Real-Time Tracking and 3D Dense Reconstruction Based on ORB-SLAM3 Extensions using a Depth Camera

JIWEI HOU[1], MONA GOEBEL[1] & DOROTA IWASZCZUK[1]

Abstract: Simultaneous localization and mapping (SLAM) is widely used for robot perception of the environment. It supports the robot in determining its own position as well as the position of surrounding objects. Due to the low cost and the intuitive approach to capture the environment, there are many visual SLAM (vSLAM) systems that have been published in the past decades. In this work, we will evaluate the SLAM algorithm ORB-SLAM3. ORB-SLAM3 is one of the best performing open source vSLAM algorithms, which can not only automatically estimate the exact position and pose of the camera, but also construct a 3D sparse point cloud map of the working area. However, the sparse 3D maps produced cannot meet the requirements of robots performing tasks such as obstacle avoidance, planning, and navigation autonomously. To improve the usability of the sparse maps extracted from ORB-SLAM3, this work investigates ways to increase the density of point clouds. For this, we reconstructed dense point cloud maps of 3D scenes using an extended ORB-SLAM3 mapping algorithm based on RGB-D images and camera poses. We tested our dense mapping system with the benchmark TUM RGB-D dataset published by the Technical University of Munich. Thereafter, we collected data with RealSense depth camera D455 and got a good real-time dense mapping result.

1 Introduction

The problem of simultaneous localization and mapping (SLAM) in robotics has attracted many researchers working on problems in this field over the last few decades. Researchers have proposed a large number of SLAM systems, incorporating sensors, optimization algorithms, and map descriptions. All SLAM systems are designed to maintain system robustness, improve tracking accuracy, and achieve real-time performance. The availability of 3D maps is an important requirement for robots in different workspace conditions to autonomously perform multiple tasks including positioning, planning and navigation. Especially in complex and dynamic environments, it is critical for robots to quickly generate and maintain 3D maps through on-board sensors. For example, self-driving vehicles require high precision real-time maps to avoid obstacles and navigate safely in the complex real world.
Today's common SLAM systems include vSLAM and LiDAR SLAM. vSLAM uses cameras as its primary sensor, tracking the pose of the sensor while creating a map of the environment (FUENTES-PACHECO et al. 2015). LiDAR-based SLAM system uses laser sensors to generate a 3D map of its environment. Due to the low cost and the intuitive approach to create a 3D map, there have been many vSLAM systems published in the past decades. Representative examples include PTAM (KLEIN et al. 2007), LSD-SLAM (ENGEL et al. 2013), SVO (FORSTER et al. 2014), RGB-D

[1] Technical University of Darmstadt, Department of Civil and Environmental Engineering Sciences, Remote Sensing and Image analysis, Franziska-Braun-Str. 7, D-64287 Darmstadt.
E-Mail: [jiwei.hou, mona.goebel, dorota.iwaszczuk]@tu-darmstadt.de

DOI: 10.24407/KXP:1841071439

SLAM (ENDRES et al. 2014), ORB-SLAM (MUR-ARTAL et al. 2015) and ORB-SLAM3 (CAMPOS et al. 2021).

Among the many vSLAM solutions, ORB-SLAM is one of the traditional feature point-based SLAM algorithm. With the release of ORB-SLAM, ORB-SLAM2 (MUR-ARTAL et al. 2017) and ORB-SLAM3, ORB-based SLAM systems have been continuously updated and improved in the past few years. Especially ORB-SLAM3 has become one of the best performing feature-based SLAM system that operates in real time, both indoors and outdoors. ORB-SLAM algorithms are lightweight and can therefore be run on CPU hosts. However, ORB-SLAM aims at long-term and globally consistent localization rather than building the most detailed dense reconstruction. The previous work of ORB-SLAM2 reconstructed dense point clouds, but ORB-SLAM3 did not. Furthermore, the dense mapping code in ORB-SLAM2 was not published open-source. We apply ORB-SLAM3 in 3D indoor mapping and modelling, where a depth camera is used to scan the room. Therefore, it is very important to obtain a 3D dense point clouds reconstruction of indoor space. In this paper, we will focus on the mapping part of SLAM and investigate ways to increase the density of point clouds. Based on ORB-SLAM3, we extend the sparse map constructed by the original system to a dense point cloud map using RGB-D images and camera poses. Thereafter, we collected data with the RealSense depth camera D455. Our results are then compared with the TUM RGB-D dataset published by the Technical University of Munich (STURM et al. 2012). Finally, we used the OctoMap library (HORNUNG et al. 2013) to reconstruct an efficient probabilistic octree map for robotic applications.

The main contents of this paper are as follows: We discuss background and related work in Section 2, describe our system and method in Section 3. Lastly, in Section 4, we present the RGB-D dense mapping results and evaluation, and draw a conclusion in Section 5.

2 Background and Related work

We will first highlight the progress in the field of vSLAM research. Thereafter, ORBSLAM3 is explained in more detail. We close this section with an overview of publications using Red-Green-Blue-Depth (RGB-D) images for vSLAM and introduce the well-known Point Cloud Library.

2.1 Visual SLAM

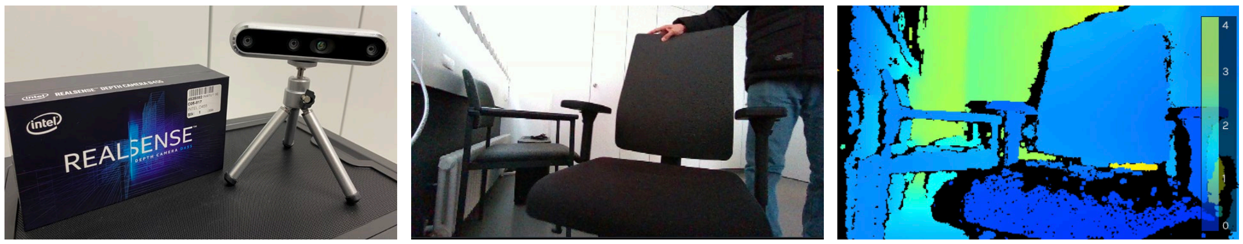
As we mentioned in the introduction, researchers have developed many SLAM solutions, most of which are open source on GitHub. We selected some typical ones as reference, especially focus on 3D dense reconstruction based on RGB-D sensors. GEORG KLEIN and DAVID MURRAY proposed the PTAM algorithm (2007), which splits tracking and mapping into two parallel threads. The system can easily track across multiple scales and provide tracking quality suitable for small workspace augmented reality (AR) applications. DTAM (GANAI et al. 2012) is a system for real-time camera tracking and reconstruction, it does not rely on feature extraction, but on a dense per-pixel approach. The DTAM algorithms are highly parallelizable throughout and rely on GPU to achieve real-time performance. LSD-SLAM (ENGEL et al. 2013) is a feature-less monocular SLAM algorithm which runs in real-time on a CPU. The algorithm changes the pixel selection to make it suitable for larger scale scenarios. However, as it is a feature-less method based on the assumption of grayscale invariance, its robustness and accuracy may be affected by unmodeled behaviors such as lens vignetting and drastic changes in illumination. RGB-D SLAM (ENDRES et al. 2014) system uses Random Sample Consensus (RANSAC) to estimate the transformations between associated key points and optimizes the pose graph using non-linear optimization. This takes advantage of the dense color and depth images provided by RGB-D cameras to estimate camera pose and 3D environment construction. RGB-D SLAM can robustly handle challenging scenarios, such as fast camera movements and feature-poor environments, while being fast enough to operate online. ZHANG et al. (2022) proposed a comprehensive visual SLAM system that extends the application of ORB-SLAM3, which realized 3D dense reconstruction with both RGB-D and stereo cameras. Although they have not made their system open source, we did get a lot of inspiration from their paper.

2.2 ORB-SLAM3

ORB-SLAM3 (CAMPOS et al. 2021) is a visual SLAM algorithm that supports multiple cameras. This algorithm optimizes several aspects such as map initialization, relocation, closed-loop detection, key frame selection, map construction, sensor support, and ultimately provides excellent performance in terms of operation speed, tracking effect and composition accuracy.
The algorithm of ORB-SLAM3 includes three main threads: tracking thread, local mapping thread, loop detection and map fusion. The tracking thread performs rough processing of the input data, including feature point extraction, frame matching, key frame filtering, and converts frames and map points into nodes and edges to provide reliable initial values for subsequent threads. The local mapping thread further filters and optimizes the key frame data, and uses G2O (KÜMMERLE et al. 2011) to optimize the pose relationship between frames and map points. Loop detection and map fusion improve the drift error and multi-map management.

2.3 RGB-D cameras

RGB-D cameras are active image sensors that can not only collect color information, but also calculate depth using these stereo images. Some have additional LiDAR sensors included, to measure distance in space. In this paper, we use Intel® RealSense™ Depth Camera D455, as shown in Fig.1a. This camera can acquire RGB three channels color images and the corresponding depth data, as in Fig.1b and 1c. It uses active Infrared (IR) stereo vision technology to measure depth information with a left and right image sensor and an optional IR projector. The detection distance is between 0.52 m and 6 m. IR projector projects invisible static infrared patterns to improve depth accuracy in low texture scenes. The left and right image sensors capture the scene and send image data to a depth imaging (vision) processor that calculates the depth value of each pixel in the image by associating points between the left image and the right image. Moreover, the RGB sensor includes a global shutter and is matched to the depth Field of View (FOV), improving not only the quality of RGB images but also the correspondence between depth and RGB images. Another benefit is that the D455 camera supports self-calibration without the need for specialized targets. Intel® RealSense™ on-chip calibration allows accurate inspection of the system to ensure the sensor is operating in the optimum range. This makes the D455 even more convenient to calibrate and use.



(a) D455 Depth Camera (b) RGB image (c) Depth image

Fig. 1: The Intel® RealSense™ Depth Camera D455 (a), with an example RGB image (b) and its corresponding depth image (c)

2.4 Point Cloud Library（PCL）

PCL (RUSU et al. 2011) is a powerful cross-platform open source C++ programming library developed based on previous research on point clouds. It started as an open-source project under the Robot Operating System (ROS), which is developed and maintained by Dr. RADU and others from the Technical University of Munich (TUM) as well as researchers from the Stanford University. ROS is mainly for robotics research applications. Furthermore, PCL implements a large number of general algorithms and efficient data structures related to point clouds such as acquisition, filtering, segmentation, alignment, retrieval, feature extraction, identification, tracking, surface reconstruction and visualization. It supports multiple operating system platforms and can run on Windows, Linux, Android, Mac OS X, and some embedded real-time systems.

3 Methods

In this paper, we reconstruct point cloud maps based on ORB-SLAM3 system. The pipeline of this work is shown in Fig. 2. ORB-SLAM3 algorithm performs high precision estimation of camera pose and sparse maps simultaneously. In the program, we create a single thread to extend this work, through invoking every single keyframe from ORB-SLAM3 directly, and combining it with the acquired optimized camera poses and RGB-D information. Using RGB images and depth images, we can acquire point clouds per session. Finally, we merge point clouds based on camera poses and compose the map. To improve the quality and accuracy of the point cloud reconstruction, we use a two-layer filtering method: (1) statistical filtering in the local map to remove the outlier points, and (2) voxel filtering in the global map to downsample which reduces the number of points in the point cloud. Thereby, the shape characteristics are preserved, and less memory is used without serious distortions. Furthermore, an octree map was constructed which can be applied to robot obstacle avoidance, navigation and interactive manipulation.

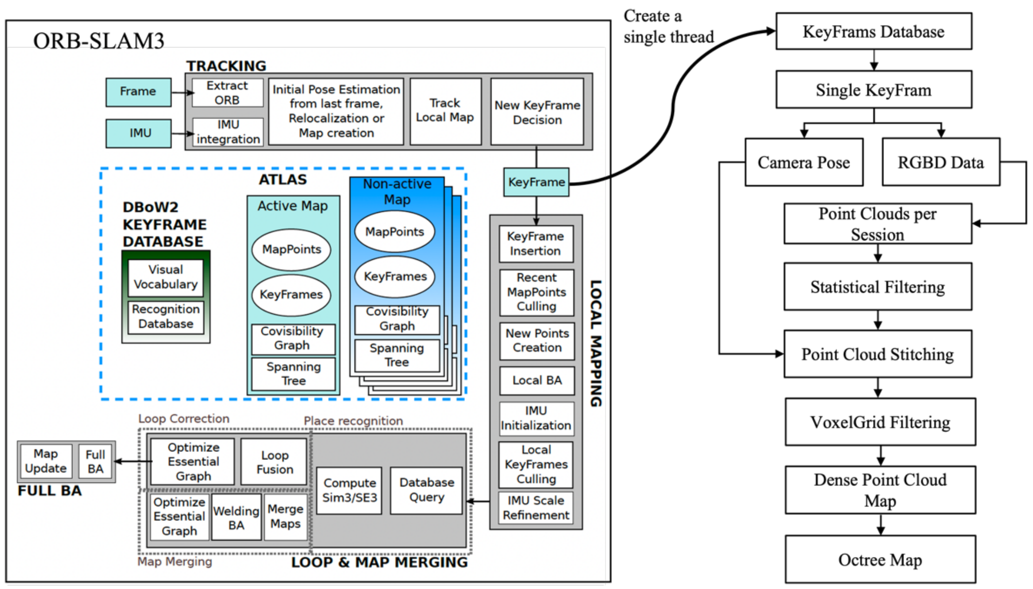


Fig. 2: Pipeline of dense point cloud map reconstruction based on ORB-SLAM3 (Campos et al. 2021)

3.1 Statistical filtering

One of the more novel methods of point cloud denoising in PCL is the StatisticalOutlierRemoval filter, which calculates the distribution of distances from each point to its neighbor in the input data and obtains the average distance from each point to all its neighbor. The result is assumed to be a Gaussian distribution whose shape is determined by the mean μ and standard deviation σ. Assume the coordinate of point P_n (X_n, Y_n, Z_n), the distance from this point to any point P_m (X_m, Y_m, Z_m) can be expressed as equation (3-1),

$$S_i = \sqrt{(X_n - X_m)^2 + (Y_n - Y_m)^2 + (Z_n - Z_m)^2} \quad . \tag{3-1}$$

The mean distance between each point to any points is calculated as in (3-2)

$$\mu = \frac{1}{n}\sum_{i=0}^{n} S_i \quad . \tag{3-2}$$

The standard deviation is given by formula (3-3)

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - \mu)^2} \quad . \tag{3-3}$$

Set the standard deviation multiplier as *std*, retain the point P_n when the mean distance between point P_n and adjacent k points is within the standard range (μ-$\sigma\times std$, $\mu+\sigma\times std$), and if it is not within that range, it is defined as an outlier and removed from the point clouds dataset.

3.2 VoxelGrid filtering

The voxel filter allows for downsampling without destroying the geometry of the point cloud itself, but it does shift the position of the points. In addition, the voxel filter can remove a certain amount of noise and outliers. The main function is to downsample.
The principle of VoxelGrid filtering is to first calculate a cube that can easily enclose the input point cloud, and then split the large cube into smaller cubes according to a set resolution. For each of the points within the small cube, their centroid is calculated, and the coordinates of the centroid are used to approximate a number of points within the cube, thus reducing the amount of data. It is therefore often used for downsampling large amounts of data, especially as pre-processing step before, for example, the alignment or surface reconstruction. This can be a good way to increase the efficiency of the program. The calculation steps of VoxelGrid filtering are as follows.
The first step is to determine the edge length L of each unit voxel, the formula is (3-4)

$$L = \alpha \sqrt[3]{\frac{s}{n}} \tag{3-4}$$

$$n = \frac{N}{V} \tag{3-5}$$

where *s* is the scale factor, *α* is the scale factor used to adjust the edge lengths of unit voxels, and *n* is the number of points per voxel, which can be calculated by (3-5). *N* represents the total number of points , *V* represents the volume of each unit voxel. By adjusting *α*, the algorithm can dynamically adapt to the sparsity of each part of the point cloud. Using (3-4) and (3-5) we can get the relationship between the unit voxel edge length and the total number of points as

$$L = \alpha \sqrt[3]{\frac{s \times V}{N}} \quad . \tag{3-6}$$

The volume of the unit voxel is ,

$$V = L_x \times L_y \times L_z \tag{3-7}$$

where L_x, L_y, and L_z represent the projections of unit voxels on the x, y, and z axes, respectively. We can then replace V in equation (3-6), resulting in

$$L = \alpha \sqrt[3]{\frac{s \times L_x \times L_y \times L_z}{N}} \quad . \tag{3-8}$$

The second step is to calculate the centroid coordinates per voxel, thus, finding the point cloud data which can represent the unit voxel.

$$\begin{cases} X_c = \sum_{i=1}^{m} \frac{x_i}{m} \\ Y_c = \sum_{i=1}^{m} \frac{y_i}{m} \\ Z_c = \sum_{i=1}^{m} \frac{z_i}{m} \end{cases} \tag{3-9}$$

In (3-9), where m is the number of points within the voxel. (x_i, y_i, z_i) are the coordinates of each point. The point cloud which is closest to the centroid coordinates is selected and retained in place of all points within the voxel to achieve VoxelGrid filtering downsampling.

4 Results and Discussion

To validate the performance of our proposed dense reconstruction system, we tested it with the benchmark RGB-D camera dataset from TUM as well as with the self-collected data from our office in real time scanning with a handheld D455 camera. All experiments were carried out on a computer with Ubuntu18.04 as operating system, AMD Epyc 7402p, 24-core processor, 256G RAM and NVIDIA RTX A4000 GPU.

4.1 Dense mapping results and evaluation

In this paper, we have chosen fr2_desk and fr3_long_office_houshold as input data. Fig. 4a and Fig. 4b are the experimental results displayed in the software CloudCompare. The well-defined and straight contours of the desks and chairs show the high accuracy alignment of our RGB-D dense mapping system. After estimating the camera trajectory of fr2_desk and fr3_long_office_houshold, the estimated result is saved in a local file. We then evaluate the error of the estimated trajectory by comparing it with ground truth data provided by the official website of TUM RGB-D dataset. There are different error metrics. The absolute trajectory error (ATE) is an ideal error metric for measuring the performance of vSLAM systems.

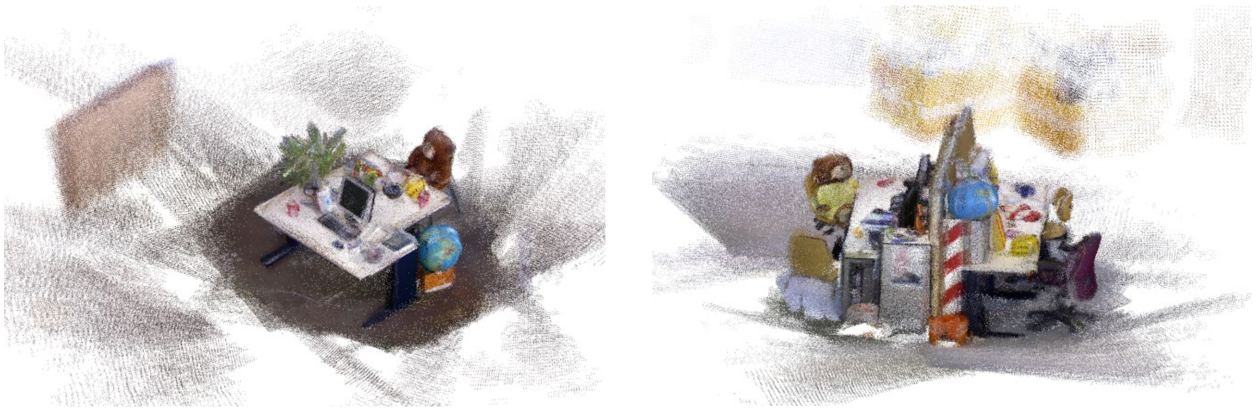
To evaluate the scanning performance of the system, we ran each data sequence three times and extracted the file size of the resulting point cloud and the number of points it contained. Tab. 1 shows the statistics of the point cloud data and the corresponding camera motion trajectory ATE. We can see that for the same TUM RGB-D sequence, the point number and file size in each point cloud data vary with each run. One reason for this could be the different number of keyframes chosen by the algorithm during point cloud reconstruction. Another reason could be the two-layer filtering process, effecting the final result.

After this first analysis, we used the ATE evaluation script provided by the official website of TUM RGB-D dataset, to compute the ATE of the estimated camera trajectories for each sequence with respect to the ground truth. Fig. 5 shows the ATE visualized for the selected sequences, and Tab. 2 shows the ATE root-mean-square error (RMSE), ATE mean, ATE median, ATE standard deviation (Std), ATE minimum and maximum in meters for the *fr2_desk* and *fr3_long_office_houshold* sequences. From these

Tab. 1: Statistics of point cloud maps for the selection of TUM RGB-D sequences. MB stands for megabytes

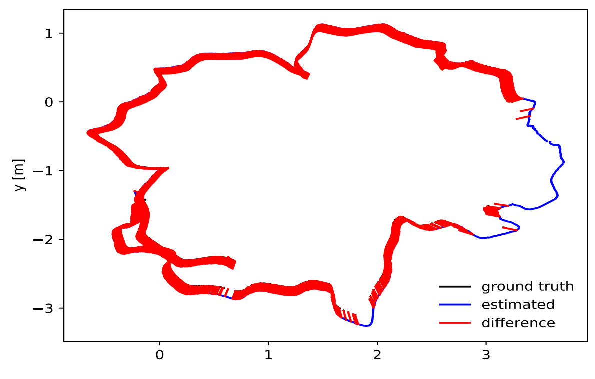
Datasets	***fr2_desk***			***fr3_long_office_houshold***		
Number of points	1,320,810	1,322,930	1,369,868	1,618,853	1,592,130	1,602,135
Mean number of points	1,337,869			1,604,373		
Number of keyframes	229	236	238	284	274	273
Mean number of keyframes	234			277		
File size (MB)	21.1	21.2	21.9	25.9	25.5	25.6
Mean file size (MB)	21.40			26.67		

values, we show that the state of our RGB-D dense mapping system extended by ORB-SLAM3 is relatively stable without major drift.
Finally, this RGB-D point cloud dense reconstruction system is also successfully used for the real time mapping of the data collected with the handheld RealSense D455 camera. The computer reads the depth and color images from the D455 via USB 3.2 interface and publishes the image data as a ROS topic. We scanned the desk in the office and the room scene around the desk separately. The results are shown in Fig. 6a and Fig. 6b.

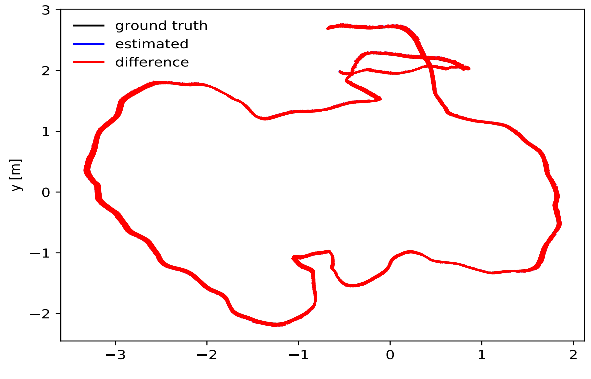


(a) *fr2_desk* (b) *fr3_Long_office_household*

Fig. 4: The dense mapping result for the TUM RGB-D dataset



(a) *fr2 desk*

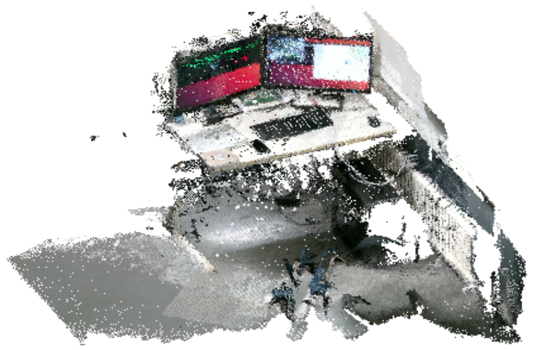


(b) *fr3 Long office household*

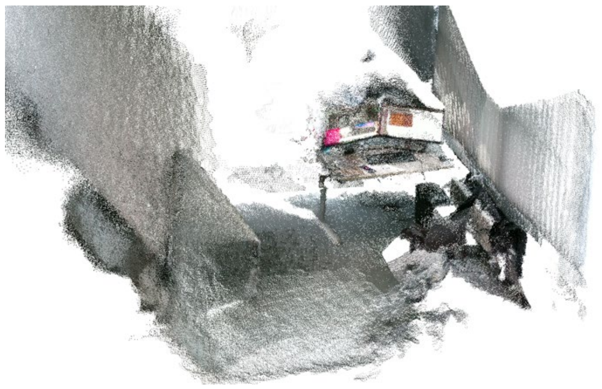
Fig. 5: The ATE results for fr2_desk and fr3_Long_office_household after comparing the estimated camera trajectory with ground truth

Tab. 2: Statistics of absolute translational error in meters for the selection of TUM RGB-D sequences.

Data	***fr2_desk***	***fr3_long_office_houshold***
RMSE	0.092832	0.031973
Mean	0.091270	0.030279
Median	0.093389	0.030554
Std	0.016960	0.010267
Min	0.049749	0.006144
Max	0.138055	0.055867



(a) Desk

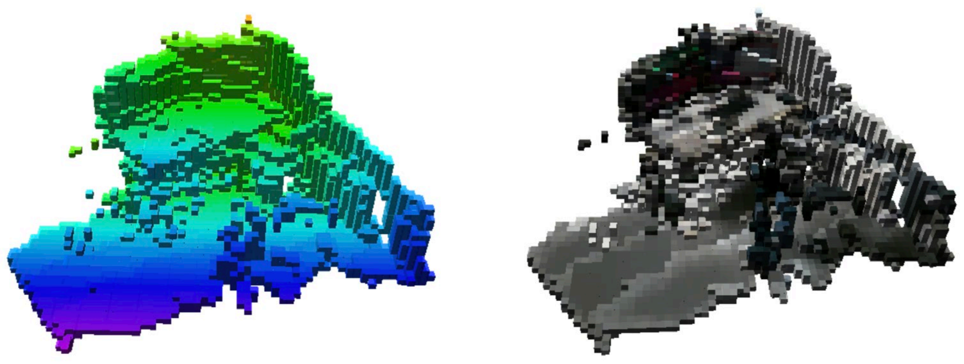


(b) Room scene around the desk

Fig. 6: The RGB-D dense mapping result for our dataset

4.2 The Preliminary results of octrees

We used OctoMap library to convert the point cloud data into octrees data, the point cloud data is shown in Fig. 6a, the octrees results are shown in Fig. 7a and Fig. 7b, Fig. 7b is octrees map result with RGB information. The file size of the colored point cloud data was reduced from 7.3 megabytes to 5 kilobytes and 77 kilobytes (1 megabyte = 1024 kilobytes), the data volume was reduced by approx. 99.93% and 98.96%.



(a) Octrees map view in height (b) Octrees map with RGB information

Fig. 7: Octomap reconstructed from desk in our office

5 Conclusion and future work

In this paper, we built a real-time 3D dense point cloud reconstruction system based on ORB-SLAM3, using the statistical and voxel filtering algorithms provided by PCL. Further, we successfully applied it on the self-collected Intel RealSense D455 camera data. This system was also tested on the TUM RGB-D. Shown through experiments indoors, the system can successfully generate indoor dense point clouds. Finally, an initial result of octrees mapping is displayed, which could be used in our future research for robotic automatic navigation.

As this is only a preliminary experimental result, we will further improve the robustness and performance of this system, such as adding filters and adding map loop detection to enhance the results. In addition, the octree map could be built in real time based on the point clouds, rather than the current offline generation. Moreover, we have only carried out real-time construction of the 3D point cloud in a small indoor room and have not yet tested it in a larger space, which is one of the next steps.

6 ACKNOWLEDGMENT

This research is supported by China Scholarship Council (CSC), Grant/Award Number: 202108130064.

7 References

CAMPOS, C., ELVIRA, R., RODRIGUEZ, J. J. G., MONTIEL, J. M. M. & TARDOS, J. D., 2021: ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. IEEE Transactions on Robotics, **37**(6), 1874-1890, https://doi.org/10.1109/TRO.2021.3075644.

ENDRES, F., HESS, J., STURM, J., CREMERS, D. & BURGARD, W., 2014: 3-D Mapping with an RGB-D camera. IEEE Transactions on Robotics, **30**(1), 177-187, https://doi.org/10.1109/TRO.2013.2279412.

ENGEL, J., STURM, J. & CREMERS, D., 2013: LSD-SLAM: Large-Scale Direct Monocular SLAM. Proceedings of the IEEE International Conference on Computer Vision, 1449-1456.

FORSTER, C., PIZZOLI, M. & SCARAMUZZA, D., 2014: SVO : Fast Semi-Direct Monocular Visual Odometry. 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 15-22. https://doi.org/10.1109/ICRA.2014.6906584.

FUENTES-PACHECO, J., RUIZ-ASCENCIO, J. & RENDÓN-MANCHA, J. M., 2015: Visual simultaneous localization and mapping: a survey. Artificial Intelligence Review, **43**(1), 55-81, https://doi.org/10.1007/s10462-012-9365-8.

GANAI, M., LEE, D. & GUPTA, A., 2012: DTAM: Dense Tracking and Mapping in Real-Time Richard. 1, https://doi.org/10.1145/2393596.2393650.

HORNUNG, A., WURM, K. M., BENNEWITZ, M., STACHNISS, C. & BURGARD, W., 2013: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, **34**(3), 189-206. https://doi.org/10.1007/s10514-012-9321-0.

KLEIN, G. & MURRAY, D., 2007: Parallel tracking and mapping for small AR workspaces. 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR, 225-234, https://doi.org/10.1109/ISMAR.2007.4538852.

KÜMMERLE, R., GRISETTI, G., STRASDAT, H., KONOLIGE, K. & BURGARD, W., 2011: G2o: A general framework for graph optimization. Proceedings - IEEE International Conference on Robotics and Automation, 3607-3613, https://doi.org/10.1109/ICRA.2011.5979949.

MUR-ARTAL, R., MONTIEL, J. M. M. & TARDOS, J. D., 2015: ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics, **31**(5), 1147-1163, https://doi.org/10.1109/TRO.2015.2463671.

MUR-ARTAL, R. & TARDOS, J. D., 2017: ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics, **33**(5), 1255-1262, https://doi.org/10.1109/TRO.2017.2705103.

RUSU, R. B. & COUSINS, S., 2011: 3D is here: Point Cloud Library (PCL). Proceedings - IEEE International Conference on Robotics and Automation, 1-4. https://doi.org/10.1109/ICRA.2011.5980567.

STURM, J., ENGELHARD, N., ENDRES, F., BURGARD, W. & CREMERS, D. 2012: A benchmark for the evaluation of RGB-D SLAM systems. IEEE International Conference on Intelligent Robots and Systems, 573-580, https://doi.org/10.1109/IROS.2012.6385773.

ZHANG, H., XU, C. & GU, J., 2022: Dense Reconstruction from Visual SLAM with Probabilistic Multi-Sequence Merging. Canadian Conference on Electrical and Computer Engineering, 2022-September, 33-40, https://doi.org/10.1109/CCECE49351.2022.9918256.