IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 July 2024, accepted 11 August 2024, date of publication 13 August 2024, date of current version 23 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3443065

TOPICAL REVIEW

A Survey of Indoor 3D Reconstruction Based on RGB-D Cameras

JINLONG ZHU, CHANGBO GAO, QIUCHENG SUN, MINGZE WANG, AND ZHENGKAI DENG
School of Computer Science and Technology, Changchun Normal University, Changchun 130032, China
Corresponding author: Changbo Gao (QX202200148@stu.ccsfu.edu.cn)

This work was supported by Jilin Provincial Natural Science Foundation under Grant YDZJ202301ZYTS529.

ABSTRACT With the advancement of consumer-grade RGB-D cameras, obtaining depth information for indoor 3D spaces has become increasingly accessible. This paper systematically reviews 3D reconstruction algorithms for indoor scenes using these cameras, serving as a reference for future research. We cover reconstruction processes and optimization algorithms for both static and dynamic scenes. Additionally, we discuss commonly used datasets, evaluation metrics, and the performance of various reconstruction algorithms. Findings indicate that the balance between reconstruction quality and speed in static scene reconstruction, as well as deformation, occlusion, and fast motion of objects in dynamic scenes are currently major concerns. Deep learning and Neural Radiance Fields (NeRF) are poised to provide new perspectives and methods to address these challenges.

INDEX TERMS 3D reconstruction, indoor scenes, static scenes, dynamic scenes, deep learning, neural radiance fields.

I. INTRODUCTION

In recent years, with the rapid development of computer vision and artificial intelligence technologies, the application of 3D reconstruction technology in indoor environments has received widespread attention. Depending on the input data, 3D reconstruction algorithms can be divided into RGB-D camera-based, stereo array-based, visual-inertial-based, and monocular pure RGB-based types. Compared to other reconstruction methods, the RGB-D camera-based approach can directly obtain the color and depth information of each pixel, reducing the complex depth calculation process and more accurately capturing the geometric structure of objects. Additionally, this method has advantages such as high reconstruction accuracy, fast speed, and high system integration, making it very suitable for complex indoor reconstruction tasks. With the advent of consumer-grade depth cameras like Kinect and RealSense, their lower cost and higher real-time performance have greatly promoted the development and application of indoor 3D reconstruction. In smart home systems, using RGB-D cameras for indoor 3D reconstruction can generate accurate home models, enhancing the automation and intelligence levels of smart home systems. In logistics and warehouse management, it can increase the automation level of warehouse management and logistics operations, improving efficiency and accuracy. In robot navigation applications, using RGB-D cameras to generate 3D maps of the environment can enhance the robot's autonomous navigation capabilities and task execution efficiency, improving its adaptability in complex environments. This has led many scholars to research indoor 3D reconstruction algorithms based on RGB-D cameras.

Reference [1] provides a comprehensive overview of the application of visual odometry and visual SLAM in the field of mobile robotics, discussing various sensor data fusion methods and emphasizing their application in actual robot navigation. Reference [2] summarizes the latest advancements in indoor scene modeling and discusses public datasets and programming libraries, but the technologies covered are only up to 2015. Reference [3] divides indoor scenes into static and dynamic scenes, mainly focusing on summarizing traditional reconstruction algorithms, with less attention to emerging deep learning methods. Reference [4] discusses in detail the working principles, applications, and role of RGB-D cameras in 3D reconstruction, introducing relevant datasets and future research directions. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Miaohui Wang.

it lacks specific performance comparisons of the latest algorithms and technologies. Reference [5] is a recent review article on the latest indoor reconstruction algorithms, covering various RGB-D camera technologies and their application scenarios, but this article mainly focuses on static 3D reconstruction algorithms, with less consideration for applications in dynamic environments.

As we can see, some scholars have already summarized indoor reconstruction algorithms based on RGB-D cameras, but these studies have their limitations. Moreover, the development of deep learning technologies and neural radiance fields (NeRF) has also provided new directions for this field. Therefore, it is necessary to comprehensively review and summarize the applications of RGB-D cameras in indoor 3D reconstruction, providing a systematic knowledge framework to help researchers quickly understand the latest advancements and key technologies in this field. The main contributions of this paper are as follows: First, we classify indoor 3D reconstruction algorithms based on RGB-D cameras into static and dynamic scenes, revealing the advantages and disadvantages of each method through classification and comparison of different technical approaches, aiding researchers in choosing the most suitable technical path and optimizing existing methods. Second, we summarize the general process of static and dynamic 3D reconstruction algorithms and outline different reconstruction algorithms at each stage. Third, we update the applications of deep learning and neural radiance fields in this field, analyzing their advantages and disadvantages, providing new directions and solutions for future research. Fourth, we provide comprehensive RGB-D datasets and evaluation standards, offering reliable resources and tools to facilitate researchers in technical validation and performance comparison.

The main structure of this paper is as follows: Section II briefly reviews the development history of major static and dynamic 3D reconstruction algorithms in recent years. Section III provides a description of static scene reconstruction, dividing the reconstruction pipeline into different steps, and detailing the optimizations made by various researchers in each step. Section IV focuses on three-dimensional reconstruction of dynamic scenes, with the processing of dynamic objects being the main research content of this chapter. Section V introduces the datasets and evaluation metrics used in the research process. Finally, Section VI summarizes this paper.

II. RELATED WORK

In this subsection, we briefly review the development history of the main algorithms for indoor 3D reconstruction based on RGB-D cameras. Figure 1 shows some classic algorithms organized according to the timeline.

In static scenes, RGB-D cameras are the only moving objects. By capturing the camera's trajectory, we can fuse the obtained depth data into a reconstructed model, and then extract the surface to generate a static 3D model. Reference [6] proposed the first algorithm, KinectFusion, which utilizes an RGB-D camera for real-time 3D reconstruction. Additionally, they outlined a typical reconstruction pipeline for static 3D reconstruction, comprising depth map processing, camera pose estimation, scene reconstruction, and surface extraction. However, KinectFusion is limited by the voxel model and memory, and it can only perform reconstruction on small scenes. Kintinuous [7] extended KinectFusion to large scene reconstruction by moving the voxel model. In addition, it integrated loop detection and optimization, greatly improving the reconstruction quality. Moreover, Voxelhashing [8] employed voxel hashing as a model storage approach, significantly enhancing storage efficiency. Redwood [9] used an offline method to segment the input RGB-D sequence and separately reconstructed each segment, and then used the keyframes that overlap between the segments to register them, thereby reducing the accumulated error and obtaining high-quality 3D models. These methods are all based on voxel models. ElasticFusion [10] creatively used surfel representation to continuously optimize the reconstructed map and improve the accuracy of reconstruction and pose estimation. It can achieve real-time high-quality surface reconstruction of small scenes. Bundlefusion [11] integrated the research ideas of predecessors and proposed a parallel optimization framework that fully utilized sparse features and dense geometry and photometric terms to perform a sparse-to-dense correspondence matching. In terms of pose optimization, They used a local-to-global blocking strategy and added robust tracking ability to recover from tracking failures (i.e., relocalization), which can generate reconstructions of higher quality in real-time compared to offline methods [9]. With the development of advanced deep learning models [12], [13] and artificial intelligence models in multimodal learning [14], [15], applying advanced neural networks to scene reconstruction has also become a significant trend. PointGroup [16] and 3D-MPA [17] applied U-Net and graph convolutional networks to 3D scenes, respectively, achieving segmentation of 3D point clouds. Reference [18] transferred pre-trained ViTs to the RGB-D domain for 3D object recognition, cross-modally fusing the RGB and depth representations co-encoded by ViT. TR3D [19] used fusion modules to transform traditional 3D object detection methods into multimodal detection methods, demonstrating impressive performance improvements. An overview of static 3D reconstruction algorithms is shown in Table 1.

In practical situations, it is inevitable to encounter dynamic objects in a scene, such as people walking or pets playing. Therefore, the assumption of a completely static environment can be easily broken. In this case, not only is the camera moving, but also the dynamic objects in the scene are moving, which makes it difficult to track the camera trajectory and leads to reconstruction failure. Therefore, we must deal with these dynamic objects. Before processing dynamic objects, it is necessary to first identify them. Since dynamic objects in a scene have different motion tendencies than static backgrounds, we can distinguish them by analyzing such motion characteristics [29], [30], [31], [33]. Another method

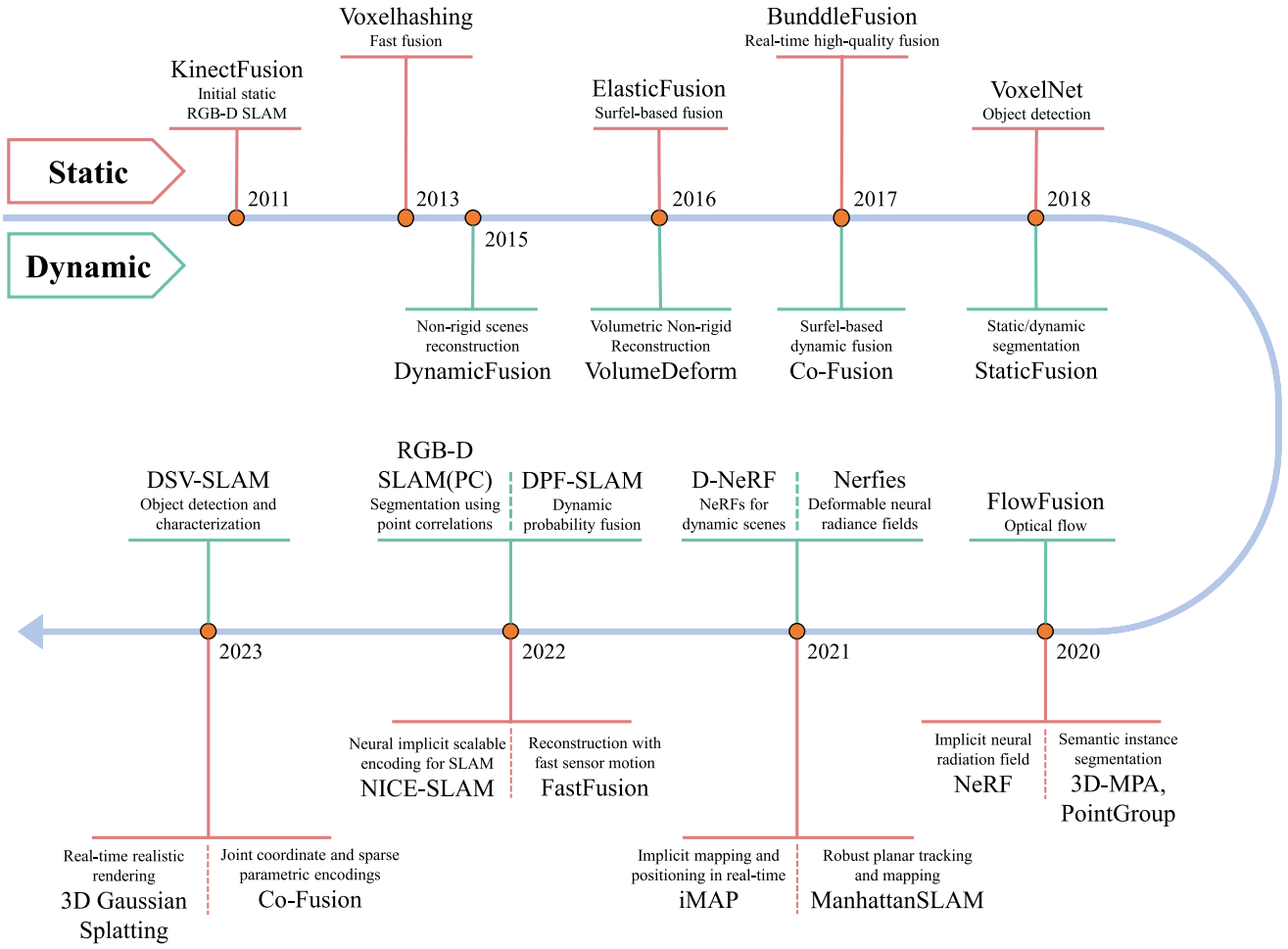

FIGURE 1. The research history of 3D reconstruction in static and dynamic scenes.

to identify dynamic objects is based on deep learning [32], [34], [35], using prior knowledge and semantic information to directly segment dynamic objects. For camera pose estimation, a straightforward method is to treat the data of dynamic objects as outliers and remove them to eliminate their influence on camera pose [36], [37], [39], [40]. However, direct removal of dynamic objects may result in information loss and affect the quality of scene reconstruction. In contrast, using the features of dynamic objects for pose estimation is more meaningful and beneficial [38], [41]. Additionally, the model fusion strategy for dynamic scene reconstruction is also improved accordingly based on the static fusion strategy [29], [42], [43]. An overview of the dynamic 3D reconstruction algorithm is shown in Table 2.

Different from traditional reconstruction methods, [44] proposed an implicit Neural Radiance Field (NeRF) to represent three-dimensional scenes. It utilizes Multilayer Perceptrons (MLP) to learn the 3D information of the scene and can synthesize new viewpoint images through volume rendering. Compared to complex traditional reconstruction processes, NeRF's reconstruction process is simpler and can provide a more continuous representation of the scene. NeRF's implicit representation method provides a new direction for indoor 3D reconstruction based on RGB-D cameras, and its implicit scene representation method further improves the quality of scene reconstruction. In static reconstruction, iMAP [23] first demonstrated that MLP can be the sole scene representation in real-time SLAM systems with handheld RGB-D cameras. NICE-SLAM [26] combined hierarchical scene representation and neural implicit representation to achieve real-time, efficient, and detailed RGB-D surface reconstruction in large-scale scenes. Reference [45] effectively utilized RGB-D data by combining implicit functions (truncated signed distance function, TSDF) and volumetric radiance fields, improving the accuracy and completeness of geometric reconstruction. However, due to the use of multilayer perceptrons and complex optimization algorithms, this method has a long computation time and is not suitable for real-time applications. GO-Surf [46] built on [45] by directly optimizing multiresolution feature grids and the signed distance function (SDF) to achieve fast and accurate surface reconstruction. Recently, [47] proposed

TABLE 1. Overview of RGB-D-based Static 3D Reconstruction Methods: This report discusses the details of different algorithms from the aspects of camera tracking, model fusion, and loop closure, all of which are key technologies for static 3D reconstruction methods based on RGB-D cameras.

Method	Camera Tracking		Model Fusion			Loop Closure	Year
	ICP	Feature	Voxel-based	Surfel-based	NeRF-based		
KinectFusion [6]	✓	-	✓	-	-	-	2012
Kintinuous [7]	✓	-	✓	-	-	✓	2012
Octree-based Fusion [20]	✓	-	-	-	-	-	2013
Voxel hashing [8]	✓	-	-	-	-	-	2013
ElasticFusion [10]	✓	-	-	✓	-	-	2015
InfiniTAM v3 [21]	✓	-	✓	-	-	✓	2017
BundleFusion [11]	✓	-	✓	-	-	✓	2017
ManhattanSLAM [22]	✓	✓	-	✓	-	✓	2021
iMAP [23]	-	✓	-	-	✓	-	2021
FastFusion [24]	-	✓	-	-	-	✓	2022
HRBF-Fusion [25]	-	✓	-	-	✓	✓	2022
NICE-SLAM [26]	-	✓	-	-	✓	-	2022
Co-SLAM [27]	-	✓	-	-	✓	-	2023
PLRF-SLAM [28]	-	✓	-	-	✓	-	2024

TABLE 2. Overview of RGB-D-based Dynamic 3D Reconstruction Methods: This report discusses the details of different algorithms from the aspects of segmentation of dynamic objects, camera tracking, and model fusion, all of which are key technologies for static 3D reconstruction methods based on RGB-D cameras.

Method	Segmentation of Dynamic Objects		Camera Tracking		Model Fusion			Year
	Motion-based	DL-based	direct	indirect	TSDF	Surfel	Point-based	
Dynamicfusion [29]	✓	-	-	✓	✓	-	-	2015
VolumeDeform [30]	✓	-	-	✓	✓	-	-	2016
Killingfusion [31]	✓	-	-	✓	✓	-	-	2017
Co-fusion [32]	✓	✓	-	✓	-	✓	-	2017
Staticfusion [33]	✓	-	✓	-	-	✓	-	2018
EM-fusion [34]	-	✓	-	✓	✓	-	-	2019
DM-SLAM [35]	✓	✓	✓	-	-	-	✓	2020
RE-SLAM [36]	✓	-	✓	-	-	-	✓	2020
Flowfusion [37]	✓	-	✓	-	-	-	✓	2020
Rigidfusion [38]	✓	✓	-	✓	-	✓	-	2021
RTCB-SLAM [39]	-	✓	✓	-	✓	-	-	2022
RGB-D SLAM(PC) [40]	✓	-	✓	-	-	-	✓	2022
DPF-SLAM [41]	-	✓	-	✓	✓	-	-	2022
RGB-D SLAM(IPE) [42]	✓	-	✓	-	-	✓	-	2022
DSV-SLAM [43]	-	✓	✓	-	-	-	✓	2023

a 3D Gaussian-based scene representation. It retains the desirable properties of the continuum radiation field while avoiding unnecessary computations in space, and greatly improves its rendering speed while ensuring the rendering quality. In dynamic reconstruction, D-nerf [48] extended the application domain of NeRF from static to dynamic scenes by introducing the time dimension and learning canonical representations of dynamic scenes. Although this method has a high computational complexity, it excels in handling non-rigid motion and generating high-detail images, showcasing the potential of neural radiance fields in dynamic scene applications. Recursive-NeRF [49] introduced uncertainty prediction, recursively passing query points to different levels of neural networks based on complexity to achieve adaptive representation at the detail level, balancing efficiency and quality. Although this method improves computational efficiency, it requires storing multiple levels of neural networks, resulting in high memory consumption, especially when handling large-scale scenes. Reference [50] proposed a new method called NDR (Neural-Dynamic Reconstruction) for recovering high-fidelity geometry and motion from monocular RGB-D cameras in dynamic scenes. Although the method is effective, it has high computational complexity, long training times, and a large demand for computational resources.

III. RECONSTRUCTION OF STATIC SCENES

As shown in Figure 2, the process of static 3D reconstruction mainly includes depth image enhancement, camera tracking, model fusion, and surface extraction. In this chapter, we will use the basic process of static 3D reconstruction as a framework to introduce the improvements made by different

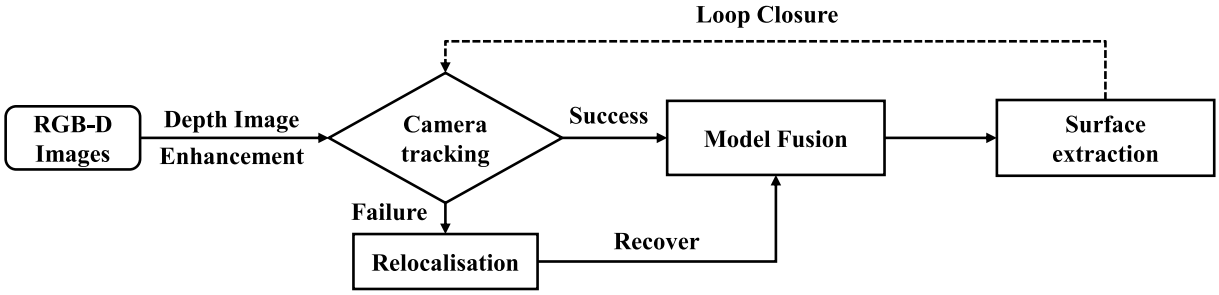

FIGURE 2. Overview of the static indoor reconstruction pipeline. The first step is to input RGB-D images and enhance the depth images. The second step is to use RGB-D data for camera tracking, which estimates the camera pose. If camera tracking fails, the camera relocation function is launched to recover from the failure. The third step involves incorporating surface information of the scene into the model using the tracked poses. The fourth step is to extract smooth and dense surfaces using surface extraction algorithms. Finally, camera pose is globally optimized through loop detection and processing.

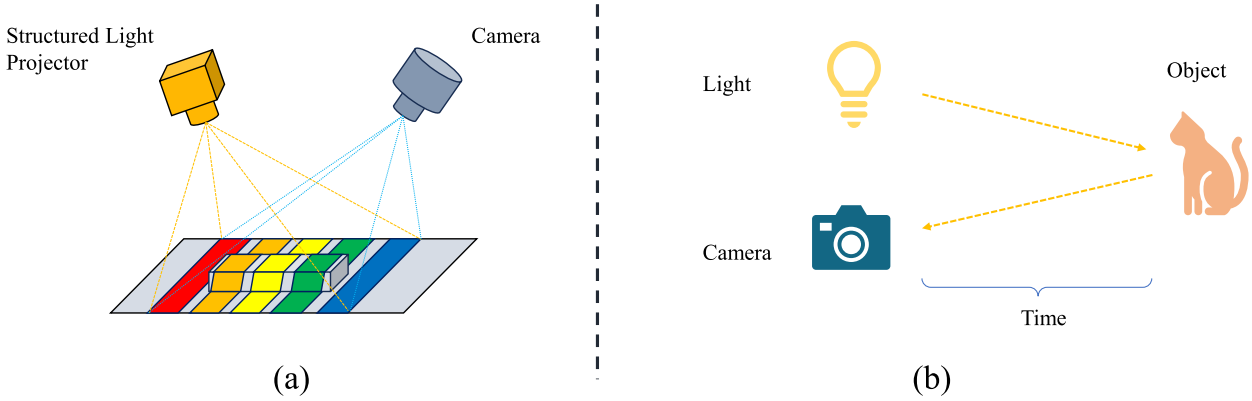

FIGURE 3. The Principles of Depth Cameras Based on Structured Light (a) and Time-of-Flight (b). Structured light-based approaches involve projecting patterned light onto an object to generate distinct phase information, which is then translated into depth data by a computational unit. Time-of-flight methods determine the distance from the camera to the target object by emitting light pulses and measuring the duration before their reflection is detected.

reconstruction algorithms in each step of the reconstruction process.

A. DEPTH IMAGE ENHANCEMENT

Currently, RGB-D cameras are mainly divided into two types: structured light and time-of-flight (TOF). As shown in Figure 3, Both types of RGB-D cameras can easily acquire color images and depth information. However, the depth images obtained are often marred by ''holes'' due to factors such as the material and structure of the object being measured, as well as rapid movements of the camera, resulting in data loss. This phenomenon is more common in consumer-level depth cameras. The goal of depth image enhancement is to denoise, refine, and enhance the depth measured initially.

Traditional methods such as median filtering, Gaussian smoothing, and bilateral filtering have been used to improve depth image quality. Reference [51] begins with median filtering in the 2D image space for noise reduction, followed by a two-step algorithm employing Gaussian smoothing on the 3D surface to enhance the depth of videos. Additionally, KinectFusion [6] utilizes bilateral filtering [52] to remove noise from original depth images, enhancing image quality. Since the RGB image obtained by the camera is often clear, [53], [54], [55] improve the accuracy of the depth image or supplement the missing parts of the depth image by corresponding the RGB map to the depth image. To achieve smooth object surfaces, [56] reconstructs locally smooth scene segments and deforms them for alignment, effectively addressing high-frequency noise and low-frequency distortion in depth images.With the advancement of super-resolution techniques, this technology has also been applied to enhance the resolution of depth images [57], [58], [59], thereby improving the precision of reconstructions.

In recent years, depth-enhancement algorithms based on deep learning have made great progress. These methods leverage the powerful learning capabilities of neural networks to predict surface normals and occlusion boundaries from RGB maps, which are then merged with depth maps captured by depth cameras to complete the missing parts in the original depth maps. A depth network [60] was developed to forecast object surface normals and occlusion boundaries from RGB maps. This predicted data was subsequently merged with depth maps captured by a depth camera, completing the missing parts in the original depth maps. Moreover, [61] proposed a cascaded CNN structure (DDRNet) to enhance

low and high frequency information in deep data. Supervised deep learning approaches often necessitate ground truth data from actual scenes, a requirement that poses significant acquisition challenges. Training networks with synthetic data presents a potential solution. However, the domain transfer issue between synthetic and real data may impair performance. References [62] introduced three methods for unsupervised domain adaptation of a depth denoising network, transitioning from synthetic to real-world data. Addressing the challenge of acquiring real datasets, researchers have turned to unsupervised [63], [64] and self-supervised [65], [66], [67] learning techniques to directly denoise depth maps in the absence of ground truth.

B. CAMERA TRACKING

In a static scene, the scene can be scanned by moving the camera to obtain RGB-D data. For each frame of image captured by the camera, the camera trajectory and poses need to be tracked to fuse the RGB-D data into the model. Nevertheless, the accuracy of this process can be compromised by factors including the precision of algorithms, occlusions, and the velocity of camera motion, necessitating subsequent optimization of the pose estimation.

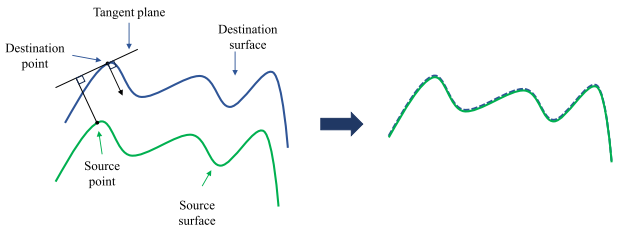

FIGURE 4. The point-to-plane ICP algorithm. It optimizes the camera's pose by minimizing the distance between the source point and the tangent plane of the corresponding destination point.

1) POSE ESTIMATION

a: ICP-BASED

To accurately estimate the camera pose between different frames, [68] introduced the Iterative Closest Point (ICP) algorithm. The ICP algorithm is a classical point cloud registration technique that iteratively aligns two or more point clouds to minimize the error between them. Assume there are two sets of points: the target point set $\mathbf{Q} = \{q_1, q_2, \ldots, q_n\}$ and the source point set $\mathbf{P} = \{p_1, p_2, \ldots, p_m\}$. The goal of the ICP algorithm is to find a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ to minimize the following mean squared error:

$$E(R, t) = \sum_{i=1}^{m} \|Rp_i + t - q_{\text{match}(i)}\|^2 \tag{1}$$

In this context, $q_{\text{match}(i)}$ represents the nearest neighbor of p_i, that is, the point in $\mathbf{Q}$ closest to p_i. Considering the heavy dependency of the point-to-point Iterative Closest Point (ICP) algorithm on initial values, suboptimal starting points may lead to an increase in the number of iterations or inaccuracies in the results. Therefore, [69] introduces a point-to-plane ICP algorithm (Figure 4), which improves camera positioning by minimizing the sum of squared distances between each source point and the tangent plane of its corresponding target point, thereby accelerating convergence:

$$E(R, t) = \sum_{i=1}^{m} \left(n_{\text{match}(i)}^T (Rp_i + t - q_{\text{match}(i)})\right)^2 \tag{2}$$

$n_{\text{match}(i)}$ is the normal vector corresponding to $q_{\text{match}(i)}$. Building upon these advancements, [70] integrated point-to-point ICP and point-to-plane ICP into a single probabilistic framework to form a new algorithm called GICP, which exhibits more robustness against incorrect matches. Due to the cumulative error generated by frame-to-frame matching, the camera trajectory can experience drift, severely affecting the accuracy of pose estimation. KinectFusion [6] used a frame-to-model matching method, greatly reducing cumulative error. Subsequently, [7], [9], [71], [72], [73], [74], and [75] added dense photometric validation based on ICP geometric registration to further optimize the matching algorithm. When matching two sets of point clouds, in order to consider the local features of point clouds (normal vectors, curvature), [76] defined an error function during the iterative solving process that not only included the projected distance of normal vectors between point clouds, but also the direction error of normal vectors, making pose estimation more robust. In addition, there are some other ICP variants, such as efficient ICP [77], non-rigid ICP [78], etc.

b: FEATURE-BASED

The ICP algorithm performs well under the assumption of minor motion between frames. However, it tends to converge to local optima during rapid camera movements, where the difference between successive frames is significant. In order to cope with this situation, [79], [80], [81], [82], [83], [84] extracted feature points (SIFT, SURF, ORB) from color images and use these sparse features to quickly match the pose of each frame. ORB-SLAM [82] is a classic feature-based visual SLAM system. It combines FAST feature detection and BRIEF feature description, ensuring the speed of feature point detection and the stability of description. However, it is limited to feature matching between image frames and is unstable under varying lighting conditions and when feature points are missing. ORB-SLAM2 [83] improves pose estimation by supporting stereo and RGB-D sensors, utilizing depth information for further optimization. Building on this, ORB-SLAM3 [84] further improves the pose estimation algorithm, supports the construction and management of multiple maps, and enhances pose estimation accuracy through the collaborative work of multiple sensors. In addition to using point features for matching, useful edge features [85], [86] can also be extracted from depth images to establish corresponding constraints, thereby enhancing the robustness of pose estimation. To ensure the real-time and accuracy of pose tracking, [87] introduced an information-theoretic approach for point selection in RGB-D direct odometry measurements.

This approach simplifies the optimization process while maintaining accuracy by identifying and utilizing data points that carry the most information. CPA-SLAM [89] models the environment using a global model composed of planes, which reduces significant image drift. Reference [88] extracts 3D facial landmarks during face reconstruction for model fine-tuning to ensure the accuracy of head pose estimation. To achieve high-precision feature tracking under rapid sensor motion, [24] performed feature tracking within an extended Kalman filter framework. This framework integrates IMU data to better accomplish sensor motion estimation.

c: HYBRID METHOD

Pose matching algorithms based on ICP usually require aligning the entire point cloud data, resulting in high computational costs, which are not suitable for real-time 3D modeling of large scenes. In contrast, feature-based matching algorithms can better cope with the limitations of real-time requirements. However, feature-based matching algorithms often require dense features, and the reconstruction quality is significantly affected when the number of matching features in the scene decreases [90]. Combining sparse feature matching with ICP is a good method to balance real-time performance and reconstruction quality. BundleFusion [11] first utilizes sparse SIFT features for coarse pose alignment, and then refines the estimated pose using dense photometric and geometric terms similar to the ICP algorithm, achieving real-time accurate pose estimation and solving the real-time issue of high-quality reconstruction. References [91] and [92] combined edge information with the ICP algorithm to enhance robustness and accuracy. Recently, [93] introduced an enhanced 3D scene reconstruction method using Fast Point Feature Histograms (FPFH) and Iterative Closest Point (ICP) techniques. It improves model robustness and accuracy by modifying the weight calculation formula and employing an enhanced FPFH descriptor for initial registration estimation. To further increase the ICP iteration speed, it also utilizes a Best Bin First (BBF) strategy to reduce data dimensionality.

d: NERF-BASED

The advent of NeRF offers a new paradigm for camera pose optimization. These methods leverage the power of neural networks to synthesize novel views and provide accurate pose estimates, significantly enhancing the robustness and accuracy of camera tracking in static scenes. iNeRF [94] estimates the camera pose by inverting NeRF. Specifically, NeRF optimizes the parameters Θ of the scene using a given set of camera poses T and observed images I, while iNeRF inversely solves the problem of recovering the camera pose T given the weights Θ and image I as inputs:

$$\hat{T} = \underset{T \in SE(3)}{\arg\min}\, \mathcal{L}(T \mid I, \Theta) \tag{3}$$

To solve this optimization problem, iNeRF obtains some estimated camera poses $T \in SE(3)$ in the coordinate system of the NeRF model and renders the corresponding image observations. To update the pose T, the same photometric loss function L used in NeRF is employed:

$$\mathcal{L} = \sum_{r \in R} \|\hat{C}(r) - C(r)\|_2^2 \tag{4}$$

Here, $r \in R$ represents a set of sampled rays, and $C(r)$ is the observed RGB value of the pixel corresponding to ray r in an image. Although iNeRF successfully applied NeRF to pose estimation and achieved excellent results, it still requires an initial pose as a starting point, which affects the convergence of the optimization and the final accuracy. With the development of deep learning, considering the exceptional performance of Generative Adversarial Networks (GANs) in image generation, [95] combines GANs with NeRF to optimize initial pose estimations during reconstruction. It does not rely on known camera poses and can optimize from a completely random initialization, which is particularly useful in uncertain and complex scenes. Additionally, [96] employs a coarse-to-fine camera registration strategy and demonstrates the impact of positional encoding on alignment, effectively optimizing the neural network's scene representation while addressing pose misalignment issues in large-scale cameras. To further address errors arising from drastic camera movements, [97] introduced an undistorted monocular depth prior based on NeRF and proposes novel loss functions to constrain the relative poses between adjacent frames.

Bundle Adjustment is a technique used to optimize camera parameters and 3D point coordinates in 3D reconstruction. Its main purpose is to improve the accuracy of 3D reconstruction by minimizing the reprojection error. Specifically, the optimization problem can be represented as:

$$\min_{P,X} \sum_{i,j} \left\| x_{ij} - \pi(P_i, X_j) \right\|^2 \tag{5}$$

where P_i represents the parameters of the i-th camera, X_j represents the coordinates of the j-th 3D point, x_{ij} is the observed 2D coordinate of the j-th 3D point in the i-th camera, and $\pi(P_i, X_j)$ is the projection of the 3D point X_j onto the 2D image plane through the camera parameters P_i.

Inspired by the Bundle Adjustment (BA) algorithm, the NBA (Neural Bundle Adjustment) method proposed by [98] optimizes the implicit surface and camera poses without relying on known camera extrinsics. Specifically, NBA updates the 3D point X at each step as follows:

$$X \leftarrow X - \phi(X)\nabla\phi(X) \tag{6}$$

where $\phi(X)$ is the distance value output by the SDF (Signed Distance Field) network in the neural radiance field, and $\nabla\phi(X)$ is the gradient at that point. After updating the 3D point X, the reprojection error is calculated based on the feature trajectory T, jointly optimizing the SDF network ϕ, the estimated camera poses P, and the updated 3D point set X.

2) LOOP CLOSURE

During the pose matching of each frame, both ICP-based and feature-based pose matching algorithms can produce errors, and these errors increase with the number of frames. After the camera completes a full rotation around the scene, accumulated errors can cause a misalignment between the starting and ending points. Therefore, we must process loop closure after pose matching to ensure global consistency and reduce the impact of accumulated errors on reconstruction quality. Reference [99] defined keyframes and registers the frames between them to eliminate local errors, and uses the entropy ratio criterion to check loop closure. Reference [71] utilized efficient Pose graph optimization and Sparse bundle adjustment for global consistency alignment. However, this global optimization distributes the residual error over the entire path, resulting in the destruction of the details of the object surface. To further optimize pose estimation, [9] divides all frames into equally sized blocks with one overlapping frame between adjacent blocks. Each small block is reconstructed first, then the overlap frames are used to register the blocks and detect loop closure, and finally erroneous loops are removed to achieve global consistency in reconstruction. Subsequently, BundleFusion [11] performed sparse-to-dense global pose optimization and solves loop closure by integrating and re-integrating previous RGB-D frames during movement, enabling it to correct all drifts. Although this method produces better positional optimization, it requires a large amount of computational resources. To enhance pose estimation accuracy in large-scale scenes, [100] introduced a two-pass loop closure detection method that integrates global and local image features to identify loop closure candidates. Recently, [24] utilized subgraph-based depth image encoding and 3D graph deformation for loop closure to maintain global consistency in the reconstructed model. Reference [101] introduced local 3D deep descriptors (L3Ds) for loop closure handling. L3Ds are compact representations of patches extracted from point clouds, learned using deep learning algorithms, significantly enhancing loop closure detection accuracy.

Another way to address cumulative errors is to assume the scene structure in the world frame and directly align each tracking frame with the scene structure, rather than with keyframes or the last frame. One of the most common assumptions is the Manhattan assumption [102], [103], which represents the scene using a set of orthogonal planes aligned with the world's three main axes, simplifying the scene understanding and enabling efficient inference of scene geometry and object position. Structure-SLAM [104] employed a convolutional neural network(CNN) to forecast normals and compute drift-free rotations leveraging geometric features under the Manhattan assumption, effectively addressing low-texture regions in indoor settings. Building upon Structure-SLAM, [105] incorporated planar features within the Manhattan framework and introduced an advanced meshing module for reconstructing scene structures, thereby enhancing localization and mapping accuracy.To make the Manhattan assumption more suitable for real-world scenes, ManhattanSLAM [22] directly detected Manhattan frames(MFs) from planes and modeled the scene as a Mixture of Manhattan Frames (MMF), estimating unbiased rotation by observing MFs across frames.

3) RELOCALIZATION

Due to factors such as high camera movement speed or changes in viewpoint, camera tracking may fail. Therefore, the ability to quickly recover and perform relocalization when camera tracking fails is essential in the 3D reconstruction process. There are several methods for camera relocalization, including the following:

a: KEYFRAME-BASED

This method requires defining and storing keyframes. When the camera tracking fails, it needs to query the images and estimate the camera pose by measuring the overall image similarity with a known set of keyframes. Reference [106] explored an effective keyframe-based relocation method. In the stage of determining keyframes, besides the threshold based on the distance in space, a similarity discrimination with previous keyframes was added to avoid collecting redundant information. In order to quickly retrieve candidate poses in case of tracking loss, this method uses an efficient frame encoding based on ferns. Keyframe-based methods can perform camera relocation in real-time, but they rely on matching input images with a keyframe database and cannot re-locate in a new pose.

b: KEYPOINT-BASED

This relocation method mainly utilizes the sparsity of feature points. During successful tracking, feature points are detected in the image, and their corresponding descriptors and positions in the world coordinate system are stored in a database. When camera tracking is lost, the current frame's key points and descriptors are calculated, and a match is performed against the database. After a successful match, the current image's pose can be obtained to complete camera relocation [107], [108], [109], [110], [111], [112], [113].The challenges of this method include: (1) the choice of feature point and descriptor calculation method, (2) how to store key points and their corresponding descriptors, and (3) how to perform feature matching between frames. Inspired by the idea of visual Bag-of-words, [108] stored the extracted SIFT feature descriptors into a vocabulary during successful tracking, and utilized the Term Frequency-Inverse Document Frequency (TF-IDF) of the visual words in each node to rank the nodes. When tracking is lost, refined relocation poses are obtained by matching the descriptor set in each node with the descriptors extracted from the query image to recover from tracking failure. Reference [110] proposed using a regression forest to directly predict the 3D correspondence of all pixels in the current image to the scene. Compared with traditional keypoint-based methods, this method does

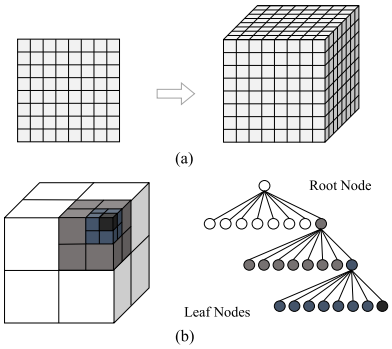

FIGURE 5. (a) is a schematic diagram of 2D pixels and 3D voxels, and (b) is a schematic diagram of the octree structure.

not require explicit detection, description, or matching of key points, making it simpler and faster. However, it must train the regression forest in advance on the scene of interest offline, and cannot achieve real-time camera relocation. Reference [112] overcame the limitation of having to train offline by dynamically adapting pre-trained forests to new scenarios.

c: HYBRID METHOD

Researchers have integrated keyframes and keypoints to enhance relocation accuracy while maintaining real-time performance. Upon tracking failure, [82] adopted the DBoW2 algorithm [114] to identify matching candidate keyframes, subsequently calculating ORB features within these keyframes and employing the PnP algorithm [115] to alternately estimate the current frame's pose. Reference [86] merged edge features with the keyframe-based method [106], securing robust loop closure and relocalization capabilities.

C. MODEL FUSION

The pose matching algorithm calculates the initial pose, which is further enhanced by closed-loop processing. After that, the surfaces of the scene need to be fused into the 3D model according to the camera's position. Currently, there are mainly two types of surface fusion models used: voxel-based and surfel-based.

1) VOXEL-BASED

As shown in the figure 5(a), an image can be represented by square pixels in 2D space, and extending the pixels to 3D is a voxel. This can intuitively reflect the shape of an object. Reference [116] was the first to propose using the TSDF (Truncated Signed Distance Function) grid model to fuse depth information on the basis of voxel representation. KinectFusion [6] further applied this model to 3D reconstruction using RGB-D cameras. This method requires fixing the size of the scene before reconstruction, making it difficult to scale the scene. For large-scale scene reconstruction, which requires substantial memory, KinectFusion falls short. Therefore, various scholars have extended the original TSDF voxel model:

a: MOVING VOLUME

To overcome the limitations of voxel representation, [7], [72], [117] expanded the reconstruction area to infinite space by moving voxels. Thomas Whelan [7] utilized a cyclic buffer data structure to effectively recycle GPU memory, addressing the issue of insufficient memory for large-scale scene reconstruction with voxel models. The algorithm enabled camera translation and rotation in the real world, incrementally enlarging the reconstructed surface. Reference [117] proposed the Moving Volume KinectFusion method, which establishes a TSDF buffer and a swap buffer. Utilizing a double buffering mechanism to map between volumetric models during camera movement, the method allows for online processing of volume rotations and translations through voxel interpolation.

b: OCTREE-BASED

The geometry of most objects is very sparse with respect to the whole scene body, which means that the voxels in the TSDF model are mostly empty and all this storage space is wasted. The octree structure is a data model first proposed by Dr. Hunter [118] in 1978. As shown in 6(b), this structure can effectively utilize memory by dividing the scene space, thereby improving storage efficiency. Although its definition is simple, it is difficult to maintain the parallelism of the GPU due to the sparsity of its nodes. Reference [20] and [119], designed a novel octree data structure to improve the reconstruction update and surface prediction parts of KinectFusion, which can fully utilize the parallelism of the GPU, greatly improving storage efficiency and further expanding the reconstruction scale. To reduce memory consumption, [99] fused the acquired depth and color information into a multiscale octree representation of a signed distance function, which can maintain low memory usage while achieving high accuracy. To further improve storage efficiency, [120] defined an octree data structure that supports volume multiresolution 3D mapping and mesh partitioning, reducing memory consumption by only allocating units close to the surface.

c: VOXEL HASHING

Although the octree structure can improve the storage efficiency of the model to some extent, complex octree structures still have additional computational complexity and pointer overhead. A simple spatial hashing scheme is used in [8] to compress space, which allows data to flow efficiently in and out of a hash table, enabling real-time access and updates of surface data in the scene without the need for complex hierarchical data structures. Voxel hashing has been widely used in real-time 3D reconstruction [21], [121], [122].

d: DEEP LEARNING-BASED

The ability of neural networks to learn rich prior knowledge provides new directions for the development of scene representation. When exploring cluttered indoor scenes with

an RGB-D camera, [123] initialized the truncated signed distance function (TSDF) reconstruction of each object with compact instance segmentation using Mask-RCNN, which resulted in a resolution related to object size and novel 3D foreground masks. Reference [124]reconstructed real-time scenes with both geometry and semantic information by incorporating semantic predictions from neural networks into the voxel-based model built on Voxel hashing.

2) SURFEL-BASED

Voxel-based methods are expensive for handling loop closures in real-time 3D reconstruction because precise compensation may involve changing the entire volume. Moreover, the size of the voxel volume is typically fixed in practice, which limits the adaptivity of representation. If an object is relatively small or thin compared to the voxel size, it can seriously affect the reconstruction quality.

In surfel-based scenes (Figure 6). This representation has the following advantages: (1) Flexibility. When performing point fusion updates, the data is updated using weighted fusion, where the radius of the surface patch is related to the distance between the camera center and the scene surface. The farther the distance, the larger the radius of the surface patch, and this updating method can effectively reconstruct the entire surface. (2) High adaptability. It can measure more densely distributed points at high resolution. (3) It can easily handle thin objects.

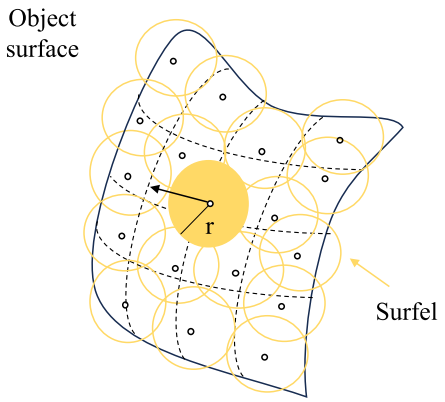

FIGURE 6. Representation of object surface by Surfels. The position information, radius of the surface patch, normal vector, color information, and time information of each point are stored.

Reference [125] first introduced the concept of surfels and provide a detailed description of surface scenes. ElasticFusion [10] utilized this representation for real-time dense scene reconstruction. The system continuously optimizes the reconstructed map to improve the accuracy of reconstruction and pose estimation, and employs Random Ferns to detect loop closures for global consistency. With the development of deep learning, in order to fully utilize the semantic information of the scene, Semanticfusion [127] combined CNN and ElasticFusion to successfully fuse semantic predictions from multiple viewpoints into a surfel-based representation.

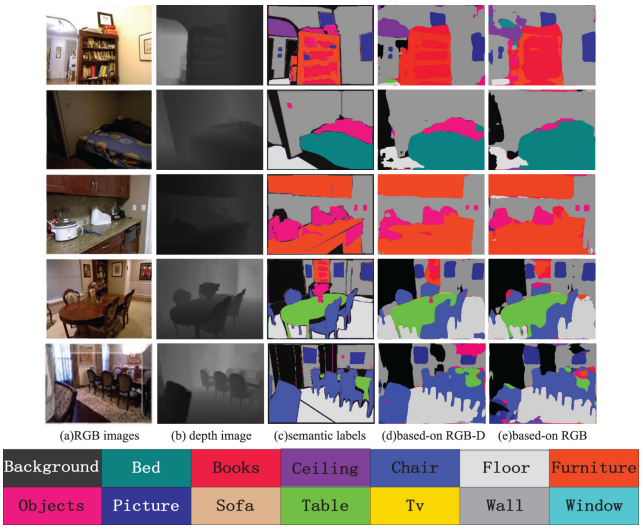

FIGURE 7. Indoor semantic segmentation results. Figure taken from [126].

Reference [126] proposed an indoor RGB-D image semantic segmentation network with multi-scale feature fusion based on ElasticFusion. It integrates the visual color features and depth geometric features of RGB-D images, improving the accuracy of image semantic segmentation. The segmentation results are shown in Figure 7. DeepSurfels [128] integrated feature information learned from RGB images with detailed representations of facets, making it possible to reconstruct large-scale scenes in real-time. To obtain high-quality surface texture, [129] employed Shape-from-Shading (SfS) and spatially-varying spherical harmonics (SVSH) techniques to simultaneously optimize geometry, texture, and camera poses. The main drawback of the surfel representation is its discreteness, which can be addressed by the meshing approach. Reference [130] created a triangle mesh and performs real-time mesh reconstruction from RGB-D video, which works well for reconstructing thin objects. However, this method requires camera poses as additional input. In contrast, [25] utilized Hermite Radial Basis Functions (HRBF) implicits for direct camera tracking and RGB-D reconstruction, which is a dynamic surface representation that effectively reduces the influence of noise and reconstructs using surface photometric constraints.

3) NERF-BASED

Explicit representations like voxel and surfel allow real-time scene reconstruction, but they face challenges in mapping accuracy and balancing memory consumption. Moreover, they lack in novel view synthesis capabilities. In recent years, with the introduction of NeRF, implicit representations have overcome limitations associated with explicit representations, generating high-fidelity reconstructions with reduced memory usage. These implicit representations achieve this by continuously querying scene properties to generate high-quality images from novel viewpoints. IMAP [23] demonstrated for the first time that Multi-Layer Perceptrons (MLPs) can serve as the sole scene representation in real-time

SLAM systems using handheld RGB-D cameras, utilizing keyframe structures and multi-processing computation flows. Reference [131] utilized the point cloud provided by COLMAP and reprojection errors to enforce depth constraints in NeRF, effectively enhancing the rendering speed and reconstruction quality of NeRF. To reduce computational costs and enhance scalability, NICE-SLAM [26] applied the hierarchical scene representation concept to NeRF. However, due to the local updates performed by NICE-SLAM's feature grid, it fails to achieve reasonable hole filling. Co-SLAM [27] combined coordinate and sparse parameter encoding for scene representation and employed dense global bundle adjustment using rays sampled from all keyframes. Simultaneously, [132] proposed a NeRF-based mapping approach using a hierarchical hybrid representation, leveraging implicit multiresolution hash encoding and explicit octree Signed Distance Function (SDF) priors to describe scenes at different detail levels, achieving real-time high-fidelity dense mapping and dynamic expansion capabilities. Since NeRF does not reconstruct actual surfaces and pseudo-shadows occur when using Marching Cubes to extract voxel-based surfaces, [45] used Truncated Signed Distance Functions (TSDF) to represent surfaces, extending them to commodity RGB-D sensors to reconstruct high-quality 3D scenes. Recently, NGEL-SLAM [133] employed a sparse octree grid integrated with implicit neural maps, ensuring memory efficiency and precise environmental depiction.

D. SURFACE EXTRACTION

Once the surface information of a scene has been fused into a model based on the camera pose, a surface extraction algorithm is required to obtain a visual representation of the surface. Depending on how the reconstructed scene is represented and stored, surface extraction algorithms can be classified into raycasting and marching Cubes.

1) RAYCASTING

The surface extraction method proposed by [134] based on Raycasting primarily involves using rays emitted from the camera center and passing through pixels to project onto the object surface to find the iso-surface.

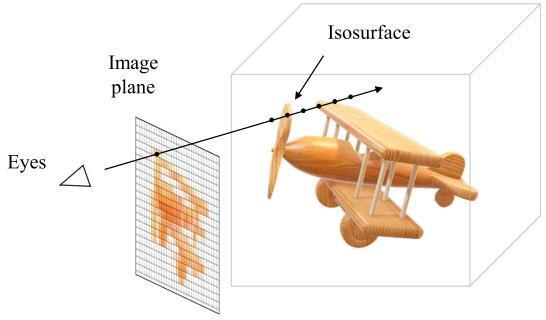

FIGURE 8. Diagram of Raycasting. It detects and calculates intersections with objects in the scene by casting rays, thereby extracting surface information.

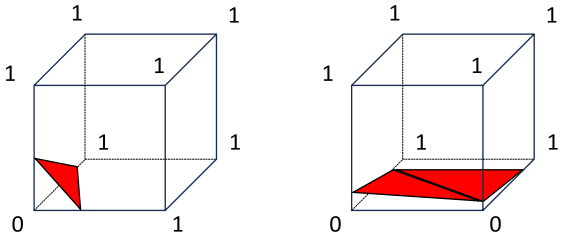

FIGURE 9. Diagram of Marching Cubes.

The basic process of this method is as follows (Figure 8): firstly, a ray is projected along the viewing direction from each pixel on the image plane, which passes through the surface of the object. Then, sampling is performed at a certain step size, and linear interpolation algorithms are used to find the intersection point with the surface. This essentially means checking the value of the truncated signed distance function at each voxel along the ray until the first zero-crossing is found. This algorithm is widely used for surface extraction in voxel models [6], [7], [8], [11], [117], [129].

2) MARCHING CUBES

The marching cubes algorithm was initially proposed by Lorensen [135], who divided the three-dimensional geometry into small cubes called voxels and defined the voxels using scalar values at the eight corners of each cube. As shown in the Figure 9, if the data value at a vertex of the cube is greater than or equal to the value of the surface we are constructing, the vertex is assigned a value of 1, and 0 otherwise. Under this assumption, when the surface intersects with a cube, the intersection points between the isosurface and the edges of the cube are calculated using interpolation, and then the intersection points of each edge are connected in a certain way to represent the isosurface inside the cube. After finding the isosurface passing through this cube, move to the next cube to continue searching for the isosurface. This is the process of extracting the surface using the marching cubes algorithm.

Due to the huge amount of storage required to reconstruct high-quality models, the octree storage method has been applied to the reconstruction models to get rid of the limitations of commercial computer memory. However, extracting the reconstructed surface from the octree representation is more complicated than extracting it from a regular voxel grid. Reference [136] and [137] extended the Marching Cubes algorithm by addressing the inconsistencies that arise when adjacent leaf nodes in the octree have different depths. Reference [138] proposed a method of marking edges with Hermite data to generate signed grid contours, and extended this method to octrees. By aligning the vertices of the dual grid with the characteristics of the implicit function, [139] can extract iso-surfaces that capture small, thin, and even sharp features in the surface without excessively refining the octree. Reference [140] introduced the concept of an edge tree to provide a method for directly extracting watertight mesh without restricting the octree topology or modifying vertex values.

With the application of deep learning and NeRF technology in 3D reconstruction, [141] proposed a data-driven method called Neural Marching Cubes (NMC) for extracting triangular meshes from discrete implicit fields. This method addresses the shortcomings of traditional surface extraction methods in recovering geometric features such as sharp edges and smooth curves. Specifically, NMC redesigns the mesh subdivision template and introduces neural networks to learn vertex positions and mesh topology, thereby better preserving geometric features. Recently, [142] proposed another method called NeuralMeshing. This method generates meshes iteratively, making it suitable for shapes of various scales and capable of adapting to local curvature, thereby significantly improving the quality of surface extraction.

In conclusion, the integration of deep learning into static 3D reconstruction has brought significant advancements, providing robust and accurate solutions for depth image enhancement, camera tracking, model fusion, and surface extraction. These methods leverage the powerful learning capabilities of neural networks to improve the quality and efficiency of 3D reconstructions, offering new perspectives and directions for future research in this field.

IV. RECONSTRUCTION OF DYNAMIC SCENES

Dynamic scenes consist of both dynamic objects and static backgrounds. Figure 10 illustrates the general process of dynamic reconstruction. In the following chapter, we will discuss recent developments in dynamic 3D reconstruction, focusing on three aspects: segmentation of dynamic objects, Camera tracking, and model fusion.

A. SEGMENTATION OF DYNAMIC OBJECTS

In contrast to static 3D reconstruction, dynamic scenes contain freely moving objects that significantly affect camera pose estimation. Moreover, entities such as human beings and animals undergo non-rigid deformations while in motion. To handle the reconstruction of these dynamic objects, the first step is to distinguish between dynamic and static features, a process known as motion segmentation. Various approaches, including motion analysis-based methods and deep learning-based methods, are employed for motion segmentation in the scene to identify the dynamic characteristics.

1) MOTION ANALYSIS-BASED METHODS

Methods based on motion analysis separate dynamic objects from the static background by detecting object movement within the scene. Examples of such methods include geometric methods, optical flow methods, etc. DynamicFusion [29] utilized geometric features to separate dynamic objects and defined a canonical model specifically for reconstructing non-rigidly deforming dynamic objects. The canonical model was transformed to the live frame using voxel deformation fields. This method addresses the deformation issues of dynamic objects during motion, enhancing the robustness and accuracy of reconstruction. Similar to this approach, Nerfies [143] enhanced NeRF by optimizing an additional continuous volume deformation field, which warps each observed point into a canonical 5D NeRF representation. D-NeRF [48] incorporated time as an additional input to the system and divides the learning process into two main stages: one stage encodes the scene into canonical space, and another stage maps the canonical representation into a deformed scene specific to a particular time. VolumeDeform [30] combined SIFT features extracted from RGB images with depth maps for motion tracking, enhancing the robustness of matching point recognition. References [33] and [144] applied K-means clustering to perform visual clustering and assigned static weights to each clustered pixel or point. Reference [145] estimated static weights based on the distances between corresponding point and line features and applied filtering to the data related to dynamic targets using these static weights, achieving precise localization and tracking of the targets. Reference [146] constructed a foreground model based on the mutual motion between two frames and combined it with RGB-D frame information to segment dynamic and static feature points. Reference [36] initially employed a simple and efficient clustering algorithm to group spatially and appearance-related pixels of each keyframe into different regions, then identified Candidate Dynamic Keypoints (CDK) in consecutive frames with large reprojection errors and recognized regions with a high CDK ratio as dynamic regions. Reference [147] observed that regardless of camera movement, the triangles formed by any three fixed points on a static object remain fixed, and these triangles formed by the three points in different camera coordinate systems are similar. Therefore, the authors determined whether a feature point is static or dynamic by comparing the similarity of the triangles formed by three sets of feature points in two keyframes. Reference [148] introduced a grid-based feature extraction approach that enables fast and efficient extraction of high-quality FAST feature points. Additionally, it combined inertial measurement units for motion prediction, achieving feature tracking and motion consistency detection.

2) DEEP LEARNING-BASED METHODS

Unlike traditional methods based on motion analysis, deep learning-based methods can learn semantic information as priors from training datasets, and the extraction of semantic information through various image processing techniques has different impacts on dynamic scene problems. Currently, many methods use semantics to make motion segmentation more robust [35], [39], [43], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158]. These methods employ deep neural networks for semantic segmentation and object recognition on RGB-D images to achieve dynamic object detection and tracking.

Mask-RCNN [39] is an instance-level segmentation algorithm based on images that can provide prior information on dynamic objects in a scene. As shown in Figure 11(a),

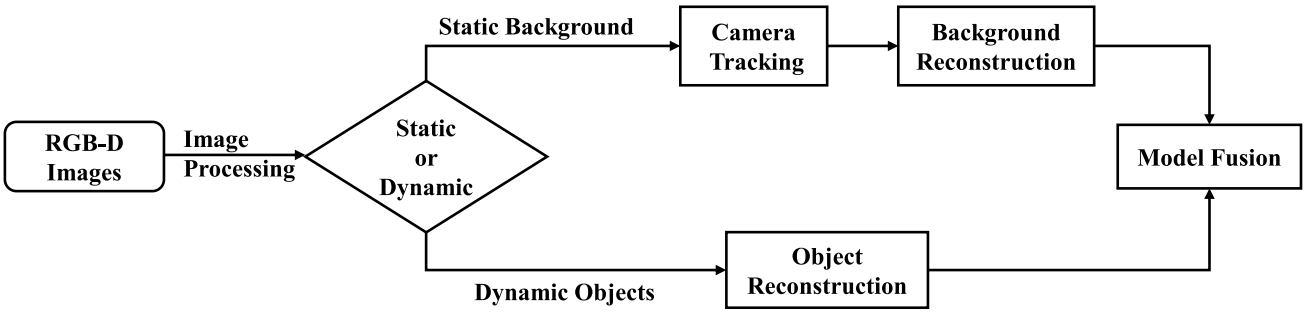

FIGURE 10. Overview of the dynamic indoor reconstruction pipeline. The first step is data acquisition, which is the same as in static 3D reconstruction. The second step is data preprocessing, which involves not only denoising the raw image data but also separating the dynamic objects from the scene. Then, camera tracking is performed using the static background information to align the current frame data with the previous frame or model, finding the correspondences between them and reconstructing the static background. Meanwhile, the dynamic objects are reconstructed separately. Finally, the dynamic objects and static background are merged to complete the reconstruction of the entire scene.

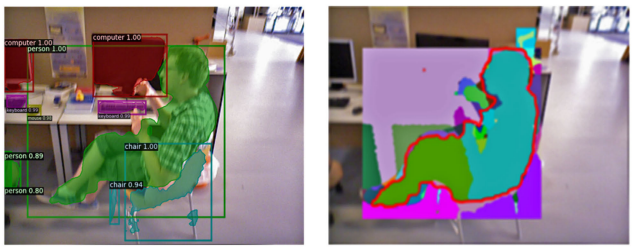

FIGURE 11. Segmentation of dynamic regions. (a) is the result of Mask-RCNN. (b) is the result optimized using the connected component analysis method. Figure taken from [39].

it can provide bounding boxes for dynamic objects. However, within the bounding boxes, some static background areas are classified as dynamic foreground areas, and some dynamic foreground objects are classified as static background. In RGB-D data, utilizing the depth difference between dynamic regions and static background can better optimize segmentation results. Additionally, compared to smooth areas within objects, the normal and variance differences at the boundary between dynamic regions and their adjacent static background are also greater. Therefore, [39] uses connected component analysis to optimize the segmentation results. Specifically, the dynamic weight of a pixel is given by the following formula:

$$\mathcal{O} = \mathcal{O}_d + \mathcal{O}_o + \gamma_1 \mathcal{O}_e \tag{7}$$

where $\mathcal{O}_d$ is the depth difference, $\mathcal{O}_\sigma$ is the variance, γ_1 is the weight for the normal difference $\mathcal{O}_e$, and is obtained by the following formula:

$$\mathcal{O}_d = \max_{i \in N} |(\mathbf{v}_i - \mathbf{v}) \cdot \mathbf{n}| \tag{8}$$

$$\mathcal{O}_e = \max_{i \in N} \begin{cases} 0, & \text{if}((\mathbf{v}_i - \mathbf{v}) \cdot \mathbf{n}) < 0 \\ 1 - (\mathbf{n}_i \cdot \mathbf{n}), & \text{else} \end{cases} \tag{9}$$

$$\mathcal{O}_\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{v}_i - \mathbf{v})^2} \tag{10}$$

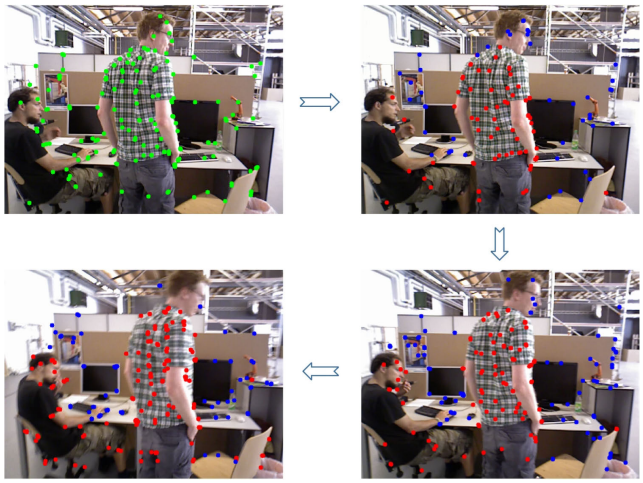

FIGURE 12. The propagation of dynamic probabilities during tracking, where green points indicate feature points with initial dynamic probabilities, blue points represent identified static feature points, and red points represent dynamic feature points. Figure taken from [154].

where $\mathbf{v}$ represents the point on the depth map, $\mathbf{n}$ is the normal of that point, N denotes the set of neighborhood point indices for $\mathbf{v}$, and $\mathbf{v}_i$ represents the neighboring points of $\mathbf{v}$. Figure 11(b) shows the results optimized using the connected component method based on the value of $\mathcal{O}$.

Additionally, PSPNet-SLAM [151] and DDL-SLAM [152] use PSP-Net and DU-Net respectively as deep learning (DL) models for segmenting dynamic scenes and static backgrounds. However, these segmentation methods using DL models require high memory consumption and computational cost. To improve the real-time performance of the reconstruction system, LRD-SLAM [153] proposed a fast deep convolutional neural network (FNet) for semantic segmentation, which can quickly and accurately identify pedestrian information in a given scene.

Moreover, [154] found that most classic semantic SLAM methods generate semantic results for each frame individually, such as DynaSLAM [155], DS-SLAM [156], and DM-SLAM [35], leading to redundant operations. Since the input to visual SLAM is a sequence of continuous

frames, the segmentation results of consecutive frames have many similarities, making it unnecessary to segment each frame. Reference [154] segments only the keyframes and propagates the segmentation results of the keyframes to their adjacent frames, significantly avoiding the time delay caused by segmenting each frame while ensuring segmentation accuracy. The experimental results are shown in Figure 12. Recently, [43] and [157] employed YOLO v5 for detecting dynamic objects in the scene, further enhancing segmentation accuracy. DDN-SLAM [158] leveraged deep semantic system priors and conditional probability fields for effective segmentation. Through the creation of depth-guided static masks and the use of joint multi-resolution hashing encoding, it achieves rapid hole filling and superior mapping quality, effectively reducing the impact of dynamic information.

B. CAMERA TRACKING

In static scene reconstruction, for slightly moving dynamic objects, we treat them as static and ignore their motion during pose estimation by utilizing rigid alignment. However, due to significant variations in the motion of dynamic objects in dynamic scenes, directly applying static algorithms for camera pose tracking would lead to failure. Therefore, the handling of dynamic objects is crucial for solving the problem of dynamic 3D reconstruction. One approach considers dynamic objects as outliers and directly removes them during the reconstruction process, focusing on pose estimation for the static scene. This method is referred to as the direct approach. Another approach involves utilizing the features of dynamic objects while simultaneously reconstructing both static and dynamic objects in the scene. This approach is known as the indirect approach. In this chapter, we discuss how to handle dynamic objects from these two perspectives in order to achieve accurate pose estimation in dynamic scenes.

1) DIRECT APPROACH

In a real indoor environment, it is inevitable to encounter dynamic objects, such as freely moving people or rolling balls. These dynamic objects can significantly impact camera tracking: during localization, the camera struggles to acquire sufficient static features due to occlusions caused by dynamic objects, leading to localization failure. When dealing with loop closure, the displacement of dynamic objects confuses the camera as the scene exhibits different visual appearances compared to when the camera last observed the dynamic objects. Direct approaches solve the interference of dynamic objects on camera pose estimation by removing the data of dynamic objects during reconstruction.

Reference [159] introduced a robust background model-based dense-visual-odometry (BaMVO) algorithm to estimate the background for each frame and perform camera pose estimation by eliminating foreground moving objects. This method effectively reduces the impact of dynamic objects on the camera trajectory, enhancing reconstruction accuracy. Similarly, [146] filtered out the data associated with dynamic objects directly in the preprocessing stage to enhance the robustness of RGB-SLAM. Reference [160] utilized a bayesian framework for dynamic region detection, considering prior knowledge and observation information generated during object detection. After obtaining the detection results, dynamic regions are removed, and only features from static regions are extracted for camera tracking. Reference [150] combined ORB-SLAM2 with the PSPNet [161] semantic segmentation network to propose the PSPNet-SLAM system, which first removes dynamic points with large optical flow values using optical flow and then performs secondary filtering using PSPNet to ensure more accurate matching. Most of these methods detect dynamic objects by analyzing only a few consecutive frames. However, since many dynamic objects may remain static for a short period of time, this can lead to failure in detecting moving objects. Based on this observation, LC-CRF SLAM [162] constructed a long-term consistent conditional random field (CRF) that provides more accurate camera trajectory estimation through long-term observations across multiple frames. Semantic information based on deep learning can eliminate the influence of dynamic objects, but it involves high computational cost and cannot handle unknown objects. Reference [163] proposed a real-time semantic RGB-D SLAM system tailored for dynamic environments, which performs semantic segmentation only on keyframes to remove known dynamic objects and maintains a static map for robust camera tracking. After removing dynamic objects, [153] repaired missing static background information using information from keyframes to facilitate subsequent point cloud reconstruction. Reference [42] introduced a hierarchical representation method for images, segmenting images into planar and non-planar regions. By removing dynamic non-planar objects, it segments and tracks multiple dynamic planar rigid objects. Reference [40] created a sparse graph using Delaunay triangulation from all map points, and then used the correlation of the mapped points in the graph to divide the points in the scene into groups, where the largest group is considered to be the static map points. Finally, only these static points were used in estimating the camera motion. Recently, [43] incorporated an improved dense point cloud generation module into ORB-SLAM3 [84], which removes dynamic objects using dynamic object information extracted by YOLO v5, resulting in a point cloud representation of the static scene and obtaining clearer camera poses by eliminating dynamic objects.

2) INDIRECT METHOD

Instead of removing dynamic objects, the indirect method analyzes features, assigning weights to both static and dynamic objects. Based on these weights, it fully utilizes static and dynamic features to achieve scene tracking and mapping. References [41], [164], and [165] assigned weights to static points and utilized these static weights to eliminate the influence of dynamic objects. Reference [164] employed depth edge points for frame-to-keyframe registration, where each edge point is assigned a static weight that is then

used in the Intensity-assisted Iterative Closest Point (IAICP) algorithm for motion estimation, thereby reducing the impact of dynamic components. Additionally, effective loop closure detection was incorporated to decrease tracking errors. Reference [165] utilized double K-means clustering to detect dynamic objects, followed by establishing static weights for the feature points in the current frame, which comprehensively consider static probability and static observation value (SON). Finally, the traditional RANSAC algorithm was modified to suit dynamic reconstruction. With the advancement of deep learning, the integration of semantic knowledge into dynamic reconstruction yields excellent results. Reference [41] proposed the DPF-SLAM algorithm, which combines the dynamic prior probability obtained from semantic segmentation with the dynamic probability obtained from dynamic point detection, thereby reducing the influence of dynamic objects on camera localization.

In dynamic scene reconstruction, it is commonly assumed that the predominant portion of image frames represents the static background. However, in complex scenes with numerous dynamic objects, without semantic segmentation, a significant portion of dynamic objects may be mistakenly identified as static backgrounds. Moreover, dynamic objects can occlude a substantial amount of color and depth information, leading to insufficient static information in visual input to support accurate self-motion estimation of the camera. Reference [32] employed a multi-model fitting approach to identify dynamic objects in the scene using motion segmentation and semantic segmentation. Each object was reconstructed individually, and over time, increasingly refined dynamic models can be obtained. Complete geometric objects play a crucial role in tracking camera trajectories. Reference [166] inferred the complete geometric shapes of each object to establish correspondences among instances, enabling the estimation of object poses in each frame. Reference [167] employed multiple motion segmentation methods to segment the motion models of different moving objects, obtaining accurate masks for the moving objects and generating 4D models and trajectories of the moving objects in a global reference frame, while reconstructing dense maps of the static background. Reference [168] introduced rigid and motion constraints to model articulated objects, allowing for the joint optimization of camera pose, object motion, and object 3D structure. This approach corrects estimation errors in camera pose, prevents tracking loss, and generates a 4D spatiotemporal map that includes both dynamic targets and static scenes.

C. MODEL FUSION

The fusion of dynamic scenes extends the fusion of static scenes and includes two approaches: voxel-based, surfel-based, and NeRF-based.

1) VOXEL-BASED

DynamicFusion [29], as a pioneering real-time dynamic 3D reconstruction algorithm, extended Projective TSDF to reconstruct and fuse scenes. VolumeDeform [30] utilized a voxel model that not only stores scene data in the undeformed pose, such as TSDF values, color, and confidence value, but also stores information about the current spatial deformation, represented by deformation field parameters. These mainly focus on scenes with individually deformable objects moving in a non-rigid manner. When multiple dynamic objects are present in the scene, MID-Fusion [169] integrates depth, color, semantic, and foreground probability information into an object model based on an octree volume representation using foreground and background masks. EM-Fusion [34] reconstructed dynamic objects based on the TSDF model and creatively used Expectation-Maximization (EM) to determine the unknown association between pixels and objects. Reference [170] introduced a neural scene flow field, which defines a set of voxel boundary implicit fields using a sparse voxel octree, to simulate local properties and achieve the reconstruction of complex dynamic scenes. These TSDF representations store different dynamic objects of the entire scene as multiple 3D models, enabling easy fusion and updating based on their respective poses. However, during surface extraction, ray casting needs to be performed separately for each object model. Additionally, occlusion handling during ray casting is required to determine surface visibility when objects are occluded. To address these issues, [171] proposed a novel map representation method called TSDF++, which allows simultaneous reconstruction of both static scenes and dynamic objects within a 3D volume model.

2) SURFEL-BASED

Volumetric methods can generate smooth triangle meshes, but they suffer from high computational and memory costs. Surfel-based methods, on the other hand, are more efficient, but post-processing is required if a mesh model is desired [172]. In the case of non-rigid objects in dynamic scenes, the computation becomes more complex, making surfel-based representations a promising solution for real-time dynamic reconstruction. Co-Fusion [32] extended the surfel-based mapping framework of ElasticFusion to handle dynamic scenes, enabling tracking and reconstruction of segmented dynamic objects based on motion and semantic cues in each frame. However, Co-Fusion lacks real-time capabilities for dynamic scene reconstruction. MaskFusion [173] built upon Co-Fusion to create a real-time dynamic 3D reconstruction system, representing each geometric entity as a set of surfaces and incorporating semantic information into the map. Reference [33] proposed a strategy to assess the feasibility of each surfel by estimating the dynamic probability of each input point and eliminating surfels that match dynamic input points. Reference [174] achieved real-time non-rigid reconstruction using a stream of depth images as input, along with surfel-based scene representation, effectively handling topological changes and tracking failures, resulting in efficient dynamic 3D reconstruction. Surfel maps typically consist of a large number of surfels, requiring powerful GPUs

for online processing. Reference [175] introduced the use of superpixels to generate surfel maps, significantly improving storage efficiency and speed. If an observation or fusion count falls below a threshold within a certain time frame, the corresponding surfels are considered outliers and are removed to mitigate the influence of dynamic objects on the reconstruction results. The SLAM system proposed in [176] relies on super-surface elements [177], which are planar patches generated from superpixels, to model the static parts of the environment.

3) NERF-BASED

There have been numerous attempts to apply NeRF to scenes from RGB video streams [178], [179]. However, these methods reconstruct scenes under the assumption of accurate camera poses, heavily relying on previous camera registration, which can fail when objects undergo significant motion. Unlike NeRF reconstruction solely from RGB data, the introduction of depth information makes these issues more manageable. Simultaneously, to handle geometries beyond the sensor's explicit range and regions with low reflectivity, raw continuous-wave ToF images are used instead of direct depth maps, effectively enhancing the view synthesis quality in dynamic scenes [180].

Reference [181] proposed a new framework called Time-Aware Neural Voxels (TiNeuVox), which by explicitly representing temporal information in dynamic scenes, making the modeling and rendering of dynamic scenes more efficient. Specifically, it inputs the time embedding $t_i = \Phi_d(\gamma(t_i))$ and the coordinates of the sample points (x, y, z) into a compact deformation network Φ_d, resulting in the deformed coordinates (x', y', z'):

$$x', y', z' = \Phi_d(x, y, z, t_i) \tag{11}$$

Additionally, to more accurately reconstruct the motion trajectories of points and improve training speed, [181] utilizes a multi-distance interpolation method to capture motion at different scales:

$$v = v_1 \oplus \cdots v_m \cdots \oplus v_M, \tag{12}$$

$$v_m = \text{interp}(x, y, z, V[:: s_m]), \tag{13}$$

v is the concatenation result of interpolated features v_m across multiple scales. v_m is obtained by interpolating at the point (x, y, z) after sampling the voxel grid V with a stride of s_m. Moreover, Neural-Dynamic Reconstruction (NDR) was proposed to recover high-fidelity geometric shapes and motion of dynamic scenes from monocular RGB-D cameras [182]. It employs a novel neural reversible deformation network to represent and constrain non-rigid deformations. A topologically aware strategy is employed to establish correspondences for fused frames. Building upon this, DNA-Net [183] modeled dynamic motions using articulated bones, aiding the model in faster convergence and making it more suitable for applications such as human pose manipulation. Due to errors in pose estimation that SFM algorithms often produce in highly dynamic scenarios with poor-textured surfaces, [184] reconstructed the entire scene using both static NeRF and dynamic NeRF. The static NeRF is responsible for reconstructing static scenes and estimating camera poses and focal lengths. Simultaneously, the dynamic NeRF simulates the dynamic aspects of the scene from the video. Recently, the effectiveness of depth-constrained NeRF in dynamic operating rooms [185] has been validated, demonstrating the generation of geometrically consistent views from novel perspectives.

V. DATASET AND EVALUATION METRICS

A. DATASETS

In this section, we present the relevant datasets used by the static and dynamic reconstruction algorithms. Table 3 briefly summarizes the scenes, details, applications and publication years included in the relevant datasets.

The TUM dataset [186] utilizes Microsoft Kinect to capture RGB-D data of office scenes and an industrial hall. The dataset consists of 39 sequences, including color images, depth maps, and ground truth camera poses associated with time. The fr1 and fr2 sequences primarily feature static scenes. The fr3 sequences are used for quality evaluation of dynamic 3D reconstruction. In the fr3/sitting sequence, two people are sitting and lightly moving while chatting at a table. The fr3/walking sequence involves two people walking around a table, exhibiting highly dynamic motion. Additionally, the dataset provides two evaluation metrics, Relative Pose Error (RPE) and Absolute Trajectory Error (ATE), which can be used to assess the performance of visual SLAM systems.

ICL-NUIM dataset [187] is used for evaluating RGB-D visual odometry, 3D reconstruction, and SLAM systems. The data acquisition involves capturing images from a camera trajectory of 3D models rendered using POVRay. The dataset includes two different scenes, a living room and an office, with the living room scene being a newly introduced scene that includes relevant 3D polygon models to assess the accuracy of the final reconstruction results. Aug-ICL-NUIM dataset [9] expands the ICL-NUIM dataset with challenging camera trajectories and realistic noise models, enhancing the dataset in various ways to accommodate the evaluation of complete scene reconstruction pipelines.

The CoRBS dataset [188] provides real depth and color data, along with ground truth camera trajectories and 3D models of the scenes. It includes 20 KinectV2 image sequences captured from four different scenes. The data is directly provided in a global coordinate system, enabling direct evaluation without the need for further alignment or calibration.

The NYUD v2 dataset [189] consists of 1449 indoor RGB-D images captured using Microsoft Kinect devices, accompanied by detailed annotations. However, due to its relatively small scale, it is challenging to apply it to deep learning architectures. The SUN3D dataset [190] offers a

TABLE 3. Overview of static and dynamic datasets.

Dataset	Scenes	Details	Applications	Years
TUM	Static/Dynamic	39 sequences from office and industrial hall	Evaluation of drift in visual odometry systems and global pose error in SLAM systems	2012
ICL-NUIM	Static	8 sequences from living room and office	Evaluation of visual odometry , SLAM trajectory estimation and surface reconstruction accuracy	2014
CoRBS	Static	20 Kinect V2 image sequences captured from four different scenes	Evaluation of 3D reconstruction and visual odometry algorithms	2016
NYU Depth	Static	64 scenes with a total of 108,617 frames (2,347 are labeled)	3D reconstruction, object recognition and semantic segmentation for indoor scenes	2011(v1)
		464 scenes and 407,024 frames (1,449 images are labeled)		2012(v2)
SUN3D	Static	Large-scale RGB-D video dataset containing indoor scenes such as homes, offices, shopping malls, etc.		2013
The RGB-D Object Dataset	Static	300 objects in 51 categories		2011
SceneNN	Static	Gridded 100 scenes with vertex-by-vertex and pixel-by-pixel annotations.		2016
SceneNet RGB-D	Static	5M rendered RGB-D images, 16K randomly generated 3D tracks in synthetic layout		2017
PASCAL VOC	Dynamic	More than 20 potential mobile object classes, such as people, cats, dogs, etc.	Training segmentation networks to segment dynamic objects	2010
COCO dataset	Dynamic	Photographs of 91 object types and a total of 2.5 million instances of markers in 328k images		2014
ShapeNet	Dynamic	More than 3M 3D models, of which 220K models are categorized into 3,135 categories		2015
The Bonn RGB-D Dynamic Dataset	Static/Dynamic	24 dynamic sequences and 2 static sequences	Evaluation of drift in visual odometry systems and global pose error in SLAM systems	2019
HRPSlam	Dynamic	Two full loops, totaling more than 50,000 frames, acquired from the on-board camera on the HRP-4 humanoid robot	RGB-D Dynamic SLAM Benchmarks for Evaluating Humanoid Robots	2019
DeepDeform	Dynamic	400 scenes, 390,000 frames, 5,533 densely aligned frame pairs	Non-rigid 3D reconstruction based on a data-driven approach	2020
The Oxford-IHM Dataset	Dynamic	60-minute video of the human body walking indoors	Motion planning, mapping and human trajectory prediction	2023

large-scale RGB-D video database that includes semantic information and corresponding pose information for objects in each scene. The RGB-D Object dataset [191] contains 300 objects from 51 categories, providing high-quality color and depth images for each frame in the videos. Additionally, it introduces RGB-D-based object recognition and detection techniques that significantly improve the reconstruction quality by leveraging both color and depth information.

In widely used RGB-D datasets, there is a challenge of lacking comprehensive and fine-grained annotations. Binh-Son Hua et al. introduced SceneNN [192], which collects RGB-D information from over 100 indoor scenes, including dormitories, offices, classrooms, etc. The authors reconstructed each scene using triangle meshes and added per-vertex and per-pixel annotations, enriching the dataset with fine-grained information. John McCormac introduced SceneNet RGB-D [193], which provides accurate pixel-level semantic information for scene understanding tasks such as object detection, semantic segmentation, and instance segmentation. Additionally, it provides camera poses and depth data to facilitate the study of geometric problems in computer vision.

The PASCAL VOC dataset [194] provides more than 20 object classes that can be used for handling potentially moving objects, such as humans, cats, and dogs. This dataset can be used to train segmentation networks for segmenting dynamic objects. Compared to PASCAL VOC, the MS COCO dataset [195] offers a larger number of categories and instances, enabling models to better learn contextual information. ShapeNet [196] is a large-scale 3D model dataset that includes 3D models from various semantic categories, making it suitable for part segmentation tasks.

The Bonn RGB-D dynamic dataset [197] provides rich and complex dynamic data, including 24 highly dynamic scenes. To apply data-driven methods to non-rigid 3D reconstruction, DeepDeform [198] utilizes a semi-supervised labeling approach and obtains a large dataset of 400 scenes, consisting of over 390,000 RGB-D frames and 5,533 densely aligned frame pairs. HRPSlam [199] is the first system to capture dynamic RGB-D data using a humanoid robot, simulating scenarios such as human walking with jitter or falling. The provided dataset includes two complete loops and can be used for evaluating global loop closure or local reconstruction in dynamic environments. The Oxford-IHM

dataset [200] uses multiple large objects as static obstacles and records the walking trajectories of people in indoor environments.

B. EVALUATION METRICS

In 3D reconstruction, the accuracy of the camera pose directly affects the reconstruction accuracy, making it crucial to evaluate pose accuracy. If the pose error is large, it can cause the reconstructed model to be distorted or inaccurate, thereby affecting the overall quality of the model. The metrics for evaluating pose accuracy mainly include Relative Pose Error (RPE), Absolute Pose Error (ATE), and Alignment Error (AE). The overall quality of the reconstruction can be measured by surface accuracy. Next, we will specifically introduce these evaluation metrics.

Relative Pose Error (RPE): RPE measures the local accuracy of the trajectory over a fixed time interval Δ. For the estimated trajectory $P_1, \ldots, P_n \in SE(3)$ and the ground truth trajectory $Q_1, \ldots, Q_n \in SE(3)$, the relative pose error at time step i is given by:

$$E_i = \left(Q_i^{-1} Q_{i+\Delta}\right)^{-1} \left(P_i^{-1} P_{i+\Delta}\right) \tag{14}$$

From a sequence of n camera poses, we can obtain $m = n - \Delta$ relative pose errors. The overall relative pose error is evaluated by calculating the root mean square error (RMSE) of the translation components across all time indices:

$$\mathrm{RMSE}(E_{1:n}, \Delta) = \left(\frac{1}{m} \sum_{i=1}^{m} \|\mathrm{trans}(E_i)\|^2\right)^{1/2} \tag{15}$$

where $\mathrm{trans}(E_i)$ represents the translation component of the relative pose error E_i. The Relative Pose Error is an important metric for evaluating the local accuracy of trajectory estimation. By comparing the relative motion over a period of time, it effectively measures and analyzes the system's pose accuracy over short time intervals.

Absolute trajectory error (ATE): ATE provides a direct numerical measure that intuitively reflects the algorithm's accuracy and the global consistency of the trajectory. When calculating ATE, the trajectories are first aligned using the Horn method [201], which finds the rigid body transformation S corresponding to the least-squares solution that maps the estimated trajectory P_i to the ground truth trajectory Q_i. After obtaining this transformation, the absolute trajectory error F_i at time step i can be computed as:

$$F_i := Q_i^{-1} S P_i \tag{16}$$

The Absolute Pose Error is evaluated by calculating the root mean square error (RMSE) of the translation components across all time indices:

$$\mathrm{RMSE}(F_{1:n}) := \left(\frac{1}{n} \sum_{i=1}^{n} \|\mathrm{trans}(F_i)\|^2\right)^{1/2} \tag{17}$$

ATE can effectively measure the accuracy and consistency of reconstruction systems, providing a standardized evaluation tool for the comparison of different algorithms.

TABLE 4. Absolute Trajectory Error (ATE) of different algorithms on the TUM dataset (m). The best results are in bold and the second best results are in italics. The best results are in bold and the second best results are in italics.

Method	fr1/desk	fr2/xyz	fr3/office	fr3/nst
DVO SLAM [74]	0.021	0.018	0.035	0.018
Kintinuous [7]	0.037	0.029	0.030	0.031
Voxelhashing [8]	0.023	0.022	0.023	0.087
ElasticFusion [10]	0.02	0.011	0.017	0.016
ORB-SLAM2 [83]	0.016	**0.004**	0.010	0.019
Redwood [9]	0.027	0.091	0.030	1.929
LC Reconstruction [122]	0.018	0.015	0.025	-
RGBDTAM [75]	0.027	**0.004**	0.027	0.016
Bundlefusion [11]	0.016	0.011	0.022	**0.012**
RH Reconstruction [203]	0.015	0.006	*0.009*	*0.014*
BAD SLAM [204]	0.017	0.011	0.017	0.019
ID-RGBDO [87]	0.051	0.007	0.038	0.018
Manhattanslam [22]	0.027	0.008	-	-
Fastfusion [24]	0.018	0.012	0.024	-
Hbrf-fusion [25]	**0.014**	*0.005*	**0.007**	0.016
iMAP [23]	0.049	0.020	0.058	-
NICE-SLAM [26]	0.027	0.018	0.030	-
Co-SLAM [27]	0.024	0.017	0.024	-
NGEL-SLAM [133]	*0.015*	*0.005*	0.010	-

TABLE 5. Surface Accuracy of different algorithms on the ICL-NUIM dataset (m). The best results are in bold and the second best results are in italics.

Method	lr_kt0	lr_kt1	lr_kt2	lrk_kt3
KinectFusion [6]	0.071	0.144	0.216	0.359
DVO SLAM [74]	0.032	0.061	0.119	0.053
Kintinuous [7]	0.011	0.008	0.009	0.150
Voxelhashing [8]	0.014	**0.004**	0.018	0.120
Redwood [9]	0.020	0.020	0.013	0.022
ElasticFusion [10]	0.007	0.007	0.008	0.028
LC reconstruction [122]	0.013	0.011	**0.001**	0.014
Bundlefusion [11]	*0.005*	0.006	0.007	0.008
RH reconstruction [203]	**0.004**	*0.005*	0.004	**0.006**
Hbrf-fusion [25]	**0.004**	**0.004**	*0.003*	*0.007*

Alignment Error (AE): AE [202] is a comprehensive metric that balances the effects of scale, rotation, and translational drift on the trajectory. Suppose $p_1, \ldots, p_n \in \mathbb{R}^3$ represent the tracked positions from frame 1 to frame n. Let $S \subset [1; n]$ and $E \subset [1; n]$ represent the frame indices for the start and end segments, respectively, for which aligned ground truth positions $\hat{p} \in \mathbb{R}^3$ are provided. Independently aligning the tracked trajectory with the start and end segments provides two relative transformations:

$$T_s^{gt} := \underset{T \in \mathrm{Sim}(3)}{\arg\min} \sum_{i \in S} \left(T p_i - \hat{p}_i\right)^2 \tag{18}$$

$$T_e^{gt} := \underset{T \in \mathrm{Sim}(3)}{\arg\min} \sum_{i \in E} \left(T p_i - \hat{p}_i\right)^2 \tag{19}$$

where Sim(3) represents the group of similarity transformations in three-dimensional space. The alignment error between the tracked trajectories when aligned at the start and

TABLE 6. Absolute Trajectory Error (ATE) of different algorithms on the TUM dataset (m). The best results are in bold and the second best results are in italics.

	Fr3_s_static	Fr3_s_xyz	Fr3_s_half	Fr3_w_static	Fr3_w_xyz	Fr3_w_half
ORB-SLAM2 [83]	0.008	0.010	0.035	0.390	0.614	0.789
Co-fusion [32]	-	0.027	-	-	0.696	-
Staticfusion [33]	0.014	0.039	0.041	0.015	0.093	0.681
DynaSLAM [155]	-	0.015	0.017	**0.006**	0.015	0.025
DS-SLAM [156]	*0.007*	-	**0.015**	0.008	0.025	0.303
Maskfusion [173]	0.021	0.031	0.052	0.035	0.104	0.106
ReFusion [197]	0.009	0.040	0.110	0.017	0.099	-
DM-SLAM [35]	**0.006**	-	0.018	0.008	*0.015*	0.027
RE-SLAM [36]	**0.006**	-	-	*0.007*	0.018	0.027
PLD-SLAM [205]	**0.006**	0.009	**0.015**	*0.007*	**0.014**	0.026
MOSD-SLAM [206]	*0.007*	0.013	0.019	0.010	**0.014**	0.028
LC-CRF SLAM [162]	-	0.013	0.026	0.028	0.020	0.030
SPON-SLAM [165]	0.009	*0.009*	**0.015**	0.011	0.016	0.036
RigidFusion [38]	-	0.097	-	-	0.195	
ORB-SLAM3 [84]	-	0.026	0.230	-	0.162	0.189
RS-SLAM [163]	-	0.012	0.017	0.011	0.019	0.029
RTCB-SLAM [39]	**0.006**	0.012	0.017	0.016	**0.014**	**0.023**
DSV-SLAM [43]	-	0.017	0.176	-	**0.014**	0.055
RTDSLAM [207]	-	**0.008**	0.020	*0.007*	*0.015*	**0.023**
SEG-SLAM [157]	**0.006**	0.011	*0.016*	0.008	**0.014**	*0.024*
CPR-SLAM [208]	-	0.011	0.022	0.010	0.016	0.049
DDN-SLAM [158]	-	0.010	0.017	0.010	**0.014**	**0.023**

TABLE 7. Results of Relative Pose Error (RPE) in translation error for different algorithms on the TUM dataset (m/s). The best results are in bold and the second best results are in italics.

	Fr3_s_static	Fr3_s_xyz	Fr3_s_half	Fr3_w_static	Fr3_w_xyz	Fr3_w_half
ORB-SLAM2 [83]	0.010	**0.012**	0.023	0.193	0.483	0.322
Co-fusion [32]	-	0.027	-	-	0.329	-
Staticfusion [33]	-	0.028	0.030	0.013	0.121	0.207
Dynaslam [155]	0.013	0.015	**0.019**	**0.009**	0.022	0.028
RigidFusion [38]	-	0.035	-	-	0.134	-
DS-SLAM [156]	*0.008*	-	-	*0.010*	0.033	0.030
Em-fusion [34]	-	0.026	-	-	0.060	-
Semantic SLAM [209]	0.009	-	-	0.010	*0.020*	*0.027*
RS-SLAM [163]	-	0.017	0.026	0.012	0.023	0.042
RTCB-SLAM [39]	**0.007**	*0.014*	*0.022*	0.014	**0.018**	**0.024**
RTDSLAM [207]	-	0.021	0.036	*0.010*	0.021	**0.024**
CPR-SLAM [208]	-	0.018	0.032	0.014	0.024	0.087

end segments is given by:

$$e_{\text{align}} := \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|T_s^{gt} p_i - T_e^{gt} p_i\|_2^2} \tag{20}$$

Surface accuracy: Surface accuracy measures the quality of algorithmic reconstruction by calculating the distance between the estimated surface and the ground truth surface. For the distances of all vertices in the reconstruction, there are five standard statistics: mean, median, standard deviation, minimum and maximum. A smaller value for this indicator indicates a better quality reconstruction.

1) EVALUATION IN STATIC SCENARIOS

In this section, we quantitatively compared the ATE and surface accuracy of the following algorithms: [6], [7], [8], [9], [10], [11], [22], [23], [24], [25], [26], [27], [74], [75], [83], [87], [122], [133], [203], [204].

As shown in Table 4, we tested different algorithms on four sequences from the TUM dataset: fr1/desk, fr2/xyz, fr3/office, and fr3/nst, and calculated the absolute trajectory error (ATE) for each algorithm on this dataset. By comparing the ATE results, we can assess the accuracy of the pose estimation of the algorithms and thereby evaluate their performance. The data in the Table is sourced from the respective papers, where "-" indicates that the corresponding data was not found in the paper. The results are reported with three decimal places of precision. From the experimental results, Hbrf-fusion [25] achieved the best results in the fr1/desk and fr3/office scenes and also performed well in the fr2/xyz and fr3/nst scenes. This is because it uses the dynamic implicit Hermite radial basis function (HRBF) as a

method for representing continuous surfaces, unlike explicit methods such as Kintinuous [7], Voxelhashing [8], and ElasticFusion [10]. Additionally, NGEL-SLAM [133], which uses neural implicit representations, also achieved good results in the fr1/desk, fr2/xyz, and fr3/office scenes. The results show that implicit representation methods can better align reconstructed trajectories, reduce trajectory errors, and thus improve reconstruction quality. For evaluating the quality of surface reconstruction, the ICL-NUIM dataset provides ground truth 3D models for generating virtual scan sequences. We use the lr_kt0, lr_kt1, lr_kt2, and lr_kt3 sequences from the living room scene in this dataset as a benchmark for estimating the algorithm's surface reconstruction performance. The surface accuracy (median distance) of each method is shown in Table 5. From the experimental results, the implicit representation method Hbrf-fusion [25] achieved the best results in the lr_kt0 and lr_kt1 scenes, and the second-best results in the lr_kt2 and lr_kt3 scenes. Its surface accuracy surpasses explicit representation methods such as KinectFusion [6], DVO SLAM [74], and Kintinuous [7]. The results indicate that the implicit representation method used by Hbrf-fusion can significantly reduce reconstruction errors and improve surface accuracy.

2) EVALUATION IN DYNAMIC SCENARIOS

In this section, we quantitatively compared the ATE and RPE of the following algorithms: [32], [33], [34], [35], [36], [38], [39], [43], [83], [84], [155], [156], [157], [158], [163], [165], [173], [197], [205], [206], [207], [208], [209].

We used dynamic sequences from the TUM dataset to evaluate the performance of dynamic reconstruction algorithms. The sequences in the sitting (s) category, where two people are conversing at a desk, are used to assess the robustness of the algorithms to slowly moving dynamic objects. The sequences in the walking (w) category, where two people are walking in an office scene, can be used to evaluate the robustness of the algorithms to quickly moving dynamic objects. Tables 6 and 7 respectively list the performance of outstanding algorithms on the TUM dataset in recent years. We use ATE and RPE as evaluation metrics for the algorithms. In the tables, static, xyz, and half represent different camera movement modes. The data in the tables are sourced from the respective papers, where "-" indicates that corresponding data was not found in the paper. The results are reported with three decimal places of precision. The best results are in bold and the second best results are in italics. From the experimental results, with the development of deep learning and the introduction of semantic information, models can achieve excellent results not only in low dynamic scenes such as Fr3_s_static, Fr3_s_xyz, and Fr3_s_half but also in highly dynamic environments like Fr3_w_static, Fr3_w_xyz, and Fr3_w_half, obtaining accurate camera trajectories. Examples include PLD-SLAM [205], RTCB-SLAM [39], RTDSLAM [207], SEG-SLAM [157], and DDN-SLAM [158].

VI. CONCLUSION

This study investigates and analyzes indoor scenes, classifying them into static and dynamic scenes, and provides a comprehensive survey of recent reconstruction algorithms.

For static scenes, we outline the general reconstruction process and introduce various optimization algorithms employed in each step. From the reviewed methods, it can be seen that traditional static reconstruction tasks often use explicit scene representations, such as voxels and surfels, which enable real-time scene reconstruction but result in artifacts and holes. This is due to the discontinuous nature of explicit representation methods. The emergence of neural radiance fields (NeRF) provides an implicit representation for 3D reconstruction, making the scene representation more continuous. However, because it requires densely sampling points in space and using MLP to learn scene information, it consumes a lot of training resources and time. Currently, to balance training time and quality, a promising direction is to combine explicit and implicit representations, such as Point-NeRF [210], H_2-Mapping [132], and NGEL-SLAM [133].

In contrast, dynamic scenes involve not only camera motion but also other moving objects, which may interfere with camera tracking. Therefore, for reconstructing dynamic scenes, it is necessary to identify dynamic objects through motion segmentation, eliminate or utilize dynamic features for pose estimation, and integrate the scene data into the reconstruction model. Currently, a popular approach is to use deep learning methods to leverage semantic information of the scene to segment dynamic objects, thereby improving the quality of scene reconstruction. Additionally, NeRF also provides a new direction for dynamic reconstruction. Because NeRF uses MLP to represent the scene more continuously, it can even fill in information that the camera has not observed.

DECLARATION OF COMPETING INTEREST

The authors have no competing interests to declare that are relevant to the content of this article.

REFERENCES

[1] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: Applications to mobile robotics," *Intell. Ind. Syst.*, vol. 1, no. 4, pp. 289–311, Dec. 2015.

[2] K. Chen, Y.-K. Lai, and S.-M. Hu, "3D indoor scene modeling from RGB-D data: A survey," *Comput. Vis. Media*, vol. 1, no. 4, pp. 267–278, Dec. 2015.

[3] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3D reconstruction with RGB-D cameras," in *Computer Graphics Forum*. Hoboken, NJ, USA: Wiley, 2018, pp. 625–652.

[4] K. A. Tychola, I. Tsimperidis, and G. A. Papakostas, "On 3D reconstruction using RGB-D cameras," *Digital*, vol. 2, no. 3, pp. 401–421, Aug. 2022.

[5] J. Li, W. Gao, Y. Wu, Y. Liu, and Y. Shen, "High-quality indoor scene 3D reconstruction with RGB-D cameras: A brief review," *Comput. Vis. Media*, vol. 8, no. 3, pp. 369–393, Sep. 2022.

[6] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[7] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," *Robot. Auto. Syst.*, vol. 1, no. 1, pp. 1–12, 2012.

[8] 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.

[9] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5556–5565.

[10] T. Whelan, S. Leutenegger, R. Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense SLAM without a pose graph," *Robot. Sci. Syst.*, vol. 7, p. 1697, Aug. 2015.

[11] 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, p. 1, Aug. 2017.

[12] Q. He, X. Sun, Z. Yan, B. Wang, Z. Zhu, W. Diao, and M. Y. Yang, "AST: Adaptive self-supervised transformer for optical remote sensing representation," *ISPRS J. Photogramm. Remote Sens.*, vol. 200, pp. 41–54, Jun. 2023.

[13] Q. He, X. Sun, Z. Yan, B. Li, and K. Fu, "Multi-object tracking in satellite videos with graph-based multitask modeling," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5619513.

[14] T. Sun and C. Jung, "NIR image colorization using SPADE generator and grayscale approximated self-reconstruction," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2020, pp. 463–466.

[15] Q. He, "Prompting multi-modal image segmentation with semantic grouping," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 2094–2102.

[16] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "PointGroup: Dual-set point grouping for 3D instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4866–4875.

[17] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9028–9037.

[18] G. Tziafas and H. Kasaei, "Early or late fusion matters: Efficient RGB-D fusion in vision transformers for 3D object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 9558–9565.

[19] D. Rukhovich, A. Vorontsova, and A. Konushin, "TR3D: Towards real-time indoor 3D object detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2023, pp. 281–285.

[20] 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, May 2013.

[21] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. S. Torr, and D. W Murray, "InfiniTAM v3: A framework for large-scale 3D reconstruction with loop closure," 2017, *arXiv:1708.00783*.

[22] R. Yunus, Y. Li, and F. Tombari, "ManhattanSLAM: Robust planar tracking and mapping leveraging mixture of Manhattan frames," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 6687–6693.

[23] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "IMAP: Implicit mapping and positioning in real-time," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6209–6218.

[24] Z. Zhu, Z. Xu, R. Chen, T. Wang, C. Wang, C. Yan, and F. Xu, "FastFusion: Real-time indoor scene reconstruction with fast sensor motion," *Remote Sens.*, vol. 14, no. 15, p. 3551, Jul. 2022.

[25] Y. Xu, L. Nan, L. Zhou, J. Wang, and C. C. L. Wang, "HRBF-fusion: Accurate 3D reconstruction from RGB-D data using on-the-fly implicits," *ACM Trans. Graph.*, vol. 41, no. 3, pp. 1–19, Jun. 2022.

[26] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12776–12786.

[27] H. Wang, J. Wang, and L. Agapito, "Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 13293–13302.

[28] A. L. Teigen, Y. Park, A. Stahl, and R. Mester, "RGB-D mapping and tracking in a plenoxel radiance field," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2024, pp. 3342–3351.

[29] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 343–352.

[30] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-time volumetric non-rigid reconstruction," in *Computer Vision–ECCV 2016*. Cham, Switzerland: Springer, 2016, pp. 362–379.

[31] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic, "KillingFusion: Non-rigid 3D reconstruction without correspondences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5474–5483.

[32] M. Runz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4471–4478.

[33] 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)*, May 2018, pp. 3849–3856.

[34] M. Strecke and J. Stueckler, "EM-fusion: Dynamic object-level SLAM with probabilistic data association," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5864–5873.

[35] J. Cheng, Z. Wang, H. Zhou, L. Li, and J. Yao, "DM-SLAM: A feature-based SLAM system for rigid dynamic scenes," *ISPRS Int. J. Geo-Information*, vol. 9, no. 4, p. 202, Mar. 2020.

[36] X. Yang, Z. Yuan, D. Zhu, C. Chi, K. Li, and C. Liao, "Robust and efficient RGB-D SLAM in dynamic environments," *IEEE Trans. Multimedia*, vol. 23, pp. 4208–4219, 2021.

[37] T. Zhang, H. Zhang, Y. Li, Y. Nakamura, and L. Zhang, "FlowFusion: Dynamic dense RGB-D SLAM based on optical flow," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 7322–7328.

[38] R. Long, C. Rauch, T. Zhang, V. Ivan, and S. Vijayakumar, "RigidFusion: Robot localisation and mapping in environments with large dynamic rigid objects," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3703–3710, Apr. 2021.

[39] F. Zhu, S. Zheng, X. Huang, and X. Wang, "Robust tracking and clean background dense reconstruction for RGB-D SLAM in a dynamic indoor environment," *Machines*, vol. 10, no. 10, p. 892, Oct. 2022.

[40] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "RGB-D SLAM in dynamic environments using point correlations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 373–389, Jan. 2022.

[41] X. Liu, S. Wen, M. Yuan, P. Li, Y. Zhao, and L. Manfredi, "DPF-SLAM: Dense semantic SLAM based on dynamic probability fusion in dynamic environments," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, Jul. 2022, pp. 360–365.

[42] R. Long, C. Rauch, T. Zhang, V. Ivan, T. L. Lam, and S. Vijayakumar, "RGB-D SLAM in indoor planar environments with multiple large dynamic objects," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8209–8216, Jul. 2022.

[43] H. Guan, C. Qian, T. Wu, X. Hu, F. Duan, and X. Ye, "A dynamic scene vision SLAM method incorporating object detection and object characterization," *Sustainability*, vol. 15, no. 4, p. 3048, Feb. 2023.

[44] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, Jan. 2022.

[45] D. Azinovic, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural RGB-D surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6280–6291.

[46] J. Wang, T. Bleja, and L. Agapito, "GO-surf: Neural feature grid optimization for fast, high-fidelity RGB-D surface reconstruction," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2022, pp. 433–442.

[47] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, Jul. 2023.

[48] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural radiance fields for dynamic scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10313–10322.

[49] G.-W. Yang, W.-Y. Zhou, H.-Y. Peng, D. Liang, T.-J. Mu, and S.-M. Hu, "Recursive-NeRF: An efficient and dynamically growing NeRF," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 12, pp. 1–14, May 2022.

[50] H. Cai, W. Feng, X. Feng, Y. Wang, and J. Zhang, "Neural surface reconstruction of dynamic scenes with monocular RGB-D camera," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2022, pp. 1–20.

[51] S.-M. Kim, "Depth video enhancement for haptic interaction using a smooth surface reconstruction," *IEICE Trans. Inf. Syst.*, vol. 89, no. 1, pp. 37–44, Jan. 2006.

[52] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.

[53] L. Wang, H. Jin, R. Yang, and M. Gong, "Stereoscopic inpainting: Joint color and depth completion from stereo images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[54] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by Kinect depth camera," in *Proc. 3DTV Conf., True Vis. Capture, Transmiss. Display 3D Video*, May 2011, pp. 1–4.

[55] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, "Color-guided depth recovery from RGB-D data using an adaptive autoregressive model," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3443–3458, Aug. 2014.

[56] Q.-Y. Zhou, S. Miller, and V. Koltun, "Elastic fragments for dense scene reconstruction," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 473–480.

[57] D. Zhou, R. Wang, X. Yang, Q. Zhang, and X. Wei, "Depth image super-resolution reconstruction based on a modified joint trilateral filter," *Roy. Soc. Open Sci.*, vol. 6, no. 1, Jan. 2019, Art. no. 181074.

[58] X. Gu, Y. Guo, F. Deligianni, and G.-Z. Yang, "Coupled real-synthetic domain adaptation for real-world deep depth enhancement," *IEEE Trans. Image Process.*, vol. 29, pp. 6343–6356, 2020.

[59] Z. Zhao, J. Zhang, S. Xu, Z. Lin, and H. Pfister, "Discrete cosine transform network for guided depth map super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5687–5697.

[60] Y. Zhang and T. Funkhouser, "Deep depth completion of a single RGB-D image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 175–185.

[61] S. Yan, C. Wu, L. Wang, F. Xu, L. An, K. Guo, and Y. Liu, "DDRNet: Depth map denoising and refinement for consumer depth cameras using cascaded CNNs," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 151–167.

[62] G. Agresti, H. Schäfer, P. Sartor, Y. Incesu, and P. Zanuttigh, "Unsupervised domain adaptation of deep networks for ToF depth refinement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9195–9208, Dec. 2022.

[63] N. Moran, D. Schmidt, Y. Zhong, and P. Coady, "Noisier2Noise: Learning to denoise from unpaired noisy data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12061–12069.

[64] W. Wang, F. Wen, Z. Yan, and P. Liu, "Optimal transport for unsupervised denoising learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2104–2118, Feb. 2023.

[65] J. Xu, Y. Huang, M.-M. Cheng, L. Liu, F. Zhu, Z. Xu, and L. Shao, "Noisy-as-clean: Learning self-supervised denoising from corrupted image," *IEEE Trans. Image Process.*, vol. 29, pp. 9316–9329, 2020.

[66] Y. Xie, Z. Wang, and S. Ji, "Noise2same: Optimizing a self-supervised bound for image denoising," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20320–20330.

[67] J. Wang, P. Liu, and F. Wen, "Self-supervised learning for RGB-guided depth enhancement by exploiting the dependency between RGB and depth," *IEEE Trans. Image Process.*, vol. 32, pp. 159–174, 2023.

[68] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. Bellingham, WA, USA: SPIE, 1992, pp. 586–606.

[69] K.-L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," *Chapel Hill, Univ. North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[70] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*. Washington, DC, USA: MIT Press, 2010, pp. 161–168.

[71] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 647–663, Apr. 2012.

[72] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 598–626, Apr. 2015.

[73] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5724–5731.

[74] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3748–3754.

[75] 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)*, Sep. 2017, pp. 6756–6763.

[76] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 742–749.

[77] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. 3rd Int. Conf. 3-D Digit. Imag. Model.*, Jun. 2001, pp. 145–152.

[78] D. Haehnel, S. Thrun, and W. Burgard, "An extension of the ICP algorithm for modeling nonrigid objects with mobile robots," *IJCAI*, vol. 3, pp. 915–920, Jun. 2003.

[79] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–8, Jul. 2013.

[80] K. Pirker, M. Rüther, G. Schweighofer, and H. Bischof, "GpSLAM: Marrying sparse geometric and dense probabilistic visual mapping," in *Proc. BMVC*, 2011, pp. 1–12.

[81] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 1691–1696.

[82] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[83] R. Mur-Artal and J. D. Tardós, "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.

[84] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[85] 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.*, Nov. 2013, pp. 1568–1575.

[86] F. Schenk and F. Fraundorfer, "RESLAM: A real-time robust edge-based SLAM system," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 154–160.

[87] A. Fontán, J. Civera, and R. Triebel, "Information-driven direct RGB-D odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4928–4936.

[88] Y. Xu and C. Jung, "Face 2D to 3D reconstruction network based on head pose and 3D facial landmarks," in *Proc. Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2021, pp. 1–5.

[89] L. Ma, C. Kerl, J. Stuckler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1285–1291.

[90] D. Thomas and A. Sugimoto, "A flexible scene representation for 3D reconstruction using an RGB-D camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2800–2807.

[91] Q.-Y. Zhou and V. Koltun, "Depth camera tracking with contour cues," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 632–638.

[92] F. Schenk and F. Fraundorfer, "Combining edge images and depth maps for robust visual odometry," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[93] P. Wu, W. Li, and M. Yan, "3D scene reconstruction based on improved ICP algorithm," *Microprocessors Microsystems*, vol. 75, Jun. 2020, Art. no. 103064.

[94] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "INeRF: Inverting neural radiance fields for pose estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 1323–1330.

[95] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, "GNeRF: GAN-based neural radiance field without posed camera," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6331–6341.

[96] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "BARF: Bundle-adjusting neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5721–5731.

[97] W. Bian, Z. Wang, K. Li, and J.-W. Bian, "NoPe-NeRF: Optimising neural radiance field with no pose prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 4160–4169.

[98] Y. Xiao, N. Xue, T. Wu, and G.-S. Xia, "Level-S2fM: Structure from motion on neural level set of implicit surfaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 17205–17214.

[99] F. Steinbrucker, C. Kerl, D. Cremers, and J. Sturm, "Large-scale multi-resolution surface reconstruction from RGB-D sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3264–3271.

[100] O. Guclu and A. B. Can, "Integrating global and local image features for enhanced loop closure detection in RGB-D SLAM systems," *Vis. Comput.*, vol. 36, no. 6, pp. 1271–1290, Jun. 2020.

[101] Y. Zhou, Y. Wang, F. Poiesi, Q. Qin, and Y. Wan, "Loop closure detection using local 3D deep descriptors," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6335–6342, Jul. 2022.

[102] J. Straub, N. Bhandari, J. J. Leonard, and J. W. Fisher, "Real-time Manhattan world rotation estimation in 3D," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 1913–1920.

[103] H. Li, J. Yao, J.-C. Bazin, X. Lu, Y. Xing, and K. Liu, "A monocular SLAM system leveraging structural regularity in Manhattan world," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2518–2525.

[104] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-SLAM: Low-drift monocular SLAM in indoor environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6583–6590, Oct. 2020.

[105] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari, "RGB-D SLAM with structural regularities," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11581–11587.

[106] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 5, pp. 571–583, May 2015.

[107] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1465–1479, Sep. 2006.

[108] E. Eade and T. Drummond, "Unified loop closing and recovery for real time monocular SLAM," in *Proc. BMVC*, vol. 13, 2008, p. 136.

[109] B. Williams, G. Klein, and I. Reid, "Automatic relocalization and loop closing for real-time monocular SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1699–1712, Sep. 2011.

[110] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2930–2937.

[111] J. Martinez-Carranza, A. Calway, and W. Mayol-Cuevas, "Enhancing 6D visual relocalisation with depth cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 899–906.

[112] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. D. Stefano, and P. H. S. Torr, "On-the-fly adaptation of regression forests for online camera relocalisation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 218–227.

[113] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, V. A. Prisacariu, L. D. Stefano, and P. H. S. Torr, "Real-time RGB-D camera pose estimation in novel scenes using a relocalisation cascade," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2465–2477, Oct. 2020.

[114] 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.

[115] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, pp. 155–166, 2009.

[116] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, Aug. 1996, pp. 303–312.

[117] H. Roth and M. Vona, "Moving volume kinectfusion," in *Proc. BMVC*, 2012, pp. 1–11.

[118] G. M. Hunter, *Efficient Computation and Data Structures for Graphics*. Princeton, NJ, USA: Princeton Univ. Press, 1978.

[119] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "A memory-efficient kinectfusion using octree," in *Proc. CVM*, 2012, pp. 234–241.

[120] F. Steinbrucker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 2021–2028.

[121] V. A. Prisacariu, O. Kähler, M. M. Cheng, C. Y. Ren, J. Valentin, P. H. S. Torr, I. D. Reid, and D. W. Murray, "A framework for the volumetric integration of depth images," 2014, *arXiv:1410.0925*.

[122] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3D reconstruction with loop closure," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 500–516.

[123] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level SLAM," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 32–41.

[124] Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Real-time progressive 3D semantic segmentation for indoor scenes," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1089–1098.

[125] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 335–342.

[126] J. Xin, K. Du, J. Feng, and M. Shan, "An improved high precision 3D semantic mapping of indoor scenes from RGB-D images," *Comput. Model. Eng. Sci.*, vol. 137, no. 3, pp. 2621–2640, 2023.

[127] 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)*, May 2017, pp. 4628–4635.

[128] M. Mihajlovic, S. Weder, M. Pollefeys, and M. R. Oswald, "DeepSurfels: Learning online appearance fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14519–14530.

[129] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner, "Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3133–3141.

[130] T. Schöps, T. Sattler, and M. Pollefeys, "SurfelMeshing: Online surfel-based mesh reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2494–2507, Oct. 2020.

[131] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised NeRF: Fewer views and faster training for free," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12872–12881.

[132] C. Jiang, H. Zhang, P. Liu, Z. Yu, H. Cheng, B. Zhou, and S. Shen, "H2-mapping: Real-time dense mapping using hierarchical hybrid representation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6787–6794, Jun. 2023.

[133] Y. Mao, X. Yu, K. Wang, Y. Wang, R. Xiong, and Y. Liao, "NGEL-SLAM: Neural implicit representation-based global consistent low-latency SLAM system," 2023, *arXiv:2311.09525*.

[134] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan, "Interactive ray tracing for isosurface rendering," in *Proc. Visualizat.*, Oct. 1998, pp. 233–238.

[135] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[136] J. Bloomenthal, "Polygonization of implicit surfaces," *Comput. Aided Geometric Design*, vol. 5, no. 4, pp. 341–355, Nov. 1988.

[137] R. Westermann, L. Kobbelt, and T. Ertl, "Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces," *Vis. Comput.*, vol. 15, no. 2, pp. 100–111, Apr. 1999.

[138] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of Hermite data," in *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 2002, pp. 339–346.

[139] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Proc. 12th Pacific Conf. Comput. Graph. Appl.*, Oct. 2004, pp. 70–76.

[140] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe, "Unconstrained isosurface extraction on arbitrary octrees," in *Proc. Symp. Geometry Process.*, 2007, pp. 125–133.

[141] Z. Chen and H. Zhang, "Neural marching cubes," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–15, Dec. 2021.

[142] M. Vetsch, S. Lombardi, M. Pollefeys, and M. R. Oswald, "Neuralmeshing: Differentiable meshing of implicit neural representations," in *Proc. DAGM German Conf. Pattern Recognit.*, 2022, pp. 317–333.

[143] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5845–5854.

[144] 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.

[145] H. Zhang, Z. Fang, and G. Yang, "RGB-D simultaneous localization and mapping based on combination of static point and line features in dynamic environments," *J. Electron. Imag.*, vol. 27, no. 5, p. 1, Sep. 2018.

[146] 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.

[147] G. Xu, Z. Yu, G. Xing, X. Zhang, and F. Pan, "Visual odometry algorithm based on geometric prior for dynamic environments," *Int. J. Adv. Manuf. Technol.*, vol. 122, no. 1, pp. 235–242, Sep. 2022.

[148] J. Liu, X. Li, Y. Liu, and H. Chen, "RGB-D inertial odometry for a resource-restricted robot in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9573–9580, Oct. 2022.

[149] L. Cui and C. Ma, "SOF-SLAM: A semantic visual SLAM for dynamic environments," *IEEE Access*, vol. 7, pp. 166528–166539, 2019.

[150] S. Han and Z. Xi, "Dynamic scene semantics SLAM based on semantic segmentation," *IEEE Access*, vol. 8, pp. 43563–43570, 2020.

[151] X. Long, W. Zhang, and B. Zhao, "PSPNet-SLAM: A semantic SLAM detect dynamic object by pyramid scene parsing network," *IEEE Access*, vol. 8, pp. 214685–214695, 2020.

[152] Y. Ai, T. Rui, M. Lu, L. Fu, S. Liu, and S. Wang, "DDL-SLAM: A robust RGB-D SLAM in dynamic environments combined with deep learning," *IEEE Access*, vol. 8, pp. 162335–162342, 2020.

[153] S. Jia, "LRD-SLAM: A lightweight robust dynamic SLAM method by semantic segmentation network," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–19, Nov. 2022.

[154] L. Chen, Z. Ling, Y. Gao, R. Sun, and S. Jin, "A real-time semantic visual SLAM for dynamic environment based on deep learning and dynamic probabilistic propagation," *Complex Intell. Syst.*, vol. 9, no. 5, pp. 5653–5677, Oct. 2023.

[155] B. Bescos, J. M. Fácil, 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.

[156] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1168–1174.

[157] P. Cong, J. Li, J. Liu, Y. Xiao, and X. Zhang, "SEG-SLAM: Dynamic indoor RGB-D visual SLAM integrating geometric and YOLOv5-based semantic information," *Sensors*, vol. 24, no. 7, p. 2102, Mar. 2024.

[158] M. Li, J. He, G. Jiang, and H. Wang, "DDN-SLAM: Real-time dense dynamic neural implicit SLAM with joint semantic encoding," 2024, *arXiv:2401.01545*.

[159] D.-H. Kim and J.-H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1565–1573, Dec. 2016.

[160] J. Cheng, H. Zhang, and M. Q.-H. Meng, "Improving visual localization accuracy in dynamic environments based on dynamic region removal," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1585–1596, Jul. 2020.

[161] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239, doi: 10.1109/CVPR.2017.660.

[162] Z.-J. Du, S.-S. Huang, T.-J. Mu, Q. Zhao, R. R. Martin, and K. Xu, "Accurate dynamic SLAM using CRF-based long-term consistency," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 4, pp. 1745–1757, Apr. 2022.

[163] T. Ji, C. Wang, and L. Xie, "Towards real-time semantic RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11175–11181.

[164] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, Oct. 2017.

[165] Y. Liu, Y. Wu, and W. Pan, "Dynamic RGB-D SLAM based on static probability and observation number," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, 2021.

[166] N. Muller, Y.-S. Wong, N. J. Mitra, A. Dai, and M. Nießner, "Seeing behind objects for 3D multi-object tracking in RGB-D sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6067–6076.

[167] C. Wang, B. Luo, Y. Zhang, Q. Zhao, L. Yin, W. Wang, X. Su, Y. Wang, and C. Li, "DymSLAM: 4D dynamic scene reconstruction based on geometrical motion segmentation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 550–557, Apr. 2021.

[168] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "AirDOS: Dynamic SLAM benefits from articulated objects," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 8047–8053.

[169] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-fusion: Octree-based object-level multi-instance dynamic SLAM," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5231–5237.

[170] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6494–6504.

[171] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto, "TSDF++: A multi-object formulation for dynamic object tracking and reconstruction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 14192–14198.

[172] 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.-3DV*, Jun. 2013, pp. 1–8.

[173] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2018, pp. 10–20.

[174] W. Gao and R. Tedrake, "SurfelWarp: Efficient non-volumetric single view dynamic reconstruction," 2019, *arXiv:1904.13073*.

[175] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense surfel mapping," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6919–6925.

[176] B. Canovas, M. Rombaut, A. Nègre, D. Pellerin, and S. Olympieff, "Speed and memory efficient dense RGB-D SLAM in dynamic scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 4996–5001.

[177] B. Canovas, M. Rombaut, A. Negre, S. Olympieff, and D. Pellerin, "A coarse and relevant 3D representation for fast and lightweight RGB-D mapping," in *Proc. 14th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2019, pp. 824–831.

[178] W. Xian, J.-B. Huang, J. Kopf, and C. Kim, "Space-time neural irradiance fields for free-viewpoint video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9416–9426.

[179] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12939–12950.

[180] B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, and M. O'Toole, "Torf: Time-of-flight radiance fields for dynamic scene view synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 26289–26301.

[181] J. Fang, T. Yi, X. Wang, L. Xie, X. Zhang, W. Liu, M. Nießner, and Q. Tian, "Fast dynamic radiance fields with time-aware neural voxels," in *Proc. SIGGRAPH Asia Conf. Papers*, Nov. 2022, pp. 1–9.

[182] H. Cai, W. Feng, X. Feng, Y. Wang, and J. Zhang, "Neural surface reconstruction of dynamic scenes with monocular rgb-d camera," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 967–981.

[183] K. Vo, T.-T. Pham, K. Yamazaki, M. Tran, and N. Le, "DNA: Deformable neural articulations network for template-free dynamic 3D human reconstruction from monocular RGB-D video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2023, pp. 3675–3684.

[184] Y.-L. Liu, C. Gao, A. Meuleman, H.-Y. Tseng, A. Saraf, C. Kim, Y.-Y. Chuang, J. Kopf, and J.-B. Huang, "Robust dynamic radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 13–23.

[185] B. G. Gerats, J. M. Wolterink, and I. A. Broeders, "Dynamic depth-supervised nerf for multi-view RGB-D operating room videos," in *Proc. Int. Workshop Predictive Intell. Medicine*, 2023, pp. 218–230.

[186] 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.

[187] 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)*, May 2014, pp. 1524–1531.

[188] O. Wasenmuller, M. Meyer, and D. Stricker, "CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–7.

[189] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 7576, 2012, pp. 746–760.
[190] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1625–1632.
[191] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1817–1824.
[192] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with aNNotations," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 92–101.
[193] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?" in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2697–2706.
[194] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[195] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
[196] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[197] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7855–7862.
[198] A. Božic, M. Zollhöfer, C. Theobalt, and M. Nießner, "DeepDeform: Learning non-rigid RGB-D reconstruction with semi-supervised data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7000–7010.
[199] T. Zhang and Y. Nakamura, "HRPSlam: A benchmark for RGB-D dynamic SLAM and humanoid vision," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Feb. 2019, pp. 110–116.
[200] M. N. Finean, L. Petrovic, W. Merkt, I. Markovic, and I. Havoutis, "Motion planning in dynamic environments using context-aware human trajectory prediction," *Robot. Auto. Syst.*, vol. 166, Aug. 2023, Art. no. 104450.
[201] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 4, no. 4, p. 629, 1987.
[202] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," 2016, *arXiv:1607.02555*.
[203] Y.-P. Cao, L. Kobbelt, and S.-M. Hu, "Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras," *ACM Trans. Graph.*, vol. 37, no. 5, pp. 1–16, Oct. 2018.
[204] T. Schöps, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle adjusted direct RGB-D SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 134–144.
[205] C. Zhang, T. Huang, R. Zhang, and X. Yi, "PLD-SLAM: A new RGB-D SLAM method with point and line features for indoor dynamic scene," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 3, p. 163, Mar. 2021.
[206] W. Xie, P. X. Liu, and M. Zheng, "Moving object segmentation and detection for robust RGBD-SLAM in dynamic environments," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.
[207] K. Wang, X. Yao, N. Ma, and X. Jing, "Real-time motion removal based on point correlations for RGB-D SLAM in indoor dynamic environments," *Neural Comput. Appl.*, vol. 35, pp. 8707–8722, Oct. 2022.
[208] X. Yu, W. Zheng, and L. Ou, "CPR-SLAM: RGB-D SLAM in dynamic environment using sub-point cloud correlations," *Robotica*, vol. 1, pp. 1–21, May 2024.
[209] Y. Fan, Q. Zhang, S. Liu, Y. Tang, X. Jing, J. Yao, and H. Han, "Semantic SLAM with more accurate point cloud map in dynamic environments," *IEEE Access*, vol. 8, pp. 112237–112252, 2020.
[210] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-NeRF: Point-based neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5428–5438.

JINLONG ZHU received the Ph.D. degree from Jilin University of Computing Science and Technology in 2016. He worked as a Visiting Scholar with Nanyang Technological University in 2024. He is now an Associate Professor with Changchun Normal University. His research interests include artificial intelligence and computer vision.

CHANGBO GAO received the bachelor's degree from Xuzhou University of Technology, in 2021. He is currently pursuing the master's degree in software engineering with the School of Computer Science and Technology, Changchun Normal University. His research interests include 3D reconstruction, image processing, and 3D rendering.

QIUCHENG SUN received the bachelor's degree in mathematics science from Jilin University, in 2003, and the Ph.D. degree in mechanical engineering from the College of Mechanical Science and Engineering, Jilin University, in 2010. He is currently a Professor with Changchun Normal University, China. His current research interests include camera calibration, image processing, and 3D vision measurement.

MINGZE WANG received the B.Eng. degree, in 2019. He is currently pursuing the master's degree in software engineering with Changchun Normal University. His main research interests include structured light, visual measurement, and 3D reconstruction.

ZHENGKAI DENG received the bachelor's degree from Harbin Normal University, in 2021. He is currently pursuing the master's degree in software engineering with the School of Computer Science and Technology, Changchun Normal University. His research interests include 3D reconstruction and 3D rendering in computer vision.