Mobile3DScanner: An Online 3D Scanner for High-quality Object Reconstruction with a Mobile Device

Xiaojun Xiang, Hanqing Jiang, Guofeng Zhang, *Member, IEEE*, Yihao Yu, Chenchen Li, Xingbin Yang
Danpeng Chen, and Hujun Bao, *Member, IEEE*

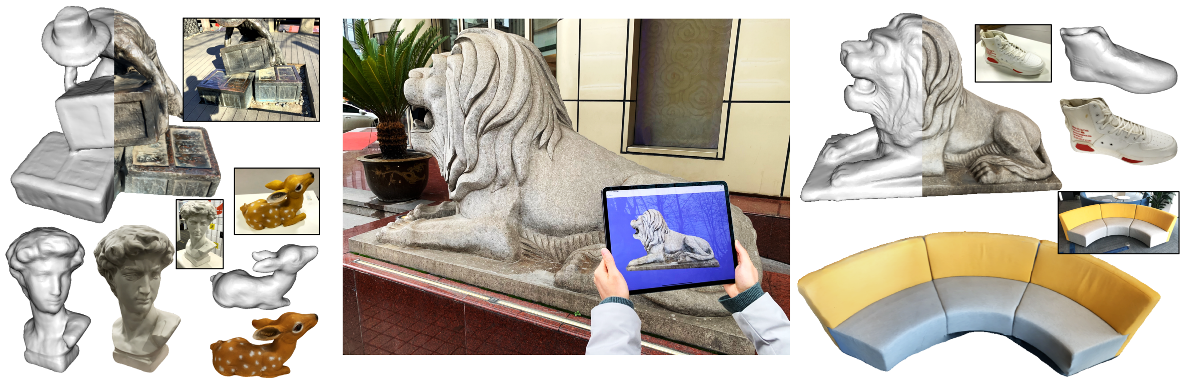

Fig. 1. Examplar 3D models with mapped textures are shown on the left and right columns, which are scanned and reconstructed online on iPad Pro 2020 using our Mobile3DScanner system. A representative keyframe image is given for each case. As shown in the middle column, our 3D scanning pipeline is capable of scanning a large immovable object such as a "Lion" statue.

Abstract—We present a novel online 3D scanning system for high-quality object reconstruction with a mobile device, called Mobile3DScanner. Using a mobile device equipped with an embedded RGBD camera, our system provides online 3D object reconstruction capability for users to acquire high-quality textured 3D object models. Starting with a simultaneous pose tracking and TSDF fusion module, our system allows users to scan an object with a mobile device to get a 3D model for real-time preview. After the real-time scanning process is completed, the scanned 3D model is globally optimized and mapped with multi-view textures as an efficient post-process to get the final textured 3D model on the mobile device. Unlike most existing state-of-the-art systems which can only scan homeware objects such as toys with small dimensions due to the limited computation and memory resources of mobile platforms, our system can reconstruct objects with large dimensions such as statues. We propose a novel visual-inertial ICP approach to achieve real-time accurate 6DoF pose tracking of each incoming frame on the front end, while maintaining a keyframe pool on the back end where the keyframe poses are optimized by local BA. Simultaneously, the keyframe depth maps are fused by the optimized poses to a TSDF model in real-time. Especially, we propose a novel adaptive voxel resizing strategy to solve the out-of-memory problem of large dimension TSDF fusion on mobile platforms. In the post-process, the keyframe poses are globally optimized and the keyframe depth maps are optimized and fused to obtain a final object model with more accurate geometry. The experiments with quantitative and qualitative evaluation demonstrate the effectiveness of the proposed 3D scanning system based on a mobile device, which can successfully achieve online high-quality 3D reconstruction of natural objects with larger dimensions for efficient AR content creation.

Index Terms—Object scanning, 3D reconstruction, visual-inertial pose tracking, adaptive voxel resizing

1 Introduction

3D object scanning is one of the core technologies of digital content creation in a wide range of graphics and augmented reality (AR) applications, where the reconstruction quality of 3D object models is of primary concern. Commercial digital 3D scanners are currently able to acquire accurate 3D models of natural objects, but rely on high accuracy depth sensors such as structure-light and expensive hardware for high-quality reconstruction computation. Due to the distance limitation of structure-light, most 3D scanners are more suitable for small object reconstruction. Recently, some research works such as [3, 12, 24] have made efforts to achieve 3D object reconstruction on a PC or mobile device connected with a consumer-level depth camera, but these systems require high performance computing hardware for complicated dense object reconstruction and the reconstruction quality strongly depends on the ranging accuracy of the depth camera. Some other systems [14, 27] tried to allow users to scan objects using mobile devices with a monocular camera, but usually cannot reconstruct high-quality 3D models without the help of depth sensor. Besides, these systems are limited in reconstructing homeware objects such as toys and shoes with small dimensions, due to the limited computation and memory resources of mobile platforms. Nowadays, researches of archeology and history usually require 3D scanning of large scale cultural relics and historic sites. Unfortunately, there are seldom 3D scanning systems which can perform high-quality online reconstruction of large dimension objects such as statues with a mobile device.

- *Xiaojun Xiang, Yihao Yu, Xingbin Yang, and Danpeng Chen are with SenseTime Research and Tetras.AI. E-mails: {xiangxiaojun,yuyihao,yangxingbin,chendanpeng}@tetras.ai.*
- *Hanqing Jiang and Chenchen Li are with SenseTime Research. E-mails: {jianghanqing,lichenchen}@sensetime.com.*
- *Guofeng Zhang and Hujun Bao are with the State Key Lab of CAD&CG, Zhejiang University. E-mails: {bao,zhangguofeng}@cad.zju.edu.cn.*
- *Corresponding Author: Hujun Bao.*
- *Xiaojun Xiang, Hanqing Jiang, and Guofeng Zhang assert equal contribution and joint first authorship.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

This paper presents a new 3D scanning system for high-quality online object reconstruction on a mobile device, which we named as Mobile3DScanner. Our system provides an online 3D object reconstruction capability for users to scan static natural objects and acquire high-quality 3D object models with texture maps, using a mobile device equipped with an embedded RGBD camera module such as iPad Pro 2020. The system consists of a real-time scanning module and an object model post-processing module. The real-time scanning module allows users to scan an object with a mobile device to get a real-time 3D model for preview. The object model post-processing module efficiently optimizes the geometric structures of the object model and maps the optimized model with multi-view texture images to get the final textured 3D model on the mobile device. Unlike most existing 3D scanning systems on mobile platforms which can only scan small dimension objects due to the limitations of depth ranging, computation and memory, our system can reconstruct objects with larger dimensions such as statues and humans. Large scale object reconstruction costs complicated computation and huge memory, which is limited on most mobile devices. Depth maps captured by embedded depth cameras such as dToF on iPad Pro usually contain depth errors or missing depths, which will significantly affect the quality of the final fused 3D model. We focus on solving these problems to make sure that large objects can be scanned and reconstructed online successfully on a mobile platform, with high accurate and complete geometric structures. Our main contributions can be summarized as:

- We propose a novel visual-inertial pose tracking method for real-time 3D reconstruction of objects. We combine iterative closest point (ICP) tracking with IMU, local mapping and loop closure for accurate real-time object tracking with a mobile device.
- We propose an adaptive TSDF voxel resizing strategy for real-time scanning of large objects on a mobile device. The voxel size is dynamically adjusted whenever too many voxels exceed the memory limitation during the online TSDF fusion, to make sure that large objects can be scanned successfully without out-of-memory on the mobile platform.
- Noticing that the embedded RGBD sensor on the mobile device usually have depth errors or over-smoothness, we propose to refine the depths from the embedded sensor by multi-view stereo (MVS), to provide more accurate object depths for better mesh generation. We incorporate the sensor depths from the sensor as priors into a multi-view semi-global matching (SGM) approach to acquire more accurate depths with better geometric details.
- We propose an efficient shape-from-shading (SFS) method with great time efficiency on the mobile device, to further improve geometric details of the object model online.

This paper is organized as follows. Section 2 briefly presents related work. Section 3 gives an overview of the proposed Mobile3DScanner system. The real-time object scanning module and the object model post-processing module are described in Sections 4 and 5 respectively. Finally, we evaluate the proposed solution in Section 6.

2 Related Work

Existing real-time static object reconstruction approaches can be generally divided into two categories: RGBD camera based 3D scanning and image-based multi-view reconstruction.

With the development of consumer RGBD cameras such as Microsoft Kinect and Intel RealSense, some real-time object scanning systems came out, which simultaneously track the real-time pose of the input depths using ICP [28], and fuse all the tracked object depths into a global TSDF model. An impressive work of this category is KinectFusion [24], which localizes a Kinect to a ray-casted global model using ICP, while fusing depths into the global model using TSDF. However, KinectFusion cannot work for large objects even on a desktop PC, due to the huge computation and memory costs by TSDF voxels. More recent works such as BundleFusion [3] achieve real-time reconstruction of large scale models on PC by introducing voxel hashing [26] to break through the limitations of TSDF fusion. InfiniTAM [12] proposed a highly efficient implementation of voxel hashing to achieve real-time scanning of large objects on a Nvidia Shield Tablet externally connected with a depth camera. Recently, some interactive in-hand object modeling systems have been proposed by [33, 42], which provide real-time registration of the input RGBD frames, while the in-hand interactivity enables the user to guide the object scanning process. Tzionas and Gall [40] improve the in-hand scanning pipeline to effectively facilitate the reconstruction of featureless and highly symmetric objects by 3D hand motion extraction. Xu et al. [44] proposed an online global non-rigid registration for high-quality small object scanning using a consumer RGBD camera, with a pause-and-restart operation to support 360-degree reconstruction. However, these in-hand works are specially designed for scanning handheld small objects. In general, very few works can reconstruct large objects on a mobile device with an embedded RGBD camera.

Although impressive dense reconstruction quality can be achieved by a consumer RGBD camera, it is inconvienent for a user to scan a object with an RGBD camera externally connected to a PC or mobile device, especially when a large object is to be scanned in an outdoor environment. This limitation encouraged researchers to explore real-time image-based multi-view object reconstruction systems on a mobile device with a monocular RGB camera, such as [14, 27, 31, 36]. Without input depths, these systems estimate depths of the input RGB frames, and fuse the estimated depths into a global 3D model by TSDF or surfels. MonoFusion [31] presented a real-time dense reconstruction with a single web camera and MobileFusion [27] proposed a real-time 3D object scanning tool on mobile devices with monocular camera. Both works perform volume-based TSDF fusion without voxel hashing, and therefore can only reconstruct small objects. For large-scale objects, Tanskanen et al. [39] proposed a live reconstruction system on mobile phones, which can perform inertial metric scale estimation while producing dense surfels of the scanned objects online. Kolev et al. [14] enhance the pipeline of [39] by introducing a confidence-based depth map fusion method. Schöps et al. [36] estimate sparse depths via motion stereo with a monocular fisheye camera on the GPU of Google's Project Tango Tablets, and integrates the filtered depths by the Tango's volumetric fusion pipeline, which is more suitable for large scale scene reconstruction. However, most of these multi-view reconstruction systems are not able to reconstruct so accurate 3D object models as the RGBD camera based scanning approaches.

RGBD registration is crucial for achieving accurate SLAM or online 3D reconstruction. To improve tracking stability on mobile devices, some works combine ICP registration with IMU, which is the same as our method. For example, Laidlow et al. [15] proposed a tightly-coupled RGBD inertial real-time tracking method on GPU. Nießner et al. [25] directly integrate the IMU data to estimate camera pose for improving ICP initialization and tracking recovery. Brunetto et al. [2] use two Kalman filters to estimate camera orientation and position respectively. [2, 25] do not optimize the bias of IMU and haven't combined IMU in ICP optimization, which might lead to worse accuracy and robustness. In contrast, our method optimizes the IMU bias and loosely coupled RGBD and IMU to achieve real-time tracking on mobile device with only CPU. Some works such as [5, 7, 23, 51] make efforts to reconstruct deformable or dynamic objects online using non-rigid registration. For example, Zollhöfer et al. [51] use an as-rigid-as-possible registration framework to do non-rigid surface fitting. DynamicFusion [23] and Fusion4D [5] employ the embedded deformation graph method [38] to track the motion of deformable objects. Guo et al. [7] proposed a more stablized non-rigid registration approach for dynamic objects by employing both L_0 and L_2 based motion regularizations to further constrain the tracking error propagation. However, these dynamic object reconstruction systems usually require high performance computing hardware for online reconstruction and are difficult to reconstruct large objects with high accuracy due to the complicated non-rigid registration and fusion computation.

There are also some offline dense reconstruction works on mobile devices. For example, 3DCapture [22] presented a dense textured model reconstruction system, which starts with an online RGB and

IMU data capturing stage followed by an offline post-processing reconstruction. The main reconstruction steps including pose tracking, depth estimation, TSDF depth fusion, mesh extraction and texture mapping, are all done as the post-processing stage on mobile devices. Poiesi et al. [29] described another cloud-based dense scene reconstruction system that performs Structure-from-Motion (SfM) and local bundle adjustment (BA) on monocular videos from smartphones to reconstruct a consistent point cloud map for each client, and run periodic full BA to align the maps of various clients on a cloud server. Some other works presented real-time dense scene reconstruction with GPU acceleration on a desktop PC, and are more suitable for reconstructing large-scale scenes instead of focusing on the reconstruction quality of a single object. For example, Merrell et al. [20] proposed a real-time 3D reconstruction pipeline on PC, which utilizes visibility-based and confidence-based fusion for merging multiple depth maps to an online large-scale 3D model. Pollefeys et al. [30] presented a complete system for real-time video-based 3D reconstruction, which captures large-scale urban scenes with multiple video cameras mounted on a driving vehicle. Recently, some methods [1, 46] adopt convolutional neural networks in 3D reconstruction, which are generally time consuming and limited in scale due to large GPU memory requirement. In comparison, our pipeline focuses on high-quality online object reconstruction, with both the scanning process and the post-process of our pipeline run on a mobile device with an embedded RGBD camera.

3 System Overview

We now outline the steps of the proposed online 3D scanning pipeline, as shown in Fig. 2. If a user wants to scan a natural object by our system, the object should be put on a horizontal planar surface such as a desk or the ground. As the user scans the object by a mobile device with a rear RGBD camera, our pipeline tracks 6DoF poses of the object in real-time using a visual-inertial ICP (VI-ICP) approach, which combines IMU and RGBD information to track the 6DoF poses on the front end, while maintaining a keyframe pool on the back end, with a local BA module and a loop closing module to refine poses of all the keyframes. The object is consistently segmented in each keyframe by a spatio-temporal planar surface tracking method. Simultaneously, the incoming depths are fused by the estimated poses to a TSDF model for real-time preview, using an adaptive voxel resizing strategy.

When the user finishes scanning, an object model post-processing module is activated to obtain the final object model. In this post-process, the keyframe poses are optimized in a global BA module, and the object depths of each keyframe are optimized by SGM. The optimized keyframe depths are fused by the globally optimized poses to a final TSDF model, followed by Marching Cubes [17], Poisson Surface Reconstruction (PSR) [13] and SFS to get the final 3D mesh. Finally, the 3D mesh model is mapped with multi-view images to get the final texture mapped 3D object model, as shown in Fig. 1. In the following sections, the main steps of our pipeline will be described in detail.

4 Real-time Object Scanning

In the real-time scanning stage, our system tracks the accurate pose of the object for each incoming frame scanned by the user on the front end, while maintaining a set of keyframes optimized by local mapping and loop closure on the back end. The optimized keyframe poses are used to fuse the depths to an implicit 3D model in real-time for previewing the scanning progress. We will describe the steps in details.

4.1 Visual-inertial Pose Tracking

Accurate pose tracking of the object is crucial for online high-quality 3D reconstruction. For each incoming RGBD frame, our system localizes the camera by loosely coupled integration of ICP and IMU in a real-time tracking thread on the front end, which predicts a prior pose by IMU estimation and integrates it into ICP tracking. Meanwhile, we optimize a sliding window of keyframes in a local mapping thread and perform loop closing in another thread on the back end for further optimization of the tracked poses. The following subsections will describe our IMU estimation, ICP tracking, local mapping and loop closing.

4.1.1 IMU Estimation

To acquire a reliable pose prediction for the current frame, the IMU state should be initialized first with the ICP tracking result of the first two frames, with the 6DoF part initialized with the ICP result, the velocity computed assuming a uniform motion, and the gravity calculated by madgwick filter [18]. After initialization, the IMU module can provide a current pose prediction based on the inertial data between the current frame and the last one. The predicted pose prior will be integrated into our ICP to enhance tracking robustness of the current frame, whose details will be given in Section 4.1.2. Then, the current frame pose tracked by ICP is used as constraints for further IMU optimization, and the optimized IMU state of the current frame is used for pose prediction of the next incoming frame, which forms a loosely coupled iterative optimization.

We follow the approach in [32] to proceed a sliding window-based IMU state optimization, which minimizes an energy function containing the residuals of IMU pre-integration proposed in [6], the relative pose from our ICP, and the same prior constraints as [32]. We use Ceres Solver [34] for the energy minimization, to get a more reliable IMU state for pose prediction.

4.1.2 Visual-inertial ICP Tracking

Each incoming RGBD frame F_t at time t is tracked by a frame-to-frame ICP approach similar to Open3D [49], with a previous frame as the reference one. Our system defines the first frame as the world coordinates, and maintains a reference frame F_r for ICP tracking of the current frame, with a 6DoF pose matrix $\mathbf{M}_r$ from global 3D space to local camera space. We propose a VI-ICP approach by adding the pose prior estimated by IMU in Section 4.1.1 to improve the robustness and accuracy of ICP tracking. The original ICP solution of [49] measures color and depth differences between the current frame and the reference one, which is defined as the following two energy terms:

$$\begin{array}{l} \mathscr{E}_d = \sum_{\mathbf{x}\in\mathbf{N}_r} \left\| \mathbf{d}(exp(\hat{\xi})P) - D_t(\pi(exp(\hat{\xi})P)) \right\| / |\mathbf{N}_r| \\ \mathscr{E}_c = \sum_{\mathbf{x}\in\mathbf{N}_r} \left\| I_r(\mathbf{x}) - I_t(\pi(exp(\hat{\xi})P)) \right\| / |\mathbf{N}_r| \end{array}, \qquad (1)$$

where ξ represents the 6DoF pose of the current frame to be estimated by ICP, and $exp(\hat{\xi})$ is its Lie Algebra format. $\mathbf{N}_r$ is the set of pixels with valid depths inside the object region of F_r. For each depth $d \in \mathbf{N}_r$ at pixel $\mathbf{x} = (u, v)$ inside the object region, we project it back to get a global 3D space point by $P = \mathbf{M}_r{}^{-1}\rho(u, v, d)$, where $\rho(u, v, d) = \left(\frac{u-c_u}{f_u}d, \frac{v-c_v}{f_v}d, d\right)$ is the back projection function, with (f_u, f_v) the focal lengths in u and v directions, and (c_u, c_v) the optical center. $\pi(x, y, z) = \left(\frac{x}{z}f_u + c_u, \frac{y}{z}f_v + c_v\right)$ is the projection function. I_r is the gray image of reference frame F_r, while I_t and D_t are the gray image and depth map at time t. $\mathbf{d}(exp(\hat{\xi})P)$ represents the depth of P at time t transformed by $exp(\hat{\xi})$.

However, ICP tracking is sensitive to depth errors or over-smoothness, which are especially common for input depths of iPad Pro, like the misalignment cases of “David” and “Worker” shown in Fig. 3(a). Symmetric structures such as spherical and cylindrical shapes will also affect tracking robustness. Although IMU module can provide us a predicted pose $\mathbf{M}_t^P$ of the current frame, this estimated pose prior is usually unreliable when the IMU state is not well initialized or optimized. Fortunately, according to our observation, the rotation part of the pose predicted by the IMU module and the gravity are always reliable. Therefore, we add rotation and gravity contraints as our new ICP energy terms as follows:

$$\begin{array}{l} \mathscr{E}_r = ln \left\| exp(\hat{\omega})\mathbf{R}_t^{P^{-1}} \right\| \\ \mathscr{E}_g = \left\| G_t - exp(\hat{\omega})\mathbf{R}_r{}^{-1} G_r \right\| \end{array}, \qquad (2)$$

where ω is the rotation components of the ξ to be estimated. $\mathbf{R}_t^P$ and $\mathbf{R}_r$ are the rotation part of $\mathbf{M}_t^P$ and $\mathbf{M}_r$ respectively. $\|\cdot\|$ represents L2-norm. G_t and G_r are the gravities in the camera coordinates of F_t and

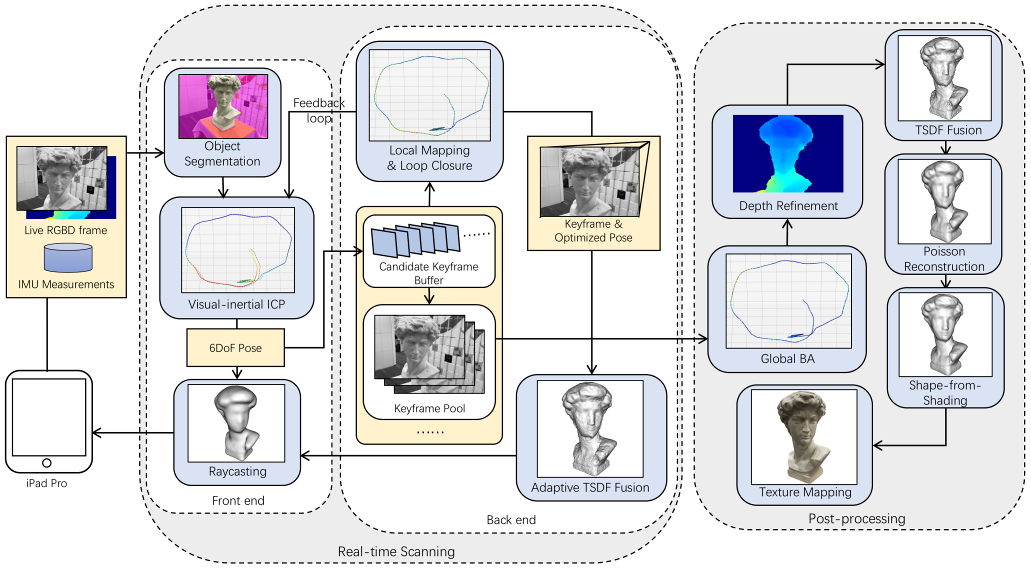

Fig. 2. System framework, which shows the real-time scanning stage including object segmentation, VI-ICP and local mapping with adaptive TSDF fusion, and the post-processing stage including the global refinement of keyframe poses and depths, final TSDF fusion, SFS, and texture mapping.

F_r respectively, acquired from the mobile device directly. In this way, our VI-ICP approach minimizes the following energy function:

$$\begin{array}{l}\mathscr{E}_{ICP} = \mathscr{E}_d + \lambda_c \mathscr{E}_c + \lambda_g \mathscr{E}_g + \lambda_r \mathscr{E}_r \\ \xi = \underset{\xi}{\arg\min}(\mathscr{E}_{ICP})\end{array}, \quad (3)$$

with weights $\lambda_c = 0.03$, $\lambda_g = 0.04$, and $\lambda_r = 0.04$. The energy minimization is solved iteratively by Gauss-Newton method similar to [49], with a coarse-to-fine multi-level pyramid scheme for speed-up. We use 3 levels and 15 iterations for each level in our experiment, with each iteration further accelerated by multi-thread parallel, to ensure real-time on the mobile platform. With the estimated ξ, we get the 6DoF pose matrix $\mathbf{M}_t = exp(\hat{\xi})$ for the current frame. Fig. 3(b) shows the effectiveness of our VI-ICP. We can see that the tracking errors are significantly reduced by introducing the rotation and gravity contraints, compared to the original ICP approach in Fig. 3(a).

After the ICP tracking finishes, we project all the pixels with valid depth of F_t to F_r to evaluate the ICP alignment. The pixels whose depth difference is within 7mm and color difference is within 30 are considered as inliers. The tracking fails if the outlier ratio exceeds 0.4 and current frame will be discarded. If ICP has five failures continuously, the system is lost and will trigger a global relocalization module. For each successfully tracked frame, we check the pose similarity of F_t and F_r. If the distance of view positions between $\mathbf{M}_t$ and $\mathbf{M}_r$ exceeds $\delta_p = 3cm$, or the difference of view angles between them exceeds $\delta_d = 1.5°$, we consider F_r no longer suitable as the reference frame, and replace it with F_t for future ICP tracking.

4.1.3 Local Mapping and Loop Closing

Although the tracking accuracy can be improved by our VI-ICP, tracking errors are still accumulated when we scan around a large object through a long distance, which will obviously affect the fused 3D model. Therefore, a local mapping module is necessary to further reduce accumulated errors by local pose optimization, with a loop closing module for global pose optimization on the back end.

On the back end, we maintains a candidate keyframe buffer which contains the historical reference frames, and a keyframe pool which consists of all the keyframes with poses and 3D map points refined by the local mapping module. When a reference frame is replaced with a new one as mentioned in Section 4.1.2, the old reference frame becomes a candidate keyframe for local mapping, and is inserted into the candidate keyframe buffer. Then, the local mapping thread is activated to refine the keyframe poses within a sliding window of no more than 6 keyframes. It continuously pops the front candidate from the candidate keyframe buffer, checks its pose similarity to the latest keyframe in the sliding window with thresholds $2\delta_p$ and $2\delta_d$, and decides whether to involve it as a new keyframe for pose optimization or discard it. If it becomes a new keyframe K_t, ORB features are extracted inside its object region, and matched with the projections of the existing map points from the sliding window to its local camera space. A local BA is performed on K_t and all the existing map points and keyframes in the sliding window matched with K_t for further pose optimization.

We improve the local BA of ORB-SLAM2 [21] by using keyframe depths as constraints. Our energy function contains a reprojection error term $\mathbf{E}_x$ and an inverse depth prior term $\mathbf{E}_d$ defined as follows:

$$\begin{array}{l}\{\mathbf{X}, \mathbf{M}_l | \mathbf{X} \in \mathbf{P}_L, K_l \in \mathbf{K}_L\} = \underset{\mathbf{X}, \mathbf{M}_l}{\arg\min} (\mathbf{E}_x + \lambda_d \mathbf{E}_d) \\ \mathbf{E}_x = \sum_{\mathbf{X} \in \mathbf{P}_L} \sum_{K_l \in \mathbf{K}_L} \|\mathbf{x}_l - \pi(\mathbf{M}_l \mathbf{X})\|_\delta \\ \mathbf{E}_d = \sum_{\mathbf{X} \in \mathbf{P}_L} \sum_{K_l \in \mathbf{K}_L} \|1/D_l(\mathbf{x}_l) - 1/\mathbf{d}(\mathbf{M}_l \mathbf{X})\|_\delta \end{array}, \quad (4)$$

where $\mathbf{P}_L$ is the set of 3D map points in the sliding window matched with K_t, and $\mathbf{K}_L$ denotes the set of keyframes in the sliding window which have common feature correspondences with K_t plus K_t itself. $\mathbf{M}_l$ is the global-to-local pose matrix of K_l, and $D_l(\mathbf{x}_l)$ is the depth measurement at pixel $\mathbf{x}_l$, which is the 2D feature correspondence of map point $\mathbf{X}$ at keyframe K_l. $\pi(\cdot)$ is the projection function, and $\mathbf{d}(\mathbf{M}_l \mathbf{X})$ represents the projection depth of $\mathbf{X}$ at keyframe K_l. λ_d is the depth prior weight and is empirically set to 5. $\|\cdot\|_\delta$ is the robust Huber cost function. We use Ceres Solver [34] for this energy minimization. After local BA finishes, we count the inliers of correspondences with reprojection error within 3 pixels. If the number of inliers is less than 20 or the inlier ratio is below 0.6, we destroy the current keyframe, clear the candidate keyframe buffer and set the system to lost status. Otherwise, we add K_t and its new 3D map points to the sliding window and the keyframe pool, with the changed 3D map points and keyframe poses in $\mathbf{P}_L$ and $\mathbf{K}_L$ updated to the keyframe pool. If the sliding window is full, the earlist keyframe is popped out.

Meanwhile, we adopt the method of ORB-SLAM2 [21] for loop detection and closure on the loop closing thread. The effectiveness of local mapping and loop closing are demonstrated in Fig. 3(d) and (e). As can be seen in the camera trajectories and the fused 3D models, local BA and loop closure significantly improve the accuracy of keyframe poses. We can also seen from Fig. 3(c-d) that depth constraints are necessary for local mapping in reducing the tracking errors.

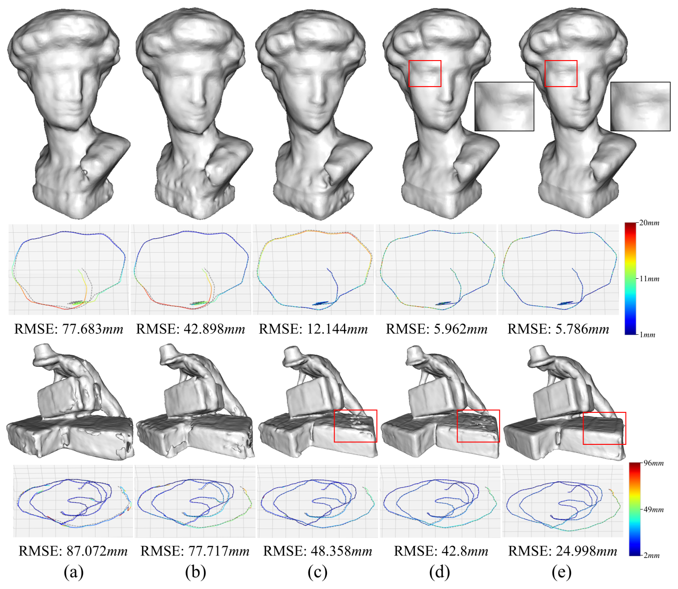

Fig. 3. ICP tracking results on cases "David" and "Worker": (a) The keyframe pose trajectories of original ICP in Open3D [49] and their fused 3D models. (b) Our VI-ICP involving IMU and gravity. (c) Our VI-ICP combined with local mapping without depth constraints. (d) Our VI-ICP combined with local mapping under depth constraints. (e) Our VI-ICP combined with both local mapping and loop closure. All the pose trajectories are compared to ground truth (GT) computed by COLMAP [35] in black color, to give quantitative accuracies in Absolute Trajectory Root Mean Squared Error (AT RMSE) [37] that aligns the two trajectories and evaluates the absolute pose differences. The tracking error of each keyframe is visualized in its mapped color.

4.1.4 360-degree Reconstruction

Our system supports 360-degree reconstruction of a movable object by allowing the user to pause the scanning, lay down the object, and continue to scan its bottom. The tracking is continued by adopting a pause-and-restart strategy similar to [44]. We relocalize the current frame by finding the most similar keyframe in the keyframe pool using DBoW2. SVD decomposition is used to initialize the current 6DoF pose by 3D-3D correspondences acquired from ORB feature matching and depth back-projection, followed by an ICP based pose refinement. If ICP has sufficient inliers, we consider the relocalization successful and allow the user to continue the scanning normally. The current frame is inserted as a new keyframe to both the sliding window and the keyframe pool, with new feature correspondences connected to the other keyframes added for further local BA.

4.2 Temporal Consistent Object Segmentation

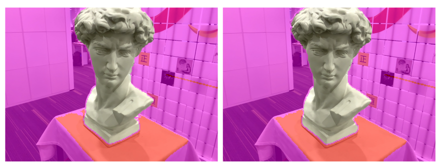

Fig. 4. Object segmentations of two frames in case "David", with red region denoting planar pixels and background pixels highlighted in purple.

As mentioned in Section 4.1, both ICP tracking and local mapping focus on the object region to speed up tracking and reduce the influence of depth calibration error on tracking accuracy. Therefore, a temporal consistent object segmentation is required to run simultaneously with tracking on the mobile platform in real-time. We use a 3D plane tracking scheme for temporal consistent segmentation of the object.

For the first frame, we fit a plane on the point cloud with normals, which is acquired by back projecting the valid depths of the whole frame to local camera space. Since we have the gravity in the local camera coordinates, a RANSAC based plane fitting algorithm is carried out, which iteratively selects a point with normals consistent with gravity as seeds to fit a candidate horizontal plane perpendicular to the gravity, and collects a set of inliers with distances to the candidate plane less than $1cm$, to evaluate the fitness of the candidate. Therefore, we only solve a variable of plane height in the opposite direction of gravity during the whole RANSAC process. The 3D equation of candidate plane with the best fitness of inliers is maintained as a global plane for further plane tracking. The object region is segmented out by eliminating the inlier pixels of the plane, the pixels with 3D positions under the plane, and those with invalid or faraway depths, followed by a morphological open operation and a maximal connected component extraction. We consider depths with more than $3m$ too faraway.

For each of a new incoming frame, the global plane is projected to the local camera space of its previous frame with its tracked 6DoF pose to get a plane height in the opposite gravity direction, which is used as a prior constraint for RANSAC plane fitting of the current frame. We constrain the selection of the seeds within a distance range of $5cm$ from the projected plane height prior, so that the current plane height fitted by RANSAC is consistent with the global plane. The object region is segmented in the same way as the first frame, which ensures temporal consistency. After the current frame pose is tracked, the fitted plane height in the local camera space is converted to global 3D space to update the global plane equation, considering the slight change of the global plane due to the accumulated tracking error on the front end.

Our object segmentation performs in real-time on iPad Pro to keep up with the frequency of pose tracking. Fig. 4 shows the results of object segmentation of two sequential frames with satisfactory temporal consistency. The segmented object regions are also used for the following TSDF fusion and depth refinement stages, to make sure that only the object depths are refined and fused to form the final 3D model.

4.3 TSDF Fusion with Adaptive Voxel Resizing

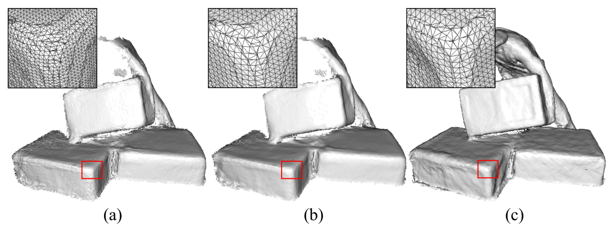

Fig. 5. Our adaptive voxel resizing on case "Worker": (a) The mesh extracted from the TSDF volume before voxel resizing. (b) The volume mesh after triggering voxel resizing. (c) The mesh of the final TSDF volume after fusing all the keyframes.

While performing the visual-inertial tracking and object segmentation, the depths inside the object region of each keyframe are fused by its tracked 6DoF pose into a global TSDF model in another fusion thread on the back end. Simultaneously, the fused TSDF volume is raycasted by the viewing pose on the front end for a normal shader based real-time rendering of the currently reconstructed model, allowing the user to preview which parts of the object are scanned.

TSDF fusion is a volumetric method for integrating range images to ensure incremental updating, representation of directional uncertainty, reconstruction gap filling, and robustness in the presence of outliers, and has demonstrated its effectiveness in literature [3, 24]. However, its huge memory consumption prevented further applications to large object reconstruction. Although voxel hashing [26] is employed in [3] to break through the memory limitation of TSDF fusion to some extent, the memory problem still occurs when a large object is being scanned and fused on a mobile device with gradually increasing volume occupancy. Larger voxel size can represent the object with a smaller number of voxels with less memory cost on the mobile platform, but will seriously affect the object reconstruction quality. Moreover, we do not know the best voxel size for the object scale to balance the memory limitation and reconstruction accuracy before we scan it. To better overcome this issue, we propose an adaptive voxel resizing strategy, so that users are able to use a mobile device with limited memory to scan as-large-as-possible objects with adaptive voxel resolution.

Our TSDF fusion follows the voxel hashing strategy adopted in [48]. Rather than allocating voxels for the entire volume, we only allocate voxels which are really occupied by the depth map fusion. To further speed up voxel allocation and hashing, we use sub-volume hashing for TSDF representation. Each sub-volume contains $16 \times 16 \times 16$ voxels, and is allocated or updated if any of its voxels is created or updated by depth fusion. The initial voxel size δ is set to $6mm$. Suppose we have the depth map D_t for keyframe K_t. For each depth $d \in D_t$ at pixel $\mathbf{x} = (u, v)$ inside the object region, we project it back to get a global 3D space point by $P = \mathbf{M}_t{}^{-1}\rho(u, v, d)$, where $\mathbf{M}_t$ is the global-to-local transformation at time t, and $\rho(\cdot)$ is the back projection function. Each voxel $\mathbf{V}$ inside the sub-volumes occupied by the truncation band $[-\tau, \tau]$ of P is created or updated as follows:

$$\begin{aligned} T_t(\mathbf{V}) &= \frac{T_{t_p}(\mathbf{V})W_{t_p}(\mathbf{V}) + (D_t(\pi(\mathbf{M}_t\mathbf{V})) - \mathbf{d}(\mathbf{M}_t\mathbf{V}))S(\pi(\mathbf{M}_t\mathbf{V}))/\tau}{W_{t_p}(\mathbf{V})+1} \\ W_t(\mathbf{V}) &= W_{t_p}(\mathbf{V}) + 1 \end{aligned}, \tag{5}$$

where $\mathbf{d}(\mathbf{M}_t\mathbf{V})$ represents the projection depth of $\mathbf{V}$ at keyframe t, and $D_t(\pi(\mathbf{M}_t\mathbf{V}))$ is the depth measurement at pixel $\pi(\mathbf{M}_t\mathbf{V})$. $S(u, v) = \sqrt{((u - c_u)/f_u)^2 + ((v - c_v)/f_v)^2 + 1}$ converts the depth difference at pixel (u, v) to distances in camera space, with focal lengths f_u, f_v and optical center c_u, c_v. $\pi(\cdot)$ is the projection function. $T_t(\mathbf{V})$ and $W_t(\mathbf{V})$ represent the TSDF value and weight of $\mathbf{V}$ respectively at time t. $T_{t_p}(\mathbf{V})$ and $W_{t_p}(\mathbf{V})$ are the TSDF value and weight at the previously updated time t_p. We set $\tau = 3cm$. For a newly generated voxel, $T_t(\mathbf{V}) = (D_t(\pi(\mathbf{M}_t\mathbf{V})) - \mathbf{d}(\mathbf{M}_t\mathbf{V}))S(\pi(\mathbf{M}_t\mathbf{V}))/\tau$ and $W_t(\mathbf{V}) = 1$.

On a mobile device, the memory usage of TSDF should be kept under an upper limitation $\hat{M}$, which we set to $200MB$ for scanning preview on the iPad Pro platform. When the memory cost M_t exceeds this limitation after a keyframe depth map fusion at time t, a voxel resizing is triggered by recreating a new TSDF volume to represent the object with a larger voxel size $\delta' = 1.5\delta$ for memory reduction. We allocate new sub-volumes according to the new voxel size. Each new sub-volume should fully contain at least one old voxel. For a newly created voxel $\mathbf{V}'$, its new TSDF value is calculated by trilinear interpolation of the old voxels as:

$$T_t(\mathbf{V}') = \sum_{\mathbf{V} \in N(\mathbf{V}')} \frac{(\delta - |\mathbf{V}'_x - \mathbf{V}_x|)(\delta - |\mathbf{V}'_y - \mathbf{V}_y|)(\delta - |\mathbf{V}'_z - \mathbf{V}_z|)}{\delta^3} T_t(\mathbf{V}), \tag{6}$$

where $N(\mathbf{V}')$ is the 8 nearest neighboring old voxels of $\mathbf{V}'$. The new weight $W_t(\mathbf{V}')$ can be calculated in the same interpolation way as Eq. (6). Through this trilinear interpolation, the voxels can be dynamically resized efficiently without recalculating their TSDF values and weights by Eq. (5). An intuitive example of this adaptive voxel resizing in case "Worker" is shown in Fig. 5, where the $4mm$ TSDF voxels in (a) are resized to $6mm$ in (b). Actually, the voxel resizing is triggered iteratively whenever the memory limitation is reached, until the object scanning is completed. A similar voxel sizing strategy is applied for TSDF fusion at the post-processing stage to handle memory limitation, which will be described in Section 5.2 with more details.

Both TSDF fusion and voxel resizing are accelerated by OS Metal GPU. For case "Worker", a TSDF fusion takes 5.5 ms/keyframe and a voxel resizing costs 65.51 ms at a time averagely on iPad Pro. The fused TSDF volume is raycasted to the current view for real-time visualization of the scanned surface to the user, in the same way as [24].

5 Object Model Post-processing

In this post-processing stage, since we have the keyframe pool on the back end with all the keyframes and their corresponding 3D mappoints, we can use these keyframes to further optimize the geometric accuracy of the object model, and fulfill texture mapping for the optimized model, to finally create a high-quality textured 3D model. We will describe our object model post-processing in key details.

5.1 Global Optimization of Keyframe Poses and Depths

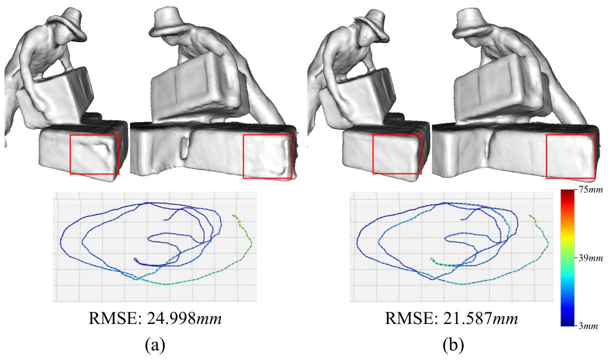

Fig. 6. Global BA of case "Worker": (a) The pose trajectory of all the keyframes optimized online by local BA and its fused 3D model. (b) The optimized keyframe pose trajectory and its finally fused 3D model after global BA, with GT given by COLMAP [35].

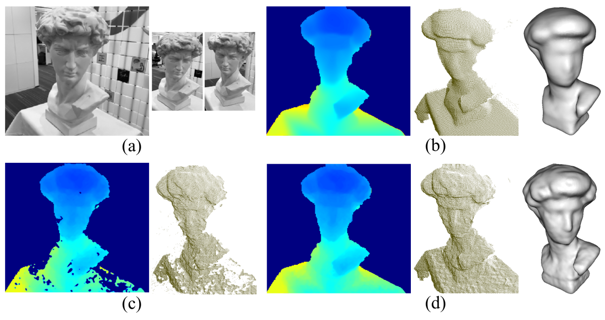

Fig. 7. Depth refinement by MVS on case "David": (a) A keyframe and its two reference ones. (b) The original depth map of the keyframe from dToF, its point cloud, and the finally fused 3D model. (c) The estimates depth map by MVS without dToF depth priors, and its point cloud. (d) The refined depth map by MVS with dToF depths as prior, its point cloud, and the finally fused 3D model of all the refined keyframe depths.

Our global BA uses the same energy function as local BA by Eq. (4), except that all the keyframes and map points in the keyframe pool participate in the global optimization. Note that the first frame is fixed during the optimization to keep the world coordinates unchanged. The effectiveness of the global BA on case "Worker" is shown in Fig. 6, with the slight pose registration drift thoroughly reduced to improve the accuracies of both tracking and geometry fusion.

Although global BA can help reducing the object pose alignment errors, directly using the optimized poses and the input depths for 3D model fusion is insufficient for a high-quality reconstruction purpose, because the input depths from consumer RGBD camera such as dToF on iPad Pro might have depth errors or over-smoothness with lost geometric details, as can be seen in Fig. 7(b). To solve this problem, we propose to estimate more accurate depths with geometric details for all the keyframes by MVS, since we have multi-view keyframes with globally optimized poses. For each keyframe, we propose to refine the depth measurements from dToF sensor using an SGM approach, which is widely used for binocular depth estimation and MVS problems like in [9,45]. We improve the multi-view SGM approach in [45] by incorporating dToF depth priors into cost aggregation to exploit the complementary advantages of MVS and dToF, according to our observation that depths from MVS is more accurate but noisy in textureless regions while depths from dToF are more complete but lost in details.

Suppose the depth measurement is bounded to a range from $d_{\min}$ to $d_{\max}$, which can be acquired from a predefined valid depth range of dToF sensor. We uniformly sample the inverse depth space to L levels, and the l-th sampled depth can be computed as follows:

$$d_l = \frac{(L-1)d_{\min}d_{\max}}{(L-1-l)d_{\min} + l\,(d_{\max} - d_{\min})}, \tag{7}$$

where $l \in \{0,1,2...,L-1\}$, and d_l is the sampled depth at the l-th level. Given a pixel $\mathbf{x} = (u,v)$ with depth d_l in the object region of keyframe K_t, its projection pixel $\mathbf{x}_{t\to t'}(d_l)$ on a reference keyframe $K_{t'}$ by d_l can be calculated by $\mathbf{x}_{t\to t'}(d_l) = \pi(\mathbf{M}_{t'}\mathbf{M}_t{}^{-1}\rho(u,v,d_l))$, where $\mathbf{M}_t$, $\mathbf{M}_{t'}$ are the 6DoF pose matrices of keyframe K_t and $K_{t'}$, with $\pi(\cdot)$ and $\rho(\cdot)$ the projection and back projection functions respectively. We resort to a variant of Census Transform (CT) [8] as the feature descriptor to compute patch similarity cost. Meanwhile, considering the dToF depth measurements have higher completeness, we combine these depths as priors to compute weights for multi-frame cost fusion. Therefore, our matching cost is determined as follows:

$$\begin{aligned} C(\mathbf{x},d_l) &= w(\mathbf{x},d_l)\sum_{t'\in N(t)} CT(\mathbf{x},\mathbf{x}_{t\to t'}(d_l)) \\ w(\mathbf{x},d_l) &= 1 - \mathcal{N}_\sigma(d_l - D_t(\mathbf{x})) \end{aligned}, \tag{8}$$

where $N(t)$ is the set of reference keyframes for K_t, which are selected by the strategy in [45]. $w(\mathbf{x},d_l)$ is the cost fusion weight using a Gaussian function with variance $\sigma = 0.45$, and $D_t(\mathbf{x})$ is the dToF depth prior. $CT(\mathbf{x},\mathbf{x}_{t\to t'}(d_l))$ is the Census cost of the two patches centered at $\mathbf{x}$ and $\mathbf{x}_{t\to t'}(d_l)$. We only compute costs of the pixels inside the object region of K_t. Therefore, by Eq. (8), we get a cost volume C with size $W \times H \times L$, where W and H are width and height of the object region bounding box. After that, the cost volume is aggregated along various directions using the method proposed in [9, 10], to generate a more reliable matching cost volume $\hat{C}$.

The final depth label $\hat{l}(\mathbf{x})$ is given by a Winner-Take-All strategy to select the depth level with the lowest cost in cost volume $\hat{C}$. In order to get a sub-level depth value, we follow the method in [45] by using parabola fitting to acquire a refined depth level $\hat{l}_s(\mathbf{x})$. We substitute l in Eq. (7) with $\hat{l}_s(\mathbf{x})$ to get a more accurate sub-level depth for $\mathbf{x}$. With the sub-level depths, we get an depth map $\hat{D}_t$ for each keyframe K_t. As can be seen in Fig. 7(d), the object depths refined by our SGM with dToF depth priors contain more geometric details than the dToF depth measurements shown in Fig. 7(b), and are more complete in textureless regions compared to results of Yang et al. [45] shown in Fig. 7(c). In this way, advantages of MVS and dToF depths are combined to provide more accurate depths for final mesh generation.

5.2 Mesh Generation with Detail Enhancement

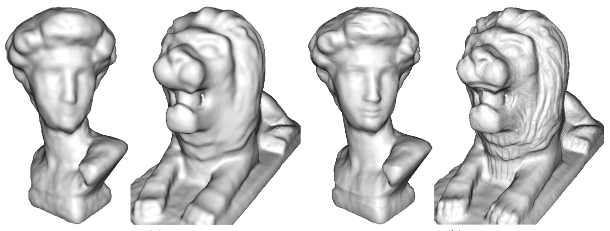

Fig. 8. SFS results on examples of "David" and "Lion": (a) The 3D models by PSR. (b) The 3D models with more details after SFS.

With the optimized keyframe poses and depths, a final TSDF fusion is performed for better geometric accuracy than the real-time preview one at the scanning stage. A larger memory limitation $\hat{M}_p = 400MB$ is adopted for this mesh generation stage, since the memory pressure is much relieved after the scanning and global optimization. Besides,

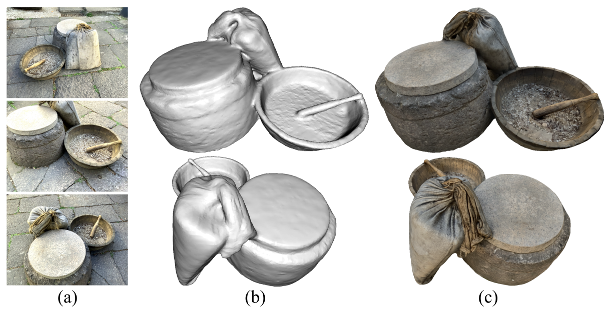

Fig. 9. Texture mapping of case "Farm": (a) shows three of the representative keyframes for texturing. (b) The input 3D model. (c) The final 3D model after our multi-view texture mapping.

it is also feasible to estimate a proper voxel size δ_p for the final TSDF volume considering the memory usage of the preview one as: $\delta_p = \max(\delta_{min}, \delta_s/\sqrt[3]{\eta\hat{M}_p/M_s})$, where δ_s and M_s denote the voxel size and memory cost of the preview TSDF volume at the end time respectively, and we set $\delta_{min} = 4mm$. η is a protection coefficiency to make sure the memory usage will not reach $\hat{M}_p$, with $\eta = 0.9$ in our experiments. With the estimated voxel size, the optimized depths are fused to the final TSDF volume using Eq. (5), with Marching Cubes [17] followed to extract a triangle mesh of the object.

The triangle mesh generated from TSDF volume is usually incomplete, because some local parts of the object with self-occlusion are hard to be scanned fully. The bottom surface is apparently missing if an immovable object cannot be laid down for bottom scanning. Therefore, PSR [13] is adopted to deal with the incompleteness, with its capability to fit watertight surfaces from the set of mesh vertices with normals. For an immovable large object, our algorithm reconstructs a presumptive bottom surface for it, by adding dense point samples on the 3D planar surface, with inverse plane normal as the sample normals. These additional sample points are combined with mesh vertices as the input for PSR, to obtain a complete 3D mesh of the object.

After TSDF fusion and PSR, we use SFS method to improve geometric details from shading clues. Although SFS is helpful for detail restoration, most existing implementations are far from feasible to mobile platform. For example, Intrinsic3D [19] refine the SDF in a coarse-to-fine manner on pyramid levels, which causes exponential growth of computation and memory. Instead, we propose a highly efficient SFS implementation for detail enhancement on mobile platform.

Most SFS works are performed on depth maps [43] or TSDF volume [19, 50], which require extra computation cost to extract the optimized triangle mesh. For time efficiency on a mobile device, we perform SFS directly on the mesh, by optimizing an intermediate triangle normal map, followed by updating the vertex positions according to the optimization of the normal map. The optimization framework is similar to that of [50], with the major difference lying in that we optimize triangle normals instead of TSDF. Our SFS based detail enhancement iteratively optimize face normals, face albedos and the SH coefficiencies, to make the estimated luminance closer to the observed irradiance. The energy function of each mesh triangle f is defined as:

$$E_{SFS}(f) = E_g(f) + \lambda_s E_s(f) + \lambda_r E_r(f) + \lambda_a E_a(f), \tag{9}$$

where $E_g(f) = (\nabla B(f) - \nabla I(f))^2$ is the gradient residual between luminance $B(f)$ and irradiance $I(f)$. $E_s(f) = ||\mathbf{n}(f) - \mathbf{n}_0(f)||^2$ is the normal stabilizer to keep the derived normal $\mathbf{n}(f)$ closer to the input one $\mathbf{n}_0(f)$, with weight $\lambda_s = 0.005$. $E_r(f) = \sum_{f'\in N(f)} ||\mathbf{n}(f) - \mathbf{n}(f')||^2$ is the normal regularizer to smooth the derived normals with weight $\lambda_r = 0.003$, and $E_a(f) = \sum_{f'\in N(f)} \phi(C(f) - C(f'))(A(f) - A(f'))^2$ is the albedo regularizer which helps to avoid texture-copy problem with weight $\lambda_a = 2$. Here, $N(f)$ is the neighboring triangles of f, $C(f)$ denotes the chromaticity, and $\phi(\cdot) = 1/(1+3||\cdot||)^3$ is the robust kernel.

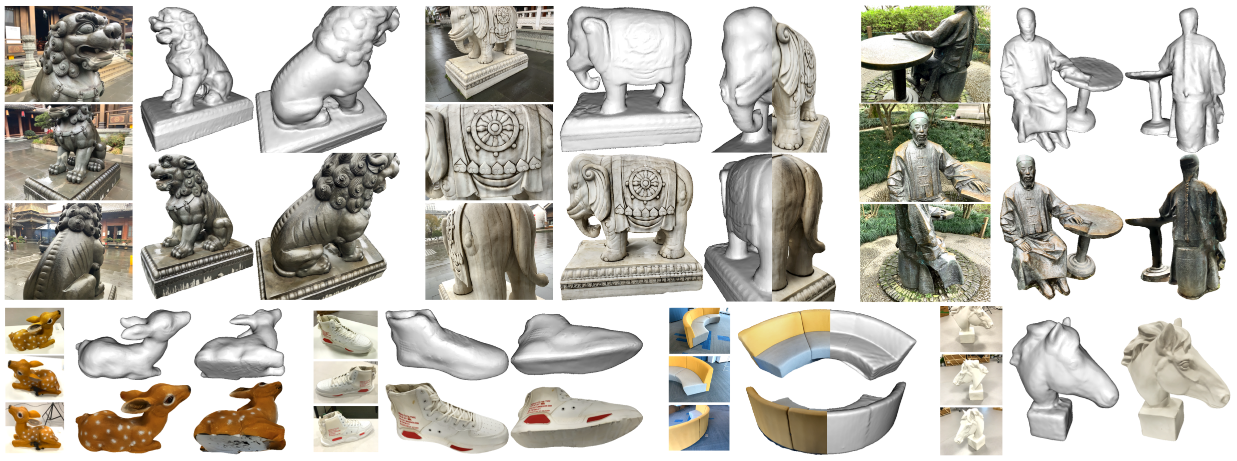

Fig. 10. The reconstructed 3D models of cases “Ancient Lion”, “Elephant”, “Qi Lin”, “Deer”, “Shoe”, “Sofa” and “Horse Head”, with three representative keyframes given for each case.

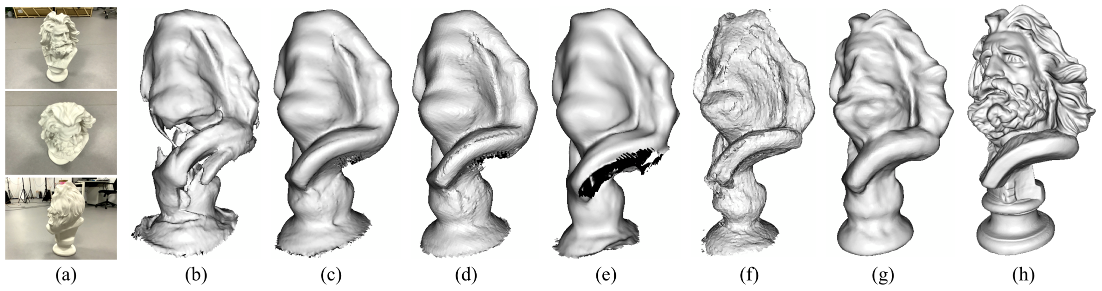

Fig. 11. Comparison of our Mobile3DScanner with other state-of-the-art methods: (a) Three representative keyframes in case “La Marseillaise”. (b) Open3D [49]. (c) KinectFusion [24]. (d) InfiniTAM [12]. (e) BundleFusion [3]. (f) 3D Scanner App. (g) Our Mobile3DScanner. (h) The GT model aqcuired by a commercial 3D scanner.

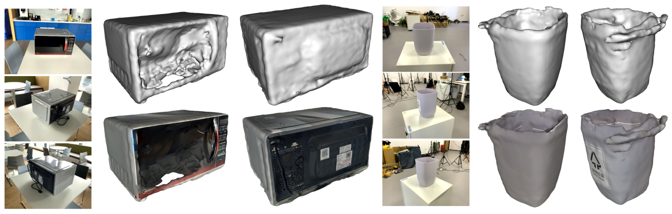

Fig. 12. Two failure cases of a microwave owen with reflecting surfaces, and a textureless trash can. Each case contains three representative keyframes, and the reconstructed 3D model with textures.

We follows the coarse-to-fine pyramid optimization strategy adopted in [19, 50], with mesh triangles upsampled by loop subdivision. The optimization framework contains three levels of pyramids. In the first level, the lighting coefficiencies are estimated, followed by optimization of albedos and normals twice. In each of the next two levels, we upsample the mesh triangles, update lighting coefficiencies on the subdivided mesh, and optimize albedos and normals once. Rather than using the conventional Gauss-Newton method, we utilize L-BFGS [16] for energy optimization, which is a light-weighted quasi-Newton method with line-search strategy. For each time of optimization, 30 iterations are sufficient to achieve a satisfactory result with significantly reduced time on the mobile device. Besides, only 25% triangles with the largest energies by Eq. (9) are upsampled for the next level optimization, so that the amount of triangles participating in optimization can be well controlled, which is essential for time efficiency on mobile platform. After the optimization, we use an efficient approach proposed in [47] to update vertex positions. Fig. 8 shows results of our SFS based mesh detail enhancement, which takes only 3.24 seconds for case “David” on iPad Pro CPU with Root Mean Squared Error (RMSE) 3.12mm and Mean Absolute Error (MAE) 2.473mm. Meanwhile, we run Intrinsic3D for “David” with exactly the same keyframes and 4mm as the voxel size, and keep all other parameters as default. Although Intrinsic3D turns out to preform slightly better with RMSE 2.89mm and MAE 2.16mm, it costs 81 minutes on an Intel Core i7 7700K CPU with 32GB RAM.

5.3 Multi-view Texture Mapping

We can use the color images of all the keyframes with optimized poses to perform texture mapping for the 3D model refined by SFS. We follow the approach in [41] to perform a multi-view texture mapping on the mobile device, with some improvements for time efficiency.

We select a set of representative keyframes for texture mapping to speed up both the data cost calculation and the graph-cuts optimization steps. Based on a prerequisite that we scan around the object, we can classify all the keyframes based on their view directions in polar angles. We defines $\Phi \times \theta = 6 \times 12$ direction bins, where Φ is the number of bins in elevation angle and θ is for azimuth angle. Each keyframe is classified into the corresponding bin by its polar angles. After all the keyframes are classified, a representative keyframe is chosen from each bin with the most similar colors in visible common regions with other keyframes in the bin, and all the representative keyframes compose the set of candidate texture frames. In this way, we have 72 candidate frames at most for texture mapping, which proves to be enough for most of the experimental cases. Besides, we use the sparse label costs proposed in [4] to further speed up graph-cuts, since each triangle is visible in only a small part of the candidate frames. An example “Farm” is demonstrated in Fig. 9 to show the effectiveness of our texture mapping, which costs 13.93s on an iPad Pro 2020.

Table 1. We report RMSEs and MAEs of the reconstruction results by our Mobile3DScanner, and [3, 12, 24, 49], and 3D Scanner App on our four experimental cases captured by iPad Pro, with each object scanned by a commercial 3D scanner as GT. To demonstrate the effectiveness of the post-processing stages, the accuracies without global BA and SFS (marked as "-GBA-SFS") and without SFS ("-SFS") are given separately.

RMSE/MAE [mm]	Open3D [49]	KinectFusion [24]	InfiniTAM [12]	BundleFusion [3]	3D Scanner App	Ours(-GBA-SFS)	Ours(-SFS)	Ours
Deer	7.507/5.698	21.156/13.152	4.905/3.476	8.248/5.306	7.8/6.501	4.701/3.159	4.564/3.024	**3.845/2.56**
David	10.021/8.417	8.532/7.472	4.579/3.568	5.877/4.555	8.389/7.172	3.186/2.525	3.153/2.513	**3.12/2.473**
La Marseillaise	16.334/11.689	9.543/7.796	6.015/4.77	5.818/4.517	8.208/6.448	4.181/3.289	4.179/3.27	**4.169/3.249**
Horse Head	12.793/9.855	5.24/4.311	5.059/3.836	4.911/3.919	8.032/6.868	4.037/2.88	3.937/2.864	**3.363/2.65**

Table 2. We report detailed computation time of our Mobile3DScanner in all the substeps of three cases "Deer", "La Marseillaise" and "Worker" on an iPad Pro 2020, which contain 487 frames, 1098 frames, and 1438 frames respectively.

Time	Real-time Scanning [ms/frame]						Post-processing [s]						
	Pose Tracking	Object Segmentation	TSDF Fusion	Voxel Resizing	Raycasting	Total	Global BA	Depth Refinement	TSDF Fusion	PSR	SFS	Texture Mapping	Total
Deer	12.62	3.64	1.76	/	2.49	18.75	0.27	16.85	0.275	0.733	1.63	4.39	24.15
La Marseillaise	15.72	3.29	1.89	/	2.69	21.7	1.17	26.56	0.446	1.17	3.31	6.76	39.42
Worker	21.87	4.19	5.5	65.51	3.32	29.38	1.75	33.23	1.381	1.9	5.64	20.31	64.21

6 Experimental Evaluation

In this section, we perform evaluation of our Mobile3DScanner, whose App is developed by iOS Object-C, with the core algorithms implemented in C++ code. We report quantitative comparisons as well as qualitative comparisons of our work with the state-of-the-art methods on our experimental benchmark captured by an iPad Pro 2020 with a rear dToF, which show that our Mobile3DScanner achieves the best performance on the benchmark. We also report the time consumption on each stage of our approach to show the efficient online 3D reconstruction using our system on iPad Pro.

6.1 Quantitative and Qualitative Evaluations

We qualitatively and quantitatively compare our Mobile3DScanner to other state-of-the-art methods on the generated 3D models of the twelves static objects captured by iPad Pro, including normal-sized movable objects like "Deer" with 30*cm* long, large movable objects like "David" with 63*cm* tall, and very large immovable objects such as "Lion" with 2.3*m* long. The input resolution of each case is 1920×1440 for image and 256×192 for depth, which are warpped and resized to 512×384 in our experiments. In Fig. 1 and 9, we have already demonstrated cases "Lion", "David", "Worker" and "Farm" in detail. Other cases are shown in Fig. 10. For cases "Deer", "David", "La Marseillaise" and "Horse Head", we compare our reconstructed 3D models against Open3D [49], KinectFusion [24], InfiniTAM [12], BundleFusion [3], and 3D Scanner App[1] (a third-party online app on iPad Pro). For 3D Scanner App, we choose object mode with the highest resolution of 5*mm*; for other methods, we use 4*mm* as the voxel size and 3*cm* as the truncation band, and leave all other parameters default. The GT models are scanned by a commercial digital 3D scanner for accuracy evaluation of each case. For model accuracy evaluation, we use CloudCompare [2] to compare the reconstructed meshes with GT: we align the mesh with GT using manual rough registration followed by ICP fine registration, then evaluate the mesh-point-to-GT-plane distances. This routine is achieved with CloudCompare's built-in functions. As the comparison results on case "La Marseillaise" shown in Fig. 11, both Open3D and 3D Scanner App have tracking drifts caused by the depth errors and over-smoothness from dToF, which significantly affect geometry of the final models. KinectFusion, InfiniTAM and BundleFusion have fewer tracking drifts, but do not guarantee geometry completeness. Besides, all these methods lack in geometric details due to the depth over-smoothness. In comparison, our system performs better than the other works in the finally generated 3D models with better geometric structure and fewer noisy arifacts. We can also see from the model accuracy evaluation in Table 1 that our Mobile3DScanner reconstructs the object models with a millimeter-level accuracy, which turns out to be the best in both RMSE and MAE.

[1]http://www.3dscannerapp.com/

[2]http://cloudcompare.org

Table 2 gives the time statistics of our pipeline on three typical cases "Deer", "La Marseillaise" and "Worker" with the longest dimension 30*cm*, 58*cm*, and 1.7*m* respectively, in stages of real-time 3D scanning and post-processing separately on iPad Pro with A12Z bionic chip. Since the TSDF fusion and voxel resizing both run on a back-end thread, the total time of the real-time scanning stage is actually the sum of all the front-end time costs from pose tracking, object segmentation and raycasting. We can see from the time consumptions that larger objects cost more time on pose tracking and TSDF fusion, and larger objects with more keyframes take more time on post-process. Even for the largest case "Worker" in Table 2 with the most keyframes, our Mobile3DScanner can still achieve real-time scanning and efficient post-processing for online 3D reconstruction on a mobile device, which makes it convenient to create 3D models as digitalized content for AR applications such as the example shown in the supplementary video.

6.2 Limitations

Since our local mapping and global BA rely on feature matching, the scanned object is required to contain lambertian features, which might be a limitation to textureless objects or reflecting surfaces. Tracking drift accumulates for these cases and is difficult to be corrected by loop or BA module due to the wrong feature matches cause by textureless or non-lambertian features. Besides, reconstructing reflecting surfaces is a well-known challenging problem [11]. Neither the depth sensor nor SGM is able to estimate accurate depths for these surfaces, which results in unsatisfactory reconstrcution results, as shown in Fig. 12.

7 Conclusion

We have presented a novel online 3D scanning system for 3D object reconstruction with a mobile device. Our system allows users to reconstruct high-quality dense textured 3D models of the scanned objects using a mobile device with an embedded RGBD camera. Unlike existing state-of-the-art methods which only support small object scanning due to the limited computation and memory on the mobile platform, our Mobile3DScanner utilizes an adaptive TSDF voxel resizing strategy to solve memory limitation of large object scanning. A novel visual-inertial ICP combined with local mapping ensures accurate object pose tracking, and the geometric details of the reconstructed model are refined through efficient optimization of poses, depths and geometry, to achieve high-quality object reconstruction. A more sophisticated tracking mechanism is preferred as a future work to better handle objects with textureless or non-lambertian surfaces. In addition, how to reconstruct high-quality objects online using a mobile device with only a monocular camera is a problem worth studying in the future.

Acknowledgments

The authors wish to thank Nan Wang, Chongshan Sheng, Li Zhou, Jianjun Xia, Jusong Zhou, and Di Zhang for their kind help in the development of the Mobile3DScanner system. This work was partially supported by NSF of China (Nos. 61822310 and 61932003).

References

[1] A. Bozic, M. Zollhofer, C. Theobalt, and M. Nießner. DeepDeform: Learning non-rigid RGB-D reconstruction with semi-supervised data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7002–7012, 2020.

[2] N. Brunetto, S. Salti, N. Fioraio, T. Cavallari, and L. Stefano. Fusion of inertial and visual measurements for RGB-D SLAM on mobile devices. In *IEEE International Conference on Computer Vision Workshops*, pp. 1–9, 2015.

[3] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 36(4):1, 2017.

[4] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

[5] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, et al. Fusion4D: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics*, 35(4):1–13, 2016.

[6] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.

[7] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai. Robust non-rigid motion tracking and surface reconstruction using L0 regularization. In *IEEE International Conference on Computer Vision*, pp. 3083–3091, 2015.

[8] M. Heikkilä, M. Pietikäinen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425–436, 2009.

[9] H. Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 807–814, 2005.

[10] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2007.

[11] I. Ihrke, K. N. Kutulakos, H. P. Lensch, M. Magnor, and W. Heidrich. Transparent and specular object reconstruction. In *Computer Graphics Forum*, vol. 29, pp. 2400–2426. Wiley Online Library, 2010.

[12] O. Kahler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1–1, 2015.

[13] M. Kazhdan. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, 2006.

[14] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3D scanners. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[15] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger. Dense RGB-D-inertial SLAM with map deformations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6741–6748. IEEE, 2017.

[16] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[17] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[18] S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 25:113–118, 2010.

[19] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner. Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *IEEE International Conference on Computer Vision*, 2017.

[20] P. Merrell, A. Akbarzadeh, W. Liang, P. Mordohai, J. M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *IEEE International Conference on Computer Vision*, pp. 1–8, 2007.

[21] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[22] O. Muratov, Y. V. Slynko, V. V. Chernov, M. M. Lyubimtseva, A. Shamsuarov, and V. Bucha. 3DCapture: 3D reconstruction for a smartphone. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 893–900, 2016.

[23] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 343–352, 2015.

[24] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, and A. W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[25] M. Nießner, A. Dai, and M. Fisher. Combining inertial navigation and icp for real-time 3D surface reconstruction. In *Eurographics*, pp. 13–16, 2014.

[26] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6CD):169.1–169.11, 2013.

[27] P. Ondrúška, P. Kohli, and S. Izadi. MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1251–1258, 2015.

[28] B. Paul J. and M. Neil D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[29] F. Poiesi, A. Locher, P. Chippendale, E. Nocerino, F. Remondino, and L. Van Gool. Cloud-based collaborative 3D reconstruction using smartphones. In *Proceedings of the 14th European Conference on Visual Media Production*, pp. 1–9, 2017.

[30] M. Pollefeys, D. Nister, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, et al. Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision*, 78(2):143–167, 2008.

[31] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *IEEE International Symposium on Mixed and Augmented Reality*, 2013.

[32] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[33] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, 2002.

[34] K. M. S. Agarwal and Others. Ceres solver. [EB/OL]. `http://ceres-solver.org`.

[35] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113, 2016.

[36] T. Schöps, T. Sattler, C. Hne, and M. Pollefeys. Large-scale outdoor 3D reconstruction on a mobile device. *Computer Vision and Image Understanding*, 157:151–166, 2017.

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

[38] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. 26(3):80–es, 2007.

[39] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3D reconstruction on mobile phones. In *IEEE International Conference on Computer Vision*, pp. 65–72, 2013.

[40] D. Tzionas and J. Gall. 3D object reconstruction from hand-object interactions. In *IEEE International Conference on Computer Vision*, pp. 729–737, 2015.

[41] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! large-scale texturing of 3D reconstructions. In *European Conference on Computer Vision*, pp. 836–850. Springer, 2014.

[42] T. Weise, B. Leibe, and L. Van Gool. Accurate and robust registration for in-hand modeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[43] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics*, 33(6):1–10, 2014.

[44] J. Xu, W. Xu, Y. Yang, Z. Deng, and H. Bao. Online global non-rigid registration for 3D object reconstruction using consumer-level depth cameras. 37(7):1–12, 2018.

[45] X. Yang, L. Zhou, H. Jiang, Z. Tang, Y. Wang, H. Bao, and G. Zhang. Mobile3DRecon: Real-time monocular 3D reconstruction on a mobile phone. *IEEE Transactions on Visualization and Computer Graphics*, 26(12):3446–3456, 2020.

[46] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. Recurrent MVS-Net for high-resolution multi-view stereo depth inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5525–5534, 2019.

[47] H. Zhao, K. Su, C. Li, B. Zhang, L. Yang, N. Lei, X. Wang, S. J. Gortler, and X. Gu. Mesh parametrization driven by unit normal flow. *Computer Graphics Forum*, 39(1):34–49, 2020.

[48] Q. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):1–8, 2013.

[49] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[50] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics*, 34(4):1–14, 2015.

[51] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics*, 33(4):1–12, 2014.