Cognitive Computation (2022) 14:1496–1508
https://doi.org/10.1007/s12559-022-10010-w

Visual-LiDAR SLAM Based on Unsupervised Multi-channel Deep Neural Networks

Yi An[1] · Jin Shi[1] · Dongbing Gu[2] · Qiang Liu[3]

Received: 24 May 2021 / Accepted: 6 March 2022 / Published online: 28 April 2022

Abstract

Recently, deep learning techniques have been applied to solve visual or light detection and ranging (LiDAR) simultaneous localization and mapping (SLAM) problems. Supervised deep learning SLAM methods need ground truth data for training, but collecting such data is costly and labour-intensive. Unsupervised training strategies have been adopted by some visual or LiDAR SLAM methods. However, these methods only exploit the potential of single-sensor modalities, which do not take the complementary advantages of LiDAR and visual data. In this paper, we propose a novel unsupervised multi-channel visual-LiDAR SLAM method (MVL-SLAM) which can fuse visual and LiDAR data together. Our SLAM system consists of an unsupervised multi-channel visual-LiDAR odometry (MVLO) component, a deep learning–based loop closure detection component, and a 3D mapping component. The visual-LiDAR odometry component adopts a multi-channel recurrent convolutional neural network (RCNN). Its input consists of front, left, and right view depth images generated from 360° 3D LiDAR data and RGB images. We use the features from a deep convolutional neural network (CNN) for the loop closure detection component. Our SLAM method does not require ground truth data for training and can directly construct environmental 3D maps from the 3D mapping component. Experiments conducted on the KITTI odometry dataset have shown the rotation and translation errors are lower than some of the other unsupervised methods, including UnMono, SfmLearner, DeepSLAM, and UnDeepVO. Experimental results show that our methods have good performance. By fusing visual and LiDAR data, MVL-SLAM has higher accuracy and robustness of the pose estimation compared with other single-modal SLAM systems.

Keywords Unsupervised deep learning · Multi-channel RCNN · Visual-LiDAR SLAM · Sensor fusion

Introduction

Simultaneous localization and mapping (SLAM) plays a key role in many fields, such as autonomous robot navigation, localization [1, 2], and self-driving systems. Traditional SLAM methods [3, 4] use manual design features to carry out inter-frame matching and loop closure detection. However, the parameters of these methods have to be set in various scenarios to achieve better performance. Recently, with the advance of deep learning techniques, significant achievements of deep learning–based methods have been made [5, 6]. Compared to traditional methods, deep learning methods can automatically extract features without manual designs, and they could perform better in some challenging scenes.

Supervised training methods are employed by some deep learning systems, which need ground truth data. However, the acquisition of ground truth data is difficult and expensive. Recently some researchers proposed unsupervised SLAM algorithms [7] to avoid the use of labelled data.

✉ Qiang Liu
qiang.liu@psych.ox.ac.uk

Yi An
anyi@dlut.edu.cn

Jin Shi
sj2796219195@mail.dlut.edu.cn

Dongbing Gu
dgu@essex.ac.uk

[1] School of Control Science and Engineering, Dalian University of Technology, Dalian 116023, China

[2] School of Computer Science and Electric Engineering, University of Essex, Colchester CO4 3SQ, UK

[3] Department of Psychiatry, University of Oxford, Oxford OX3 7JX, UK

Unsupervised learning methods reduce the difficulty of acquiring training data, so it is easier to expand the training dataset. Given a larger dataset, the pose and depth estimation accuracy and robustness can be further increased.

Most unsupervised learning SLAM methods only use single-modal data like RGB images or light detection and ranging (LiDAR) data. Visual SLAM requires relatively stable lighting changes, and some of them only use monocular images, which cannot obtain the absolute scale directly. Compared to visual SLAM, LiDAR SLAM has higher accuracy. However, the collected 3D point cloud data are distorted due to the moving of LiDAR sensors. Moreover, the vertical resolution of LiDAR sensors is low.

In this paper, we propose a novel unsupervised visual-LiDAR SLAM method to compensate for the weakness of each sensor and explore the complementary advantages. Our SLAM consists of an unsupervised visual-LiDAR odometry component, a deep learning–based loop closure detection component, and a 3D mapping component. The core of our odometry component is a multi-channel recurrent convolutional neural network (RCNN). Its input includes monocular RGB images and multi-channel depth images generated from 3D LiDAR data. Our loop closure detection component adopts a convolutional neural network (CNN) to detect the loop closure. Then, the general graph optimization (g2o) [8] is used to conduct the global graph optimization. Finally, a global 3D map can be constructed from our 3D mapping component. Figure 1 shows the framework of our system.

Since the system adopts an unsupervised training method, no ground truth data is used. During the training process, consecutive RGB images and multi-channel depth images are fed into the network. The outputs of the network are 6D pose and 3D maps. Our experiments are based on the KITTI odometry dataset [9]. Results have shown that our SLAM and odometry are better than some of the state-of-the-art unsupervised visual odometry (VO) and SLAM methods in terms of translation and rotation accuracy.

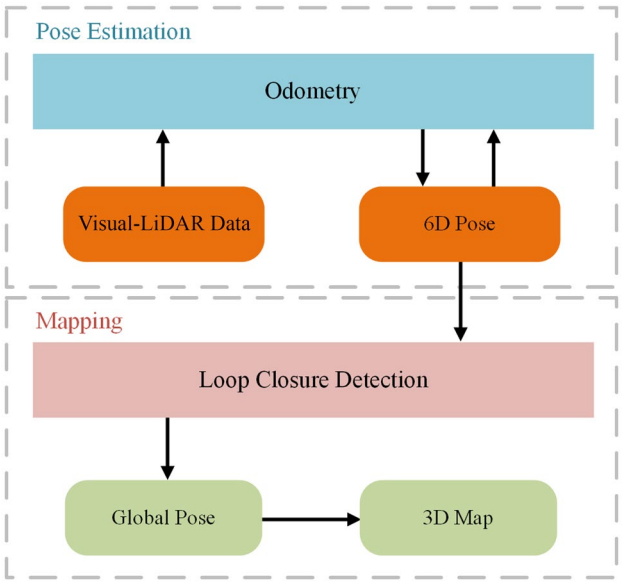


Fig. 1 Overview of our proposed visual-LiDAR SLAM

Our main contributions are summarized as follows:

- Our SLAM method fuses multi-modal data, including RGB images and LiDAR data, to estimate the pose and 3D map, improving the accuracy and robustness of pose estimation.
- An unsupervised learning method is proposed for the multi-channel visual-LiDAR odometry component, which reduces the costs of training dataset collection.
- Our loop closure detection is implemented with a CNN model to extract the loop closure information from RGB images and multi-channel depth maps.

The rest of the paper is organized as follows. "Related Works" describes the literature related to visual or LiDAR SLAM methods. "Our SLAM System" shows our SLAM method in detail. Experimental results are listed in "Experiments". And the last part is a summary and the future work.

Related Works

Traditional Visual SLAM

Davison et al. [10] proposed a monocular SLAM method (MonoSLAM). MonoSLAM is the first real-time monocular SLAM, which is based on extended Kalman filter (EKF). MonoSLAM extracts Shi and Tomasi features [11] from monocular images to estimate the pose. Endres et al. [12] proposed RGB-D SLAM V2. Oriented fast and rotated brief (ORB) [13], features from scale-invariant (SIFT) [14], and speeded up robust features (SURF) [15] can be used in the feature extraction stage. Mur-Artal and Tardós [16] proposed ORB-SLAM2, expanding the previously proposed monocular SLAM to monocular, binocular, and RGB-D SLAM.

Supervised Visual SLAM

Since the significant achievement of deep learning methods in image recognition and classification tasks, many researchers have introduced deep learning methods into their visual SLAM methods. Compared with the traditional methods, deep learning–based methods automatically perform feature extractions, feature matching, or complex geometric operations, which makes the deep learning–based methods more attractive. Kendall et al. [17] firstly applied the CNN to VO. The PoseNet they proposed takes monocular images as the inputs, and the outputs are 6D poses. Handa et al.

[18] extended the spatial transform network [19] to deliver an RGB-D VO method. The network is inspired by the VGG-16 network [20], and it also can estimate depths from the monocular RGB images. Wang et al. [21] extended the CNN by adding a recurrent neural network (RNN) structure. The CNN extracts features, and the RNN performs feature matching. However, all the above methods need ground truth data.

Unsupervised Visual SLAM

The main advantage of unsupervised methods is no need for ground truth data which decreases the difficulty of acquiring training data. Zhou et al. [22] proposed an unsupervised system to recover poses and depth information from videos, paving the way for unsupervised visual VO methods. However, this system only uses RGB images, and the predicted results do not include an absolute scale. Mahjourian et al. [23] proposed an unsupervised system using 3D geometric constraints. Except for the pixel-wise or gradient-based information in small local neighborhoods, the system also considers the 3D geometry of a scene to enforce the consistency of the estimated 3D point clouds and ego-motion across consecutive frames. The principle is similar to iterated closest point (ICP) [24]. Liu et al. [25] proposed an unsupervised monocular VO method, but depth information is still needed for training. Li et al. [7] proposed an unsupervised monocular SLAM method (DeepSLAM). It requires stereo images for training and monocular images for testing. However, DeepSLAM does not use LiDAR data.

Traditional LiDAR SLAM

Compared with visual methods, LiDAR SLAM methods have higher accuracy. However, the acquisition frequency of LiDAR sensors is lower. Zhang and Singh [3] proposed a LiDAR odometry method (LOAM), which extracts and matches geometric features in Cartesian space and has a lower requirement on the cloud density. LOAM includes two algorithms. One algorithm performs odometry at a high frequency but with low accuracy. The other algorithm runs at a lower frequency for mapping. Then, they fused visual and LiDAR sensors and proposed a visual-LiDAR odometry method (VLOAM) [26] based on LOAM. Compared with LOAM, VLOAM has higher accuracy. VLOAM includes visual odometry to estimate ego-motion at a high frequency and LiDAR odometry to refine the motion estimation at a lower frequency. Deschaud [27] proposed a 3D LiDAR SLAM method based on a scan-to-model matching framework, which used the implicit moving least squares (IMLS) surface representing LiDAR sweeps.

Supervised LiDAR SLAM

Li et al. [28] proposed a supervised LiDAR odometry (LO-Net) method, which has the similar accuracy with LOAM. LO-Net can be trained in an end-to-end manner. LO-Net can learn feature representation from LiDAR data efficiently. Cho et al. [29] proposed DeepLO. DeepLO is based on a geometric constraint and incorporates the ICP algorithm into a deep learning framework. Li et al. [30] proposed a semantic SLAM method including a semantic segmentation network. The system can construct a 3D semantic map. Lu et al. [31] proposed a learning-based LiDAR localization system (L3-Net). L3-Net achieves centimeter-level localization accuracy. 3D convolutions are used to enhance the accuracy.

Unsupervised LiDAR SLAM

Yin et al. [32] proposed a LO method, which is based on an unsupervised convolutional auto-encoder structure (CAE-LO). However, CAE-LO is not an end-to-end method, which only uses a neural network to extract features from 3D LiDAR data. Cho et al. [33] proposed an unsupervised LiDAR odometry method based on geometric information (UnGLO). UnGLO can output the poses directly. UnGLO utilizes a 2D spherical projection for input representation and uses point-to-plane ICP to formulate the loss function.

Our SLAM System

Our SLAM consists of an unsupervised visual-LiDAR odometry component, a deep learning–based loop closure detection component, and a 3D mapping component. The loop closure detection is used to decrease the accumulated errors of our odometry. The 3D map is constructed by using an optimized global pose. Figure 1 shows the overview of our SLAM system.

Visual-LiDAR Odometry Component

The odometry needs to predict the pose from a consecutive input sequence. We propose an RCNN to be the core of our odometry. The network adopts an unsupervised training framework. The inputs are RGB images and multi-channel depth images generated from 3D LiDAR point clouds. The outputs are the 6D pose with an absolute scale.

Data Preparation

To ensure the unity of the network structure, our odometry does not use LiDAR data directly. We convert the 3D LiDAR data to multi-channel depth images. The multi-channel depth

images include three depth images from left, right, and front views. The purpose of using multi-channel depth images is to get more information about the sensor motion.

We project the LiDAR data onto the imaging plane to obtain the front depth images, which correspond to RGB images at the pixel level. $I = \{I_t | 1 \leq t \leq n\}$ represents an RGB image sequence, and $D^f = \{D_t^f | 1 \leq t \leq n\}$ denotes a front depth image sequence corresponding to I. Let $\boldsymbol{q}_t = \left[x_t, y_t, z_t\right]^{\mathrm{T}}$ denote a 3D LiDAR point, and let $\boldsymbol{p}_t = \left[u_t, v_t\right]^{\mathrm{T}}$ be its projection on I_t, and d_t is the depth value of $\boldsymbol{p}_t$. The projecting process can be described as:

$$d_t \widetilde{\boldsymbol{p}}_t = K[R\ \boldsymbol{t}]\widetilde{\boldsymbol{q}}_t \tag{1}$$

where $\widetilde{\boldsymbol{p}}_t = \left[u_t, v_t, 1\right]^{\mathrm{T}}$ and $\widetilde{\boldsymbol{q}}_t = \left[x_t, y_t, z_t, 1\right]^{\mathrm{T}}$ are the homogenous coordinates of $\boldsymbol{p}_t$ and $\boldsymbol{q}_t$ respectively. $[R\ \boldsymbol{t}]$ is the extrinsic parameter matrix (rotation matrix and translation vector) between the 3D LiDAR data and the camera. K is the intrinsic matrix of the camera. Through Eq. (1), d_t can be solved. The corresponding depth value of I_t namely D_t^f can be obtained. Due to the sparse distribution of point clouds, only a few pixel points have depth values. We use the barycentric interpolation method to fill the holes in the depth images.

For the left and right depth images, we build two virtual cameras generated from the real camera rotating 90° clockwise and anticlockwise with itself as the center, and then we obtain the left and right cameras. Similarly, we use Eq. (1) to project the LiDAR data onto imaging planes of the left and right cameras to get the left and right depth images. $D^l = \{D_t^l | 1 \leq t \leq n\}$ and $D^r = \{D_t^r | 1 \leq t \leq n\}$ represent consecutive left and right depth image sequences respectively. The LiDAR points that cannot be projected onto the three imaging planes (front, left, right cameras) are discarded.

Network Architecture

The network is a four-channel architecture. The inputs of the four channels are RGB images, front depth images, left depth images, and right depth images. The inputs of each channel are multiple consecutive images. We input five consecutive images to the network in training. The RGB images and the depth images are resized to $416 \times 128 \times 3$. The architecture of our odometry component is shown in Fig. 2.

The design of our CNN framework is based on Oxford's visual geometry group (VGG) [20]. We modify the VGG network. Each channel of the CNN starts with a 7×7 convolutional layer, followed by two 5×5 convolutional layers. Both are used to capture the basic features of images. The remaining structure is five 3×3 convolutional layers. We reduce the size of the convolutional layer to obtain more detailed features. In the proposed network, the strides used

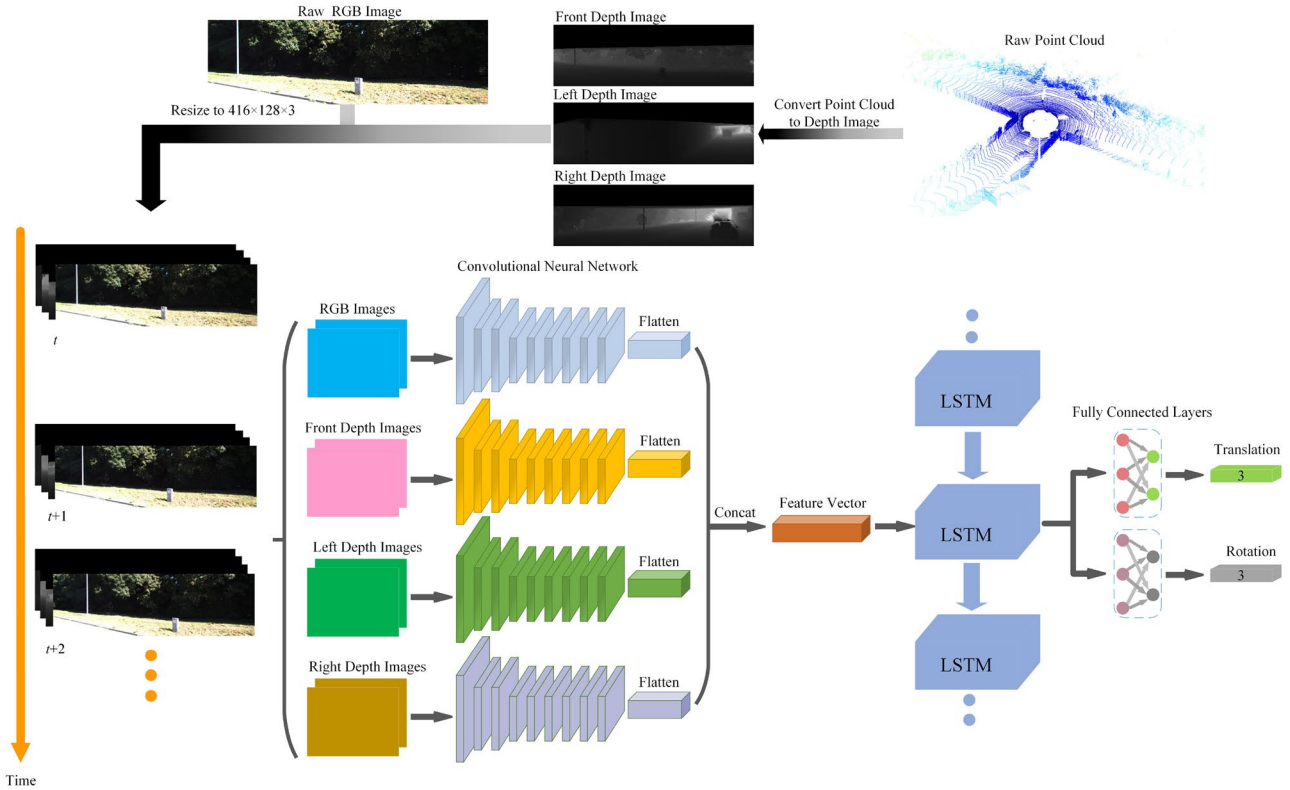


Fig. 2 Architecture of the visual-LiDAR odometry component based on RCNN

Table 1 Convolutional layers

Layer	Filter size	Stride	Padding	Channel number
Conv1	7×7	2	3	8
Conv2	5×5	2	2	16
Conv3	5×5	2	1	32
Conv4	3×3	2	1	64
Conv5	3×3	1	1	128
Conv6	3×3	1	1	256
Conv7	3×3	1	1	256
Conv8	3×3	2	1	256

are 2 and 1. We also use zero padding to prevent the image size from decreasing too quickly. A rectified linear unit (ReLU) activation function follows each convolutional layer to introduce nonlinearity to the network. The convolutional layers are detailed in Table 1.

The CNN as a feature extractor converts images into feature vectors, while the RNN as a pose prediction module combines the feature vectors in consecutive time steps. The RNN is a long short-term memory (LSTM) structure. Compared with a standard RNN, the LSTM can avoid the problem of long-term dependence. The core of an LSTM is the cell state, which is controlled by three regulators, namely the forget, input, and output gates. The number of hidden units in an LSTM cell is 256. According to [34], we set the bias of the forget gate to 1.

The full connection layer follows the LSTM. The fully connected structure has two layers. The dimension of each layer is set to 256. The outputs of the full connection layer are the rotation and translation vectors, namely the 6D pose. The rotation vector is represented by 3 values using Euler angles, and the translation vector includes 3 values in the Euclidean space.

Loss Function

We use RGB images and multi-channel depth images to calculate the loss function. The loss function consists of two parts: 2D and 3D spatial loss. The 2D spatial loss is the difference of pixel values between a raw image and a reconstructed image. And the 3D spatial loss is the difference between a raw 3D point cloud and a reconstructed 3D point cloud. The illustrations of loss calculations are shown in Figs. 3 and 4 for the 2D and 3D spatial loss respectively. $T_{t->t+1}$ represents the predicted transformation matrix from the network.

- 2D spatial loss: We project the pixels of an RGB image from I_t to I_{t+1}. $\hat{I}_{t+1}$ is the reconstructed image, which is constructed by $R_{t->t+1}$, $\boldsymbol{t}_{t->t+1}$, and I_t. $R_{t->t+1}$ and $\boldsymbol{t}_{t->t+1}$ represent the predicted rotation matrix and translation vector from the network. The reconstruction process of $\hat{I}_{t+1}$ can be described as:

$$\begin{bmatrix} \hat{\boldsymbol{p}}_{t+1} \\ 1 \end{bmatrix} = \frac{1}{s} K(R_{t->t+1} d_t K^{-1} \widetilde{\boldsymbol{p}}_t + \mathbf{t}_{t->t+1}) \tag{2}$$

where $\hat{\boldsymbol{p}}_{t+1}$ is a pixel coordinate of $\hat{I}_{t+1}$, and the corresponding pixel value is same as $\boldsymbol{p}_t$. s is a scale factor. Since $\hat{\boldsymbol{p}}_{t+1}$ is not a integer in most cases, it is projected to its four neighboring pixel points: top-left, top-right, bottom-left, and bottom-right as recommended in [19].

Similarly, we use $\hat{\boldsymbol{p}}_t$ to construct $\hat{I}_t$. $\hat{\boldsymbol{p}}_t$ can be solved by Eq. (3). The process is described as:

$$\begin{bmatrix} \hat{\boldsymbol{p}}_t \\ 1 \end{bmatrix} = \frac{1}{d_{t+1}} K R_{t->t+1}^{-1} (s K^{-1} \widetilde{\boldsymbol{p}}_{t+1} - \mathbf{t}_{t->t+1}) \tag{3}$$

where $\widetilde{\boldsymbol{p}}_{t+1}$ is the homogenous coordinate of $\boldsymbol{p}_{t+1}$.

- 3D spatial loss: We convert the depth images to the 3D space to generate a point cloud, that is, a pixel of depth images is converted to a 3D point. $C = \{C_t | 1 \le t \le n\}$ is the point cloud generated from D^f, D^l, and D^r in the camera coordinate. $\boldsymbol{c}_t = [x_t, y_t, z_t]^{\mathrm{T}}$ is a 3D point of C_t. Translating $\boldsymbol{c}_t$ to the coordinate of C_{t+1} can be expressed by the following equation:

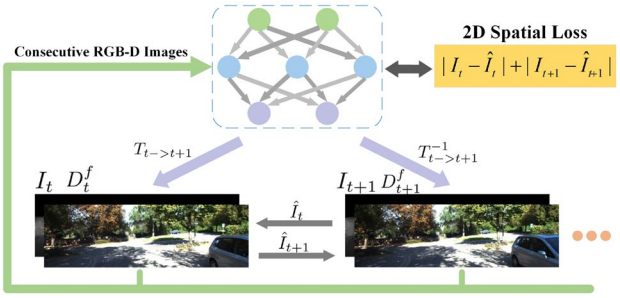


Fig. 3 2D spatial loss calculation for training

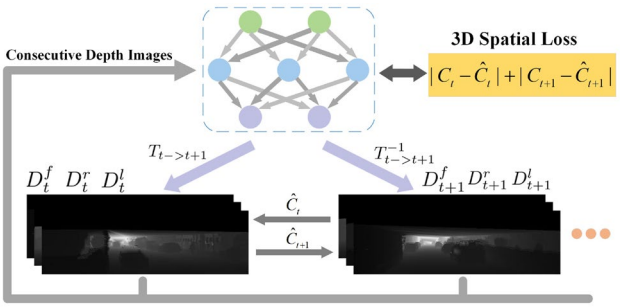


Fig. 4 3D spatial loss calculation for training

$$\hat{\boldsymbol{c}}_{t+1} = R_{t->t+1}\boldsymbol{c}_t + \mathbf{t}_{t->t+1} \tag{4}$$

The point cloud $\hat{C}_{t+1}$ is constructed by $\hat{\boldsymbol{c}}_{t+1}$. The difference between $\hat{C}_{t+1}$ and C_{t+1} is used to calculate the 3D spatial loss. We can also use Eq. (4) backwards to reconstruct C_t. The inverse equation is described as follows:

$$\hat{\boldsymbol{c}}_t = R^{-1}_{t->t+1}(\boldsymbol{c}_{t+1} - \boldsymbol{t}_{t->t+1}) \tag{5}$$

where $\hat{\boldsymbol{c}}_t$ can be used to construct the point cloud $\hat{C}_t$. The 2D spatial loss is described as:

$$L_{2D} = \sum_{t=1}^{n-1} \left(||I_t - \hat{I}_t||^2 + ||I_{t+1} - \hat{I}_{t+1}||^2 \right) \tag{6}$$

The 3D spatial loss is described as:

$$L_{3D} = \sum_{t=1}^{n-1} \left(||C_t - \hat{C}_t||^2 + ||C_{t+1} - \hat{C}_{t+1}||^2 \right) \tag{7}$$

We introduce two independent weights λ_{2D} and λ_{3D} to balance the 2D and 3D spatial loss. The total loss L is described as:

$$L = \lambda_{2D}L_{2D} + \lambda_{3D}L_{3D} \tag{8}$$

Loop Closure Detection Component

To reduce the accumulated errors of odometry, we use a loop closure detection component to find loop closures in the trajectory and improve the pose accuracy. The architecture of the loop closure detection component is shown in Fig. 5.

We adopt a pre-trained CNN for loop closure detection. The inputs of the pre-trained CNN are RGB images and multi-channel depth images, which are converted into feature vectors. We choose VGG19 to extract the feature vectors from the images.

At the time step t, we can obtain four images (I_t, D_t^f, D_t^l, and D_t^r). The four images are converted to four feature vectors. We concatenate the four feature vectors to generate a new vector $\boldsymbol{v}_t (1 \le t \le n)$ to describe a scene. By comparing the cosine distance between two feature vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ $(i \ne j)$, we can judge if the trajectory has a loop closure. The calculation equation of cosine distance is described as:

$$d_{\cos} = \cos(\mathbf{v}_i, \mathbf{v}_j) \tag{9}$$

where d_{cos} represents the similarity between $\mathbf{v}_i$ and $\mathbf{v}_j$. If d_{cos} is higher than a threshold d_{TH}, it can be considered that a loop closure has been found.

We convert the image sequence, including RGB images and multi-channel depth images, to feature vectors. Every scene has a corresponding feature vector. These feature vectors form a vector sequence $V = \{\boldsymbol{v}_t | 1 \le t \le n\}$. We use Eq. (9) to extracting key frames. The first frame is set as the current key frame. The cosine distance between the current key frame and the next key frame cannot exceed d_{KEY}. Following the vector sequence V, the next key frame can be found. Then, we set the next key frame as the current key frame. This process is repeated until the vector sequence V is traversed. Extracting the key frames also can reduce the

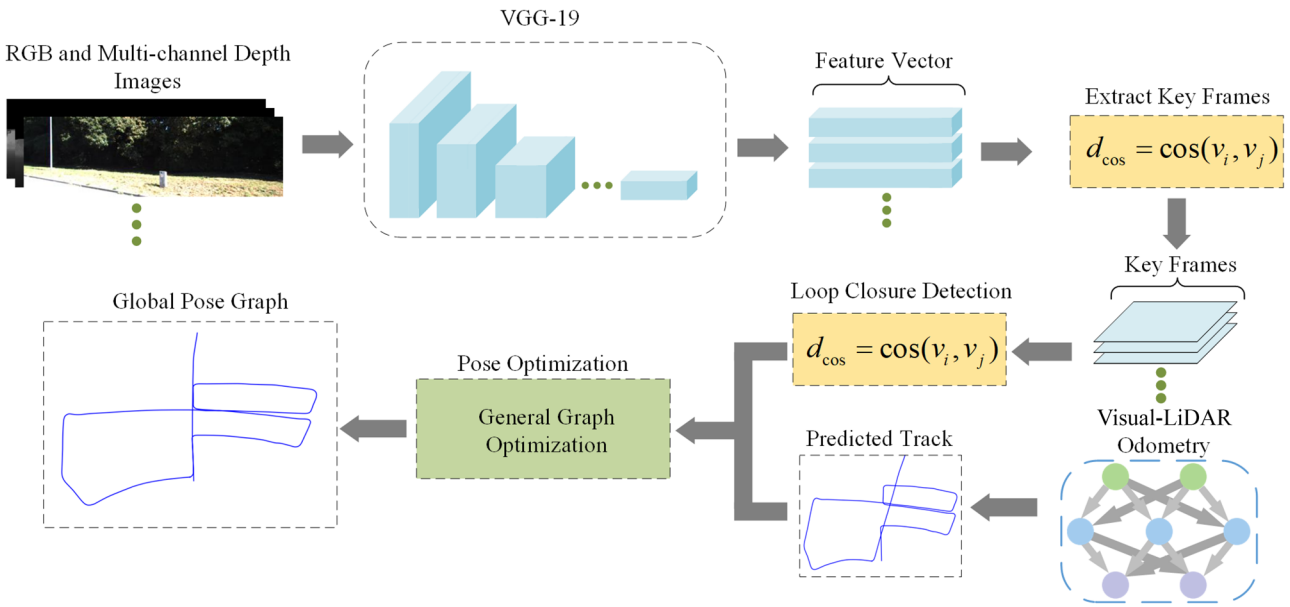


Fig. 5 Architecture of the loop closure detection component

time consumption of loop closure detection. Then, we calculate the similarity, namely the consine distance between key frames, to find the loop closures in the image sequence.

If the cosine distances of all feature vectors from key frames are lower than the d_{TH}, it is considered that there is no loop closure in the sequence. In this case, g2o is not used to optimize the pose.

After the loop closure detection, we feed the loop closure information and the predicted pose into the pose optimization system. Then, we use g2o [8] to optimize the predicted pose.

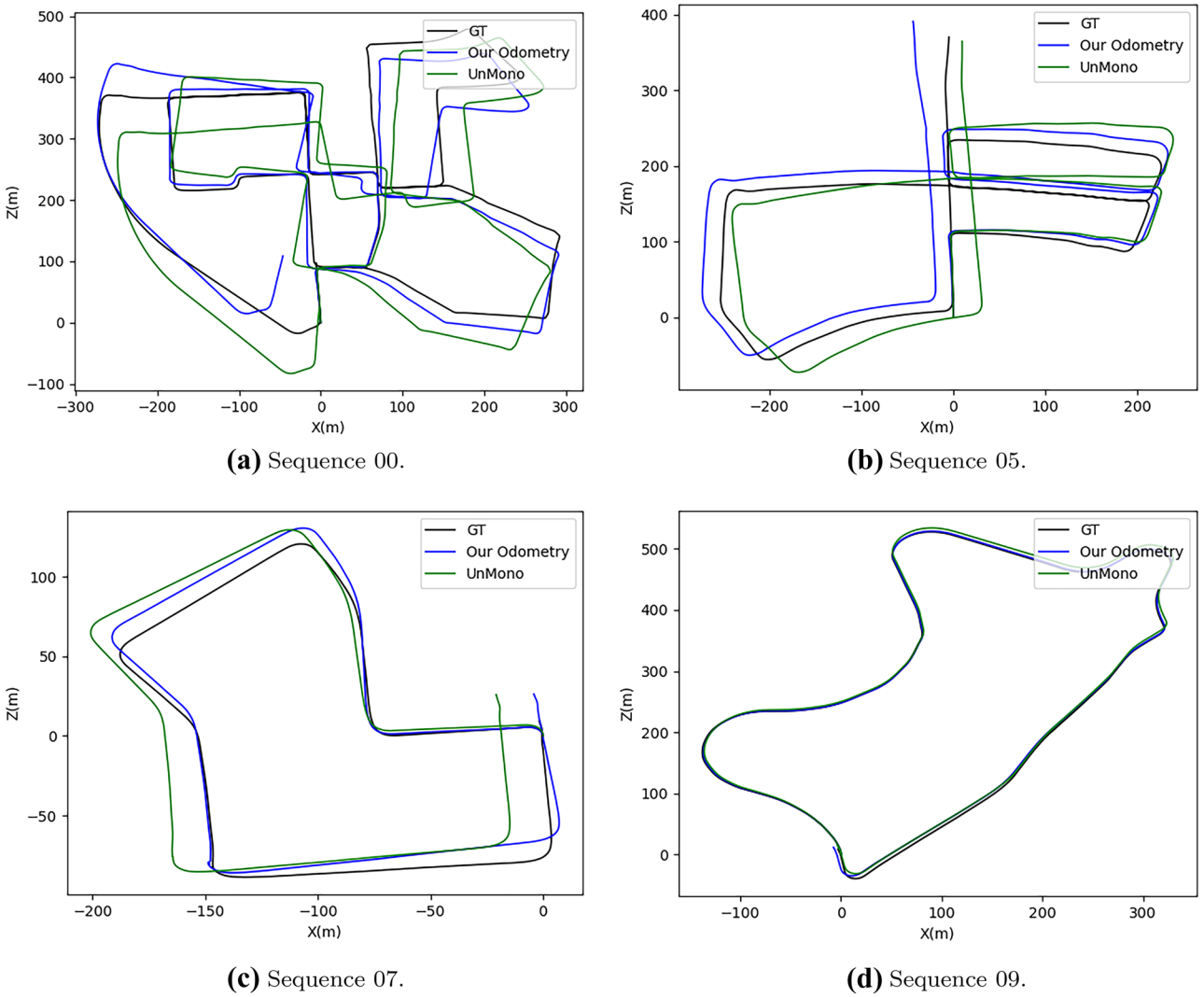
Experiments

In this part, we present our experimental results. We compare our odometry component and SLAM system with other state-of-the art odometry and SLAM methods, such as SfMLearner [22], UndeepVO [35], UnMono [25], DeepSLAM [7], UnGLO [33], VISO2-Mono, and VISO2-Stereo [36].

Training

The network was trained on a desktop with an Intel Core E5-1650 v3 @3.50GHz CPU and a Nvidia GeForce GTX 1080Ti 11GB Memory GPU.

Our code was based on Tensorflow. We used the KITTI odometry dataset as the training data, including 22 sequences captured by cars in cities, suburbs, highways. Ground truth data are provided in sequences 00–10, while 11–21 are not. UnGLO used 3D LiDAR data for training. SfMLearner and UnMono used monocular RGB images for training. UndeepVO and DeepSLAM used binocular RGB images for training. Our system was trained by monocular RGB images



(a) Sequence 00. **(b)** Sequence 05.

(c) Sequence 07. **(d)** Sequence 09.

Fig. 6 The trajectories of odometry with ground truth. (**a**) Sequence 00. (**b**) Sequence 05. (**c**) Sequence 07. (**d**) Sequence 09

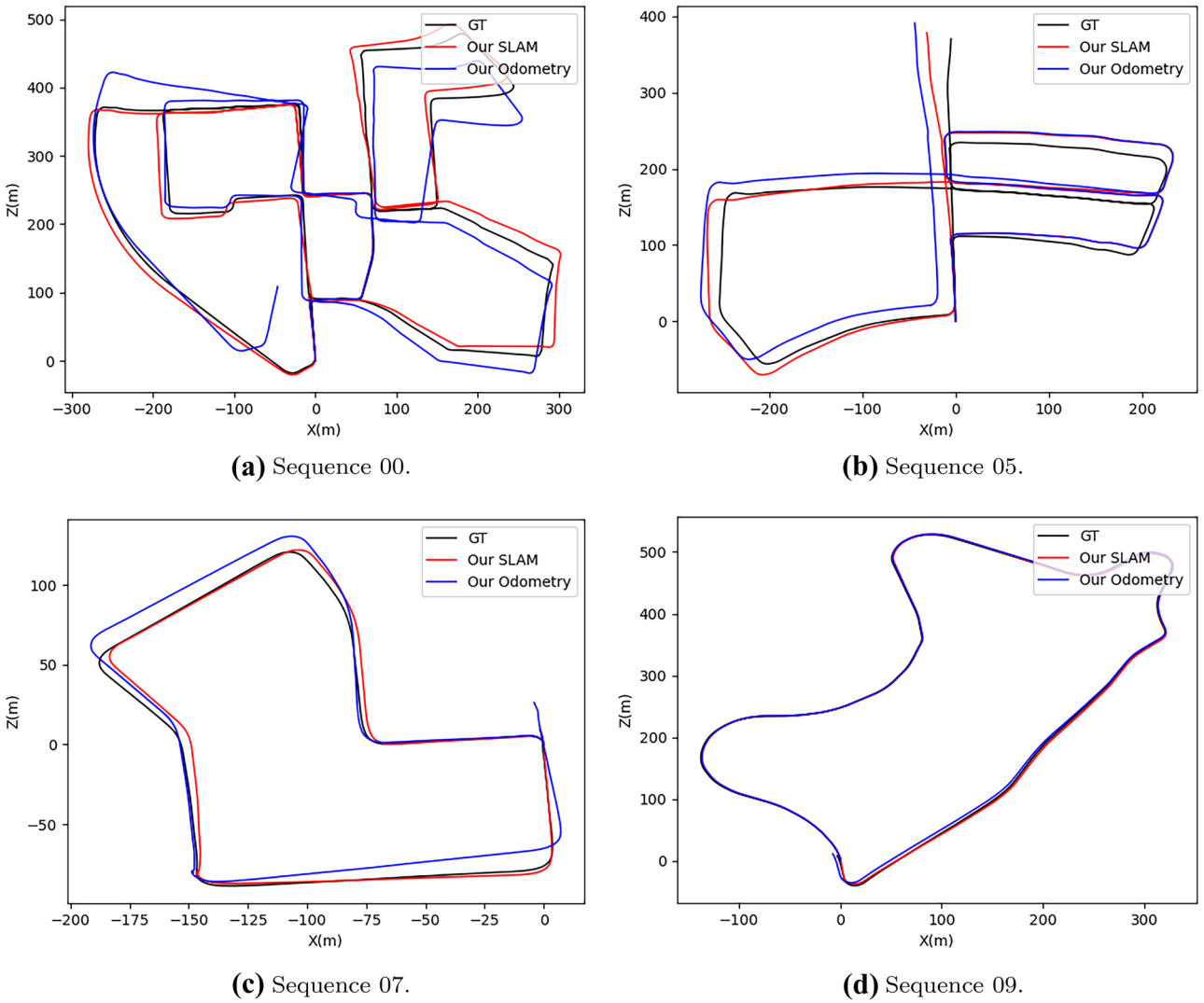
and multi-channel depth images generated from 3D LiDAR data. The size of training images was set to $416 \times 128 \times 3$, which was used by SfMLearner, UnMono, UndeepVO, DeepSLAM, our SLAM, and odometry. The input image sizes of VISO2-Mono and VIS2O-Stereo were 1242×376. Due to the relatively small training dataset, we enhanced the training data to ensure the system with stronger robustness. We enhanced the training data in the following aspects:

- **Luminance**: We randomly changed the light intensity on RGB images, and the adjustment range γ was [0.7, 1.3].
- **Scale**: We randomly changed the scale of RGB images and multi-channel depth images. The adjustment range of X and Y were [1.0, 1.2], and then they were clipped to 416×128.
- **Rotation**: We randomly rotated RGB and multi-channel depth images in the range of $r \in [-5, 5]$ degrees.

After completing the above steps, the processed data were fed into the network. The optimizer of our network was Adam [37]. We adopted the recommendations for the first and second attenuation indices: $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The network inputs were five consecutive RGB images and multi-channel depth images in one batch. The batch size of the training was 32. Within a total training length of 40 epochs, the loss function value tended to be stable around 15 epochs under normal conditions.

Performance Evaluation

Compared to our odometry, our SLAM adds a loop closure detection component. According to the loop closure information, SLAM uses g2o to optimize the pose from the odometry. DeepSLAM adopted 00–02, 08, 09, and 11–21 sequences for training. Other unsupervised methods used



(a) Sequence 00. **(b)** Sequence 05.

(c) Sequence 07. **(d)** Sequence 09.

Fig. 7 The trajectories of SLAM with ground truth. (**a**) Sequence 00. (**b**) Sequence 05. (**c**) Sequence 07. (**d**) Sequence 09

00–08 sequences as the training dataset. In the training process, UnMono, DeepSLAM, SfmLearner, and UnDeepVO are all trained by the enhanced training dataset.

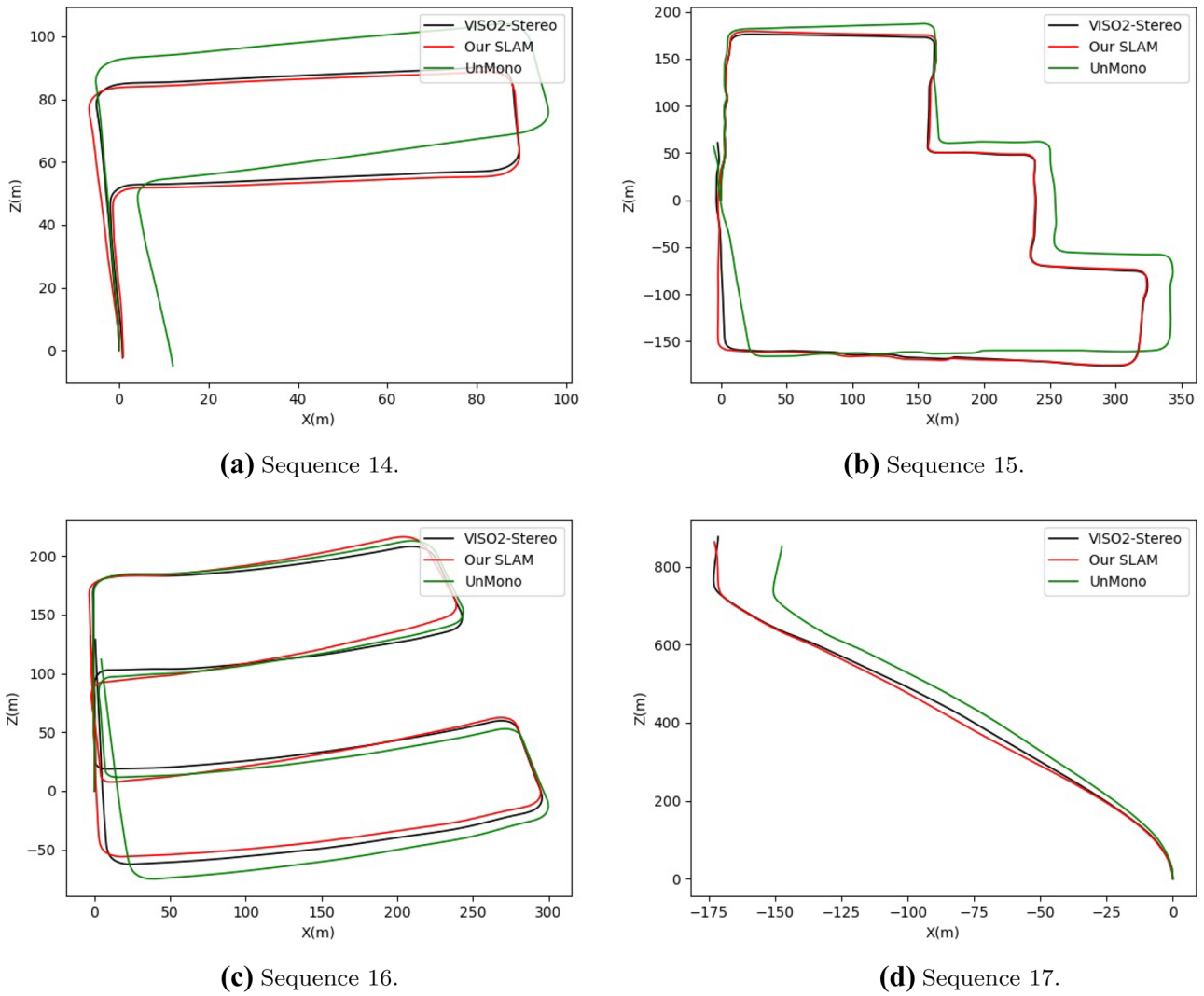
The experimental comparison is divided into four aspects:

- Our odometry was compared with other odometry methods in 00–10 sequences.
- Our SLAM was compared with our odometry in 00–10 sequences.
- Our SLAM was compared with VISO2-Stereo and UnMono in 11–21 sequences.
- The 3D map was constructed through the predicted global pose graph.

Since the KITTI odometry dataset is taken from an outdoor moving vehicle, only minor deviations were generated in the vertical direction. To clearly present the differences between trajectories, the *y* axis is omitted in trajectory diagrams. The initial point of each trajectory is set as (0,0). Intuitively, the closer the trajectory is to the ground truth (GT) curves, the higher the accuracy.

The trajectories of odometry selected from 00 to 10 sequences are presented in Fig. 6. The trajectories of GT, our odometry, and UnMono are represented by black, blue, and green curves respectively. As can be observed, the trajectories of our odometry are closer to GT curves than UnMono. The accuracy of our odometry is higher than UnMono. It can be seen the importance of adding multi-channel depth data to the system, which can improve the predicted accuracy of trajectories.

The trajectories of our SLAM selected from 00 to 10 sequences are presented in Fig. 7. The trajectories of our SLAM are represented by red curves. As can be observed, the trajectories of our SLAM are closer to GT curves than



(a) Sequence 14.

(b) Sequence 15.

(c) Sequence 16.

(d) Sequence 17.

Fig. 8 The trajectories of SLAM without ground truth. (**a**) Sequence 14. (**b**) Sequence 15. (**c**) Sequence 16. (**d**) Sequence 17

our odometry. The accuracy of our SLAM is higher than our odometry. Due to the influence of accumulated errors, the prediction accuracy of turning angles plays a pivotal role. It can be seen, adding loop closure detection to our odometry, the deviation between prediction trajectories and GT trajectories was reduced.

The trajectories of our SLAM selected from 11 to 21 sequences are presented in Fig. 8. The trajectories of VISO2-Stereo are represented by black curves. Since there is no ground truth in 11–21 sequences, VISO2-Stereo is used as a reference. As can be observed, the trajectories of our SLAM are closer to VISO2-Stereo than UnMono, which again proves the importance of multi-channel depth data.

The detailed error analysis is shown in Tables 2 and 3. We computed the root mean square error (RMSE) of translation and rotation on the lengths of 100, 200,..., 800 meters. As can be observed, the average error of our odometry is less than UnMono. It can be seen the training and testing results are improved by adding the multi-channel depth images. Only the rotation errors of our odometry in sequences 04 and 05 are higher than UnMono. The pose transformation includes rotation and translation. When the sensors (the camera and LiDAR) go straight, the rotation angle is small. When the sensors turn, the rotation angle is large. The translation is a linear process, while rotation is a non-linear process. Therefore, rotation is more difficult to predict for the network. In the training process, UnMono adds many turning data to improve the rotation prediction accuracy. However, this may result in decreased translation accuracy. Therefore, the proposed method does not add too many turning data during the training process. The training data of turning and going straight are balanced, so the proposed odometry may have higher rotation errors than UnMono in some sequences.

Compared with other unsupervised visual odometry and SLAM methods, our SLAM system has higher accuracy. Compared with VISO2-Stereo, our SLAM and odometry methods automatically extract features without manual designs, and they could perform better in some challenging scenes. The data of sequence 01 are collected from the expressway, which is more difficult to estimate the pose than urban and rural streets. As can be observed, our SLAM and odometry methods perform better than VISO2-Stereo in sequence 01. It can be seen, our methods are more robust than VISO2-Stereo. Compared with UnGLO, our odometry has a lower translation error. Since UnGLO adopts LiDAR data directly, UnGLO has a lower rotation error. From the predicted results of sequences 09 and 10, which are not used for training, our odometry has higher accuracy. It can be seen our odometry is more robust than UnGLO, and by comparing the mean errors of our SLAM and odometry, it can be seen adding loop closure detection in the system can reduce translation and rotation errors. However, in some cases, the optimization of the posture may have some negative effects. For example, our SLAM has a higher rotation error than our odometry in sequence 07. The error of the odometry accumulates over time, the initial error is small, and as time increases, the error gradually increases. G2o can effectively reduce the gradually accumulated small errors.

Table 2 Translational and rotational errors of our SLAM, our odometry, UnMono, DeepSLAM, VISO2-Mono, and VISO2-Stereo. Our SLAM, our odometry, UnMono, and DeepSLAM are unsupervised deep learning based. VISO2-Mono and VISO2-Stereo are feature based

	Our SLAM		**Our odometry**		**UnMono**		**DeepSLAM**		**VISO2-Mono**		**VISO2-Stereo**	
	Visual-LiDAR		Visual-LiDAR		Monocular		Monocular		Monocular		Stereo	
	Unsupervised		Unsupervised		Unsupervised		Unsupervised		Feature based		Feature based	
Seq.	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	2.53	0.79	2.77	1.78	5.14	2.13	NA	NA	18.35	2.73	1.87	0.59
01	3.76	0.80	3.76	0.80	15.64	0.95	NA	NA	36.52	7.69	8.61	1.23
02	3.95	1.05	4.82	2.26	4.86	2.30	NA	NA	4.36	1.19	2.01	0.41
03	2.75	1.39	2.75	1.39	6.03	1.83	7.66	4.3	8.47	8.82	3.21	0.73
04	1.81	1.48	1.81	1.48	2.15	0.89	4.56	1.90	4.69	4.69	2.12	0.24
05	3.49	0.79	3.81	1.43	3.84	1.29	3.25	1.31	19.22	17.58	1.53	0.53
06	1.84	0.83	4.03	1.24	4.29	1.33	4.97	1.53	7.15	1.93	1.57	0.32
07	3.27	1.51	3.61	1.41	3.80	1.71	4.71	1.84	23.61	29.11	1.85	0.78
08	2.75	1.61	2.75	1.61	2.92	1.63	NA	NA	24.47	2.53	1.92	0.56
09	3.7	1.83	3.76	1.92	5.58	2.77	NA	NA	7.17	1.25	1.94	0.54
10	4.65	0.51	4.65	0.51	5.14	3.34	8.35	3.93	44.61	3.26	1.18	0.48
Mean	3.14	1.14	3.50	1.43	5.40	1.94	5.58	2.47	18.28	2.99	2.52	0.60

t_{rel}(%): average translational RMSE drift (%) on length of 100–800m

r_{rel}(°/100m): average rotational RMSE drift (°/100m) on length of 100–800m

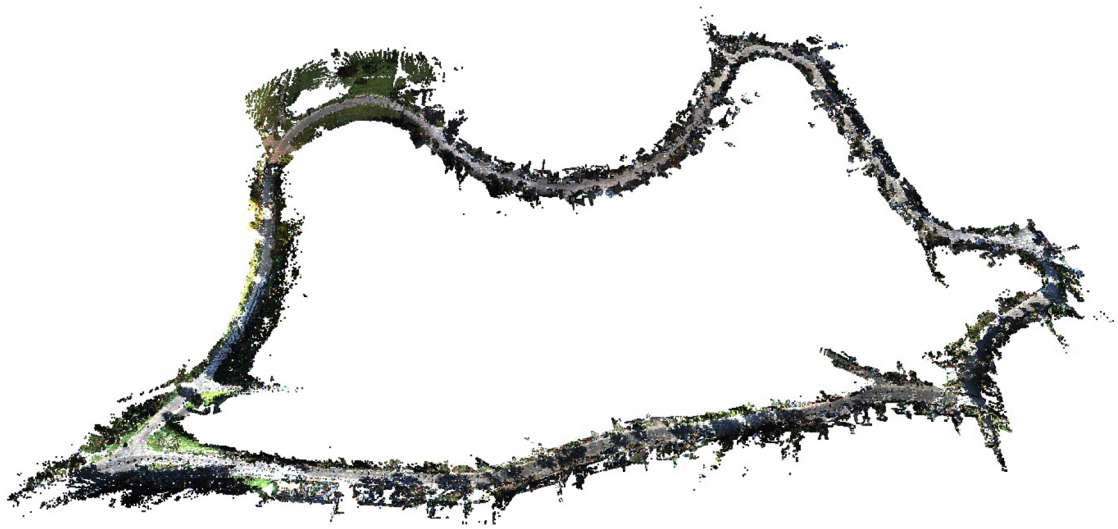
Table 3 KITTI odometry evaluation

Methods	Seq. 00–08		Seq. 09		Seq. 10	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Our odometry	3.34	1.49	3.76	1.92	4.65	0.51
UnDeepVO	4.54	2.55	7.01	3.61	10.63	4.65
UnMono	4.18	1.61	5.59	2.57	5.14	3.34
UnGLO	3.68	0.87	4.87	1.95	5.02	1.83
SfmLearner	28.52	4.67	18.77	3.21	14.33	3.30

$t_{rel}(\%)$ and $r_{rel}(^\circ/100m)$ are the average translational RMSE drift and rotational RMSE drift on length of 100 to 800m. The sequences 00–08 are the training dataset, and the sequences 09 and 10 are used to test

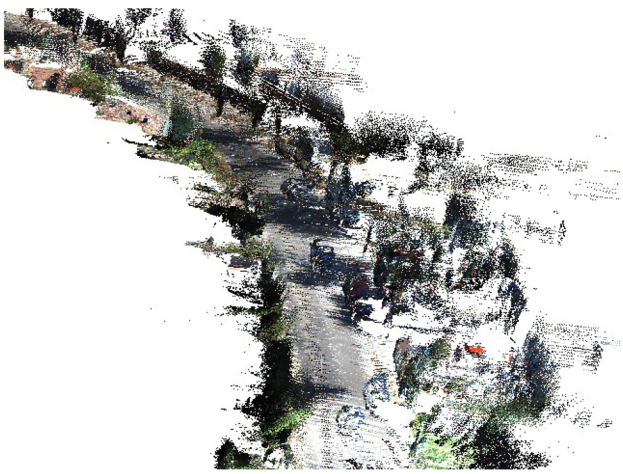
However, if the predicted pose of the odometry at a certain time produces an abrupt large error as shown in sequence 07, using g2o does not always have a good optimization result. Since g2o optimizes the global pose, in order to reduce this abrupt large errors, other poses will be over-corrected and new errors will be introduced.

The 3D reconstruction on sequence 09 is shown in Fig. 9. The 3D LiDAR data was projected to the image plane, and the



(a) Sequence 09.

(b) Sequence 09.



(c) Sequence 09.

Fig. 9 3D reconstruction of sequence 09. (**a**) Sequence 09. (**b**) Sequence 09. (**c**) Sequence 09

RGB pixel points corresponding to the 3D points were found, that is, the RGB information of 3D points was obtained. We used the color point clouds to reconstruct the 3D scene.

Finally, we analyze the time complexity of UnMono and our odometry. In the testing process, the batch size was set to 1. The time consumption of UnMono and our odometry is 25.63ms and 29.01ms respectively. Compared with UnMono, our odometry spends a longer running time due to the additional network structure.

Conclusions

In this paper, we proposed an unsupervised multi-channel visual-LiDAR SLAM method (MVL-SLAM), which consists of an unsupervised multi-channel visual-LiDAR odometry component (MVLO), a loop closure detection component, and a 3D mapping component. The estimation accuracy of poses is improved after adding multi-channel depth information generated from LiDAR data. The absolute scale can be obtained directly compared with other monocular systems. The loop closure detection uses a CNN model to extract features from the depth and RGB images. We combine the global poses, RGB images, and LiDAR data to construct 3D maps.

In the future, we will explore other options of combing LiDAR point clouds and RGB images together to improve its robustness under challenging circumstances.

Acknowledgements This work was supported in part by the National Natural Science Foudation of China under Grant 61673083, and in part by the Science and Technology Major Projecct of Shanxi Province under Grant 20191191014.

Declarations

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed Consent Informed consent was obtained from all individual participants included in the study.

Conflict of Interest The authors declare no competing interests.

References

1. Liu Q, Li R, Hu H, Gu D. Indoor topological localization based on a novel deep learning technique. Cogn Comput. 2020;12(3):528–41.
2. Wu H, Wu Y, Liu C, Yang G, Qin S. Fast robot localization approach based on manifold regularization with sparse area features. Cogn Comput. 2016;8(5):856–76.
3. Zhang J, Singh S. LOAM: Lidar odometry and mapping in real-time. In: Proceedings of Robotics: Science and Systems. 2014.
4. Mur-Artal R, Montiel J, Tardós J. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot. 2015;31(5):1147–63.
5. Deng C, Qiu K, Xiong R, Zhou C. Comparative study of deep learning based features in SLAM. In: 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS). 2019. p. 250–254.
6. Li C, Li Z, Feng Y, Liu Y, Shi G. Development of a human-robot hybrid intelligent system based on brain teleoperation and deep learning SLAM. IEEE Trans Autom Sci Eng. 2019;16(4):1664–74.
7. Li R, Wang S, Gu D. DeepSLAM: a robust monocular SLAM system with unsupervised deep learning. IEEE Trans Ind Electron. 2021;68(4):3577–87.
8. Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. G2O: A general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation. 2011. p. 3607–3613.
9. Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets robotics: the KITTI dataset. Int J Robot Res. 2013;32(11):1231–7.
10. Davison A, Reid I, Molton N, Stasse O. MonoSLAM: real-time single camera SLAM. IEEE Trans Pattern Anal Mach Intell. 2007;29(6):1052–67.
11. Shi J, Tomasi. Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 1994. p. 593–600.
12. Endres F, Hess J, Sturm J, Cremers D, Burgard W. 3-D mapping with an RGB-D camera. IEEE Trans Robot. 2014;30(1):177–87.
13. Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision. 2011. p. 2564–2571.
14. Lowe D. Distinctive image features from scale-invariant keypoints. Int J Comput Vis. 2004;20:91–110.
15. Bay H, Tuytelaars T, Gool LV. SURF: speeded up robust features. In: European Conference on Computer Vision. 2006. p. 404–417.
16. Mur-Artal R, Tardós J. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D. IEEE Trans Robot. 2017;33(5):1255–1262.
17. Kendall A, Grimes M, Cipolla R. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2015. p. 2938–2946.
18. Handa A, Bloesch M, Pătrăucean V, Stent S, McCormac J, Davison A. GVNN: neural network library for geometric computer vision. In: European Conference on Computer Vision. Springer; 2016. p. 67–82.
19. Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K. Spatial transformer networks. In: Advances in neural information processing systems 28. Curran Associates, Inc.; 2015. p. 2017–2025.
20. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICRA). 2014.
21. Wang S, Clark R, Wen H, Trigoni N. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). 2017. p. 2043–2050.
22. Zhou T, Brown M, Snavely N, Lowe D. Unsupervised learning of depth and ego-motion from video. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. p. 6612–6619.

23. Mahjourian R, Wicke M, Angelova A. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018. p. 5667–5675.
24. Besl P, Mckay H. A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell. 1992;14(2):239–56.
25. Liu Q, Li R, Hu H, Gu D. Using unsupervised deep learning technique for monocular visual odometry. IEEE Access. 2019;7:18076–88.
26. Zhang J, Singh S. Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). 2015. p. 2174–2181.
27. Deschaud JE. IMLS-SLAM: Scan-to-model matching based on 3D data. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018. p. 2480–2485.
28. Li Q, Chen S, Wang C, Li X, Wen C, Cheng M, Li J. LO-Net: Deep real-time Lidar odometry. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. p. 8465–8474.
29. Cho Y, Kim G, Kim A. DeepLO: Geometry-aware deep Lidar odometry. arXiv preprint arXiv:1902.10562. 2019.
30. Li R, Gu D, Liu Q, Long Z, Hu H. Semantic scene mapping with spatio-temporal deep neural network for robotic applications. Cogn Comput. 2018;10(2):260–71.
31. Lu W, Zhou Y, Wan G, Hou S, Song S. L3-Net: Towards learning based Lidar localization for autonomous driving. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. p. 6382–6391.
32. Yin D, Zhang Q, Liu J, Liang X, Wang Y, Maanpää J, et al. CAE-LO: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description. arXiv preprint arXiv:2001.01354. 2020.
33. Cho Y, Kim G, Kim A. Unsupervised geometry-aware deep Lidar odometry. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). 2020. p. 2145–2152.
34. Kawakami K. Supervised sequence labelling with recurrent neural networks. Ph. D. dissertation, PhD thesis. Ph. D. thesis. 2008.
35. Li R, Wang S, Long Z, Gu D. UnDeepVO: Monocular visual odometry through unsupervised deep learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018. p. 7286–7291.
36. Geiger A, Ziegler J, Stiller C. Stereoscan: Dense 3D reconstruction in real-time. In: 2011 IEEE Intelligent Vehicles Symposium (IV). 2011. p. 963–968.
37. Kingma D, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.