CuFusion2: Accurate and Denoised Volumetric 3D Object Reconstruction Using Depth Cameras

Chen Zhang
College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China;
Corresponding author: Chen Zhang (e-mail: zhangxaochen@163.com).

ABSTRACT 3D object reconstruction from depth image streams using Kinect-style depth cameras has been extensively studied. In this paper, we propose an approach for accurate camera tracking and volumetric dense surface reconstruction assuming a known cuboid reference object is present in the scene. Our contribution is threefold. (a) We maintain drift-free camera pose tracking by incorporating the 3D geometric constraints of the cuboid reference object into the image registration process. (b) We reformulate the problem of depth stream fusion as a binary classification problem, enabling high-fidelity surface reconstruction, especially in the concave zones of objects. (c) We further present a surface denoising strategy to mitigate the topological inconsistency (e.g., holes and dangling triangles), which facilitates the generation of a noise-free triangle mesh. We extend our public dataset CU3D with several new image sequences, test our algorithm on these sequences and quantitatively compare them with other state-of-the-art algorithms. Both our dataset and our algorithm are available as open-source content at https://github.com/zhangxaochen/CuFusion for other researchers to reproduce and verify our results.

INDEX TERMS 3D object reconstruction, depth cameras, Kinect sensors; open source, signal denoising, SLAM

I. INTRODUCTION

Reconstructing a 3D surface model from a sequence of provided range images has been an active research topic during the last decade. In recent years, the emergence of depth cameras based on either 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 at a high frame rate as video streams. The KinectFusion algorithm [1] introduced by Newcombe *et al.* is one of the seminal works for real-time camera tracking and dense environment reconstruction, turning depth sensors into consumer-grade 3D scanners. It uses fast iterative closest point (ICP) algorithms [2], [3] for camera pose estimation and a volume known as the truncated signed distance function (TSDF) for scene representation. The connected mesh surfaces are later extracted using the marching cubes algorithm [4].

On the reconstruction accuracy, however, the KinectFusion algorithm suffers from a number of limitations, including the ICP image registration algorithm that is prone to accumulating drift in the presence of structure-less surfaces, the inability to recover from drift, and the problem of surface deformation for highly curved and concave zones of the scanned objects [5].

Many researchers have been working on solving these problems. ICP variants such as point-to-plane ICP [6] and generalized ICP (GICP) [7] have been proposed for better image alignment. Loop closures have been detected, and pose graphs have been built and optimized online [8]–[12] or offline [13], [14] to produce robust and globally consistent maps. To address the surface deformation problem, Whelan *et al.* proposed the ElasticFusion framework [10], [15] to activate nonrigid model-to-model refinement, which also relies on local loop closure detection. Slavcheva *et al.* proposed the SDF-2-SDF algorithm [16], [17], which focuses on small-scale object reconstruction. Similarly, we also proposed a CuFusion framework [18] for accurate camera localization and object modeling under the assumption that a known cuboid reference object is present in the scene. A prediction-corrected TSDF fusion strategy is applied instead of a simple moving average fusion to resolve the surface deformation problem.

However, the reconstruction quality of [18] relies heavily on the quality of the raw depth measurements. To maintain the reconstruction fidelity in particularly highly curved zones of objects, the input depth images should contain as little motion blur as possible, which requires the camera to orbit steadily. Even mild blurring in depth measurements caused by slightly faster camera motion may lead to reconstruction failure on the sharp edges of objects. To address these problems, in this paper, we introduce a novel approach that is an extension of our previous work [18]. The major contributions are as follows.

- We propose an ICP variant that takes the constraints of the known reference object into account for robust and accurate camera pose estimation. With the supplementary information of the reference object, we maintain near-optimal camera tracking for each frame, making it possible for accurate object reconstruction. We build pose graphs and solve for optimized camera poses and compare them with those without graph optimization, demonstrating that our method is accurate enough.
- We reformulate the data fusion task as a per voxel binary classification task to maintain the reconstruction fidelity and resistance to motion blur result from camera jitters.
- We present a denoising strategy, which performs noise reduction directly on the generated volume during scanning, resulting in cleaner mesh surface outputs, taking it one step further for industrial applicability of the output surface models such as in 3D printing.

Compared with most existing simultaneous localization and mapping (SLAM)-like algorithms, our method focuses on geometric fidelity, taking depth streams as the only input to generate topologically consistent mesh models. We perform qualitative and quantitative evaluations on reconstructions from both synthetic and real-world sequences of the CU3D dataset. Both the camera trajectory and the reconstruction accuracy are compared with state-of-the-art approaches. We show the fidelity of the reconstruction of our method and release our code and dataset to the community for future work.

II. RELATED WORK

The SLAM problem has been extensively studied. Monocular RGB camera tracking systems such as monoSLAM [19] and parallel tracking and mapping (PTAM) [20] allow users to obtain camera trajectories and sparse point cloud models. Dense reconstruction systems [21]–[23] have also been proposed to replace point cloud-based systems. With the advent of Kinect-style active depth sensors, the KinectFusion [1] algorithm permits dense volumetric reconstruction of the scene in real time, enabling mesh model output for physics-based augmented reality (AR) [24] and 3D printing [25]. Improved frameworks have then been proposed in the aspects of memory efficiency [26]–[28], large space representation [8], [11], [27], [29], camera trajectory accuracy with loop closure detection and optimization [12], [27], [30], and scene representation such as surfels [31] or hybrid data structure [32].

Some researchers have used structural priors for accurate camera localization. Zhou *et al*. [33] introduced an approach for robust contour cue extraction and integrated the contour constraints into the registration objective and stabilizing camera tracking in challenging scenarios. High-level features such as planes [34]–[38] and objects [39], [40] have also been used as primitives to provide more constraints to camera pose estimation. However, such systems may fail when those priors do not present in the scene. In this paper, we make full use of the information of a precisely human-made cuboid reference object, namely, the orthogonal or parallel planar facets and the contour cues of certainly known lengths as constraints for camera localization stabilization. The accurate and robust camera trajectories are later used in the model generation process to integrate single images into consistent models in the global coordinate.

In addition to pairwise local registration methods for camera localization, many recent works [13], [41]–[43] have been devoted to registering point clouds globally, which does not rely on good camera pose initialization. As a representative work, fast global registration (FGR) [42] uses fast point feature histograms (FPFH) [44] to find feature correspondences, filters them in a “mutual best” strategy to reduce false matches, and defines an objective function that minimizes the distances between the corresponding points, leading to globally consistent camera pose estimation without local refinement. Deep learning-based algorithms such as 3DMatch [45] and point pair feature net (PPFNet) [46] have also been proposed for direct global registration. [45] used dense local grids and 3D convolutional neural networks (CNNs) to learn the local patch descriptor, and [46] further employed raw points and normals to improve the descriptor representation. These global registration algorithms, however, focus mostly on the camera pose estimation rather than the quality of the final reconstruction. To be more specific, balancing the manifoldness and sharpness of the reconstructed surface models remains a challenge to us.

Different dense scene representations have also been explored in the literature. Occupancy mapping using a grid of cells to represent the space has been popular in robotics. A probability of occupancy in each cell is accumulated via Bayesian updates every time a new informative observation is provided [47]. Similarly, the SDF volumetric representation introduced in [48] is often used in graphics to fuse partial depth scans into one global model. The SDF represents the surface interfaces implicitly as zeros, and the mesh models can be extracted using the marching cubes type algorithm [4]. Instead of volumes, surfels [10], [31], [49], [50] are also exploited to represent the scene, which renders the scene with the surface-splatting technique [51] and reduces the computational complexity and memory overhead compared with the volumetric approaches. The volumetric representation has been reported to be difficult to resolve the highly curved and concave details, such as the folds in the garment [5] or thin geometries [18], even if the voxels are small enough. [18] introduced a prediction-corrected data fusion strategy for geometry detail preservation. By storing surface normal and view ray vectors per voxel as additional information, it enables fast correction of the surface where deformation is previously accumulated. The main problem of [18] resides in its inability to deal with motion blur in raw depth measurements as well as its high memory consumption. Different from either conventional volumetric or surfel-based approaches, we propose a novel method for data fusion with a volumetric representation. We convert the typical moving average to a scheme of per voxel probabilistic binary classification dedicated to the reconstruction fidelity, especially in sharp geometries.

There have been many popular RGB-D datasets created for the evaluation of indoor 3D reconstruction. The TUM RGB-D dataset [52] offers a set of RGB-D images with accurate and time-synchronized ground-truth camera poses from a motion capture system. It aims mainly at trajectory estimation and lacks ground-truth scene models. To assess the accuracy of the scene reconstruction, the ICL-NUIM dataset [53] generates both ground-truth poses and models for quantitative evaluation with two synthetic scenes. Slavcheva *et al.* provided the first object dataset with ground-truth computer-aided design (CAD) models and camera trajectories. The dataset consists of a 3D-printed selection of small objects scanned with a markerboard placed below them.

Similarly, we provide a dataset CU3D in [18] with both synthetic and real-world sequences. Our synthetic data provide both ground-truth object models and camera poses, and the real-world data are generated by scanning six 3D-printed objects with only ground-truth models available. In this work, we extend this dataset with several supplementary sequences. Different from the previous real-world scans in our CU3D dataset, we have no per-vertex ground-truth for these newly added scans. We verify the reconstruction accuracy by evaluating the total length of the reconstructed model compared with Vernier caliper measurements as the ground-truth.

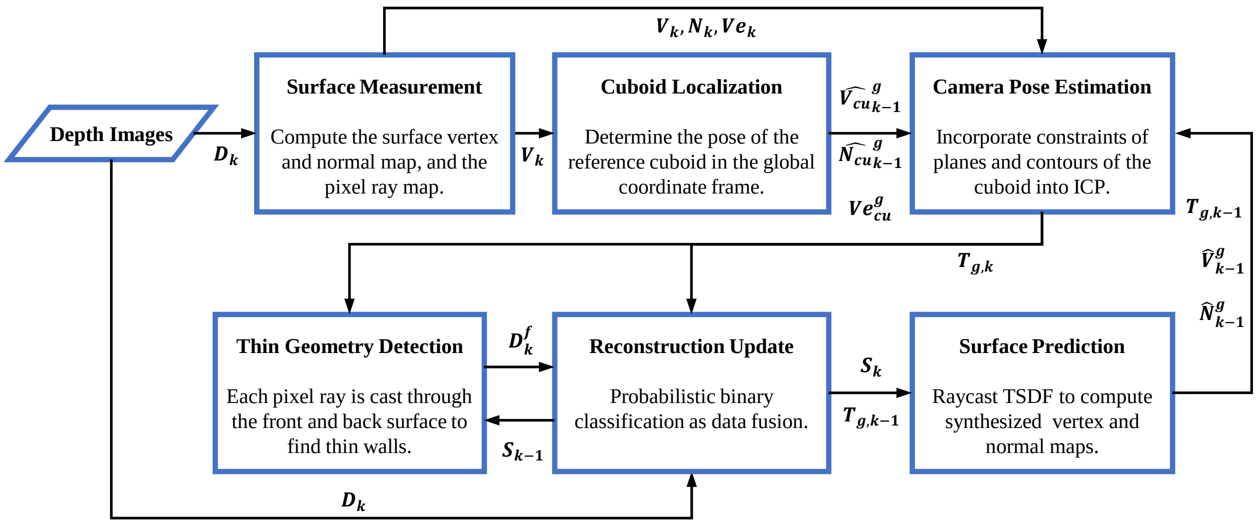

Depth Images
D_k
Surface Measurement
Compute the surface vertex and normal map, and the pixel ray map.
V_k
Cuboid Localization
Determine the pose of the reference cuboid in the global coordinate frame.
V_k, N_k, Ve_k
$V_{cu\,k-1}^{g}$
$N_{cu\,k-1}^{g}$
Ve_{cu}^{g}
Camera Pose Estimation
Incorporate constraints of planes and contours of the cuboid into ICP.
$T_{g,k-1}$
$\hat{V}_{k-1}^{g}$
$\hat{N}_{k-1}^{g}$
$T_{g,k}$
Thin Geometry Detection
Each pixel ray is cast through the front and back surface to find thin walls.
D_k^f
S_{k-1}
Reconstruction Update
Probabilistic binary classification as data fusion.
S_k
$T_{g,k-1}$
Surface Prediction
Raycast TSDF to compute synthesized vertex and normal maps.
D_k

Figure 1. An overview of our system pipeline. Details of the notations are described in section III.

III. METHOD

We base our work on the open-sourced implementation of the KinectFusion algorithm from the point cloud library (PCL) [54]. Our reconstruction pipeline is illustrated in Fig. 1, which is described in detail in the following subsections.

A. MATHEMATICAL NOTATION

We define the image domain as $\Omega \subset \mathbb{N}^2$, and a depth image at time k is defined as $D_k : \Omega \to \mathbb{R}$, where each single pixel $p \in \Omega$ in the image stores the distance from the camera to the surface. Assuming that the camera intrinsic matrix is known, we define the projection and dehomogenization function $\pi : p = \pi(P)$ to map a 3D point $P = (x, y, z)^T \in \mathbb{R}^3$ in the camera coordinate to a pixel $p = (u, v)^T \in \mathbb{N}^2$ in the image plane. We present the 6-degrees-of-freedom (6DOF) camera pose at time k in the global coordinate frame by a rigid transformation matrix:

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

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 $P_k \in \mathbb{R}^3$ in the camera coordinate frame to a global point $P_g = R_{g,k}P_k + t_{g,k} \in \mathbb{R}^3$. For simplicity, we omit the conversion between the 3-vectors and their corresponding homogeneous 4-vectors. A depth pixel p can be back-projected to the global coordinate frame: $P_g = T_{g,k}\pi^{-1}(p, D_k(p))$. An organized vertex map V_k is computed by bilateral filtering and back-projecting the raw depth image D_k, and its corresponding normal map N_k is computed using the principal component analysis (PCA) method.

B. CUBOID LOCALIZATION AS INITIALIZATION

Given a depth image D_k and the reference cuboid with edge lengths $L_{cu} = (a, b, c)$, we localize the cuboid; namely, we calculate its pose in the global coordinate frame. Live depth frames will be later aligned against the cuboid when scanning around it to mitigate the accumulating camera drift.

We first perform plane segmentation using the agglomerative hierarchical clustering (AHC) algorithm [55]. 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., $|\Theta_p - 90°| < \varepsilon_\Theta$). Once we find three planes that are orthogonal to each other, we check the length of the intersecting line segments between them. If the three line segments' length parameter L approximately matches the cuboid edge length parameter L_{cu} (i.e., $|L - L_{cu}| < \varepsilon_L$), we claim to find the cuboid and mark the three planes as its adjacent planes. Empirically, we set $\varepsilon_\Theta = 5°, \varepsilon_L = 10\,mm$.

We consequently define the cuboid coordinate frame of reference. We set the 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 intrinsic camera calibration, orthogonality between the normal vectors of the segmented adjacent planes is not strictly guaranteed. We obtain the nearest orthogonal axes $[X_{cu}, Y_{cu}, Z_{cu}]$ of the frame by solving the orthogonal Procrustes problem, where $[X_{cu}, Y_{cu}, Z_{cu}]$ are 3×1 column vectors. The cuboid pose in the camera frame at time k is:

$$T_{k,\,cu} = \begin{bmatrix} R_{k,\,cu} & t_{k,\,cu} \\ 0^T & 1 \end{bmatrix} \in \mathbb{SE}(3) \tag{2}$$

$$R_{k,\,cu} = [X_{cu}, Y_{cu}, Z_{cu}] \tag{3}$$

$$t_{k,\,cu} = O_{cu}^T \tag{4}$$

Assuming the camera pose $T_{g,\,k}$ at time k is known, the cuboid pose $T_{g,\,cu}$ in the global frame of coordinate could then be derived: $T_{g,\,cu} = T_{g,\,k}\, T_{k,\,cu}$. Fig. 2 illustrates the notations used in the paper.

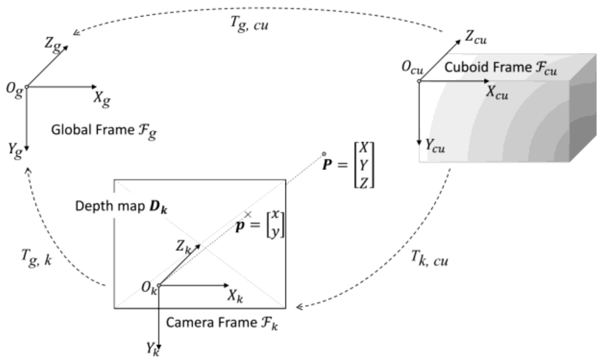

$T_{g,\,cu}$
Z_g
O_g
X_g
Global Frame $\mathcal{F}_g$
Y_g
Z_{cu}
O_{cu}
Cuboid Frame $\mathcal{F}_{cu}$
X_{cu}
Y_{cu}
$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
Depth map D_k
$p = \begin{bmatrix} x \\ y \end{bmatrix}$
Z_k
$T_{g,\,k}$
$T_{k,\,cu}$
O_k
X_k
Camera Frame $\mathcal{F}_k$
Y_k

Figure 2. Illustration of the notations used in this paper.

Note that the cuboid pose in the global frame is evaluated only once when we find a triplet of orthogonal planes and left unchanged afterwards. Each incoming depth image is aligned against the virtual cuboid to reduce camera drift, as described in the following sections.

C. 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 current camera pose $T_{g,\,k}$ by registering the live depth map to both the globally reconstructed surface model and the reference cuboid.

1) FRAME-TO-MODEL REGISTRATION

Given the implicit TSDF surface model S and the previously estimated camera pose $T_{g,\,k\text{-}1}$ at time $k-1$, an organized vertex and normal map $(\hat{V}_{k-1}, \hat{N}_{k-1})$ can be obtained via per-pixel raycast and then transformed into the global frame as $(\hat{V}_{k-1}^g, \hat{N}_{k-1}^g)$. For frame-to-model (*f2m*) registration, a transformation $T_{g,\,k}$ is pursued to minimize the point-to-plane error between $T_{g,\,k}V_k$ and $\hat{V}_{k-1}^g$:

$$E_{f2m}\left(T_{g,\,k}\right) = \textstyle\sum_{(p,\hat{p})\in\mathbb{K}_1}\Big(\Big(T_{g,\,k}V_k(p) - \hat{V}_{k-1}^g(\hat{p})\Big)\,\hat{N}_{k-1}^g(\hat{p})\Big)^2 \tag{5}$$

where $\mathbb{K}_1 = \{(p, \hat{p})\}$ is the set of correspondences associated with projective data association [1]:

$$\hat{p} = \pi\left(\tilde{T}_{k-1,k}V_k(p)\right) \tag{6}$$

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

2) FRAME-TO-CUBOID REGISTRATION

Assuming the cuboid pose has previously been initialized, for each camera pose $T_{g,\,k\text{-}1}$, per-pixel ray casting is performed on the reference cuboid to synthesize a proxy depth map $\hat{D}_{k-1}^{cu}$. An organized vertex and normal map in the global frame $(\widehat{Vcu}_{k-1}^{\,g}, \widehat{Ncu}_{k-1}^{\,g})$ is then derived by back-projecting the proxy depth map and transforming local maps to the global space. Similar to the frame-to-model registration, the distance between the current depth measurement and the cuboid surface (frame-to-cuboid, i.e., *f2c*) is minimized:

$$E_{f2c}\left(T_{g,\,k}\right) = \textstyle\sum_{(p,\hat{p})\in\mathbb{K}_2}\Big(\Big(T_{g,\,k}V_k(p) - \widehat{Vcu}_{k-1}^{\,g}(\hat{p})\Big)\,\widehat{Ncu}_{k-1}^{\,g}(\hat{p})\Big)^2 \tag{7}$$

In addition, we exploit the edge-to-contour (*e2c*) distance as a constraint term to mitigate the potential camera drift. Contours of the reference cuboid can be discretized into a 3D point set Ve_{cu}^g in the global frame with an interval of 1 mm once the cuboid is successfully localized. Given the inpainted depth map D_k', we find the edge pixel set C_k at depth discontinuities in the live depth map, as proposed in [33]. The 3D edge point set Ve_k could then be derived by back-projection of C_k. The edge-to-contour error to minimize is:

$$E_{e2c}(T_{g,k}) = \sum_{(s,t)\in\mathbb{K}_3}\left(\left(T_{g,k}Ve_k(s) - Ve_{cu}^{g}(t)\right)\widehat{N_{cu}}_{k-1}^{g}(t)\right)^2 \quad (8)$$

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

3) JOINT OPTIMIZATION

We combine (5), (7), and (8) to form a joint cost function:

$$E_{track} = E_{f2m} + w_{f2c}E_{f2c} + w_{e2c}E_{e2c} \quad (9)$$

where w_{f2c} and w_{e2c} are the weights that determine the influence of correspondences on the cuboid surfaces and contours. When setting $w_{f2c} = w_{e2c} = 0$, our optimization objective is equivalent to KinectFusion. We empirically set $w_{f2c} = 4$ and $w_{e2c} = 24$ in our experiments, enforcing the constraints of the contour correspondences. Note that the two weights are set larger because the correspondence sets $\mathbb{K}_2$ and $\mathbb{K}_3$ are much smaller than $\mathbb{K}_1$. We compute the camera pose $T_{g,k}$ by iteratively minimizing the linear approximation [6] of the overall cost function E_{track}.

D. POSE GRAPH OPTIMIZATION

We build pose graphs for optimization based on the geometric characteristics of the reference cuboid. Each time a triplet of mutually orthogonal planes of the cuboid is observed in the global frame, the orthogonal normal vectors span the space $\mathbb{R}^3$, during which strong geometric constraints ensure accurate camera pose estimation. We select a keyframe as a vertex of the pose graph each time a new trihedron enters the camera's field of view and the relative transformation from the interframe alignment as its edges. We optimize the pose graph using the open-source framework "g2o" [56] and compare our camera trajectories with the optimized trajectories, showing that our camera poses are accurate enough even without graph optimization.

E. SURFACE RECONSTRUCTION

In this paper, we introduce a method that is an improvement to [18]. We detect sharp or thin geometric zones by raycasting through the negative TSDF area. Similar to [18], we substitute the simple moving average data fusion strategy with a non-uniform strategy. We turn the problem of data fusion into a probabilistic binary classification problem and adopt a denoising scheme, leading to more accurate and cleaner mesh models than other methods.

1) TRUNCATED SIGNED DISTANCE FUNCTION

The signed distance function (SDF) $F : \mathbb{R}^3 \to \mathbb{R}$ introduced in [48] represents the scene non-parametrically. Two components are stored at each location of volume S: the SDF value F and a weight W:

$$S \mapsto [F, W] \quad (10)$$

Each SDF value in the voxel corresponds to the signed distance from the cell to the closest surface. In most volumetric reconstruction systems, the projective SDF is computed along the optical axis, which is view dependent. Instead, we multiply the projective SDF by the cosine value of the incidence angle of each view ray for an approximation of the real SDF value.

The TSDF is obtained by normalizing and truncating the SDF value with a constant truncation distance δ, which is usually set empirically. When we set it large, the reconstruction is more noise resistant, whereas the surface details are lost, and when it is set small, the case is the opposite. Based on the observation that depth measurements near image edges are highly uncertain, we choose an adaptive truncation distance δ_a to ensure the fineness of the details of the surface reconstruction. Given a 3D point P in the global frame, its TSDF is computed as follows:

$$F_{proj}(P) = D(p) - P_z \quad (11)$$

$$p = \pi\left(T_{g,k}^{-1}P\right) \quad (12)$$

$$\eta = F_{proj}(P) * cos(\theta_{in}) \quad (13)$$

$$F(P) = \begin{cases} min(1, \eta/\delta) & iff\ \eta \geq -\delta_a \\ null & otherwise \end{cases} \quad (14)$$

$$\delta_a = max(\sigma, min(1, \lambda/\Lambda)) * \delta \quad (15)$$

$$\lambda = dist(p) * P_z/f \quad (16)$$

where P_z is the depth of point P in the camera frame, η is the approximation of the real SDF value, θ_{in} is the incidence angle of the view ray, and f is the camera focal length. The $dist$ function performs edge detection and distance transformation to determine the distance from a pixel to its nearest edge. The pixel distance is converted to a physical distance λ and normalized by a constant physical length Λ. When $\lambda < \Lambda$, the pixel is near the depth edge, and the truncation distance δ_a is adapted to a smaller value. We empirically set $\sigma = 0.3$ and $\Lambda = 30\ mm$.

2) RAYCAST FOR THIN GEOMETRY DETECTION

Data fusion around thin geometric zones may be problematic. To detect thin areas efficiently, we propose a "cast through" strategy, checking whether zero-crossing is found twice when performing per-pixel raycast. Different from the raycast procedure in KinectFusion, our ray continues after a $+\varepsilon$ to $-\varepsilon$ zero-crossing is detected and does not stop until a $-\varepsilon$ to $+\varepsilon$ back face, or a $-\varepsilon$ to a void cell of zero weight is found, or finally when exiting the working volume. The latter two cases indicate that no thin geometry is met temporarily along the current ray.

Our raycast process outputs a "fake" depth map D^f consisting of both positive and negative values. The sign of a value indicates whether a ray finally intersects the front or back surface, and the absolute depth value shows the distance between the camera and the final intersection point along the principal optical axis.

Moreover, our modified raycast for thin geometry detection does not continue straight all the time. When a pixel ray intersects the surface at a grazing angle (empirically $\theta_{in} > 60°$), ray refraction is performed. The primary reason for this is that when the ray is nearly parallel to the surface, erroneous zero-crossings are often detected. Additionally, when a ray penetrates a thin area, it usually passes through the internal space faster if it bends towards the negative normal direction, leading to more accurate measurements of the negative pixels in D^f. The ray refraction is performed by weighted averaging the original pixel ray and the negative surface normal vector at the incident point. Let $L_g(p)$ denote the view ray along pixel p and $N_g(p)$ the surface normal at pixel p in the global frame. The refracted view ray $L'_g(p)$ is updated by:

$$L'_g(p) = (1 - \rho_r)L_g(p) - \rho_r N_g(p) \qquad (17)$$

where we set $\rho_r = 1/3$ empirically. Fig. 3 illustrates our modified raycast strategy at different incidence angles.

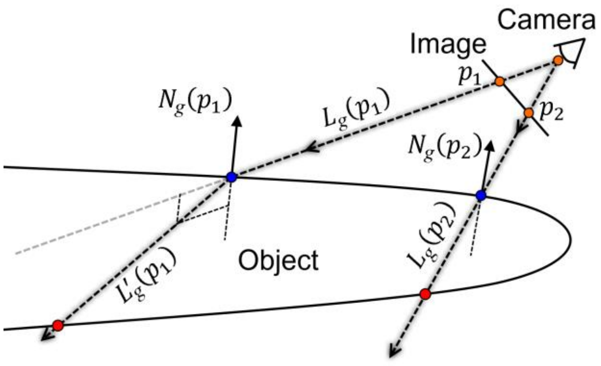


Figure 3. Our modified raycast strategy for thin geometry detection. Note that the ray along pixel p_1 is refracted closer to the surface normal at the incident point on the surface due to the large incidence angle, which makes it more quickly penetrate the object. The ray along pixel p_2 continues straight without refraction.

3) DATA FUSION AS CLASSIFICATION

A voxel grid located around a thin geometric zone may be seen from opposite perspectives, resulting in entirely different TSDF measurements. For example, in Fig. 4, voxel P is found at the back and far away from the surface when viewed from the left with a negative TSDF value F_1 representing the occupied space of large magnitude. When the camera orbits to the right side, however, P is measured in front of the surface with a small positive TSDF value F_2 representing the free space near the surface. Since a TSDF represents the distance from a voxel to its nearest surface, F_2 is closer to the true value of P. However, averaging F_1 and F_2 inaccurately produces a negative TSDF value, which is not correct and is the common reason why high-frequency geometries are often smoothed or deformed.

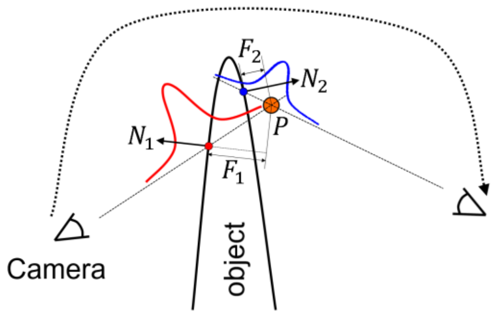


Figure 4. Illustration of the cause of deformation around thin geometries. The voxel P is considered at the back of the surface when first viewed from the left, whereas it is found in front of another surface when the camera orbits to the opposite side. Averaging these observations leads to erroneous TSDF results. To address this problem, we propose a probabilistic binary classification strategy for anisotropic data fusion.

Since we recognize thin geometries with our raycast procedure in advance, we can address this problem by efficiently fusing data around sharp zones in an anisotropic manner. Apart from the original volume S, we maintain a "ghost" volume S' as extra storage, so the TSDF of each voxel is determined by both volumes. At time k, for each voxel P in the global frame, we first transform and project it to an image pixel p, then check the value of $D_k^f(p)$ and compare it with $D_k(p)$ to decide in which volume the current measurement should be fused. Algorithm 1 describes this process in detail as follows:

```
Algorithm 1: Integrate TSDF Volumes
Input: ${D_k, T_{g,k}, D_k^f \mid k = 1,2,\dots,N}$
Output: $S, S'$
1:  For each: $P \in S$
2:      $p \leftarrow \pi(T_{g,k}^{-1} P)$
3:      $S_{D_k}(P) \leftarrow [F_{D_k}(P), W_{D_k}(P)]$
4:      If $\eta < -\delta_a$ then
5:          If $\lambda \geq \Lambda$ and $W_{k-1}(P) < \phi$ then
6:              $W_k(P) \leftarrow W_{k-1}(P) - 1$
7:          End If
8:          If $W'_{k-1}(P) > 0$ then
9:              $W'_k(P) \leftarrow W'_{k-1}(P) - 1$
10:         End If
11:         continue
12:     End If
13:     If $D_k^f(p) > 0$ then
14:         $S_k(P) \leftarrow fuse\left(S_{k-1}(P), S_{D_k}(P)\right)$
15:     Else
16:         $D_{diff}(p) \leftarrow D_k(p) + D_k^f(p)$
17:         If $D_{diff}(p) > 0$ then
18:             continue
19:         Else If $D_{diff}(p) < -\xi$ then
20:             $S_k(P) \leftarrow fuse\left(S_{k-1}(P), S_{D_k}(P)\right)$
21:         Else
22:             If $W'_{k-1} < \phi$ then
```

```
23:                 $S'_k(P) \leftarrow fuse\left(S'_{k-1}(P), S_{D_k}(P)\right)$
24:             Else
25:                 $sdf_{front} \leftarrow D_k(p) - P_z$
26:                 $sdf_{back} \leftarrow P_z + D_k^f(p)$
27:                 $\rho \leftarrow \frac{\min(0, sdf_{back})}{\min(0, sdf_{back}) + \min(0, sdf_{front})}$
28:                 $\rho' \leftarrow 1 - \rho$
29:                 $F_k(P) \leftarrow \frac{\rho W_{k-1}(P) F_{k-1}(P) + \rho' W'_{k-1}(P) F'_{k-1}(P)}{\rho W_{k-1}(P) + \rho' W'_{k-1}(P)}$
30:                 $W_k(P) \leftarrow \rho W_{k-1}(P) + \rho' W'_{k-1}(P)$
31:                 $W'_k(P) \leftarrow 0$
32:             End If
33:         End If
34:     End If
35:End for
```

where the $fuse$ function performs a weighted moving average. The threshold ξ denotes the width of a narrow band near the surface, within which the classification strategy is employed. When the difference of the front and back surface $D_{diff}(p)$ along the pixel ray $L_g(p)$ lies in the range $[-\xi, 0]$, we attempt to fuse the incoming data to the "ghost" volume S'. When the weight of S' is above a confidence threshold ϕ, we merge it to the main volume S based on the SDF of each voxel P to both the front and back surface. Note that sdf_{back} is calculated opposite to sdf_{front}, resulting in negative SDF for voxels before (seen from the current camera's viewpoint) the back surface, and the positive values are behind.

We finally employ a simple volume denoising scheme for cleaner mesh outputs (Algorithm 1, lines 4~12). For each voxel P at time k, when $\eta < -\delta_a$, we check the values of both W_k and W'_k. If W_k is below ϕ and the corresponding pixel is away from image edges ($\lambda \geq \Lambda$) or if W'_k is not zero, we gradually decrease their values by one. This strategy is simple but effective in the presence of highly uncertain depth measurements, e.g., at depth discontinuities or grazing viewing regions.

IV. EVALUATION

A. DATASET

In our previous work [18], we released a dataset, CU3D, consisting of three synthetic depth image sequences with ground-truth camera trajectories and ground-truth mesh models and six real-world noisy data sequences with 3D-printed ground-truth models but no ground-truth camera trajectories. We extend the dataset with several new depth sequences by scanning some models obtained from daily life, including four dinosaur toys (a Stegosaurus, a Spinosaurus, a Pterosaur, and a Diplodocus; the former three are soft rubber products, but the last one has a rigid body) and a small cardboard box, as shown in Fig. 5. We choose these objects because of their challenging structural details, e.g., the sharp claws, tail tips, thin spines of the toy dinosaurs and the thin cardboard walls. Since we have no per-vertex ground-truth for these objects, we measure the distances between manually selected points or the thickness of thin structures with a Vernier caliper (to 0.02 mm) for a quantitative evaluation of the compared algorithms. These measurements include the distance from nose to tail tip of the Stegosaurus (abbreviated as "stego-n2t"), the Spinosaurus (*abbrev.* "spino-n2t"), and the Diplodocus (*abbrev.* "diplo-n2t"), the distance from the cranial crest tip to tail tip of the Pterosaur (*abbrev.* "ptero-c2t"), the thickness of the right forelimb of the Diplodocus (*abbrev.* "diplo-r-forelimb"), and the thickness of the wall of the cardboard box (*abbrev.* "box-wall"). The measured results as ground-truth are listed in Table I. Note that the scale of our measured objectives varies from hundreds of millimeters to only a few millimeters, which challenges the performance of the compared algorithms.

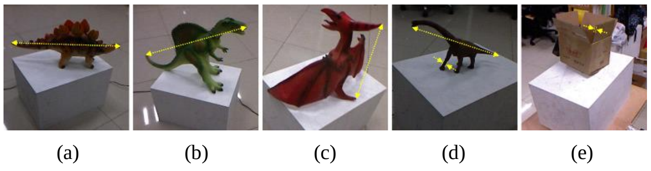


Figure 5. The new scanned objectives added to our CU3D dataset, including (a) a Stegosaurus, (b) a Spinosaurus, (c) a Pterosaur, (d) a Diplodocus, and (e) a small thin cardboard box. The yellow arrowed dashed lines in each subfigure indicate the manually chosen distance to measure with a Vernier caliper.

TABLE I
PHYSICAL DISTANCES BETWEEN THE MANUALLY CHOSEN POINTS MEASURED WITH A VERNIER CALIPER.

Measurement item	Ground-truth distance (mm)
stego-n2t	491.78
spino-n2t	589.38
ptero-c2t	276.80
diplo-n2t	321.26
diplo-r-forelimb	12.22
box-wall	6.24

B. CAMERA TRAJECTORY ACCURACY

Our reference-based odometry algorithm is compared with the following: KinectFusion [1] (PCL's Kinfu implementation [54]), the boundary odometry of Zhou *et al.* [33], the SDF-2-SDF algorithm of Slavcheva *et al.* [17] (our implementation), and our previous work CuFusion [18]. We compute the absolute trajectory error (ATE) of the 6DOF camera poses on the synthetic depth image sequences – the armadillo, dragon, and bunny sequences from the CU3D dataset. Although planar surfaces of the cuboid occupy the majority of the depth images, the compared algorithms achieve decent camera trajectories without prominently accumulating drift, as listed in Table II. The ATE of the proposed algorithm in this work is slightly larger (less than 0.5 mm) than that of our previous version of CuFusion, which has little impact on the accuracy of reconstruction. Note that the truncation distance is set to a small value (5 mm) for most of the cases, whereas when testing Kinfu on sequence “armadillo”, it is set to 25 mm because the small truncation distance in this test case results in severe camera drift and thus failure of the reconstruction. We discuss this phenomenon in section V.

TABLE II
COMPARISON OF ATE ON THE SYNTHETIC SEQUENCES OF CU3D.

Algorithm	Armadillo	Dragon	Bunny
KinFu	3.6	3.0	4.2
Zhou *et al.*	5.6	2.9	4.3
SDF-2-SDF	5.7	4.4	6.6
CuFusion	**1.7**	**1.7**	**1.3**
Our approach	**1.7**	1.8	1.7

C. COMPARISON WITH POSE GRAPH OPTIMIZATION

We further compare our online results with those optimized with pose graph optimization. The camera poses where a triplet of mutually orthogonal planes of the reference cuboid is observed are selected as keyframes and added as fixed vertices into the pose graph, between which the poses are added as floating vertices and optimized with the “g2o” framework. We test the optimization results on real-world noisy sequences and find that they have little accuracy gain compared with our online results. Fig. 6 shows the per-frame trajectory difference on the sequence “lambunny” of CU3D. The maximum difference is less than 0.1 mm, demonstrating that our algorithm outputs camera trajectories comparable to the graph optimized ones.

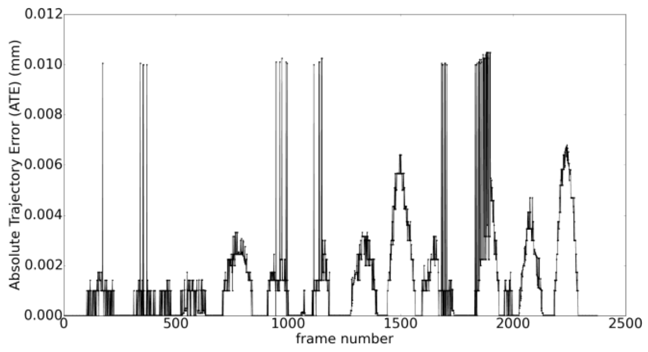


Figure 6. Comparison of per-frame ATE between our algorithm and the graph optimized result on sequence “lambunny”.

D. SURFACE RECONSTRUCTION ACCURACY

We quantitatively evaluate the algorithms in the following three ways – the per-vertex error, the physical scale fidelity, and finally the mesh noise of the reconstructions.

1) CLOUD-TO-MESH DISTANCE

We first test the cloud-to-mesh (C2M) distance from the reconstructions to the ground-truth models on the three synthetic depth streams and six scanning sequences of the 3D-printed models. Surface reconstructions are first aligned against the ground-truth models, and the C2M distances in millimeters between the reconstructed and ground-truth models are computed using the CloudCompare software [57]. The C2M distance is quantified by two standard statistics: the mean and standard deviation (Std.). Fig. 7 plots the reconstructions of the “buddhahead” sequence and the corresponding heat maps of the C2M distance. Table III provides the error evaluation details of the five compared algorithms on the nine data sequences.

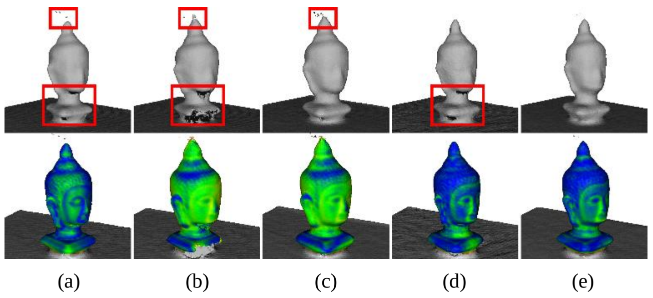


Figure 7. Qualitative comparison of the algorithms, namely, (a) Kinfu, (b) Zhou *et al.*, (c) SDF-2-SDF, (d) CuFusion, and (e) our proposed method, on sequence “buddhahead”. Top row: the reconstructed mesh models, bottom row: the heat maps corresponding to the C2M errors.

TABLE III
SURFACE RECONSTRUCTION ACCURACY ON OUR SYNTHETIC AND REAL-WORLD DATA, WITH THE C2M ERROR METRIC (MEAN±STD.) IN MILLIMETERS.

Sequence	KinFu	Zhou *et al.*	SDF-2-SDF	CuFusion	Our approach
armadillo	1.0±1.1	0.2±0.2	0.4±0.3	**0.2±0.1**	0.2±0.2
dragon	0.2±0.2	0.1±0.1	0.4±0.3	0.1±0.1	**0.1±0.1**
bunny	0.2±0.4	0.2±0.4	0.4±0.4	**0.1±0.1**	0.2±0.3
mug	1.4±1.1	1.4±0.9	0.8±0.9	0.9±0.8	**0.7±0.8**
lambunny	**0.9±1.2**	1.1±1.2	1.4±1.1	1.3±1.2	1.1±1.1
owl	2.4±1.1	1.0±1.0	2.2±1.2	1.2±1.1	**0.9±1.2**
tooth	2.7±1.4	1.3±1.3	1.3±1.3	1.5±1.5	**1.3±1.3**
wingedcat	1.5±2.0	1.5±2.0	2.7±2.6	1.4±2.2	**1.3±1.9**
buddahead	1.6±2.1	3.0±2.0	2.7±2.0	**1.3±2.0**	1.5±2.1

2) PHYSICAL SCALE OF THE RECONSTRUCTION

As seen in Table III, on the reconstruction of small-sized objects, it is hard to determine which method has a significant advantage over the others as long as no noticeable camera drift occurs. To test the reconstruction fidelity, we also measured the physical scale of the reconstructions (total length or body part thickness) of the five newly added scanning sequences. Distance in millimeters is measured using the point-picking tool of the CloudCompare software. We compare the measuring results with the corresponding ground-truth values listed in Table I and use the absolute distance (listed in the parentheses) between the measurements and the ground-truth values as an indicator of the accuracy of the tested algorithms, as illustrated in Table IV. Fig. 8 qualitatively demonstrates the reconstruction of the “Diplodocus” and “cardboard box” sequences. Note that the SDF-2-SDF method fails to create the Diplodocus model due to drift in the camera trajectory estimation, and neither the Kinfu nor the Zhou *et al.* method successfully reconstructs the Diplodocus's right forelimb.

TABLE IV
MEASUREMENT RESULTS OF THE MANUALLY CHOSEN OBJECT PARTS IN MILLIMETERS. THE ABSOLUTE DISTANCE BETWEEN THE MEASUREMENTS OF THE MESH MODELS AND THE REAL OBJECTS ARE SHOWN IN THE PARENTHESES BEHIND (SMALLER IS BETTER).

Measurement item	Ground-truth (mm)	KinFu	Zhou *et al.*	SDF-2-SDF	CuFusion	Our approach
stego-n2t	491.78	479.07 (12.71)	479.18 (12.6)	**485.62 (6.16)**	475.6 (16.18)	483.4 (8.38)
spino-n2t	589.38	595.48 (6.1)	601.41 (12.03)	595.66 (6.28)	596.76 (7.38)	**592.98 (3.60)**
ptero-c2t	276.80	269.11 (7.69)	271.99 (4.81)	272.11 (4.69)	270.28 (6.52)	**276.77 (0.03)**
diplo-n2t	321.26	293.91 (27.35)	306.66 (14.6)	--	300.29 (20.97)	**316.63 (4.63)**
diplo-r-forelimb	12.22	--	--	--	6.7 (5.52)	**12.1 (0.12)**
box-wall	6.24	6.66 (0.42)	7.32 (1.08)	**5.92 (0.32)**	11.63 (5.39)	6.76 (0.52)

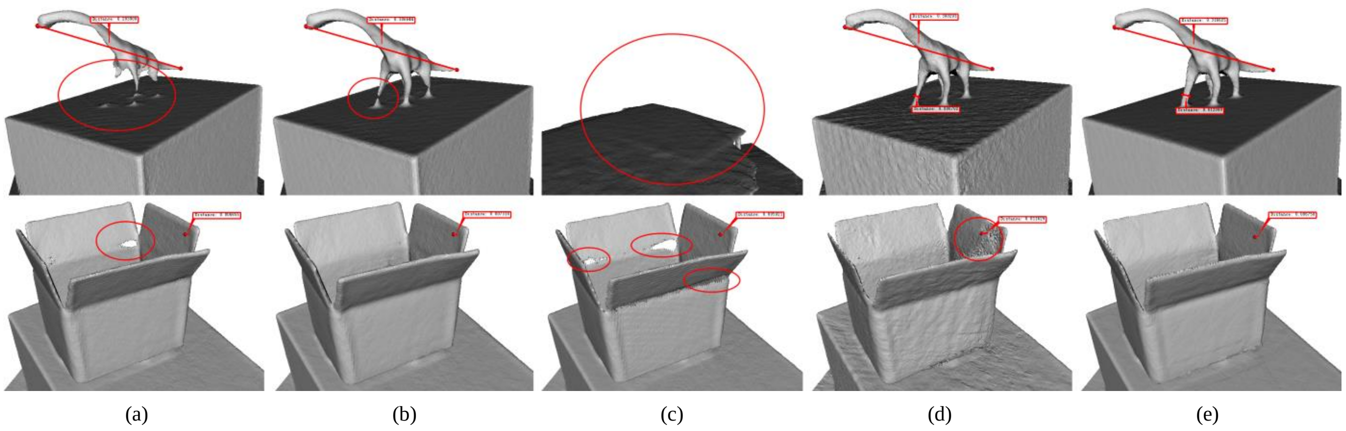


Figure 8. **Qualitative comparison of the algorithms, namely, (a) Kinfu, (b) Zhou *et al.*, (c) SDF-2-SDF, (d) CuFusion, and (e) our proposed method, on sequence "Diplodocus" (top row) and sequence "cardboard box" (bottom row).**

3) MESH NOISE

We finally compare the mesh and point cloud generated by the evaluated algorithms. For applications such as 3D printing, clean and topologically consistent mesh output are of vital importance. Due to the accurate camera pose estimation and the denoising strategy during the data fusion process, our method can generate globally consistent mesh models superior to other methods. Fig. 9 qualitatively demonstrates the reconstruction of the compared methods on sequence "Spinosaurus", where sectional views are provided for visualizing both sides of the surface. As seen from the sectional views, our mesh model contains the least noise and outliers among the compared algorithms.

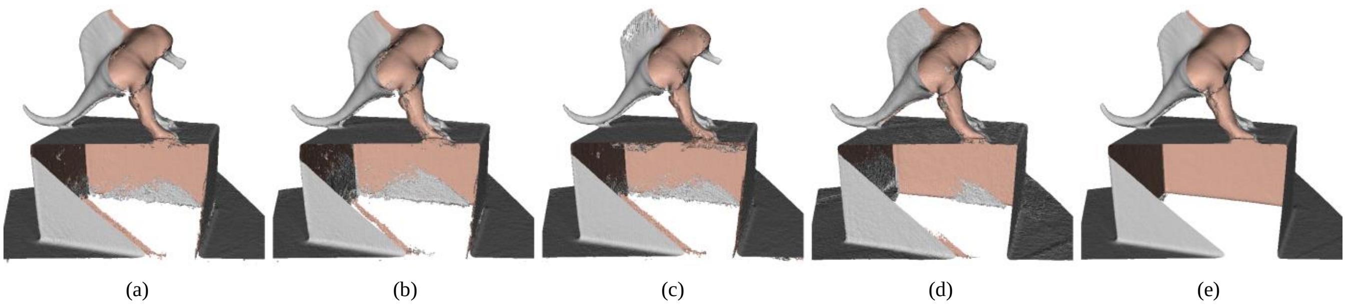


Figure 9. **Sectional views of the algorithms, namely, (a) Kinfu, (b) Zhou *et al.*, (c) SDF-2-SDF, (d) CuFusion, and (e) our proposed method, on the reconstruction of sequence "Spinosaurus".**

TABLE V
FACE AND VERTEX (LISTED IN THE PARENTHESES) COUNT OF THE SURFACE MODELS GENERATED BY THE COMPARED ALGORITHMS. A SMALLER VALUE IS BETTER IF NO DRASTIC FAILURE OF THE RECONSTRUCTION OCCURS, INDICATING CLEANER MODEL OUTPUT, AS DEMONSTRATED IN FIG. 9.

Sequence	KinFu	Zhou *et al.*	SDF-2-SDF	CuFusion	No denoising	Our approach
armadillo	387601 (194909)	380642 (191541)	382897 (192600)	382161 (192372)	385066 (196022)	**379444 (190934)**
dragon	427022 (216153)	425901 (215487)	432404 (217893)	**421269** (213688)	425755 (215111)	422416 **(213238)**
bunny	381912 (192908)	382453 (193282)	385194 (193916)	381273 (192264)	382929 (193144)	**380457 (191965)**
mug	371373 (194913)	395920 (213064)	378467 (198830)	349920 (181024)	354388 (185482)	**328169 (166157)**
lambunny	390045 (210086)	399374 (215993)	374001 (196184)	347144 (179696)	339088 (177399)	**324172 (165600)**
owl	373064 (200120)	381390 (204482)	362957 (186951)	364773 (188779)	346981 (181748)	**333662 (171376)**
tooth	374547 (202142)	399135 (216662)	368753 (192225)	365439 (188833)	352377 (184166)	**327058 (166900)**
wingedcat	404647 (220129)	404900 (220328)	497058 (257532)	371897 (193442)	362470 (190451)	**336911 (171508)**
buddhahead	411831 (222773)	445290 (241586)	417893 (219216)	389424 (200526)	381963 (199972)	**364055 (184930)**
Stegosaurus	495461 (264761)	482179 (259652)	827302 (421941)	467333 (240470)	424704 (223578)	**400986 (204506)**
Spinosaurus	395476 (209303)	417042 (221981)	389946 (198736)	391549 (201743)	389678 (205489)	**345417 (175003)**
Pterosaur	462870 (245659)	477072 (254872)	623249 (321049)	445836 (227873)	453611 (236747)	**424727 (214826)**
Diplodocus	517564 (274194)	421678 (223022)	--	407979 (211236)	402631 (210564)	**365913 (185250)**
Cardboard box	515780 (273150)	532378 (282620)	600319 (320499)	501039 (259089)	473092 (245859)	**449884 (227851)**

Table V quantitatively illustrates the number of triangular faces and vertices (listed in the parentheses) of each model generated by the algorithms for each depth stream of our CU3D dataset (currently 14 sequences in total). On the first three synthetic sequences, the statistical differences are approximately 1% since there is no noise in the input depth images. On the real-world data, however, the differences reach up to 10%~30%. Our algorithm generates the fewest outliers in the point clouds as well as the triangle meshes on all of the noisy sequences, proving the effectiveness of our denoising scheme. The column "no denoising" is an ablation experiment with the denoising functionality removed from our approach, which will be discussed in detail in the following "Ablation Study" subsection.

E. Computational Performance

We maintain a volume of 0.6m^3 with a resolution of 256^3 throughout our experiments. All methods are tested on a PC with a 3.2 GHz Intel Core i5-3470 CPU and NVidia GTX 960 GPU. To speed up the method, we follow the coarse-to-fine iterative registration scheme but reduce the maximum number of iterations in the finest level from 10 to 1. In addition, we also reduce the edge-to-contour optimizations to only once at the end of each iteration level. Due to the compelling constraints of the reference cuboid, these two simplifications do not hinder the camera tracking performance. In Table VI, we present the average execution times in milliseconds for each step of our framework. Note that we solve E_{e2c} on the CPU during camera pose estimation, slightly slowing down the iteration. The generation of the cuboid's depth maps is also implemented on the CPU. Our integration of depth maps into the global volume is much slower than the simple moving average strategy at 10.6 ms versus 2 ms. The total processing time for one frame is approximately 50 ms, resulting in approximately 20 FPS.

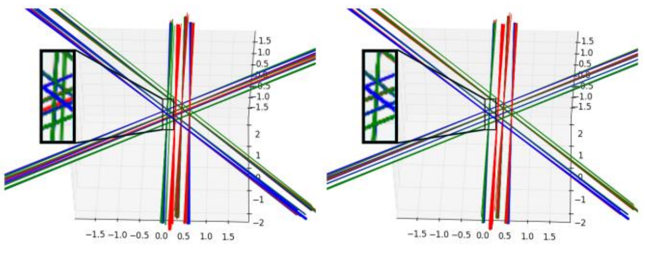
TABLE VI
COMPUTATIONAL PERFORMANCE RESULTS FOR EACH MAIN STEP IN OUR PIPELINE

Steps	Time cost (ms)
Preprocessing	12.3
Cuboid depth map generation	5.4
Camera pose estimation	5.7
Raycast with refraction	5.6
Integrate TSDF volumes	10.6
Other	11.2
Total	50.8

F. Ablation Study

We further perform three experiments to evaluate the effectiveness of each part of our algorithm.

Do the cuboid constraints help in odometry? To answer this question, we qualitatively compare the odometry results from our optimization objective E_{track} with the baseline E_{f2m} on real-world sequences to show the effect of the cuboid constraints E_{f2c} and E_{e2c}. This is achieved by plotting the edges with common vertices of the cuboid in the global coordinate frame when orthogonal trihedra are seen at each frame. Ideally, these edges form a cuboid in the global space. However, due to inaccuracies in the raw data and estimation of camera poses, the edges are scrambled when transformed into the global space to varying degrees. To visualize this result more clearly, we scale these edges to 5 meters in Fig. 10. Our method produces more consistent edges in the global space than the baseline, proving its effectiveness.



(a) (b)

Figure 10. Qualitative comparison between the (a) baseline and (b) our odometry method on sequence "Diplodocus". We plot the detected edges (scaled to 5 m) of the reference cuboid in the global frame as an indication of the odometry accuracy. Note that although no noticeable camera drift occurs with both methods, our approach produces more consistent edges.

Does our data fusion strategy help in reconstruction? To investigate the effect of our data fusion algorithm, we replace it with the baseline moving average TSDF fusion strategy. We use our cuboid odometry method in both experiments to ensure the accuracy of camera trajectories. Fig. 11 shows the mesh reconstructions on the sequence "cardboard box" at frame 1379. Although the same camera trajectories are provided, a simple moving average (Fig. 11(a)) produces holes at the surface of the box's thin wall due to the fusion of highly noisy raw depth measurements into the global TSDF volume. By contrast, our fusion strategy succeeds in preserving the manifoldness of the surface (Fig. 11(b)).

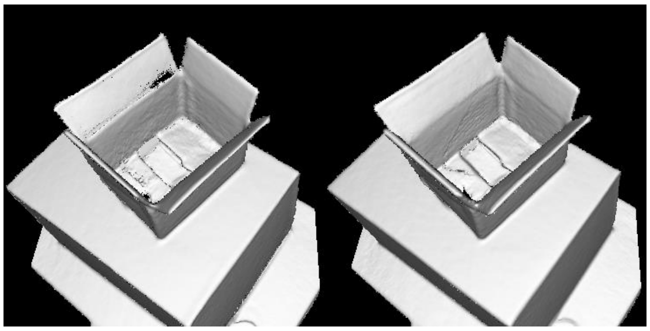


Figure 11. Qualitative comparison between (a) moving average and (b) our data fusion strategy on sequence "cardboard box", both with the same camera trajectory. Our method produces a surface free of topological holes, preserving its manifoldness.

Does the denoising strategy help? As shown in the "no denoising" column in Table V, the results of removing the denoising strategy perform similarly to the other methods but inferior to our approach despite using the same odometry and data fusion scheme as ours. This exhibits the effectiveness of our denoising strategy, which is only weakly coupled with our other two contributions and is mainly aimed at the topological consistency of the mesh outputs. As shown in Fig. 9, topological noise, including isolated and dangling triangles, mostly exists at the back of the surface. This problem is primarily caused by the truncation of the SDF at the back of the surface and is inevitable as long as the TSDF fusion scheme is adopted within the framework.

V. DISCUSSION AND CONCLUSION

In this paper, we proposed an approach for accurate and clean object reconstruction. Given a stream of depth images and a known cuboid reference object present in the scene, we maintained drift-free camera pose estimation with the constraints added by the reference object without the need for pose graph optimization. We then fused the living data into one globally consistent model in real time by transforming the problem of data fusion into a probabilistic binary classification problem, ensuring the reconstruction fidelity, especially in highly curved and concave zones. We preserved the surface smoothness and cleanliness utilizing a simple denoising strategy, especially in invisible areas near the back of the surface. Our method went further for applications that demand fine-grained reconstruction details, such as 3D printing.

Several limitations remain in our framework. First, our method works under the assumption that the scanned objects remain relatively static in the scene. Any deformation or relative motion of the objects may lead to failure in the reconstruction, especially in the presence of thin geometries. Second, the need for the reference cuboid for accurate odometry limits the generalization and scalability of our method. Third, despite our data fusion and denoising strategy, the manifoldness of the surface is not always guaranteed. Our system performs well in mitigating topological noise of the mesh output but may still fail when the raw input depth data are too noisy (e.g., severe motion blur). Finally, the computational complexity and memory overhead are not optimal in our current implementation. Cuboid depth map generation and the optimization of E_{e2c} can be ported from CPU to GPU to speed up the tracking process. Additionally, our volumetric representation still costs twice as much GPU memory as KinectFusion at the same resolution; this can be further optimized with the voxel hashing technique.

In the future, we will first refine our code for improvements in terms of time and memory usage. Second, we will explore the fidelity preservation of thin geometry in motion to make our system work with dynamic scenes. Furthermore, because our system currently performs geometric reconstruction only, we will explore the possibility of real-time high-quality color mapping onto the model outputs.

REFERENCES

[1] R. A. Newcombe *et al.*, “KinectFusion: Real-time Dense Surface Mapping and Tracking,” in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2011, pp. 127–136.

[2] P. J. Besl and N. D. McKay, “A method for registration of 3-D shapes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[3] Y. Chen and G. Medioni, “Object Modelling by Registration of Multiple Range Images,” *Image Vis. Comput*, vol. 10, no. 3, pp. 145–155, Apr. 1992.

[4] W. E. Lorensen and H. E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm,” *SIGGRAPH Comput Graph*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[5] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann, “When can we use kinectfusion for ground truth acquisition,” in *Proc. Workshop on Color-Depth Camera Fusion in Robotics*, 2012, vol. 2.

[6] K.-L. Low, “Linear least-squares optimization for point-to-plane icp surface registration,” *Chap. Hill Univ. N. C.*, vol. 4, 2004.

[7] A. Segal, D. Haehnel, and S. Thrun, “Generalized-ICP.,” in *Robotics: science and systems*, 2009, vol. 2, p. 435.

[8] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, “Kintinuous: Spatially extended kinectfusion,” 2012.

[9] T. Weise, T. Wismer, B. Leibe, and L. Van Gool, “In-hand scanning with online loop closure,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 2009, pp. 1630–1637.

[10] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, “ElasticFusion: Dense SLAM Without A Pose Graph,” in *Proceedings of Robotics: Science and Systems*, Rome, Italy, 2015.

[11] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, “Real-time large-scale dense RGB-D SLAM with volumetric fusion,” *Int. J. Robot. Res.*, vol. 34, no. 4–5, pp. 598–626, 2015.

[12] O. Kähler, V. A. Prisacariu, and D. W. Murray, “Real-time large-scale dense 3d reconstruction with loop closure,” in *European Conference on Computer Vision*, 2016, pp. 500–516.

[13] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust reconstruction of indoor scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5556–5565.

[14] Q.-Y. Zhou and V. Koltun, “Dense scene reconstruction with points of interest,” *ACM Trans. Graph. ToG*, vol. 32, no. 4, p. 112, 2013.

[15] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, “ElasticFusion: Real-time dense SLAM and light source estimation,” *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[16] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic, “SDF-2-SDF: highly accurate 3D object reconstruction,” in *European Conference on Computer Vision*, 2016, pp. 680–696.

[17] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic, “SDF-2-SDF Registration for Real-Time 3D Reconstruction from RGB-D Data,” *Int. J. Comput. Vis.*, pp. 1–22, 2018.

[18] C. Zhang and Y. Hu, “CuFusion: Accurate real-time camera tracking and volumetric scene reconstruction with a cuboid,” *Sensors*, vol. 17, no. 10, p. 2260, 2017.

[19] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *null*, 2003, p. 1403.

[20] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007, pp. 225–234.

[21] R. A. Newcombe and A. J. Davison, “Live dense reconstruction with a single moving camera,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 1498–1505.

[22] J. Stühmer, S. Gumhold, and D. Cremers, “Real-time dense geometry from a handheld camera,” in *Joint Pattern Recognition Symposium*, 2010, pp. 11–20.

[23] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2320–2327.

[24] S. Izadi *et al.*, “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2011, pp. 559–568.

[25] J. Sturm, E. Bylow, F. Kahl, and D. Cremers, “CopyMe3D: Scanning and Printing Persons in 3D,” in *Pattern Recognition - 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, 2013, pp. 405–414.

[26] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, “Real-time 3D Reconstruction at Scale Using Voxel Hashing,” *ACM Trans Graph*, vol. 32, no. 6, pp. 169:1–169:11, Nov. 2013.

[27] V. A. Prisacariu *et al.*, “InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure,” *ArXiv Prepr. ArXiv170800783*, 2017.

[28] A. Dai, M. Nie\s sner, M. Zollhöfer, S. Izadi, and C. Theobalt, “Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration,” *ACM Trans. Graph. TOG*, vol. 36, no. 4, p. 76a, 2017.

[29] H. Roth and M. Vona, “Moving Volume KinectFusion.,” in *BMVC*, 2012, vol. 20, pp. 1–11.

[30] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, “Deformation-based loop closure for large scale dense RGB-D SLAM,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 548–555.

[31] D. Lefloch, M. Kluge, H. Sarbolandi, T. Weyrich, and A. Kolb, “Comprehensive Use of Curvature For Robust And Accurate Online Surface Reconstruction,” *IEEE Trans. Pattern Anal. Mach. Intell. PAMI*, p. 10.1109/TPAMI.2017.2648803, 2017.

[32] W. Dong, Q. Wang, X. Wang, and H. Zha, “PSDF Fusion: Probabilistic Signed Distance Function for On-the-fly 3D Data Fusion and Scene Reconstruction,” *ArXiv Prepr. ArXiv180711034*, 2018.

[33] Q.-Y. Zhou and V. Koltun, “Depth camera tracking with contour cues,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 632–638.

[34] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas, “Tracking an RGB-D camera using points and planes,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 51–58.

[35] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, “Point-plane SLAM for hand-held 3D sensors,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 5182–5189.

[36] M. Kaess, “Simultaneous localization and mapping with infinite planes.,” in *ICRA*, 2015, vol. 1, p. 2.

[37] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, “Dense planar SLAM,” in *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, 2014, pp. 157–164.

[38] L. Ma, C. Kerl, J. Stückler, and D. Cremers, “Cpa-slam: Consistent plane-model alignment for direct rgb-d slam,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016, pp. 1285–1291.

[39] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, “Slam++: Simultaneous localisation and mapping at the level of objects,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1352–1359.

[40] N. Fioraio and L. Di Stefano, “Joint detection, tracking and mapping by semantic bundle adjustment,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1538–1545.

[41] N. Mellado, D. Aiger, and N. J. Mitra, “Super 4pcs fast global pointcloud registration via smart indexing,” in *Computer Graphics Forum*, 2014, vol. 33, pp. 205–215.

[42] Q.-Y. Zhou, J. Park, and V. Koltun, “Fast global registration,” in *European Conference on Computer Vision*, 2016, pp. 766–782.

[43] J. Yang, H. Li, D. Campbell, and Y. Jia, “Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, 2016.

[44] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (FPFH) for 3D registration,” in *IEEE International Conference on Robotics & Automation*, 2009.

[45] A. Zeng, S. Song, M. Nie\s sner, M. Fisher, J. Xiao, and T. Funkhouser, “3dmatch: Learning local geometric descriptors from rgb-d

reconstructions," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017, pp. 199–208.

[46] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3d point matching," *Comput. Vis. Pattern Recognit. CVPR IEEE*, vol. 1, 2018.

[47] A. Elfes and L. Matthies, "Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representataion," in *26th IEEE Conference on Decision and Control*, 1987, vol. 26, pp. 1802–1807.

[48] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1996, pp. 303–312.

[49] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *3DTV-Conference, 2013 International Conference on*, 2013, pp. 1–8.

[50] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, pp. 742–749.

[51] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface Elements As Rendering Primitives," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2000, pp. 335–342.

[52] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 573–580.

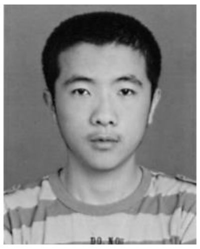
[53] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, 2014.

[54] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[55] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6218–6225.

[56] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3607–3613.

[57] "CloudCompare - Open Source project." [Online]. Available: http://www.danielgm.net/cc/. [Accessed: 19-Jul-2017].



CHEN ZHANG received his B.S. degree in computer science from Zhejiang University, China, in 2013. After September 2013, he studied as a Ph.D. student in College of Computer Science and Technology, Zhejiang University, China. He is currently working with the Computer Animation & Perception Group under the State Key Lab of CAD & CG, Zhejiang University. His primary research interests include simultaneous localization and mapping (SLAM) and 3D reconstruction.