Online Reconstruction of Indoor Scenes from RGB-D Streams

Hao Wang*, Jun Wang*, Liang Wang
Baidu Research - Institute of Deep Learning
{wanghao29, wangjun21, wangliang18}@baidu.com

Abstract

A system capable of performing robust online volumetric reconstruction of indoor scenes based on input from a handheld RGB-D camera is presented. Our system is powered by a two-pass reconstruction scheme. The first pass tracks camera poses at video rate and simultaneously constructs a pose graph on-the-fly. The tracker operates in real-time, which allows the reconstruction results to be visualized during the scanning process. Live visual feedbacks makes the scanning operation fast and intuitive. Upon termination of scanning, the second pass takes place to handle loop closures and reconstruct the final model using globally refined camera trajectories. The system is online with low delay and returns a dense model of sufficient accuracy. The beauty of this system lies in its speed, accuracy, simplicity and ease of implementation when compared to previous methods. We demonstrate the performance of our system on several real-world scenes and quantitatively assess the modeling accuracy with respect to ground truth models obtained from a LIDAR scanner.

1. Introduction

The paper is about generating dense models of rigid indoor scenes using inputs streamed from a handheld RGB-D camera. The reconstruction is performed online with low delay as the sensor moves. The system also continuously estimates the 6 degrees of freedom (6DOF) pose of the sensor and returns the globally optimized camera trajectory. An example is shown in figure 1. In computer vision, structure from motion (SFM) and visual odometry have been highly active research topics for a long time. In the robotics community, closely related is what is known as simultaneous localization and mapping (SLAM). Seminal works [20, 21, 5, 14] demonstrate that robust tracking and sparse 3D mapping is possible in real-time from visual input alone. Attempts at real-time dense surface modelling from passive cameras to date have had limited results, mostly on outdoor scenes [23, 30], small objects and compact workspaces [16, 18, 24].

Dense modeling research has recently experienced somewhat of a new era, as a result of the emergence of consumer-level depth sensors. The availability of real-time, reliable depth sensing capabilities together with the advance of general purpose graphics processing units (GPGPU) open new avenues for camera tracking and dense mapping. The pioneer work in RGB-D reconstruction literature is KinectFusion [17]. Since [17], many methods (both online and offline) have been proposed to address the RGB-D scene reconstruction problem [34, 12, 3, 19, 38, 36, 13, 37, 35, 4, 32], from graphics, vision and robotics communities.

Previous work on RGB-D reconstruction can be categorized into two groups based on the method's emphasis and target application. Robotics researchers in general formulate the problem in a SLAM framework [8, 34, 12, 13, 31, 32], minimizing the absolute trajectory error (ATE) is their main focus and the root mean square error (RMSE) of the ATE is used as the evaluation metric. These SLAM systems are seldom designed for high fidelity modeling, thus mesh quality and geometry details are not fully appreciated. By contrast, graphics and vision communities focus more on the fidelity of the reconstructed model [36, 38, 37, 35, 4]. Qualitative or quantitative analysis of 3D models is performed to assess the surface quality. Note that to obtain high quality meshes, expert users are required to operate the sensor and image surfaces at close range. The scanning process is inherently different from some of the dense SLAM applications, in which a robot drives a rigidly mounted camera to create a dense map for an unknown environment.

Our work belongs to the second category, i.e., meshes are the main outcome of our system and high accuracy is desired in our applications. While methods for modeling small objects are relatively mature [29, 17], modeling indoor scenes automatically still remains a tough challenge. To one part this is because real-world environments contain sophisticated structures and must be reconstructed from complex camera trajectories, with each view covering only a small portion of the scene, making tracking prone to gross failure. More importantly, scanning large environments suffers

*indicates equal contributions and joint first authors

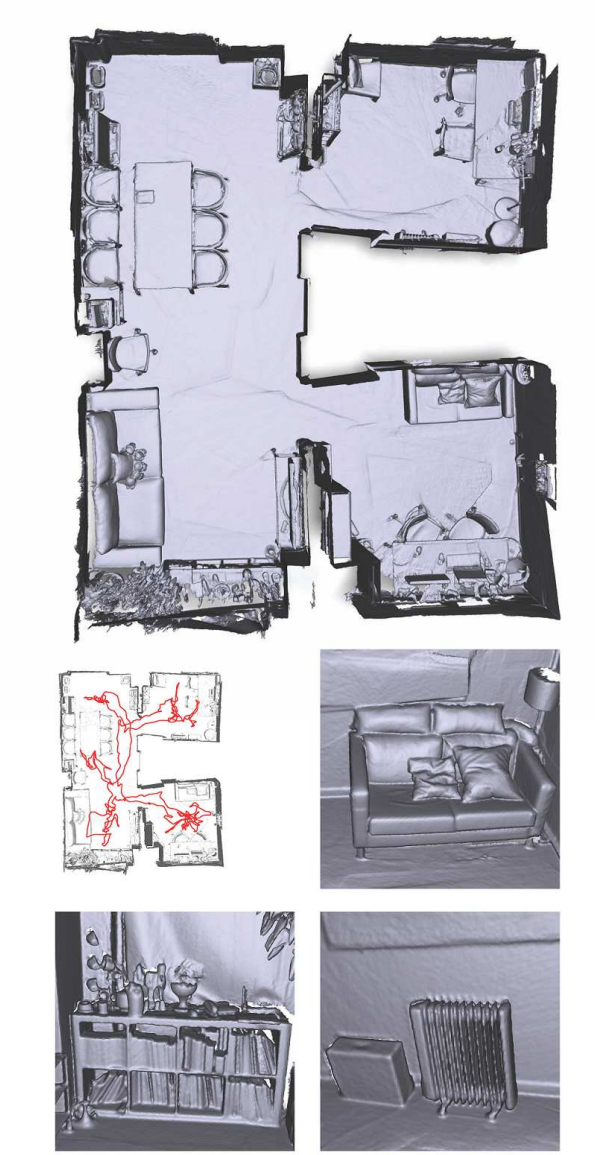

Figure 1. Automatic online reconstruction from a sequence of handheld RGB-D images. The top shows the reconstructed 3D model of our two-pass system. The middle shows our optimized camera trajectory and the rest show some details of the model. The area of this scene is about $40m^2$ and the camera trajectory is about 79 meters long. Scanning the room takes about 360 seconds and the online reconstruction finishes within 170 seconds, i.e. 170 seconds after the user terminates the scanning process.

from odometry drift. Error accumulation can distort the reconstructed surface model.

The rest of the paper presents a system for reconstructing indoor scenes from live RGB-D streams. The philosophy behind our system is to achieve computationally reasonable reconstruction online. Robustness, simplicity and ease of implementation are also crucial factors considered when designing the system. In addition to system contributions, we quantitatively investigate the surface reconstruction quality using geometrical ground truth acquired by a LIDAR system. We prepare evaluation sets for benchmarking dense RGB-D reconstruction approaches in a principled way. This is an important point which enables a fair comparison between different reconstruction algorithms.

2. Relation to Previous Work

This paper owes a lot to a sizable body of literature on dense RGB-D reconstruction, more than we hope to account for here. For the scope of this paper, of particular interest are automatic indoor reconstruction systems designed for quality demanding applications. This excludes methods that use high level semantic cues [15] or human interaction [35].

The most influential related work is KinectFusion [17]. It represents the scene with a signed distance field (SDF) which is defined on a volumetric grid. Each incoming depth image is registered to the SDF and integrated with it using a frame-to-model alignment scheme. Real-time performance is achieved by leveraging GPGPU. Systems built upon the original KinectFusion framework are later proposed to improve its scalability and odometry accuracy [3, 19, 2]. These systems perform online tracking but are lack of effective mechanisms to protect against error propagation and error buildup, which are serious concerns from the system point of view. Therefore in practice they can not well handle furnished scenes that require comprehensive scanning with complex camera trajectories.

The importance of loop closure and map optimization for large scale RGB-D SLAM has been recognized by [12, 34, 13, 7, 32]. These works follow a SLAM architecture and their systems can be broken up into three modules, i.e. frontend, backend, and map generation [7]. Typically the frontend tracks the 6DOF camera pose using either frame-to-frame [12, 13, 7] or frame-to-model approaches [32]. The backend maintains a pose graph which models the geometric relations between keyframes. State-of-the-art RGB-D SLAM systems have complicated system architectures. Frontend (pose tracker and loop detection) and backend (graph optimization and map correction) modules run in several asynchronous threads. Each time a loop closure is encountered, a constraint is added to the graph and an optimization process is triggered. Since the reconstructed map and refined trajectories are from asynchronous threads, a non-rigid map optimization is coupled with incremental pose graph optimization to obtain consistent results. The backend latency is high according to the statistics reported in [32], e.g., about 10 seconds for an indoor sequence that contains two looping points. Recently, [33] uses incremental bundle adjustment with non-rigid deformation for real-time consistent reconstruction. However, its point-based representation might lack general applicability

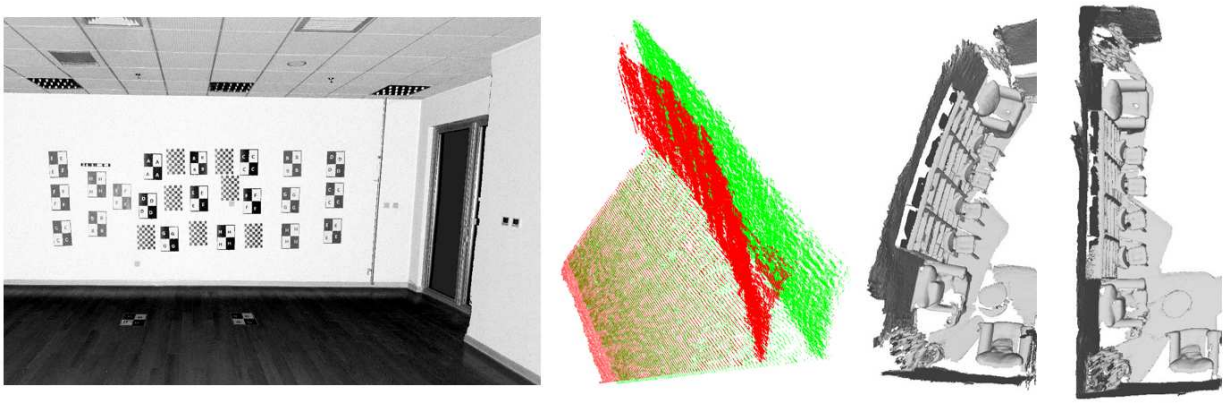

Figure 2. Illustration of the intrinsic RGB-D sensor calibration. From left to right: ground truth scan from LIDAR, plots of 3D measurements of a single shot before (red) and after (green) distortion compensation, partial RGB-D reconstruction from inputs without and with applying distortion compensation.

compared with continuous surface representations.

For fidelity demanding applications, several authors report high quality RGB-D reconstructions by performing intensive offline optimization to mitigate alignment inconsistencies and odometry drift [36, 37, 4]. While their results are visually very plausible, long processing time makes these approaches inapplicable for online applications. We argue that online reconstruction is both favorable and crucial for obtaining accurate 3D models because by seeing a continuously-updated model as the surface is imaged, user can control the coverage by moving the camera appropriately, making the scanning process intuitive and fast.

3. Compensate for Geometric Distortion

Depth measurements from most consumer RGB-D sensors suffer from distortions that increase with range. Discussion about the exact origin of this distortion is not within the scope of this paper. We detail the geometric distortion compensation step because we experimentally found it brings significant accuracy improvements and also suppresses error buildup during the online scene reconstruction. We emphasize that distortion compensation (sensor pre-calibration) should be considered the gold standard for RGB-D reconstruction and SLAM applications. The accuracy gain is well worth the trouble of implementation. This problem, unfortunately, were rarely discussed in previous RGB-D mapping literature. An exception is [38], in which the authors mentioned that the calibration approach of [28] has been applied to their sensor.

A general, unsupervised calibration procedure is described in detail in [28]. In this work, we adopt a supervised method by leveraging a high accuracy LIDAR scanner that we have in house. Multiple markers are attached on a flat wall. The ground truth structure of the wall is obtained from the LIDAR system. The wall is also observed using our RGB-D sensor from different distances and orientations. At each observation spot, we collect 50 to 60 independent measurements and average them to compute a depth map. The depth camera is then registered into the ground truth's coordinate frame from manually labeled 2D-3D correspondences. The infrared image from depth sensor and the laser points' reflection factors provide visual cues for humans to establish correspondences. Depth cameras' extrinsic parameters (pose) w.r.t. the LIDAR metric system are computed via perspective-n-point (PnP) algorithm followed by iterative non-linear optimization. Using the observed depth measurements and LIDAR ground truth, a 3D look-up table (LUT) can be built offline to compensate for depth distortion. In detail, the image plane is spatially divided into 80×80 rectangular tiles. The Z-axis is sampled from various distances and for each discrete range a depth multiplier image is fitted to form a slice of the 3D LUT. At runtime, to avoid discrete jumps we apply linear interpolation along the beam to compute the per-pixel correction factor. The undistorted depth maps are used as our system input. Examples of our calibration results are shown in figure 2. For users who have difficulties obtaining ground truth measurements, unsupervised methods such as [28] is also a good option.

4. Method

Our system adopts a two-pass scheme. The first pass performs parallel camera tracking and pose graph construction, while the second pass handles loop closure and reconstructs the final model. The online system consists of the following three modules.

Camera Tracking: We extends the voxel hashing based volumetric fusion [19] approach to achieve robust real-time camera tracking. The reconstruction results can be visualized instantly to guide the scanning.

Figure 3. Reconstruction of the Reading Room (top row) and the UE Lab (bottom row) datasets. From left to right are results produced by DVO-SLAM [13], indoor reconstruction method [4] with nonrigid refinement, method [4] with rigid optimization, ElasticFusion [33], and our method without/with pose graph optimization.

Pose Graph Construction: We detect loops with visual information and construct a pose graph incrementally on-the-fly.

Model Reconstruction: Upon termination of scanning, we perform pose graph optimization to refine the trajectory and handle loop closures. After that, we make use of the volumetric fusion to fuse these depth images into a final 3D model with refined trajectory.

4.1. Preliminaries

The depth camera outputs a sequence of RGB-D frames $\{I_i, D_i\}$, where I_i and D_i denote the ith color and depth image respectively. To reconstruct the 3D scene, for each incoming frame, we need to estimate the camera transformation matrix

$$T_i = \begin{bmatrix} R_i & t_i \\ 0^\top & 1 \end{bmatrix} \tag{1}$$

where $R_i \in \mathcal{SO}(3)$ is the rotation matrix and $t_i \in \mathcal{R}^3$ is a translation vector. An inhomogeneous world point $\tilde{\mathbf{X}}$ in 3D is transformed into the camera's coordinate frame as $R\tilde{\mathbf{X}} + t$.

We will use the pinhole camera model with pre-calibrated intrinsic parameters. Suppose f_x, f_y are focal lengths and c_x, c_y are the image centers of the camera, a 3D point $\tilde{\mathbf{X}} = (X, Y, Z)$ in the camera coordinate system is projected to the image plane under perspective projection

$$\pi(\tilde{\mathbf{X}}) = (f_x \frac{X}{Z} + c_x, f_y \frac{Y}{Z} + c_y)^\top \tag{2}$$

Conversely, a pixel $(u, v) \in \mathcal{R}^2$ with depth $Z = D(u, v)$ can be backprojected through

$$\phi(u, v, Z) = (\frac{u - c_x}{f_x} Z, \frac{v - c_y}{f_y} Z, Z)^\top \tag{3}$$

In this paper, the 3D world scene is represented as a *TSDF* (truncated signed distance function)

$$\psi : \mathcal{R}^3 \to \mathcal{R} \tag{4}$$

4.2. Camera Tracking with Volumetric Fusion

To perform real-time 6DOF camera tracking, we adopt the framework of volumetric fusion with voxel hashing [19], which can handle large-scale reconstruction with fine-grained details. The reconstructed scene is represented as a *TSDF* on a volumetric grid with voxel hashing data structure. Voxel hashing structure allows real-time access and update using GPGPU. With streaming algorithms, data can be easily streamed in and out the hash table while the camera is moving. The volumetric fusion framework stores the *TSDF* value $\psi(\mathbf{p})$ and an associated weighting factor $W(\mathbf{p})$ in a voxel whose center locates at point $\mathbf{p}$. And it has two important components, *i.e.*, frame-to-model registration and model integration.

4.2.1 Frame-to-Model Registration

For frame-to-model registration, we adopt a weighted *TSDF* tracking method, which is similar to [2] and performs better than the original projective ICP method [17]. It registers each incoming depth image to the reconstructed *TSDF* model instead of a rendered depth image generated from ray casting. For each pixel (u, v), suppose the corresponding inhomogeneous 3D point is $\tilde{\mathbf{X}}_{u,v}$, to compute the camera transformation $\{R, t\}$, we need to minimize an objective function of the form

$$\mathbf{E}_{R,t} = \sum_{u,v} w_{u,v} \psi(R\tilde{\mathbf{X}}_{u,v} + t)^2 \tag{5}$$

For the rest of the paper we will omit the subscript u, v for concision. To solve equation (5), we use Gaussian-Newton nonlinear minimization method. At iteration $k+1$ ($k \geq 0$), we start to linearize T^{k+1} around T^k by a linear function

$$T^{k+1} \approx \begin{bmatrix} 1 & -\gamma & \beta & a \\ \gamma & 1 & -\alpha & b \\ -\beta & \alpha & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} T^k \tag{6}$$

A 6-dimensional vector $\xi^{k+1} = (\alpha, \beta, \gamma, a, b, c)$ is a twist coordinate representing the incremental transformation relative to T^k at iteration k. (α, β, γ) and (a, b, c) are the angular velocity and the translation vector, respectively.

By denoting $\mathbf{Y} = T^k\mathbf{X}$, ξ^{k+1} is computed from equation

$$\sum w \nabla\psi(\tilde{\mathbf{Y}}) \nabla\psi(\tilde{\mathbf{Y}})^\top \xi^{k+1} = -\sum w\psi(\tilde{\mathbf{Y}}) \nabla\psi(\tilde{\mathbf{Y}}) \tag{7}$$

where $\tilde{\mathbf{Y}}$ represents $\mathbf{Y}$'s inhomogeneous version (a 3-vector), $\nabla\psi(\tilde{\mathbf{Y}})$ is the derivative of the *TSDF* function evaluated at point $\tilde{\mathbf{Y}}$. ξ^{k+1} can be computed efficiently by solving this 6×6 linear equation. The incremental transformation can then be obtained from ξ^{i+1}. We update T^{k+1} and iterate this process until convergence. We terminate when either the change of ξ between iterations is small enough or a certain number of iteration is reached. The max number of allowed iteration is set to 20 in our implementation. Similar to [2], we parallelize the per-pixel computation using GPU to obtain real-time performance.

4.2.2 Model Integration

For model integration, we apply a non-uniform weighting strategy to integrate the *ith* incoming depth image with the previously reconstructed model (ψ_{i-1}, W_{i-1}). Suppose $\mathbf{p}$ is the center of a voxel that is close to the underlying surface of the incoming depth image. Through projection we can compute its distance to the *ith* camera, denoted by $C_i(\mathbf{p})$, and the depth value on the projected pixel $D_i(\pi(\mathbf{p}))$. The distance of $\mathbf{p}$ to the underlying surface is $f_i(\mathbf{p}) = C_i(\mathbf{p}) - D_i(\pi(\mathbf{p}))$. Then the voxel with center $\mathbf{p}$ is updated by:

$$\psi_i(\mathbf{p}) = \frac{W_{i-1}(\mathbf{p})\psi_{i-1}(\mathbf{p}) + \lambda_i(\mathbf{p}) f_i(\mathbf{p})}{W_{i-1}(\mathbf{p}) + \lambda_i(\mathbf{p})} \tag{8}$$

$$W_i(\mathbf{p}) = W_{i-1}(\mathbf{p}) + \lambda_i(\mathbf{p}) \tag{9}$$

The parameter $\lambda_i(\mathbf{p})$ makes the model integration favor points from the near range, which usually has better accuracy than points from a far distance. In our experiments, we use a linear weighting function $\lambda_i(\mathbf{p}) = 1 - (D_i(\pi(\mathbf{p})) - d_{min})/(d_{max} - d_{min})$ with $d_{min} = 0.5m, d_{max} = 4.0m$.

4.3. Online Pose Graph Construction

To alleviate error accumulation and prevent the system from drifting, we construct a pose graph incrementally in a backend thread parallel to the frontend tracking thread. We denote the pose graph by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. When the ith frame arrives, we create a new node $\mathcal{V}_i$ and add an edge connecting $\mathcal{V}_i$ with its predecessor (previous frame) $\mathcal{V}_{i-1}$. Similar to fragment based methods [36, 4], we make a valid assumption that the frame-to-model tracker returns locally accurate pose estimates. Therefore the relative pose between nodes $\mathcal{V}_i$ and $\mathcal{V}_{i-1}$ is simply computed from their absolute poses returned by the tracker. Note that unlike [13], we do not add additional local edges to $\mathcal{G}$ thus simplifying our graph structure.

In addition to edges that link successive frames, there are loop edges which are added to the graph through loop detection. To reduce data redundance and system delay, we only establish loop edges between keyframes. For keyframe selection, the incoming frame is identified as a new keyframe if its pose is sufficiently different from the last keyframe in the pose graph's keyframe queue. In our implementation we set the threshold for this relative pose change to be (5°, 0.02m). To enable real-time performance, loops have to be detected efficiently. We amend the visual place recognition technique DBoW [9] by replacing original SURF feature with fast ORB feature [25]. Once a new incoming frame is being recognized as a keyframe, we treat it as a query image and DBoW returns the best matched keyframe from the keyframe queue.

To estimate the relative pose between two keyframes, we first extract 2D feature correspondences using a direct index strategy, *i.e*. only feature pairs that are within the same visual word are considered as matching candidates. Then, we use Random Sample Consensus (RANSAC) and Nister's three-point pose algorithm [22] to calculate their relative transformation δT. In our implementation, if RANSAC inlier ratio is less than 25% or inlier number is lower than 15, we consider it as a false detection. Otherwise, we refine δT using all the inliers via Levenberg-Marquardt (LM) optimization.

By now, we obtain an initial estimate of the relative pose from 2D-3D correspondences. We further validate δT's correctness and continue refining it before adding it to the pose graph. We experimentally found that cautious validation of edge constraints is crucial for pose graph construction because false loop edges and incorrect pairwise transformations can distort the final trajectory. First, in practice the distribution of matched visual features can be non-uniform, leading to biased pose estimation. To address this problem, we use δT as an initial guess and employ point-to-plane ICP to refine it (we experimentally found point-to-plane ICP outperforms its point-to-point counterpart for robustness). Due to the narrow field of view, aligning two indi-

vidual depth frames can be unreliable. To increase robustness, we instead align two 3D point clouds formed using the keyframes and their K-nearest neighbors in the RGB-D sequence (in our system we set K=10). To speedup ICP, depth maps are downsampled by half and we project 3D points to depth image to reduce the ICP search space form 3D to 2D. Second, since geometric-based ICP is sensitive to planar structures, we perform PCA analysis on one of the two 3D point clouds and discard the loop edge if points are mostly sampled from a planar object.

4.4. Model Reconstruction

Before the second pass model reconstruction, we need to optimize the pose graph to achieve globally consistent frame poses. We incrementally optimize the pose graph but not coupled with the graph construction procedure. We control the optimization frequency to balance the delay between threads and accuracy. Denoting $c_{i,j}$ as a 6-vector constraint derived from either tracking measurement or loop detection, and $f(T_i, T_j)$ as the relative transformation (also 6-vector) from pose T_i and T_j, we optimize the pose graph through minimizing the following energy function

$$r_{i,j} = f(T_i, T_j) - c_{i,j}$$
$$E_{\mathcal{G}}(\{T\}) = \sum_{e_{i,j} \in \mathcal{E}} \rho(r_{i,j}^T \mathbf{\Omega}_{i,j}^{-1} r_{i,j}) \quad (10)$$

, where $\mathbf{\Omega}_{i,j}$ is the covariance matrix of edge $e_{i,j}$, and $\rho(.)$ is the Cauchy robust function. To approximate and unify the covariance from different measurements, we adopt the Monte Carlo estimation [11]. We use the ceres-solver [1] to solve this minimization problem. Finally, we apply the volumetric fusion described in section 4.2 to reconstruct the final model. Camera tracking is replaced by the optimized trajectory and only depth image integration is performed.

5. Experiments

5.1. Trajectory Evaluation

To evaluate the trajectory accuracy of our method, we test our system on the RGB-D benchmark presented in [27]. This benchmark provides synchronized ground truth camera poses for the RGB-D sensor, recorded by a precise motion capture system. As shown in table 1, we compare our system with four other state-of-the-art RGB-D based SLAM systems: RGB-D SLAM [6], MRS-MAP [26], DVO SLAM [13] and Kintinuous SLAM [32]. We also provide our camera tracking results with volumetric fusion only (without pose graph optimization). We use the RMSE of the ATE as evaluation metric in our comparison. From the table, we can see that our system achieves consistent performance except the fr1/room dataset which has motion blur caused by high angular velocity. Our camera tracking via volumetric fusion shows a comparable performance on trajectory evaluation.

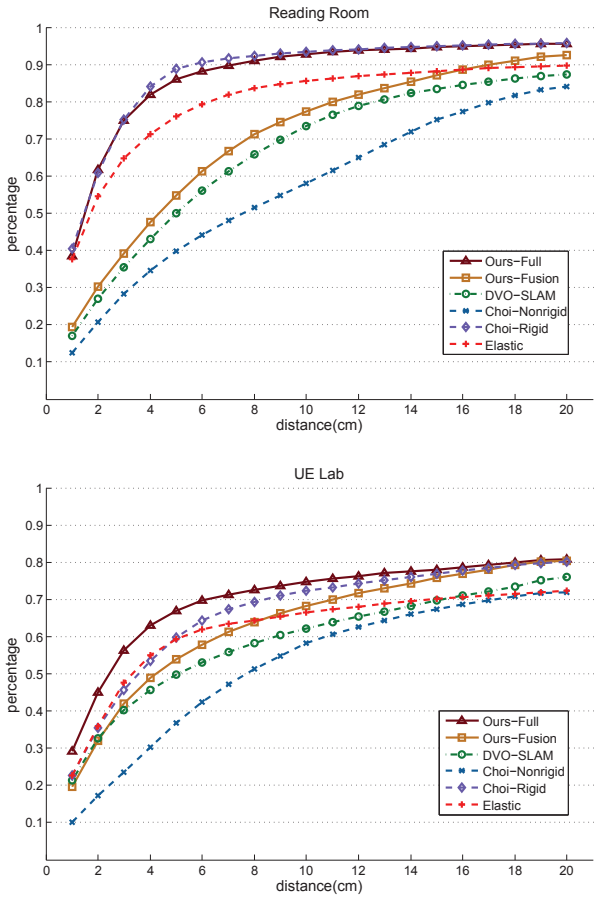

Figure 5. Cumulative histogram of errors from ground truth surface to the reconstructed surface on Reading Room and UE Lab datasets.

Table 1. Comparison of the ATE RMSE on the RGB-D SLAM benchmark datasets[27].

Dataset	RGB-D SLAM	DVO SLAM	MRS Map	Kintinuous SLAM	Ours	
					Tracking	Optimize
fr1/desk	0.023	0.021	0.043	0.037	0.023	0.024
fr1/desk2	0.043	0.046	0.049	0.071	0.050	0.044
fr1/room	0.084	0.053	0.069	0.075	0.238	0.093
fr1/xyz	0.014	0.011	0.013	0.017	0.013	0.013
fr1/rpy	0.026	0.020	0.027	0.028	0.037	0.029
fr1/plant	0.091	0.028	0.026	0.047	0.055	0.050
fr2/desk	0.057	0.017	0.052	0.034	0.100	0.044
fr2/xyz	0.008	0.018	0.020	0.029	0.027	0.017
fr3/office	0.032	0.018	0.042	0.030	0.056	0.036

5.2. Model Quality Evaluation

For RGB-D dense reconstruction, based on our knowledge, there is no ground truth on real data for evaluating reconstruction accuracy. We thus make use of a high pre-

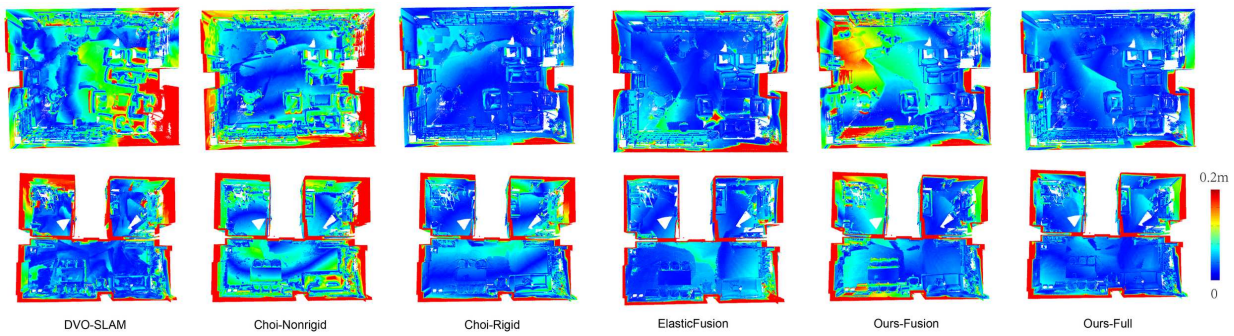

Figure 4. Heat maps showing errors from ground truth surface to the reconstructed surface on Reading Room and UE Lab datasets with different methods. From blue to red, the error increases from zero to 0.2m.

Table 2. Ground truth information of the two datasets.

Dataset	Vertex	Frame	Area(m^2)
Reading Room	7468347	10465	40
UE Lab	9691886	10414	36

cision LIDAR system (Riegl VZ 400) to sense the environments and build ground truth 3D point clouds for two indoor scenes (named Reading Room and UE Lab dataset, respectively). Each dataset contains a ground truth point clouds, a RGB-D video stream, and corresponding calibration information (see 2 and 6).

With the ground truth data, we compare our system with state-of-the-art RGB-D SLAM and reconstruction methods: the DVO-SLAM [13], robust indoor reconstruction system [4] and ElasticFusion [33]. For DVO-SLAM, we use its optimized trajectory as inputs for volumetric fusion to obtain the reconstructed model. For [4], we compare with its non-rigid refinement and rigid refinement versions. For ElasticFusion, we compare with the reconstructed point-based representation. We also evaluate the performance of our volumetric fusion without pose graph optimization.

We use two kinds of error metrics. One is the mean and median distance of the reconstructed surface to the ground truth surface as adopted in [10]. The other is cumulative histogram of the distance from the ground truth surface to the reconstructed surface. Both metrics evaluate the reconstruction accuracy and the second one can also reflect the reconstruction completeness.

From the heat map 4, the cumulative histogram in Figure 5, the reconstruction results in Figure 3 and Table 3, it can be seen that our system performs better than DVO-SLAM and nonrigid indoor reconstruction system, and is comparable to rigid indoor reconstruction system and Elastic Fusion on both datasets. Compared with DVO-SLAM, our frame-to-model registration is more suitable for surface reconstruction. DVO-SLAM detects much more loop edges and might fail to estimate the relative camera pose correctly for such complicated handheld RGB-D sequences. For the nonrigid indoor reconstruction system, it can produce better reconstruction locally, however, the reconstructed model bears some contraction due to non-rigid deformation. From Figure 3, we can see the size of result produced by nonrigid indoor reconstruction is smaller than other approaches. And from the heat map in Figure 4, we can also see that its errors are mainly introduced by the walls. For ElasticFusion, we can see that it produces better reconstruction accuracy while our system obtains superior completeness. This is because ElasticFusion removes unreliable (low confidence) points to guarantee high reconstruction accuracy at the cost of sacrificing model completeness.

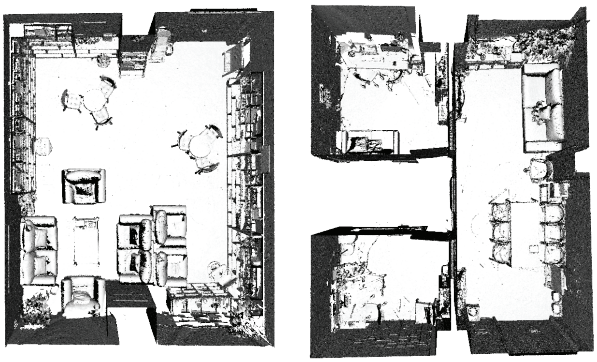

Figure 6. Ground truth of Reading Room and UE Lab.

Table 3. Comparison of reconstructed surface to ground truth distance error (cm), measured by mean and median statistics.

Dataset	DVO	Choi-Nonrigid	Choi-Rigid	Elastic	Ours-Fusion	Ours-Full
Reading	(8.54,5.89)	(7.63, 6.11)	(2.77,1.91)	(1.90,1.35)	(7.16,4.62)	(2.76,1.93)
UE Lab	(5.99,4.26)	(6.31,4.96)	(4.25,2.99)	(2.50,1.99)	(5.25,3.42)	(3.29,2.24)

5.3. Speed Evaluation

We perform all experiments on a Laptop PC with an Intel i7-4710 HQ CPU with @2.50GHz, 16GB of RAM and a nVidia Geforce GTX 980M GPU with 4GB of memory. We evaluate the performance of our system on the Reading Room and UE Lab datasets. Our system is composed of tracking and fusion threads (*Tr*), pose graph construction

Table 4. Computational performance of our system and comparison with other methods. Quantities of processing time of each component and graph structures are illustrated. Delay here measures the latency between end time of online tracking and starting time of reconstruction.

Dataset	Method	Time(s)				Graph	
		Tr	Tr+Opt	Delay	Recon	Nodes	Edges
Reading	Ours	361.4	362.9	1.5	169.5	10465	83
	DVO	666.9	1295.8	628.9	-	688	2867
	Choi	about 8 hours in total				-	-
UE Lab	Ours	359.0	360.4	1.4	157.2	10385	105
	DVO	647.8	1079.1	431.4	-	599	2464
	Choi	about 8 hours in total				-	-

and optimization threads (*Opt*), and the second-pass reconstruction module (*Recon*). We illustrate the processing time of these components as well as the number of nodes and edges of our pose graphs in Table 4. The tracking thread can process at real-time and the parallel optimization thread keeps up closely (the 1.5s delay comes from the pose graph optimization). The second-pass reconstruction and its visualized results can be presented to user at high frame-rate (about 70 FPS).

We also compare our performance with the offline method [4] and online method DVO [13]. Our system can achieve comparable reconstruction accuracy but with a much lower time cost compared with [4]. The DVO system has longer delay of its graph optimization thread. This is caused by the dense graph structure constructed in DVO's backend thread, which is much denser than our pose graph structure. Thanks to the frame-to-model tracking strategy, we are able to maintain a sparse graph structure and achieve better efficiency.

5.4. Limitation

Like most real-time camera tracking methods, the online tracking module in our system tends to fail when there are fast motions, especially fast rotation. But we argue that live visual feedbacks can alleviate this problem. By observing the quality of online reconstruction, the scan operator can adjust his/her motion accordingly. Our reconstruction system makes use of visual features for loop detection and pose graph construction. It will have troubles for loop closure in large texture-less scenes and scenes with repeated textures/structures.

6. Conclusion

We present a system for online dense reconstruction of indoor scenes using inputs from a handheld RGB-D camera. Encouraged by the low latency, accuracy and robustness of our results, we describe all the key components and their implementation details, including offline depth sensor calibration, real-time frame-to-model tracking, online pose graph construction and surface model reconstruction. Unlike existing dense RGB-D SLAM systems, we concentrate on the fidelity of the reconstructed models. The output meshes are evaluated quantitatively by comparing with LIDAR ground truth using meaningful metrics. Thorough experiments demonstrate that our system produces coherent and accurate results comparable with state-of-the-art offline methods and meanwhile it is efficient, simple, and easy to implement. We also hope that the LIDAR datasets, of which we created for evaluating our system, can contribute on benchmarking indoor RGB-D reconstruction algorithms, encouraging further research. The full dataset is available for downloading at: http://research.baidu.com/institute-of-deep-learning/rgbd-recon/.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[2] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *RSS 2013*, volume 9, 2013.

[3] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(4):113, 2013.

[4] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR 2015*, pages 5556–5565.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *PAMI, IEEE Transactions on*, 2007.

[6] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.

[7] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

[8] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, pages 1450–1457, 2011.

[9] D. Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[10] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *ICRA 2014*.

[11] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[13] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *IROS*, 2013.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007*, pages 225–234. IEEE.

[15] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012.

[16] 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*, pages 1498–1505. IEEE, 2010.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[18] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision, 2011 IEEE International Conference on*.

[19] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169, 2013.

[20] D. Nister. Automatic passive recovery of 3d from images and video. In *3D Data Processing, Visualization and Transmission (3DPVT)*, pages 438–445, 2004.

[21] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[22] D. Nistér and H. Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[23] M. Pollefeys, D. Nister, J.-M. 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. *IJCV*, 78(2-3):143–167, 2008.

[24] 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 ISMAR*, pages 83–88, 2013.

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision, 2011 IEEE International Conference on*, pages 2564–2571.

[26] F. Steinbrucker, C. Kerl, D. Cremers, and J. Sturm. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *ICCV*, 2013.

[27] 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*, pages 573–580. IEEE, 2012.

[28] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via slam. In *RSS*, 2013.

[29] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *Computer Vision Workshops, 2009 IEEE International Conference on*. IEEE.

[30] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1450–1457. IEEE, 2012.

[31] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *ICRA*, 2013.

[32] 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. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[33] 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, July 2015.

[34] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[35] Y. Zhang, W. Xu, Y. Tong, and K. Zhou. Online structure analysis for real-time indoor scene reconstruction. *ACM Trans. on Graph*, 2014.

[36] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Trans. on Graph*, 2013.

[37] Q.-Y. Zhou and V. Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *CVPR*, 2014.

[38] Q.-Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *ICCV*, 2013.