Unsupervised Motion Removal for Dynamic SLAM

CHEN Hao[1,2], ZHANG Kaijiong[1,2], CHEN Jun[1,2], ZHANG Ziwen[1,2], JIA Xia[1,2]
(1. ZTE Corporation, Shenzhen 518057, China;
2. State Key Laboratory of Mobile Network and Mobile Multimedia Technology, Shenzhen 518055, China)

DOI: 10.12142/ZTECOM.202404010
https://kns.cnki.net/kcms/detail/34.1294.TN.20241031.1625.002.html, published online November 1, 2024
Manuscript received: 2024-02-29

Abstract: We propose a dynamic simultaneous localization and mapping technology for unsupervised motion removal (UMR-SLAM), which is a deep learning-based dynamic RGBD SLAM. It is the first time that a scheme combining scene flow and deep learning SLAM is proposed to improve the accuracy of SLAM in dynamic scenes, in response to the situation where dynamic objects cause pose changes. The entire process does not require explicit object segmentation as supervisory information. We also propose a loop detection scheme that combines optical flow and feature similarity in the backend optimization section of the SLAM system to improve the accuracy of loop detection. UMR-SLAM is rewritten based on the DROID-SLAM code architecture. Through experiments on different datasets, it has been proven that our scheme has higher pose accuracy in dynamic scenarios compared with the current advanced SLAM algorithm.

Keywords: dynamic RGBD SLAM; update module; motion estimation; scene flow

Citation (Format 1): CHEN H, ZHANG K J, CHEN J, et al. Unsupervised motion removal for dynamic SLAM [J]. ZTE Communications, 2024, 22(4): 67 - 77. DOI: 10.12142/ZTECOM.202404010
Citation (Format 2): H. Chen, K. J. Zhang, J. Chen, et al., "Unsupervised motion removal for dynamic SLAM," *ZTE Communications*, vol. 22, no. 4, pp. 67 - 77, Dec. 2024. doi: 10.12142/ZTECOM.202404010.

1 Introduction

Simultaneous localization and mapping (SLAM) is an important technology in computer vision and autonomous robot navigation research. Its main goal is to enable mobile devices (robots, autonomous vehicles, unmanned aerial vehicles or AR/VR devices) to achieve autonomous positioning and map building by interacting with sensors in unknown or changing environments. SLAM systems typically include the following key components: data collection and preprocessing, front-end data processing, state estimation, map construction, and backend optimization, such as the typical Oriented FAST and Rotated BRIEF (ORB)-SLAM system. At present, although SLAM technology faces many challenges, such as sensor accuracy errors, computational complexity, and real-time requirements, it is still the core technology of many autonomous systems, providing the possibility for machines and equipment to navigate and work efficiently in complex environments.

In recent years, many deep learning and traditional SLAM fusion schemes have been proposed to improve the performance and robustness of environmental perception and pose estimation. Deep neural networks can play a crucial role in image feature extraction, semantic map construction, and loop detection. Extracting feature points or descriptors through deep learning models can improve the accuracy of image matching and feature tracking in visual SLAM. Deep learning can be used to establish a loopback detection model for visual and semantic contexts, which is used to detect whether the robot has returned to the previously visited position, and then perform global optimization to reduce cumulative errors. Deep learning can also be used for semantic map construction, enabling robots to understand the semantic information of different objects and regions in the environment, which contributes to autonomous decision-making and path planning in the field of autonomous driving. In addition, eliminating dynamic objects can also effectively improve the accuracy of pose estimation.

Semi-supervised or unsupervised SLAM is an emerging research field that explores how to train SLAM systems with unlabeled or limited labeled data. This method helps to address the dependency on large numbers of labeled data in traditional SLAM methods. Deep learning can also achieve end-to-end SLAM, directly generating maps and trajectories from sensor data without the need for intermediate steps. Recently, some research efforts have been devoted to designing more effective multimodal fusion strategies. The goal is to fuse data from different types of sensors, such as vision, light detection and ranging (LiDAR), GPS and inertial mea-

surement unit (IMU), together to improve the robustness and accuracy of SLAM systems. Real-time performance has always been a key challenge for SLAM systems. Researchers are striving to improve the computational efficiency of SLAM systems to meet the requirements of real-time applications such as autonomous driving, virtual reality, and robot navigation. Their main tasks include hardware acceleration, low-power algorithms, and distributed SLAM.

The current state-of-the-art algorithm DROID-SLAM[30] combines traditional methods with deep learning and has the advantages of high accuracy, strong robustness, and good generalization. However, it does not perform well in dynamic scenarios like the KITTI dataset, and pose estimation is easily affected by passing vehicles and pedestrians. As shown in Fig. 1, when a truck moves from left to right across the camera frame, the algorithm may mistakenly perceive the camera as undergoing a left-turning motion, leading to erroneous pose estimation outputs. Therefore, based on DROID-SLAM, we propose a new SLAM scheme to address the issue of dynamic objects affecting algorithm accuracy. The main contributions of our work are summarized as follows:

• We propose for the first time a scheme that combines scene flow and deep learning SLAM to improve the accuracy of SLAM in dynamic scenes while outputting dynamic object masks. The entire process does not require explicit object segmentation as supervisory information.

• We propose a new update module that can iteratively update the camera pose.

• We propose a loop detection scheme that combines optical flow and feature similarity to improve the accuracy of loop detection without increasing additional computational complexity.

2 Related works

2.1 Dynamic SLAM

SLAM solutions typically assume that the scene is almost static or has a low level of dynamism. However, there are often a large number of dynamic objects in real-world scenarios, including pedestrians, animals, cars, bicycles, and other dynamic objects, which can cause erroneous changes in feature matching relationships, resulting in inaccurate results due to the lack of reliable features in SLAM solutions.

To solve the above problem, some methods adopt object detection or semantic segmentation schemes to eliminate potential dynamic targets[1-4]. However, a large number of semantic segmentation objects in the camera's field of view may lead to insufficient features, which in turn can lead to problems with map matching and motion tracking, such as decreased system accuracy, tracking failures, and trajectory loss. In fact, dynamic objects may be static in the scene. Due to the limitations of semantic categories, on the one hand, they cannot cover all potential dynamic targets; on the other hand, some static objects are dynamic in the scene, such as books in people's hands. Many studies have introduced additional constraints to confirm the true dynamic objects in the scene. Based on semantic segmentation, authors in Ref. [5] utilize deep inconsistency checking to remove potential dynamic objects. Some methods do not rely on prior semantic information but distinguish between dynamic and static through the association with feature points[6-9]. Refs. [10] and [11] use dense optical flow methods and semantic segmentation to estimate the motion of objects in the scene, which helps to construct a globally consistent scene map and improve the robustness and accuracy of the system. Refs. [12] and [13] predict the camera's self-motion iteratively by correlating the camera's self-motion with the segmentation of dynamic objects, achieving their joint optimization in a single learning framework. Unlike the above research, the method proposed can output pixel-level pose changes unsupervised, segment true dynamic objects, and have higher robustness to different dynamic scenes.

▲Figure 1. Two images from the KITTI07 sequence

2.2 Optical Flow and Scene Flow

In SLAM algorithms, optical flow is commonly used to represent motion information between adjacent frames in a video sequence. This motion information can help SLAM systems estimate camera motion more accurately when processing dynamic scenes.

Deep learning-based optical flow estimation methods have gradually become a mainstream research direction. The FlowNet series[14-15] is the first to use an end-to-end deep learning architecture for optical flow estimation, emphasizing the importance of the training data sequence. Ref. [16] introduces many novel improvements of unsupervised optical flow models to enhance performance metrics. Refs. [17] and [18] consider the use of coarse-to-fine techniques to improve the performance metrics of optical flow networks. Refs. [19] and [20] construct multi-scale 4D correlation volumes for all pixels and iteratively update the optical flow

field through recurrent units that search the correlation volumes. Refs. [21] and [22] use multi-frame information for optical flow fusion to enhance optical flow computation performance. Additionally, some approaches[23-24] have made significant efforts to improve the computational efficiency of lightweight optical flow estimation networks for mobile and low-power usage scenarios.

2.3 Loop Detection

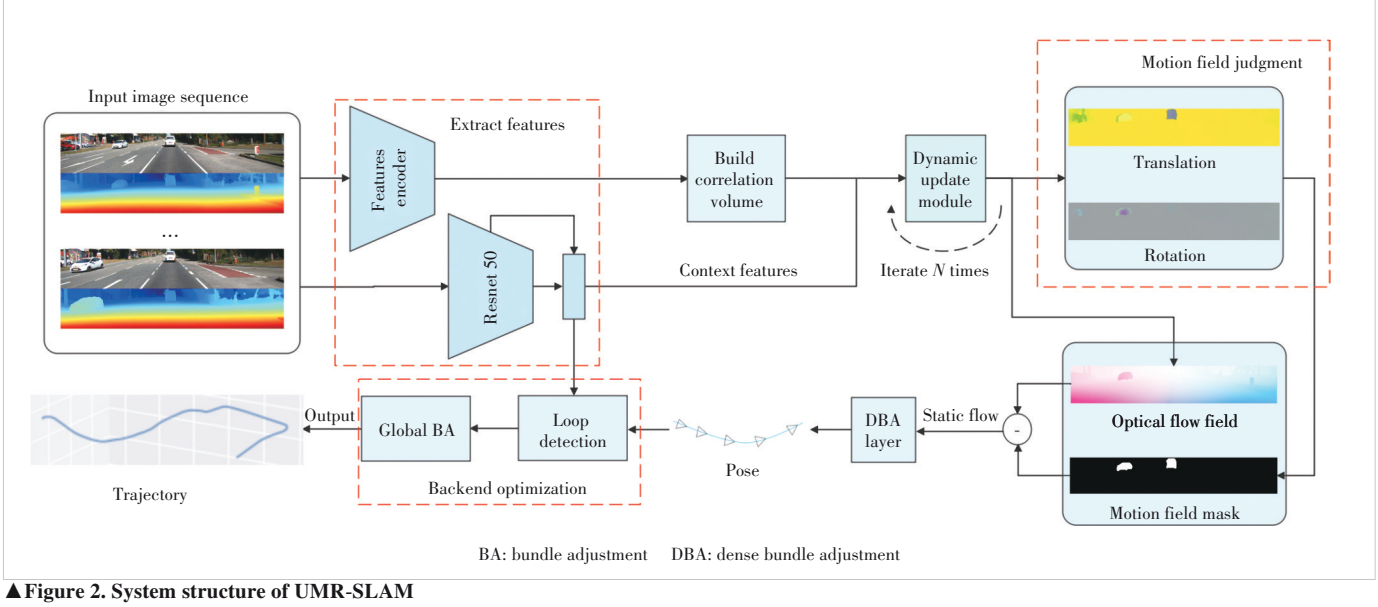
Backend optimization is a key step in SLAM systems to improve the accuracy of positioning and mapping. Usually, graph optimization or nonlinear optimization techniques are used to minimize estimation errors. Closed loop detection is the key to optimizing the backend of SLAM systems. Closed loop detection or position recognition is also an important module for reducing trajectory errors in the SLAM backend. The traditional loop detection scheme is based on a bag of words (BoW)[25] for storage and uses manually designed visual features. The BoW method first extracts features, including scale invariant feature transform (SIFT), speeded-up robust features (SURF), ORB, etc., from a large number of training images, and classifies these features (Word) with K-means clustering algorithms to obtain leaf nodes called dictionaries. Therefore, an image can be described as a vector under the dictionary based on whether the corresponding word (Word) appears. However, changes in lighting, weather, viewpoints, and moving objects in real-world scenes makes this problem more complex. Different from traditional word bag-based methods, deep networks can typically learn complex internal structures in image data without manual design of visual features.

To address this issue, previous research works like Ref. [26] use ConvNet features that are more robust to changes in viewpoints and conditions and derived from pretrained models on a universal large-scale image processing dataset. This scheme can predict the matching landmark candidate boxes between images and extract features. To improve the algorithm's efficiency, Gaussian random projection (GRP) is used to reduce the data dimension for feature similarity calculation. However, for high-dimensional data with partially non-uniform distribution, using GRP is not conducive to preserving the original variance.

Other representative works[27-29] are based on deep automatic encoder structures to extract compact representations that compress scenes unsupervisedly.

3 Proposed Method

Our proposed simultaneous localization and mapping technology for unsupervised motion removal (UMR-SLAM) structure is shown in Fig. 2. We input a set of RGBD image sequences and use encoders to extract features and context features respectively. By calculating the correlation volume through the feature dot product, the update operator iteratively updates the pose changes of each pixel, and calculates the optical flow and dynamic region mask based on the pixel-by-pixel pose. After removing the optical flow from the dynamic region, the camera pose is obtained through bundle adjustment (BA) optimization. Finally, in the backend optimization, the global pose and trajectory are optimized based on the loop detection results to reduce cumulative errors. This method takes RGBD image sequences as input and outputs camera pose. UMR-SLAM has an end-to-end differentiable architecture, which combines the advantages of classical methods and deep learning networks. We use the scene



▲Figure 2. System structure of UMR-SLAM

flow method to unobservedly remove dynamic objects, and integrate the results of the two detection schemes, optical flow and feature similarity in loop detection, making the SLAM system more robust in dealing with challenging dynamic scenes. Specifically, distinguished from DROID-SLAM, which iteratively updates camera pose and depth, we iteratively update the poses of all pixels. In DROID-SLAM, the optical flow is used to perform every update of camera poses, while we only use the optical flow caused by camera motion to calculate camera poses. Next, we elaborate on the details of our method.

3.1 Network Architecture

The core of the proposed dynamic SLAM network is to use a dynamic update module to estimate the 3D rigid body motion of all pixels in the scene, and then calculate the optical flow and dynamic region. The camera pose is optimized through BA. Compared with DROID-SLAM, which uses optical flow as the intermediate motion representation, we can determine the pixels of dynamic objects by unsupervised learning technology and only use the static optical flow caused by camera motion to calculate camera pose, which can greatly improve the positioning accuracy of the algorithm in dynamic scenes.

In contrast to DROID-SLAM, we directly input the RGBD images for feature extraction and optical flow calculation. For each pair of RGBD images, in the dynamic update module, we iteratively update the 3D motion of all pixels, rather than the camera pose and inverse depth map. Our update iteration not only runs on adjacent frames but can be applied to any number of frames to obtain higher accuracy scene flow information and achieve joint global refinement of all camera poses, which helps to minimize long trajectories and loop drift. In the backend optimization, we also use the Gaussian Newton method to execute BA to adjust the camera pose T to minimize the cost function.

We use a frame graph to represent the covisibility between frames. We determine whether two frames are coviewed through optical flow and establish a coview frame map. Differing from DROID-SLAM, we use the completed depth map instead of the original one to calculate the frame graph. In the coview, nodes represent each input image, and edges, which are the connections between nodes, indicate that the two images are covisible. During the training and inference process of the model, the frame graph is dynamically constructed and updated. Each time a new optical flow is calculated to remove dynamic objects, the frame map with new visibility is updated.

3.1.1 Features Extraction

In the feature extraction module, we use conventional residual modules and downsampling convolution modules to obtain high-dimensional dense feature maps with a resolution of 1/8 of the original image. At the same time, we use pretrained ResNet50 with skip connections to extract context features at 1/8 resolution. ResNet50 can extract features with a greater degree of semantic information and a larger receptive field, which can be better used for loop detection and rigid motion object grouping.

3.1.2 Building Correlation Volume

1) Correlation pyramid: For two frames I_i and I_j with a common view, the correlation volume C is calculated using the dot product of two position feature vectors in the feature map f, as shown in the following equation.

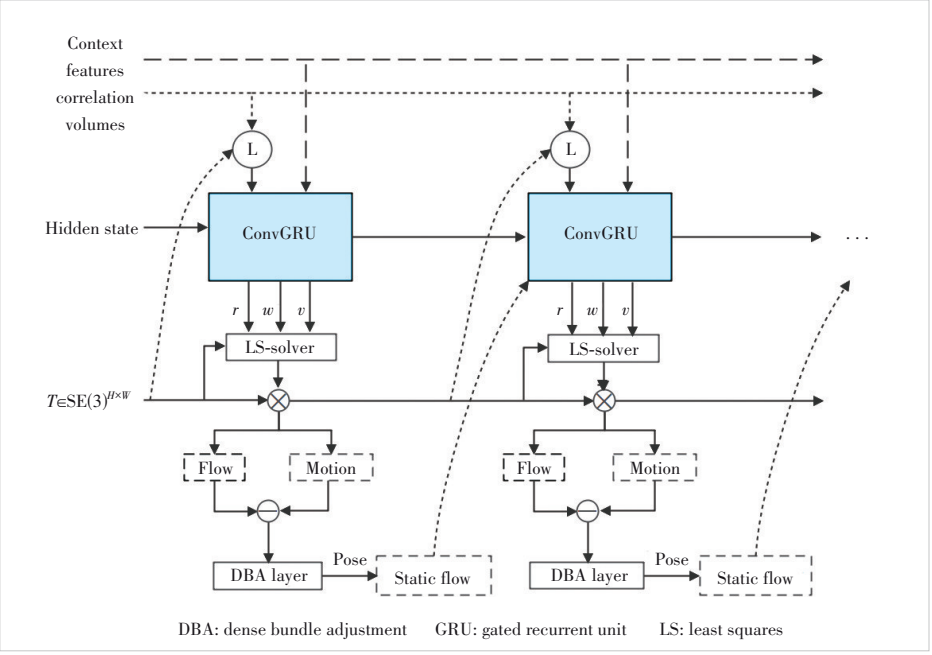
$$C^{ij}_{u_1v_1u_2v_2} = f(I_i)_{u_1v_1} \cdot f(I_j)_{u_2v_2}, \tag{1}$$

where $C^{i,j}_{u_1,v_1,u_2,v_2} \in R^{H \times W \times H \times W}$ represents the correlation between the features of image I_i at position (u_1, v_1) and image I_j at position (u_2, v_2). Then we use average pooling concatenation to establish a four-layer correlation pyramid.

2) Correlation lookup: The lookup operator is L_r: $R^{H \times W \times H \times W} \times R^{H \times W \times 2} \rightarrow R^{H \times W \times (r+1)^2}$. This operator uses bilinear interpolation to index the relevant volume using an optical flow field coordinate grid with a radius of r. Splice the relevant features found at each layer of the relevant pyramid into a feature vector.

3.1.3 Dynamic Update Module

Fig. 3 shows the dynamic update module of UMR-SLAM. We find the relevant features of the optical flow calculated by the current pixel pose through the relevant volume. The obtained features are fed together with static optical flow and global feature dynamics into two convolutional layers, resulting in intermediate features. These features are then fed into convolutional gated recurrent unit (ConvGRU), and then optical flow residuals, their confidence levels, and rigid motion embedding vectors are obtained through convolutional layers. The dense pixel pose T can be updated using the least squares method. According to the dense pose calculation, the dynamic region of the optical flow is deducted and fed into the DBA layer, which combines the optical flow confidence to optimize the camera pose. Finally, the optimized camera pose is used to calculate optical flow and provided for the next iteration. Not similar to the iterative operation of the update module in DROID-SLAM, the update operator combines neural networks and optimization algorithms to update the dense pixel pose, and then performs subsequent optical flow and camera pose calculations based on the pixel pose. The update operator is based on the ConvGRU of recurrent neural networks (RNN) for iterative updates. The optical flow and the pixel density pose are fed into the next iteration as new optimization terms. During each iteration of the update process, the module generates dense pixel pose increments, optical flow generated by cam-



▲**Figure 3. Dynamic update module**

era motion, dynamic object masks, and camera pose.

1) Update operator: The update operator is a GRU unit based on a recurrent neural network, mainly composed of 3×3 convolutional kernels with dilation rates of 1 and 3. It uses index operators to retrieve features from correlated volumes and output optical flow correction quantities. We use depth maps and the current estimated pose to estimate the 2D correspondence. Taking the edges in the frame graph $(i,j) \in \varepsilon$ as an example, p_i is the grid coordinate of frame i. p_{ij} is the corresponding projection coordinate of frame i in frame j, and the projection transformation process is as follows:

$$p_{i,j} = \left(\boldsymbol{T}s_{i,j} \cdot \prod_c^{-1}(p_i, d_i) \right), \tag{2}$$

where $\prod_c$ is a pinhole camera model that maps a set of 3D points to the image, while $\prod_c^{-1}$ is the inverse projection function that maps the inverse depth map d_i and p_i to the 3D point cloud. The pose transformation matrix between image i and image j is $\boldsymbol{T}s_{i,j}$. We can obtain p_{ij} through the transformation of 3D points in the world coordinate system and the pinhole camera model.

Based on the corresponding relationship, we can retrieve correlation features in the correlation volume and calculate the optical flow field $p_{ij} - p_j$. The input of GRU includes optical flow field, current dense pixel pose, depth residual $d' - d^*$, and correlation features. In the depth residual term, the inverse depth d' is the depth component of p_{ij}, and d^* is the inverse depth map of p_{ij} index image frame j. Each feature is extracted with high-dimensional information through two convolutional layers, and then fed into the GRU module.

Then, three two-layer convolutions are applied to the hidden states of the GRU output to calculate the rigid motion embedding map V, the revision map optical flow field correction map $r = (rx, ry, rz)$, and the corresponding confidence $w \in [0, 1]$. The correction amount r is the correction of the optical flow caused by the current SE3 field. Three outputs serve as inputs to the dense-SE3 layer to generate updates to the SE3 sports field. The confidence level of w as the optical flow correction is used to calculate the cost function.

The resolution of the SE3 motion field estimated by the network is 1/8 of the original image resolution. To obtain the original resolution map, we perform convex upsampling in Lie Algebra and then use exponential mapping back to the manifold.

2) Dense-SE3 layer: This layer is a differentiable optimization layer used to update the current pose of pixels. It maps the optical flow revision mapping r to the SE3 field update. The rigid motion embedding vector $\boldsymbol{v}$ is used to soft-group pixels into rigid objects. We use embedding vectors to build an attention matrix between all pairs of (i,j). We calculate the similarity $a_{ij} \in [0, 1]$ between two embedding vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ by taking the sigmoid activation function σ of the negative L2 distance.

$$a_{ij} = 2*\sigma\left(-\left\| \boldsymbol{v}_i - \boldsymbol{v}_j \right\|^2 \right) \in [0,1]. \tag{3}$$

We use similarity to define an objective function based on reprojection error to solve the updated pose δ_i for each pixel i:

$$E(\delta) = \sum_{i \in \Omega} \sum_{j \in N_i} a_{ij} \left\| r_j + \prod_c (T_j X_j) - \prod_c (e^{\delta_i} T_i X_j) \right\|_{w_j}^2, \tag{4}$$

where $\| X \|_w^2 = X^T \mathrm{diag}(w) X$. The above equation indicates that for each pixel i in the image area Ω, the transformation T_i is described as a transformation of pixel j in the neighborhood N_i of pixel i. Only objects with similar embedding vectors that may belong to the same rigid motion as (i, j) have a significant contribution to the objective function. To reduce the memory footprint when solving Eq. (4), we implement Gaussian Newton updates in CUDA to estimate the next SE3 pose.

3) Motion field judgment: Moving objects in the image

greatly affects the calculation of camera pose. Therefore, it is very necessary to remove dynamic objects. We filter the rotation and translation of each pixel predicted by the scene flow algorithm as shown in Figs. 4b and 4c, and if the pose τ_i and ϕ_i of pixel i differ from the average motion pose τ_{mean} and ϕ_{mean} of the entire image by more than a certain threshold (μ set to 0.01), it is regarded as a motion point as shown in Fig. 4d. We set the optical flow mask M of the moving point directly to 0, without performing subsequent camera pose estimation.

$$M_i^{\text{motion}} = [\|\tau_i - \tau_{\text{mean}}\| \geqslant \mu_\tau] + [\|\varphi_i - \varphi_{\text{mean}}\| \geqslant \mu_\varphi]. \tag{5}$$

4) DBA: After obtaining the corrected static flow field, we use the Gaussian Newton-based dense bundle adjustment (DBA) layer algorithm in DROID-SLAM to optimize the camera pose G. The DBA layer does not affect gradient backpropagation. The error function is defined as follows:

$$E(G') = \sum_{(ij)\in\varepsilon} \left\| p_{ij}^* - \prod_c (G'_{ij} \cdot \prod_c^{-1}(p_i, d'_i)) \right\|_{\Sigma_{ij}}^2, \tag{6}$$

$$\Sigma_{ij} = \text{diag} w_{ij}, \tag{7}$$

where $p_{ij}^* = r_{ij} + p_{ij}$ represents the updated and corrected p_{ij}. Eq. (7) calculates the Mahalanobis distance weight, and the error term is weighted based on the combined confidence w_{ij}. The $\boldsymbol{H}\Delta x = g$ problem can be solved by Schur elimination using the sparsity property of matrix $\boldsymbol{H}$ to accelerate the solution process.

3.2 Supervision

We use pose loss and flow loss supervision to train our network. Both loss functions are applied to paired training sequences. We calculate the static optical flow f_{static} based on the camera pose predicted through each iteration. The optical flow calculated from the true depth and camera pose truth is used as the supervisory information f_{gt}.

$$f_{\text{static}}^k = \prod_c \left(G \cdot \prod_c^{-1}(p) \right) - p, \tag{8}$$

where G represents the camera pose, p represents the image coordinate grid, and k represents the number of iterations. We design the loss as the average L2 distance between two optical flow fields.

$$L_{\text{flow}} = \sum_{k=1}^{N} \gamma^{N-k} \left\| f_{\text{static}}^k - f_{gt} \right\|_2, \tag{9}$$

with $\gamma = 0.9$. We also apply an additional loss function to the GRU-predicted optical flow increment and set the weight to 0.2.

Pose loss uses the actual pose T and the predicted camera pose G after each update to calculate the loss.

$$L_{\text{pose}} = \sum_{k=1}^{N} \gamma^{N-k} \left\| \text{Log}_{\text{SE3}}(T^{-1} \cdot G_k) \right\|_2. \tag{10}$$

The overall loss function is the sum of pose loss and optical flow loss. To ensure that the two types of losses are on the same order of magnitude, we use coefficients w_1 and w_2 to adjust the weights of the two types of losses. We set w_1 to 0.1 and w_2 to 1.

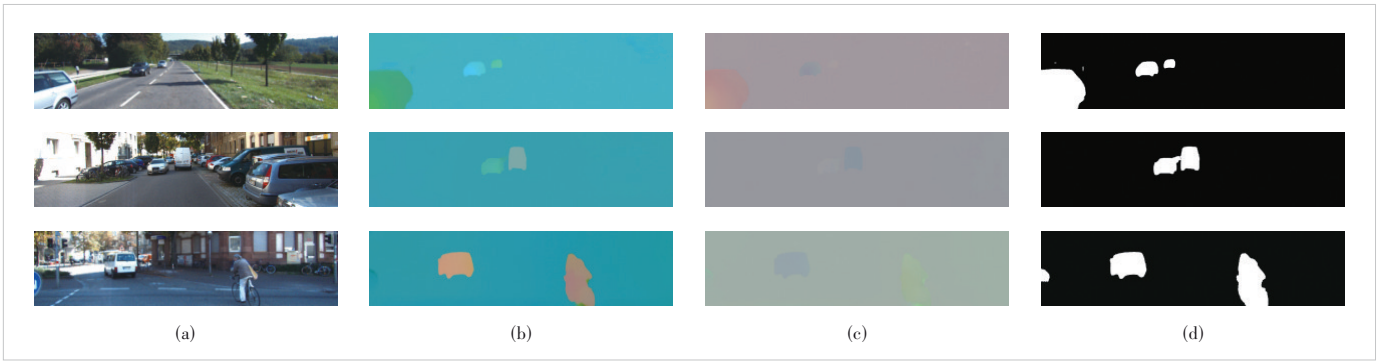
$$L = w_1 L_{\text{flow}} + w_2 L_{\text{pose}}. \tag{11}$$

3.3 SLAM System

In terms of input modes, our SLAM system only supports RGBD data input. In the inference stage, we embed the above network structure into the entire SLAM system.

During initialization, our algorithm accumulatively receives 12 keyframes based on optical flow differences, constructs frame maps for them, and uses our dynamic update module to calculate their initial pose. In addition, we process on the depth image.

1) Depth image estimation: The importance of depth information has been well demonstrated by a large amount of re-



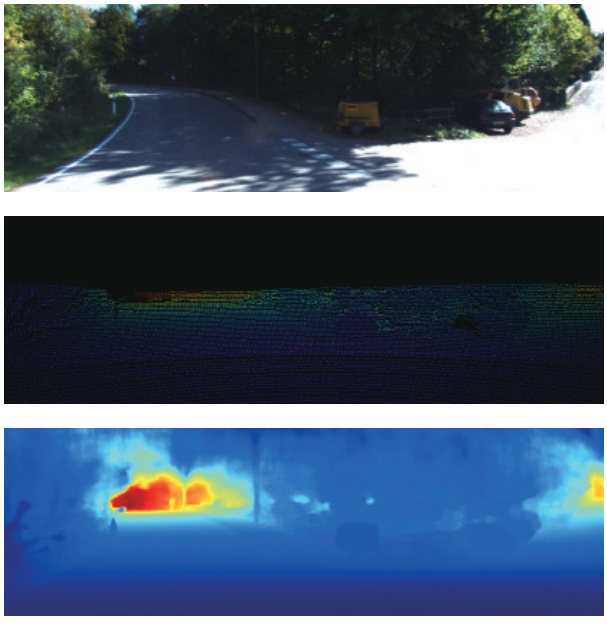
▲**Figure 4. (a) Original image; (b) visualization image of translation amount;(c) visualization image of rotation amount; (d) dynamic region mask**

search work in the past. Usually, we use depth sensors to obtain accurate and reliable distance measurements, while also possessing real scene scales. However, neither LiDAR nor other commonly used RGBD cameras can provide dense pixellevel depth maps. The holes and blank pixels in sparse depth maps indicate a serious lack of information at the application level, leading to algorithm reliability failure. Therefore, it is necessary to fill in these blank pixels in practical applications.

Specifically, for the KIITI dataset, we directly obtain laser data from the odometer dataset and convert it into depth images. Obviously, as shown in Fig. 5, the lack of depth information is very severe, with only 5% of the available pixel points in the entire image. Here, we use the CompletionFormer[31] algorithm to complete the depth map. Before and after processing, as shown in Fig. 5, we also attempt to use the GA-net[32] algorithm to estimate disparity maps using binocular data and obtain depth maps.

In front-end processing, when a new frame arrives, the system uses the three closest adjacent frames and the new frame to create a temporary graph, and optimize and calculate the pose of the new frame.

In backend optimization, the system creates a new frame graph containing each reserved keyframe. The edges between key frames are generated according to specific rules to eliminate excess edges. We use a dynamic update module to optimize the final pose of the entire shape. Then, through the proposed loop detection scheme, we directly add edges between the two frames with loops to the frame graph for subsequent global BA optimization.



▲Figure 5. RGB, original laser depth map, and completed depth map: from top to bottom

2) Loop detection: In backend optimization, closed-loop detection is used to detect and correct path drift. It identifies features that appear on previously visited locations and uses this information to adjust device location estimates and maps to reduce cumulative errors. We found that DROID-SLAM uses optical flow for loop detection, which often fails to detect loops well in real-world scenarios and has poor robustness. Using only optical flow to determine whether a loop exists may lead to missed detection.

In UMR-SLAM, we propose a solution that combines high-dimensional image feature similarity with optical flow calculation to improve loop detection accuracy. Image feature similarity schemes typically handle image noise more robustly, especially in scenarios with lighting changes, occlusions, or other complex environments, where similarity is more reliable than solely relying on optical flow. Compared with optical flow, which focuses only on local motion information, image feature similarity schemes compare the content of entire images, capturing similarity between images from a global perspective. This can compensate for the limitation of local motion and improve matching accuracy. Combining image feature similarity schemes with optical flow calculation provides additional information to validate the existence of loops. For example, if optical flow calculation detects significant motion but the image feature similarity is high, it may indicate environmental similarity, allowing for more confident loop detection and reducing missed detection. Integrating these two types of information enhances loop detection accuracy and robustness, particularly in situations where optical flow calculation may be affected by noise or encounters significant motion. Therefore, we add a branch to the entire SLAM system for loopback detection. We directly utilize the global context Resnet50 features of existing keyframes for feature similarity comparison.

To improve the algorithm's efficiency, we use the principal component analysis (PCA) for feature dimensionality reduction before feature similarity comparison. PCA can identify the intrinsic patterns of data based on the relationship between features. By calculating the eigenvalues of the covariance matrix and corresponding eigenvectors, the direction of maximum variance is found in high-dimensional data, and the data are mapped to a new subspace with a dimensionality not greater than the original data. We use the feature maps of the first 128 channels for PCA dimensionality reduction to 128×30.

Next, cosine distance d is used to measure the distance between two features for loop detection.

$$d_{\cos} = \cos(\boldsymbol{v}_1, \boldsymbol{v}_2), \tag{12}$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ represent the feature vector expressions of the

image pair. If it is less than a certain threshold τ (set to 0.12), it is considered that a loopback has been detected.

4 Experiments

Initially, UMR-SLAM is trained on datasets Virtual KITTI2[33] and KITTI[34], followed by a comprehensive evaluation of the methodologies on various real and synthetic datasets, encompassing dynamic sequences from Virtual KITTI2, KITTI, and TUM-RGBD[35]. During the experimentation phase, the absolute trajectory error (ATE) serves as the metric for assessing the accuracy of the estimated camera trajectory. Subsequently, a series of ablation experiments are devised to validate the efficacy of the proposed method in dynamic scenarios. Afterward, comparisons are drawn between our method and other advanced algorithms in dynamic scenarios, such as ORB-SLAM2[36], DROID-SLAM, and DynaSLAM[37], to showcase the effectiveness of our method and the robustness of pose estimation.

4.1 Datasets

We train and test using partial Virtual KITTI2 and KITTI datasets. KITTI is a dataset captured in real-world traffic conditions, ranging from highways in rural areas to city-center scenes with many static and dynamic objects. The KITTI dataset is typically used as a benchmark test set for stereo vision, optical flow, depth prediction, object detection, and visual mileage calculation methods. We mainly use 00 – 08 of the visual odometer data as the training set, and 09 and 10 as the test set.

Virtual KITTI2 is a synthetic dataset modeled after the KITTI dataset, consisting of 5 sequences. These sequences enhance data by overlaying different weather conditions (such as fog and rain) and modifying camera directions and angles. In the ablation study, we use the default camera orientation as the training set and configurations of 15 and 30 degrees as the validation set.

4.2 Training Implementation Details

We use 8 NVIDIA GPU V100 for training. Considering the possibility of inaccurate depth completion values in the sky and limited graphics memory, we randomly crop 208×960 sized images below the images for training, while modifying the internal parameter data.

We make minor adjustments to KITTI to perform an additional 50k iterations with an initial learning rate of 5×10^{-5}, and perform spatial and photometric enhancements. To estimate parallax, we provide input depth maps for our method using GA-Net.

For all experiments, we use the AdamW optimizer with weight attenuation set to 1×10^{-5}, and expand the update operator for 12 iterations. We use partial model weights from ImageNet and RAFT-3D as pretrain weights. Training RAFT-3D involves differentiating a computational graph composed of Euclidean tensors (such as network weights and feature activation) and Lie group elements (such as the SE3 transformation domain). We use the LieTorch library to perform backpropagation in the tangent space of manifold elements in computational graphs.

Adjusting the weights of the two loss functions simultaneously to make their order of magnitude similar. Due to the fast camera movement in the KITTI dataset, we optimize the optical flow filtering threshold and expand the range of optical flow selection when establishing frame maps before training. For pixels with missing depth, we directly assign their depth value to 0.01, resulting in an inverse depth of 100. This point is not considered when calculating the overall optical flow, and the mask is set to 0.

4.3 Results

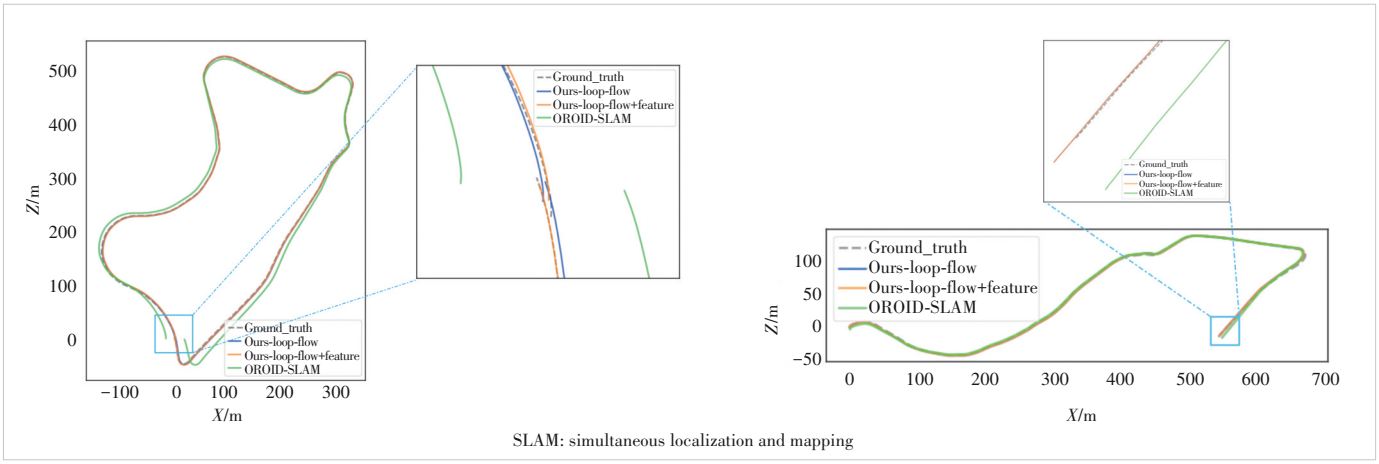
In this section, we compare our proposed approach with current state-of-the-art methods on the main-stream SLAM datasets.

1) Ablations experiment: We conduct ablation experiments on various components of the UMR-SLAM model on the Virtual KITTI2 and KITTI datasets, and report the results in Table 1. We compare the indicators using different depth completion methods and also provide indicators on whether to use loopback detection and whether to use dynamic object removal. The indicators in Table 1 are all tested using RGBD data, and the test indicator is automatic test equipment absolute trajectory error (ATE)[M] (RMSE).

We use different depth estimation algorithms to test the final algorithm metrics. The PENet and CompletionFormer methods both complete the depth map converted by laser, while GA-net calculates the disparity map using binoculars and then uses camera internal references to obtain the depth map. We found that the depth maps obtained by GA-net can provide better performance. We also test the impact of deducting dynamic objects on SLAM, and the experiment shows that the ATE index after deducting dynamic objects would decrease by about 2.5. We also attempt to use differ-

▼**Table 1. Ablations experiment of UMR-SLAM, where the best results are displayed in bold**

Experiment	Configuration		K09	K10
Depth estimate	-		11.527	4.775
	PENet		4.413	3.366
	CompletionFormer		3.569	2.748
	GA-net		**2.689**	**1.414**
Dynamic region removal	No		5.131	2.351
	Yes		**2.689**	**1.414**
Loop detection	-		4.058	**1.412**
	Flow		3.835	1.414
	Flow and feature	τ=0.5	3.665	1.414
		τ=1.2	**2.689**	1.420
		τ=2	3.872	1.427



▲Figure 6. Trajectory comparison between our method and different loop detection algorithms and DROID-SLAM in KITTI sequences 09 (Left) and 10 (Right)

ent detection schemes in loop detection, with the highest accuracy achieved when optical flow is used together with features and the feature similarity threshold is 1.2. Fig. 6 shows a trajectory comparison between our UMR-SLAM and the DROID SLAM algorithm, and the results indicate that our trajectories are closer to the ground truth.

We test the performance of the proposed UMR-SLAM on sequences 09 and 10 of the KITTI dataset and all sequences of the Virtual KITTI2 dataset, and provide camera motion trajectories. The ATE results are shown in Table 2 below. Compared with DROID-SLAM, our UMR-SLAM is more accurate and robust in dynamic scenarios. We also evaluate TUM RGBD dynamic sequences with different dynamic ratios, and the comparison results in Table 3 indicate that UMR-SLAM achieves competitive and even the best performance compared with other classical methods such as DVO-SLAM, ORB-SLAM2, PointCorr, DROID-SLAM. All methods in the table are tested using the RGBD dataset.

5 Conclusions

In this paper, we introduce UMR-SLAM, an end-to-end visual SLAM algorithm. We combine scene flow with deep learning SLAM to improve SLAM accuracy in dynamic scenes, without the need for explicit object segmentation as supervisory information throughout the entire process. In the backend optimization section, we propose a loop detection scheme that combines optical flow and feature similarity, which can improve the accuracy of loop detection. The results of experiments on different datasets prove that our scheme has higher accuracy compared with the current state-of-the-art deep learning scheme, DROID-SLAM, especially in dynamic scenarios. Overall, the flexibility of deep learning and powerful feature extraction capabilities provide new solutions to SLAM systems, which can cope with various complex environments and tasks. However, the integration of deep learning and SLAM still faces significant challenges in real-time performance and computational complexity, and further research and innovative methods need to be sought to address them.

▼Table 2. ATE [M] metric for dynamic SLAM on the KITTI (K) and Virtual KITTI2 (VK) datasets, where we achieve the best results. All test results are based on RGBD

Method	K09	K10	VK01	VK02	VK06	VK18	VK20
DROID-SLAM	5.453	2.514	0.197	0.192	0.007	1.030	3.041
Our URM-SLAM	**2.689**	**1.414**	**0.128**	**0.030**	**0.007**	**0.812**	**1.189**

SLAM: Simultaneous localization and mapping

▼Table 3. Dynamic SLAM results of TUM dynamic sequences, measured as ATE [M]. The best results are displayed in bold

Method	Input Modes	DVO-SLAM[38]	ORB-SLAM2	PointCorr[39]	DROID-SLAM	Ours
Slightly dynamic	fr2/desk-person	0.104	**0.006**	0.008	0.019	0.014
	fr3/sitting-static	0.012	0.008	0.010	**0.006**	0.007
	fr3/sitting-xyz	0.242	0.010	**0.009**	0.011	**0.009**
	fr3/sitting-rpy	0.176	0.025	0.023	0.022	**0.020**
	fr3/sitting-halfsphere	0.220	0.025	0.024	0.023	**0.022**
Highly dynamic	fr3/walking-static	0.752	0.408	0.011	0.007	**0.004**
	fr3/walking-xyz	1.383	0.722	0.087	0.015	**0.013**
	fr3/walking-rpy	1.292	0.805	0.161	0.050	**0.045**
	fr3/walking-half-sphere	1.014	0.723	0.035	**0.029**	0.032

SLAM: simultaneous localization and mapping

References

[1] ZHONG F W, WANG S, ZHANG Z Q, et al. Detect-SLAM: making object detection and SLAM mutually beneficial [C]//Proceedings of IEEE

Winter Conference on Applications of Computer Vision (WACV). IEEE, 2018: 1001 – 1010. DOI: 10.1109/WACV.2018.00115
[2] WU W X, GUO L, GAO H L, et al. YOLO-SLAM: a semantic SLAM system towards dynamic environment with geometric constraint [J]. Neural computing and applications, 2022, 34(8): 6011 – 6026. DOI: 10.1007/s00521-021-06764-3
[3] YU C, LIU Z X, LIU X J, et al. DS-SLAM: a semantic visual SLAM towards dynamic environments [C]//International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 1168 – 1174. DOI: 10.1109/IROS.2018.8593691
[4] LIU Y B, MIURA J. RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods [J]. IEEE access, 2021, 9: 23772 – 23785. DOI: 10.1109/ACCESS.2021.3050617
[5] BESCOS B, FÁCIL J M, CIVERA J, et al. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes [J]. IEEE robotics and automation letters, 2018, 3(4): 4076 – 4083. DOI: 10.1109/LRA.2018.2860039
[6] WANG C J, LUO B, ZHANG Y, et al. DymSLAM: 4D dynamic scene reconstruction based on geometrical motion segmentation [J]. IEEE robotics and automation letters, 2021, 6(2): 550 – 557. DOI: 10.1109/LRA.2020.3045647
[7] SUN Y X, LIU M, MENG Q H. Motion removal for reliable RGB-D SLAM in dynamic environments [J]. Robotics and autonomous systems, 2018, 108: 115 – 128. DOI: 10.1016/j.robot.2018.07.002
[8] DAI W C, ZHANG Y, LI P, et al. RGB-D SLAM in dynamic environments using point correlations [J]. IEEE transactions on pattern analysis and machine intelligence, 2022, 44(1): 373 – 389. DOI: 10.1109/TPAMI.2020.3010942
[9] YUAN C F, XU Y L, ZHOU Q. PLDS-SLAM: point and line features SLAM in dynamic environment [J]. Remote sensing, 2023, 15(7): 1893. DOI: 10.3390/rs15071893
[10] ZHANG J, HENEIN M, MAHONY R, et al. VDO-SLAM: a visual dynamic object-aware SLAM system [EB/OL]. (2020-05-22) [2021-12-14]. http://arxiv.org/abs/2005.11052
[11] CHO H M, KIM E. Dynamic object-aware visual odometry (VO) estimation based on optical flow matching [J]. IEEE access, 1961, 11: 11642 – 11651. DOI: 10.1109/ACCESS.2023.3241961
[12] YE W C, LAN X Y, CHEN S, et al. PVO: panoptic visual odometry [C]//Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2023: 9579 – 9589. DOI: 10.1109/CVPR52729.2023.00924
[13] SHEN S H, CAI Y L, WANG W S, et al. DytanVO: joint refinement of visual odometry and motion segmentation in dynamic environments [C]//International Conference on Robotics and Automation (ICRA). IEEE, 2023: 4048 – 4055. DOI: 10.1109/ICRA48891.2023.10161306
[14] DOSOVITSKIY A, FISCHER P, ILG E, et al. FlowNet: learning optical flow with convolutional networks [C]//International Conference on Computer Vision (ICCV). IEEE, 2015: 2758 – 2766. DOI: 10.1109/ICCV.2015.316
[15] ILG E, MAYER N, SAIKIA T, et al. FlowNet 2.0: evolution of optical flow estimation with deep networks [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017: 1647 – 1655. DOI: 10.1109/CVPR.2017.179
[16] JONSCHKOWSKI R, STONE A, BARRON J T, et al. What matters in unsupervised optical flow [M]//Lecture notes in computer science. Cham: Springer International Publishing, 2020: 557 – 572. DOI: 10.1007/978-3-030-58536-5_33
[17] RANJAN A, BLACK M J. Optical flow estimation using a spatial pyramid network [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017: 2720 – 2729. DOI: 10.1109/CVPR.2017.291
[18] SUN D Q, YANG X D, LIU M Y, et al. PWC-net: CNNs for optical flow using pyramid, warping, and cost volume [C]//Conference on Computer Vision and Pattern Recognition. IEEE, 2018: 8934 – 8943. DOI: 10.1109/CVPR.2018.00931
[19] TEED Z, DENG J. RAFT: recurrent all-pairs field transforms for optical flow [M]//lecture notes in computer science. Cham: Springer International Publishing, 2020: 402 – 419. DOI: 10.1007/978-3-030-58536-5_24
[20] TEED Z, DENG J. RAFT-3D: Scene Flow using Rigid-Motion Embeddings [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021: 8371 – 8380. DOI: 10.1109/CVPR46437.2021.00827
[21] REN Z L, GALLO O, SUN D Q, et al. A fusion approach for multi-frame optical flow estimation [C]//IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2019: 2077 – 2086. DOI: 10.1109/WACV.2019.00225
[22] SHI H, ZHOU Y F, YANG K L, et al. CSFlow: learning optical flow via cross strip correlation for autonomous driving [C]//Intelligent Vehicles Symposium (IV). IEEE, 2022: 1851 – 1858. DOI: 10.1109/IV51971.2022.9827341
[23] GARREPALLI R, JEONG J, RAVINDRAN R C, et al. DIFT: dynamic iterative field transforms for memory efficient optical flow [C]//Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, 2023: 2220 – 2229. DOI: 10.1109/CVPRW59228.2023.00216
[24] HUI T W, TANG X O, LOY C C. A lightweight optical flow CNN: revisiting data fidelity and regularization [J]. IEEE transactions on pattern analysis and machine intelligence, 2021, 43(8): 2555 – 2569. DOI: 10.1109/TPAMI.2020.2976928
[25] GALVEZ-LÓPEZ D, TARDOS J D. Bags of binary words for fast place recognition in image sequences [J]. IEEE transactions on robotics, 2012, 28(5): 1188 – 1197. DOI: 10.1109/TRO.2012.2197158
[26] SUENDERHAUF N, SHIRAZI S, JACOBSON A, et al. Place recognition with ConvNet landmarks: viewpoint-robust, condition-robust, training-free [C]//Proceedings of Robotics: Science and Systems XI. Robotics: Science and Systems Foundation, 2015: 1 – 10. DOI: 10.15607/rss.2015.xi.022
[27] GAO X, ZHANG T. Unsupervised learning to detect loops using deep neural networks for visual SLAM system [J]. Autonomous robots, 2017, 41(1): 1 – 18. DOI: 10.1007/s10514-015-9516-2
[28] MERRILL N, HUANG G Q. Lightweight unsupervised deep loop closure [EB/OL]. (2018-05-24) [2023-10-10]. http://arxiv. org/abs/1805.07703
[29] MEMON A R, WANG H S, HUSSAIN A. Loop closure detection using supervised and unsupervised deep neural networks for monocular SLAM systems [J]. Robotics and autonomous systems, 2020, 126: 103470. DOI: 10.1016/j.robot.2020.103470
[30] TEED Z, DENG J. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras [J]. Advances in neural information processing systems, 2021, 34: 16558 – 16569
[31] ZHANG Y M, GUO X D, POGGI M, et al. CompletionFormer: depth completion with convolutions and vision transformers [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2023: 18527 – 18536. DOI: 10.1109/CVPR52729.2023.01777
[32] ZHANG F H, PRISACARIU V, YANG R G, et al. GA-net: guided aggregation net for end-to-end stereo matching [C]//Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019: 185 – 194. DOI: 10.1109/CVPR.2019.00027
[33] CABON Y, MURRAY N, HUMENBERGER M. Virtual KITTI 2 [EB/OL]. (2020-01-29) [2023-10-10]. http://arxiv.org/abs/2001.10773
[34] GEIGER A, LENZ P, STILLER C, et al. Vision meets robotics: the KITTI dataset [J]. The international journal of robotics research, 2013, 32(11): 1231 – 1237. DOI: 10.1177/0278364913491297
[35] SCHUBERT D, GOLL T, DEMMEL N, et al. The TUM VI benchmark for evaluating visual-inertial odometry [C]//International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 1680 – 1687. DOI: 10.1109/IROS.2018.8593419
[36] MUR-ARTAL R, TARDÓS J D. ORB-SLAM2: An open-source SLAM

system for monocular, stereo, and RGB-D cameras [J]. IEEE transactions on robotics, 2017, 33(5): 1255 – 1262. DOI: 10.1109/TRO.2017.2705103
[37] BESCOS B, FÁCIL J M, CIVERA J, et al. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes [J]. IEEE robotics and automation letters, 2018, 3(4): 4076 – 4083. DOI: 10.1109/LRA.2018.2860039
[38] KERL C, STURM J, CREMERS D. Robust odometry estimation for RGB-D cameras [C]//IEEE International Conference on Robotics and Automation. IEEE, 2013: 3748 – 3754. DOI: 10.1109/ICRA.2013.6631104
[39] DAI W C, ZHANG Y, LI P, et al. RGB-D SLAM in dynamic environments using point correlations [J]. IEEE transactions on pattern analysis and machine intelligence, 2022, 44(1): 373 – 389. DOI: 10.1109/TPAMI.2020.3010942
[40] YE W C, YU X Y, LAN X Y, et al. DeFlowSLAM: self-supervised scene motion decomposition for dynamic dense SLAM [EB/OL]. [2023-10-10]. https://arxiv.org/pdf/2207.08794v2

Biographies

CHEN Hao (chen.hao16@zte.com.cn) received his BS and MS degrees in control theory and control engineering from Harbin Engineering University, China in 2018 and 2020. He has been engaged in deep learning technologies in ZTE Corporation since his graduation. His research interests include digital humans, SLAM, and image recognition.

ZHANG Kaijiong received his MS degree from Shanghai Jiao Tong University, China in 2020. He is currently an algorithm engineer with ZTE Corporation. His research interests include computer vision, image/video processing and artificial intelligence.

CHEN Jun received his master's degree in aerospace science and technology from Nanjing University of Aeronautics and Astronautics, China. He has been engaged in the R&D of computer graphics, computer vision, and cloud computing for more than 10 years in ZTE Corporation, and has accumulated rich experience in solution and engineering.

ZHANG Ziwen received his bachelor's degree in instrument science and technology and master's degree in instrument engineering from Harbin Institute of Technology, China in 2018 and 2020 respectively. After graduation, he worked at ZTE Corporation as a computer vision algorithm engineer. He has been engaged in algorithm research, design, improvement and end-to-end deployment optimization in the fields of face detection and recognition, image matching, SLAM, digital human generation, and portrait stylization migration for a long time, and has accumulated rich experience in these fields.

JIA Xia received her BS and MS degrees in control theory and control engineering from Taiyuan University of Technology, China, and Dalian University of Technology, China in 1995 and 2001, respectively. She joined ZTE Corporation in 2001 and worked in the State Key Laboratory of Mobile Network and Mobile Multimedia Technology. Her main research interests include deep learning techniques, face detection and recognition, Re-ID, and activity detection and recognition.