sensors

MDPI

Article

A Dense Mapping Algorithm Based on Spatiotemporal Consistency

Ning Liu [1,2,3], Chuangding Li [2], Gao Wang [1,2], Zibin Wu [2] and Deping Li [1,3,*]

[1] Robotics Intelligence Technology Research Institute, Jinan University, 601 Huangpu Avenue West, Guangzhou 510632, China
[2] College of Information Science and Technology, Jinan University, 601 Huangpu Avenue West, Guangzhou 510632, China
[3] School of Intelligent Systems Science and Engineering, Jinan University, 206 Qianshan Road, Zhuhai 519070, China
* Correspondence: lideping@jnu.edu.cn

Abstract: Dense mapping is an important part of mobile robot navigation and environmental understanding. Aiming to address the problem that Dense Surfel Mapping relies on the input of a common-view relationship, we propose a local map extraction strategy based on spatiotemporal consistency. The local map is extracted through the inter-frame pose observability and temporal continuity. To reduce the blurring of map fusion caused by the different viewing angles, a normal constraint is added to the map fusion and weight initialization. To achieve continuous and stable time efficiency, we dynamically adjust the parameters of superpixel extraction. The experimental results on the ICL-NUIM and KITTI datasets show that the partial reconstruction accuracy is improved by approximately 27–43%. In addition, the system achieves a greater than 15 Hz real-time performance using only CPU computation, which is improved by approximately 13%.

Keywords: dense mapping; local map extraction; spatiotemporal consistency; point cloud fusion

Citation: Liu, N.; Li, C.; Wang, G.; Wu, Z.; Li, D. A Dense Mapping Algorithm Based on Spatiotemporal Consistency. *Sensors* **2023**, *23*, 1876. https://doi.org/10.3390/s23041876

Academic Editors: Alfred Stein and Adrian Munteanu

Received: 2 November 2022
Revised: 30 January 2023
Accepted: 6 February 2023
Published: 7 February 2023

1. Introduction

Simultaneous Localization and Mapping (SLAM) [1] is a critical technology. It is important for mobile robots to be able to locate and construct maps in unfamiliar environments autonomously. A mobile robot's map reconstruction ability plays a crucial role in recognizing its 3D environment, navigating safely, and completing tasks [2].

Existing mature SLAM frameworks mainly include keyframe-based and mapping-based reconstruction methods. The former is more flexible in management, and the latter can achieve higher precision. Keyframe-based frameworks focus on localization. These frameworks have become mainstream because the positioning algorithm they employ can achieve real-time requirements. However, the map obtained by directly overlaying point clouds is usually not sufficiently accurate. Mapping-based frameworks, on the other hand, take accurate maps as the main goal and basically require a GPU for acceleration. The research direction of real-time 3D reconstruction is developing towards the reconstruction of large-scale scenes. However, there are still bottlenecks in terms of reconstruction accuracy, real-time performance, and adaptability to the environment. These bottlenecks are due to the physical characteristics of RGB-D sensors and the limitations of computing resources. In 2017, Wang et al. proposed that a usable reconstructed map for mobile robot applications should satisfy the following: (1) The map can densely cover the environment to provide sufficient environmental information for the robot; (2) The system has good scalability; (3) The system has good global consistency; (4) The system can fuse different sensors and depth maps of different quality. To meet the above requirements, Dense Surfel Mapping [3] was proposed. The algorithm is based on the surfel model, which extracts superpixels [4] from the depth and intensity images to model the surfel and applies depth images of different qualities.

The resulting map achieves global consistency thanks to the fast map deformation [3]. Most importantly, the algorithm can work in real time with only CPU computation.

However, Dense Surfel Mapping has the following problems: (1) The lack of general-purpose ability of local map extraction: the extraction relies on the covisibility graph of ORB-SLAM2 [5], and pose estimation algorithms without covisibility graphs can only extract based information based on time series. Thus, we extract the local map based on the pose relationship between frames. This eliminates the dependence on the covisibility graph of the input, and it makes the input simpler and the system more versatile. (2) Simple weighted average fusion may lead to inaccuracy in the surfels with a better viewing angle. We add normal constraints to the surfel weight's initialization. Surfels with better view angles will be initialized with greater weights. For surfels with large differences from normal, we only keep the one with the better viewing angle instead of using weighted average fusion. This improves the reconstruction accuracy. (3) The superpixel extraction traverses the entire image. It is unnecessary to handle the regions with invalid depth or beyond the maximum mapping distance, so we filter out the invalid regions before performing superpixel extraction, and we dynamically adjust the parameters of the superpixel extraction based on spatial continuity and temporal stability. Thanks to the dynamic superpixel extraction, the time efficiency of the system are further improved.

In summary, the main contributions of this paper are the following.

- We propose a local map extraction and fusion strategy based on spatiotemporal consistency. The local map is extracted through the inter-frame pose observability and temporal continuity. This eliminates the dependence on the common-view relationship of the pose estimation algorithm and is suitable for various pose estimation algorithms.
- A dynamic superpixel extraction. We dynamically adjust the parameters of superpixel extraction based on spatial continuity and temporal stability, achieving continuous and stable time efficiency.
- The normal constraints are added to the surfel weight initialization and fusion so that surfels with better viewing angles are kept during map fusion.
- The experimental results on the ICL-NUIM dataset show that the partial reconstruction accuracy is improved by approximately 27–43%. The experimental results on the KITTI dataset show that the method proposed in this paper is effective. The system achieves a greater than 15Hz real-time performance, which is an improvement of approximately 13%.

2. Related Work

This section mainly introduces the development of dense reconstruction methods and their scalability and efficiency.

With the commercialization of RGB-D sensors such as Kinect[6], a 3D reconstruction based on RGB-D sensors gradually attracted the attention of researchers, steadily developing and maturing. At present, dense mapping methods are mainly divided into voxel-based methods [7–10], surfel-based methods [3,11], and so on. KinectFusion [12] realized real-time 3D reconstruction based on an RGB-D camera for the first time. This system uses the TSDF (truncated signed distance function) [13] model to reconstruct the environment, but it takes a lot of memory to store the voxel grid. ElasticFusion [14] is a rare reconstruction model using the surfel model [15] model, which focuses on the fine construction of the map. ElasticFusion also improves the pose estimation and reconstruction accuracy by continuously optimizing the reconstructed map. However, ElasticFusion is only suitable for small scenes because of the large computation required. BundleFusion [16] achieves detailed local surface detail registration using the sparse-to-dense registration strategy and achieves real-time continuous model updates using the re-integration model update strategy. It is currently one of the best algorithms for dense 3D reconstruction based on an RGB-D camera. In recent years, many researchers have focused on the combination of neural networks and 3D reconstruction techniques. NICE-SLAM [17] used a hierarchical neural implicit encoding to reconstruct large-scale scenes. Guo et al. used neural implicit

representation to model the scene with the Manhattan-world constraint [18]. Azinović et al. effectively incorporated the TSDF model in the NeRF framework [19]. SimpleRecon [20] tried to learn the depth map directly by using an encoder–decoder architecture based on cost volume, and it introduced metadata into the cost volume to provide more prior knowledge for model training. BNV-Fusion [21] proposed a bi-level fusion algorithm to achieve superior performance. The above reconstruction algorithms need GPU acceleration to achieve good real-time performance because of the huge amount of calculation required. Wang et al. proposed a novel mapping system named Dense Surfel Mapping [3]. The system can fuse sequential depth maps into a globally consistent model in real time without GPU acceleration. Because of the novel superpixel model, the system is suitable for room-scale and urban-scale environments.

The scalability of the voxel-based method is general. It requires a lot of memory to store voxels, so the voxel-based method is, therefore, not suitable for large-scale scenarios, such as KinectFusion [12]. Kintinuous [7] uses a cyclical buffer to improve the scalability of the mapping system. Nießner et al. [22] proposed a voxel hashing method that only stores reconstructed sparse surfaces. This method greatly improves the model's scalability. Compared with voxel-based methods, surfel-based methods are more scalable. This is because surfel-based systems only store reconstructed surface point clouds. Dense Surfel Mapping and [23] further improve scalability by maintaining local maps. Dense Surfel Mapping [3] extracts local maps according to the common-view relationship provided by the ORB-SLAM2. Similar to Dense Surfel Mapping, we use a more general local map extraction method to improve the scalability. The method eliminates the model's dependence on the input. It is extracted through the inter-frame pose observability and temporal continuity. It is more versatile and can be compatible with various pose estimation algorithms.

Runtime efficiency is an essential indicator of the mapping algorithm. Different algorithms offer unique methods to improve runtime efficiency. Voxblox [9], based on voxels, proposes grouped raycasting: each point is projected to a voxel, all points in the same voxel are averaged, and only one raycasting process is performed to speed up fusion. FIESTA uses Indexing Data Structures and Doubly Linked Lists for map maintenance [10]. The efficient data structures and BFS framework of FIESTA allow the system to update as few nodes as possible. Steinbrucker et al. [24] represent scenes using an octree, which is an efficient way to store 3D surfaces. FlashFusion [25] filters out invalid chunks using valid chunk selection; that is, only the chunks in the frustum of the camera view are considered. This highly efficient method allows the algorithm to render at 25 Hz. Dense Surfel Mapping [3] uses superpixels to extract surfels quickly. A local map is maintained to reuse the existing surfels and reduce the memory burden. We further filter out the invalid regions of the image and dynamically adjust the parameters of the superpixel extraction. Thanks to the dynamic superpixel extraction method, our system achieves better time efficiency.

3. System Overview

As shown in Figure 1, the system framework is mainly divided into five parts.

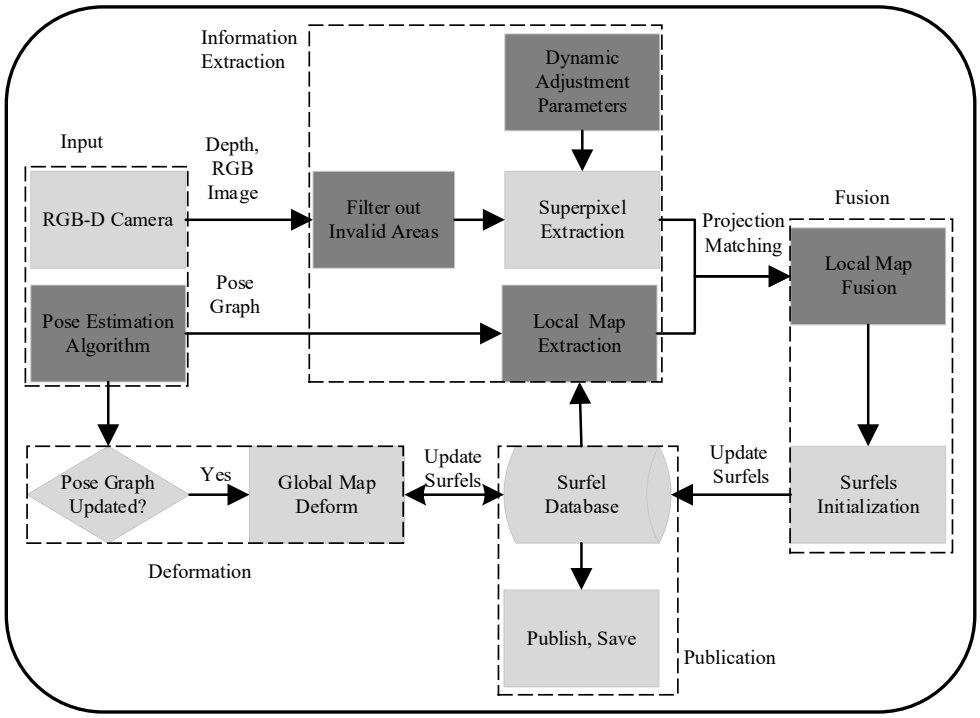


Figure 1. System framework. The system is mainly divided into five parts, as shown by the dotted boxes.

3.1. System Input

The system input is mainly divided into two parts: one is the depth and RGB image obtained by the RGB-D sensor, and the other is the pose graph obtained by the pose estimation algorithms (e.g., ORB-SLAM series [5,26,27], VINS-Mono [28], VINS-Fusion [29]). The pose graph in [3] is similar to the covisibility graph of ORBB-SLAM2. It includes the path and the common-view relationships of the keyframes because it needs the covisibility graph to extract the local map. The input of a pose graph is complex, so it cannot be widely used in various pose graph inputs. Different from [3], the pose graph used in this paper is just the path of keyframes or the ordinary frames. It is simpler and more generic for pose estimation algorithms, and the constraints are relatively loose.

3.2. Global Consistency Deformation

Same as [3], if the input pose graph is updated, the previous poses are optimized. The map is quickly deformed according to the pose difference between the current pose graph and the database. Surfels attached to frame F are deformed according to the matrix $T_2\ T_1^{-1}$, where $T_1 \in \mathbb{R}^{4\times4}$ is the pose of the frame F in the database and $T_2 \in \mathbb{R}^{4\times4}$ is the pose of the frame F in the current pose graph. Then, T_1 is replaced by T_2 and stored in the database. The pose is a homogeneous transformation matrix that includes a rotation matrix and a translation vector.

3.3. Superpixel and Local Map Extraction

In [3], superpixels are extracted by a k-means approach adapted from the extended SLIC [30]. Pixels are clustered [31] according to their intensity, depth, and pixel location. Finally, a down-sampled superpixels image is obtained. The superpixel extraction in [3] traverses the entire image. It is unnecessary to handle the regions with invalid depth or beyond the maximum mapping distance. So, as shown in Figure 1, we first filter out the invalid regions before the superpixel's extraction. Meanwhile, we dynamically adjust the parameters of the superpixel extraction based on spatial continuity and temporal stability. This allows the system to achieve better time efficiency. More details are described in Section 4.2. The local map extraction in [3] is based on the covisibility graph of the input.

Keyframes with the number of minimum edges to the current keyframe below G_δ are locally consistent. Surfels attached to these keyframes are extracted as the local map [3]. To make the system more versatile, we simplify the input in Section 3.1, and we propose a spatiotemporal consistent local map extraction strategy that is simple and effective. We extract the local map based on the pose relationship between frames and continuity in time. More details are described in Section 4.1.

3.4. Map Fusion

In this part, extracted surfels in the local map are fused with extracted surfels in the current frame. The work of [3] transforms the local surfels into the current frame. A weighted average is used to fuse the transformed surfel, and the surfel is extracted in the current frame with a similar depth and normals. However, simple weighted average fusion may lead to inaccurate surfels with better viewing angles. We, thus, add the normal constraints to the surfel weight initialization so that a surfel with a better view angle will be initialized with a greater weight. For surfels with a large difference in normals, we directly keep the one with the better viewing angle instead of performing weighted average fusion. This improves the accuracy of the surfels. And more details are described in Section 4.3.

3.5. Map Publication

In this part, the publication is an independent thread. We retrieve the reconstructed map from the database regularly and publish it as a ROS topic. The topic can be subscribed to for use in later applications, such as navigation and planning.

4. Methods and Principles

4.1. Spatiotemporally Consistent Local Map Extraction

Reconstructing large-scale environments may generate millions of surfels. To reduce map growth, local maps are extracted to reuse and fuse previous surfels and redundant surfels. In this paper, we extract the relevant common-view frames as a local map based on the pose relationship between the two frames. As shown in Figure 2, the pose relationship between two frames is mainly divided into three cases.

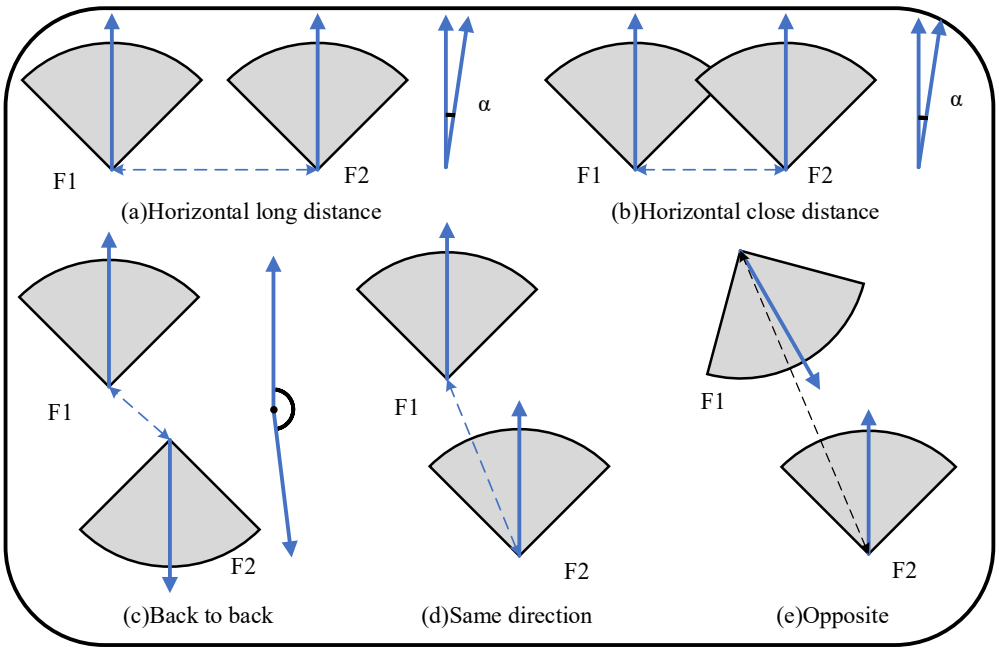


Figure 2. Inter-frame pose relationship. The gray sectors represent the view of the camera, and the arrows in the sectors are their directions.

4.1.1. In the Same Direction Horizontally

As shown in Figure 2a,b, two frames (F_1 and F_2) are nearly parallel. The distance between the two frames is calculated as:

$$D = \| p_1 - p_2 \| , \tag{1}$$

where $p_1 \in \mathbb{R}^3$ and $p_2 \in \mathbb{R}^3$ are the 3D coordinates of frames F_1 and F_2. The cosine of the angle between the two frames' directions is determined as:

$$\cos \alpha = \frac{n_1 \cdot n_2}{\| n_1 \| \| n_2 \|} , \tag{2}$$

where $n_1 \in \mathbb{R}^3$ and $n_2 \in \mathbb{R}^3$ are the direction vectors of frames F_1 and F_2, respectively. The constraints should satisfy: (1) the distance D between two frames is less than the maximum mapping distance $k \cdot far\_dist$, where k is the scale factor, and (2) their angle α is less than the camera's field of view (FOV), denoted as θ_{th}. There is a common area between two frames only when constraints (1) and (2) are both satisfied.

4.1.2. In the Same Direction or Opposite

As shown in Figure 2d,e, frames F_1 and F_2 are in forward or opposite motion. The coordinates of F_1 are projected to the coordinate system of F_2, and the pixel coordinates are calculated as follows:

$$[{p_{1\_F_2}}^T, 1]^T = T_{wF_2}^{-1} [{p_{1\_w}}^T, 1]^T , \tag{3}$$

$$[u_1, v_1, 1]_{F_2}^T = K p_{1\_F_2} , \tag{4}$$

where $T_{wF_2} \in \mathbb{R}^{4\times4}$ is the pose matrix of the frame F_2 in global coordinates, $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic matrix, and $p_{1\_w} \in \mathbb{R}^3$ is the 3D global coordinate of the frame F_1. Similarly, the coordinates of F_2 are projected to the coordinate system of F_1 and the pixel coordinates are calculated as follows:

$$[{p_{2\_F_2}}^T, 1]^T = T_{wF_1}^{-1} [{p_{2\_w}}^T, 1]^T , \tag{5}$$

$$[u_2, v_2, 1]_{F_1}^T = K p_{2\_F_1} , \tag{6}$$

where $T_{wF_1} \in \mathbb{R}^{4\times4}$ is the pose matrix of the frame F_1 in global coordinates, $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic matrix, and $p_{2\_w} \in \mathbb{R}^3$ is the 3D global coordinate of the frame F_2. F_1 's pixel coordinates $[u_1, v_1]_{F_2}^T \in \mathbb{R}^2$ are in the valid coordinate range of the image ($V^{2\times1} \in \mathbb{R}^2$). This means that u_1 is between 0 and the image's width, while v_1 is between 0 and the image's height. The depth $p_{1\_F_2}|_z$ is less than the maximum mapping distance $k \cdot far\_dist$. F_1 and F_2 are considered to have a common-view area when the above two conditions are satisfied, and it is the same for F_2. Surfels attached to this frame can be used as local map frames.

4.1.3. Back to Back

As shown in Figure 2c, the directions of frames F_1 and F_2 are almost opposite. There is no overlap in the fields of view between them. The projection of each frame is not within the other's field of view, and the direction angle is greater than θ_{th}. In general, this case does not satisfy Sections 4.1.1 and 4.1.2 at the same time. In this case, the two frames have no common area and cannot be used as local map frames.

4.1.4. Summary

In summary, the current frame F_j and extracted frames F_i should satisfy:

$$\begin{cases} \| p_i - p_j \| < k \cdot far\_dist \\ \cos^{-1} \frac{n_i \cdot n_j}{\|n_i\| \|n_j\|} < \theta_{th} \\ or \\ (K\ (T_{wF_i}^{-1} [{p_{j\_w}}^T, 1]^T)\ |_{3\times1})\ |_{2\times1} \in V^{2\times1} \\ (T_{wF_i}^{-1} [{p_{j\_w}}^T, 1]^T)\ |_z < k \cdot far\_dist \\ or \\ (K\ (T_{wF_j}^{-1} [{p_{i\_w}}^T, 1]^T)\ |_{3\times1})\ |_{2\times1} \in V^{2\times1} \\ (T_{wF_j}^{-1} [{p_{i\_w}}^T, 1]^T)\ |_z < k \cdot far\_dist\ , \end{cases} \tag{7}$$

where $V^{2\times1} \in \mathbb{R}^2$ is the valid coordinate range of the image. To further enhance the temporal continuity of the local map, frames that are continuous in time are also extracted. For a value of F_i that satisfies the above constraints, $2n$ frames in the time-series $\{F_{i-n}, F_{i-n+1}, \cdots, F_{i-1}, F_{i+1}, \cdots, F_{i+n-1}, F_{i+n}\}$ are continuously extracted as the local map at the same time.

The complete algorithm is shown in Algorithm 1.

Algorithm 1 Local Map Extraction.

Input: j is the index of the current frame. T_{wF_j} is the pose of the current frame. *poseDatabase* is the pose database that stores the poses of each frame and their surfels. far_dist is the maximum mapping distance.

Output: *localIndexes* is a vector of the local frame indexes. *localSurfels* is a vector of the local surfels.

localIndexes.CLEAR()
localSurfels.CLEAR()
for each $F_i \in$ *poseDatabase* **do**
  $flag \leftarrow false$
  $T_{F_jF_i} \leftarrow$ transform(T_{wF_i}, T_{wF_j})
  $[u, v] \leftarrow$ project($T_{F_jF_i}$)
  if isValidRange($[u, v]$) && $T_{F_jF_i}\ |_z \leq k \cdot far\_dist$ **then**
    $flag \leftarrow true$
  end if
  $T_{F_iF_j} \leftarrow$ transform(T_{wF_j}, T_{wF_i})
  $[u, v] \leftarrow$ project($T_{F_iF_j}$)
  if isValidRange($[u, v]$) && $T_{F_iF_j}\ |_z \leq k \cdot far\_dist$ **then**
    $flag \leftarrow true$
  end if
  if distance(T_{wF_i}, T_{wF_j}) $\leq k \cdot far\_dist$ && angle(T_{wF_i}, T_{wF_j}) $\leq \theta_{th}$ **then**
    $flag \leftarrow true$
  end if
  if $flag$ **then**
    for $t \leftarrow -n, -n+1 ... n-1, n$ **do**
      localIndexes.PUSH($i + t$)
    end for
  end if
end for
for each $i \in$ *localIndexes* **do**
  localSurfel.INSERT(*poseDatabase*[i].surfels)
end for

4.2. Dynamic Superpixel Extraction

Reconstructing large-scale scenes puts a large burden on memory. Superpixels can solve this problem well. Similar to [3], the superpixels are extracted from the intensity and depth images.

The cluster center is described as $C_i = [x_i, y_i, d_i, c_i, r_i]^T$, where $[x_i, y_i]^T$ is the average location of clustered pixels and $d_i \in \mathbb{R}^+$, $c_i \in \mathbb{R}^+$, and $r_i \in \mathbb{R}^+$ are the average depth, intensity value, and the radius of the superpixel, respectively. Each pixel u is assigned to a cluster center according to the distance D between itself and its neighborhood cluster center C_i as follows:

$$D = \frac{(x_i - \mathbf{u}_x)^2 + (y_i - \mathbf{u}_y)^2}{N_s^2} + \frac{(c_i - \mathbf{u}_i)^2}{N_c^2} + \frac{(1/d_i - 1/\mathbf{u}_d)^2}{N_d^2}, \tag{8}$$

where $[u_x, u_y, u_d, u_i]^T$ are the location, depth, and intensity of pixel u. N_s, N_c, and N_d are used for normalization. This is the same as in [3].

To enhance the time efficiency of the superpixel extraction, we only handle the depth-valid pixels in the assignment. The superpixel size sp_size and the maximum mapping distance far_dist are the main parameters that affect the time efficiency. We periodically resize the superpixels in time-series frames with the high common-view area:

$$sp\_size_{i+1} = \begin{cases} SP\_SIZE, & e_{acc} \geq FAR\_DIST \text{ or } e_{rot} \geq c_1 \\ c_2 \cdot SP\_SIZE, & sp\_size_i \geq c_2 \cdot SP\_SIZE \\ sp\_size_i + 1, & others \end{cases}, \tag{9}$$

where SP_SIZE and FAR_DIST are the basic superpixel size and maximum mapping distance, c_1 is the rotation difference threshold (default is 0.1), c_2 is the scale constant, and $e\_rot \in \mathbb{R}^+$ and $e\_acc \in \mathbb{R}^+$ are the rotation errors and the accumulated pose errors, respectively, between two consecutive frames. The maximum mapping distance far_dist is dynamically adjusted according to the real-time efficiency as follows:

$$far\_dist_{i+1} = \begin{cases} c_3 \cdot far\_dist_i, & k \leq -c_4 \\ far\_dist_i / c_3, & k \geq c_4 \\ far\_dist_i, & others \end{cases}, \tag{10}$$

where c_3 (default is 1.1) is the scale factor, c_4 (default is 3) is a positive integer. k means that the time cost of consecutive $\mid k \mid$ frames is lower than the average time cost when k is negative and the time cost of consecutive $\mid k \mid$ frames is higher than the average time cost when k is positive.

4.3. Projection Matching and Optimal Observation Normal Map Fusion

There will be a large number of redundant surfels between the surfels generated by the current frame and the local map because of the similar poses. The same surfels observed in different orientations of the frame should be fused to reduce map growth. In this paper, the same surfels are matched by projection and then culled and fused according to their position and normal constraints.

Different from the surfel in [3], the surfel in this paper is $S = [S_p, S_n, S_c, S_w, S_i, S_t, S_v, S_r]^T$, where $S_p \in \mathbb{R}^3$ is the global coordinate, $S_n \in \mathbb{R}^3$ is the unit normal, $S_c \in \mathbb{R}^+$ is the color vector, $S_w \in \mathbb{R}^+$ is the weight coefficient, $S_i \in \mathbb{N}$ is the frame number to which it belongs, $S_t \in \mathbb{N}$ is the number of updates, $S_v \in \mathbb{R}^+$ is the observation cosine in frame S_i, and S_r is the radius of the surfel. An observation cosine is added for the screening of better observation surfels. Project the surfel S^j in the local map to the coordinate system of the current frame F_i:

$$[S^j_{p\_i}{}^T, 1]^T = T^{-1}_{wi}[S^j_{p\_w}{}^T, 1]^T, \tag{11}$$

$$S^j_{n\_i} = R^{-1}_{wi} S^j_{n\_w}, \tag{12}$$

where $S^j_{p\_i} \in \mathbb{R}^3$ and $S^j_{n\_i} \in \mathbb{R}^3$ are the 3D coordinates and normals of S^j in the coordinate system of the current frame, and $T_{wi} \in \mathbb{R}^{4\times4}$ is the pose matrix of the current frame F_i. $R_{wi} \in \mathbb{R}^{3\times3}$ is the rotation matrix of F_i.

As shown in Figure 3a, the red squares are surfels generated by the current frame, and the dots are surfels of the local map. Surfels can be divided into three categories based on the relationship between surfels of the local map and the newly generated ones:

1. Outlier surfels, such as the blue dots in Figure 3a, whose projections are not within the field of view of the current frame:

$$[u_j, v_j]^T = (KS^j_{p\_i})\ |_{2\times1} \notin V^{2\times1}, \tag{13}$$

where $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic matrix, $V^{2\times1} \in \mathbb{R}^2$ is the valid coordinate range of the image, or the projection depth is much larger than the depth of the corresponding surfel S^i in the current frame:

$$S^j_{p\_i}\ |_z - S^i_{p\_i}\ |_z > th, \tag{14}$$

$$th = \min\left\{ min\_th, \frac{\left(S^j_{p\_i}\ |_z\right)^2 \cdot \sigma}{b \cdot f \cdot k \cdot S^j_v} \right\}, \tag{15}$$

where th is the depth difference threshold of outliers, set to $0.5m$ in the first culling and calculated by the Formula (15) in secondary culling, min_th is the minimum threshold constant, b is the baseline of the camera, f is the focal length of the camera, σ is the parallax standard deviation, k is the scale factor of the observed cosine (default is 1.5). The equation shows that there will be a larger tolerance threshold with the farther distance and the larger viewing angle. Thus, the farther surfels are considered more lenient for fusion. Surfels that satisfy condition (13) or do not satisfy condition (14) are not considered for fusion.

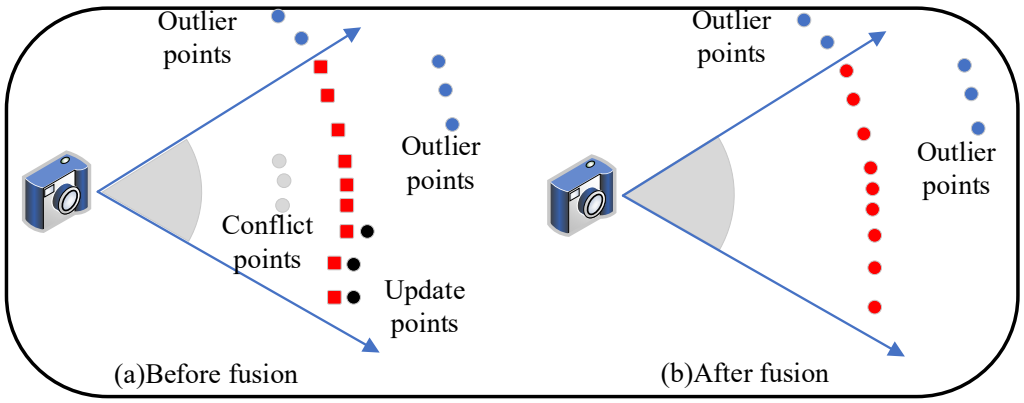


Figure 3. Data association. New generated surfels (red squares) are fused with surfels (dots) of the local map.

2. Conflict surfels, such as the gray dots in Figure 3a, satisfy $[u_j, v_j]^T \in V^{2\times1}$. If its depth difference is less than $-th$, then these surfels are considered to be conflicting and need to be replaced.

3. Update surfels, such as the black dots in Figure 3a, satisfy $[u_j, v_j]^T \notin V^{2\times1}$ after projection, and its depth difference is within $\pm th$. These surfels are considered to be similar to the corresponding newly generated surfels and must be fused and updated to reduce map growth. After the projection constraint, a normal constraint is applied to the matching local map surfel S^j and the newly generated S^i:

$$S_n^j \cdot S_n^i > v_{th}\,, \tag{16}$$

where v_{th} defaults to 0.9. If the matching surfels are not satisfied with the constraint from (16), a strategy based on the best view angle is applied to reserve better surfels. As shown in Figure 4, *pose*1 and *pose*2 observe the same superpixel sp. Compared with *pose*2, which is easily affected by reflection and inaccurate depth, *pose*1 observes it in a better view. Because there is a smaller angle with the normal, *pose*1 obtains a high-quality depth and normal that better describes the superpixel.

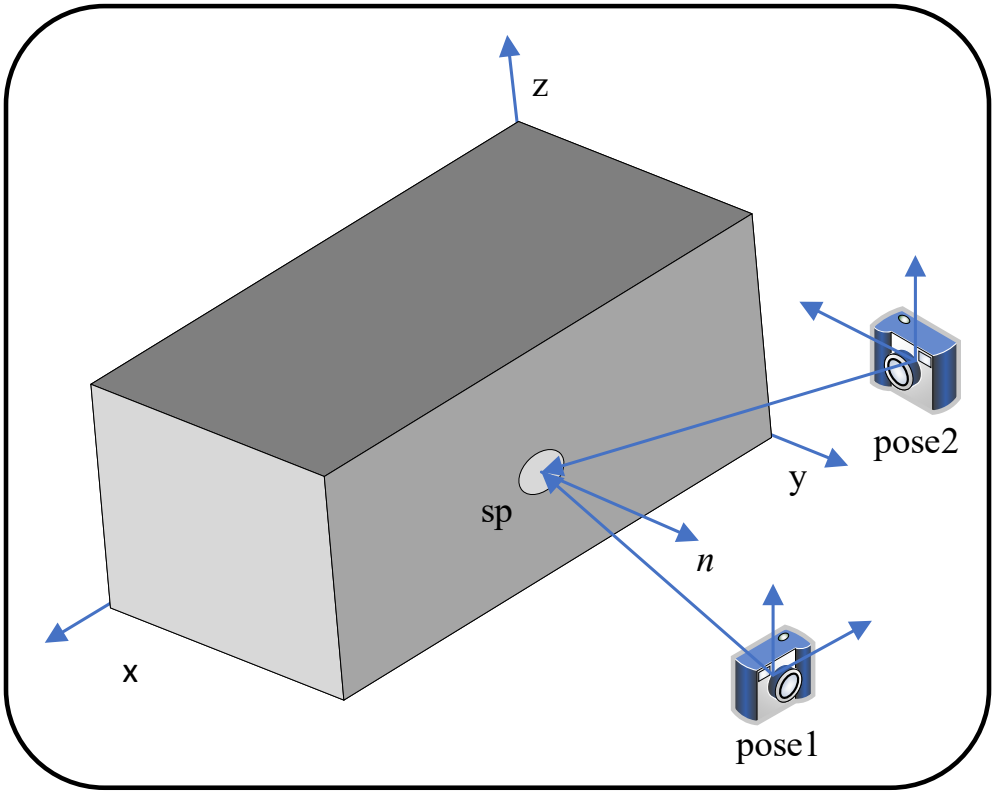


Figure 4. Best viewing angle. sp is a superpixel, and n is its normal. *Pose*1 observes sp in a better view because of its smaller angle with the normal.

In summary, the results of surfel fusion are shown in Figure 3b. The weighted average fusion with normal constraints of the matching surfels is as follows:

$$
\begin{aligned}
S_p^j &\leftarrow \begin{cases} \frac{S_w^j \cdot S_p^j + S_w^i \cdot S_p^i}{S_w^j + S_w^i}, & S_n^j \cdot S_n^i \geq v_{th} \\ S_p^j, & S_n^j \cdot S_n^i < v_{th},\ S_v^j > S_v^i \\ S_p^i, & S_n^j \cdot S_n^i < v_{th},\ S_v^j \leq S_v^i \end{cases} \\
S_n^j &\leftarrow \begin{cases} \frac{S_w^j \cdot S_n^j + S_w^i \cdot S_n^i}{S_w^j + S_w^i}, & S_n^j \cdot S_n^i \geq v_{th} \\ S_n^j, & S_n^j \cdot S_n^i < v_{th},\ S_v^j > S_v^i \\ S_n^i, & S_n^j \cdot S_n^i < v_{th},\ S_v^j \leq S_v^i \end{cases} \\
S_r^j &\leftarrow \begin{cases} \min(S_r^i, S_r^j), & S_n^j \cdot S_n^i \geq v_{th} \\ S_r^j, & S_n^j \cdot S_n^i < v_{th},\ S_v^j > S_v^i \\ S_r^i, & S_n^j \cdot S_n^i < v_{th},\ S_v^j \leq S_v^i \end{cases} \\
S_w^j &\leftarrow \begin{cases} S_w^j + S_w^i, & S_n^j \cdot S_n^i \geq v_{th} \\ S_w^j, & S_n^j \cdot S_n^i < v_{th},\ S_v^j > S_v^i \\ S_w^i, & S_n^j \cdot S_n^i < v_{th},\ S_v^j \leq S_v^i \end{cases} \\
S_c^j &\leftarrow \begin{cases} S_c^j, & S_v^j > S_v^i \\ S_c^i, & others \end{cases} \\
S_v^j &\leftarrow \max(S_v^i, S_v^j) \\
S_t^j &\leftarrow S_t^j + 1 \\
S_i^j &\leftarrow S_i^i
\end{aligned}
\quad . \tag{17}
$$

Considering the inaccuracy caused by distant observations and oblique observations, the weight coefficient of initialization S_w is related to the depth and observation cosine:

$$
S_w = \min\left(1, \frac{k \cdot S_v}{d}\right), \tag{18}
$$

where d is the depth of S in the current camera coordinate system. Because our input of the pose graph is loose, only the paths of the keyframes or ordinary frames are needed. For ordinary frame reconstruction, especially in large-scale scenes, the rate of pose estimation is high. Surfels whose last update was 15 frames ago and which have been updated less than five times are considered outliers and will be removed. Of course, this is not suitable for reconstruction with a low pose estimation rate input.

5. Experiments

This section mainly evaluates the algorithm through public datasets. The algorithm's accuracy is evaluated using the ICL-NUIM dataset [32] and compared with other state-of-the-art algorithms such as Dense Surfel Mapping [3], ElasticFusion [14], BundleFusion [16], and FlashFusion [25]. The local consistency and the time efficiency in large-scale environments are evaluated using the KITTI odometry dataset [33].

The platform used to evaluate our method is a four-core, 4G memory Ubuntu18.04 system configured by VMware under an AMD Ryzen5 4600H. To maintain the same conditions as the comparison method, we also use ORB-SLAM2 in RGB-D mode to track the camera motion and provide the pose graph.

5.1. ICL-NUIM Reconstruction Accuracy

The ICL-NUIM [32] dataset is a synthetic virtual dataset provided by Imperial College London and the National University of Ireland. It is designed to evaluate RGB-D, visual odometry, and SLAM algorithms and is compatible with the TUM dataset. The dataset mainly includes two scenes: a living room and an office room. In addition to the ground truth, the living room scene also has a 3D surface ground truth [32]. It is perfectly suited not just for benchmarking camera trajectories but also reconstruction. To simulate real-world data, the dataset adds noise to both RGB images and depth images.

This experiment uses the living room scene with noise to evaluate the reconstruction accuracy of the algorithm. The input image resolution is 640 × 480. A superpixel size of $SP\_SIZE = 4$, $FAR\_DIST = 3m$ is used for surfel fusion. The mean error of the reconstruction results is calculated using the CloudCompare tool:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} \| p_i - \hat{p}_i \|, \tag{19}$$

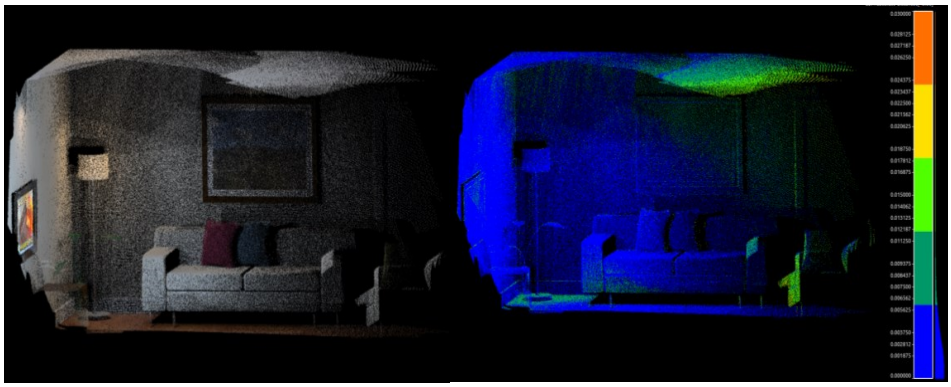
where p_i is the 3D coordinate of the reconstructed point cloud, and $\hat{p}_i$ is the closest true value of the 3D surface to p_i. The experimental results are compared with algorithms such as Dense Surfel Mapping [3], ElasticFusion [14], BundleFusion [16], and FlashFusion [25].

The reconstruction map and the corresponding error heat map are shown in Figure 5. The accuracy evaluation results are shown in Table 1.

Table 1. Reconstruction accuracy on ICL-NUIM (cm).

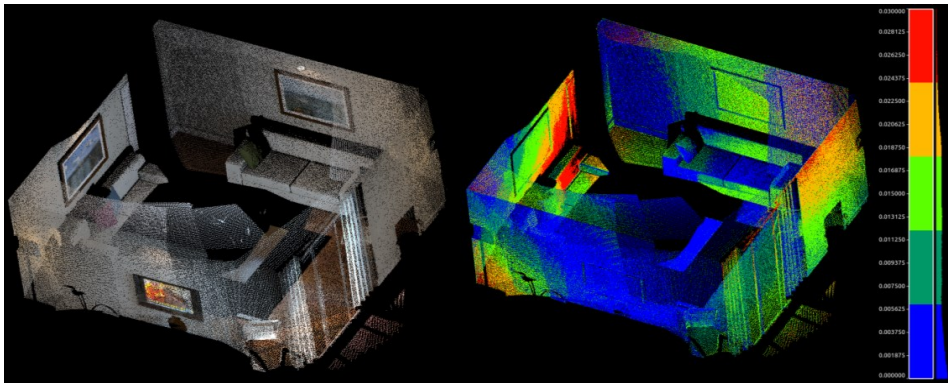
Methods	*kt*0	*kt*1	*kt*2	*kt*3
ElasticFusion	0.7	0.7	0.8	2.8
BundleFusion	0.5	0.6	0.7	0.8
FlashFusion	0.8	0.8	1.0	1.3
Dense Surfel Mapping	0.7	0.9	1.1	0.8
Ours	0.4	1.0	0.8	0.8

Among the algorithms in Table 1, both ElasticFusion and BundleFusion require GPU acceleration. FlashFusion and Dense Surfel Mapping can be directly run in real-time under the CPU. Based on Dense Surfel Mapping, our method can also be run in real-time without GPU acceleration. In terms of reconstruction accuracy, our accuracy of *kt*0 reaches 0.4 cm, which is slightly higher than the 0.5 cm of BundleFusion, and the accuracy of *kt*3 also reaches 0.8 cm, which is the same as BundleFusion. Compared with Dense Surfel Mapping, the accuracy of our method is slightly higher in *kt*0 and *kt*2, the same in *kt*3, and slightly worse in *kt*1.

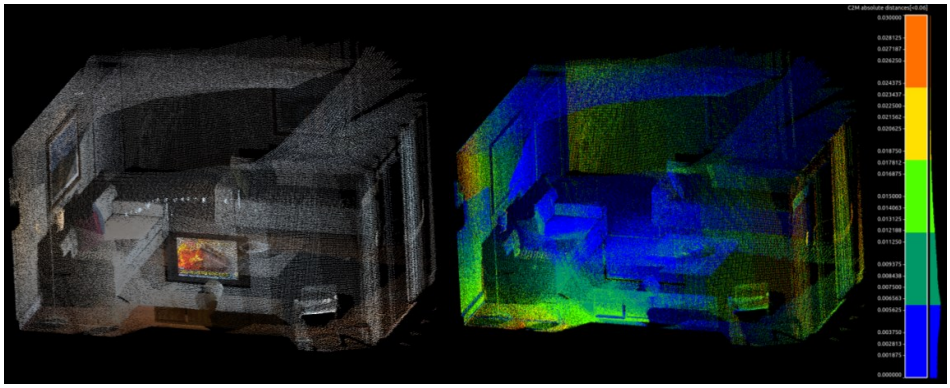


(**a**) *kt*0

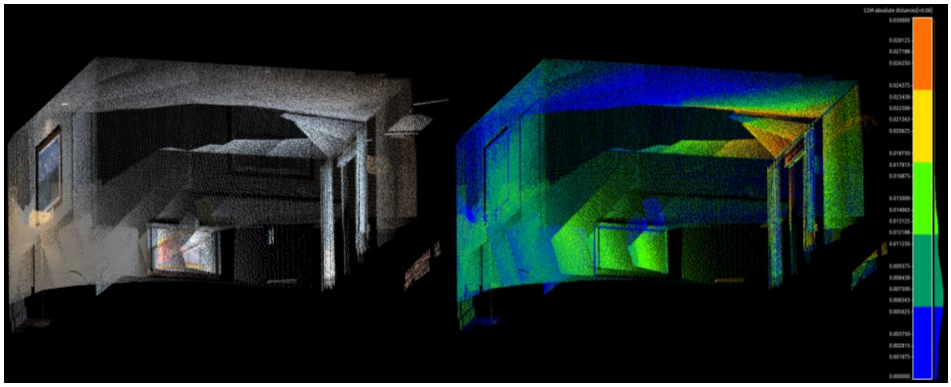
Figure 5. *Cont.*



(**b**) *kt*1



(**c**) *kt*2



(**d**) *kt*3

Figure 5. Reconstruction results of the ICL-NUIM dataset. The left shows the reconstructed point clouds. The right shows the error heatmap, in which the red part represents a 3 cm error and the blue part represents a 0 cm error.

As shown in Figure 5, the reconstruction point clouds of sofas and murals are clear, and even the text in them is faintly visible. It can be seen from the heat map that the main errors are concentrated within 1 cm. There are some errors around 1 cm of *kt*1, mainly on the walls on both sides of the z-axis. These are mainly caused by the inaccurate pose estimation of ORB-SLAM2. There is always a unidirectional deviation of 2 cm–3 cm between the estimated pose and the ground truth in the y-axis and z-axis. This also reflects the side that the algorithm has a certain tolerance for error in pose estimation. It can be seen in Figure 5a that the error of the walls is small. This is because the walls of *kt*0 have been reconstructed from the front. This is a wonderful perspective for observing the object. According to the strategy presented in Section 4.3, these surfels will have a great weight in the fusion, and even surfels reconstructed from the front will directly replace surfels in

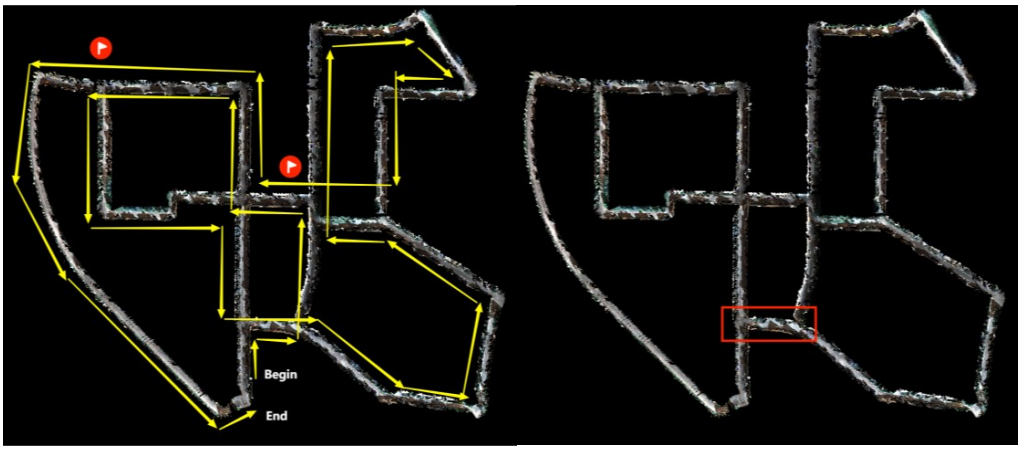
the local map instead of weighted average fusion. This is also the case in *kt*2. This also explains why the accuracies of *kt*0 and *kt*2 are improved in Table 1.

5.2. Kitti Reconstruction Efficiency

This section mainly shows the method's reconstruction performance in large-scale environments. The KITTI dataset is a computer vision algorithm evaluation dataset created by the Karlsruhe Institute of Technology (KIT) and the Toyota University of Technology at Chicago (TTIC) for autonomous driving scenarios. The dataset mainly contains large outdoor scenes such as urban areas, villages, and highways. The KITTI odometry used in this section mainly consists of 22 binocular sequences, 11 of which (00–10) have real trajectories. Here we only use the sequence 00.

The classic PSMNet [34] depth prediction neural network is used to predict depth images using binocular images. This is because the KITTI odometry does not provide depth images. To verify the spatiotemporally consistency of the local map extraction and fusion method proposed in this paper, we use the ground truth trajectories provided by the dataset directly.

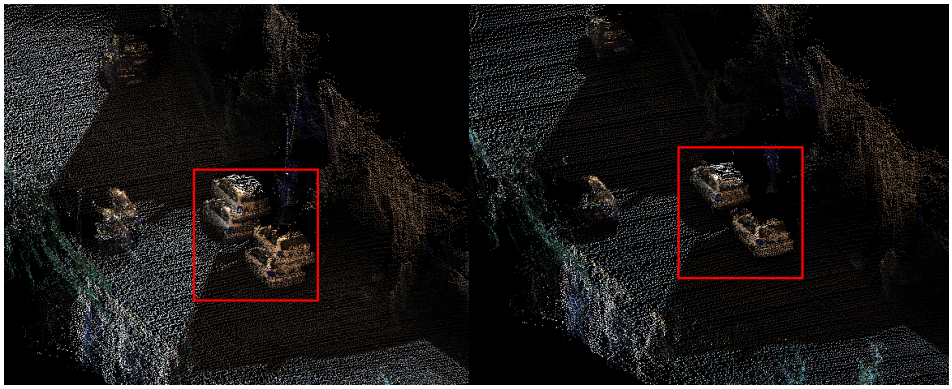
The reconstruction results are shown in Figure 6. The left shows the motion trajectory of the camera, and the right is the map reconstructed by our method in real-time. The reconstructed map covers all the areas that the camera passes through without problems, such as large-scale blurring and disappearance.



(a) The trajectory (b) Reconstruction result

Figure 6. Reconstruction results of KITTI odometry sequence 00. (**a**) Motion trajectory of the camera. (**b**) Point clouds of reconstruction. The area in the red frame of (**b**) is the revisit area.

Figure 7 shows the local detail of the reconstruction selected from the red box area in Figure 6b, which is a revisited area. The left is the result with a local map extraction only based on time series, and the right is the result of our method. The reconstructed cars in the red box on the left appeared misaligned, and the right solves the problem. As can be seen from Figure 6, it takes hundreds of frames to pass through the red box area twice. The left in Figure 7 fails to extract the previous surfels for fusion. The error of the pose when reconstructing two frames leads to ghosting. The right side of the figure extracts the first reconstructed surfels as a local map for fusion so that there is no such problem. It can, thus, be seen that our method of local map extraction and fusion performs well on the consistency of the local map.



(a) (b)

Figure 7. Reconstruction details of the revisited area. (**a**) Reconstruction details with a local map extraction based on time-series. (**b**) Reconstruction details of our method. The red frames show the performance differences between the two methods. Compared with (**b**), the cars of (**a**) are misaligned.

The memory usage of the surfels throughout the runtime is shown in Figure 8. The orange curve is the result of our method without removing outliers. The black one is the result of our method with removing outliers. The blue one is the result of extracting local maps only based on time-series. There is almost no difference in the first 3200 frames because the car was moving to an unknown area in the scene. Between about 3200 and 4000 frames, the memory usage of our method stays almost unchanged because the car revisits the area between two red flags in Figure 6a, but the blue curve is still growing In addition, it can be seen that the memory usages of the black curve and orange curve are quite different. That is because large-scale scenes can easily generate outliers, and the input pose graph rate is high (10 Hz). If the outliers are not removed, the number of reconstructed surfels will greatly increase. Of course, when the rate of the input pose graph is low, the strategy of removing outliers is not advisable, causing the normal surfels to be removed and resulting in an incomplete reconstruction scene.

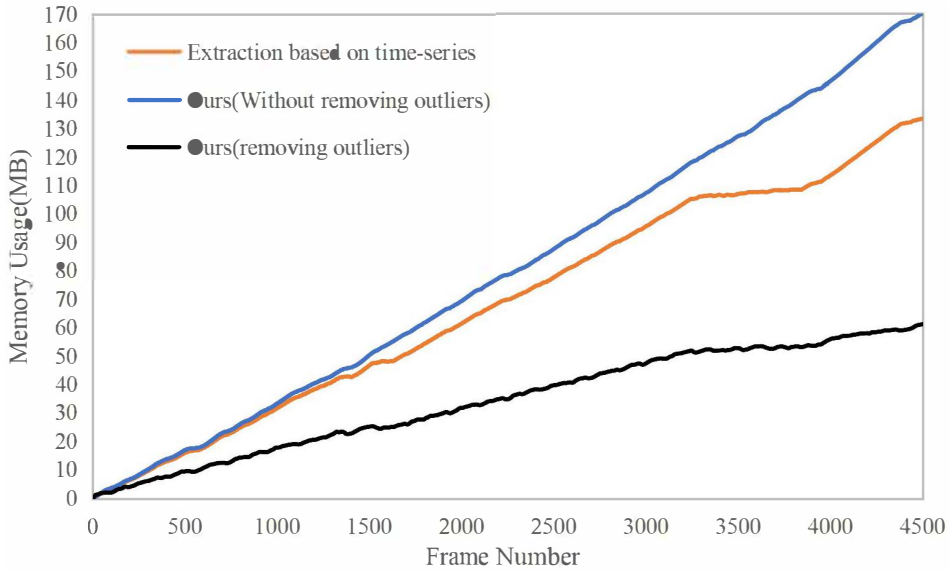


Figure 8. Memory usage of reconstructing KITTI odometry sequence 00. Between about 3200 and 4000 frames, the memory usage of our method stays almost unchanged.

As shown in Figure 9 and Table 2, as the superpixel's size becomes smaller, the average time cost per frame increases. As the maximum mapping distance increases, the average time cost per frame increases, too. This is because we filter the invalid pixels and only handle the valid regions. When $SP\_SIZE = 8$ and $FAR\_DIST = 20$, the average time cost

is around 60 ms per frame, making our system about 15 Hz in real-time. Compared with [3], our time efficiency is improved by approximately 13% under the same conditions (SP_SIZE = 8 and FAR_DIST = 30).

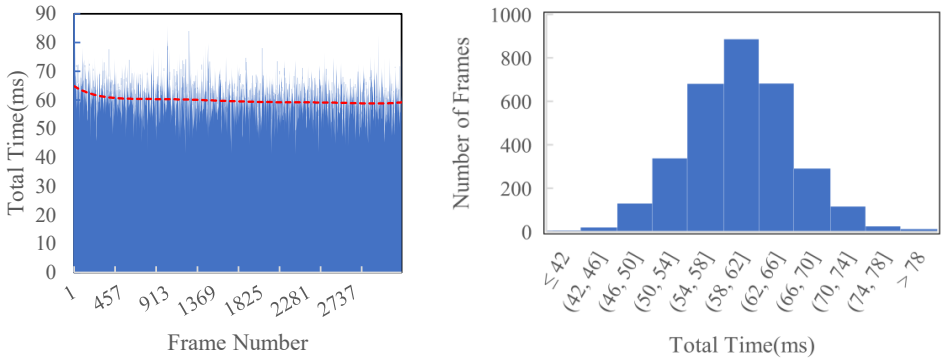


Figure 9. Time efficiency of reconstructing KITTI odometry sequence 00 (SP_SIZE = 8, FAR_DIST = 20 m). The average cost time is around 60 ms per frame.

Table 2. Time efficiency (average).

SP_SIZE	*FAR_DIST* (m)	Generate Superpixels (ms)	Fusion (ms)	Total (ms)
8	10	35.6	1.3	38.8
8	20	56.1	1.4	59.9
8	30	60.5	1.4	64.7
4	10	37.1	1.4	41.2
4	20	60.8	2.0	67.6
4	30	63.4	2.1	70.6
8 [3]	30	≈70.0	≈1.0	≈75.0

To verify the effect of c_3 and c_4 of Formula (10) on time efficiency, we control the values of c_3 and c_4 in our experiments. SP_SIZE = 8 and FAR_DIST = 30 are used in the experiments. The results are shown in Figures 10 and 11 and Table 3. With the increase in c_3 and c_4, large jitters in the running time occasionally appear, which have a larger standard deviation. This is because a larger c_4 will cause a delay in the dynamic adjustment parameters that will not be adjusted in time according to the current running state. A larger c_3 results in a larger change in far_dist, which is not conducive to smooth and stable time efficiency.

Table 3. Time efficiency with different c_3 and c_4.

c_3	c_4	Average Time (ms)	Standard Deviation
1.1	2	66.5	5.3
1.1	3	65.5	5.8
1.1	4	67.4	7.1
1.1	5	68.0	6.4
1.05	3	66.9	5.2
1.15	3	68.2	7.3
1.2	3	71.5	7.3

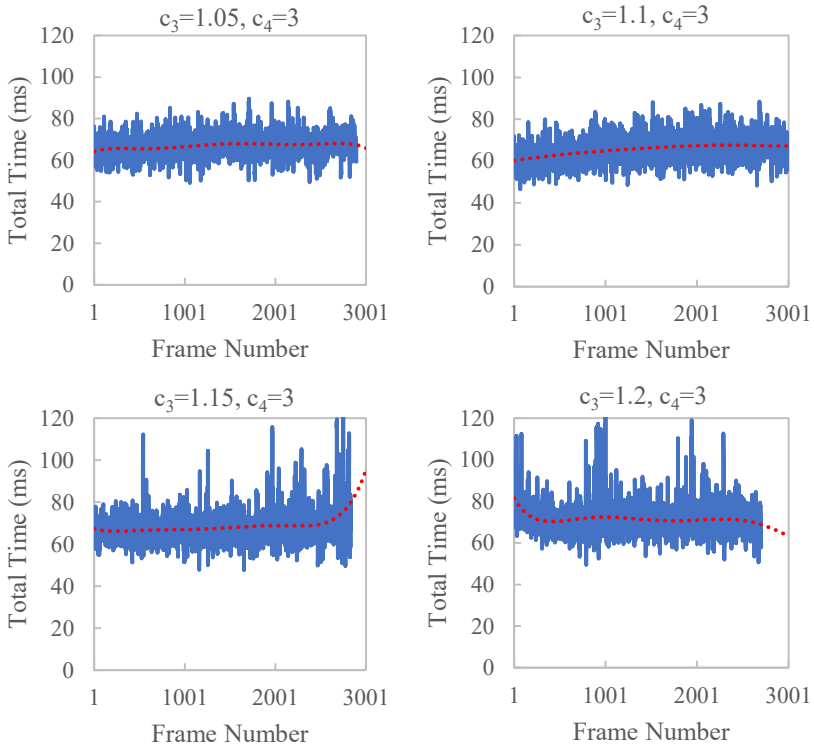


Figure 10. Time efficiency of reconstruction KITTI odometry sequence 00 with different c_3. When c_3 is larger than 1.1, large jitters in the running time occasionally appear.

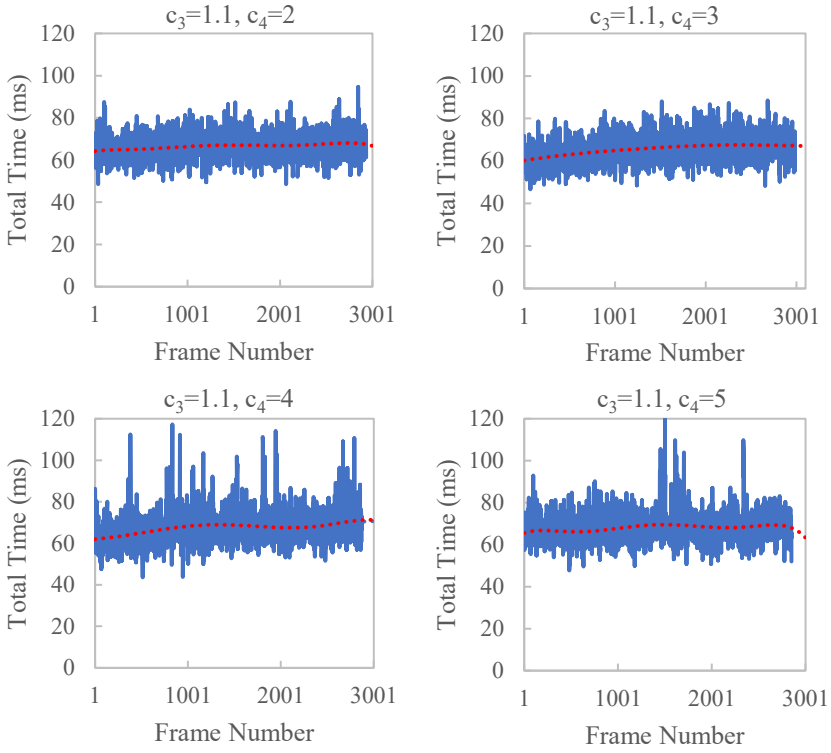


Figure 11. Time efficiency of reconstruction KITTI odometry sequence 00 with different c_4. When c_4 is larger than 3, large jitters in the running time occasionally appear.

6. Conclusions

Aiming to improve the generalization ability of Dense Surfel Mapping, we propose a spatiotemporally consistent local map extraction method. It makes the system widely

applicable to various pose estimation algorithms that only need to provide the path of poses. Meanwhile, the system achieves local accuracy and local consistency. An optimal observation normal fusion strategy is used for better surfels fusion. Compared with [3], the partial reconstruction accuracy of ICL-NUIM is improved by approximately 27–43%. Thanks to the dynamically adjusted superpixel extraction strategy, we achieve a greater than 15 Hz real-time performance. This is 13% higher than [3]. The mapping system is suitable for room-scale and large-scale environments. The local map reuses the previous surfels in space so that the memory usage grows according to the environment's scale instead of the runtime. Adjusting superpixels according to their time cost makes the runtime more stable and efficient. The system achieves a balance between memory usage and time efficiency.

Author Contributions: N.L. conceived the method and designed the experiments. C.L. assisted with the experiment design and analysis of the results analysis and wrote the paper. D.L. researched the literature and contributed to the paper's revision. G.W. and Z.W. reviewed the paper and gave some suggestions. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Nature Science Foundation of China under Grants 62172188 and 62276114, and Zhuhai Industry-University-Research Cooperation Project under Grant ZH22017001210107PWC, and Zhuhai Industrial Core and Key Technology Program under Grant 2220004002352.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Leonard, J.J.; Durrant-Whyte, H.F. Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robot. Autom.* **1991**, *7*, 376–382. [CrossRef]
2. Gao, J.; Li, B. Research on Automatic Navigation System Construction Based on SLAM Algorithm and Deep Neural Network. In Proceedings of the ICASIT 2020: 2020 International Conference on Aviation Safety and Information Technology, Weihai, China, 14–16 October 2020.
3. Wang, K.; Gao, F.; Shen, S. Real-time scalable dense surfel mapping. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6919–6925.
4. Stutz, D.; Hermans, A.; Leibe, B. Superpixels: An evaluation of the state-of-the-art. *Comput. Vis. Image Underst.* **2018**, *166*, 1–27. [CrossRef]
5. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
6. Zhang, Z. Microsoft kinect sensor and its effect. *IEEE Multimed.* **2012**, *19*, 4–10. [CrossRef]
7. Whelan, T.; Kaess, M.; Fallon, M.; Johannsson, H.; Leonard, J.; McDonald, J. Kintinuous: Spatially Extended Kinectfusion. In *Robotics & Autonomous Systems*; MIT Press: Amsterdam, The Netherlands, 2012.
8. Amanatides, J.; Woo, A.; A fast voxel traversal algorithm for ray tracing. In Proceedings of the Eurographics, Amsterdam, The Netherlands, 24–28 August 1987; Volume 87, pp. 3–10.
9. Oleynikova, H.; Taylor, Z.; Fehr, M.; Nieto, J.; Siegwart, R. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1366–1373. [CrossRef]
10. Han, L.; Gao, F.; Zhou, B.; Shen, S. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 4423–4430.
11. Schöps, T.; Sattler, T.; Pollefeys, M. Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *42*, 2494–2507. [CrossRef] [PubMed]
12. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A.J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. Kinectfusion: Real-time dense surface mapping and tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011; pp. 127–136.
13. Curless, B.; Levoy, M. A volumetric method for building complex models from range images. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 4–9 August 1996; pp. 303–312.

14. Whelan, T.; Leutenegger, S.; Salas-Moreno, R.; Glocker, B.; Davison, A. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems*; MIT Press: Cambridge, MA, USA, 2015.
15. Pfister, H.; Zwicker, M.; Van Baar, J.; Surfels, M.G. Surface Elements as Rendering Primitives. In *Computer Graphics, SIGGRAPH 2000 Proceeding*; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 2000; pp. 343–352.
16. Dai, A.; Nießner, M.; Zollhöfer, M.; Izadi, S.; Theobalt, C. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph. ToG* **2017**, *36*, 24. [CrossRef]
17. Zhu, Z.; Peng, S.; Larsson, V.; Xu, W.; Bao, H.; Cui, Z.; Oswald, M.R.; Pollefeys, M. Nice-slam: Neural implicit scalable encoding for slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 12786–12796.
18. Guo, H.; Peng, S.; Lin, H.; Wang, Q.; Zhang, G.; Bao, H.; Zhou, X. Neural 3D Scene Reconstruction with the Manhattan-world Assumption. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 5511–5520.
19. Azinović, D.; Martin-Brualla, R.; Goldman, D.B.; Nießner, M.; Thies, J. Neural RGB-D surface reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 6290–6301.
20. Sayed, M.; Gibson, J.; Watson, J.; Prisacariu, V.; Firman, M.; Godard, C. SimpleRecon: 3D Reconstruction Without 3D Convolutions. In *Proceedings of the European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–19.
21. Li, K.; Tang, Y.; Prisacariu, V.A.; Torr, P.H. BNV-Fusion: Dense 3D Reconstruction using Bi-level Neural Volume Fusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022, pp. 6166–6175.
22. Nießner, M.; Zollhöfer, M.; Izadi, S.; Stamminger, M. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph. ToG* **2013**, *32*, 169. [CrossRef]
23. Fu, X.; Zhu, F.; Wu, Q.; Sun, Y.; Lu, R.; Yang, R. Real-time large-scale dense mapping with surfels. *Sensors* **2018**, *18*, 1493. [CrossRef] [PubMed]
24. Steinbrücker, F.; Sturm, J.; Cremers, D. Volumetric 3D mapping in real-time on a CPU. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 2021–2028.
25. Han, L.; Fang, L. FlashFusion: Real-time Globally Consistent Dense 3D Reconstruction using CPU Computing. In *Robotics: Science and Systems*; MIT Press: Cambridge, MA, USA, 2018; Volume 1, p. 7.
26. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
27. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
28. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
29. Qin, T.; Cao, S.; Pan, J.; Shen, S. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv* **2019**, arXiv:1901.03642.
30. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2274–2282. [CrossRef] [PubMed]
31. Hartigan, J.A.; Wong, M.A. Algorithm AS 136: A k-means clustering algorithm. *J. R. Stat. Soc. Ser. Appl. Stat.* **1979**, *28*, 100–108. [CrossRef]
32. Handa, A.; Whelan, T.; McDonald, J.; Davison, A.J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 1524–1531.
33. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res. IJRR* **2013**, *32*, 1231–1237. [CrossRef]
34. Chang, J.R.; Chen, Y.S. Pyramid stereo matching network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5410–5418.