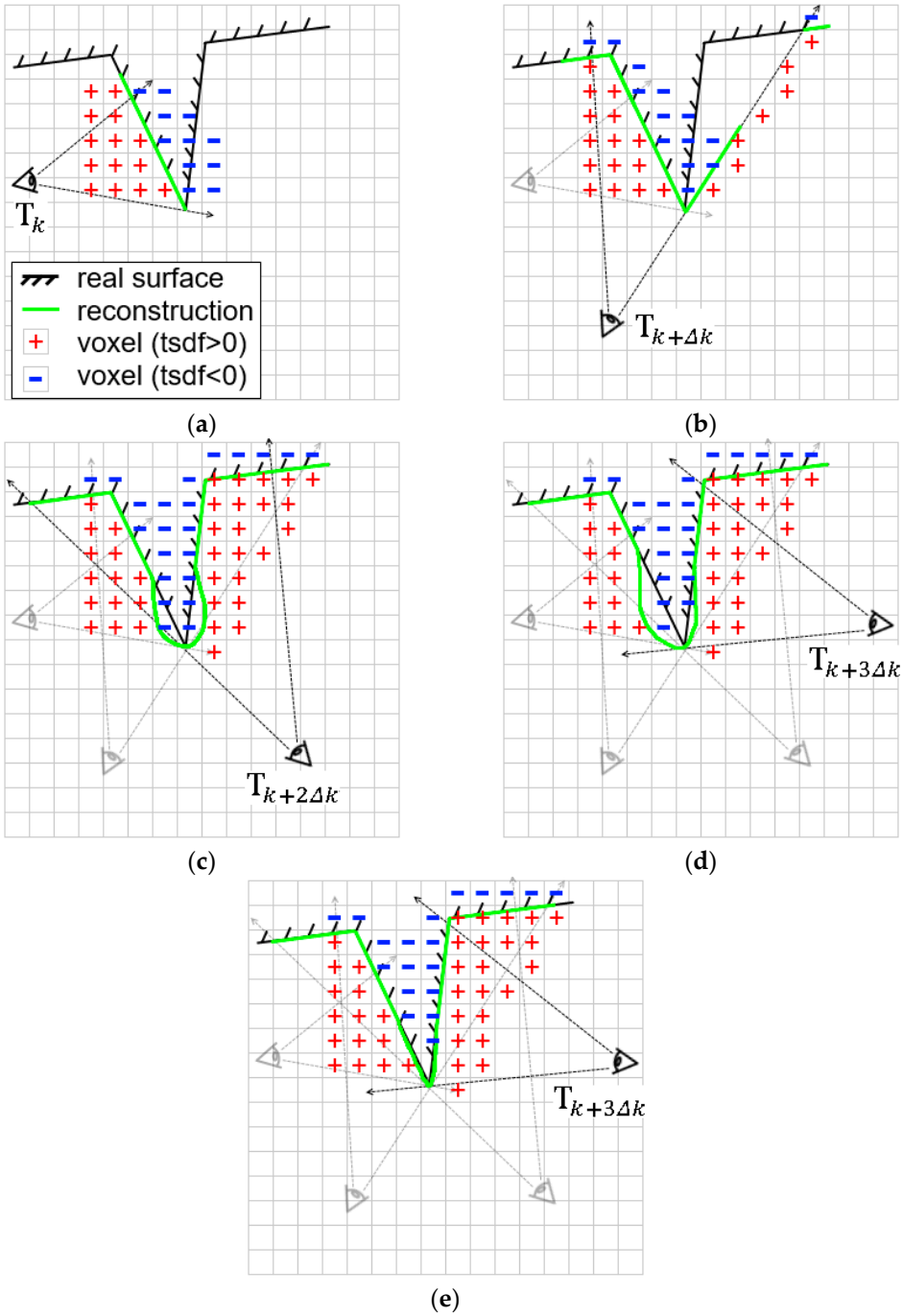


Figure 6. Illustration of the TSDF update process of KinectFusion [1]. (**a**–**d**) denote four different camera poses and update of surface reconstruction (green line) at time k, $(k+\Delta k)$, $(k+2\Delta k)$, $(k+3\Delta k)$ respectively. Note the inflated reconstruction of the highly-convex surface (black line) during the camera movement; (**e**) denotes our reconstruction at time $(k+3\Delta k)$. Compared with (**d**), our result preserves the sharpness of the protrusion area of the surface.

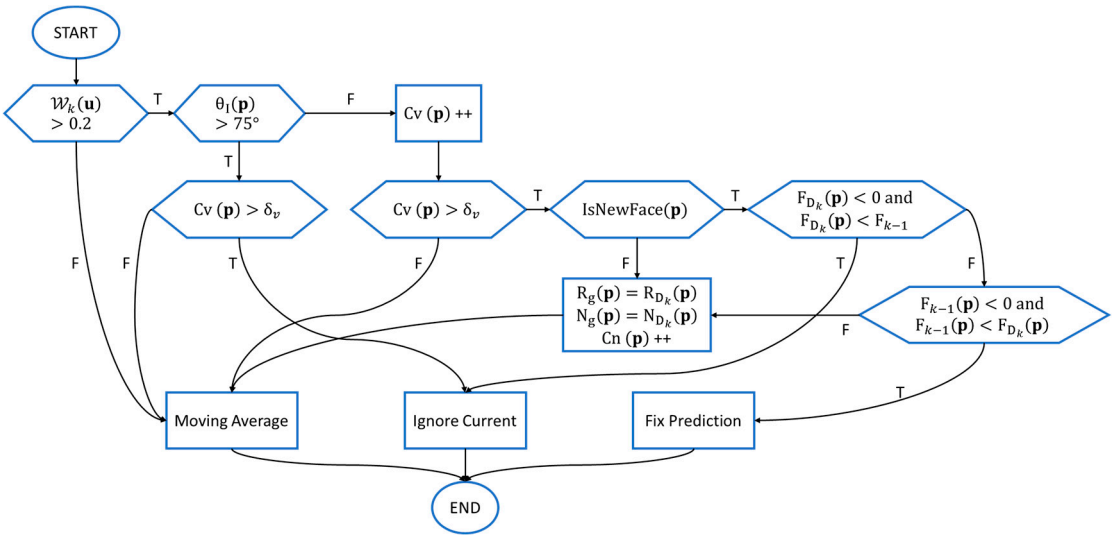


Figure 7. Description of our TSDF fusion algorithm as a flowchart.

Note the update of $\left[\mathrm{R}_g(\mathbf{p}),\ \mathrm{N}_g(\mathbf{p}),\ \mathrm{Cv}(\mathbf{p}),\ \mathrm{Cn}(\mathbf{p})\right]$ is performed independently from the three fusion strategies. With our subdivided fusion algorithm, different surface areas are reconstructed elaborately, resulting in the good preservation of high-curvature surface areas, as illustrated in Figure 6e.

4. Evaluation

We compare our algorithm with four other dense tracking and mapping approaches: KinectFusion [1] (PCL implementation [37]), the work of Zhou et al. [23], ElasticFusion [20] of Whelan et al., and the NICP algorithm [32] of Serafin et al. ElasticFusion jointly aligns RGB and depth information, while the other four are pure depth camera tracking and reconstruction approaches. We set the weight $w_{rgb} = 0$ for RGB alignment component in ElasticFusion, to make it relies only on depth camera tracking as in others' work. On the evaluation of NICP, we run their CPU implementation offline at full resolution, with default configuration (only the camera parameters are updated). We use the point clouds for reconstruction accuracy evaluation.

Since the scales of our scanned objectives are small, we use a volume of size 1 m^3 with 256^3 voxels for all the compared algorithms, where each voxel is approximately 3.9 mm^3.

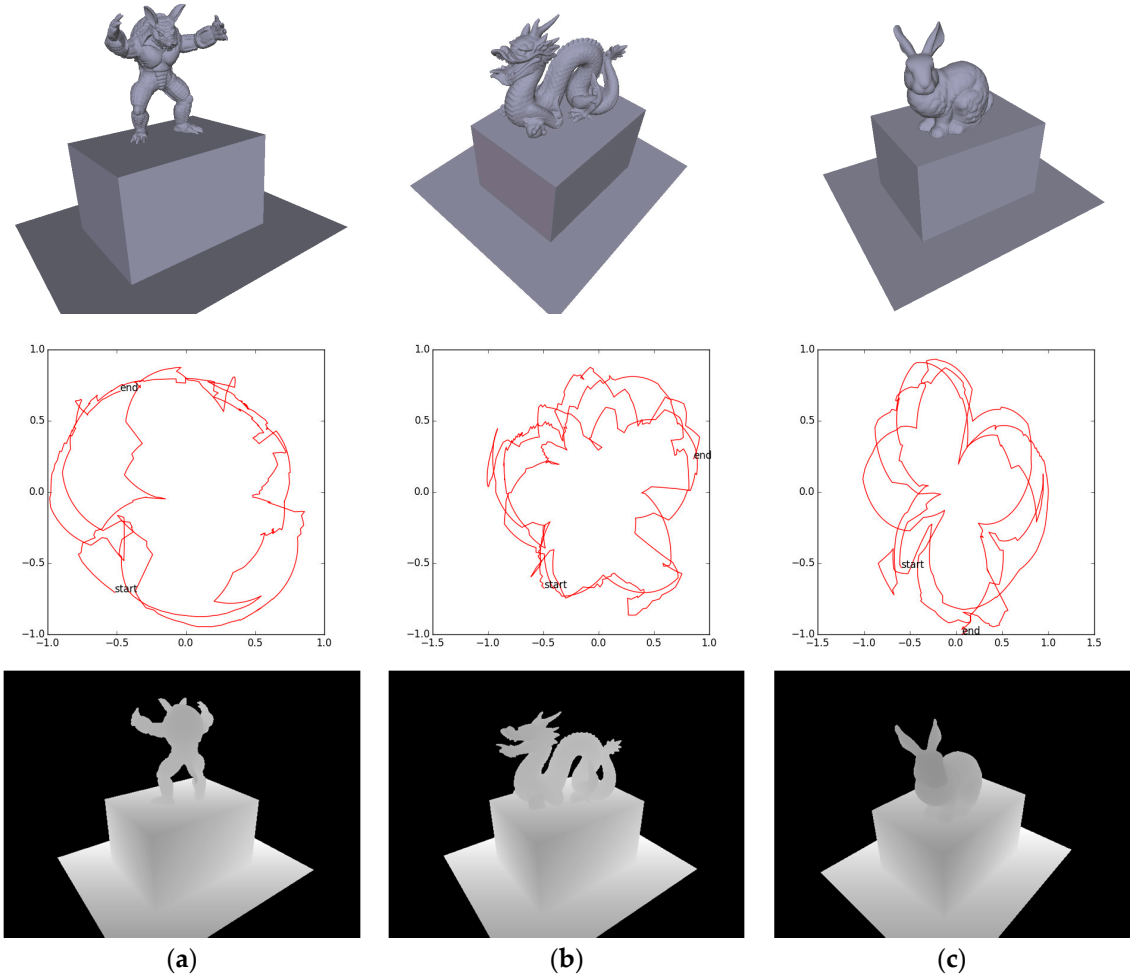
4.1. Dataset

A. Noiseless Synthetic Data

We synthesize three depth image sequences with ground-truth (GT) mesh surface models and GT camera trajectories. A camera intrinsic matrix K_s is given to generate images of resolution 640×480, as shown in Table 1. We choose from "The Stanford Models" [41] the armadillo, dragon and bunny, and scale and place them respectively on top of a synthetic cuboid of edge lengths $\mathcal{P}_{cu} = (400,\ 300,\ 250)$ mm. We then move the camera freely around the scene to generate GT trajectories and depth images, as illustrated in Figure 8. Note that neither the depth measurement noise nor the motion blur is modeled and the only measurement inaccuracy comes from data type casting from floats to integers when saving the depth images.

Table 1. Camera intrinsic parameters used in our dataset, including the focal lengths (f_x, f_y) and the optical center (c_x, c_y). Note that on real-world data the RGB and the depth camera share one intrinsic matrix K_r since they are pre-aligned together.

Scenario	Intrinsic Matrix	f_x	f_y	c_x	c_y
synthetic	K_s (RGB)	525.50	525.50	320.00	240.00
real-world	K_r (RGB-D)	529.22	528.98	313.77	254.10



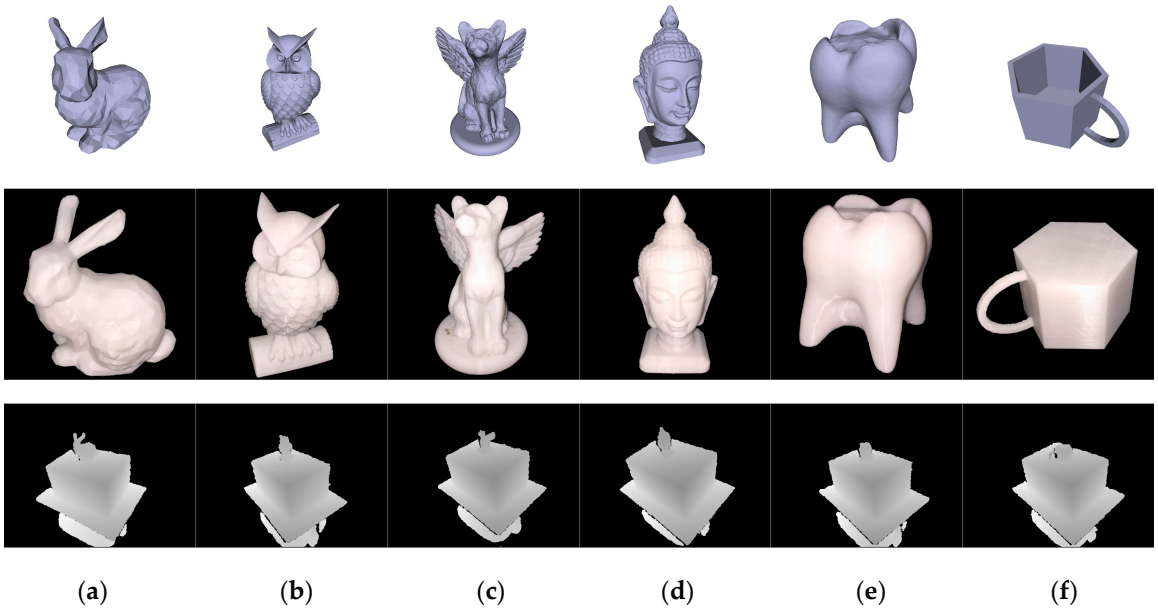
(**a**) (**b**) (**c**)

Figure 8. Synthetic data. The (**a**) armadillo; (**b**) dragon; and (**c**) bunny are set upright on top of a cuboid ($400 \times 300 \times 250$ mm^3) respectively. The top row shows the snapshot of the GT models, the middle row shows the GT camera trajectories (top view), and the bottom row shows the generated depth images (at time 0).

B. Noisy Real-World Data

We manufacture six rigid objects using a 3D printer and put them on a precisely manufactured cuboid with dimensions $400 \times 300 \times 250$ mm^3, same as the one used in our synthetic data. The cuboid is placed on a turntable which is turned by hand, and we held and moved a Kinect camera slowly to perceive more details of the objectives. 640×480 pre-aligned RGB-D images are generated at 30 Hz, with the camera intrinsic matrix K_r (Table 1). We pre-process the depth sequences by truncating depth pixels of values larger than 1.5 m, to remove static background areas. Figure 9 demonstrates our GT mesh models, the 3D printed objects and the captured depth images with the scanning objectives placed on top of the cuboid reference object. Note that in data "lambunny," a simplified bunny model

with merely 640 vertices and 1247 faces is used, and in data "mug," a regular hexagonal mug resting upside down on the cuboid is scanned.



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

Figure 9. Real-world data: (**a**) lambunny; (**b**) owl; (**c**) wingedcat; (**d**) buddhahead; (**e**) tooth; and (**f**) mug. The top row shows the snapshots of the GT models, the middle row shows the 3D printed rigid objects, and the bottom row shows the depth images of models placed on our man-made cuboid resting on a turntable.

4.2. Error Metrics

On synthetic data, both GT camera trajectories and GT mesh surfaces are provided. We quantify the accuracy of camera trajectory using absolute trajectory error (ATE) proposed by Sturm et al. [42], and evaluate the root mean squared error (RMSE) of the translational components over all time indices, which gives more influence to outliers. We further quantify the surface reconstruction accuracy using the cloud to mesh (C2M) distance metric [43] after aligning the GT model with the reconstructed model using the CloudCompare software [44]. We use two standard statistics: Mean and Std. over the C2M distances for all vertices in the reconstruction. On our real-world data, GT camera trajectories are not available nor do we have GT surface models of the entire scenes. We focus on the evaluation of the reconstructed 3D printed models using the C2M error metric.

4.3. Camera Trajectory Accuracy

We evaluate the absolute trajectory error (ATE) of the camera trajectories on synthetic depth image sequences. Although planar surfaces of the cuboid occupy the majority of the depth images, KinectFusion [1], Zhou et al. [23], ElasticFusion [20] and our approach achieve decent camera trajectories without prominently accumulating drift, as listed in Table 2. However, NICP [32] produces inaccurate trajectories with ATE up to hundreds of millimeters. With the additional information from the cuboid a priori, our approach significantly outperforms the reference algorithms, reducing the RMS odometry error from 3~8 mm to less than 2 mm.

Since the errors of all the trajectory estimations (NICP excluded) on synthetic data are small (<10 mm), we plot the per frame ATE (as in Figure 10) for each algorithm (NICP excluded) rather than the trajectory overviews. Our approach (cyan line) keeps the least drift on most of the frames compared with the other three algorithms.

Table 2. Evaluation of the odometry accuracy with absolute trajectory error (ATE) RMSE metric in millimeters.

Synthetic Data	KinectFusion [1]	Zhou et al. [23]	ElasticFusion [20]	NICP [32]	Our Approach
armadillo	3.2	6.4	7.1	454.6	**1.5**
dragon	4.2	6.7	8.0	292.8	**1.7**
bunny	3.9	5.1	6.6	417.8	**1.3**

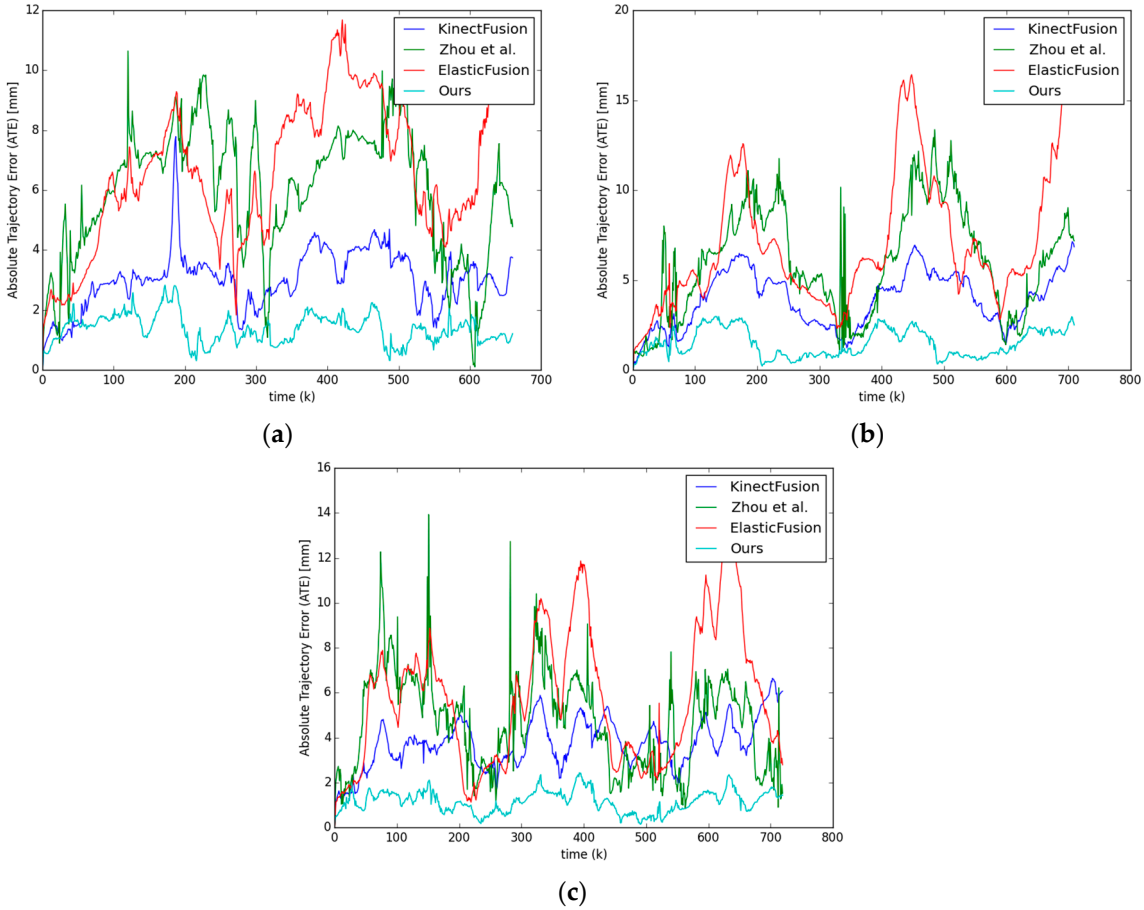


Figure 10. Illustration of per frame ATE on the synthetic (**a**) armadillo; (**b**) dragon; and (**c**) bunny data sequences.

4.4. Surface Reconstruction Accuracy

The surface reconstruction accuracy is evaluated with the cloud to mesh (C2M) distances between the reconstructions and the ground-truth mesh models. For our synthetic data, GT models of the whole scenes are provided while for our real-world data, we have only GT models for the 3D printed objectives placed on the reference cuboid. Surface reconstructions are first aligned against the GT models for C2M distance computation, and heat maps of the C2M distances are plotted in Figure 11 for qualitatively reconstruction accuracy evaluation. Rows 1~3 show the reconstruction of the synthetic data inputs, and rows 4~9 show the real-world ones. The outputs of ElasticFusion in column 3 are not watertight, since it outputs clouds instead of meshes. NICP is excluded from comparison, since its inaccurate camera trajectories result in invalid scene clouds on our benchmark dataset. Note how tightly our approach preserves the scale of the reconstruction and maintains high-fidelity particularly on sharp geometries.

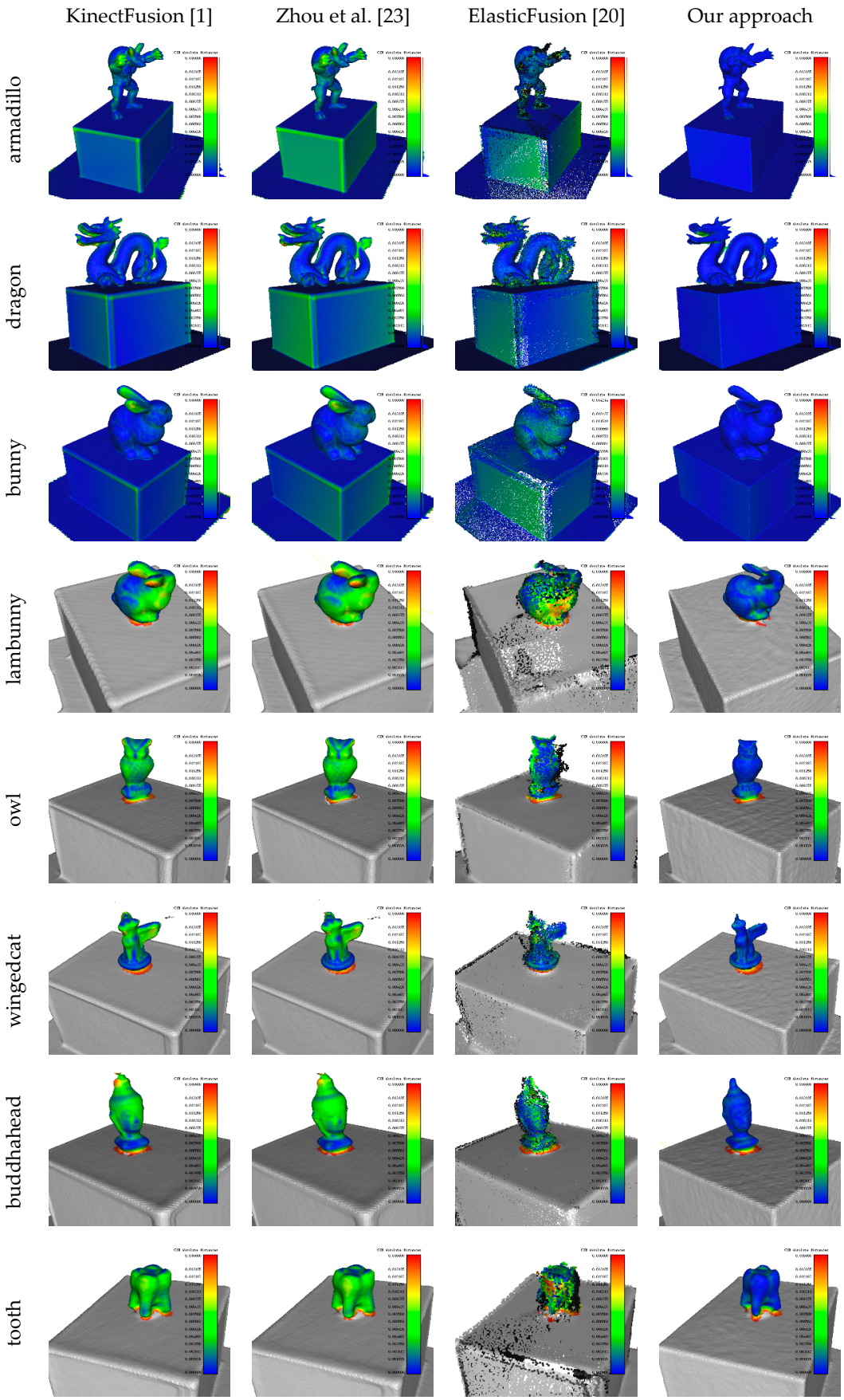


Figure 11. *Cont.*

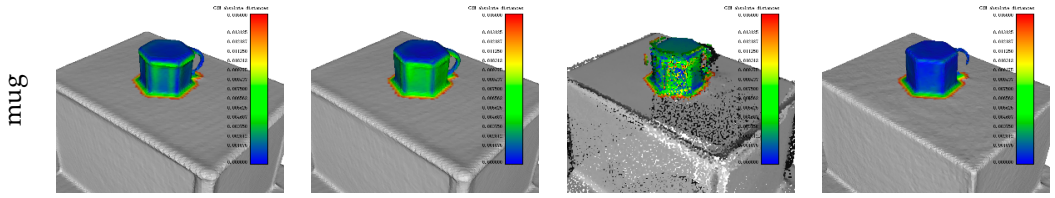


Figure 11. Heat maps of cloud to mesh (C2M) distances for qualitative evaluation of the reconstructions. The compared algorithms are (**a**) KinectFusion [1]; (**b**) Zhou et al. [23]; (**c**) ElasticFusion [20]; and (**d**) our approach. Rows 1~3 are reconstructions of the synthetic data, rows 4~9 are the real-world reconstructions (only the 3D printed objectives are evaluated, with other areas of the scenes grayed out). The scale of the color bar is $0 \sim 15$mm among all the tests.

We quantitatively evaluate the C2M errors for each algorithm with Mean and Std. statistics, as shown in Tables 3 and 4. Our approach keeps the minimum values on both Mean and Std. in all experimental datasets, indicating its superiority in accuracy over the compared algorithms. Close-up views of the reconstructions are detailed in Figure 12, for further comparison between KinectFusion and our approach.

Table 3. Surface Reconstruction accuracy on our synthetic data, with C2M error metric (Mean $\pm$ Std.) in millimeters.

Synthetic Data	KinectFusion [1]	Zhou et al. [23]	ElasticFusion [20]	Our Approach
armadillo	0.9 ± 1.1	1.6 ± 1.4	1.3 ± 1.6	$\mathbf{0.2 \pm 0.5}$
dragon	1.0 ± 1.2	1.5 ± 1.4	1.3 ± 1.6	$\mathbf{0.3 \pm 0.6}$
bunny	0.9 ± 1.9	1.3 ± 1.8	1.0 ± 1.1	$\mathbf{0.4 \pm 1.1}$

Table 4. Surface Reconstruction accuracy on our real-world data, with C2M error metric (Mean $\pm$ Std.) in millimeters. Note that for real-world data, the evaluation is only performed on the 3D printed objectives but not the whole scene.

Real-World Data	KinectFusion [1]	Zhou et al. [23]	ElasticFusion [20]	Our Approach
lambunny	4.0 ± 3.3	4.5 ± 3.0	3.5 ± 3.7	$\mathbf{1.3 \pm 1.5}$
owl	4.4 ± 3.1	4.9 ± 2.9	5.1 ± 4.4	$\mathbf{1.1 \pm 1.3}$
wingedcat	5.0 ± 3.3	5.2 ± 3.1	3.2 ± 3.1	$\mathbf{1.5 \pm 1.8}$
buddhahead	4.8 ± 3.3	5.3 ± 3.0	4.5 ± 3.7	$\mathbf{1.0 \pm 0.8}$
tooth	4.4 ± 1.7	4.8 ± 1.8	3.9 ± 3.6	$\mathbf{1.2 \pm 1.0}$
mug	2.7 ± 2.0	3.5 ± 2.3	5.0 ± 3.2	$\mathbf{0.9 \pm 0.8}$

Additionally, we evaluate the reversed C2M errors, namely the distance from the point clouds of the GT models to the mesh of the surface reconstructions. ElasticFusion is excluded from this comparison, since it produces no surface meshes. Tables 5 and 6 show the quantitative results of this evaluation. On average, the error of this metric is slightly larger than that of the normal C2M distance metric shown in Tables 3 and 4, which results from the inabilities of the compared algorithms to accurately reconstruct extremely sharp surface geometries.

Table 5. Reversed C2M error in millimeters on our synthetic data.

Synthetic Data	KinectFusion [1]	Zhou et al. [23]	Our Approach
armadillo	1.7 ± 1.5	2.0 ± 1.5	$\mathbf{0.5 \pm 0.6}$
dragon	1.8 ± 2.1	2.0 ± 2.2	$\mathbf{0.9 \pm 2.0}$
bunny	1.4 ± 2.3	1.7 ± 2.3	$\mathbf{0.9 \pm 2.3}$

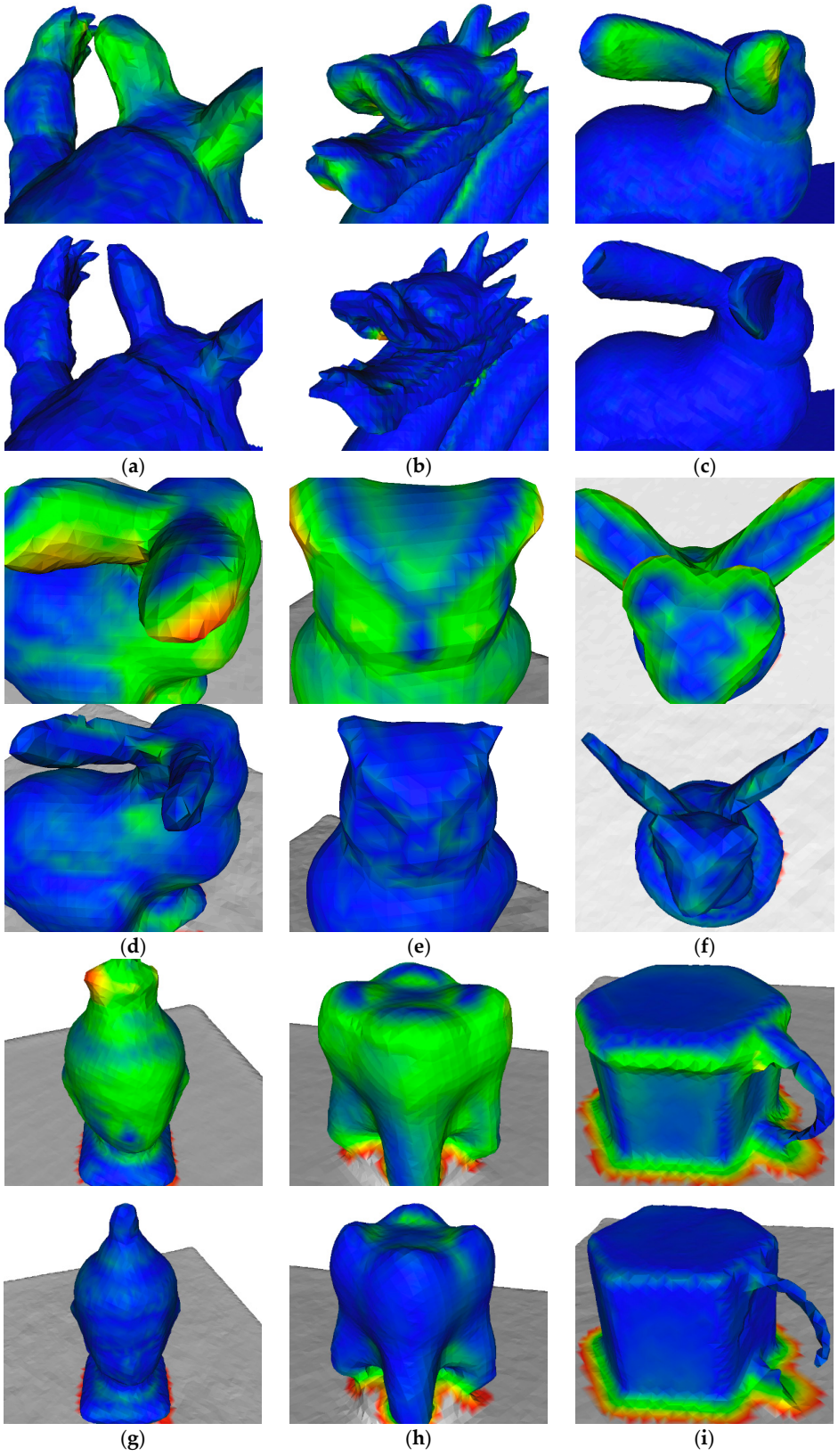


Figure 12. Close-up views of the reconstructions, colored with C2M distances. The synthetic data are (**a**) armadillo; (**b**) dragon; (**c**) bunny; and the real-world data are (**d**) lambunny; (**e**) owl; (**f**) wingedcat; (**g**) buddhahead; (**h**) tooth; and (**i**) mug. The odd rows are reconstructions of KinectFusion [1] as a comparison, and the even rows are of our approach.

Table 6. Reversed C2M error in millimeters on our real-world data.

Real-World Data	KinectFusion [1]	Zhou et al. [23]	Our Approach
lambunny	3.3 ± 2.6	3.9 ± 2.5	$\mathbf{1.2 \pm 1.0}$
owl	4.0 ± 2.1	4.8 ± 2.1	$\mathbf{1.5 \pm 1.6}$
wingedcat	7.6 ± 5.9	8.0 ± 6.0	$\mathbf{4.3 \pm 5.0}$
buddhahead	4.7 ± 1.8	5.6 ± 1.6	$\mathbf{1.1 \pm 0.8}$
tooth	4.1 ± 1.6	4.6 ± 1.7	$\mathbf{1.0 \pm 1.0}$
mug	2.3 ± 1.5	3.1 ± 1.9	$\mathbf{1.1 \pm 1.2}$

5. Discussion and Conclusions

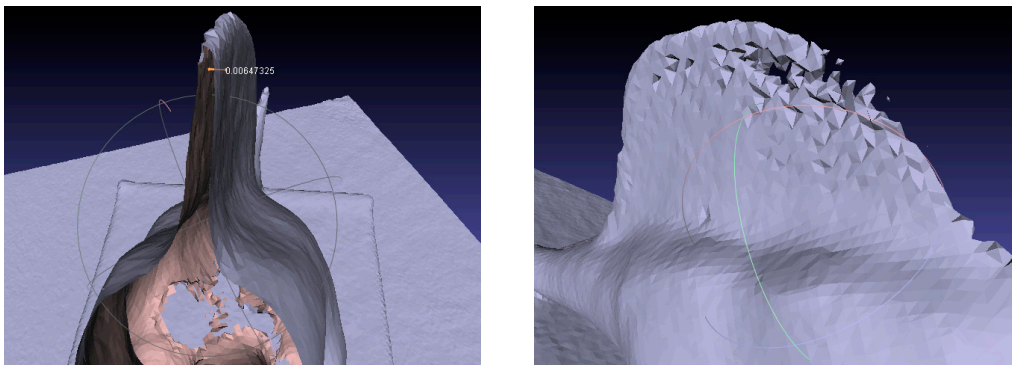
We have presented a novel approach called CuFusion for real-time 3D scanning and accurate surface reconstruction using a Kinect-style depth camera. A man-made cuboid, the scale of which is accurately known, is used as a reference object for accurate camera localization without explicit loop closure detection, and a novel prediction-corrected TSDF fusion strategy is employed for reconstruction update. By solving the surface inflation problem introduced by the simple moving average fusion strategy, our approach preserves the surface details especially when scanning tiny objects or edge areas with high curvatures, resulting in high-fidelity surface reconstruction, which also improves the camera odometry accuracy in turn. We provide a dataset CU3D for the quantitative evaluation of our algorithm and have made our code open-source for scientific verification.

There are several limitations for future work to overcome. First, our modified dense volumetric representation needs 16 bytes per voxel—four times as much memory as KinectFusion at the same resolution—which limits our reconstruction to small-sized scenes. Second, to be capable of reconstructing high-curvature geometries, the camera should be moved as steadily as possible to reduce motion blur and uncertainty in depth measurements. Our algorithm trades off the robustness for reconstruction accuracy, which may fail in the presence of camera jitter or large motion. Figure 13 shows an example of the reconstruction failure result from depth motion blur artifact. Although no noticeable tracking drift happens, the reconstruction is delicate due to our prediction-corrected TSDF fusion strategy. Third, despite our efforts, the reconstructions are yet to be perfected due to sensor noise and the limitation of the volume resolution. As illustrated in Figure 12, engraved surfaces such as the armadillo shell, the facial expression of the owl, wingedcat and buddhahead are smoothed out—additionally, very thin geometries such as the owl's ears and the mug's handle are partly gone.

Our future work will focus on the memory efficiency of our modified volumetric representation, enabling higher volume resolution and a larger scale of reconstruction. The octree-based framework OctoMap [30] could be used for volume data compression. Another interesting challenge might be the surface smoothing problem, which we will focus on mitigating using the surface curvature consistency among the captured frames.

(**a**) (**b**)

Figure 13. *Cont.*



(**c**) (**d**)

Figure 13. A failure case of our approach when scanning a Spinosaurus model with thin spines on its back. (**a**) Color (not used) and (**b**) depth image at time $k = 912$. Slightly faster camera motion around time k leads to mild motion blur, as can be seen from the color image; (**c**) A sectional view of part of the reconstruction before time k. Note how accurately our approach reconstructs the thin spines of the model; (**d**) A profile view of the reconstruction failure of the spine area at time k.

Acknowledgments: We thank Qianyi Zhou [23], Thomas Whelan [20] and Jacopo Serafin [32] for providing their implementations, and thank Guofei Sun for dataset collection.

Author Contributions: Chen Zhang conceived and designed the study, performed the experiments, and analyzed the results; Chen Zhang and Yu Hu prepared the benchmark dataset and wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A.J.; Kohli, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time Dense Surface Mapping and Tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR '11), Basel, Switzerland, 26–29 October 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 127–136.
2. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
3. 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 (SIGGRAPH '96), New Orleans, LA, USA, 4–9 August 1996; ACM: New York, NY, USA, 1996; pp. 303–312.
4. Lorensen, W.E.; Cline, H.E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87), Anaheim, CA, USA, 27–31 July 1987; Volume 21, pp. 163–169. [CrossRef]
5. Rusinkiewicz, S.; Levoy, M. Efficient variants of the ICP algorithm. In Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, 28 May–1 June 2001; pp. 145–152.
6. Hernandez, C.; Vogiatzis, G.; Cipolla, R. Probabilistic visibility for multi-view stereo. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
7. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D Mapping with an RGB-D Camera. *IEEE Trans. Robot.* **2014**, *30*, 177–187. [CrossRef]
8. Axelsson, P. Processing of laser scanner data—Algorithms and applications. *ISPRS J. Photogramm. Remote Sens.* **1999**, *54*, 138–147. [CrossRef]
9. Vosselman, G.; Gorte, B.G.; Sithole, G.; Rabbani, T. Recognising structure in laser scanner point clouds. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *46*, 33–38.

10. Cui, Y.; Schuon, S.; Chan, D.; Thrun, S.; Theobalt, C. 3D shape scanning with a time-of-flight camera. In Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 1173–1180.
11. Lange, R.; Seitz, P. Solid-state time-of-flight range camera. *IEEE J. Quantum Electron.* **2001**, *37*, 390–397. [CrossRef]
12. Scharstein, D.; Szeliski, R. High-accuracy stereo depth maps using structured light. In Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, WI, USA, 16–22 June 2003; Volume 1, pp. I–195–I–202.
13. Henry, P.; Krainin, M.; Herbst, E.; Ren, X.; Fox, D. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Robot. Res.* **2012**, *31*, 647–663. [CrossRef]
14. Segal, A.; Haehnel, D.; Thrun, S. Generalized-ICP. In Proceedings of the Robotics: Science and Systems Conference, Seattle, WA, USA, 28 June–1 July 2009; Volume 2, p. 435.
15. Nistér, D.; Naroditsky, O.; Bergen, J. Visual odometry. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Washington, DC, USA, 27 June–2 July 2004; Volume 1, pp. I–652–I–659.
16. Endres, F.; Hess, J.; Engelhard, N.; Sturm, J.; Cremers, D.; Burgard, W. An evaluation of the RGB-D SLAM system. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN, USA, 14–18 May 2012; pp. 1691–1696.
17. Whelan, T.; Kaess, M.; Fallon, M.; Johannsson, H.; Leonard, J.; McDonald, J. Kintinuous: Spatially Extended Kinectfusion. Available online: https://dspace.mit.edu/handle/1721.1/71756 (accessed on 30 September 2017).
18. Whelan, T.; Kaess, M.; Johannsson, H.; Fallon, M.; Leonard, J.J.; McDonald, J. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *Int. J. Robot. Res.* **2015**, *34*, 598–626. [CrossRef]
19. Kerl, C.; Sturm, J.; Cremers, D. Dense visual SLAM for RGB-D cameras. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 2100–2106.
20. Whelan, T.; Leutenegger, S.; Moreno, R.S.; Glocker, B.; Davison, A. ElasticFusion: Dense SLAM without A Pose Graph. In Proceedings of the 2015 Robotics: Science and Systems, Rome, Italy, 13–17 July 2015.
21. Bose, L.; Richards, A. Fast depth edge detection and edge based RGB-D SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1323–1330.
22. Choi, C.; Trevor, A.J.B.; Christensen, H.I. RGB-D edge detection and edge-based registration. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1568–1575.
23. Zhou, Q.-Y.; Koltun, V. Depth camera tracking with contour cues. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 632–638.
24. Lefloch, D.; Kluge, M.; Sarbolandi, H.; Weyrich, T.; Kolb, A. Comprehensive Use of Curvature for Robust and Accurate Online Surface Reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *PP*, 1. [CrossRef] [PubMed]
25. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
26. Ma, L.; Kerl, C.; Stückler, J.; Cremers, D. Cpa-slam: Consistent plane-model alignment for direct RGB-D Slam. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1285–1291.
27. Nguyen, T.; Reitmayr, G.; Schmalstieg, D. Structural modeling from depth images. *IEEE Trans. Vis. Comput. Graph.* **2015**, *21*, 1230–1240. [CrossRef] [PubMed]
28. Salas-Moreno, R.F.; Glocken, B.; Kelly, P.H.; Davison, A.J. Dense planar SLAM. In Proceedings of the 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 10–12 September 2014; pp. 157–164.
29. Taguchi, Y.; Jian, Y.-D.; Ramalingam, S.; Feng, C. Point-plane SLAM for hand-held 3D sensors. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5182–5189.

30. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robots* **2013**, *34*, 189–206. [CrossRef]
31. Keller, M.; Lefloch, D.; Lambers, M.; Izadi, S.; Weyrich, T.; Kolb, A. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In Proceedings of the 2013 International Conference on 3DTV-Conference, Zurich, Switzerland, 29 June–1 July 2013; pp. 1–8.
32. Serafin, J.; Grisetti, G. NICP: Dense normal based point cloud registration. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 742–749.
33. Rusinkiewicz, S.; Hall-Holt, O.; Levoy, M. Real-time 3D model acquisition. *ACM Trans. Graph.* **2002**, *21*, 438–446. [CrossRef]
34. Weise, T.; Wismer, T.; Leibe, B.; Van Gool, L. In-hand scanning with online loop closure. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), Kyoto, Japan, 27 September–4 October 2009; pp. 1630–1637.
35. Pfister, H.; Zwicker, M.; Van Baar, J.; Gross, M. Surfels: Surface elements as rendering primitives. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000; pp. 335–342.
36. Meister, S.; Izadi, S.; Kohli, P.; Hämmerle, M.; Rother, C.; Kondermann, D. When can we use kinectfusion for ground truth acquisition. In Proceedings of the Workshop on Color-Depth Camera Fusion in Robotics, Algarve, Portugal, 7 October 2012; Volume 2.
37. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4.
38. Feng, C.; Taguchi, Y.; Kamat, V.R. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6218–6225.
39. Khoshelham, K.; Elberink, S.O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* **2012**, *12*, 1437–1454. [CrossRef] [PubMed]
40. Low, K.-L. *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*; University of North Carolina: Chapel Hill, NC, USA, 2004; Volume 4.
41. The Stanford 3D Scanning Repository. Available online: http://graphics.stanford.edu/data/3Dscanrep/ (accessed on 18 July 2017).
42. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580.
43. Handa, A.; Whelan, T.; McDonald, J.B.; Davison, A.J. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014.
44. CloudCompare—Open Source Project. Available online: http://www.danielgm.net/cc/ (accessed on 19 July 2017).