A Stereo SLAM System with Dense Mapping

Ben Zhang¹, ², Denglin Zhu²

¹College of Mechanical and Electrical, Sanjiang University, Nanjing, China
²College of Mechanical and Electrical, Hohai University, Changzhou, China

Corresponding author: Ben Zhang (e-mail: runeberry@gmail.com).

This paper is supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant No. 18KJD510008).

ABSTRACT The development of simultaneous localization and mapping (SLAM) technology plays an important role in robot navigation and autonomous vehicle innovation. The ORB-SLAM2 is a unified SLAM solution for monocular, binocular, and RGBD cameras which constructs a sparse feature point map for real-time positioning. However, a sparse map based approach cannot effectively meet the requirements of robot navigation, environment reconstruction, and other tasks. In this paper, a dense mapping thread is added to the existing ORB-SLAM2 system. The depth map and color image obtained by the stereo matching of a binocular camera are used to generate a three-dimensional point cloud for keyframes; then, the point cloud is fused by tracking and optimizing the motion track of a feature frame to obtain a real-time point cloud map. Through the experiments conducted on the KITTI dataset and the real environment under the ROS, it is proved that the proposed system constructs a clear three-dimensional point cloud map while constructing an accurate trajectory.

INDEX TERMS Dense mapping; Mobile robot; Point cloud; SLAM; Stereo matching.

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) approaches receive information through a specific sensor without prior information in order to estimate an object’s position and motion as well as establishing an environmental model [1] [2]. The development of SLAM technology has a history of more than 30 years. Recently, advances in computer vision and robotics have supported numerous developments and breakthroughs in SLAM technology; these have made great contributions to robot navigation, three-dimensional (3D) reconstruction, virtual reality, and other fields [3].

Visual SLAM approaches can acquire a large amount of redundant texture information from the environment using cameras, and have a strong capability to recognize scenery. Early visual SLAM approaches are based on filtering theory; their non-linear error models and high computational costs have deterred their practical application. In recent years, with the development of the sparse non-linear optimization theory (e.g., bundle adjustment) and advances in computer technology, real-time visual SLAM approaches are no longer a dream. Mainstream visual SLAM approaches include visual odometry (VO), back end optimization, loop closing, and mapping. VO approaches estimate the motion of successive images. The back end receives the position and pose of the camera measured by VO and loop detection information. These data are optimized to obtain a globally consistent trajectory and map. Loop closing determines whether the robot reaches the previous position, and mapping is based on the trajectory estimation.

VO (front end) estimations for motion using images can be classified into feature-based and direct methods [4]. The feature-based methods detect and match the key points. Then, the camera motion and scene geometry are computed using epipolar geometry. This approach has an important inherent limitation: only feature information can be used in VO and mapping. Thus, the generated map is sparse and cannot be directly used for navigation and obstacle avoidance. In contrast, direct SLAM optimizes the geometry directly from the image’s
intensities. This enables the utilization of all the information in the images [5]. However, it is computationally expensive and vulnerable to external illumination. When the camera moves or rotates in a large scale, the image intensities cannot be tracked very well.

We propose an improved hybrid SLAM method that combines feature-based VO and dense mapping by depth estimation. The approach extracts FAST corners [6] and matches them using the ORB descriptor [7] to estimate the motion between successive frames by 3D-2D and 2D-2D. After inserting and culling keyframes, we estimate the depth map by a weighted guided filtering method for stereo matching on keyframes which focuses on dense mapping. By adopting a multi-threaded structure, the proposed SLAM system can achieve accurate localization and dense mapping results in real-time.

The rest of this paper is organized as follows: Section 2 reviews the mainstream SLAM system and local stereo matching method. Section 3 describes the proposed SLAM system. Experimental results are also carried out to demonstrate the effectiveness of the proposed framework in Section 4. Section 5 concludes this paper.

II. RELATED WORK

In this section, we review the related work with respect to the two fields that we integrate within our framework, i.e. methods for visual SLAM and depth estimation with the use of stereo cameras.

A. VISUAL SLAM METHODS

The research of Visual slam can be divided into three times: classical era, algorithm analysis era and multi-sensor fusion era. In the classical era, the robot pose is estimated mainly based on probability theory, including Extended Kalman Filter, Particle Filter and maximum likelihood estimation.

In the era of algorithm analysis, many new algorithms were born, and their main research includes the observability, convergence and consistency of the algorithm.

Andrew Davison’s MonoSLAM [8] is the first real-time monocular SLAM system. Its back end traces the sparse feature points of the front end based on EKF method, and creates sparse maps online in the framework of probability.

The PTAM [9] is the first multithreaded SLAM algorithm proposed by Georg Klein, which divides tracking and mapping into two separate tasks. These tasks are processed with two parallel threads. The PTAM is the first scheme to use non-linear optimization as the back end rather than a filter. A keyframe mechanism is proposed in which several keyframes are structured together to optimize their trajectories and maps. The LSD-SLAM proposed in 2014 is a large-scale direct monocular SLAM method, which directly processes the pixels of the image [5]. It proposes an image matching algorithm for estimating the similarity transformation and scale perception between keyframes. The approach not only calculates its own position and posture, but also constructs a global semi-dense and accurate environmental map.

In the later SLAM work, a large number of researchers have improved PTAM. ORB-SLAM is a representative successor of PTAM. Mur-Artal has proposed the ORB-SLAM and the ORB-SLAM2 [10] [11] methods. As shown in Figure 1, they adopt a multi-threading idea. The front end uses the ORB feature point extraction; the back end uses a graph optimization method and adds the word bag loop closing. The ORB-SLAM method realizes feature-based sparse SLAM based on a monocular camera, and the ORB-SLAM2 proposes a unified solution for monocular, binocular, and RGB-D cameras. The ORB-SLAM2 is considered to be the most successful SLAM solution at this stage. It achieves high accuracy in real time, but generates a sparse map that does not meet the needs of some robotic applications [11].

LSD-SLAM (Large-scale Direct Monocular SLAM) is a direct visual SLAM system proposed by Engel J et al [12]. It can build a large-scale environment consistency map, has good real-time performance on the CPU, and can realize semi-dense reconstruction of the scene. However, the system is sensitive to light by using the direct method, which is not suitable for fast-moving occasions, and the system does not design the function of dynamic object processing, so it cannot remove moving objects in the environment. In addition, the system is a monocular SLAM system, and its scalability is not strong.

SVO (Semi Direct Monocular Visual Odometry) is a visual odometry method published by Scaramuzza of Zurich University in 2014[13], which is a visual odometry combining feature point method and direct method. The system extracts the image blocks in the image for feature matching, and the camera pose is pre estimate by brightness change. Because the
structure is simplified in the system design, SVO runs very fast and is suitable for occasions with limited computing resources. However, in order to improve the running speed, the system abandons the back-end optimization and loop detection functions, so there will be large errors in a wide range of operation, and it can not deal with moving objects in the environment. At present, its main application platform is UAV.

DSO (Direct Spark Odometry) is a visual odometer method released by Dr. Jakob Engel of the Technical University of Munich in 2016[14]. The system uses the sparse direct method to realize the visual odometer. By calculating the projection residuals from map points to target image frames for pose estimation, it can better deal with the lack of corners or repeated textures in the environment. The scheme runs fast and can generate dense point clouds to realize dense reconstruction. However, because the visual odometry is a direct method, it is sensitive to illumination changes, and the scheme can not deal with moving objects. In addition, there is no loop detection module in the system, so it is impossible to reposition the loop of the scene, and it is easy to produce cumulative errors when running in large-scale scenes.

### TABLE I

| Schemes | Accuracy   | Stability  | Speed   | Loop closing | Scalability |
|---------|------------|------------|---------|--------------|-------------|
| ORB-SLAM2 | Relatively high | Relatively good | Relatively fast | Yes | Relatively strong |
| LSD-SLAM | Relatively high | Relatively good | Relatively fast | Yes | Average |
| SVO     | Average    | Average    | Fast    | No           | Average     |
| DSO     | Relatively high | Relatively good | Relatively fast | No | Average |

The comparison of common visual SLAM schemes is shown in table I.

In the era of multi-sensor fusion, vision sensors are fused with different sensors such as IMU and laser to improve the accuracy of SLAM. The accurate estimation of fast motion by IMU in a short time can make up for the short board of camera recognition of fast moving objects, so as to better deal with the situation of high-speed driving and rotation of moving objects. Lidar can get the 3D point cloud map of the environment in real time. Bloesch et al. [15] proposed a monocular vision-inertial odometer based on direct method and iterative extended Kalman filter, considering photometric error and image block, which improves the robustness and tracking performance of the system. Zhang et al. [16] proposed a coupled SLAM scheme combining Lidar, camera and IMU. After field verification, the scheme has high pose estimation and map accuracy, and realizes a high-precision odometer.

### B. DEPTH MAP ESTIMATION

Depth information plays a key role in 3D reconstruction, navigation, and obstacle avoidance because only when the depth is determined can we know its spatial location exactly. Binocular cameras generally consist of left and right cameras placed horizontally. The center of the two apertures is located on the x-axis, and the distance between them becomes the baseline. Figure 2 shows that $d = u_L - u_R$ is the difference between the left and right coordinates, called the disparity, which is inversely proportional to depth. According to the disparity, the distance between the pixel and the camera can be estimated.
III. PROPOSED METHOD

Our method is mainly based on the ORB-SLAM2 approach that we extend to dense mapping. By using stereo cameras, dense mapping is used to process the keyframes provided by the ORB-SLAM2 system to reconstruct the geometry of the environment. The outline of our method is as follows; it is illustrated in Figure 3:[17]:

1) The tracking component extracts the ORB feature of each stereo image \( I_l \) and \( I_r \) and matches them based on the features. The approach continuously tracks every new camera image (i.e., the current frame \( F_{curr} \)) that is captured by monocular or binocular cameras. It determines whether the current frame is a keyframe \( K_{Fi} \).

2) Each keyframe \( K_{Fi} \) is processed by the local mapping thread. Except for the mappoints decision and local BA, the local mapping culls the redundant keyframes in two methods for loop closing and dense mapping, respectively.

3) The loop closing thread searches for loops and, if a loop is detected, corrects the loop to achieve global consistency.

4) For every keyframe \( K_{Fi} \), the dense mapping thread matches the images pairs again by the dense method and generates the point cloud map. After post-processing the point cloud, it is fused (if a loop is detected in the loop closing thread, the loop correction result is added in the fusion step).

A. TRCKING THREAD

In the front end of the SLAM, feature-based VO matches the ORB feature points extracted from image frames, and then estimates the egomotion of the camera. The ORB feature points are extremely fast to compute and match in addition to having good invariance to the viewpoint [7]. Thus, in accordance with the ORB-SLAM2, we use the ORB feature throughout the tracking thread, local mapping, and loop closing of the whole system.

Feature-based VO methods can be divided into 2D-2D, 3D-2D, and 3D-3D according to the matching feature dimensions involved. These methods optimize the camera orientation \( R \in SO(3) \) and position \( t \in \mathbb{R}^3 \) in order to minimize the reprojection error between consecutive frames [13] [18].

As shown in Figure 3, black, red, and blue points represent the matched map points between a 3D point of \( F_{curr-1} \) and a left image point, right image point, or stereo point of \( F_{curr} \) respectively. As shown in equation 1, the 3D-3D minimization of the reprojection error between the matched 3D points \( X^i \in \mathbb{R}^3 \) of \( F_{curr-1} \) and stereo keypoints (3D) \( X^s_i \in \mathbb{R}^3 \) of \( F_{curr} \). These matches are between the left and right images in the same image pair or between the left images in consecutive frames. \( \pi_i \) is the projection function of 3D-3D, and \( \| X_i^s - \pi_i (RX^i + t) \|_\Sigma \) is the minimize function of 3D points between \( F_{curr-1} \) and \( F_{curr} \) which use the ICP method. Meanwhile, we minimize the reprojection error between 3D points of \( X^i \in \mathbb{R}^3 \) of \( F_{curr-1} \) and 2D points \( x^l_i \in \mathbb{R}^2 \), \( x^r_i \in \mathbb{R}^2 \) of \( F_{curr} \). We use the PnP method to optimize the pose of the camera. We put these three items together and use BA method to optimize the pose of the camera as a whole. Because 3D and 2D points are considered at the same time, the accuracy of BA is improved. In equation 1, After the initial pose estimation by tracking consecutive frames, we use methods in [18] to track the local map and identify the new keyframe.

\[
\{ R, t \} = \arg \min_{R,t} \sum_{i \in \mathcal{X}} r\left( \| X^s_i - \pi_i (RX^i + t) \|_\Sigma \right) + \| X^l_i - \pi_l (RX^l + t) \|_\Sigma + \| X^r_i - \pi_r (RX^r + t) \|_\Sigma \) (1)

FIGURE 3. Proposed SLAM system overview showing the four threads: tracking, local mapping, loop closing, and dense mapping.
B. KEYFRAME CULLING

In the local mapping thread, as new keyframes are inserted the local BA updates and optimizes the map points and the keyframes. In order to maintain a compact sparse reconstruction (i.e. a dense reconstruction), this thread needs to delete redundant keyframes [19] [20] [21]. For sparse map points, reducing unnecessary keyframes can abate the complexity of the loop closing and unnecessary repetitive work. On the other hand, dense mapping without culling increases the number of points that do not contribute to the reconstruction accuracy and increase the storage requirements [22].

We propose a keyframe culling method for the loop closing and dense mapping threads. The dense map thread provides the dense stereo matching of keyframes to obtain a 3D point cloud, and then uses a point cloud fusion method to generate a global point cloud map. Because of the high sampling frequency and the limited motion speed, there is a substantial amount of redundant information between adjacent keyframes. By adopting more strict culling strategy, a dense map can be established quickly and efficiently. In the ORB-SLAM method, all of the keyframes in KF in which 90% of the map points have been seen in at least other three keyframes are discarded. We also use this method to delete the keyframes for the loop closing. At the same time, for dense mapping we use a more stringent method: 1, discard all of the keyframes in KF in which 75% of the map points have been seen in at least other five keyframes; and 2, delete adjacent keyframes whose motion does not reach a certain displacement as shown in equation 2,

\[ \| \Delta t \| + \min (2\pi - \| r \|, \| r \|) < \min_m \]  

(2)

where \( \Delta t \) and \( r \) are the translation and rotation of adjacent keyframes, use their norms to express the motion. The value of the minimum threshold of camera motion, \( \min_m \), is 0.4 throughout the experiment.

C. LOCAL STEREO MATCHING

For obtaining a dense 3D point cloud, we use the weighted guided image filtering (WGIF) method that is based on local stereo matching. This method is accurate and suitable for real-time applications [23]. The WGIF is a novel approach for binocular stereo systems for fast and accurate matching. It consists of four main steps: matching cost computation, cost aggregation, disparity computation, and disparity refinement [24] [25]. The specific calculation process is described in [26].

The matching cost computation algorithms are similarity measurement processes for different disparities. We propose a multi-measure stereo matching algorithm based on the fusion of absolute differences(AD), census transform, and gradient methods.

\[ C_{(p,d)} = \alpha \cdot C_{AD}(p,d) + \beta \cdot C_{CT}(p,d) + \gamma \cdot C_{grad}(p,d) \]  

(3)

where \( \alpha \), \( \beta \), and \( \gamma \) are the weights computed using the AD, census, and gradient algorithms, and \( \alpha + \beta + \gamma = 1 \). \( C_{AD}(p,d) \), \( C_{CT}(p,d) \) and \( C_{grad}(p,d) \) are the similarity calculation based on color information, census change and gradient information respectively.

After a pixel-wise computation is performed, the resulting cost output needs to be filtered by a guided image filter. We use the left image as the guide image \( I \), and the cost image corresponding to each pixel in the original cost volume as the image to be filtered \( p \). For color guidance images \( I = (I_r, I_g, I_b)^T \), the linear model can be written as

\[ q_t = a_t l_t + b_k \]  

(4)

Each slice of the cost volume \( C_{(p,d)} \) is filtered using the guidance image \( I \) (usually \( I = I_r \)) as

\[ W_{ij}^{GF} = \sum_{k} \omega_i \omega_j \frac{1}{|w_i| |w_j|} - \frac{1 + \frac{(I_j - \mu_k)^T (I_j - \mu_k)}{\Sigma_k \gamma(p')}}{\Sigma_k \gamma(p')} \]  

(5)

The regularization parameters are adjusted adaptively using the Canny method, which further reduces the errors. The Canny algorithm is used to detect the edge of the image, and the edge weight is defined as

\[ G_{(p')}(p') = \begin{pmatrix} G_{r}(p') & 0 & 0 \\ 0 & G_{g}(p') & 0 \\ 0 & 0 & G_{b}(p') \end{pmatrix} \]  

(6)

\[ G_{i}(p') = \frac{1}{N} \sum_{p=1}^{N} \frac{C_{i}(p') + \epsilon}{C(p') + \epsilon}, \quad i = r, g, b \]  

(7)

Thus, the conclusion of the cost aggregation can be written as

\[ C_{(p,d)} = W_{ij}^{GF} \cdot C_{(p,d)} \]  

(8)

A disparity map is generated using the winner-take-all strategy (WTA); this map is subsequently refined using a densification method to reduce errors.
D. POINT CLOUD POST-PROCESSING

The number of point clouds directly obtained by a stereo matching disparity map is huge. On the one hand, it increases the calculation and reduces the efficiency of 3D reconstruction; on the other hand, it is not conducive to the smoothness of the reconstructed surface and impacts the effect of the 3D reconstruction. Therefore, after obtaining a dense 3D point cloud, it is necessary to simplify the point cloud [27] [28]. In the reconstruction of a 3D environment based on binocular vision, it is required that not only the simplification speed be addressed but also the characteristics of the 3D environment must be kept, so as to better serve the mobile robot in the environment. Therefore, a fast point cloud reduction algorithm is proposed as shown in Figure 4. Firstly, the spatial topological relationship of a point cloud is established by a k-d tree. Secondly, the point cloud is smoothed and denoised by using a bilateral filtering method. For any point cloud \( p_i \), its unit normal vector is \( n_i \).

\[
p_i' = p_i + \lambda n_i
\]  

Among them, the weighting factor of bilateral filtering \( \lambda \) by

\[
\lambda = \frac{\sum_{p_j \in N(p_i)} W_k(||p_j-p_i||)W_c(||n_i-n_j||)(n_i \cdot p_j-p_i)}{\sum_{p_j \in N(p_i)} W_k(||p_j-p_i||)W_c(||n_i-n_j||)}
\]  

Where \( N(p_i) \) is point \( p_i \)'s neighborhood point set, \( n_j \) is the corresponding normal vector of neighborhood point \( p_j \). \( \sigma_c \) indicates the influence factors on this point of distance between \( p_i \) and \( p_j \); \( \sigma_s \) indicates the influence factors on this point of the projection of \( p_i \) to \( p_j \) distance vector on \( n_i \).

Thirdly, the boundary of the point cloud is judged, and the point cloud boundary is preserved. In this paper, the least square plane is constructed by point \( p \) and neighborhood points, and all points are projected into the plane. The projection point \( p' \) of point \( p \) and the projection point \( q_i' \) of neighborhood point \( q_i \) is connected to get vector set \( \vec{p'q_i'} \), the angle between vector \( \vec{p'q_i'} \) and the reference vector \( \vec{p'q_k} \) is calculated, which is arranged in a vector sequence according to the size. The difference between the adjacent angles in the sequence, that is, the angle between the adjacent vectors is calculated. If the angle of biggest angle is greater than the threshold (120°), the data point \( p \) is considered as the boundary point, otherwise it is not.

Fourthly, the feature threshold of the point cloud is calculated by using a multi-measure fusion approach, and the feature is judged. For any point \( p \) in the space, the local average distance in its k neighborhood is as follows.

\[
d(p) = \frac{1}{k} \sum_{i=1}^{k} |p_i - p|
\]  

And we use principal component analysis to realize the normal vector estimation. The surface variation defined as point \( p \) in k neighborhood is as follows.

\[
\varphi = \frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}
\]  

Local average distance and curvature are important parameters to characterize the geometric characteristics of spatial point clouds. In order to better distinguish the characteristics of point clouds, the feature parameter \( f(p) = \varphi / d(p) \) is defined to describe them comprehensively. \( \varphi \cdot d(p) \) are the curvature and local average distance of space point \( p \) in k neighborhood. For any point in the spatial point cloud, the smaller the local average distance in the neighborhood and the larger the curvature, the more important the point is in the representation of spatial geometric features.

In the fifth step, the feature points are preserved and the non-feature points are classified and simplified by bounding box method [30]. We take \( \overline{f(p)} \) as the threshold to judge whether point \( p \) is a feature point. When \( f(p) > \overline{f(p)} \), this point is a feature point and should be retained, when \( f(p) < \overline{f(p)} \), the points are segmented and simplified. Specifically, the points are divided into 3 intervals by the ratio \( c = (f(p))/\overline{(f(p))}(c \in (0,1)) \), which are \((0,0.3] , (0.3,0.5] \) and \((0.5,1) \). The feature points of the first region are reserved, and the weak feature points of the second region are reduced to a lower degree; For the third region, the non-feature points are reduced to a higher degree.

Finally, after realizing the point cloud map acquisition and post-processing based on each keyframe, we use the camera motion parameters obtained by VO to fuse the point cloud data. The point cloud data are based on the camera coordinate system. As long as the real-time position and posture of the camera in the map is obtained, we can transform the point cloud into the map by

\[
X_w = T_{cvw}X_c
\]
After the 3D point cloud data of the keyframes is registered in the unified world coordinate system, the repeated image area between the keyframes leads to the presence of redundant point cloud information, resulting in the redundancy and inconsistency of the data. Therefore, after data registration, the point cloud data must be fused to eliminate overlapping information and establish a complete 3D point cloud model without redundancy. Our paper sets a relatively small distance threshold to compare the Euclidean distance between each point of the two point clouds after registration. If the distance is less than the threshold, the average value between the two points is used.

IV. IMPLEMENTATION AND EXPERIMENT

The image used in this paper is collected by a binocular camera. The CPU of the platform is i7 running at 2.6 GHz with a GPU (We use GPU to accelerate for stereo matching). All experiments were performed on image sequences of the well-known KITTI dataset [30] and real scenes.

A. KITTI DATASET

Note that the Stereo SLAM KITTI dataset is a challenging benchmark as it contains fast movement. We use the Stereo SLAM with local stereo matching and obtain the depth map for each pair. Our method has been evaluated on the KITTI dataset which contains stereo sequences recorded from a car in city,

| Sequence | Proposed method Mean time (ms) | RMSE (m) | ORB-SLAM2 Mean time (ms) | RMSE (m) | LSD-SLAM(Stereo) Mean time (ms) | RMSE (m) | SVO(Monocular) Mean time (ms) | RMSE (m) | RGBD-SLAM(RGB-D) Mean time (ms) | RMSE (m) |
|----------|-------------------------------|---------|-------------------------|---------|---------------------------------|---------|-----------------------------|---------|-------------------------------|---------|
| 00       | 127.3                         | 1.25    | 117.2                   | 1.26    | 123.5                           | 1.01    | 85.1                        | 1.87    | 87.2                          | 2.21    |
| 01       | 146.2                         | 9.57    | 133.7                   | 9.65    | 141.6                           | 8.90    | 90.3                        | 10.11   | 93.5                          | 12.38   |
| 02       | 128.0                         | 6.11    | 116.6                   | 6.01    | 119.8                           | 2.56    | 85.2                        | 7.31    | 89.6                          | 10.21   |
| 03       | 129.8                         | 0.87    | 117.3                   | 0.90    | 124.5                           | 1.45    | 88.2                        | 1.54    | 90.1                          | 2.01    |
| 04       | 113.6                         | 0.21    | 103.7                   | 0.23    | 101.8                           | 0.25    | 79.7                        | 1.11    | 83.7                          | 1.21    |
| 05       | 118.1                         | 0.78    | 105.6                   | 0.73    | 109.5                           | 1.45    | 80.1                        | 1.81    | 84.9                          | 1.77    |
| 06       | 129.1                         | 0.64    | 118.2                   | 0.66    | 121.6                           | 1.31    | 85.2                        | 1.54    | 93.1                          | 1.43    |
| 07       | 115.2                         | 0.59    | 102.3                   | 0.59    | 107.9                           | 0.48    | 78.1                        | 1.47    | 81.2                          | 1.31    |
| 08       | 116.5                         | 3.41    | 103.6                   | 3.49    | 112.1                           | 3.87    | 78.8                        | 4.98    | 82.7                          | 6.01    |
| 09       | 116.7                         | 2.98    | 103.3                   | 3.01    | 102.8                           | 5.56    | 77.9                        | 4.77    | 81.5                          | 5.34    |
| 10       | 115.1                         | 1.01    | 98.9                    | 1.01    | 101.7                           | 1.51    | 76.2                        | 1.76    | 80.1                          | 1.97    |
As evidenced by Figure 5, we show a challenging trajectory that includes a loop closure of sequence 05 of KITTI, which has a public ground truth. Our system outperforms with a relative error below 0.5% as shown in Figure 5 c). Table II shows the comparison between the proposed method and state-of-the-art in terms of the root mean square error (RMSE) of estimated trajectory and mean time. Among them, LSD-SLAM uses binocular camera, SVO uses monocular camera, RGBD-SLAM uses RGB-D camera, and RGBD-SLAM directly outputs dense map using depth map. As can be seen from Table II, the root mean square error of the proposed method is lower than ORB-SLAM2 as the mean time is longer than ORB-SLAM2 because of the more computation in dense mapping and stereo matching. The mean time is longer than RGBD-SLAM (Dense SLAM) because of the latter can get the depth map directly.

We also evaluate our 3D reconstruction as shown in Figure 6 in order to better represent the results of the 3D reconstruction; we use four different scenes in the database, such as city, road, residential, and campus. Table III shows the point cloud number and storage size of dense map.

**B. REAL SCENE ONLINE EXPERIMENT**

In addition to using KITTI database, we take indoor and outdoor scenes as experimental scenes, and carry out online experiments with a hand-held binocular camera. Myntai binocular camera (standard version S1020) was used to conduct online experiments under ROS system.

As shown in Figure 7, it is the vehicle trajectory obtained on the satellite map by using the dense SLAM algorithm proposed with the car as the mobile carrier in the outdoor campus environment. The red line is the real vehicle trajectory, and the green line is the trajectory obtained by SLAM algorithm. It can be seen from the results that the trajectory of the algorithm itself is basically consistent with the road information of the satellite map and the real trajectory; and in the whole SLAM process, the closed-loop optimization is realized and the accuracy of the overall trajectory is improved. The experiment in indoor environment is similar to that in outdoor environment. Using wheeled mobile robot turnlebot2 as mobile carrier, similar indoor moving trajectory can be obtained.

The environmental cloud point map obtained in the indoor and outdoor environment are shown in Figure 8. It can be seen
FIGURE 6. Dense map of KITTI dataset. a) 0926_0001_city, b) 0926_0015_road, c) 0926_0019_residential, d) 0928_0037_campus.

TABLE III
Point cloud number and storage size of dense map

| Point cloud number | Storage size(MB) |
|--------------------|-----------------|
| City 20630         | 1.9M            |
| Road 58637         | 5.5M            |
| Residential 63715  | 6M              |
| Campus 11769       | 1.1M            |

FIGURE 7. Trajectory of SLAM based on true outdoor scene.
from the results that through the proposed SLAM algorithm, dense indoor and outdoor point cloud map can be obtained by bincoular camera, which provides reliable 3D map data for navigation.

V. CONCLUSIONS

In order to realize visual navigation and 3D reconstruction, a Stereo SLAM system with dense mapping has been introduced. Firstly, the keyframes are obtained on an ORB-SLAM2 architecture; the depth map and dense point cloud corresponding to the keyframes are obtained by binocular matching. Then, the keyframes are culled for loop closing and dense mapping in separate threads. Finally, the dense point cloud is post processed based on the location information to obtain a reasonable dense map which can be used for robot navigation, and realize a real-time dense SLAM method based on a binocular camera. The standard dataset and real scenes are used to experimentally evaluate the proposed approach. The results demonstrate that:

1) The proposed algorithm constructs a clear 3-D point cloud map while constructing an accurate trajectory;
2) Compared with ORB-SLAM2, dense map are built in real-time;

3) The map constructed by the method proposed has high accuracy and can accurately describe the environment of the space. The map can fully meet the needs of mobile robot autonomous positioning and navigation.

References

[1] Thrun S, Burgard W, Fox D. Probabilistic robotics. Cambridge, MIT Press, 2005.
[2] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., & Neira, J., Simultaneous localization and mapping: present, future, and the robust-perception age, IEEE Transactions on Robotics, 32(6) (2016) 1309-1332.
[3] Yasuda YDV, Martins LEG and Cappabianco FAM, Autonomous Visual Navigation for Mobile Robots: A Systematic Literature Review, ACM COMPUTING SURVEYS, 53(1) (2020) 13.
[4] Scaramuzza D, Fraundorfer F, Visual odometry: part I: the first 30 years and fundamentals, IEEE Robotics & Automation Magazine 18(4) (2011) 80-92.
[5] Engel, Jakob, T. Sch, and D. Cremers, LSD-SLAM: large-scale direct monocular SLAM, in ECCV (2014), pp. 834-849.
[6] Herbert Bay,Tinne Tuytelaars,Luc Van Gool, SURF : Slepded-Up Robust Features, in ECCV(2006), pp.404-417.
[7] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, ORB: An efficient alternative to SIFT or SURF, in IEEE International Conference on Computer Vision (ICCV) (2011), pp. 2564-2571.
[8] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, MonoSLAM: Real-Time Single Camera SLAM, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6) (2007) 1052-1067.
[9] G. Klein and D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, in 6th IEEE and ACM International Symposium on Mixed and Augmented Reality(2007), pp. 225-234.
[10] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, ORB-SLAM: A Versatile and Accurate Monocular SLAM System, IEEE Transactions on Robotics, 31(5) (2017) 1147-1163.
[11] R. Mur-Artal and J. D. Tardós, ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras, IEEE Transactions on Robotics, 33(5) (2017) 1255-1262.

[12] Engel, J., T. Schps, and D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, in European Conference on Computer Vision Springer (ECCV) (2014), pp. 834-849.

[13] C. Forster, M. Pizzoli and D. Scaramuzza, SVO: Fast semi-direct monocular visual odometry, in IEEE International Conference on Robotics and Automation (ICRA) (2014), pp. 15-22.

[14] J. Engel, V. Koltun and D. Cremers, Direct Sparse Odometry, IEEE Transactions on Pattern Analysis and Machine Intelligence 40(3) (2018) 611-625.

[15] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, Robust visual inertial odometry using a direct EKF-based approach, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015), pp. 298-304.

[16] Ji Z, Singh S, Laser-visual-inertial Odometry and Mapping with High Robustness and Low Drift. Journal of Field Robotics 35(8) (2018) 1242-1264.

[17] P. Li and Z. Ke, Feature-based SLAM for Dense Mapping, in International Conference on Advanced Mechatronic Systems (ICAMechS) (2019), pp. 372-377.

[18] Y. Zhao and P. A. Vela, Good Feature Matching: Toward Accurate, Robust VO/VSLAM with Low Latency, IEEE Transactions on Robotics, 36(3) (2020) 657-675.

[19] Zhang, WT, Zhai, et al. Research on Visual Positioning of a Roadheader and Construction of an Environment Map, APPLIED SCIENCES-BASEL, 11(11) (2021) 4969.

[20] Yin, Ma, Zhong, Wu, & Huang, SLAM-Based Self-Calibration of a Binocular Stereo Vision Rig in Real-Time, Sensors, 20(3) (2020) 621.

[21] K. Joo, T. -H. Oh, F. Rameau, J. -C. Bazin and I. S. Kweon, Linear RGB-D SLAM for Atlanta World, in IEEE International Conference on Robotics and Automation (ICRA) (2020), pp. 1077-1083.

[22] Yang Y, et al, Multi-camera visual SLAM for off-road navigation, ROBOTICS AND AUTONOMOUS SYSTEMS, 128(2020) 103505.

[23] Zhengguo Li, et al. Weighted Guided Image Filtering. IEEE Transactions on Image Processing 24(1) (2014) 120-129.

[24] J. Li, B. Chen, S. Wu and J. Peng. Stereo Refinement Based on Gradient Domain Guided Filtering, in IEEE Conference on Industrial Electronics and Applications (ICIEA) (2020), pp. 522-526.

[25] J. Li, B. Chen, S. Wu and J. Peng, Stereo Refinement Based on Gradient Domain Guided Filtering, in IEEE Conference on Industrial Electronics and Applications (ICIEA)(2020), pp. 522-526.

[26] Zhang B , Zhu D , Local Stereo Matching: An Adaptive Weighted Guided Image Filtering-Based Approach, International Journal of Pattern Recognition and Artificial Intelligence 35(3) (2021) 2154010.

[27] Y. Fan et al., Semantic SLAM With More Accurate Point Cloud Map in Dynamic Environments, IEEE Access, 8(2020) 112237-112252.

[28] R. B. Rusu and S. Cousins, 3D is here: Point Cloud Library (PCL), in IEEE International Conference on Robotics and Automation (2011), pp. 1-4.

[29] A. Ruchay, K. Dorofeev, V. Kalschikov and A. Kober, Accuracy analysis of surface reconstruction from point clouds, in International Conference on Information Technology and Nanotechnology (ITNT), (2020), pp. 1-4.

[30] The KITTI vision benchmark suite, http://www.cvlibs.net/datasets/kitti/.