An enhanced light detection and ranging simultaneous localization and mapping based on three-dimensional moving object tracking in dynamic scene

Tao Gao[a], Hu Ran[a], Hongyu Chi[a], Dunwen Wei[a,*]

[a] *The School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, China*

Abstract

Moving objects can cause incorrect feature matches, significantly affecting light detection and ranging (LiDAR) odometry accuracy and mapping quality in simultaneous localization and mapping (SLAM). This paper proposes an SLAM framework incorporating motion estimation of moving objects by a learning-based method to reduce its impact on SLAM. We design a deep dynamic LiDAR odometry (DyLO) network by introducing a hierarchical attention mechanism and a dynamic mask network in point clouds to improve odometry performance in dynamic scenes. Moreover, combined with DyLO, we designed a complete SLAM framework based on our previous work on a dynamic object segmentation (DySeg) network in the point cloud. Our SLAM system is validated on the public dataset with 64-line LiDAR and a self-established campus dataset with 16-line LiDAR. We compare the proposed DyLO with several outstanding iterative closest point (ICP) methods, the deep learning-based method of LiDAR odometry network(LO-Net), and the feature-based odometry approach. Compared to prominent LO-Net, the proposed algorithm reduces the translation error by 4.5% and the rotation error by 3.3% on average. In addition, our ablation experiments demonstrate that incorporating motion estimation of moving objects and backend

☆Research supported in part by the National Natural Science Foundation of China (Grant No. 52275282), Natural Science Foundation of Sichuan Province (Grant No. 2022NSFSC1960), and Fundamental Research Funds for the Central Universities of China (Grant No. ZYGX2021YGCX011)

*Corresponding author.

Email address: weidunwen@uestc.edu.cn (Dunwen Wei)

Preprint submitted to Engineering Applications of Artificial Intelligence November 8, 2024

optimization can significantly enhance the odometry position results.

Keywords: Simultaneous localization and mapping, Deep learning for odometry, Dynamic mask network, Moving object tracking, Dynamic scene

1. Introduction

Light detection and ranging (LiDAR) has emerged as a fundamental sensor choice for outdoor autonomous devices, owing to its capacity to deliver precise measurements of the surrounding environment, remaining unaffected by variations in ambient light conditions. LiDAR-based Simultaneous localization and mapping (SLAM) has become an integral technology for ground mobile vehicles or mobile robots in performing localization and map-building navigation tasks [37]. However, LiDAR-based SLAM algorithms have been numerous in outdoor scenarios and have good accuracy in some specific structured scenarios. However, there are still challenges for dynamic scenarios [6, 23], mainly because the uncertain motion of moving objects in dynamic scenarios affects the odometry pose estimation.

Two common ideas for solving odometry estimation in dynamic scenarios are segmentation and filtering [38, 27], or detection and tracking [35, 26, 8]. Although the filtering method can satisfy the assumption of a static environment in the traditional position calculation, the effect of filtering is very limited in the absence of kinematic information, and it will make the point cloud more sparse after filtering the dynamic points. Detection and tracking methods can make full use of the motion information in the point cloud, and the moving object motion estimation can also establish constraints with the SLAM pose estimation to further optimize the pose estimation results in the odometry or backend. Current detection and tracking methods mainly rely on deep odometry networks for implementation. In previous work, we referenced the work [25] and designed a network for dynamic object segmenting (DySeg) in the LiDAR point cloud.

Deep LiDAR odometry, capable of predicting odometry results directly from two-frame raw point cloud data, has become a prominent research topic in recent years, particularly for dynamic scene SLAM problems. Learning-based odometry has demonstrated significant improvements on datasets compared to traditional manual feature methods or registration methods such as iterative closest point (ICP). In visual odometry, deep odometry networks have made tremendous progress. The point feature pyramid, pose warping,

and cost volume network (PWC-Net) [24] is a compact and efficient convolutional neural network (CNN) model for estimating optical flow variations in 2D images, which is designed according to simple and proven principles: pyramid processing, warping transform, and cost volume. In terms of deep learning models for 3D point clouds, Wang et al. [31] proposed a hierarchical attentional learning method for scene flow in 3D point clouds for correlation coding of two-frame point clouds. The work of PWC LiDAR odometry (PWCLO-Net) [32] demonstrates that introducing correlation coding leads to better odometry performance. Based on the above work PWC-Net and point cloud attention learning mechanism, we design a dynamic mask network to reduce the influence of moving objects in the scene and implement a deep odometry network for estimating the odometry results of two consecutive frames of a point cloud in a dynamic scene.

Some new work combines SLAM systems with moving target tracking to achieve more robust localization and map building in dynamic environments. DL-SLOT [26] uses PointCNN [22] in the front-end part for detecting the possible dynamic objects in the scene and removing among them the LiDAR points that may affect the accuracy of SLAM localization estimation, and then use the LeGO-LOAM [19] approach to computing the frame-to-frame odometry results. DL-SLOT uses cubic polynomials to fit the positions of moving objects in history frames to predict the position trajectories in the backend and constructs a local optimization framework based on the factor graph for position optimization. However, the effect of using a cubic polynomial for the polynomial fitting used in the backend may not be the optimal choice, and a global optimization framework can be built by introducing the motion smoothing factor and loop closure detection factor in the optimization framework of the factor graph.

Therefore, the main contributions of this paper can be summarized as follows:

- We designs a dynamic mask network with spatiotemporal attention by integrating point cloud inter-frame correlation encoding and proposed a deep dynamic LiDAR odometry (DyLO) network. Experiments show that the proposed deep LiDAR odometry network outperforms many excellent ICP methods and the deep learning-based LO-Net.
- We analyze the effectiveness of the SLAM that integrates moving object tracking, along with a comprehensive evaluation of the SLAM system

combining DyLO and factor graph optimization in the backend. Furthermore, we thoroughly examine the impact of polynomial fitting for predicting moving object positions and the sliding window approach on pose estimation. Experiments demonstrate that integrating moving object tracking significantly enhances the accuracy of pose estimation.

2. Related works

2.1. Deep LiDAR odometry

Learning-based LiDAR odometry networks have emerged as a key area of SLAM research, focusing on estimating interframe positional transformations using end-to-end neural network models. For instance, Nicolai et al. [17] projected two consecutive point clouds onto a cylindrical plane to generate depth images, which were then used to estimate the interframe pose.

By employing 2D convolutions and fully connected layers, Nicolai et al. demonstrated the feasibility of deep learning for laser odometry. Velas et al. [29] extended this approach by using three channels—height, range, and intensity—to encode point cloud data, achieving effective pose regression for translational motion. Li et al. [11] proposed LO-Net, which used cylindrical projections to preprocess point clouds and established frame consistency using 3D point normals, while also incorporating uncertainty masks to handle dynamic regions. Wang et al. [36] merged panoramic depth images from two frames, estimating translation and orientation with translation and FlowNet subnetworks, respectively. Li et al. [13] generated high-confidence point correspondences via neural networks and applied singular value decomposition for pose estimation. Cho et al. [3] introduced an unsupervised LiDAR odometry approach, though it lags behind supervised methods in accuracy, offering a novel direction for research. However, these approaches assume a static environment, limiting their effectiveness in highly dynamic scenes.

When addressing the problem of laser odometry in dynamic environments, one common approach is to treat data associated with dynamic objects as outliers and remove them, ensuring that the static environment assumption holds. For instance, Shen et al. [20] preprocess the point cloud using a tandem dynamic segmentation network, effectively filtering out dynamic objects. However, this method increases dataset requirements and computational costs by focusing on both semantic labels and accurate position annotations during training. Huang et al. [7] proposed a PV-RCNN-based [21] solution for dynamic object removal, utilizing an offline-trained 3D DNN

network for detection, followed by precise map refinement using LOAM. Another approach embeds a dynamic mask module into the odometry network [11, 30], allowing dynamic points to be filtered during processing. More recent work explores leveraging both static and dynamic information from point clouds. For example, Huang et al. [8] used simulation to analyze the impact of object distribution, count, and motion speed on odometry accuracy, proposing an adaptive weighting mechanism for dynamic features to enhance performance, rather than simply discarding them.

2.2. Backend optimization methods

Backend optimization in SLAM aims to synthesize data accumulated over extended periods to achieve globally optimal estimations. Leveraging the data provided by odometry, the backend optimization constructs a more complex model to optimize robot trajectory and map construction over longer time scales. Increasingly, researchers are exploring the integration of SLAM with dynamic object tracking, utilizing dynamic target information in the environment to enhance localization accuracy in dynamic scenarios.

Lin et al. [14] introduced the first framework for simultaneously estimating both self-motion and multi-target motion, decomposing the estimation problem into two separate filter-based estimators. Huang et al. [9] proposed a multi-level probabilistic data association strategy for dynamic targets, achieving low-level feature-target association and high-level detection-cluster association, using decoupled factor graph optimization to refine the target's trajectory. Yang et al. [39] introduced CubeSLAM, which employs bundle adjustment to jointly optimize self-localization, target states, and feature points, using a 2D bounding box and filtered point sampling for target detection. Dynamic objects, identified by sparse feature points, are incorporated into the optimization framework rather than being treated as outliers. Zhang et al. [40] proposed VDO-SLAM, which uses optical flow to track feature points on dynamic objects, applying motion constraints within a factor graph for joint optimization of self-positioning and object state, though limited marginalization in the estimation window leads to a large factor graph. Finally, Gonzalez et al. [5] used panoramic segmentation to cluster objects, applied optical flow for data association, and improved the estimation of camera and target positions by incorporating inter-cluster constraints modeled by mechanical joints.

The methods mentioned above primarily address optimizing dynamic targets in visual SLAM systems, where dynamic objects exist in 2D image space.

Inspired by these backend optimization strategies for visual SLAM, Tian et al. [26] proposed a joint optimization method for LiDAR SLAM focused on target tracking. Their approach optimizes both the robot's motion and that of dynamic objects by sampling dynamic target sequences using a sliding-window strategy, significantly enhancing computational efficiency. Lin et al. [15] introduced LIO-SEGMOT, which models the robot's motion estimation and multiple target tracking problem within a nonlinear factor graph and includes a measurement model for asynchronously estimating the states of both the robot and dynamic objects, improving pose estimation accuracy. Li et al. [12] proposed a residual mechanism that segments moving objects based on the original 3D point cloud, followed by backend factor graph optimization after removing points corresponding to moving objects. However, this method does not fully utilize the dynamic information from moving objects. Liu et al. [16] developed a semantic-assisted tightly coupled SLAM approach using LiDAR, which calculates static semantic probabilities to segment static and dynamic points, while simultaneously eliminating static dynamic objects and environmental obstacles. They incorporated semantic constraints in point cloud feature extraction and matching to enhance pose estimation accuracy. Lastly, Li et al. [10] proposed an FA-RANSAC algorithm based on feature information and adaptive thresholding in conjunction with F-LOAM [33] for dynamic target removal. This method extracts static edge and planar feature points for initial distortion compensation and refines the estimated pose using static feature points for secondary distortion compensation.

Existing methods [26, 15, 18] typically rely on a uniform velocity or uniform acceleration model to predict target trajectories, providing a coarse estimation of the target's state. However, in real-world scenarios, target motion is often far more complex than these simple models can capture. As a result, relying on such basic assumptions can lead to significant prediction errors, which propagate through the system and degrade the accuracy of backend optimization.

3. System overview

In this section, we present the proposed SLAM framework for fused motion estimation of moving objects, which comprises three main components: dynamic object segmentation (DySeg) to remove moving objects, deep dynamic LiDAR odometry (DyLO), and a backend optimization module with moving object tracking. Fig. 1 illustrates the system framework. First,

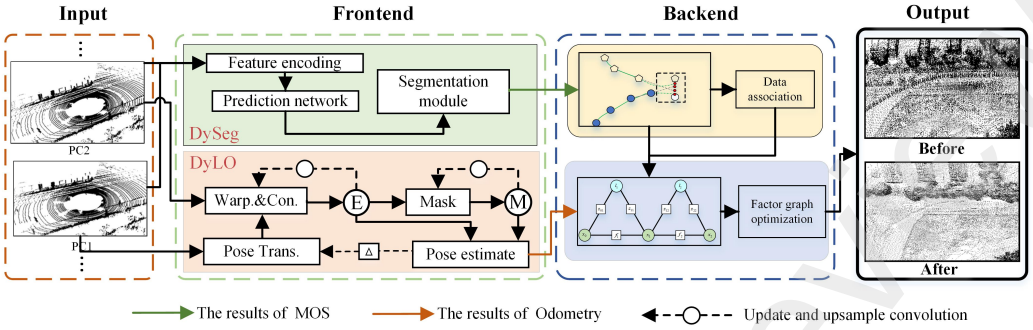

Figure 1: The overview of our proposed LiDAR-based SLAM system fused moving object motion estimate. The system mainly consists of DySeg, DyLO, and backend optimization.

building on our previous work on DySeg (Section3.1), moving targets in the point cloud are identified by predicting scene flow information and applying the segmentation module. Next, we combine the principles of hierarchical attention and PWC-Net, designing a dynamic mask module that integrates the hierarchical attention mechanism to propose the depth odometry, DyLO (Section3.2). For backend optimization, we draw from DL-SLOT and enhance the method for fitting the moving target positions. By introducing the motion smoothing factor and loop closure factor into the factor graph, we further optimize the motion trajectories (Section 3.3). Our SLAM framework aims to achieve more accurate localization and construct higher-quality static maps in dynamic environments.

3.1. Moving object segmentation network: DySeg

Our previous work proposed a scene flow-guided moving object segmentation network, DySeg[1]. The network has three main components: a bird's eye view (BEV) feature encoder, a scene flow prediction update network, and a segmentation module. The BEV feature encoder performs BEV projection coding on the original two consecutive point cloud frames and inputs them to the scene flow update prediction network. The scene flow update prediction network refers to the iterative update mechanism of RAFT[25] to improve it for scene flow prediction. The segmentation module will process the differ-

[1]This work is temporarily unpublished for public release, and we have submitted it to the ICRA2025. Title : a scene flow guided and bird's eye view based moving objects segmentation learning-based method

ence between predicted and static scene flow to segment the dynamic points in the original point cloud. Fig. 2 shows the network structure of DySeg.

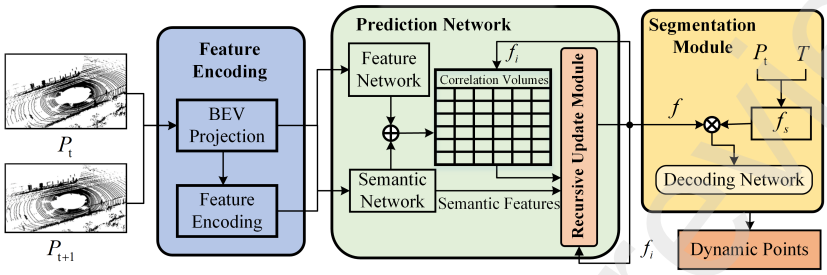

Figure 2: Network architecture of DySeg, a moving target segmentation network based on scene flow guidance and BEV projection.

3.2. Deep LiDAR odometry: DyLO

Most of the current odometry networks do not consider the influence of dynamic targets on pose estimation, and a few construct dynamic masks to reduce the impact of dynamic targets through point cloud features but do not utilize temporal information. We refer to [31], which uses a dynamic mask network with temporal and spatial attention fusion to compensate for dynamic objects and improve pose estimation's effectiveness and robustness. Three practical modules of PWC-Net [24]: pyramid processing, transform operation, and cost computation, as well as the odometry network framework of PWCLO-Net [32] are referenced for the design of the pose estimation network, and the dynamic mask network is designed from scratch to compensate for the negative impact of dynamic objects on the odometry pose estimation.

The framework of the deep LiDAR odometry we proposed is shown in Fig. 3.

3.2.1. Dynamic mask networks

The correlation features $E = \{e_i | e_i \in \mathbb{R}^c\}_{i=1}^n$ of the two frames of the point cloud has been obtained by the motion embedding coding [31] with attention mechanism. In this section, considering the presence of dynamic objects such as cars and pedestrians in road environments, which can negatively affect the estimation of the interframe relative position, a dynamic mask network with spatiotemporal attentional fusion is designed to compensate for the dynamic objects to improve the effectiveness and robustness of the pose estimation.

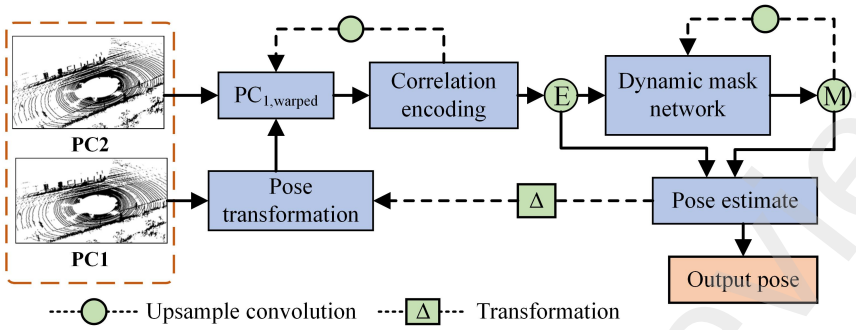

Figure 3: Pipeline diagram of our proposed deep LiDAR odometry. The result of correlation coding come form work [31].

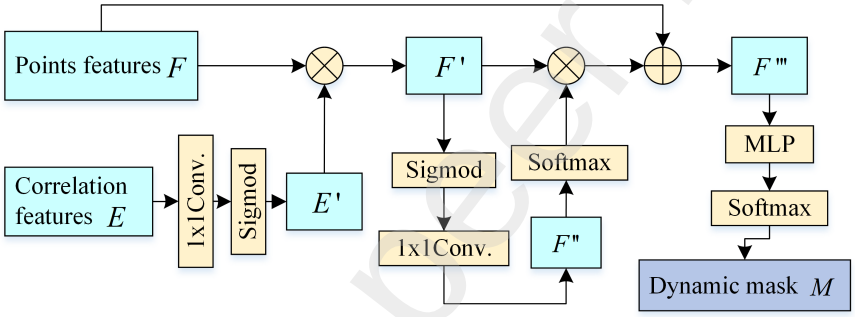

Figure 4: The structure of proposed dynamic mask networks for spatiotemporal attention fusion fused in deep LiDAR odometry.

The dynamic mask network shown in Fig. 4, motion information is further extracted from the correlation feature $E = \{e_i | e_i \in \mathbb{R}^c\}_{i=1}^n$ with the feature F_1 of the point cloud PC_1 through a spatial and channel attention module. This information is comparatively less expensive than the single correlation information utilized to the raw spatial information of the point cloud. The spatial attention mechanism is first used to generate spatially correlated point cloud features F' by emphasizing the spatial correlation position on the point cloud features F through the correlation feature E; then the channel attention mechanism is employed to enhance the correspondence of its essential attributes to generate spatiotemporal fusion features F'''; and finally, through a shared parameterized MLP layer in the dimensions corresponding to each point, a softmax operation to obtain a mask output.

$$M = softmax(MLP(E \oplus F_1)) \tag{1}$$

where $M = \{m_i | m_i \in [0,1]\}_{i=1}^{n}$ is the mask output from the dynamic mask network, the mask ranges from 0 to 1. It characterizes the confidence weight that the point corresponds to being a static point, i.e., the closer the weight is to 1, the more certain that the end is static, and conversely, the smaller the weight is, the more likely that the point will need to be filtered away.

3.3. Backend optimization

3.3.1. Dynamic object motion estimation

The backend joint optimization of fused dynamic targets can be defined as the simultaneous optimization of the mobile robot's position and the state of each target, given a sequence of odometry position results T_{ego} and a sequence of target history positions $\{Tr_0, \ldots, Tr_j\}$. In the whole problem, the primary coordinate system is shown in Fig. 5.

For the 3D spatial coordinate $\mathbf{t}_{k,O_j}^{L_k}$ of the target j under the coordinate system of the mobile robot at the moment k, it is necessary to further correlate this coordinate with the historical trajectory to clarify which historical trajectory it belongs to. Traditional methods use only the target information from neighboring frames, i.e., they adopt the assumption of uniform motion, which may lead to an incorrect association of the target with the historical trajectory. For target trajectory Tracking and historical trajectory prediction, DL-SLOT [26] proposes a polynomial-based fitting for predicting the object plane position and a data association method based on the association matrix. However, this paper further explores the influence of the degree of polynomial fits on the experimental results. As shown in Fig. 5, the uniform motion model can not consider the overall motion, which can lead to erroneous predictions. In contrast, by using polynomial fitting to obtain the overall trajectory of the target, the predicted target position obtained is closer to the actual situation, which helps to improve the success rate of the association.

It is assumed that the target in the environment only performs motion in the horizontal direction, so only the motion trajectories in the x-direction and y-direction are fitted. Considering the complexity and computational efficiency of the target history trajectories, the sliding window method limits the length of the history trajectories. The length of the history trajectory Tr_{t-1} of each target is $n \in [1, K-1]$, where K is the size of the sliding window.

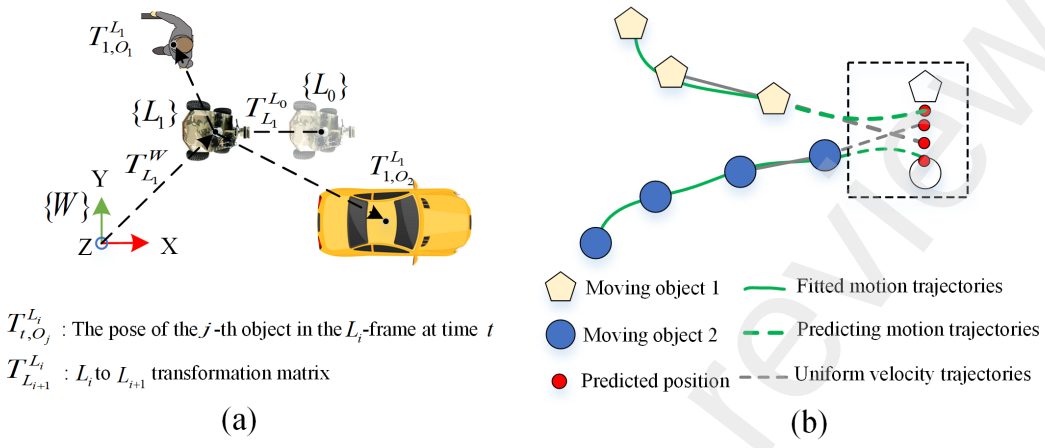

Figure 5: The definition of coordinate systems and position prediction methods. (a) Definition of the coordinate system. (b) Schematic of target trajectory prediction and data association in backend optimization.

Firstly, the coordinate system of each target is unified. The coordinates of the target under the camera coordinate system are converted to the world coordinate system to obtain the coordinates of the target under the world coordinate system $\mathbf{t}_{O_j}^{W} = [x, y, z]^{\mathrm{T}}$.

Polynomial fitting of a target trajectory is achieved using the least squares method. Taking the fitting of a target's trajectory in the x-direction as an example, given the coordinates $x_0, x_1, \ldots, x_t$ from moment 0 to moment t, it is desired to construct a polynomial $f_n(t)$ such that:

$$f_n(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots + a_1 t + a_0 \tag{2}$$

where $a_0, a_1, \ldots, a_n$ are the polynomial coefficients to be solved for, and the coefficients are solved for by minimizing the sum of squares of the errors by least squares to obtain the trajectory of the target in the x-direction $Tr_x(t)$, and the trajectory in the y-direction $Tr_y(t)$.

By constructing a design matrix A and an observation vector $\mathbf{b}$ and solving for the coefficient vector $\mathbf{a}$.

$$\mathbf{A} = \begin{pmatrix} t_0^n & t_0^{n-1} & \ldots & t_0^1 & 1 \\ t_1^n & t_1^{n-1} & \ldots & t_1^1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_t^n & t_t^{n-1} & \ldots & t_t^1 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_t(t) \end{pmatrix} \tag{3}$$

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{4}$$

This yields the trajectory of the target in the x-direction $Tr_x(t)$ and the trajectory in the y-direction $Tr_y(t)$.

Using the association method between historical trajectories and predicted positions in DL-SLOT [26], the association matching matrix S_t is constructed to quantify the degree of matching between historical trajectories and detection targets with dimensions (M, N), where M is the number of historical trajectories, and N is the number of newly added detection targets to be matched. The association matching matrix represents the degree of matching between the predicted positions of the historical trajectories and the detection targets. The closer the expected location is to the detection location, the higher the degree of matching between the two. After obtaining the association matching matrix, the continuous shortest path (Dijkstra) algorithm [34] is used to solve the data association problem, and the successfully matched target j is added to the historical trajectory Tr^j_{t-1}, and then it is updated to the current trajectory $Tr^j_x(t)$ and $Tr^j_y(t)$, that is, the following is obtained The current trajectory Tr^j_{t1}; the initial estimation of the target state is made based on the current trajectory, and the velocity vector of the target is obtained by the derivation of $Tr^j_x(t)$ and $Tr^j_y(t)$ at time t

$$\mathbf{V}^t_j = [\frac{d}{dt}Tr^j_x(t), \frac{d}{dt}Tr_x(t), 0]^{\mathrm{T}} \tag{5}$$

Finally, the target is divided into dynamic and static according to the target velocity magnitude, and the velocity threshold V_{thres} is defined. When $||V^t_j|| > V_{thres}$, the target is regarded as a dynamic target, otherwise, it is a static target.

3.3.2. Factor graphs fusing dynamic objects

To eliminate the error caused by the motion of dynamic objects to the factor graph optimization, the factor graph shown in Fig. 6 considers the motion of each dynamic object separately. The figure represents variable nodes with circles, including four kinds of variable nodes: mobile robot position node x, static target position node s, dynamic target position node l, and target motion velocity node t; and represents factor nodes with squares, including five kinds of factor nodes: odometry factor f, loop closure detection factor h, target observation factor z, smooth motion factor o and target motion factor m.

The factors that make up the factor map are described in Fig. 6.

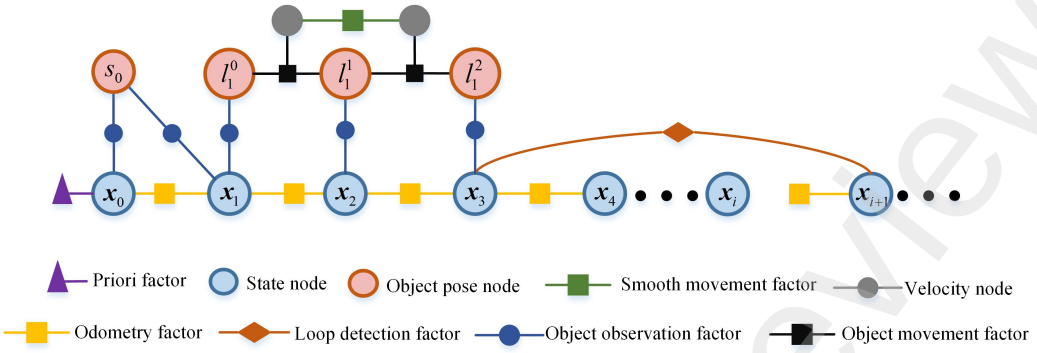

Figure 6: A factor graph optimization framework incorporating dynamic objectives. The system fuses the motion estimation of moving objects through object position nodes, velocity nodes, and smoothing motion factors for factor graphs.

LiDAR odometry factor. The front-end laser odometry outputs the transformation matrix $\mathbf{T}_k^{k-1}$ of point cloud frames from $k-1$ moments to from k moments by performing the pose estimation of the point cloud of the front and back frames, constructing the factor $f(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{T}_k^{k-1})$ to optimize the pose estimation of the mobile robot, and the odometry factor corresponding to the residuals are:

$$\mathbf{r}_f = (\mathbf{T}_{k-1}^{\hat{W}}{}^{-1} \cdot \mathbf{T}_k^{\hat{W}}) \cdot \mathbf{T}_k^{k-1-1} \tag{6}$$

Loop detection factor. Loop detection constraints can be added to the back-end to maintain the consistency of the map across temporal locations when the mobile robot retraces its steps to an area that has been previously explored, especially when performing long-distance locomotion, where loop detection is critical for correcting localization bias due to cumulative errors.

Real-time monitoring of the Euclidean distance between the newly added mobile robot position node $\mathbf{x}_k$ and the existing robot position nodes in the factor graph $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{k-1}\}$ the Euclidean distance between them. A loop closure may have occurred if the distance is less than a preset threshold. At this point, interframe pose estimation is performed on the point cloud data of the two sets of keyframes to determine the transition relationship T_l between the current pose $\mathbf{x}_k$ and the historical pose $\mathbf{x}_{k-d}$ when the loop closure occurs. Then, the loop factor $h(\mathbf{x}_{k-d}, \mathbf{x}_k, \mathbf{T}_l)$ is generated, where $\mathbf{x}_{k-d}$ and $\mathbf{x}_k$ represent the robot's historical and current bit-posture, respectively, at the time when the loop detection occurred. The loop factor is added to the factor graph to optimize the robot's pose estimation.

The loop closure detection factor corresponding to the residuals is:

$$\mathbf{r}_h = (\hat{\mathbf{T}_{k-d}^{W}}^{-1} \cdot \hat{\mathbf{T}_k^{W}}) \cdot {\mathbf{T}_l}^{-1} \tag{7}$$

Object observation factor. The mobile robot may observe the same target O_j at different moments during its motion, and the variable nodes are adjusted by introducing target observation factors to establish the constraints of the same target at different moments. For the pose $T_{O_j}^{L_k}$ of the target O_j at k moments, in order to establish the target observation factor $z(\mathbf{x}_k, T_{O_j}^{L_k})$, the residual of the target observation factor is defined as:

$$\mathbf{r}_z = (\hat{\mathbf{T}_{L_k}^{W}}^{-1} \cdot \hat{\mathbf{T}_{O_j}^{W}}) \cdot {\mathbf{T}_{O_j}^{L_k}}^{-1} \tag{8}$$

Object movement factor. The target motion factor constructs the constraints between the dynamic target position nodes $\mathbf{l}_{k-1}$, $\mathbf{l}_k$ and the target motion velocity node $\mathbf{V}_k$ in the forward and backward frames, so the target motion factor is a ternary factor related to the three nodes, and the target motion factor $m(\mathbf{V}_k, \mathbf{T}_{O,k-1}^{W}, \mathbf{T}_{O,k}^{W})$, followed by defining the residuals of the target motion factor:

$$\mathbf{r}_m = \rho(\hat{\mathbf{T}_{O,k-1}^{W}}^{-1} \cdot \hat{\mathbf{T}_{O,k}^{W}}) - \hat{\mathbf{V}}_k \cdot \Delta t \tag{9}$$

Smooth movement factor. Assuming that the dynamic target moves at a constant velocity over a short period, and therefore, the target's change in position at the same time interval is relatively close, a smoothing motion factor $o(V_k, V_{k+1})$ is constructed, and the corresponding residuals of the smoothing motion factor is:

$$\mathbf{r}_o = ||\hat{\mathbf{V}_{k+1}} - \mathbf{V}_k||_2 \tag{10}$$

By defining the above five factors, the constructed factor graph is defined as the following optimization problem:

$$\begin{aligned} \mathbf{x}^{\star} = \arg\min_{\mathbf{x}} \Bigg\{ & \sum_{k\in[0,t]} \|\mathbf{r}_f\|_{\Sigma}^2 + \sum_{k\in[t-d+1,t]} \|\mathbf{r}_f\|_{\Sigma}^2 + \sum_{k\in[0,t]} \|\mathbf{r}_h\|_{\Sigma}^2 \\ & + \sum_{k\in[0,t]} \sum_{j\in[0,M]} (\|\mathbf{r}_z\|_{\Sigma}^2 + \|\mathbf{r}_o\|_{\Sigma}^2 + \|\mathbf{r}_m\|_{\Sigma}^2) \Bigg\} \end{aligned} \tag{11}$$

4. Experiments

In this section, we evaluate the performance of the proposed system and its modules on the public semanticKITTI dataset [1] and private dataset. The system was tested using three datasets: the 64-line LiDAR SematicKITTI Odometry dataset, KITTI-Tracking, and a private dataset with 16-line LiDAR data sampled at 10 Hz. The accuracy of the proposed deep odometry method, DyLO, is assessed using the KITTI odometry dataset and compared with several classic ICP methods, the advanced deep odometry method LO-Net, and the feature-based odometry method LOAM. Additionally, we evaluate the runtime performance of DyLO. We used the KITTI-Tracking dataset to evaluate the performance of the backend optimization method that integrates dynamic object tracking. On the private dataset, we tested the proposed SLAM system.

We evaluate the accuracy by comparing the absolute trajectory error(ATE) from KITTI metric [4]: the average translation error (ATE_T), and the average rotation error (ATE_R).

All the networks used in this paper were trained using PyTorch and on a single NVIDIA RTX4060Ti. We use the Adam optimizer for training, with smoothing constants set to 0.9 and 0.999, the initial learning rate set to 0.001, and the learning rate decreasing exponentially until it reaches 0.000001 after 10 epochs on the training dataset, with the training batch size set to 2.

4.1. Performance of DyLO

We employed data augmentation to enhance the performance of DyLO. The rotation matrix $\mathbf{R}_{aug}$ is generated by drawing different Euler angles from a Gaussian distribution with 0 as the mean, ensuring the randomness and uniformity of the rotation angles, and $\mathbf{R}_{aug}$ is obtained by computing using these randomly generated Euler angles. The generation process of the translation vector $\mathbf{t}_{aug}$ is similar to that of the rotation matrix, which is also based on the random generation of Gaussian distribution. Finally, $\mathbf{R}_{aug}$ and $\mathbf{t}_{aug}$ are combined to form $\mathbf{T_{aug}}$, which is used to transform the original point cloud $\mathbf{PC_1}$ to generate the enhanced point cloud $\mathbf{PC_{1,aug}}$:

$$\mathbf{PC_{1,aug}} = \mathbf{T}_{aug}\mathbf{PC_1} \tag{12}$$

To maintain consistency in data enhancement, the ground truth of the pose can be obtained as follows:

$$\mathbf{T}_{trans} = \mathbf{T}_{aug}\mathbf{T}_p \tag{13}$$

where $\mathbf{T}_p$ denotes the original true bit-pose transformation matrix from $\mathbf{PC_1}$ to $\mathbf{PC_2}$. Then $\mathbf{T}_{trans}$ is used to generate $\mathbf{q}_{gt}$ and $\mathbf{t}_{gt}$ to supervise the training of the network. The data augmentation expands the training dataset and also ensures the accuracy of the ground truth bit-posture during network training to improve the model's adaptability to pose changes and generalization performance.

4.1.1. Evaluation of odometry accuracy

We evaluated the proposed deep odometry method, DyLO, using sequences 00-10 of the KITTI Odometry dataset. Table 1 presents the results, where the lowest error for each sequence is highlighted in bold, and the second-lowest error is underlined. These results are compared against classical ICP methods, the advanced deep odometry method LO-Net, and the feature-based odometry method LOAM. For a more comprehensive comparison, we also include the performance of LOAM without its mapping module optimization, denoted as LOAM w/o mapping.

Table 1: Odometry results on KITTI-odometry sequrnce00-10.

Seq.	LOAM[41]		ICP-po2po[2]		ICP-po2pl[2]		CLS[28]		Velas[29]		LO-Net[13]		LOAM w/o mapping		DyLO(ours)	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	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	**1.10**	**0.53**	6.88	2.99	3.80	1.73	2.11	0.95	3.02	NA	1.47	0.72	15.99	6.25	1.26	0.64
01	2.79	0.55	11.21	2.58	13.53	2.58	4.22	1.05	4.44	NA	1.36	0.47	3.43	0.93	**0.73**	**0.41**
02	1.54	**0.55**	8.21	3.39	9.00	2.74	2.29	0.86	3.42	NA	**1.52**	0.71	9.40	3.68	1.86	0.78
03	1.13	**0.65**	11.07	5.05	2.72	1.63	1.63	1.09	4.94	NA	1.03	0.66	18.18	9.91	**0.96**	0.75
04	1.45	**0.50**	6.64	4.02	2.96	2.58	1.59	0.71	1.77	NA	0.51	0.65	9.59	4.57	**0.45**	0.56
05	**0.75**	**0.38**	3.97	1.93	2.29	1.08	1.98	0.92	2.35	NA	1.04	0.69	9.16	4.10	1.28	0.74
06	0.72	**0.39**	1.95	1.59	1.77	1.00	0.92	0.46	1.88	NA	**0.71**	0.50	8.91	4.63	1.17	0.63
07	**0.69**	**0.50**	5.17	3.35	1.55	1.42	1.04	0.73	1.77	NA	1.70	0.89	10.87	6.76	1.32	0.67
08	**1.18**	**0.44**	10.04	4.93	4.42	2.14	2.14	1.05	2.89	NA	2.12	0.77	12.72	5.77	1.81	0.70
09	**1.20**	**0.48**	6.93	2.89	3.95	1.71	1.95	0.92	4.94	NA	1.37	0.58	8.10	4.30	1.53	0.70
10	**1.51**	**0.57**	8.91	4.74	6.13	2.60	3.46	1.28	3.27	NA	1.80	0.93	12.67	8.79	1.60	0.73
Mean	1.278	**0.506**	7.763	3.978	4.013	1.968	2.148	0.995	3.218	NA	1.33	0.688	11.09	6.405	**1.27**	0.665

t_{rel} denotes the mean relative translational error (%) on length of 100-800 m; r_{rel} denotes the mean relative rotational error (°) on length of 100-800 m. ICP-po2po and ICP-po2pl refer to iterative nearest-point algorithms for point-to-point and point-to-plane, respectively. Velas, deep learning-based odometry, estimates only the translation vectors. LO-Net[11] is one of the better-performing deep learning odometrys in recent years, achieving a balance of accuracy and speed.

Comparing only the odometry part, the DyLO odometry network proposed achieves better localization accuracy than the above odometry in most of the sequences, and compared with LO-Net, which is the best-performing odometry, this algorithm reduces the translation error by an average of 4.5%

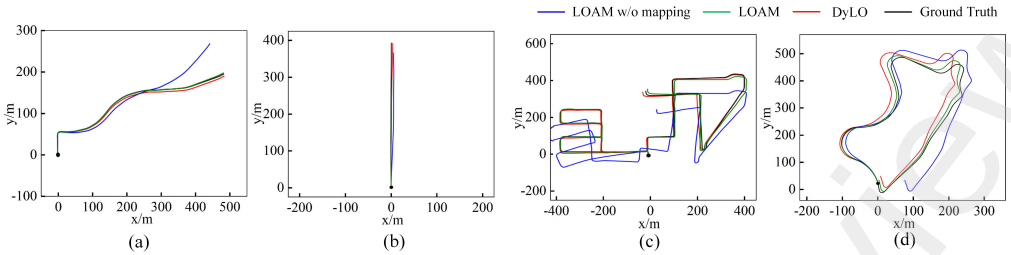

Figure 7: Comparison of trajectories on partial sequences of the KITTI-Odometry dataset. (a) Sequence 03. (b) Sequence 04. (c) Sequence 08. (d) Sequence 09.

and the rotation error by an average of 3.3%. In addition, the proposed DyLO odometry network achieves similar accuracy to the full LOAM in most sequences, with an average reduction in translation error of 0.6% in all sequences.

Fig. 7 presents the results of a qualitative analysis conducted on selected sequences of the KITTI Odometry dataset. This figure displays the algorithmic trajectories alongside the ground truth trajectories for DyLO, full LOAM, and LOAM with odometry only. While the full LOAM algorithm performs well, its localization accuracy significantly degrades when its mapping component is removed, leading to a considerable deviation between the trajectory and the ground truth. The full LOAM algorithm mitigates this deviation through map-to-map matching during map construction. In contrast, the DyLO network proposed in this chapter demonstrates performance comparable to the full LOAM system, relying solely on frame-to-frame matching without backend optimization. The trajectories generated by DyLO are notably closer to the ground truth, underscoring its accuracy advantage over LOAM odometry.

4.1.2. Evaluation of the dynamic mask module

To verify the effectiveness of the dynamic mask module proposed in this paper, ablation experiments are carried out on the dynamic mask module therein, with the rest of the DyLO network unchanged. Table 2 shows the results of the relative position errors for the highly dynamic 01 and 04 sequences in the KITTI-Odometry dataset. The odometry using the dynamic mask module outperforms the odometry without the dynamic mask module, and the odometry using the dynamic mask module in the LO-Net for the majority of the metrics for both the 01 and 04 sequences. The root mean

square error (RMSE) of the relative position error is reduced by 37.7% and 6.0% in sequence 01 compared with the previous two methods and by 28.0% and 18.9% in sequence 01 compared with the previous two methods. The experimental results show that the point-cloud dynamic masking module we proposed for spatiotemporal attention fusion has less relative pose error in high dynamic sequences compared to the dynamic coding module, proving the effectiveness of the dynamic mask module.

Table 2: Relative position error(m) corresponding to the mileage calculation method for different dynamic mask modules.

Methods	01						04					
	Max	Mean	Median	Min	RMSE	Std	Max	Mean	Median	Min	RMSE	Std
W/O mask	4.31	3.46	3.29	0.80	3.50	0.96	3.52	2.92	2.82	1.10	2.86	0.65
mask in LO-Net	3.08	2.25	2.20	0.45	2.32	**0.44**	2.68	2.54	2.50	**0.84**	2.54	0.35
ours	**2.83**	**2.11**	**2.09**	**0.32**	**2.18**	0.60	**2.37**	**2.05**	**2.02**	1.03	**2.06**	**0.22**

4.1.3. Runtime of DyLO

To evaluate the odometry running speed, experiments were carried out to compare the running time of the DyLO odometry method we proposed with that of the LOAM. The results of the experiments, as shown in Table 3, are as follows: for the LOAM, it is necessary to add the map building part to achieve high accuracy pose estimation, while the proposed DyLO network relies only on odometry to achieve performance and efficiency similar to those of LOAM. The odometry and map building of LOAM are separate threads, so deleting the map building part will not significantly reduce its time consumption.

Table 3: Odometry operating speed(ms).

Methods	Preprocess	Pose estimation	Overall
LOAM w/o mapping	24	70	94
LOAM	/	140	140
DyLO	**15**	**71**	**86**

4.2. Performance of backend optimization

4.2.1. Sliding window

The overall target trajectory is fitted using polynomials to make the obtained predicted target position closer to the true position. The root mean

square error (RSME) between the predicted and true positions of the target is obtained by designing experiments for trajectory prediction to verify the method's effectiveness. First, the effect of different sliding window sizes on the accuracy of the predicted target position is explored by taking the trajectory prediction of target 0 of the KITTI-Tracking dataset 10 sequence as an example.

Table 4: The impact of different polynomial degrees and sliding window sizes on localization results(m).

Methods	5	10	15	20	30	40	50	Mean
Linear poly.	0.24	0.23	0.26	0.28	0.43	0.62	0.84	0.414
Quadratic poly.	0.41	0.30	0.26	0.21	0.23	0.28	0.32	**0.287**
Cubic poly.	1.30	0.48	0.39	0.33	0.26	0.27	0.28	0.473
Quartic poly.	1.34	0.77	0.60	0.49	0.38	0.31	0.33	0.603
Quintic poly.	1.37	1.50	1.00	0.74	0.56	0.43	0.40	0.857
Mean	0.932	0.656	0.502	0.410	**0.372**	0.382	0.434	

The RMSE of the predicted and true target positions of the polynomial fitting functions of different orders corresponding to sliding window sizes from 5 to 50 frames are shown in Table 4. From the table, it can be concluded that different sliding window sizes and the polynomial order affect the prediction performance. Higher orders of polynomial fit predictions correspond to more minor errors as the sliding window size increases. In the table, the smallest value of the root mean square error is marked by bolding, and the next smallest value of the root mean square error is marked by underlining. Considering the root mean square error corresponding to the sliding window size and the polynomial order, the fitting function with a sliding window of 30 and a quadratic polynomial are the optimal choices.

4.2.2. Effect of predicted position

To verify the effectiveness of the target trajectory prediction method based on the polynomial fitting proposed, the target position is predicted by different trajectory prediction methods on the KITTI-Tracking dataset for targets of partial sequences, and the compared methods include the traditional uniform model method and the trajectory prediction method with polynomial fitting of different orders proposed. The homogeneous model method only utilizes the previous moment's motion to predict the next moment's position according to the homogeneous motion model. Compared to the uniform model, the polynomial-fitted trajectory prediction method, in

which the linear polynomial corresponding to polynomial order one also predicts the position at the next moment according to the uniform motion model but uses the least-squares method to fit multiple targets within a sliding window, which takes advantage of the more extended history information.

The RMSE between the predicted and true values of the corresponding target positions for the different trajectory prediction methods are shown in Table 5. The polynomial-fitting-based target trajectory prediction method corresponds to a sliding window size of 30. Compared with the homogeneous modeling method, the trajectory prediction methods with different orders of polynomial fitting achieve smaller root mean square errors, among which the trajectory prediction method with quadratic polynomials achieves the smallest error values in most sequences, which proves that the polynomial fitting-based target trajectory prediction method has higher prediction accuracy on the KITTI-Tracking dataset.

Table 5: RMSE of target position prediction corresponding to different trajectory prediction methods(m).

Methods	04/2	08/8	08/13	10/0	18/2	18/3	Mean
Uniform velocity	1.33	1.49	1.37	1.42	0.57	0.65	1.14
Linear poly.	1.21	0.65	**0.54**	0.28	0.41	0.45	0.59
Quadratic poly.	0.71	**0.64**	0.60	**0.21**	0.14	**0.22**	**0.42**
Cubic poly.	**0.64**	0.81	0.85	0.33	**0.13**	0.23	0.50
Quartic poly.	0.81	1.07	1.17	0.49	0.18	0.29	0.67

4.2.3. Effect of moving object tracking

We selected long sequences from the KITTI-Tracking dataset for ablation experiments on backend optimization, which combines dynamic objective Tracking to verify the effectiveness of moving object Tracking for backend optimization.

The experiments focus on verifying the effectiveness of introducing dynamic objectives in the backend optimization for improving trajectory accuracy. The initial trajectory is the position output of the DyLO odometer proposed in this paper, which is called DyLO. DyLO-SAM is the output trajectory obtained by not introducing the trajectory of the dynamic target on the result of the DyLO output position and using the backend optimization method of LIO-SAM. The LIO-SEGMOT* used in this experiment results from LIO-SEGMOT [15] based on removing the IMU pre-integration fac-

tor and using the DyLO output pose as the odometer pose input. DyLO-SAM-MOT results from backend co-optimization fusing the dynamic target Tracking with the DyLO output pose.

The RMSE of the absolute trajectory errors for the translational and rotational components of some sequences in the KITTI-Tracking dataset are presented in Table 6. The lowest error values for each sequence across the different methods are highlighted in bold. The results demonstrate that the absolute trajectory error of the robot decreases after backend optimization in most sequences. For translational error, the optimized output pose with the introduction of dynamic objectives shows an average reduction of 7.3% compared to the initial trajectory error, 4.5% compared to the optimization method without dynamic objectives, and 0.9% compared to LIO-SEGMOT*. Additionally, the rotational error also decreases. These findings indicate that incorporating constraints on the motion of dynamic objects within the factor graph enhances the performance of the backend optimization algorithm.

Table 6: Outputting absolute trajectory errors for position on the KITTI-Tracking dataset.

Sequences	DyLO		DyLO-SAM		LIO-SEGMOT*		DyLO-SAM-MOT	
	ATE_T[m]	ATE_R[rad]	ATE_T[m]	ATE_R[rad]	ATE_T[m]	ATE_R[rad]	ATE_T[m]	ATE_R[rad]
04	2.947	0.167	2.915	**0.161**	**2.754**	0.167	2.841	0.167
07	4.821	**0.108**	4.671	0.118	4.356	0.120	**4.142**	0.114
08	2.216	0.207	2.227	0.213	2.205	**0.201**	**2.144**	0.206
10	1.163	0.153	1.121	0.150	1.115	0.152	**1.105**	**0.146**
15	0.427	0.099	0.420	0.103	0.401	0.112	**0.366**	**0.098**
18	3.894	0.186	3.620	0.185	**3.425**	0.189	3.526	0.185
19	3.452	0.161	**3.391**	0.162	3.464	0.160	3.430	**0.158**
Mean	2.703	0.154	2.624	0.156	2.531	0.157	**2.507**	**0.153**

Fig. 8 illustrates the trajectories before and after optimization on sequence 07 of the KITTI-Tracking dataset. The initial trajectory, marked in red, shows a notable deviation from the ground truth, highlighting the limitations of using odometry data alone for localization. In contrast, the optimized trajectories, marked in green and blue, closely align with the ground truth, indicating that backend optimization enhances positioning accuracy. Furthermore, the trajectories obtained with the introduction of dynamic objectives are even closer to the ground truth than those using backend optimization without dynamic objectives. This result demonstrates the effectiveness of incorporating dynamic objectives in optimizing the mobile robot's trajectories.

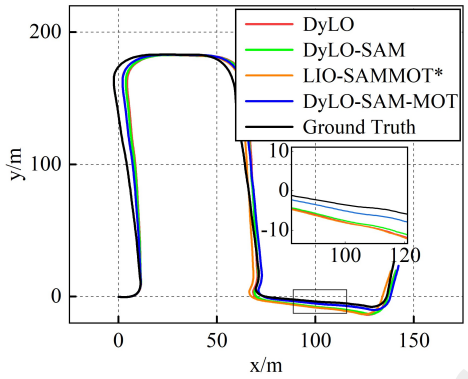

Figure 8: Trajectories obtained by different methods on KITTI-Tracking sequences 07.

4.2.4. Effect of smooth motion factor

To validate the effectiveness of the smoothed motion factor integrated into the backend factor graph, we analyzed the target's motion velocity using the KITTI-Tracking dataset. The results are shown in Fig. 9. Due to inaccuracies in the manual labeling of the target's position, the true velocity exhibits large fluctuations. By applying smoothed constraints on the motion factors, we obtain a more consistent predicted velocity in the factor graph, which more closely reflects the target's actual motion state in real-world scenes. The figure shows that the predicted velocities are situated within the fluctuation range of the true velocities, demonstrating the effectiveness of the smoothing motion factor constraint.

4.3. SLAM system evaluation

We evaluated the proposed SLAM system on our established campus dataset, and the proposed algorithm is denoted as DyLO-SAM-MOT. Fig. 10 shows the ground-moving vehicle and the dataset scenes used in the private dataset. The dataset comprises three sequences named UESTC-GNSS, recorded in campus environments and featuring dynamic objects such as pedestrians, vehicles, and bicycles. To evaluate the performance of the proposed SLAM system, the ground truth for these odometry sequences was obtained using GNSS/RTK.

The results of ATE of LOAM, DyLO odometry, DyLO-SAM-MOT, and the ground truth are shown in Table 7. The table shows that the trajectory outputs after backend co-optimization have lower errors than LOAM on all sequences, especially for the 01 sequence, which has a higher degree

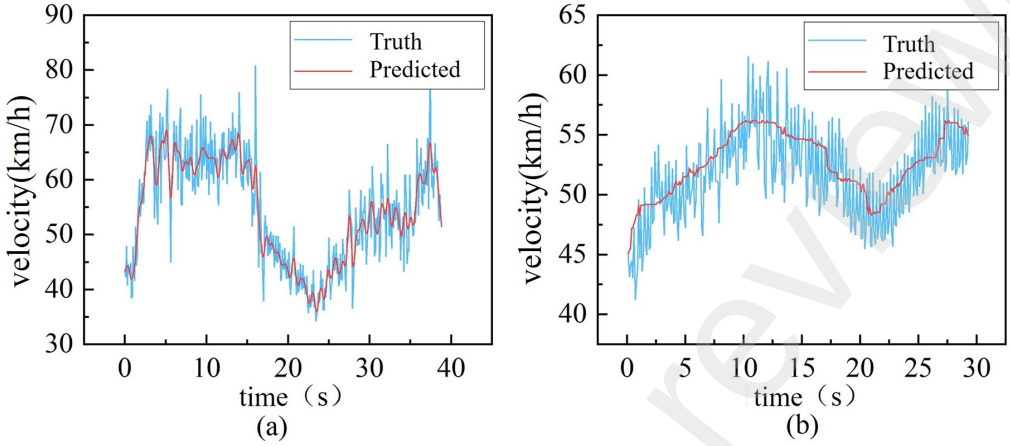

Figure 9: The velocity of truth and predicted for the KITTI-Tracking dataset targets. (a) Target 8 for sequence 08. (b) Target 0 for sequence 10.

of dynamics. The RMSE of the algorithm proposed has been reduced by 70.8% compared to the LOAM algorithm. There is also a certain degree of reduction in the 00 and 02 sequences, which proves that the laser of the iterative optimization of the positional pose under dynamic environment proposed odometry and the backend joint optimization algorithm incorporating dynamic target tracking in complex dynamic scenarios.

Table 7: Comparison of absolute trajectory errors(m) among algorithms on the campus dynamic scene dataset.

Sequence	Methos	Max	Mean	Median	Min	Rmse	Std
00	LOAM	2.475	0.944	0.819	0.115	1.113	0.589
	DyLO	3.895	1.400	1.270	0.398	1.568	0.708
	DyLO-SAM-MOT	**1.731**	**0.758**	**0.714**	**0.028**	**0.895**	**0.474**
01	LOAM	9.641	2.576	2.377	0.071	3.121	1.762
	DyLO	5.519	1.361	1.083	0.115	1.713	1.041
	DyLO-SAM-MOT	**2.395**	**0.774**	**0.659**	**0.055**	**0.910**	**0.480**
02	LOAM	0.360	0.136	0.138	0.022	0.146	**0.054**
	DyLO	1.386	0.303	0.218	**0.020**	0.396	0.255
	DyLO-SAM-MOT	**0.303**	**0.132**	**0.114**	0.025	**0.144**	0.059

Experiments are carried out to compare the algorithm proposed in our paper with the LOAM algorithm in sequences 00, 01, and 02. The output trajectories and the true value trajectories of the three algorithms are shown in Fig. 11, and the closer the trajectories of the different algorithms to the true value trajectory, the higher the accuracy is indicated. The figure

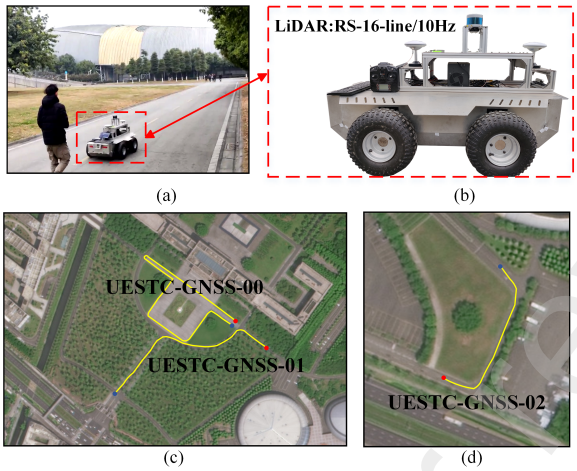

Figure 10: Trajectories obtained by different methods on KITTI-Tracking sequences 07. (a) The real-world environment where the dataset was collected. (b) The unmanned ground vehicle was used for data collection. (c) Satellite imagery of sequences 00 and 01. (d) Satellite imagery of sequence 02.

shows that the backend joint optimization trajectory is closer to the true value trajectory than the LOAM trajectory, and the positioning accuracy is higher.

Fig. 11 shows the trajectory results, and static point cloud maps were constructed and visualized on the campus dynamic scene dataset 02 sequence. Our method eliminates the dynamic point cloud to obtain the static point cloud. When building the map, the corresponding static point cloud is used for the keyframe point cloud for map stitching. Fig. 11(e) shows the result of without dynamic point cloud rejection, the dynamic noise brought to the point cloud map due to the movement of pedestrians is circled in the red dotted line box, which is manifested in the form of a black point cloud residuals on the map. Fig. 11(f) is the static point cloud map obtained after the dynamic point cloud rejection and the dynamic noise in the map is reduced. The resulting static point cloud map ensures the consistency and stability of the data. It can more accurately represent the environmental structure, which is more valuable than the point cloud map filled with dynamic noise.

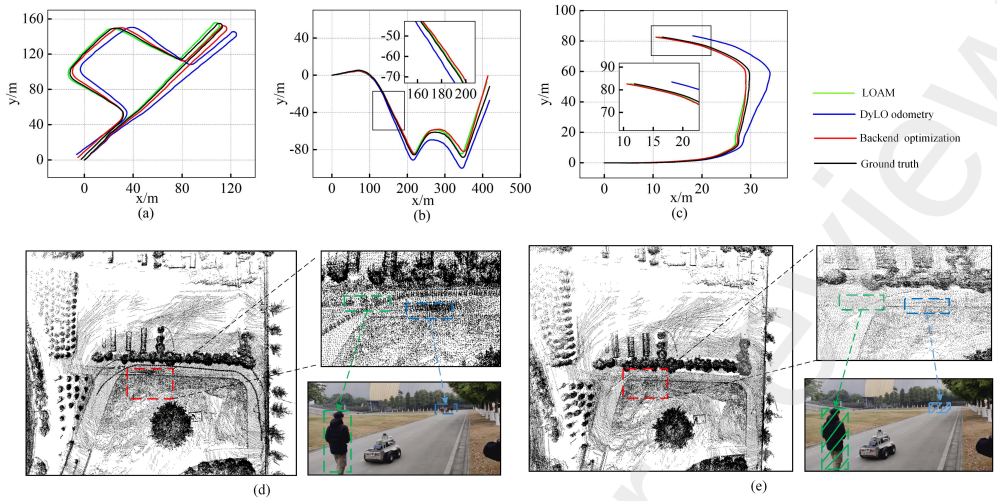

Figure 11: Trajectory of sequence and point cloud map constructed by our method. (a) 2D trajectory plots of sequences 00. (b) 2D trajectory plots of sequences 01. (c) 2D trajectory plots of sequences 02. (d) The map before removing dynamic point clouds. (e) The map after removing dynamic point clouds.

5. Conclusion

In this study, we propose an SLAM method for fused moving object Tracking in dynamic scenes. By segmenting the moving objects in the environment and fusing the results of our deep LiDAR odometry in the backend for further motion estimation of the moving objects. Our method reduces the impact of dynamic objects on localization and map building, improves the accuracy of the odometry, and also accomplishes the segmentation and tracking of the moving objects. The point cloud map built by our method can filter out the moving objects in the environment and improve the global consistency of the point cloud map with the environment. Our experimental results in this study show that the odometry network DyLO outperforms some of the baseline methods. Compared to excellent baseline methods, LO-Net, the proposed algorithm reduces the translation error by 4.5% and the rotation error by 3.3% on average. The moving object tracking results in the backend are significant for improving odometry accuracy. In future work, we will consider fusing moving object information in the deep odometry to improve the odometry estimation results further.

References

[1] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J., 2019. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences, in: Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV).

[2] Besl, P., McKay, N.D., 1992. A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 239–256.

[3] Cho, Y., Kim, G., Kim, A., 2020. Unsupervised geometry-aware deep lidar odometry, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Paris, France. pp. 2145–2152.

[4] Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Providence, RI, USA. pp. 3354–3361.

[5] Gonzalez, M., Marchand, E., Kacete, A., Royan, J., 2022. TwistSLAM: Constrained slam in dynamic environment. IEEE Robotics and Automation Letters 7, 6846–6853.

[6] Hu, H., Qiao, Z., Cheng, M., Liu, Z., Wang, H., 2021. Dasgil: Domain adaptation for semantic and geometric-aware image-based localization. IEEE Transactions on Image Processing 30, 1342–1353.

[7] Huang, F., Shen, D., Wen, W., Jiachen, Z., Hsu, L.T., 2021. A coarse-to-fine lidar-based slam with dynamic object removal in dense urban areas, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3162–3172.

[8] Huang, F., Wen, W., Zhang, J., Wang, C., Hsu, L.T., 2024. Dynamic object-aware lidar odometry aided by joint weightings estimation in urban areas. IEEE Transactions on Intelligent Vehicles 9, 3345–3359.

[9] Huang, J., Yang, S., Zhao, Z., Lai, Y.K., Hu, S.M., 2019. ClusterSLAM: A slam backend for simultaneous rigid body clustering and motion estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South). pp. 5875–5884.

[10] Li, J., Zhang, X., Zhang, Y., Chang, Y., Zhao, K., 2023a. Rf-loam: Robust and fast lidar odometry and mapping in urban dynamic environment. IEEE Sensors Journal 23, 29186–29199.

[11] Li, Q., Chen, S., Wang, C., Li, X., Wen, C., Cheng, M., Li, J., 2019. LO-Net: Deep real-time lidar odometry, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA. pp. 8473–8482.

[12] Li, Q., Zhuang, Y., Huai, J., 2023b. Multi-sensor fusion for robust localization with moving object segmentation in complex dynamic 3d scenes. International Journal of Applied Earth Observation and Geoinformation 124, 103507.

[13] Li, Z., Wang, N., 2020. DMLO: Deep matching lidar odometry, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Las Vegas (NV) USA. pp. 6010–6017.

[14] Lin, K.H., Wang, C.C., 2010. Stereo-based simultaneous localization, mapping and moving object tracking, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Taipei, Taiwan. pp. 3975–3980.

[15] Lin, Y.K., Lin, W.C., Wang, C.C., 2023. Asynchronous state estimation of simultaneous ego-motion estimation and multiple object tracking for lidar-inertial odometry, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, London, United Kingdom. pp. 10616–10622.

[16] Liu, P., Bi, Y., Shi, J., Zhang, T., Wang, C., 2024. Semantic-assisted lidar tightly coupled slam for dynamic environments. IEEE Access 12, 34042–34053.

[17] Nicolai, A., Skeele, R., Eriksen, C., Hollinger, G.A., 2016. Deep learning for laser based odometry estimation, in: RSS workshop Limits and Potentials of Deep Learning in Robotics, Michigan, USA. p. 1.

[18] Pang, Z., Li, Z., Wang, N., 2022. Simpletrack: Understanding and rethinking 3d multi-object tracking, in: European Conference on Computer Vision, Springer, Tel Aviv, Israel. pp. 680–696.

[19] Shan, T., Englot, B., 2018. Lego-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Madrid, Spain. pp. 4758–4765.

[20] Shen, S., Cai, Y., Wang, W., Scherer, S., 2023. DytanVO: Joint refinement of visual odometry and motion segmentation in dynamic environments, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, London, United Kingdom. pp. 4048–4055.

[21] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H., 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10526–10535.

[22] Shi, S., Wang, X., Li, H., 2019. Pointrcnn: 3d object proposal generation and detection from point cloud, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–779.

[23] Shu, C., Luo, Y., 2023. Multi-modal feature constraint based tightly coupled monocular visual-lidar odometry and mapping. IEEE Transactions on Intelligent Vehicles 8, 3384–3393.

[24] Sun, D., Yang, X., Liu, M.Y., Kautz, J., 2018. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, USA. pp. 8934–8943.

[25] Teed, Z., Deng, J., 2020. Raft: Recurrent all-pairs field transforms for optical flow, in: European Conference on Computer Vision (ECCV), Springer, Glasgow, UK. pp. 402–419.

[26] Tian, X., Zhu, Z., Zhao, J., Tian, G., Ye, C., 2024. Dl-slot: Tightly-coupled dynamic lidar slam and 3d object tracking based on collaborative graph optimization. IEEE Transactions on Intelligent Vehicles 9, 1017–1027.

[27] Vaquero, V., Fischer, K., Moreno-Noguer, F., Sanfeliu, A., Milz, S., 2019. Improving map re-localization with deep 'movable' objects segmentation on 3d lidar point clouds, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 942–949.

[28] Velas, M., Spanel, M., Herout, A., 2016. Collar line segments for fast odometry estimation from velodyne point clouds, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Stockholm, Sweden. pp. 4486–4495.

[29] Velas, M., Spanel, M., Hradis, M., Herout, A., 2018. Cnn for imu assisted odometry estimation using velodyne lidar, in: 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, Torres Vedras, Portugal. pp. 71–77.

[30] Wang, G., Wu, X., Jiang, S., Liu, Z., Wang, H., 2022. Efficient 3d deep lidar odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 5749–5765.

[31] Wang, G., Wu, X., Liu, Z., Wang, H., 2021a. Hierarchical attention learning of scene flow in 3d point clouds. IEEE Transactions on Image Processing 30, 5168–5181.

[32] Wang, G., Wu, X., Liu, Z., Wang, H., 2021b. Pwclo-net: Deep lidar odometry in 3d point clouds using hierarchical embedding mask optimization, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15905–15914.

[33] Wang, H., Wang, C., Chen, C.L., Xie, L., 2021c. F-loam:fast lidar odometry and mapping, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4390–4396.

[34] Wang, H., Yu, Y., Yuan, Q., 2011. Application of dijkstra algorithm in robot path-planning, in: 2011 second international conference on mechanic automation and control engineering, IEEE, Inner Mongolia, China. pp. 1067–1069.

[35] Wang, R., Xu, Y., Sotelo, M.A., Ma, Y., Sarkodie-Gyan, T., Li, Z., Li, W., 2019a. A robust registration method for autonomous driving pose estimation in urban dynamic environment using lidar. Electronics 8.

[36] Wang, W., Saputra, M.R.U., Zhao, P., Gusmao, P., Yang, B., Chen, C., Markham, A., Trigoni, N., 2019b. Deeppco: End-to-end point cloud odometry through deep parallel neural network, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Macau, China. pp. 3248–3254.

[37] Wang, Y., Zhao, C., Liang, J., Wen, M., Yue, Y., Wang, D., 2023. Integrated localization and planning for cruise control of ugv platoons in infrastructure-free environments. IEEE Transactions on Intelligent Transportation Systems 24, 10804–10817.

[38] Xiang, L., Ren, Z., Ni, M., Jenkins, O.C., 2015. Robust graph slam in dynamic environments with moving landmarks, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2543–2549.

[39] Yang, S., Scherer, S., 2019. CubeSLAM: Monocular 3-d object slam. IEEE Transactions on Robotics 35, 925–938.

[40] Zhang, J., Henein, M., Mahony, R., Ila, V., 2020. Vdo-slam: a visual dynamic object-aware slam system. arXiv preprint arXiv:2005.11052 .

[41] Zhang, J., Singh, S., 2014. LOAM: Lidar odometry and mapping in real-time, in: Robotics: Science and Systems, Berkeley, CA, Rome, Italy. pp. 1–9.