IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 31, 2020, accepted January 15, 2021, date of publication January 21, 2021, date of current version February 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053188

Survey and Evaluation of RGB-D SLAM

SHISHUN ZHANG, LONGYU ZHENG, AND WENBING TAO, (Member, IEEE)
National Key Laboratory of Science and Technology on Multi-Spectral Information Processing, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Wenbing Tao (wenbingtao@hust.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61772213, Grant 91748204, and Grant 61991412.

ABSTRACT The traditional visual SLAM systems take the monocular or stereo camera as input sensor, with complex map initialization and map point triangulation steps needed for 3D map reconstruction, which are easy to fail, computationally complex and can cause noisy measurements. The emergence of RGB-D camera which provides RGB image together with depth information breaks this situation. While a number of RGB-D SLAM systems have been proposed in recent years, the current classification research on RGB-D SLAM is very lacking, and their advantages and shortcomings remain unclear regarding different applications and perturbations, such as illumination transformation, noise and rolling shutter effect of sensors. In this paper, we mainly introduced the basic concept and structure of the RGB-D SLAM system, and then introduced the differences between the various RGB-D SLAM systems in the three aspects of tracking, mapping, and loop detection, and we make a classification study on different RGB-D SLAM algorithms according to the three aspect. Furthermore, we discuss some advanced topics and open problems of RGB-D SLAM, hoping that it will help for future exploring. In the end, we conducted a large number of evaluation experiments on multiple RGB-D SLAM systems, and analyzed their advantages and disadvantages, as well as performance differences in different application scenarios, and provided references for researchers and developers.

INDEX TERMS Computer vision, evaluation, RGB-D SLAM, robotics, survey.

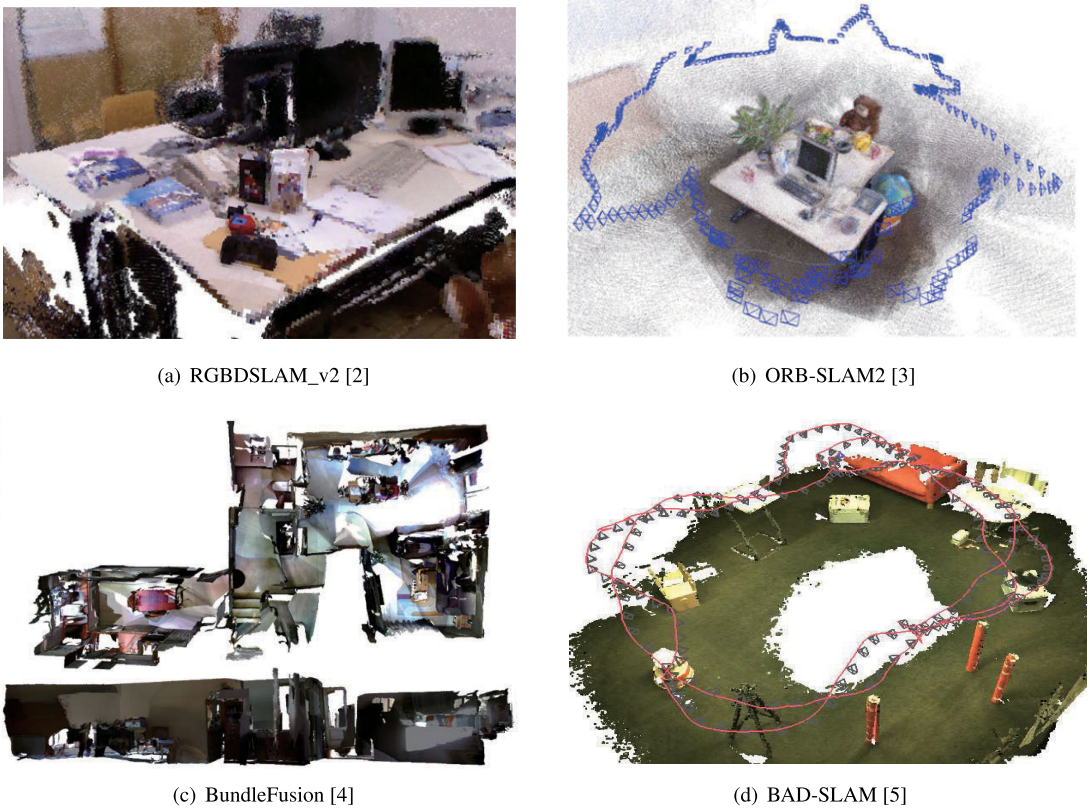
I. INTRODUCTION

SLAM (Simultaneous Localization and Mapping) is a technique developed for solving the problem of self-localization and mapping in an unknown environment. Since it was proposed in the 1980s for the first time, it has made great progress and has been widely used in robot navigation, autonomous driving, augmented reality and virtual reality. For the merits of size, cost and power consumption, SLAM systems using image data captured by cameras as input is becoming more and more popular, which is also called visual SLAM(vSLAM).

Most vSLAM methods have been traditionally based on low-level feature matching and multiple view geometry. This introduces several limitations to monocular vSLAM. For example, a large-baseline motion is needed to generate sufficient parallax for reliable depth estimation; and the scale is unobservable. This can be partially alleviated by including additional sensors (e.g., stereo cameras, inertial measurement units (IMUs), sonar) or the prior knowledge of the system or the scene. Another challenge is the dense reconstruction of low texture areas. Although recent approaches using deep learning have shown impressive results in this direction, more research is needed regarding their cost and dependence on the training data [1].

RGB-D cameras, which can provide both colored image and depth image at the same time, are becoming more and more used for indoor scene reconstruction and can be used to solve the challenges mentioned above. The camera intrinsic parameters calibrated beforehand provide the scale factor for reconstruction and camera tracking. And RGB-D camera can provide depth information for all areas in the field of view with or without textures, making dense reconstruction easy to be done and removing the need for map initialization. It is no wonder that research of mapping and localization using an RGB-D camera has flourished in the last decade. Figure 1 shows the reconstruction results of several state-of-the-art RGB-D SLAM systems. Nowadays, RGB-D cameras have become the most popular sensors for indoor applications in robotics and AR/VR. In the future, it will be promising to use a single RGB-D camera or with other sensors to complete the SLAM task with better-designed algorithms.

The associate editor coordinating the review of this manuscript and approving it for publication was Christopher H. T. Lee.



(a) RGBDSLAM_v2 [2] (b) ORB-SLAM2 [3]

(c) BundleFusion [4] (d) BAD-SLAM [5]

FIGURE 1. Reconstructions of state-of-the-art RGB-D SLAM systems.

In this paper, we review real-time RGB-D SLAM algorithms, which remarkably evolve forward from 2011. We conclude the common designation of most RGB-D SLAM systems and give the basic architecture of an RGB-D SLAM. And we introduce each part of the RGB-D SLAM with relevant works and make some evaluation to help readers understand the advantages and disadvantages of them and finally know how to design an RGB-D SLAM. The remainder of the paper is organized as follows: Section II and III give an overview of the most common RGB-D SLAM pipeline and the notation and preliminaries of the following formulations. Section IV, V, VI introduces camera tracking, local mapping and loop closing algorithms with specific examples separately. Section VII discusses relevant advanced topics and open problems that were not covered in the previous sections. Section VIII gives some evaluation of tracking accuracy, mean tracking time, reconstruction accuracy, etc. of 7 different SLAM systems under 3 different datasets. Section IX lists the open source code and datasets we used. Section X gives a brief conclusion of the paper.

II. BASIC ARCHITECTURE OF RGB-D SLAM

After several decades' development, the pipeline of RGB-D SLAM or vSLAM(more generally) are basically fixed. Modern vSLAM systems are mostly designed using the idea of PTAM [6], which divides the task of SLAM into camera tracking and local mapping, completed by separate threads.

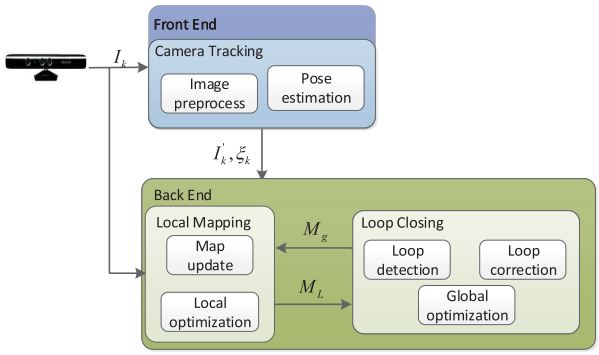


FIGURE 2. Basic architecture of RGB-D SLAM. I_k : k_{th} RGB-D image, I'_k : k_{th} preprocessed RGB-D image, ξ_k : k_{th} camera pose, M_L: local map, M_G: global map.

After PTAM has been proposed, some works [7], [8] added other threads tackling loop closing, global BA, etc. As a result, most state of the art vSLAM systems are built on top of multi-threads and can be divided into two parts: front end and back end. The front end is responsible for providing real-time camera poses while the back end is responsible for slowly map update and optimization.

The basic architecture of RGB-D SLAM is described as Fig 2 and our article will expand based on this architecture. In the front end, RGB-D image I_k and global map M_G are used for image preprocess and pose estimation. In the back

end, Local Mapping thread utilizes camera pose ξ_k, preprocessed RGB-D image I'_k and global map M_G to perform map update and local optimization; Loop Closing thread makes use of the local map M_L to accomplish loop detection, loop correction and global optimization. What's more, the front end of the SLAM without back end optimization can be called RGB-D Odometry, which we will not differentiate with RGB-D SLAM in the following of the work.

III. NOTATION AND PRELIMINARIES

We denote the RGB-D image as $I : \Omega \mapsto \mathbb{R}^4$, and $\Omega \subset \mathbb{R}^2$ is the image plane with width w and height h. We denote the pixel coordinates as a 2D vector $\mathbf{p} = (u, v)^T$ and the corresponding homogeneous coordinates as a 3D vector $\hat{\mathbf{p}} = (\lambda u, \lambda v, \lambda)$. The two coordinates can be transformed by dehomogenization operator π_{2D} and homogenization operator π_{2D}^{-1}, where $\mathbf{p} = \pi_{2D}(\hat{\mathbf{p}}) = (\hat{u}/\lambda, \hat{v}/\lambda)$ and $\hat{\mathbf{p}} = \pi_{2D}^{-1}(\mathbf{p}) = (u, v, 1)^T$. The color value and depth value of a pixel is denoted as $I(u, v) = (r, g, b)^T$ and $D(u, v) = d$ respectively. The Euclidean coordinate and corresponding homogeneous coordinate of a 3D point k in frame i can be denoted as $\mathbf{P}_k^i = (X_k^i, Y_k^i, Z_k^i)^T$ and $\hat{\mathbf{P}}_k^i = (\lambda X_k^i, \lambda Y_k^i, \lambda Z_k^i, \lambda)^T$. These two coordinates can be transformed by dehomogenization operator π_{3D} and homogenization operator π_{3D}^{-1}, where $\mathbf{P}_k^i = \pi_{3D}(\hat{\mathbf{P}}_k^i) = (\hat{\mathbf{X}}_k^i/\lambda, \hat{\mathbf{Y}}_k^i/\lambda, \hat{\mathbf{Z}}_k^i/\lambda)$ and $\hat{\mathbf{P}}_k^i = \pi_{3D}^{-1}(\mathbf{P}_k^i) = (X_k^i, Y_k^i, Z_k^i, 1)^T$.

The transformation matrix $T_{ji} = \begin{bmatrix} R_{ji} & t_{ji} \\ 0 & 1 \end{bmatrix} \in SE(3)$ transforms a 3D point $\mathbf{P}_k^i$ in frame i to $\mathbf{P}_k^{ji}$ in reference frame j, where $R_{ji} \in SO(3)$, $t_{ji} \in \mathbb{R}^3$. The transformation formulation is represented as follow:

$$\mathbf{P}_k^{ji} = \pi_{3D}(T_{ji}\hat{\mathbf{P}}_k^i) = \pi_{3D}(T_{ji}\pi_{3D}^{-1}(\mathbf{P}_k^i)) \tag{1}$$

Using lie algebra, we represent T_{ji} by its minimal representation $T_{ji} = \exp_{\mathrm{SE(3)}}(\xi_{ji})$ with $\xi_{ji} \in \mathbb{R}^6$. And equation 1 can be write as:

$$\mathbf{P}_k^{ji} = \pi_{3D}(\exp_{\mathrm{SE(3)}}(\xi_{ji})\pi_{3D}^{-1}(\mathbf{P}_k^i)) \tag{2}$$

According to the pinhole model of the camera for projection, a 3D point $\mathbf{P}_k^i$ is projected onto 2D point $\mathbf{p}_k^i$ in frame i:

$$\mathbf{p}_k^i = \pi_{2D}(\hat{\mathbf{p}}_k^i) = \pi_{2D}(K\mathbf{P}_k^i) \tag{3}$$

where $K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ is the calibration matrix containing the coordinates of the principal point $(u_0, v_0)^T$ and the focal lengths (f_x, f_y).

Given the 2D point $\mathbf{p}_k^i$ and its corresponding depth value d_k^i, the 3D point $\mathbf{P}_k^i$ can be calculated by back-projection:

$$\mathbf{P}_k^i = K^{-1}\hat{\mathbf{p}}_k^i = K^{-1}d_k^i\pi_{2D}^{-1}(\mathbf{p}_k^i) \tag{4}$$

Given the 2D point $\mathbf{p}_k^i$ in frame i, its reprojection $\mathbf{p}_k^{ji}$ in frame j can be represented as:

$$\mathbf{p}_k^{ji} = \pi_{2D}(K\mathbf{P}_k^{ji}) \tag{5}$$

According to equation 2, equation 5 can be written as:

$$\mathbf{p}_k^{ji} = \pi_{2D}(K(\pi_{3D}(\exp_{\mathrm{SE(3)}}(\xi_{ji})\pi_{3D}^{-1}(\mathbf{P}_k^i))) \tag{6}$$

According to equation 4, equation 6 can be written as:

$$\mathbf{p}_k^{ji} = \pi_{2D}(K(\pi_{3D}(\exp_{\mathrm{SE(3)}}(\xi_{ji})\pi_{3D}^{-1}(K^{-1}d_k^i\pi_{2D}^{-1}(\mathbf{p}_k^i)))) \tag{7}$$

IV. CAMERA TRACKING

In this section, we detailed the camera tracking methods used for estimating 6 Dof motion. Camera tracking usually preprocesses the image before pose estimation, such as extracting features. According to the residual minimized for pose estimation, camera tracking can be divided into three kinds: direct methods, indirect (feature-based) methods and hybrid methods. Direct methods estimate camera motion-based on minimizing photometric error over corresponding pixels in two frames, while indirect methods estimate camera motion based on minimizing geometric error over matching features. And hybrid methods minimize a combination of the aforementioned errors to align two frames. Compared to indirect methods based on features, direct methods use photometric value over chosen pixels, without the need for complex feature extraction and matching, thus being more robust in low texture environments. Recent results show that direct methods present a higher accuracy than those based on geometric alignment [9], [10]. However, direct methods have their limits, like not being robust to moving objects because pixels in small baseline will cause many mismatches, limited accuracy in wide baselines cases because of the small basin of convergence and being sensitive to calibration errors, rolling shutter or unsynchronisation between the color and depth images [1]. Hybrid methods combine these two methods to achieve better accuracy and avoid demerits of both methods.

A. DIRECT METHODS

Based on photometric error, direct methods assume that corresponding points on two images have the same photometric value. The relative pose between two images is calculated by minimizing the pixel intensity difference. The photometric error of the same point $\mathbf{p}_k^i$ between frame i and j can be represented as follows:

$$r_{pho}^k(\xi_{ji}) = I_i(\mathbf{p}_k^i) - I_j(\mathbf{p}_k^{ji}) \tag{8}$$

where $I(\cdot)$ represents the photometric value of a point on the image plane.

The complete cost function $E_{pho}(\xi_{ji})$ is the weighted squared sum of the photometric error of all points and we minimize the cost function to get the camera pose:

$$\xi_{ji}^* = \arg\min_{\xi_{ji}} E_{pho}(\xi_{ji}) = \arg\min_{\xi_{ji}} \sum_k \omega(r_{pho}^k)(r_{pho}^k(\xi_{ji}))^2 \tag{9}$$

with some weight function ω, e.g., robust weight function such as Huber's [11].

1) KinectFusion

Newcombe *et al.* proposed KinectFusion in 2011 [12]. KinectFusion preprocesses the raw input depth image by a bilateral filter to reduce noise. What's more, KinectFusion builds a dense vertex map and normal map pyramid from the preprocessed depth image for coarse to fine pose estimation. Kinectfusion does not need to track frame-to-frame to estimate the camera motion, but uses frame-to-model mechanism to align the current frame with the global model, without the need for explicit loop closing. The vertex map and normal map of each frame are used to build a global model represented by a volumetric, truncated signed distance function (TSDF). When tracking the live depth frame, the global model raycasts the signed distance function into the estimated frame to provide a synthetic frame against which the live depth frame is aligned.With the synthetic frame and its vertex map and normal map provided, KinectFusion establishes a global point-plane energy under the L_2 norm and minimizes it to get the camera pose, as:

$$\mathbf{E}\left(\mathrm{T}_{g,k}\right)=\sum_{\substack{\mathbf{u}\in U\\ \Omega_k(\mathbf{u})\neq\text{null}}}\left\|\left(\mathrm{T}_{g,k}\dot{\mathbf{V}}_k(\mathbf{u})-\hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}})\right)^{\top}\hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})\right\|_2 ! \tag{10}$$

where $\mathrm{T}_{g,k}$ denotes the desired camera pose estimation, $\dot{\mathbf{V}}_k$ is the vertex map of the current frame, $\hat{\mathbf{V}}^g_{k-1}$ and $\hat{\mathbf{N}}^g_{k-1}$ relatively correspond to the prediction vertex map and normal map of the previous frame.

2) RGBDTAM

Alejo Concha and Javier Civera proposed a direct RGBD-SLAM system with state-of-the-art accuracy and robustness at a los cost named RGBDTAM [13] in 2017. RGBDTAM minimizes a weighted sum of photometric error and the error between inverse depth. RGBDTAM only minimizes photometric error of points belonging to Canny edges represented as below:

$$r^k_{pho}(\xi_{ji})=\omega_p\frac{(I_i(\mathbf{p}^i_k)-aI_j(\mathbf{p}^{ji}_k)+b)^2}{\sigma^2_{pho}} \tag{11}$$

where a and b are the global illumination change factors (i.e. the gain and brightness of the current frame with respect to the current keyframe). ω_p is the Geman-McClure robust cost function, used to remove the influence of occlusions and dynamic objects and σ_{pho} is the mean variance of r^k_{pho}. The inverse depth error can be represented as:

$$r^k_d(\xi_{ji})=\omega_p\frac{((e^T_z\mathbf{P}^{ji}_k)^{-1}-D^{-1}(\mathbf{p}^{ji}_k))^2}{\sigma^2_g} \tag{12}$$

where ω_p is still the Geman-McClure robust cost function. $e^T_z=[0,0,1]^T$ is a 3D vector. $D(\cdot)$ represents the corresponding depth value of 2D point $\mathbf{p}_k$ and σ_g is the mean variance of r^k_d.

3) ID-RGBDO

While most direct methods take as many pixels as possible to ensure the accuracy, Alejandro Fontán *et al.* proposed ID-RGBDO in 2020 [14], which takes only 24 points for tracking and can achieve an accuracy similar to the state of the art (some times out perform it). Based on an information matrix, ID-RGBDO selects only the most informative points in the local Bundle Adjustment and pose tracking optimizations. Based on the multivariate Gaussians theory widely used in SLAM context, this work said that to get most informative points is to minimize the differential entropy of the Gaussian distribution which is comformed by the camera pose. As shown in equation 13, the differential entropy H is positively correlated to the determinant of the covariance matrix $\Sigma_{\xi_{ji}}$.

$$H(\xi_{ji})=\frac{1}{2}\log((2\pi e)^k|\Sigma_{\xi_{ji}}|) \tag{13}$$

So the task is transformed to maximize the determinant of the information matrix $\Lambda_{\xi_{ji}}$:

$$\Lambda_{\xi_{ji}}=\Sigma^{-1}_{\xi_{ji}}=\sum_{p\in\mathcal{P}}\Delta_p\Sigma_{\xi_{ji}} \tag{14}$$

With equation 14 the variation of determinant of $\Lambda_{\xi_{ji}}$ can be expressed as follow, which depends on the residual covariance σ^2_r, the p-th row of the Jacobian $\mathbf{J}_{\xi_{ji},\mathbf{p}^i}$ (Jacobian of residual $r^k_{pho}(\xi_{ji})$ with respect to Lie-algebra increments ξ_{ji}) and the current adjoint information matrix Σ^{adj}_x:

$$\Delta_p|\Sigma_x|=\sigma_r^{-2}\mathbf{j}_{x,p}\Sigma^{adj}_x\mathbf{j}^T_{x,p} \tag{15}$$

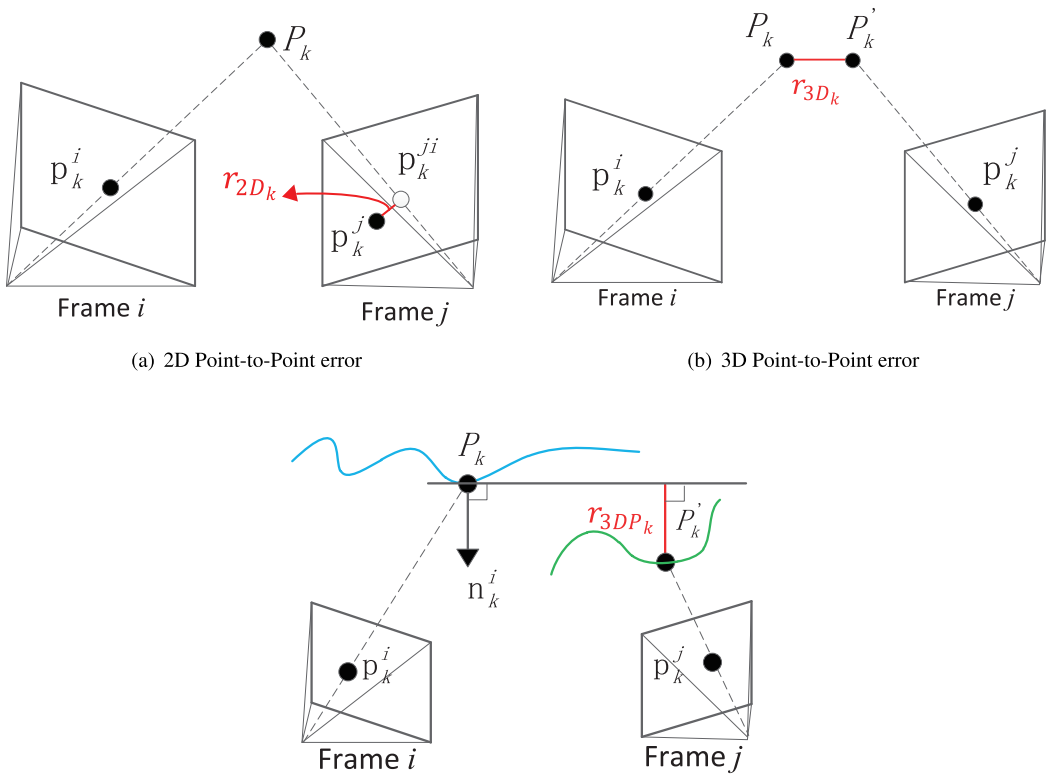
Based on the conclusions above, ID-RGBDO firstly chooses a set of points from high-gradient areas and then follows an algorithm that selects points maximizing the determinant of the information matrix.

B. INDIRECT METHODS

Unlike direct methods which directly estimate the camera motion by minimizing the photometric error of whole or partial pixels, indirect methods, or feature-based methods firstly extract and match features from different images and track the camera using geometric error of the features. The geometric error used to align two images is usually the distance between the two corresponding sets of 2D features or 3D features. There are three kinds of geometric error that is usually used: 2D Point-to-Point error, 3D Point-to-Point error and 3D Point-to-Plane error [1], as shown in Fig 3.

- 2D Point-to-Point error: A geometric reprojection error is the most used 2D error in VO and VSLAM. As shown in Fig 3 (a), given a point feature p^i_k in the reference frame and its matching point p^j_k in the current frame, the distance between p^i_k's reprojection p^{ji}_k and p^j_k is defined as the reprojection error:

$$r^k_{2D}(\xi_{ji})=||\mathbf{p}^j_k-\mathbf{p}^{ji}_k|| \tag{16}$$



(a) 2D Point-to-Point error

(b) 3D Point-to-Point error

(c) 3D Point-to-Plane error

FIGURE 3. Different types of geometric error: (a) Given a 2d point in the reference frame and its matching point in the next frame, a 2d-2d error is the image distance between the point's reprojection and its matching point in the next frame. (b) A 3d-3d error is the Euclidean distance between the two matched points's corresponding 3d points. (c) A 3d-plane error is the Euclidean distance between the tangent plane at the back-projected reference point in frame i and the back-projected reprojection of the point in frame j.

The complete cost function $E_{2D}(\xi_{ji})$ is the weighted squared sum of the 2D Point-to-Point error of all points and we minimize the cost function to get the camera pose:

$$\begin{aligned}\xi_{ji}^* &= \arg\min_{\xi_{ji}} E_{2D}(\xi_{ji}) \\ &= \arg\min_{\xi_{ji}} \sum_k \omega(r_{2D}^k)(r_{2D}^k(\xi_{ji}))^2 \end{aligned} \tag{17}$$

with some weight function ω.

- 3D Point-to-Point error: Instead of using 2D reprojection error to estimate the camera motion, we can also calculate the distance between back-projected points directly. With the RGB-D image and pinhole camera model already known, we can reconstruct the point cloud. For dense point clouds, the 3D error could be the distance between closest points, while for sparse point clouds the 3D error should be the distance between matched 3D points. As shown in Fig 3 (b), given two corresponding 3D points in the frame i and frame j, the error between them can be defined as:

$$r_{3D}^k(\xi_{ji}) = ||\mathbf{P}_k^j - \mathbf{P}_k^{ji}|| \tag{18}$$

The complete cost function $E_{3D}(\xi_{ji})$ is the weighted squared sum of the 3D Point-to-Point error of all points and we minimize the cost function to get the camera pose:

$$\begin{aligned}\xi_{ji}^* &= \arg\min_{\xi_{ji}} E_{3D}(\xi_{ji}) \\ &= \arg\min_{\xi_{ji}} \sum_k \omega(r_{3D}^k)(r_{3D}^k(\xi_{ji}))^2 \end{aligned} \tag{19}$$

with some weight function ω.

- 3D Point-to-Plane error: As shown in Fig 3 (c), the point-to-plane distance, that minimizes the distance along the target point normal, is commonly used in dense RGB-D point cloud alignment [1]. The error can be defined as:

$$r_{3DP}^k(\xi_{ji}) = ||n_k^i(\mathbf{P}_k^j - \mathbf{P}_k^{ji})|| \tag{20}$$

The complete cost function $E_{3DP}(\xi_{ji})$ is the weighted squared sum of the 3D Point-to-Plane error of all points and we minimize the cost function to get the camera pose:

$$\begin{aligned}\xi_{ji}^* &= \arg\min_{\xi_{ji}} E_{3DP}(\xi_{ji}) \\ &= \arg\min_{\xi_{ji}} \sum_k \omega(r_{3DP}^k)(r_{3DP}^k(\xi_{ji}))^2 \end{aligned} \tag{21}$$

with some weight function ω. According to [15], compared to Point-to-Point error, Point-to-Plane error is more robust and its convergence speed is faster in practice.

1) RGB-D SLAM

Felix Endres *et al.* proposed RGB-D SLAM system in 2012 and tested it on a publicly available RGB-D dataset [16]. This method firstly extracts features (SIFT/SURF/ORB [17]) from color image and matches features from different images. With the depth value known in the depth image, one can back project the 2D matching points to 3d and get 3D Point-to-Point matches. RGB-D SLAM then uses Random Sample Consensus (RANSAC) to compute the rigid transformation in SE(3) filtering out outliers. Then RGB-D SLAM uses the inliers to compute a refined transformation, which forms an edge of the pose graph in the backend. To optimize the pose graph, RGB-D SLAM also performs a minimization of a non-liner error function like most mainstream SLAM systems. The error function can be represented as:

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right)^{\top} \Omega_{ij} \mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right) \tag{22}$$

$$\mathbf{x}^* = arg\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) \tag{23}$$

where $\mathbf{x} = \left(\mathbf{x}_1^{\top}, \ldots, \mathbf{x}_n^{\top}\right)^{\top}$ is a vector of pose representations, $\mathbf{z}_{ij}$ and Ω_{ij} respectively correspond to the mean and information matrix of a constraint relating the pose $\mathbf{x}_j$, and $\mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right)$ is the vector error function.

2) ORB-SLAM2

Raúl Mur-Artal *et al.* proposed ORB-SLAM2 in 2017 [3] based on 2D Point-to-Point error. Compared to ORB-SLAM [8], ORB-SLAM2 adds support for stereo cameras and RGB-D cameras. For the speed and rotation invariance, ORB-SLAM2 chooses ORB as the 2D features for all SLAM tasks: tracking, mapping, relocalization and loop closing. If tracking was successful for the previous frame, ORB-SLAM2 uses a constant velocity motion model to predict the camera pose and perform a guided search of the map points observed in the last frame. If not enough matches were found (i.e., motion model is clearly violated), ORB-SLAM2 uses a wider search of the map points around their position in the last frame. The pose is then optimized with the found 2D correspondences using BA(bundle adjustment). With fittest strategy that selects the points and keyframes of the reconstruction and excellent algorithms maintaining a covisibility graph and an Essential Graph.

The 2D Point-to-Point error in ORB-SLAM2 can be represented by the 2D reprojection error of map points to relative matched keypoints as:

$$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \pi_i(\mathbf{T}_{iw}, \mathbf{X}_{w,j}) \tag{24}$$

where $\mathbf{e}_{i,j}$ represent the reprojection error of a map point j in a keyframe i, $\mathbf{x}_{i,j}$ is the matched keypoint, $\mathbf{X}_{w,j} \in \mathbb{R}^3$ is 3D locations of map point j, $\mathbf{T}_{iw} \in SE(3)$ is the pose of keyframe i, and π_i is the projection function.

To get the camera pose, there is a cost function to be minimized, as:

$$C = \sum_{i,j} (\rho_h(\mathbf{e}_{i,j}^T \Omega_{i,j}^{-1} \mathbf{e}_{i,j})) \tag{25}$$

where ρ_h is the Huber robust cost function, and $\Omega_{i,j}$ is the covariance matrix associated with the scale as which the keypoint was detected.

3) PLANE-EDGE-SLAM

Qinxuan Sun *et al.* proposed Plane-Edge-SLAM in 2020 [18] based on the seamless fusion of the plane and edge features. Compared with most SLAM systems based on point features(SIFT, SURF, ORB etc.), this system take the advantage of the robustness and accuracy of the edge and plane features when locating in low-texture regions and structural indoor environments. With the constraint provided by the fusion of these features, the tracking estimation remain well-posed in all circumstances, as illustrated by the article.

Specifically, the tracking of the system is estimated by a plane-edge-fusion-based alignment method, which is an ICP-like scan alignment. The residual errors of the plane and edge primitives are defined in different spaces, so the weights in the total cost function need to be different. The cost function for the plane-edge based motion estimation is designed as below, in which W_p is used to balance the contribution of two kinds of features in the optimization:

$$F(\xi) = \sum_{i=1}^{N_\pi} e_{\pi i}^T \boldsymbol{\Omega}_{\pi i} \boldsymbol{e}_{\pi i} + W_p \sum_{k=1}^{N_p} w_{pk} \boldsymbol{e}_{pk}^T \boldsymbol{\Omega}_{pk} \boldsymbol{e}_{pk} \tag{26}$$

where $\xi = \left[\boldsymbol{t}^T, \boldsymbol{\omega}^T\right]^T$ represents the 6 Dof camera motion, N_π and N_p are numbers of the plane and edge-point pairs respectively, $\boldsymbol{e}_{\pi i}$ and $\boldsymbol{e}_{pk}$ are the residual vectors that measure how well the planes and edge-points aligned, $\boldsymbol{\Omega}_{\pi i}$ and $\boldsymbol{\Omega}_{pk}$ are the information matrix of the two residual vectors relatively. The weights w_{pk} is the essential part of the framework, which is used to provide enough constraints to determine all the components of camera motion.

C. HYBRID METHODS

Hybrid methods combine direct methods and indirect method by minimizing both photometric error and geometric error to get the camera pose. Compared to indirect methods, direct methods can leverage raw photometric information from a designated area in the image, without the need for costly per-frame feature extraction and matching. As a result, direct methods are proved to be more robust in low-texture scenes. However, direct methods have their demerits, such as being less robust to moving objects which is caused by small baselines matching. Hybrid methods or semi-direct methods, inherit the advantages of both methods including the robustness of direct VO and map-reusing capability of indirect

methods [19]. The cost function of hybrid methods can be represented as:

$$E_{hyb} = \omega_{pho} E_{pho} + \omega_{geo} E_{geo} \tag{27}$$

where ω_{pho} and ω_{geo} are the weight factors.

1) CPA-SLAM

Lingni Ma *et al.* proposed CPA-SLAM in 2016 [20]. In this method, except for direct image alignment, a global plane model is built aiding the pose estimation. CPA-SLAM applies an agglomerative hierarchical clustering algorithm to segment planes from the RGB-D image and uses a numerical rule of data association to form a global map made up of different planes. In the estimation of camera pose, CPA-SLAM establishes photometric residual, point-to-plane residual and plane-to-plane residual, which will be jointly minimized by an EM framework. The benefit of this work is that extra frame-to-plane alignment makes use of the dense image information available in keyframes for accurate short-term tracking and the global plane model is used to reduce drift.

2) BundleFusion

Angela Dai *et al.* proposed BundleFusion [4] in 2017. BundleFusion performs a hierarchical local-to-global pose optimization to achieve real-time performance. On the first hierarchy level, every consecutive n frames form a chunk inside which poses of each frame are resolved. On the second hierarchy level, all chunks are correlated with respect to each other and globally optimized [4]. When optimizing the camera pose, BundleFusion uses a sparse-then-dense strategy: A set of sparse features (SIFT [21]) are extracted and matched to obtain a coarse alignment by minimizing the 3D Point-to-Point error; the coarse poses are then refined by dense photometric error and geometric error(3D Point-to-Plane error). These two error are shown below:

$$E_{\text{photo}}(X) = \sum_{(i,j)\in \mathrm{E}} \sum_{k=0}^{|I_i|} \left\| I_i\left(\pi\left(\mathbf{d}_{i,k}\right)\right) - I_j\left(\pi\left(\mathcal{T}_j^{-1}\mathcal{T}_i \mathbf{d}_{i,k}\right)\right)\right\|_2^2 \tag{28}$$

$$E_{\text{geo}}(X) = \sum_{(i,j)\in \mathrm{E}} \sum_{k=0}^{|\mathcal{D}_i|} \left[\mathbf{n}_{i,k}^T \left(\mathbf{d}_{i,k} - \mathcal{T}_i^{-1}\mathcal{T}_j \pi^{-1}\left(\mathcal{D}_j\left(\pi\left(\mathcal{T}_j^{-1}\mathcal{T}_i \mathbf{d}_{i,k}\right)\right)\right)\right)\right]^2 \tag{29}$$

where $\mathcal{T}_i$ and $\mathcal{T}_j$ are the rigid camera transforms of i-th and j-th frame, π is the perspective projection, and $\mathbf{d}_{i,k}$ is the 3D position associated with the k-th pixel of the i-th depth frame, $\mathbf{n}_{i,k}$ is the normal of the k-th pixel of the i-th input frame.

3) KDP-SLAM

Ming Hsiao *et al.* proposed KDP-SLAM [22] in 2017. KDP-SLAM uses planar constraint to reduce drift and key-frame based hierarchical odometry to estimate the camera poses. KDP-SLAM utilizes fast dense odometry method to estimate the pose of every frame relative to that of the most recent reference frame roughly and precise transformations are estimated only for specially selected frames: keyframes, reference frames, and fusion frames [22]. This strategy removes the need for GPU acceleration, while maintaining high accuracy. For both rough and precise pose estimation, KDP-SLAM combines photometric error and geometric error to estimate the camera pose. KDP-SLAM combines a novel iterative projected plane (IPP) method (as the geometric component) and a pyramid dense RGB-D odometry method using Laplacian images (as the photometric component) to estimate the rough odometry. Similarly, KDP-SLAM combines IPP with a semi-dense RGB-D odometry (SRO) method, as the geometric and photometric components, respectively, to estimate the precise odometry [22].

4) SEMI-DIRECT TRACKING AND MAPPING

Ke Liu *et al.* proposed a method called semi-direct tracking and mapping [19] in 2019. This method firstly uses the direct method to estimate the initial pose and the photometric error is defined as the weight SSD over the 8-point neighborhood pixels as proposed in [9]. The optimization is performed using an iteratively reweighted Gauss-Newton algorithm in a coarse-to-fine scheme [19]. After the rigid body transformation is calculated, this method refines camera pose by minimizing the geometric error with respect to the local feature map using motion only BA. When calculating the geometric error, this method uses ORB features with two descriptors: ORB feature descriptor and a novel descriptor described by depth information, the Hamming distance of which is the measure of 2D Point-to-Point error. With the use of depth information in feature matching, this method is able to filter out wrong matches that are only visually similar. What's more, the use of depth information gets this method rid of motion model in tracking, instead of which brute force matching between local map points and feature points in the current frame is used for tracking.

D. ARCHITECTURE DIFFERENCES

In the previous subsections, we classified different RGB-D SLAM systems into direct-method-based, indirect-method-based and hybrid-method-based systems according to their tracking methods, and explained their algorithm structure and optimization approach. Here, we mainly compare and analyze the important differences of the algorithm architectures between direct-method-based and indirect-method-based systems.

Here, we make a comparison and discuss the difference between two typical architecture of a direct-method-based and an indirect-method-based SLAM system. For the direct method, we choose the structure of KinectFusion [12] as a typical representative, because this structure is the basic framework of many subsequent direct method RGB-D SLAM, which is not only simple but universal. For the

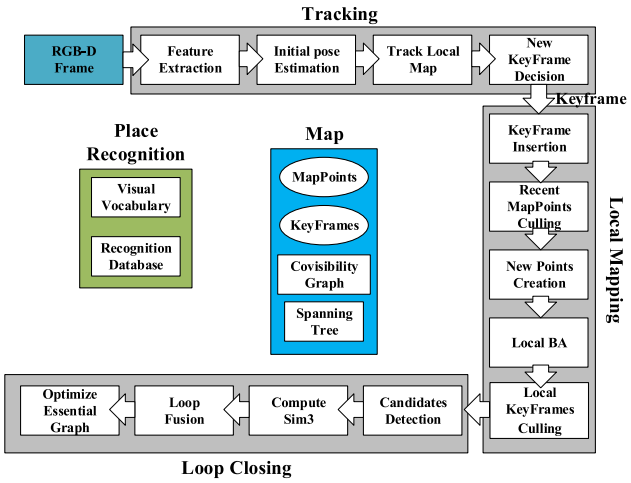


FIGURE 4. Architecture of ORB-SLAM2 [3].

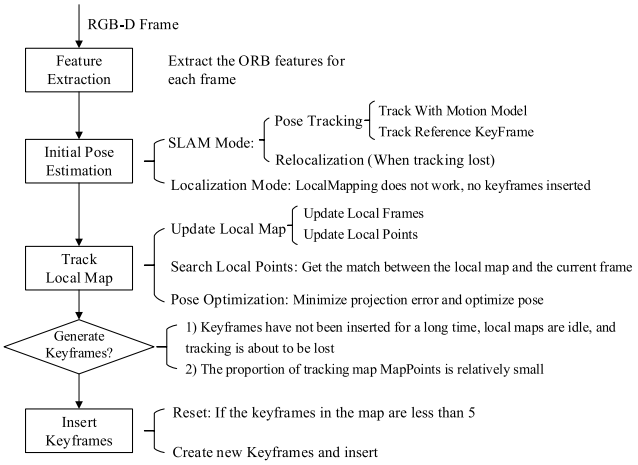


FIGURE 5. The tracking module of ORB-SLAM2 [3].

indirect method, we select ORB-SLAM2 [3] as the comparison object, since it is the most classic and representative indirect-method-based RGB-D SLAM, and it is also the universal basis of many subsequent systems.

Figure 4 and Figure 6 show the overall architectures of ORB-SLAM2 and KinectFusion respectively, and we also drew the basic framework of their respective tracking modules based on their papers, as shown in Figure 5 and Figure 7. From the overall architectures of these two systems we can see that, both of them include the process of tracking(pose estimation) and mapping(reconstruction), although the more traditional method KinectFusion does not have a loop detection process, but this is not common to all direct-based systems.

From the overall architectures and tracking modules of these two representative systems, we can summary the differences between direct-based and indirect-based architectures mainly exist in three aspects:

- Data preprocess: From the Figure 4 and Figure 6, we can see that, for the indirect method, at the beginning of the tracking module, it is necessary to perform feature extraction operations on each frame. For the direct method, it is generally necessary to calculate the surface vertex and normal maps according to each RGB-D frame before the tracking process(which is also the initial process of many direct methods).
- Pose optimization: As shown in Figure 5 and Figure 7, both of these two representative architectures optimize the pose by minimizing a projection error, while both the optimization function and optimization process are different. As mentioned in the previous introduction section, KinectFusion minimizes a global point-to-plane ICP energy to get the camera pose, which is defined in equation 10, while ORB-SLAM2 minimizes a 2D point-to-point reprojection error of map points to relative matched keypoints, which is illustrated in equation 24. We can say that the biggest difference between the direct method and the indirect method lies in the definition of pose optimization, which is directly related to whether the system perform the feature matching process.
- Data association: From the Figure 5 and Figure 7, we can see that, data association and pose estimation are coupled in the direct method, while decoupling in the indirect method. The advantage of decoupling is that the data association can be processed more integrally. The advantage of decoupling is that it can only use image information to solve the data association problem when the pose is uncertain. Therefore, the direct method should be better at solving continuous image positioning, and the indirect method is more suitable for loop detection and relocation. In addition, the sparse direct method is more suitable for occasions with high real-time performance and limited computing resources.

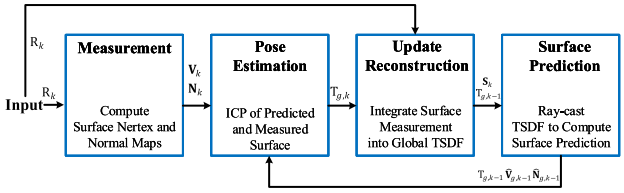


FIGURE 6. Architecture of KinectFusion [12].

V. LOCAL MAPPING

Local Mapping is responsible for building a local map and jointly optimizing camera pose and local map. According to the elements forming the map, we divides local mapping into two kinds: Point-Based methods and Volumetric methods. Point-based methods construct raw point clouds while Volumetric methods construct voxel volumes of the environment.

A. POINT-BASED METHODS

1) DVO-SLAM

Christian Kerl *et al.* proposed DVO-SLAM in 2013 [23]. DVO-SLAM combines dense visual odometry based on a

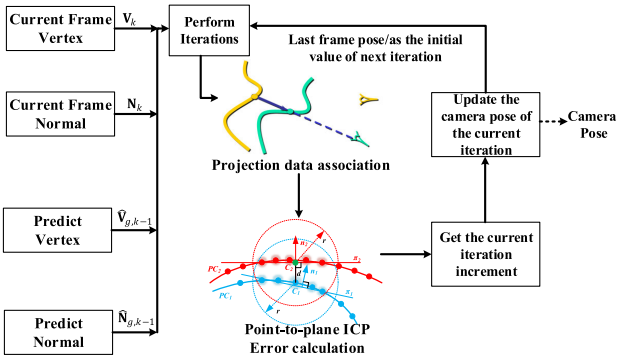


FIGURE 7. The pose estimation(tracking) module of KinectFusion [12].

joint photometric and geometric error minimization and Pose SLAM. In Pose SLAM the map is represented as a graph in which the vertices represent absolute sensor pose of keyframes and the edges represent relative transformations between them. DVO-SLAM introduces a new keyframe selection strategy based on differential entropy: if the entropy ratio of the current frame is below a pre-defined threshold, the previous frame is selected as a new keyframe and inserted into the map. In DVO-SLAM, local map optimization is not performed and global pose graph optimization is performed instead to reduce accumulated drifts.

2) ORB-SLAM2

ORB-SLAM2 maintains a map made up of MapPoints, KeyFrames, Covisibility Graph and Spanning Tree. Covisibility information between keyframes is used in several tasks of ORB-SLAM2 and is represented as an undirected weighted graph. Each node of the Covisibility Graph is a keyframe, and the edge between two keyframes shows that these two frames share observations of the same map points (at least 15). The number of common map points is the weight θ of the edge. ORB-SLAM2 performs a pose graph optimization to correct the loop error and using the whole covisibility graph for that will be very time-consuming. So ORB-SLAM2 uses the subgraph of covisibility graph for loop closing, which is made up of the spanning tree, the subset of edges from the covisibility graph with high covisibility ($\theta_{min} = 100$), and the loop closure edges. ORB-SLAM2 creates new map points by triangulating ORB from connected keyframes κ_c in the covisibility graph. ORB-SLAM2 performs local BA to optimize the currently processed keyframe K_i, the local keyframe set K_L connected to it in the covisibility graph κ_c, and the map point set P_L seen by those keyframes. All other keyframe set K_F that see those map points but are not connected to K_i are included in the optimization but remain fixed. The optimization function can be written as:

$$\{\mathcal{P}_L, \mathcal{K}_L\} = \underset{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l}{\operatorname{argmin}} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{X}_k} \rho\left(E_{kj}\right)$$
$$E_{kj} = \left\| \mathbf{x}^j_{(\cdot)} - \pi_{(\cdot)}\left(\mathbf{R}_k \mathbf{X}^j + \mathbf{t}_k\right) \right\|^2_{\Sigma} \tag{30}$$

where $\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l$ represent the parameters to be optimized: the position of map point, the rotation matrix and translation vector of keyframe pose. As introduced in section IV, $\rho(\cdot)$ is the Huber robust cost function.

3) BAD-SLAM

Thomas Schöps proposed BAD-SLAM in 2019 [5]. BAD-SLAM uses dense *surfels* and *keyframes* to represent the map, reducing the amount of data for optimization(local BA). A *surfel* s is an oriented disc defined by a 3D center point $\mathbf{p}_s$, a surface normal vector $\mathbf{n}_s$, a radius r_s, and a scalar visual descriptor d_s. BAD-SLAM chooses surfels as scene representation for the reason that they can be efficiently fused and updated by BA, and can be quickly deformed to adapt to loop closures. BAD-SLAM establishes the first dense BA approach for RGB-D SLAM that runs in real-time for smaller scenes. The cost function for BA is made up of geometric and photometric terms. The geometric term is defined as point-to-plane error and the photometric term is calculated only in gradient areas.

$$r_{\text{geom}} = \left(\mathbf{T}^k_G \mathbf{n}_s\right)^T \left(\pi^{-1}_{D,k}\left(\hat{\pi}_{D,k}\left(\mathbf{T}^k_G \mathbf{p}_s\right)\right) - \mathbf{T}^k_G \mathbf{p}_s\right) \tag{31}$$

where $\hat{\pi}_{D,k}$ maps the 3D local coordinates of a keyframe k to the corresponding depth map, ($\pi^{-1}_{D,k}$ maps a depth pixel to its 3D point in the keyframe's local coordinates, and $\mathbf{T}^k_G$ is the transformation that maps the center point $\mathbf{p}_s$ and normal vector $\mathbf{n}_s$ to the local coordinates of k.

What's more, BAD-SLAM proves that direct RGB-D SLAM systems are highly sensitive to rolling shutter, RGB and depth sensor synchronization, and calibration errors. BAD-SLAM proposes a novel, well-calibrated benchmark using synchronized global shutter RGB and depth cameras and finds that it is superior to all existing methods in this benchmark, including the best ORB-SLAM2 method so far.

4) ElasticFusion

Thomas Whelan *et al.* proposed ElasticFusion in 2015 [24]. ElasticFusion represents environment as an unordered list of surfels $\mathcal{M}$, where each surfel $\mathcal{M}^s$ has the following attributes: a position $\mathbf{p} \in \mathbb{R}^3$, normal $\mathbf{n} \in \mathbb{R}^3$, color $\mathbf{c} \in \mathbb{N}^3$, weight $\omega \in \mathbb{R}$, radius $r \in \mathbb{R}$, initialisation timestamp t_0 and last updated timestamp t. ElasticFusion follows the method proposed by [25] for performing surfel initialization and depth map fusion but differs in a way that ElasticFusion defines a time window threshold δ_t which divides $\mathcal{M}$ into surfels which are active and inactive and only surfels which are marked as active model surfels are used for depth map fusion. According to the distance from the viewing ray, the angle from the surfel normal and confidence count, ElasticFusion associate new surfel with the global surfel map $\mathcal{M}$. When merging two surfels, ElasticFusion uses a weighted average strategy and introduces an explicit sample confidence applying a Gaussian weight on the current depth measurement, which leads to higher quality denoising.

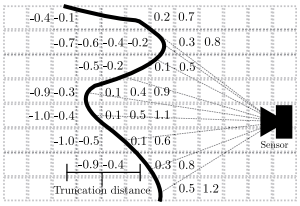


FIGURE 8. Two-dimensional example of an implicit surface represented by TSDF. Shown are example signed distance values stored at voxels within the truncation distance of the observed surface, with rays cast from the observing sensor. Figures taken from [26].

In order to ensure the consistency of the local and global surfaces in the map, ElasticFusion apply a deformation graph to the surface based on the embedded deformation technology. Each surfel $\mathcal{M}^s$ identifies a set of influencing nodes in the graph. The deformed position of a surfel is given by:

$$\hat{\mathcal{M}}_{\mathbf{p}}^s = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n\left(\mathcal{M}^s\right)\left[\mathcal{G}_{\mathbf{R}}^n\left(\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n\right) + \mathcal{G}_{\mathbf{g}}^n + \mathcal{G}_{\mathbf{t}}^n\right] \tag{32}$$

where $\mathcal{G}_{\mathbf{g}}^n$,$\mathcal{G}_{\mathbf{R}}^n$ and $\mathcal{G}_{\mathbf{t}}^n$ relatively correspond to the positon, rotation matrix and translation vector of each node, and the deformation normal of the surface is given by:

$$\hat{\mathcal{M}}_{\mathbf{n}}^s = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} w^n\left(\mathcal{M}^s\right) \mathcal{G}_{\mathbf{R}}^{n-1^\top} \mathcal{M}_{\mathbf{n}}^s \tag{33}$$

Here $w^n\left(\mathcal{M}^s\right)$ is defined by:

$$w^n\left(\mathcal{M}^s\right) = \left(1 - \left\|\mathcal{M}_{\mathbf{p}}^s - \mathcal{G}_{\mathbf{g}}^n\right\|_2 / d_{\max}\right)^2 \tag{34}$$

where $d_{\max}$ is the Euclidean distance to the $k+1$ nearest node of $\mathcal{M}^s$.

B. VOLUMETRIC METHODS

Most volumetric methods adopt volumetric representation of scenes implicitly such as TSDF(truncated signed distance function) [12], as shown in Fig 8. In TSDF, each grid contains the truncated distance to the nearest surface and a weight for the distance measurement.

1) KINTINUOUS

Thomas Whelan *et al.* proposed Kintinuous in 2012 [26]. Kintinuous is developed based on KinectFusion and expanded three capabilities: the capability to reconstruct unfixed scenes, the capability to use photometric information to estimate camera pose and the capability to close the loop. KinectFusion build a map of the scene implicitly by merging RGB-D scans into TSDF. In the case of KinectFusion, the TSDF is stored as a three-dimensional voxel grid in GPU memory and its size is fixed, which means that the scene mapping is limited around the space where TSDF is initialized. To overcome this problem, Kintinuous uses volume shifting, which virtually translates the camera pose to bring the camera's position to within a new boundary and updates the global position of the TSDF.

2) VOXEL HASING

Matthias Nießner *et al.* proposed voxel hashing in 2013 [27]. Volumetric approaches have demonstrated compelling results at real-time, but meanwhile, these methods rely on memory inefficient regular voxel grids which restrict scale. This has led to either moving volume variants [26], [28], which stream voxel data out-of-core as the sensor moves, but still constrain the size of the active volume, or hierarchical data structures that subdivide space more effectively, but do not parallelize efficiently given added computational complexity [29]–[31]. Based on a simple memory and speed efficient spatial hashing technique that compresses space, voxel hashing allows for real-time fusion without not the need for either a memory-constrained voxel grid or a hierarchical data structure of heavy computational overheads. Voxel hashing firstly allocates voxel blocks according to camera view frustum, then calculates TSDF for every voxel. After that, voxel hashing collects garbage, finding out all voxel blocks that have a zero maximum weight or have a minimum TSDF larger than a threshold and removing them. Finally, voxel hashing integrates voxel blocks and extract surfaces by raycasting.

3) BundleFusion

In BundleFusion, scene geometry is represented by an implicit truncated signed distance(TSDF) reconstructed by incrementally fusing all input RGB-D data. TSDF is defined over a volumetric grid of voxels and BundleFusion adopt sparse volumetric voxel hashing approach proposed by Nießner [27] to store and process TSDF data. Compared to original voxel hashing method, BundleFusion allows for RGB-D frames to be both integrated into the TSDF as well as de-integrated for camera pose updates. For integration each voxel is updated by

$$\begin{aligned} \mathbf{D}'(\mathbf{v}) &= \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) + w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v}) + w_i(\mathbf{v})} \\ \mathbf{W}'(\mathbf{v}) &= \mathbf{W}(\mathbf{v}) + w_i(\mathbf{v}) \end{aligned} \tag{35}$$

where $\mathbf{D}(\mathbf{v})$ denotes the signed distance of voxel, $\mathbf{W}(\mathbf{v})$ the voxel weight, $d_i(\mathbf{v})$ the projective distance, $w_i(\mathbf{v})$ the integration weight for a sample of the depth frame D_i.

For de-integration, each voxel is updated by

$$\begin{aligned} \mathbf{D}'(\mathbf{v}) &= \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) - w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v}) - w_i(\mathbf{v})} \\ \mathbf{W}'(\mathbf{v}) &= \mathbf{W}(\mathbf{v}) - w_i(\mathbf{v}) \end{aligned} \tag{36}$$

VI. LOOP CLOSING

Loop Closing is responsible for finding and correcting the loop. For loop detection, the simplest way is to perform a linear search over all existing keyframes, i.e., a new keyframe is matched against all others. This is impossible as the amount of keyframes increases quickly as the camera moves. Therefore, it is important to restrict the search space into the most likely loop candidates. A common method is using the bag of words, which extracts visual feature descriptors from

images and trains an efficient data structure for fast candidate matching. There are also other approaches using metrical or probabilistic nearest neighbour search to find loop candidates [23]. After a loop is found, global optimization is performed.

A. DVO-SLAM

DVO-SLAM searches loop closure candidates in a sphere with a predefined radius around the keyframe position. DVO-SLAM computes for every candidate the relative transformation between the two keyframes and the associated covariance matrix on a coarse resolution. After that, to validate a candidate DVO-SLAM employes a similar entropy ratio test as for keyframe selection. If the candidate passes the test, the relative transformation between the two keyframes and the associated covariance matrix are computed using higher resolutions and the same entropy ratio test is performed again. Keyframes pass the final test is seen as the loop and are inserted into the pose graph. After loop closure is detected, DVO-SLAM uses g2o framework to perform a pose graph optimization, distrubuting the accumulated error over edges in the loop. What's more, at the end of the dataset, DVO-SLAM search for additional loop closure constraints for every keyframe and optimize the whole graph again.

B. KINTINUOUS

Kintinuous maintains a pose graph and uses Speeded Up Robust Feature (SURF) descriptors with the bag-of-words-based DBoW [32] loop detector for place recognition. For optimal performance, Kintinuous adds part of the frames to the place recognition system, the movement distance between which should go beyond a certain threshold. For every frame, a set of SURF keypoints, associated descriptors $U_i \in \Omega \times \mathbb{R}^{64}$ and the depth image for that frame are cached in memory for future queries. Once a loop closure constraint is accepted, Kintinuous performs two optimization steps, firstly on the pose graph and secondly on the dense vertex map. The pose graph optimization provides the measurement constraints for the dense map deformation optimization in place of user specified constraints that were necessary in the original embedded deformation approach [26]. Pose graph optimization is performed using the iSAM framework [33].

C. ORB-SLAM2

In ORB-SLAM2, loop closing is performed in two steps: first, a loop is detected and validated using bag-of-words-based DBoW [32] loop detector, and second, the loop is corrected optimizing a pose graph. Compared to monocular ORB-SLAM, where scale drift may occur [8], the stereo/depth information makes scale observable and the geometric validation and pose-graph optimization no longer require dealing with scale drift and are based on rigid body transformations instead of similarities. The way to achieve pose-graph optimization is to minimize the cost function as:

$$C = \sum_{i,j} (\mathbf{e}_{i,j}^T \Lambda_{i,j} \mathbf{e}_{i,j}) \tag{37}$$

where $\Lambda_{i,j}$ is the information matrix in an edge of the pose-graph, $\mathbf{e}_{i,j}$ is the error of the edge, as:

$$\mathbf{e}_{i,j} = \log_{\mathrm{sim}(3)} \left(\mathbf{S}_{ij} \mathbf{S}_{jw} \mathbf{S}_{iw}^{-1} \right) \tag{38}$$

where $\mathbf{S}_{ij}$ is the relative Sim(3) transformation between both keyframes. The $\log_{\mathrm{sim}(3)}$ transforms to the tangent space.

After pose graph optimization, ORB-SLAM2 performs a full BA optimization in a separate thread to achieve the optimal solution.

D. ElasticFusion

ElasticFusion performs local loop closing and global loop closing for small loops and global loops. For local loop closure detection, ElasticFusion divides map $\mathcal{M}$ into two disjoint sets: active set Θ and inactive set Ψ, according to their collection time. Then ElasticFusion registers the predicted surface renderings of Θ and Ψ from the latest pose estimate $\mathbf{P}_t$ and gets the relative transformation matrix $\mathbf{H} \in \mathbb{SE}_3$ from Θ to Ψ. ElasticFusion will check the quality of this registration and decide whether or not to carry out a deformation, which will optimize the camera pose and the map. If the quality of the registration is high enough, ElasticFusion will produce a set of surface constraint $\mathcal{Q}$ as the deformation graph optimisation to align the surfels, where $\mathcal{Q}$ is defined as:

$$\mathcal{Q}^p = \left((\mathbf{H}\mathbf{P}_t)\, \mathbf{p}\left(\mathbf{u}, \mathcal{D}_t^a\right); \mathbf{P}_t \mathbf{p}\left(\mathbf{u}, \mathcal{D}_t^a\right); \mathcal{T}_t^i(\mathbf{u}); t \right) \tag{39}$$

where $\mathcal{D}_t^a$ is the estimation of depth map of the latest pose, $\mathcal{T}_t^i(\mathbf{u})$ represents the estimated transformation of every pixel.

As for global loop closure, ElasticFusion utilises an improved randomised fern encoding approach for appearance-based place recognition with the difference that instead of encoding and matching against raw RGB-D frames, ElasticFusion uses predicted views of the surface map once they are aligned and fused with the live camera view. The surface constraint $\mathcal{Q}$ for global loop closure is defined as:

$$\mathcal{Q}^p = \left(\left(\mathbf{H}\mathcal{E}_{\mathbf{P}}^i \right) \mathbf{p}\left(\mathbf{u}, \mathcal{D}_t^a\right); \mathbf{P}_t \mathbf{p}\left(\mathbf{u}, \mathcal{D}_t^a\right); \mathcal{E}_t^i; t \right) \tag{40}$$

After the global loop was found, ElasticFusion align the matched frame with the current model prediction. If the alignment is successful, a relative transformation matrix $\mathbf{H} \in \mathbb{SE}_3$ which brings the current model prediction into alignment with the matching frame will be resolved. And the remaining operation is the same as local loop closing.

VII. ADVANCED TOPICS AND OPEN PROBLEMS

A. SEMANTIC MAPPING

As the rapid development of semantic segmentation, it is beneficial to combine SLAM and semantic segmentation to get a semantic map as well as enhance the SLAM's accuracy and robustness. SLAM++ [34] builds an object database before scanning and uses PPFs to detect objects and obtain their 6Dof pose. The object pose is added to SLAM optimization to achieve higher accuracy. In 2017, Sunderhauf *et al.* proposed a semantic mapping system, which uses object detection algorithm to detect objects at real-time and builds a map made up

of objects [35]. In the same year, McCormac *et al.* proposed a semantic mapping system that fuses dense 3d map with semantic labels [36]. What's more, object detection results can be used to detect dynamic objects and remove them, improving the system's robustness and accuracy [25], [37].

B. DYNAMIC ENVIRONMENTS RECONSTRUCTION

The traditional RGB-D SLAM methods mainly focus on localization and reconstruction in static scenes. For more realistic situations, there are often dynamic objects in the scene, which brings challenges to previous methods. How to improve the performance of RGB-D SLAM in dynamic scenes has become a new research hotspot. In 2017, Sun *et al.* proposed a motion removal approach acted as a preprocessing module to remove the associated data and moving objects, and integrated this module into the front end of RGB-D SLAM to improve the performance in challenging dynamic environments. Following this work, in 2018, Sun *et al.* proposed an online motion removal method that does not require prior knowledge, such as the semantic or visual appearance information of moving objects. Different from these works, in 2020, Vincent et al proposed a pipeline that uses deep neural networks to extend Kalman filters and visual SLAM to improve tracking and mapping in dynamic environment.

C. EDGE-BASED AND PLANE-BASE METHODS

Traditional geometric RGB-D SLAM algorithms only use point feature or part of pixels for camera pose estimation, while being robust to camera rotation, light variation and scale change to some extent, they are easy to drift and can not handle textureless environment very well. The application of lines and planes in the RGB-D SLAM will help exploit the structural regularities of indoor scenes, thus improving the system's performance when the scene has weak texture but strong structural priors.

Changhyun Choi *et al.* is of the first ones that combine edge detection and RGB-D SLAM [44]. In their method, both 3D shape information and photometric texture information are used to detect 3D edges in RGB-D point clouds. Finally, edge points with ICP algorithm are used for camera pose estimation. RGB-D SLAM based on lines performs well in textureless environment or in the case of motion blur, but can not work in scenes with little lines. To overcome this limitation, [45] uses both points and lines for RGB-D SLAM. In [46]–[50] and [51], direct edge alignment is proposed that minimizes the sum of squared distances between the reprojected and the nearest edge point using the distance transform of the edge-map, with other errors like photometric error or ICP-based point-to-plane distance minimized together [1].

D. MULTI-SENSOR FUSION

The result of RGB-D SLAM can be combined with other sensor estimations to achieve better robustness and accuracy. [52] incorporates semantic features and IMU measurements for RGB-D odometry and local bundle adjustment. [53] mounts an RGB-D camera on a robot hand and performs SLAM in the configuration space of the robot instead of the pose space of the camera, which improves robustness to missing or ambiguous depth data compared to approaches that are unconstrained by the robot's kinematics. [54] combines a direct VO with IMU measurements and robot kinematics to obtain a semi-dense map which can be used for robot collision free motion planning.

E. NON-RIGID RECONSTRUCTIONS

Dynamic scenes with deformable or moving objects will disturb normal camera registration and map building, which makes non-rigid reconstruction a popular research area in recent years. As for the reconstruction of scenes with moving objects, [55] classifies the scene parts into static and dynamic using probabilistic segmentation, and fuses the static parts and discard the dynamic ones. [56] tracks and reconstructs moving objects while reconstructs the static environment. [57] performs a two-fold segmentation of the scene and divides the scene into static or moving elements of rigid clusters. This method gets robust and accurate motion estimation results in dynamic environments on a multi-core CPU. As for the reconstruction of deformable objects, [58]–[60] set up multi-camera systems, at the cost of high cost, complex maintenance, and lack of portability. [61] use a single camera for non-rigid reconstruction but relies on a template prior to reducing the difficulty. [62] proposed a template-less non-rigid reconstruction method with a single RGB-D camera, using an efficient local-to-global hierarchical optimization framework. [63] removes the need for template prior to compute deformations by optimizing a global alignment problem and use an as-rigid-as-possible constraint to eliminate the shrinkage problem of the deformed shapes, especially near open boundaries of scans.

F. ROLLING SHUTTER DISTORTION

To achieve accurate camera pose estimation, it is important to consider a shutter type. In rolling shutter cameras, each row of a captured image is taken by different camera poses and it is difficult to calculate a pose for each row respectively. Most RGB-D SLAM algorithms assume a global shutter, and these algorithms estimate one camera pose for each frame, while most consumer RGB-D cameras employ a rolling shutter for its low cost. To overcome this problem, [5] proposes a benchmark with global shutter image data and [64]–[66] uses the interpolation-based approach to estimate rolling shutter camera pose, which applies a spline function to interpolate a camera trajectory.

VIII. EVALUATION

To obtain an intuitive understanding of the performance of different methods for different application scenarios, we select 7 RGB-D SLAM systems to perform the quantitative evaluation with TUM [67], ICL-NUIM [68] and ETH3D [5] datasets with the help of relevant source code.

As we have introduced in the survey part, classified by camera tracking, ORB-SLAM2 and RGB-D SLAM v2 are

TABLE 1. Table summarizing the algorithms used in our experiments. We mark with a √ when the functionality is included.

Algorithm	Camera Tracking	Local Mapping	Loop Closure	Implementations	Year
DVO-SLAM [23]	Direct	Point-Based	√	C++	2013
KinectFusion [12]	Direct	Volumetric		C++, OpenMP, OpenCL, CUDA	2011
ElasticFusion [24]	Direct	Point-Based	√	C++,CUDA,OpenGL	2015
RGBDSLAM_v2 [2]	Indirect	Volumetric	√	C++, ROS, OpenGL, SiftGPU	2014
ORB-SLAM2 [3]	Indirect	Point-Based	√	C++	2016
BundleFusion [4]	Hybrid	Volumetric	√	C++, CUDA	2017
BAD-SLAM [5]	Direct	Point-Based	√	C++, CUDA, OpenGV	2019
RGBDTAM [13]	Direct	Point-Based	√	C++	2017
ID-RGBDO [14]	Direct	Point-Based		C++	2020
Plane-Edge-SLAM [18]	Indirect	Point-Based	√	C++	2020

the most popular SLAM systems based on the indirect method, which track the camera pose using geometric error of the features. KinectFusion, ElasticFusion, DVO-SLAM and BAD-SLAM are four representative direct methods, which evaluate the motion of the camera using photometric error, based on frame-to-frame or frame-to-model tracking method. BundleFusion is a hybrid method that combine direct methods and indirect methods by minimizing both photometric error and geometric error to get the camera pose. When it comes to the representation model of local mapping, KinectFusion, BundleFusion and RGB-D SLAM v2 are volumetric method, the first two use the TSDF model to represent scene geometry, while RGB-D SLAM v2 is based on the occupancy voxel. Other four are point-based methods. Among all methods, KinectFusion is the only one without the loop closing module.

The criterion for our selection is to cover direct-based/indirect-based tracking; volumetric-based/point-based mapping, and with/without loop detection strategy, also we want to compare more about the classic and advanced work. So that we choose these representative systems, according to the methods and attributes of them, and enumerate their attribute in Table 1. We have also introduced 3 relatively advanced RGB-D SLAM systems, include two direct-based tracking methods: RGBDTAM [13], ID-RGBDO [14] and one indirect-based methods Plane-Edge-SLAM [18], and we have also introduced their tracking method in the survey part. Due to the lack of open source code or implementation difficulties, we did not conduct experiments and only refer to the experimental data provided in their articles. This may not guarantee an absolute fair comparison. (different hardware conditions) but the main purpose of this article as a review paper is only to gain some intuitive understanding of different RGB-D SLAM systems.

A. EVALUATION ENVIRONMENT

1) EVALUATION CRITERIA

In this paper, we use the absolute trajectory error with SE(3) alignment (SE(3) ATE RMSE c.f. [67]) to measure the tracking accuracy of the RGB-D SLAM systems above. The absolute trajectory error, ATE for brevity, is the absolute error between the estimated pose and the ground truth pose, which can intuitively reflect the accuracy and the global consistency of the trajectory. It should be noted that the estimated pose and ground truth pose are usually not in the same coordinate system, so we need to calculate a conversion matrix $\mathbf{S} \in \mathbb{SE}_3$ from the estimated pose to the ground truth pose by the least square method to align them. For a set of estimated poses $P_{1:n}$ and ground truth poses $Q_{1:n}$, the ATE at time i is defined as:

$$F_i = {Q_i}^{-1}\mathbf{S}P_i \tag{41}$$

The ATE RMSE is:

$$\mathbf{RMSE}(F_{i:n}) = (\frac{1}{n}\sum_{i=1}^{n} \|trans(F_i)\|^2)^{1/2} \tag{42}$$

where $trans\,(\cdot)$ represents the translation part of ATE.

2) DATASET

We conduct a series of experiments on the TUM [67], ICL-NUIM [68] and ETH3D [5] datasets. The TUM dataset is a huge dataset containing RGB-D data, which is widely used to evaluate the quality and performance of visual odometry and visual SLAM algorithms. The true value of the camera trajectory was collected using a high-precision camera motion capture system composed of eight high-speed motion cameras (100 Hz). The ICL-NUIM dataset is an RGB-D dataset of two different synthetic scenes (the living room and the office room scene), provided with ground truth trajectories and depth-maps. The ETH3D dataset is a well-calibrated benchmark dataset for RGB-D SLAM, captured by synchronized global shutter cameras(while the other two dataset are both captured by local shutter cameras).

3) PARAMETERIZATION

We use the relative open source code to evaluate the 7 SLAM systems in different datasets. For RGBD-SLAM v2, we run it with different parameters in different sequences, while for other SLAM systems, we use the default parameter of the relative source code for every sequences, that is because using

TABLE 2. Tracking accuracy results on the easy TUM RGB-D dataset. ATE RMSE (*m*).

Method	Tracking	fr1_desk	fr1_floor	fr1_room	fr2_xyz	fr3_office	fr3_nst
DVO-SLAM	Direct	0.021	0.016	0.047	0.018	0.035	0.018
ElasticFusion	Direct	0.025	0.245	0.226	0.011	0.017	0.016
RGBDSLAM_v2	Indirect	0.026	0.029	0.084	0.008	0.032	0.017
ORB-SLAM2	Indirect	**0.016**	0.016	**0.045**	**0.004**	**0.010**	0.019
BundleFusion	Hybrid	**0.016**	0.124	0.136	0.011	0.022	**0.012**
BAD-SLAM	Direct	0.017	0.035	0.112	0.011	0.017	0.019
RGBDTAM	Direct	0.027	-	0.155	**0.004**	0.027	0.016
ID-RGBDO	Direct	0.051	0.020	-	0.007	0.038	0.018
Plane-Edge-SLAM	Indirect	0.020	**0.009**	-	0.008	0.015	0.015

The results of the bottom three lines are from the corresponding literature. "-" means no result due to tracking failure or missing data, and bold means the best result, same below.

different parameters for different sequences has a greater impact on the performance of RGB-D SLAM v2, while we have also did some fine tuning on other systems, there is no significant change compared to the default. Specifically, for RGB-D SLAM v2, we mainly fine tune three parameters to get a reasonable running result for every sequences. These three parameters include: the number of matching immediate predecessors(represented as n), randomly sampled keyframes(represented as k) and randomly sampled frames for loop closures(represented as l). For feature-rich and short sequences, we set the three values low, while for longer or texture-less sequences, we increase the values, according to [2]. For the short and feature-rich sequences "fr1_desk","fr1_floor","fr1_room", "fr3_office" and "fr3_nst" in TUM RGB-D dataset and all sequences in ICL-NUIM dataset, we set $n = 2$, $k = 2$, $l = 5$, for more texture-less sequences "fr2_xyz" in TUM, we set $n = 4$, $k = 4$, $l = 10$, for the long and texture-less sequences "fr2_360_hemisphere", "fr2_large_no_loop" and "fr2_large_with_loop" in TUM, we set $n = 8$, $k = 8$, $l = 20$.

4) TEST ENVIRONMENT

We use a PC with an Intel Core i7 3.6GHz CPU and 32 GB of memory for experiments. The systems were implemented from the open source code. The web address of the source code and RGB-D SLAM dataset we used and mentioned in the article are all listed in the section IX.

B. EVALUATION ON TRACKING ACCURACY

1) TUM RGB-D DATASET

The Table 2 and Table 3 shows the result of tracking accuracy in TUM RGB-D dataset of the 10 SLAM systems we choosed, where Table 2 contains relatively simple sequences, while the sequences in Table 3 are more challenging, which have longer trajectories, sparser textures and less structure. (Note that we did not show the experimental results of KinectFusion on all sequences, and BAD-SLAM on Table2, because tracking lost occurred on these sequences.) We can see that in all the SLAM systems, ORB-SLAM2 clearly has the best tracking accuracy on most sequences. Comparing with the SIFT feature-based tracking used by RGBD-SLAM_v2 and BundleFusion, ORB feature-based tracking method is more robust especially in the cases of fast camera motion and vigorous rotation, as the result shown in the sequence "fr1_room" and "fr3_office". In addition, ORB-SLAM2 has more advanced global mapping and loop closure modules. In combination with ORB features, it can reliably reconstruct a globally consistent map, hence having better accuracy. It is worth noting that the new feature-based method Plane-Edge-SLAM also has good performance, and even better than ORB-SLAM2 in "fr1_floor", the scenes with more plane structures. The reason for which, we analyze is that, through the fusion of plane and edge, the estimated pose of Plane-Edge-SLAM is fully constrained, and the plane fitting method can adapt to various measurement noises corresponding to different depth measurements, so that it can achieve higher accuracy in the planar scene. Compared with the feature-based methods mentioned above, the direct methods are generally more inaccurate and there are more tracking failures (as we regard the tracking error above 0.5m or tracking lost as the situation of tracking failures). And in the more challenging large-scale situations, the accuracy difference between the direct methods and feature-based methods is even greater, as the result shown in the Table 3. The cause we analyze is that in the front-end, when the camera moves or rotates on a large scene, it is difficult for direct methods to satisfy the condition of constant gray level in large scene, and the vigorous camera rotation will cause short exposure time and blurred images, which is easy to cause methods based on photometric errors to fall into local extremes. The accuracy difference between direct and indiret method is not that significant in sequence "fr1_desk", "fr2_xyz" and "fr3_nst", and we can see that in "fr2_xyz", the result of RGBDTAM is equal to the ORB-SLAM2. This is because these sequences are not in large-scale, the camera moves slowly and the rotation is not that vigorous, which ensures

TABLE 3. Tracking accuracy results on the hard TUM RGB-D dataset. ATE RMSE (*m*).

Method	Tracking	fr2_360_hemisphere	fr2_large_no_loop	fr2_large_with_loop
DVO-SLAM	Direct	0.189	0.082	0.128
ElasticFusion	Direct	0.627	0.750	1.157
RGBDSLAM_v2	Indirect	0.408	0.134	0.250
ORB-SLAM2	Indirect	**0.124**	**0.117**	**0.097**
BundleFusion	Hybrid	0.430	0.706	0.823

TABLE 4. Tracking accuracy results on the ICL-NUIM dataset. ATE RMSE (*m*).

Method	kt0	kt1	kt2	kt3
DVO-SLAM	0.104	0.029	0.191	0.152
KinectFusion	0.119	0.012	0.167	0.210
ElasticFusion	0.009	0.009	0.014	0.106
RGBDSLAM_v2	0.026	0.008	0.018	0.433
ORB-SLAM2	0.009	0.007	0.016	0.109
BundleFusion	**0.006**	**0.004**	**0.006**	**0.011**

TABLE 5. Tracking accuracy results on the ETH3D dataset. ATE RMSE (*m*).

Method	Buddha	Desk1	Drone	Trashbin
DVO-SLAM	0.036	0.023	0.165	0.019
ElasticFusion	0.734	0.246	0.017	0.581
ORB-SLAM2	0.012	0.009	0.004	**0.002**
BundleFusion	0.045	0.083	-	0.038
BAD-SLAM	**0.003**	**0.004**	**0.003**	**0.002**

that there is almost no motion blur and rolling shutter effects in the data.

2) ICL-NUIM DATASET

As we can see in Table 4, for these sequences in ICL-NUIM dataset, which only cover small scenes and simple camera trajectories (without fast camera motion and vigorous rotation), the only hybrid method BundleFusion has the best tracking accuracy on all chosen sequences, and the feature-based method ORB-SLAM2 and the direct method ElasticFusion are almost in equal performance. This result does not conflict with the result in TUM dataset, as we have explained above, the direct methods rely on gray-scale consistency and are easily affected by factors such as rolling shutter effects, motion blur and the change of external light, while in a sequence without these limits, the direct method can achieve better accuracy. The reason why BundleFusion achieve the best result we analyze is that there are many planar structure (mainly untextured wall and floor) and repeated textures, and lack of corner points in the dataset environment, which can make the feature-based method have larger errors even in a small range, while direct methods are robust to these, so we can say that the increase in accuracy comes from the combination of the direct method and feature-based method adopted by BundleFusion. Specifically, the BundleFusion employs correspondences based on sparse features and dense geometric and photometric matching, which not only ensures less feature matching time, but also achieves higher precision tracking accuracy in low-texture areas.

3) ETH3D DATASET

From the experimental results in Table 5, we can see that BAD-SLAM is fully dominant in each chosen sequence of the ETH3D dataset, only ORB-SLAM2 can compare with it, while other direct methods are much worse than BAD-SLAM. Firstly, we have to mention again, ETH3D dataset is the only dataset captured by synchronized global shutter cameras, while the other two datasets are both captured by local shutter cameras, which means there is no rolling shutter effects for all the SLAM systems in ETH3D dataset. From the previous evaluation results in Table1 and Table 2, we can see that the rolling shutter effects can introduce further geometric distortions, which may strongly affect direct methods, while ORB-SLAM2 as an indirect method is less affected by rolling shutter than direct methods, and can achieve better accuracy. Without the rolling shutter effects, BAD-SLAM achieves higher accuracy than ORB-SLAM. The main reason we analyze is that in the back-end, BAD-SLAM use gradients rather than raw pixel intensities to be robust to photometric changes, and it alternates between refining the 3D map and the camera poses to minimize the number of parameters considered at each point in time, which guaranteed the efficiency of BA optimization, and largely improved the overall tracking accuracy.

C. EVALUATION ON TRACKING TIME

Table 6 shows the tracking time result of the seven SLAM systems on the TUM RGB-D dataset. We can see that the direct method KinectFusion and ElasticFusion have the shortest mean tracking time, followed by the hybrid method BundleFusion, and indirect method ORB-SLAM2. The indirect method RGBDSLAM_v2 has the longest mean tracking time, and significantly longer than other methods. In order to make a clearer comparison of methods besides RGBDSLAM_v2, we plot the result in Fig. 9.

The result proves that direct methods usually run faster than indirect methods, because the preprocessing time and the time-consuming feature matching process is removed. The exception here is ORB-SLAM2, as an indirect method, the mean tracking time is much less than RGBDSLAM_v2,

TABLE 6. Mean tracking time results on the TUM RGB-D dataset. (*s*).

Method	fr1_desk	fr1_floor	fr1_room	fr2_360_hemisphere	fr2_large_no_loop
DVO-SLAM	0.032	0.037	0.032	0.035	0.035
KinectFusion	**0.021**	**0.023**	**0.021**	**0.018**	0.024
ElasticFusion	0.026	0.027	0.025	0.022	**0.023**
RGBDSLAM_v2	0.065	0.071	0.067	0.137	0.147
ORB-SLAM2	0.030	0.031	0.034	0.025	0.025
BundleFusion	0.026	0.028	0.028	0.027	0.028
BAD-SLAM	0.034	0.037	0.035	-	-

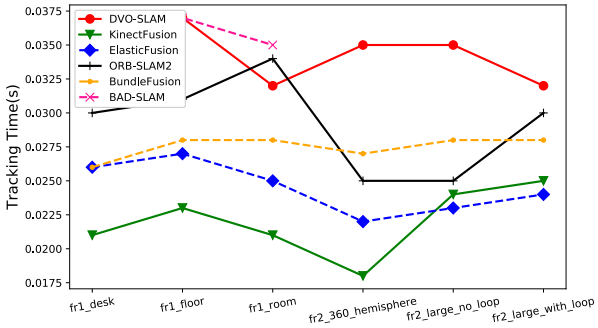


FIGURE 9. Mean tracking time result besides RGBDSLAM_v2.

and even less than some direct methods. The reason we analyze is that the ORB descriptor used in the whole process is computed on the retained FAST corners, which makes the process of corner extraction and descriptor computation very fast. Actually, the ORB feature extraction is much less than 33ms per image, which excludes the SIFT (∼300ms) used by RGBDSLAM_v2, and promises the real-time performance without GPUs.

D. EVALUATION ON RECONSTRUCTION ACCURACY

The ICL-NUIM dataset also provides the ground truth 3D model used to generate the virtually scanned sequences. In addition to the camera tracking evaluation, we evaluate surface reconstruction accuracy (mean distance of the model to the ground truth surface) for the living room model. From the Table 7 we can see that the hybrid method BundleFusion has the best reconstruction accuracy, followed closely by ElasticFusion. In comparison, RGB-D SLAM_v2 and DVO-SLAM are far from them, and KinectFusion, the only method without loop closing, has the lowest accuracy. We can see that the result of reconstruction accuracy corresponds to the result of the tracking accuracy in Table 4, which illustrates the correlation between the two. For the methods which use the frame-to-model mechanism, we can say that high tracking accuracy brings high reconstruction accuracy, and vice versa. The results of ORB-SLAM2 and BAD-SLAM are not given, because the method ORB-SLAM only reconstructs a sparse map, and BAD-SLAM failed in this dataset.

The Fig. 10 shows the quality reconstruction results of three methods in the sequence "fr1_room" on the TUM dataset. Visually, BundleFusion has the highest reconstruction quality in terms of scan completeness and alignment accuracy. While ElasticFusion as the only point-based method among them, has the most sparse reconstruction model, and there are more empty space compared with the other two volumetric-based systems. We can also see that no method can reconstruct the thin structures and objects well, like the spindly legs of tables and chairs. The reason we analyze is that the thin structures often lack distinct point features and have severe self-occlusion, so it is very hard for the traditional image-based or depth-based reconstruction. The recent work Vid2Curve [69] proposes the first approach that simultaneously estimates camera motion and reconstructs the geometry of complex 3D thin structures in high quality, which may be the new research hotspot.

TABLE 7. Surface reconstruction accuracy on the ICL-NUIM dataset. (*m*).

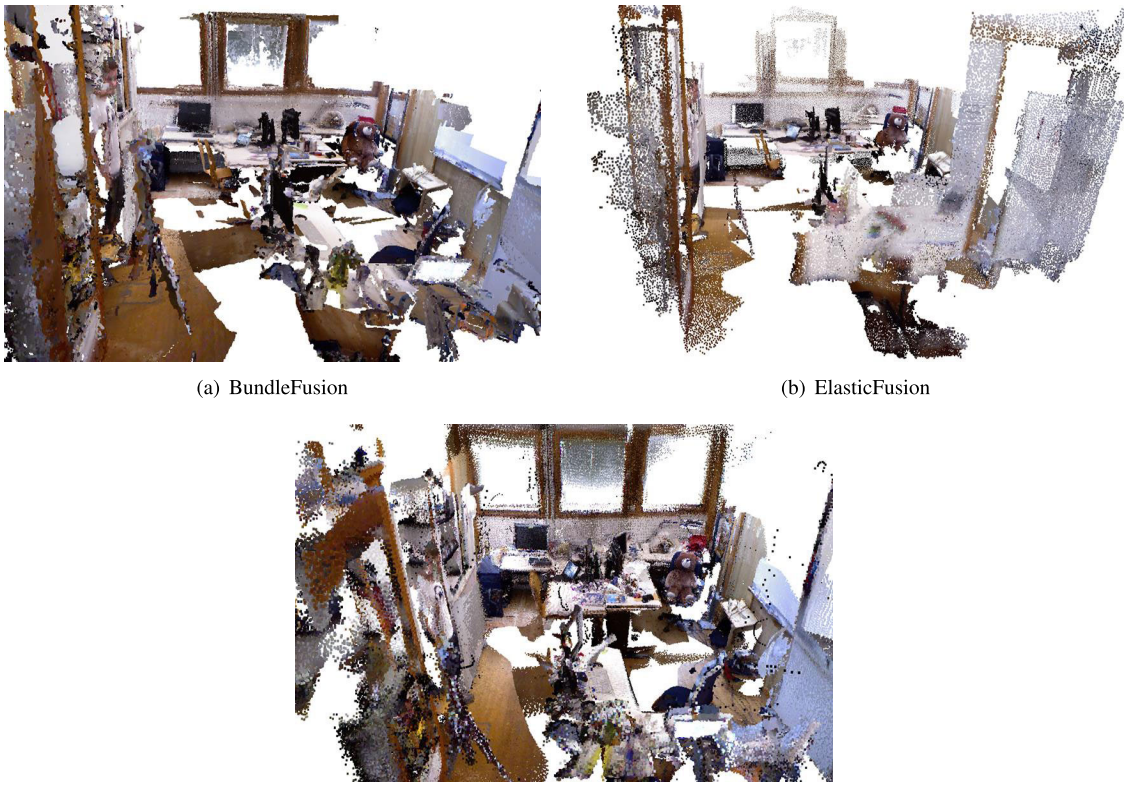
Method	kt0	kt1	kt2	kt3
DVO-SLAM	0.032	0.061	0.119	0.053
KinectFusion	0.071	0.144	0.216	0.359
ElasticFusion	0.007	0.007	0.008	0.028
RGBDSLAM_v2	0.044	0.032	0.031	0.167
BundleFusion	**0.005**	**0.006**	**0.007**	**0.008**

TABLE 8. Loop closing effect evaluation on the TUM RGB-D dataset ATE RMSE. (*m*).

Sequence	with loop closing	without loop closing
fr1_desk	0.016	0.066
fr1_floor	0.016	0.207
fr1_room	0.045	0.161
fr2_360_hemisphere	0.124	0.472
fr3_nst	0.019	0.150

E. EVALUATION ON LOOP CLOSING

We choose ORB-SLAM2 as the test subject, and mainly study the effect of the two cases with or without loop closing on the TUM RGB-D dataset. The result in Table 8 shows that in all chosen sequences, the tracking accuracy of the case with loop closing is much better than without loop closing, and the difference between with and without loop closing is particularly significant in large-scale scenes, as the result shown in the sequence "fr2_360_hemisphere". The



(a) BundleFusion (b) ElasticFusion

(c) RGBDSLAM_v2

FIGURE 10. Reconstruction quality of three systems on the "fr1_room" sequence in TUM.

Fig. 11 shows the tracking trajectories on different datasets. It is obvious that the trajectories of the cases without loop closing reflects that with the long-distance movement of the camera, the error continues to accumulate and increase, and the error even reached nearly one meter in the final stage of the trajectory. The experimental results show that the loop closing mechanism plays a very important role in solving error accumulation and trajectory drift.

F. SUMMARY AND DISCUSSION

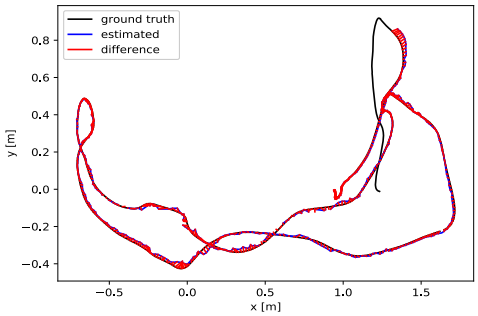
In this subsection, we mainly made a summary about the direct method and indirect method. The mean tracking time result proves the rapidity of the direct methods, since there is no time-consuming feature matching process, and it is precisely because it does not rely on feature matching that the direct method is less prone to matching errors in a scene with sparse texture or large planar structure. However, in the situation of rapid camera motion and vigorous rotation, the direct method is easier to encounter tracking lost, and both the accuracy and robustness are worse than indirect methods. From the evaluation result, we have also found that the indirect methods are less affected by rolling shutter effects, as exhibited by the camera used in the TUM dataset, than direct methods. Here, we summarized the advantages and disadvantages of direct and indirect methods, as well as their suitable applications.

1) ADVANTAGES OF DIRECT METHODS:

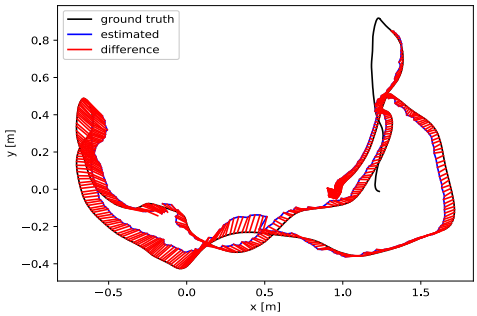
- Compared with the feature point method (only the information around the feature point is used), all the information in the image is used.
- Direct methods can save a lot of time for feature extraction and matching, and easy to be transplanted to embedded systems, and integrated with IMU.
- The pixel gradient is used instead of corner points. It can be used in situations where features are missing. For example, there are many repeated textures or lack of corner points in the environment, but there are many areas with edges or light variables that are not obvious.

2) DISADVANTAGES OF DIRECT METHODS

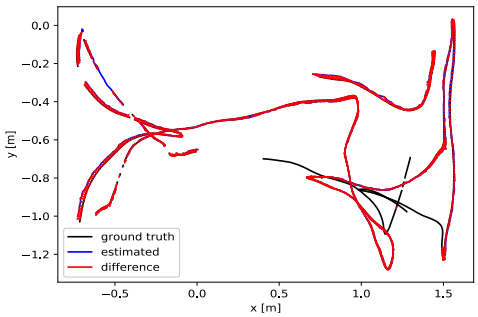
- Generally, the requirements for the camera are relatively high, and a global shutter camera is required for photometric calibration, etc.
- Invariable gray level is a strong assumption that is difficult to satisfy (susceptible to exposure and blurred images).
- There is no discrimination in the single-pixel area, and the image block or correlation needs to be calculated.
- The prerequisite for the success of the direct method is that the objective function has been declining from the initial value to the optimal value, but the image is not convex. Therefore, a fairly good initial estimate and a good quality image are in need.



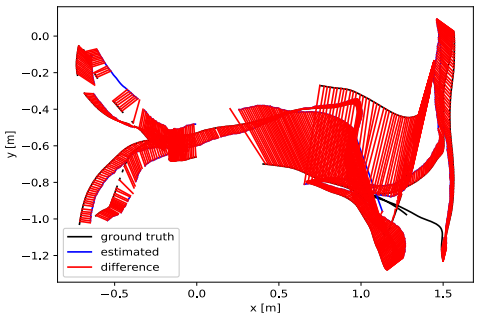
(a) fr1_desk with loop closing



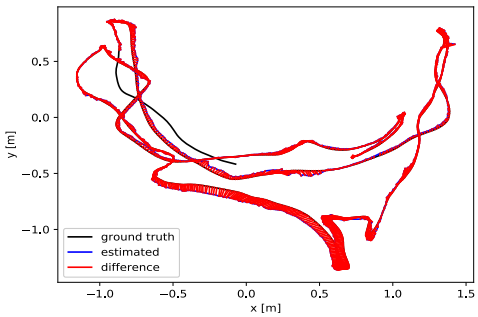
(b) fr1_desk without loop closing



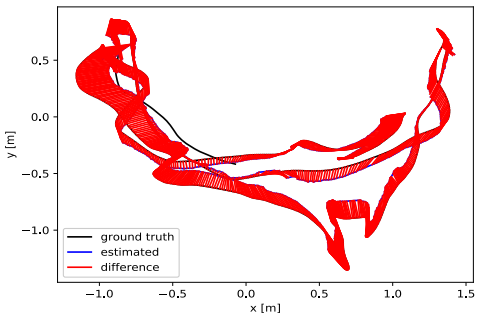
(c) fr1_floor with loop closing



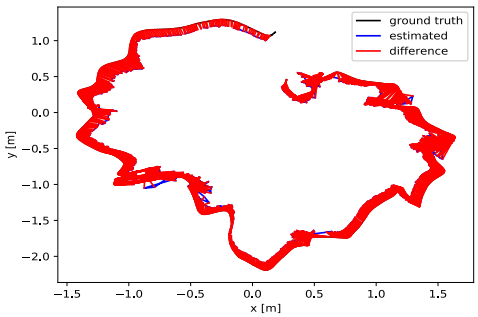
(d) fr1_floor without loop closing



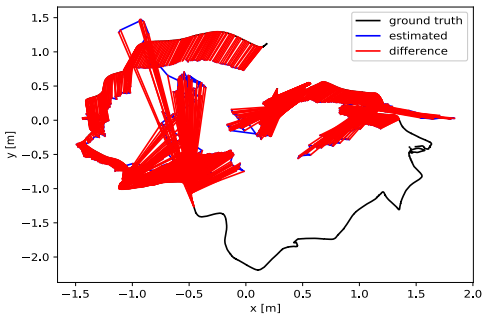
(e) fr1_room with loop closing



(f) fr1_room without loop closing



(g) fr2_360_hemisphere with loop closing



(h) fr2_360_hemisphere without loop closing

FIGURE 11. Tracking trajectories with or without loop detection on the TUM dataset.

- It is difficult to realize map reuse, loop detection and relocation after tracking lost. Unless all keyframe images are stored, it is difficult to use previously built maps, even if there is a way to store all keyframe images. When reusing the map, it also needs to have a more accurate initial estimate of the pose, which is usually difficult.

3) ADVANTAGES OF INDIRECT METHODS

- When the motion is too large, as long as the matching point is still within the pixel, it is unlikely to cause non-matching, which is more robust than the direct method.
- Less affected by rolling shutter effects, which makes it less demanding for the camera than direct methods.
- The robustness for the situation of a fast move and vigorous rotation is much better than direct methods.
- More suitable for loop detection and relocation than direct methods.

4) DISADVANTAGES OF INDIRECT METHODS

- The indirect method can not work well in textureless environments or scenes with very sparse features.
- Spend a lot of time on calculating feature descriptors and matching, which may make it fall short of real-time requirements.
- The feature calculation process puts forward high requirements for computing resources, and may not succeed when computing resources are limited.

IX. RESOURCES

A. BENCHMARK

- RGB-D SLAM Dataset and Benchmark [67]:
  (https://vision.in.tum.de/data/datasets/rgbd-dataset)
  With color, depth images and accelerometer data captured by a Microsoft Kinect, the dataset provides ROS bags and compressed files for downloading. The dataset is recorded in a wide variety of conditions: rotation-only and general motion, static and dynamic environments and small and mid-size scene coverage. What's more, the dataset proposes an evaluation criterion for measuring the quality of the estimated camera trajectory of visual SLAM systems.
- The ETH3D dataset [5]:(https://www.eth3d.net/)
  It is a well-calibrated benchmark dataset for RGB-D SLAM, captured by synchronized global shutter cameras.
- The ICL-NUIM dataset [68]:
  (https://www.doc.ic.ac.uk/%7Eahanda/VaFRIC/iclnuim.html)
  It is an RGB-D dataset of two different synthetic scenes (the living room and the office room scene), provided with ground truth trajectories and depth-maps.

B. SOURCE CODE

- BundleFusion [4]:
  (https://github.com/niessner/BundleFusion)
  Real-time, high-quality 3D mapping method for large scenes.
- DVO-SLAM [23]:
  (https://github.com/tum-vision/dvo_slam)
  RGB-D SLAM combining dense visual odometry and Pose SLAM.
- KinectFusion [12]:
  (https://github.com/GerhardR/kfusion)
- ElasticFusion [24]:
  (https://github.com/mp3guy/ElasticFusion)
  RGB-D scene-centered SLAM system that models the scene as a set of surfels that are deformed to accommodate loop closures.
- ORB-SLAM2 [3]:
  (https://github.com/raulmur/ORB_SLAM2)
  SLAM based on ORB features with high accuracy.
- RGBDSLAM_v2 [2]:
  (https://github.com/felixendres/rgbdslam_v2)
  Implementation of an feature-based RGB-D SLAM system, with a perfect and easy to use GUI.
- BAD-SLAM [5]:
  (https://github.com/ETH3D/badslam)
  RGB-D SLAM based on direct Bundle Adjustment.

X. CONCLUSION

As the development of robotics and VR/AR, vSLAM is receiving more and more attention because of its capability to estimate the camera pose and reconstruct 3D structure of the scene with low cost, small size, and power-efficient cameras. RGB-D camera, a new kind of camera which provides depth information except for RGB information, has many advantages over traditional cameras, like scale awareness and the ability to reconstruct 3D structures for even low texture areas easily and quickly. As a result, RGB-D cameras have been the most popular sensors for indoor reconstruction in the past decade.

In this paper, we summed up the basic architecture of RGB-D SLAM. We divide common RGB-D SLAM algorithm into three main components: camera tracking, local mapping and loop closing. For camera tracking, we described the principle and formulation of direct methods, indirect methods and hybrid methods. And we highlighted the difference between them. For local mapping, we introduced the difference between point-based and volumetric methods. For loop closing, we detailed their process with specific examples. In the experiment part, we evaluated several related properties of the chosen SLAM systems on the main RGB-D datasets, and summarized their ability and analyzed their advantages and disadvantages and their adaptability in different situations. And our evaluation presents valuable guidances to developers for the choice of a proper SLAM systems selection method regarding a particular application.

ACKNOWLEDGMENT

The authors would like to thank R. Mur-Artal, R. A. Newcombe, Thomas Schops, Thomas Whelan, and Angela Dai

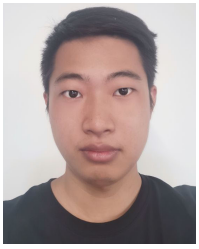
for opening the source codes of their methods for them, and the developers of PCL for making many methods evaluated in this article publicly available. They would also like to thank the Computer Vision Group in Department of Informatics of Technical University of Munich, the Geosensors and Engineering Geodesy group at ETH Zurich, and the Imperial College London for providing their datasets. *(Shishun Zhang and Longyu Zheng contributed equally to this work.)*

REFERENCES

[1] J. Civera and S. H. Lee, "RGB-D odometry and SLAM," in *RGB-D Image Analysis and Processing*. Cham, Switzerland: Springer, Oct. 2019, pp. 117–144.
[2] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.
[3] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[4] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 24:1–24:18, Jul. 2017.
[5] T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle adjusted direct RGB-D SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Long Beach, CA, USA: Computer Vision Foundation, Jun. 2019, pp. 134–144.
[6] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*. Nara, Japan: IEEE Computer Society, Nov. 2007, pp. 225–234.
[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision—ECCV 2014* (Lecture Notes in Computer Science), vol. 8690, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Zürich, Switzerland: Springer, 2014, pp. 834–849.
[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[9] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.
[10] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct sparse mapping," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1363–1370, 2020.
[11] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Cham, Switzerland: Springer, 2011, pp. 1248–1251.
[12] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.
[13] A. Concha and J. Civera, "RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 6756–6763.
[14] A. Fontan, J. Civera, and R. Triebel, "Information-driven direct RGB-D odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4929–4937.
[15] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. 3rd Int. Conf. 3-D Digit. Imag. Modeling*. Quebec City, QC, Canada: IEEE Computer Society, May/Jun. 2001, pp. 145–152.
[16] J. Sturm *et al.*, "Towards a benchmark for RGB-D SLAM evaluation," in *Proc. RGB-D Workshop Adv. Reasoning Depth Cameras Robot., Sci. Syst. Conf. (RSS)*, Los Angeles, CA, USA. Cambridge, MA, USA: MIT Press, 2011.
[17] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision With the OpenCV Library*. Newton, MA, USA: O'Reilly Media, 2008.
[18] Q. Sun, J. Yuan, X. Zhang, and F. Duan, "Plane-edge-SLAM: Seamless fusion of planes and edges for SLAM in indoor environments," *IEEE Trans. Autom. Sci. Eng.*, early access, Nov. 4, 2020, doi: 10.1109/TASE.2020.3032831.
[19] K. Liu, X. Gu, M. Yang, Y. Zhang, and S. Guan, "Semi-direct tracking and mapping with RGB-D camera," in *Intelligent Robotics and Applications* (Lecture Notes in Computer Science), H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and and D. Zhou, Eds. Shenyang, China: Springer, 2019, pp. 461–472.
[20] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, D. Kragic, A. Bicchi, and A. D. Luca, Eds. Stockholm, Sweden: IEEE Press, May 2016, pp. 1285–1291.
[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[22] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 5110–5117.
[23] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, Nov. 2013, pp. 2100–2106.
[24] T. Whelan, S. Leutenegger, F. R. Salas-Moreno, B. Glocker, and J. A. Davison, "Elasticfusion: Dense SLAM without A pose graph," in *Robotics: Science and Systems XI*, L. E. Kavraki, D. Hsu, and J. Buchli, Eds. Rome, Italy: Sapienza Univ. Rome, Jul. 2015.
[25] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. Int. Conf. 3D Vis.* Seattle, WA, USA: IEEE Computer Society, Jun. 2013, pp. 1–8.
[26] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. Mcdonald, "Kintinuous: Spatially extended kinectfusion," Comput. Sci. Artif. Intell. Lab. (CSAIL), MIT, Cambridge, MA, USA, Tech. Rep. MIT-CSAIL-TR-2012-020, 2012.
[27] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, Nov. 2013.
[28] H. Roth and M. Vona, "Moving volume KinectFusion," in *Proc. Brit. Mach. Vis. Conf.*, R. Bowden, J. P. Collomosse, and K. Mikolajczyk, Eds. Surrey, U.K.: The British Machine Vision Association (BMVA), 2012, pp. 1–11.
[29] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "Octree-based fusion for realtime 3D reconstruction," *Graph. Models*, vol. 75, no. 3, pp. 126–136, 2013.
[30] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–16, Jul. 2013.
[31] F. Steinbrucker, C. Kerl, and D. Cremers, "Large-scale multi-resolution surface reconstruction from RGB-D sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, Dec. 2013, pp. 3264–3271.
[32] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.
[33] M. Kaess, A. Ranganathan, and F. Dellaert, "ISAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.
[34] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* Portland, OR, USA: IEEE Computer Society, Jun. 2013, pp. 1352–1359.
[35] N. Sunderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 5079–5085.
[36] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 4628–4635.
[37] B. Bescos, J. M. Facil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.
[38] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auto. Syst.*, vol. 89, pp. 110–122, Mar. 2017.
[39] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robot. Auto. Syst.*, vol. 108, pp. 115–128, Oct. 2018.
[40] J. Vincent, M. Labbé, J.-S. Lauzon, F. Michaud, F. Grondin, and P.-M. Comtois-Rivet, "Dynamic object tracking and masking for visual SLAM," 2020, *arXiv:2008.00072*. [Online]. Available: http://arxiv.org/abs/2008.00072
[41] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," 2020, *arXiv:2010.07820*. [Online]. Available: http://arxiv.org/abs/2010.07820
[42] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robot. Auto. Syst.*, vol. 117, pp. 1–16, Jul. 2019.

[43] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Robust semantic mapping in challenging environments," *Robotica*, vol. 38, no. 2, pp. 256–270, Feb. 2020.
[44] C. Choi, A. J. B. Trevor, and H. I. Christensen, "RGB-D edge detection and edge-based registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, Nov. 2013, pp. 1568–1575.
[45] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*. Santiago, Chile: IEEE Computer Society, Dec. 2015, pp. 3934–3942.
[46] M. Kuse and S. Shen, "Robust camera motion estimation using direct edge alignment and sub-gradient method," in *Proc. IEEE Int. Conf. Robot. Automat.*, D. Kragic, A. Bicchi, and A. D. Luca, Eds. Stockholm, Sweden: Elsevier, May 2016, pp. 573–579.
[47] X. Wang, W. Dong, M. Zhou, R. Li, and H. Zha, "Edge enhanced direct visual odometry," in *Proc. Brit. Mach. Vis. Conf.*, R. C. Wilson, E. R. Hancock, and W. A. P. Smith, Eds. York, U.K.: The British Machine Vision Association (BMVA), 2016, pp. 1–12.
[48] F. Schenk and F. Fraundorfer, "Combining edge images and depth maps for robust visual odometry," in *Proc. Brit. Mach. Vis. Conf.*, London, U.K., 2017, pp. 1–12.
[49] Y. Zhou, H. Li, and L. Kneip, "Canny-VO: Visual odometry with RGB-D cameras based on geometric 3-D–2-D edge alignment," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 184–199, Feb. 2019.
[50] C. Kim, P. Kim, S. Lee, and H. J. Kim, "Edge-based robust RGB-D visual odometry using 2-D edge divergence minimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 1–9.
[51] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 3748–3754.
[52] N. Patel, F. Khorrami, P. Krishnamurthy, and A. Tzes, "Tightly coupled semantic RGB-D inertial odometry for accurate long-term localization and mapping," in *Proc. 19th Int. Conf. Adv. Robot. (ICAR)*, Belo Horizonte, Brazil, Dec. 2019, pp. 523–528.
[53] M. Klingensmith, S. S. Sirinivasa, and M. Kaess, "Articulated robot motion for simultaneous localization and mapping (ARM-SLAM)," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 1156–1163, Jul. 2016.
[54] R. Scona, S. Nobili, R. Y. Petillot, and M. Fallon, "Direct visual SLAM fusing proprioception for a humanoid robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada: IEEE Press, Sep. 2017, pp. 1419–1426.
[55] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 1–9.
[56] M. Rünz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, D. Chu, J. L. Gabbard, J. Grubert, and H. Regenbrecht, Eds. Munich, Germany, Oct. 2018, pp. 10–20.
[57] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3992–3999.
[58] J. Starck, A. Maki, S. Nobuhara, A. Hilton, and T. Matsuyama, "The multiple-camera 3-D production studio," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 6, pp. 856–869, Jun. 2009.
[59] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3D full human bodies using kinects," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 4, pp. 643–650, Apr. 2012.
[60] G. Ye, Y. Liu, N. Hasler, X. Ji, Q. Dai, and C. Theobalt, "Performance capture of interacting characters with handheld kinects," in *Computer Vision—ECCV 2012* (Lecture Notes in Computer Science), vol. 7573, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Florence, Italy: Springer, pp. 828–841.
[61] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai, "Robust non-rigid motion tracking and surface reconstruction using l0 regularization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*. Santiago, Chile: IEEE Computer Society, Dec. 2015, pp. 3083–3091.
[62] K. Wang, G. Zhang, and S. Xia, "Templateless non-rigid reconstruction and motion tracking with a single RGB-D camera," *IEEE Trans. Image Process.*, vol. 26, no. 12, pp. 5966–5979, Dec. 2017.
[63] J. Yang, D. Guo, K. Li, Z. Wu, and Y.-K. Lai, "Global 3D non-rigid registration of deformable objects using a single RGB-D camera," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4746–4761, Oct. 2019.
[64] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," in *Proc. Brit. Mach. Vis. Conf.*, T. Burghardt, D. Damen, W. W. Mayol-Cuevas, and M. Mirmehdi, Eds. Bristol, U.K.: The British Machine Vision Association (BMVA), Sep. 2013, p. 8.
[65] C. Kerl, J. Stückler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras," in *Proc. IEEE Int. Conf. Comput. Vis.* Santiago, Chile: IEEE Computer Society, Dec. 2015, pp. 2264–2272.
[66] J.-H. Kim, C. Cadena, and D. I. Reid, "Direct semi-dense SLAM for rolling shutter cameras," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, D. Kragic, A. Bicchi, and A. D. Luca, Eds. Stockholm, Sweden: IEEE Press, May 2016, pp. 1308–1315.
[67] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.
[68] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 1524–1531.
[69] P. Wang, L. Liu, N. Chen, H.-K. Chu, C. Theobalt, and W. Wang, "Vid2curve: Simultaneous camera motion estimation and thin structure reconstruction from an RGB video," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 132–141, 2020.

SHISHUN ZHANG received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018, where he is currently pursuing the master's degree with the National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Automation. His research interests include RGB-D SLAM, pointcloud registration, and deep learning.



LONGYU ZHENG received the master's degree from the Huazhong University of Science and Technology, Wuhan, China, in 2020. His research interests include RGB-D SLAM, semantic segmentation, and neural networks.

WENBING TAO (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligent systems from the Huazhong University of Science and Technology, Wuhan, China, in 2004. He was a Research Fellow with the Division of Mathematical Sciences, Nanyang Technology University, Singapore, from 2008 to 2009. He is currently a Full Professor with the School of Automation, Huazhong University of Science and Technology. He has authored or coauthored numerous articles in image processing and object recognition. His current research interests include computer vision, image segmentation, object recognition, and tracking. He also serves as a reviewer for many journals, including the *International Journal of Computer Vision*, *Pattern Recognition*, and the IEEE TRANSACTIONS ON IMAGE PROCESSING.