IOP Publishing Measurement Science and Technology
Meas. Sci. Technol. **36** (2025) 026318 (14pp) https://doi.org/10.1088/1361-6501/ada8c8

Online lane mapping based on multi-sensor SLAM and Catmull–Rom splines

Tianmu Gao, Danping Zou, Chao Ping Chen*, Xiaojun Wu and Haiyang Hu

State Key Laboratory of Avionics Integration and Aviation System-of-Systems Synthesis, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, People's Republic of China

E-mail: ccp@sjtu.edu.cn

Received 29 September 2024, revised 28 December 2024
Accepted for publication 10 January 2025
Published 29 January 2025

Abstract

Traditional offline high-definition maps have significant limitations owing to their high cost and difficulty in updating. Online vector maps represent a new trend in the development of autonomous driving. In this paper, we propose an online lane mapping system, which leverages both multi-sensor simultaneous localization and mapping (SLAM) and Catmull–Rom splines. The map is constructed in real-time through the localization provided by SLAM and the fitting of splines. Existing SLAM algorithms perform poorly in complex urban environments, a deep learning network is incorporated into the visual front-end of our system to improve stability. Deep learning can extract and study deeper and less observable features, so that the system enhances the accuracy of visual tracking in scenarios characterized by low texture, lighting variations, and rapid motion. The system state is optimized by a factor graph framework to achieve precise localization. To construct continuous and consistent online lane maps, our system combines SLAM localization results with spline curves to achieve lane connection. Additionally, we develop a convenient method for initializing, extending, and optimizing the control points of splines to obtain more precise results. Our method combines the geometric characteristics with historical observations of lanes, enhancing the association and consistency of maps. Experiments conducted on the mainstream datasets demonstrate that our system exhibits greater robustness and improves the accuracy of localization and mapping across diverse environments.

Keywords: SLAM, multi-sensor fusion, deep learning, lane detection, Catmull–Rom spline, vector map

1. Introduction

Localization and mapping are two important modules in autonomous driving that provide support and assistance for the vehicle's navigation, planning, and decision-making. The purpose of simultaneous localization and mapping (SLAM) is to solve these two problems. SLAM can achieve high-precision localization and high-definition (HD) mapping.

Localization is typically defined as a state estimation problem. SLAM processes sensor observations and vehicle's motion through Kalman filter or nonlinear optimization [1]. In recent years, sensor fusion has become a trend in SLAM development. Since a single sensor is insufficient for all scenarios, multiple sensors are able to complement each other and provide robust and accurate estimates [2]. Cameras provide rich visual information and are popular in autonomous driving because of their low cost. Inertial measurement unit (IMU) can

* Author to whom any correspondence should be addressed.

© 2025 IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

provide precise orientation information without external interference. Global navigation satellite system (GNSS) can obtain global position and mitigate cumulative drift of cameras and IMU [3]. HD maps are composed of localization and vehicle perception results that contain rich environmental semantic information. However, the generation of HD maps usually requires high-accuracy LiDAR, which is costly to maintain and has a long update cycle. This is a limitation in autonomous driving. Compared with offline maps, online real-time HD maps have advantages in terms of cost and updates. Online maps are constructed through the image perception of road elements such as lanes, markings, and traffic signals.

In this work, a real-time lane mapping system is proposed that can achieve high-precision localization and generate vectorized lanes. The system receives input from three sensors: cameras, IMU and GNSS. A learning-based feature extraction network is utilized as the visual front-end to improve robustness and accuracy in edge cases. In the back-end, all states are optimized with factor graph, thus achieving globally consistent pose estimation. For lane mapping, conventional function curves are inadequate for representing lanes. Based on the characteristics of lanes, Catmull–Rom splines are appropriate for modeling. This spline is lightweight and has a C1 continuity. In addition, all control points lie on the curve, facilitating subsequent optimization [4]. Combined with high-precision positioning results, a method is designed for initializing, extending, and optimizing control points to generate lane maps in real time. The main contributions of this study shall be condensed into:

- A multi-sensor SLAM system that replaces the visual front-end with a learning feature extraction network is proposed. Compared to traditional methods like GVINS, it is able to handle various complex environments and offer better positioning performance.
- A lane mapping method based on multi-sensor SLAM and Catmull–Rom splines is designed, which includes the processes of lane detection, fitting, extending and optimization. This method effectively combines the geometric characteristics with historical observations of lanes.
- Improvement in localization accuracy and lane mapping quality. In complex environments, our method outperforms some of the latest methods while not incurring significant latency.

2. Related work

2.1. Traditional multi-sensor SLAM

Early SLAM research primarily applied a single sensor, including methods such as PTAM [5], SVO [6], and ORB-SLAM [7]. However, relying solely on a single sensor has significant limitations. Complementing multiple sensors can help to address various extreme scenarios. A common fusion approach is to integrate the camera with the IMU, which is commonly referred to as visual-inertial odometry (VIO). Sun *et al* proposed MSCKF-VIO [8], which is a tightly-coupled VIO system based on a multi-state constraint Kalman filter (MSCKF). This system uses a sliding window approach to ensure that the poses of all image frames within the window are constrained by the observed feature points. Qin *et al* proposed VINS-Mono [2], which includes automatic initialization, online extrinsic calibration, relocalization, loop closure detection, and global pose graph optimization.

Multi-sensor SLAM usually contains more than three sensors, including camera, LiDAR, IMU and GNSS. VIO or LiDAR -inertial odometry (LIO) always operates in a local coordinate frame. This leads to four unobservable directions: three global translations and rotations along the gravity direction (yaw angle) [9]. Over time, VIO and LIO accumulate uncorrectable drift in these four directions. However, GNSS can provide global drift-free positioning. VINS-Fusion [10] is a loosely-coupled framework that builds on VINS-Mono by integrating GPS and VIO. Cao *et al* proposed GVINS [3], a tightly-coupled system that integrates raw GNSS measurements with the VIO framework. It can achieve accurate global estimates even in complex environments and regions, where GNSS signals are prone to interference or invalid. InGVIO [11] is an invariant filter-based method that integrates GNSS measurements with VIO. Li *et al* fuses VIO with precise point positioning under the factor graph framework [12]. Additionally, the fusion of LiDAR with IMU, and GNSS is also a common approach in multi-sensor SLAM [13]. However, existing multi-sensor SLAM systems commonly use traditional methods to extract image or point cloud features, which can lead to decreased accuracy or even failure in complex environments. Our system employs deep learning methods to resist environmental interference because it can extract deeper and less observable features.

2.2. Learning-based SLAM

The application of deep learning in SLAM primarily includes front-end feature detection and tracking, end-to-end pose generation, and 3D scene reconstruction. DX-SLAM [14] uses convolutional neural networks (CNN) to extract feature points and descriptors for pose estimation and loop closure detection, achieving higher computational efficiency and robustness. Superpoint [15] is designed to extract feature points and descriptors. It is a self-supervised network framework, but hard to ensure real-time performance. Li *et al* extract point and line features and use a deep neural network for loop closure detection [16]. Kang *et al* present a method of SLAM by feeding off the deep-SORT-based object tagging and lightweight network for loop closure [17]. Another approach is to generate pose estimation directly through an end-to-end network. DeepVO [18] can compute the pose directly from a sequence of raw images without using any modules from traditional SLAM frameworks. However, it is difficult to apply in practical applications. There are also methods that use neural networks to fuse data from multiple sensors, such as [19] and [20]. However, existing approaches either require labeled data for supervised training or have high model complexity, leading to insufficient generalization and real-time performance. In

contrast, the network used in our system is based on unsupervised learning, and it is enable to adapt to a variety of scenarios and remains lightweight enough to ensure good generalization and real-time performance. In terms of scene reconstruction, some researchers have combined NeRF [21] and 3D Gaussian splatting [22] with SLAM, achieving good results. However, this method requires substantial computational resources and cannot be used in real time.

2.3. Online mapping

Traditional HD maps rely on LiDAR to gather point clouds, which are both expensive and difficult to maintain. Online maps are generated in real-time by sensing the surrounding environment, making them more cost-effective and convenient. Qin *et al* [23, 24] uses semantic segmentation to extract visual features and constructs local maps online. However, these maps are constructed from point clouds and do not contain instance information. Zhou *et al* [25] integrated pose optimization with semantic segmentation and constructed an instance-level map through inverse perspective mapping (IPM) transformation. But this method uses straight-line approximations for lanes, which can result in significant errors in curved sections. A better approach is to apply spline curves to represent lanes. Qiao *et al* [26] provided a good approach, but their control point expansion design is relatively complex. In addition, end-to-end methods [27, 28] for map generation have emerged in recent years. However, they can only be implemented on devices with high computational power. Currently, online maps rely on SLAM to integrate single-frame perception results, but they lack efficient methods for modeling and optimizing map elements or use complex post-processing algorithm. Our system uses a simple and efficient method to represent lane maps based on Catmull–Rom splines.

3. Methodology

3.1. System overview

The structure of our proposed system is illustrated in figure 1. The system inputs consist of three parts: camera, IMU and GNSS. First, inputs from these three sensors are preprocessed. The satellite receiver obtains raw measurements, including the pseudorange and Doppler shifts. Low-elevation and unstable satellite signals are filtered out based on the number of satellite tracks and elevation angles. For IMU, pre-integration is performed to obtain the pose transformation between consecutive frames [2]. A feature extraction network functions as the visual front-end to extract and track feature points, which will be utilized for structure from motion in initializing and will become observation constraints in factor graph optimization. After preprocessing, next step is the initialization of the three sensors. The motion pose is calculated based on tracked feature points and then incorporate IMU to obtain the scale, velocity, and gravity direction. The yaw angle and global anchor points are computed by aligning the satellite trajectory. Finally, the global positions are obtained through nonlinear optimization of all system state within a sliding window.

For lane mapping, a lane detection network is employed to extract lane points from camera image. Subsequently, all lane points are projected onto the same coordinate system based on global positioning from SLAM and then merged. Following this, the unordered lane points are fitted and sampled. Then the existing control points are extended and optimized to finally obtain an ordered lane map.

We define some frames in this study. $(\cdot)^e$ is an earth-centered earth-fixed (ECEF) frame. Its center is at Earth's center of mass, the x-axis points towards the intersection of the Greenwich meridian with the equator, the y-axis points in a direction perpendicular to the x-axis and within the equatorial plane, and the z-axis points towards the north pole. $(\cdot)^n$ is the east–north–up (ENU) frame, with the x-axis, y-axis, and z-axis pointing towards the east, north, and upward directions, respectively. $(\cdot)^w$ is the world frame, whose z-axis is aligned by gravity. $(\cdot)^b$ and $(\cdot)^c$ are the IMU frame and the camera frame respectively.

3.2. Pre-processing

The preprocessing stage primarily involves processing of sensor data to provide the necessary conditions for initialization. The ALIKE [29] network is used to extract feature points and descriptors from images. ALIKE is a lightweight network that achieves precise detection results using a differentiable keypoint detection module. The network can replace the manual feature extraction and tracking methods. In situations with low texture, extreme illumination changes, and significant viewpoint changes, this network can maintain good performance. This effectively addresses the problem of traditional methods failing in corner cases. In addition, compared to other deep learning methods, the ALIKE network is lightweight. Thus, it can extract feature points at high speed. The network input is an image, and the output is the pixel coordinates of keypoints along with their corresponding descriptors. Each image has 150 keypoints extracted by the ALIKE network. Moreover, a minimum radius is set for the keypoints detection to avoid clustering.

Traditional methods use optical flow method for feature tracking. Because optical flow are premised on the assumptions of constant illumination and small motion, it is easy to fail under varying lighting conditions and rapid motion. So an adaptive strategy are implemented for selecting the method in feature tracking. Under normal circumstances, optical flow is employed for tracking because it offers faster processing speed and better real-time performance. However, when it fails in complex environments, specifically when the number of successfully tracked points falls below 20% of the total number, we switch to deep learning-based descriptor matching for feature tracking. Learning-based descriptors offer higher accuracy and robustness but are more computationally intensive. For descriptor matching, a mutual nearest neighbor (MNN) matching algorithm is utilized. The MNN algorithm computes a similarity matrix between descriptors using cosine similarity and identifies descriptor matches that are mutual nearest neighbors based on this matrix. This ensures that only the most reliable matches are selected, where each descriptor's nearest neighbor

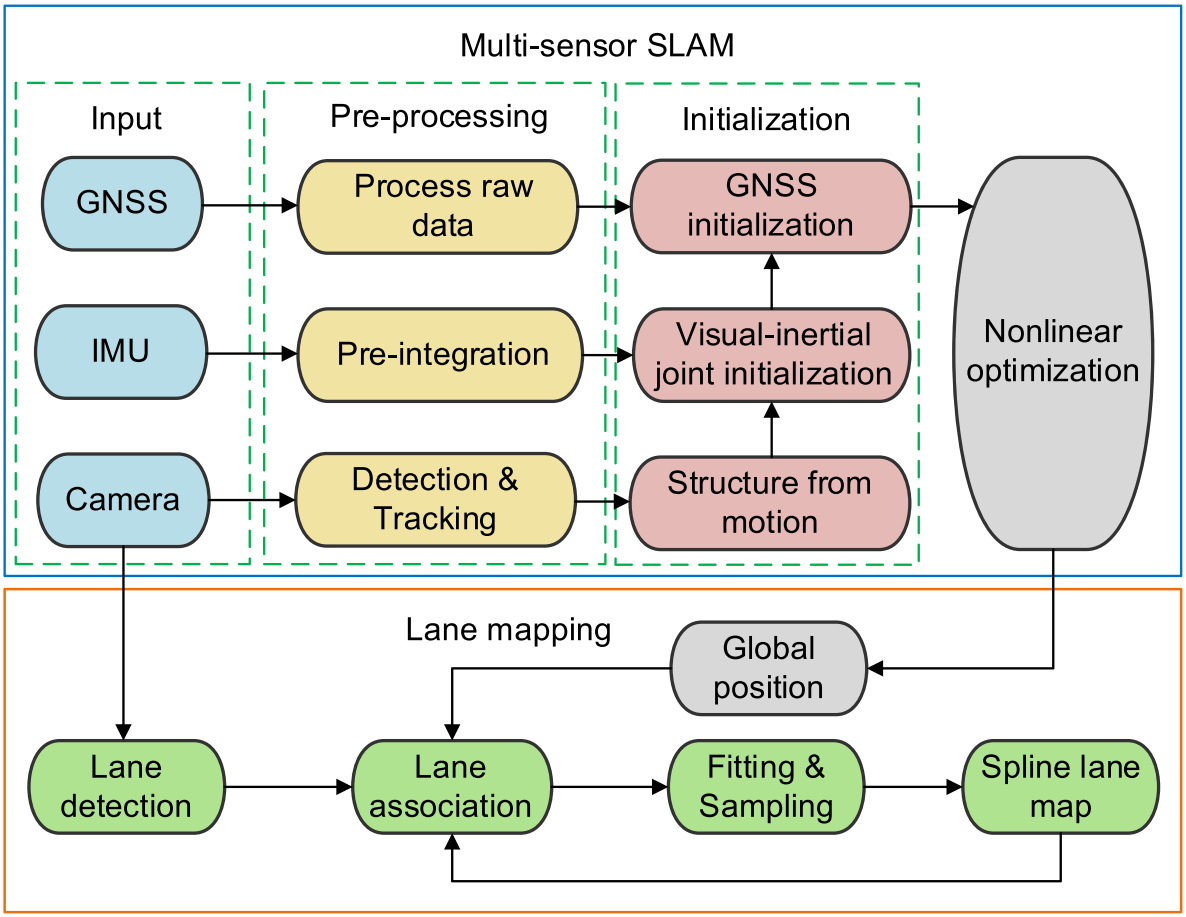

Figure 1. Diagram of the whole pipeline of our proposed system. Our system is divided into two parts: multi-sensor SLAM and lane mapping. At first, we obtain global positioning through preprocessing, initialization, and nonlinear optimization. Then, based on the global positioning results, the lane points are merged, fit, and sampled. Finally, by extending and optimizing the control points, global lane map is generated.

in one set is also its counterpart's nearest neighbor in the other set. After tracking, random sample consensus (RANSAC) and fundamental matrix [30] are applied to remove outliers and perform distortion correction on the remaining points.

Using the IMU motion model, the formula for IMU pre-integration [2] is derived as follows:

$$\begin{cases} \boldsymbol{\alpha}_{b_{k+1}}^{b_k} = \iint_{t\in[k,k+1]} \mathbf{R}_{b_t}^{b_k}\left(\mathbf{a}_t - \mathbf{b}_{a_t} - \mathbf{n}_a\right) \mathrm{d}t^2 \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \mathbf{R}_{b_t}^{b_k}\left(\mathbf{a}_t - \mathbf{b}_{a_t} - \mathbf{n}_a\right) \mathrm{d}t \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \frac{1}{2}\boldsymbol{\Omega}\left(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_{\omega_t} - \mathbf{n}_\omega\right)\boldsymbol{\gamma}_{b_t}^{b_k} \mathrm{d}t \\ \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\hat{\boldsymbol{\omega}} & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \end{cases} \tag{1}$$

Here, b_k stands for the body frame at time k, and $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}, \boldsymbol{\beta}_{b_{k+1}}^{b_k}, \boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ the position, velocity, and rotation between b_k and b_{k+1}. $\tilde{\boldsymbol{a}}_t$ and $\tilde{\boldsymbol{\omega}}_t$ are the linear acceleration and angular velocity of the IMU measurement at time t. $\mathbf{R}_{b_t}^{b_k}$ stands for the rotation from the moving frame b_t at time t to the body frame b_k. $\mathbf{b}_{a_t}$ and $\mathbf{b}_{\omega_t}$ are the accelerometer bias and gyroscope bias at time t. The additive noises $\mathbf{n}_a$ and $\mathbf{n}_\omega$ are assumed to be zero-mean Gaussian distributed. $\hat{\boldsymbol{\omega}}$ is the skew-symmetric matrix. Using equation (1), the motion change between two frames can be obtained. This formula requires only IMU measurements, so it does not need to be repeatedly updated during back-end optimization, thus saving a significant amount of computation.

For GNSS, unsuitable measurements are filtered out through conditional screening. First, we check the number of currently observed satellites and select the L1 frequency. Subsequently, the current message is determined by the epoch time of the ephemeris parameters. Finally, unstable and low-elevation satellites are removed based on their tracking status and elevation angles.

3.3. Initialization

All sensors in the system are tightly coupled, whereas an appropriate initial value is necessary for the optimization. To meet this requirement, an initialization procedure is designed. The initialization process is loosely coupled and does not require an initial value. In this process, each sensor independently computes the system state, and the results are then fused together. The initialization is divided into three steps: structure

from motion, visual-inertial joint initialization, and global initialization. To limit the computational load, all initialization operations are performed within a sliding window of length N (typically 10).

3.3.1. Structure from motion. This stage relies on the feature points extracted and matched by the visual front-end. First, the camera pose and feature points are estimated based on image relationships. All frames within the sliding window are compared to the most recent frame. If they meet the criteria of having more than 30 matched points and a disparity of more than 20 pixels, they are set as the reference frame, which serves as the initial frame of the world coordinate. Next, the relative rotation and translation between these two frames are recovered using epipolar geometry, and then all co-visible features points are triangulated. Based on these triangulated points, the perspective-n-point (PnP) method is used to estimate the poses of all other frames in the window and triangulate feature points continually. Finally, global bundle adjustment is applied to optimize all poses and points within the window.

3.3.2. Visual-inertial joint initialization. The purpose of joint initialization of vision and IMU is to align the camera and IMU sensor data and obtain initial values for gyroscope bias, velocity, scale, and gravity vector. First step is calibrating the gyroscope bias by minimizing the following cost function:

$$\min_{\delta \mathbf{b}_\omega} \sum_{k \in N} \left\| \mathbf{q}_{b_{k+1}}^{c_0}{}^{-1} \otimes \mathbf{q}_{b_k}^{c_0} \otimes \boldsymbol{\gamma}_{b_{k+1}}^{b_k} \right\|^2 \tag{2}$$

$\mathbf{q}_{b_k}^{c_0}$ and $\mathbf{q}_{b_{k+1}}^{c_0}$ stands for the rotation from two consecutive frames b_k and b_{k+1} to the initial camera frame, which are obtained from camera motion. $\boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ comes from IMU pre-integration. $\otimes$ represents multiplication operation of quaternion.

After initializing the gyroscope bias, the following equation can be constructed through IMU pre-integration increments.

$$\begin{bmatrix} -\mathbf{I}\Delta t_k & 0 & \frac{1}{2}\mathbf{R}_{c_0}^{b_k}\Delta t_k^2 & \mathbf{R}_{c_0}^{b_k}\left(\mathbf{p}_{c_{k+1}}^{c_0} - \mathbf{p}_{c_k}^{c_0}\right) \\ -\mathbf{I} & \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0} & \mathbf{R}_{c_0}^{b_k}\Delta t_k & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{b_k}^{b_k} \\ \mathbf{v}_{b_{k+1}}^{b_{k+1}} \\ \mathbf{g}^c \\ s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{b_{k+1}}^{b_k} + \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0}\mathbf{p}_c^b - \mathbf{p}_c^b \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} \end{bmatrix} \tag{3}$$

Here $\mathbf{R}$ and $\mathbf{p}$ stand for the rotation and translation between two coordinates, where b_k stands for the body frame and c_k the camera frame at time k. $\mathbf{I}$ is the Identity matrix. Δt_k refers to the time difference between the body frame b_k and b_{k+1}. $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$ and $\boldsymbol{\beta}_{b_{k+1}}^{b_k}$ represent the changes in position and velocity between frame b_k and b_{k+1}. $\mathbf{v}_{b_k}^{b_k}$ is velocity in the body frame b_k. $\mathbf{g}^c$ is the projection of the gravity vector from the world frame into the camera frame. And s is a scale factor that represents the ratio between the camera-estimated depth and the true world depth. By solving this equation, the initial values of the state variables are obtained.

3.3.3. GNSS initialization. The role of GNSS in the system is to provide global observation. Therefore, during the initialization stage, it is necessary to calibrate the anchor point of the local world origin in the ECEF coordinate and the yaw angle. Since the gravity vector has already been aligned, the local coordinate differs from the ENU coordinate by only a yaw offset. Based on GNSS data, single point positioning is first used to obtain a rough location. Next, the yaw angle and anchor point are calibrated by minimizing the Doppler residuals and pseudorange residuals, respectively. A detailed definition of these two residual terms will be presented in section 3.4.3. At this point, the initialization phase is complete, and all necessary system variables have been assigned initial values.

3.4. Optimization

Initialization is merely a loosely coupled process of sensors, aimed at obtaining the initial values of the system state. During the optimization process, all sensors will be tightly coupled through nonlinear algorithms to achieve precise state estimation. Tight coupling refers to the process of directly combining the preprocessed data from each sensor for joint optimization. First, SLAM can be defined as maximum a posteriori (MAP) problem. Given these observations, the goal is to estimate the states that maximize posterior probability [2]. According to Bayes' theorem, the MAP estimation can be transformed into finding the maximum of the product of the likelihood and the prior, thereby converting it into a maximum likelihood estimation (MLE) problem. Assuming all measurements are independent and the noise follows a zero-mean Gaussian distribution, it is common to use the minimization of the negative log-likelihood to find the maximum likelihood estimate. In this way, finding the maximum likelihood is equivalent to minimizing the sum of squared errors. If there are multiple observations, the overall MLE can be transformed into minimizing the sum of squared errors. Ultimately, the MAP estimation can be converted into a least squares problem that can be solved using nonlinear optimization methods.

According to these assumptions, the optimization problem can be formulated as follows:

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p\mathbf{X}\|^2 + \sum_{i=1}^{n} \mathbf{r}\left(\mathbf{z}_i, \mathbf{X}\right)_{\mathbf{P}_i}^2 \right\} \tag{4}$$

where $\mathbf{X}$ stands for the system state, including position, velocity, orientation, acceleration bias, gyroscope bias and inverse depth of feature points. $\mathbf{H}_p$ and $\mathbf{r}_p$ contain the prior information about the system states. $\mathbf{r}(\cdot)$ represents the residual function for each measurement. $\mathbf{z}$ is different sensor measurements and $(\cdot)_{\mathbf{P}_i}$ the Mahalanobis norm. In the following section, the residual factors for each sensor will be introduced. The residual terms are treated as factors to construct the factor graph. As illustrated in figure 2, the optimization process involves obtaining the optimal estimate of the system state under the

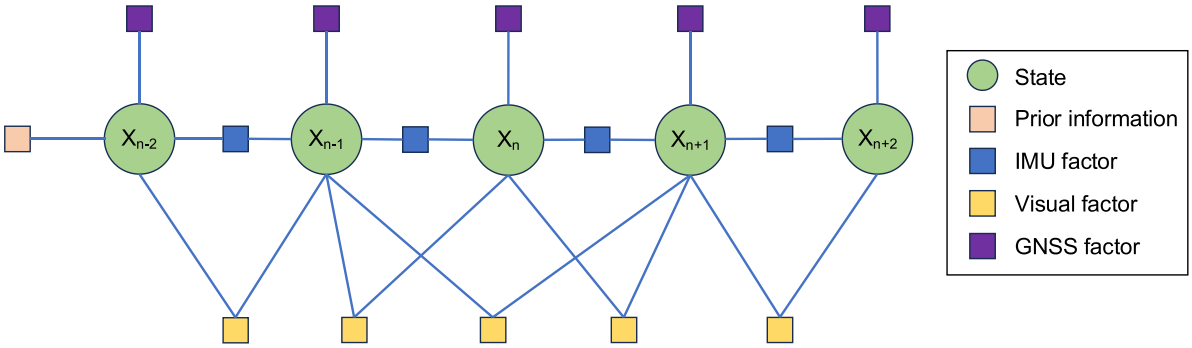

Figure 2. Factor graph of our system. Circles denote the system state at different times, whereas squares represent constraints and prior information. Our system simultaneously optimizes all states within the sliding window. The state is updated by minimizing the residuals of the factors.

factor constraints. Solving nonlinear optimization problems primarily involves constructing increment equations of the following form:

$$\left(\mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})+\lambda\mathbf{I}\right)\Delta\mathbf{x}=-\mathbf{J}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) \tag{5}$$

Here $\mathbf{J}(\mathbf{x})$ represents the Jacobian matrix of residuals, $\mathbf{r}(\mathbf{x})$ the residuals, and λ damping factor that dynamically adjusts the convergence rate based on the magnitude of the current residuals. $\mathbf{I}$ is the identity matrix. By iteratively solving for the increment, and updating the Jacobian and damping factor, the residuals finally comes to a minimum value. The Ceres solver [31] is used for solving nonlinear optimization problem, which is a nonlinear optimization library.

3.4.1. Visual residuals.

$$\begin{aligned}&\mathbf{r}_C(\tilde{\mathbf{z}}_l,\mathbf{X})=\tilde{P}_l^{C_j}-P_l^{C_j}\\&P_l^{C_j}=\mathbf{R}_b^c\left(\mathbf{R}_w^{b_j}\left(\mathbf{R}_{b_j}^w\left(\mathbf{R}_c^b\frac{1}{\rho_c}\pi_c^{-1}\tilde{P}_l^{C_i}+\mathbf{p}_c^b\right)+\mathbf{p}_{b_j}^w\right)+\mathbf{p}_w^{b_j}\right)+\mathbf{p}_b^c\end{aligned} \tag{6}$$

The visual residual is modeled using a reprojection error. In equation (6), $\tilde{\mathbf{z}}_l$ is the measurement of camera which contains ρ_c the depth of landmark, $\tilde{P}_l^{C_j}$ and $\tilde{P}_l^{C_i}$ the normalized coordinate value of the lth landmark point in the camera coordinate observed by the camera frame at time j and i. $\mathbf{X}$ is system state, which in this equation is $\mathbf{R}_w^{b_j}$ $\mathbf{R}_{b_j}^w$ $\mathbf{p}_{b_j}^w$ and $\mathbf{p}_w^{b_j}$ the rotation and position between the body frame and world frame. π_c refers to the camera projection function, which is composed of the camera's intrinsic parameters.

$$\mathbf{r}_B\left(\tilde{\mathbf{z}}_{b_{k+1}}^{b_k},\mathbf{X}\right)=\begin{bmatrix}\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}\\\delta\boldsymbol{\beta}_{b_{k+1}}^{b_k}\\\delta\boldsymbol{\gamma}_{b_{k+1}}^{b_k}\\\delta\mathbf{b}_a\\\delta\mathbf{b}_\omega\end{bmatrix}=\begin{bmatrix}\mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_{k+1}}^w-\mathbf{p}_{b_k}^w+\frac{1}{2}\mathbf{g}^w\Delta t_k^2-\mathbf{v}_{b_k}^w\Delta t_k\right)-\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}\\\mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_{k+1}}^w+\mathbf{g}^w\Delta t_k-\mathbf{v}_{b_k}^w\right)-\tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\\2\left[\mathbf{q}_{b_k}^{w-1}\otimes\mathbf{q}_{b_{k+1}}^w\otimes\left(\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}\right)^{-1}\right]_{xyz}\\\mathbf{b}_{a_k}-\mathbf{b}_{a_{k+1}}\\\mathbf{b}_{\omega_k}-\mathbf{b}_{\omega_{k+1}}\end{bmatrix} \tag{7}$$

3.4.2. IMU residuals. The IMU residual refers to the error between the measured and predicted values, as expressed in equation (7). $\tilde{\mathbf{z}}_{b_{k+1}}^{b_k}$ is the IMU measurements between the body frame b_k and b_{k+1}, which includes the pre-integrated terms $\tilde{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k},\tilde{\boldsymbol{\beta}}_{b_{k+1}}^{b_k},\tilde{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$. $\mathbf{X}$ is system state which in this equation includes $\mathbf{p}$, $\mathbf{v}$, $\mathbf{R}$ the position, velocity and rotation between the body frame and world frame, $\mathbf{b}_{a_k}$ and $\mathbf{b}_{\omega_k}$ the accelerometer bias and gyroscope bias at time k. $\mathbf{g}^w$ is the gravity vector in the world frame. The quaternion $\mathbf{q}$ is equivalent to rotation. $[\cdot]_{xyz}$ represents the vector part of the quaternion with $\otimes$ representing multiplication operation of quaternion. Δt_k is time interval between b_k and b_{k+1}.

3.4.3. GNSS residuals. According to the pseudo range and Doppler measurement, the residuals can be computed:

$$\mathbf{r}_P(\tilde{\mathbf{z}}_r^s,\mathbf{X})=\|\mathbf{p}_s^e-\mathbf{p}_r^e\|+c\delta\mathbf{t}+D_r^s-\tilde{P}_r^s \tag{8}$$

$$\mathbf{r}_D\left(\tilde{\mathbf{z}}_r^s, \mathbf{X}\right) = \frac{1}{\lambda}\left(\mathbf{v}_s^e - \mathbf{v}_r^e\right) + \frac{c}{\lambda}\dot{\delta}t + \Delta\tilde{f}_r^s \tag{9}$$

Equations (8) and (9) represent the residuals between the predicted and measured pseudoranges and Doppler shifts, respectively. Here, $\tilde{\mathbf{z}}_r^s$ is the GNSS measurement which in this equation is $\tilde{P}_r^s$ and $\Delta\tilde{f}_r^s$ the pseudorange and Doppler frequency difference between satellite and receiver. is system state including $\mathbf{p}_r^e$ and $\mathbf{v}_r^e$. $\mathbf{p}_s^e/\mathbf{p}_r^e$ is the position of satellite/receiver in ECEF frame, and $\mathbf{v}_s^e/\mathbf{v}_r^e$ the velocity of; satellite/receiver in ECEF frame. $\delta\mathbf{t}$ and $\dot{\delta}t$ are the receiver's clock biases and drifting rates. D_r^s is the delay term including tropospheric, ionospheric and multipath delay between satellite and receiver. c is the signal speed, and λ the signal wavelength. The position and velocity of receiver can be expressed as:

$$\mathbf{p}_r^e = \mathbf{R}_n^e\mathbf{R}_w^n\left(\mathbf{p}_b^w + \mathbf{R}_b^w\mathbf{p}_r^b\right) + \mathbf{p}_w^e \tag{10}$$

$$\mathbf{v}_r^e = \mathbf{R}_n^e\mathbf{R}_w^n\mathbf{v}_b^w \tag{11}$$

Here, $\mathbf{R}_n^e$ is the rotation matrix between the ECEF frame and ENU frame, and $\mathbf{R}_w^n$ the rotation matrix between the ENU frame and world frame computed form yaw angle. $\mathbf{p}_b^w$and $\mathbf{R}_b^w$ stand for system state. $\mathbf{p}_r^b$ is the offset of receiver in the body frame and $\mathbf{p}_w^e$ the anchor point of the local world origin in the ECEF coordinate. Through the aforementioned formulas, the constraints between the system state and GNSS residuals are established. It is applied in the initialization and optimization processes.

3.4.4. Marginalization. Similar to initialization, optimization is also performed within a sliding window. To limit the computational load, a marginalization strategy is employed to control the number of optimization parameters within the sliding window. During marginalization, unnecessary frames need to be removed. The disparity is used to determine whether the new frame should be considered a keyframe. If it is determined to be a keyframe, it will be retained in the window, and the oldest frame will be removed from the window. Otherwise, the new frame will be discarded directly. Marginalization is performed using the Schur complement [32]. To retain constraints from old frames, prior factors are obtained based on the marginalized measurements for the next optimization. By computing the Schur complement of the system's optimization equations, the incremental equation for the marginalized variables can be obtained. Its form is $\mathbf{H}\Delta x = \mathbf{r}$, where $\mathbf{H}$ and $\mathbf{r}$ serve as prior information for the next optimization step.

3.5. Lane mapping

The localization results are used to merge multiple local lane points into a global map. Lane mapping is a process that involves fitting all detected lane points using Catmull–Rom splines in global coordinate system.

3.5.1. Lane representation. There is an introduction to the modeling methods. Lanes in the map are represented by Catmull–Rom splines. This spline is composed of a series of control points, called $\mathbf{p}_i \in \mathrm{R}_3$. Every four control points determine a segment of the curve $\mathbf{p}(u)$ between $\mathbf{p}_{i-2}$ and $\mathbf{p}_{i+1}$. In equation (12), $u \in [0,1]$ is the parameter. τ controls the curvature of the splines and is usually set to 0.5.

$$\mathbf{p}(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 2-\tau & \tau-2 & \tau \end{bmatrix} \times \begin{bmatrix} \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix} \tag{12}$$

The benefits of using a Catmull–Rom splines include first-order derivative continuity and the fact that all control points lie on the splines. And changing one control point only affects the local segment of the splines, which facilitates optimization. Since the shape of the curve depends on four control points, extension and optimization of splines can be performed by focusing on these points.

3.5.2. Lane detection. The ultra-fast-lane-detection (UFLD) [33] network is used to detect and extract lanes. It combines real-time performance with accuracy, and can directly output instance-level lanes. This network generates two-dimensional (2D) lane points. Constructing lane maps requires three-dimensional (3D) points. Because all lanes are on the ground, they can be converted to 3D lane points using simple geometric methods. As shown in figure 3, we assume that the x-z plane of the camera coordinate system is parallel to the ground. The y-coordinates of points on the ground in the camera coordinate system corresponds to the height of the camera above the ground. Thus, the following conclusion can be easily reached:

$$\lambda = \frac{1}{h} * \left(\pi^{-1}(u)\right)_y \tag{13}$$

where λ is the inverse depth, h is the camera height, $\pi^{-1}(\cdot)$ is the inverse camera projection function, and u is the pixel coordinate. According to equation (13), the positions of the 2D lane points in the 3D camera coordinate system can be obtained.

3.5.3. Lane association. All lane points can be transformed into the same coordinate system through rotation and translation between the camera frame and world frame. In the process of real-time mapping, it is necessary to associate newly detected lane points with the previously constructed lanes to prepare for subsequent lane mapping. Specifically, it means determining which lane in 3D space the detected lane points belong to. The Kuhn–Munkres (KM) algorithm [34, 35] is applied in this problem. The KM algorithm is used to solve the optimal matching problem in a weighted bipartite graph. It

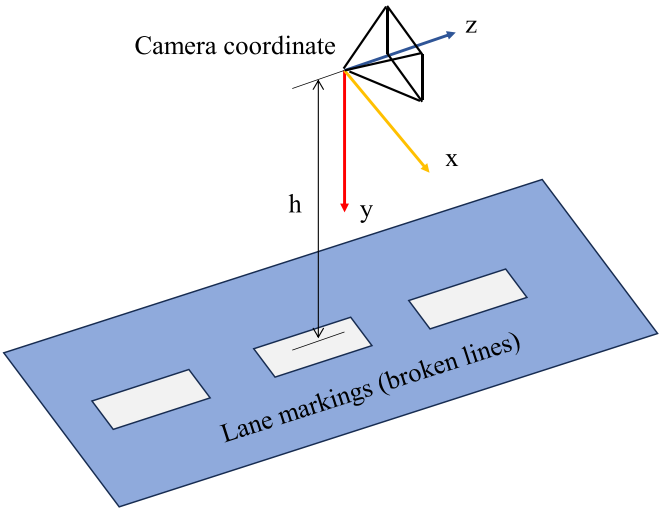

Figure 3. Illustration of lane markings in the camera coordinate. The x–z plane of the camera coordinate system is parallel to the ground. The lane markings are on the plane where $y = h$.

sets the objects as vertices in the graph, and if there is a potential match between two objects, an edge is created between the corresponding vertices. The weight of the edge represents the distance between the two objects. Initially, the value of each vertex is set to the maximum weight of the edges connected to it. Each object only has one unique match. To maximize the total sum of the matching weights, the algorithm continuously adjusts the values of the vertices and ultimately finds a perfect matching. In this paper, the detected lane points and the lanes represented by splines are the targets that need to be matched. The weight of an edge is defined as the distance from the detected lane point to the last control point of the splines. Given that the lane detection network has already categorized the lane points for each image, it is efficient to use the average distance from the lane points of each category to the control point as the edge weight. This approach significantly reduces the computational load required for matching by summarizing the distances for each category rather than considering each point individually. At the same time, if the distance exceeds a threshold of 10 m, it indicates that no suitable match exists. In this case, a new lane needs to be generated. The lane association process links the lane points detected in a single frame image with historical lane information. After lane association is completed for every 20 image frames, the system proceeds to the next step, which is lane mapping.

3.5.4. Spline generation. Catmull–Rom splines are used to fit lane points, generating coherent curves. The key to the splines lies in the selection of control points. Thus, the spline mapping problem can be viewed as generating an ordered sequence of control points from an unordered set of lane points. A coarse-to-fine method is designed for selecting control points. Initially, as shown in figure 4, a cubic curve is used to fit unordered lane points and then outliers are removed. To prevent an insufficient number of points and ensure the fitting effect, these lane points usually include detection results from

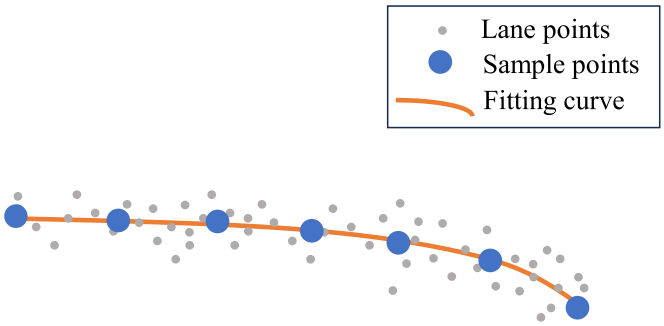

Figure 4. Process of fitting and sampling lane points. Cubic curves are used for the initial fitting of lane points. Sample points are obtained at equal distances along the curve.

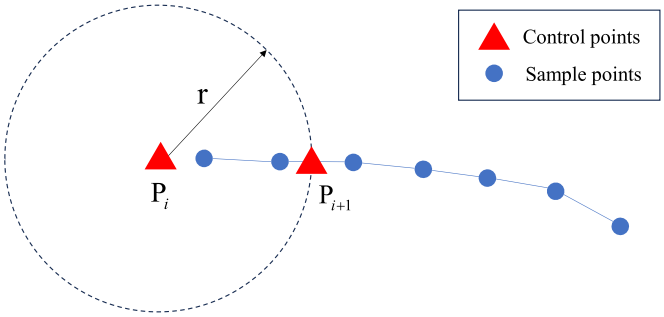

Figure 5. Method of extending splines. Next control point is determined by the intersection of a sphere centered on the previous control point and a line segment connecting sample points.

multiple frames. Curve fitting is performed separately for the x–y plane and the x–z plane. Subsequently, sampling points are obtained at equal intervals along the curve. Sample points help to extend the control points.

The distance between control points is equal and set to r, so the next control point must lie on a sphere centered on the previous control point with radius r. To extend the control points, one method is to find the intersection points between a sphere and a cubic polynomial. However, this is difficult to achieve. Our method converts the problem into finding the intersection points between the sphere and line segments to the sampled points, as shown in figure 5. Next control point lies on the intersection of a sphere centered on the previous control point and a line segment connecting sample points. This method significantly simplifies the calculations.

After updating the control points, we choose the latest two control points as normal vector to construct a plane. Any lane points that fall within this normal plane will be removed. A more intuitive way to describe this process is removing points behind the current position and keep points in front. The extension process will be repeated until all sample points are within the range of the control points. Finally, the positions of the control points will be further optimized. The residual between the lane points and spline can be defined as follows:

$$r = |p_l - \mathbf{p}(u)| \tag{14}$$

where p_l represents the detected lane points and $\mathbf{p}(u)$ is the spline, which is also a linear function of the control points. Each lane point has a corresponding parameter u associated. The parameter is the projection of the lane points onto the curve. The curve is divided into four equal parts according to u and converted into four-line segments. Lane points can find their corresponding orthogonal projection on the nearest line segment, and then obtain their parameter. After the control points are optimized, the complete lane map can be constructed using the curve equation. When new sensor data is obtained, the map can be extended in the same manner.

4. Results and discussion

We evaluate our proposed system's positioning and mapping performance on two real-world datasets, i.e. GVINS [3] and OpenLane [36]. GVINS dataset contains synchronized visual, inertial and GNSS raw measurements, including the sports field and urban environments. The OpenLane dataset is selected, because it includes 3D lane markings and covers various weather conditions and road scenarios. Our experiments are performed on a laptop with an Intel CPU (i5-12500 H, 2.5 GHz), 16-GB memory, and an NVIDIA graphics card (GeForce RTX 3060).

4.1. Localization

Two scenarios are selected in GVINS dataset to validate the localization performance of our proposed system. One is a sports field with open terrain, and the other is an urban road with complex a environment. The sports field has small view changes, with only a few buildings and stable satellite signals, making localization less challenging. City roads encompass various conditions, including tall buildings, low illumination, rapid movement, and highly-dynamic environments. This poses a significant challenge for localization. The performance is evaluated using the absolute pose error (APE).

$$\mathrm{APE} = \left\| \ln\left(T_{gt}^{-1} * T_{est}\right)^{\vee} \right\|_2 \tag{15}$$

where T_{gt} and T_{est} stand for the ground truth and estimate of pose. APE measures the error between the measured poses and ground truth. Main statistical metrics include the root mean square error (RMSE), mean error, standard deviation (STD) error, and maximum error. In this section, our proposed system is compared with GVINS. GVINS uses traditional rule-based feature detection and optical flow tracking for visual localization, whereas our system employs a learning-based approach to extract feature points and combines feature matching with optical flow tracking. A factor graph-based back-end optimization method is both used in our method and GVINS. Results are compared in the ENU coordinate system. Figure 6 shows the comparison between the trajectory computed by our system and the reference ground truth. Different colors represent the magnitude of the error. Figure 7 is a comparison of the trajectory generated by our method versus that of GVINS. It can be seen that the trajectory produced by our system aligns more closely with the reference trajectory than that of GVINS.

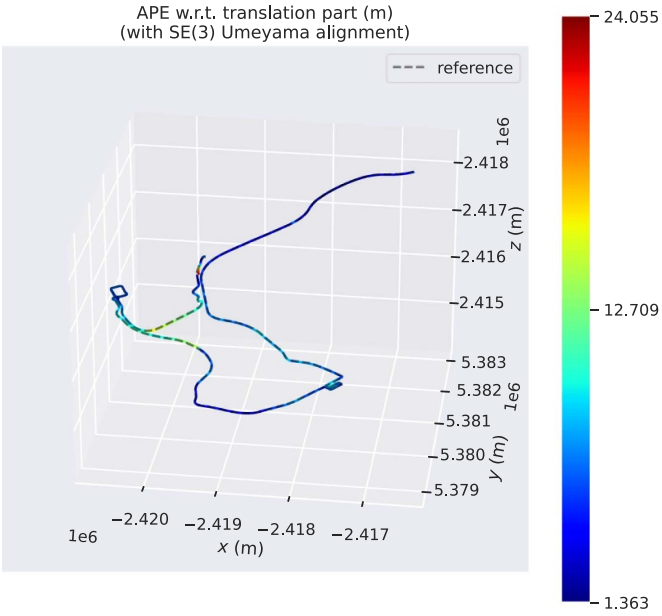

Figure 6. Comparison between the trajectory computed by our system and the reference ground truth.

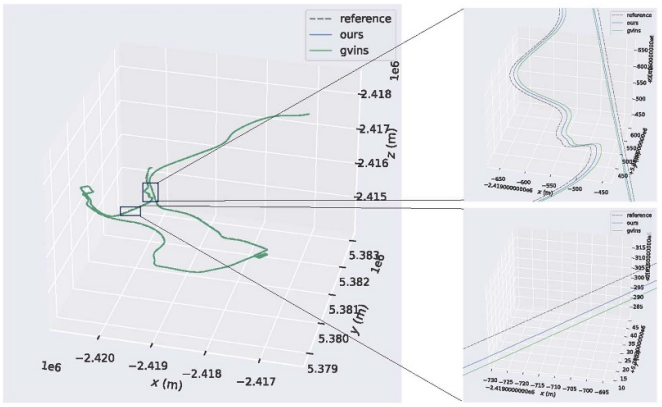

Figure 7. Comparison of the trajectory generated by our method versus GVINS.

Table 1 shows the comparison of APE results between the proposed method and GVINS on two different datasets. In the sports field, the results of our method and GVINS are similar. Because this is an open scene with very few buildings causing interference, the traditional optical flow is able to track feature points effectively. However, our deep learning visual front-end can extract more stable feature points even in low-texture scenes, leading to better localization accuracy. In the urban environment, with varying lighting, more buildings, and dynamic objects, our method achieves a 27.9% improvement in localization accuracy compared to GVINS. This is because traditional feature extraction and tracking methods easily fail in complex environments, whereas deep learning methods are more robust. When optical flow tracking fails, the learning-based descriptors is used for feature matching to ensure that the system can stably track feature points under any circumstances. We visualize the errors of both the methods in the urban scenario, as shown in figures 8 and 9. It can be observed

Table 1. Comparison of our method with GVINS based on APE results.

Scenario	Method	RMSE (m)	Mean (m)	STD (m)	Max (m)
Sports field	GVINS	0.59	0.52	0.26	1.33
	Ours	**0.50**	**0.44**	**0.22**	**1.02**
Urban driving	GVINS	9.94	8.98	4.27	26.76
	Ours	**7.17**	**6.67**	**2.64**	**24.06**

Note: The bold is used to highlight better experimental results.

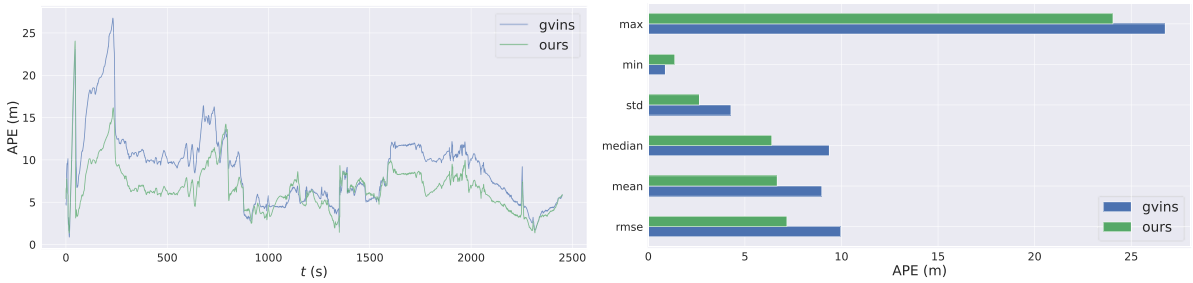

Figure 8. Comparison of our method with GVINS in terms of overall error and statistical metrics. (a) APE, which varies over time. (b) Various statistical metrics of the APE.

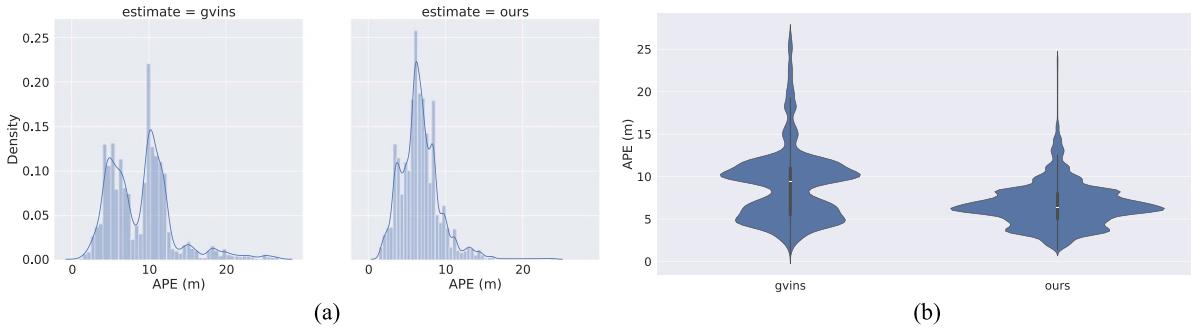

Figure 9. Comparison of error distribution between our method with GVINS. (a) Density distribution of the APE. (b) Violin plot of the APE.

that the error of our method is significantly lower compared to GVINS. Furthermore, the statistical data show that our proposed method performs better in the challenging scenes. From the error distribution, it can be seen that the errors of our method are generally smaller. This indicates that the stability of the error distribution is improved to some extent. To validate the system's localization performance in different environments, comparisons are also conducted on the fixed-wing dataset [11]. This dataset was collected from a fixed-wing aircraft and includes the entire process from take-off to landing, as well as the cruise phase in between. It also includes data from three types of sensors: visual, IMU and GNSS. It is challenging for localization due to rapid viewpoint changes and lack of stable feature points. We compared our system with GVINS and InGVIO [11] in two different scenarios. InGVIO is a multi-sensor SLAM system based on consistent invariant filter, which is different from the optimization method in our system and GVINS. Table 2 shows the comparison of APE results.

For the two scenarios, fw_gvi_medium contains spirals and loops, which add greater difficulty to the localization task compared to fw_gvi_easy. Traditional methods make it difficult for GVINS to stably extract and track feature points. Our method, using a deep learning front-end, provides sufficient robustness to handle these challenging scenarios. As a result, our system achieves better localization accuracy compared to both GVINS and InGVIO.

4.2. Lane mapping

OpenLane dataset provides the ground truth, which allows us to measure the accuracy of lane mapping and evaluate its performance. The F1 score is used as the evaluation metric,

$$\mathrm{F1} = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$
$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{16}$$

where TP, FP and FN represent the true positives, false positives and false negatives respectively. OpenLane dataset

Table 2. Comparison of the APE on fixed-wing dataset.

Scenario	Method	RMSE (m)
fw_gvi_easy	GVINS	4.66
	InGVIO	4.21
	Ours	**4.18**
fw_gvi_medium	GVINS	7.06
	InGVIO	6.33
	Ours	**6.25**

Note: The bold is used to highlight better experimental results.

Table 3. Comparison of our lane mapping with other 3D lane detection method based on F1 score on OpenLane dataset.

	Up & down	Curve	Extreme weather	Night	Intersection	Merge & split	Overall
3D-LaneNet [37]	40.8	46.5	47.5	41.5	32.1	41.7	44.1
Persformer [36]	45.6	58.7	54.0	50.0	41.6	53.1	50.5
Anchor3DLane [38]	45.5	56.2	51.9	47.2	44.2	50.5	53.1
BEV-LaneDet [39]	**48.7**	**63.1**	53.4	53.4	50.3	53.7	58.4
Ours	47.5	61.2	**55.2**	**56.6**	**52.2**	**57.4**	**60.1**

Note: The bold is used to highlight better experimental results.

provides only lane annotations for single frame. Therefore, the global map is projected onto one camera frame for comparison purpose. As the OpenLane dataset does not contain sensor data related to localization, the lane map is based on the pose information provided. We select lanes within a 50 m range ahead. If the distance between map points and ground truth points is less than 1.5 m, they are considered correct. When more than 75% of the points are correct, the frame is classified as TP.

The experimental results are listed in table 3. The results indicate that our proposed method can improve the recall and precision of lane mapping in most scenarios. In challenging scenarios such as extreme weather and nighttime, where visual interference is significant, our method achieves higher robustness through the association and optimization of control points. When faced with road intersections and splitting, spline curves, which are closely related to geometric relationships, are more suitable for representing lanes. Although our method is less effective than BEV-LaneDet in up/down and curved scenarios, it shows better performance in complex environments. Our spline-based extension and optimization method fully utilizes historical information to construct more consistent and accurate lanes.

Additionally, the performance of lane mapping is compared on the ONCE-3DLane dataset [40]. It is a real-world 3D lane dataset, which contains different locations, lighting conditions, and weather conditions. The experimental results are presented in table 4.

CD represents the chamfer distance between the predicted lanes and the ground truth. It can be observed that our method outperforms existing methods on this dataset. The results from the above two datasets demonstrate that our method has better performance and potential in complex environments.

Table 4. Comparison with 3D lane detection methods on ONCE-3DLane dataset.

Method	F1-Score	CD error (m)
3D-LaneNet	44.73	0.127
Persformer	74.33	0.074
Anchor3DLane	74.87	0.060
Ours	**77.65**	**0.055**

Note: The bold is used to highlight better experimental results.

The performance of the complete system is tested in an urban scenario of the GVINS dataset. Figure 10 shows real-world scenes and the results of lane mapping, including two scenarios featuring roundabouts and intersections respectively. With the help of a high-precision localization module and the spline curve fitting module, our method can construct complete lane maps even in such complex scenarios. Owing to the reliance on lane detection, the lane map may be incomplete and susceptible to outliers.

4.3. Running time

To demonstrate real-time performance of the system, we measured the runtime of certain system modules. The effect of each module on the system performance is detailed in table 5, where FT represents the feature extraction and detection, LD denotes lane detection and LM indicates lane mapping.

It can be observed that the deep learning module does not consume much time thanks to GPU acceleration. The feature extraction and tracking module take nearly the same amount of time as traditional methods, whereas lane detection takes only 3 ms. Furthermore, the spline fitting approach is also fast in terms of runtime. Our method adds lane detection and mapping

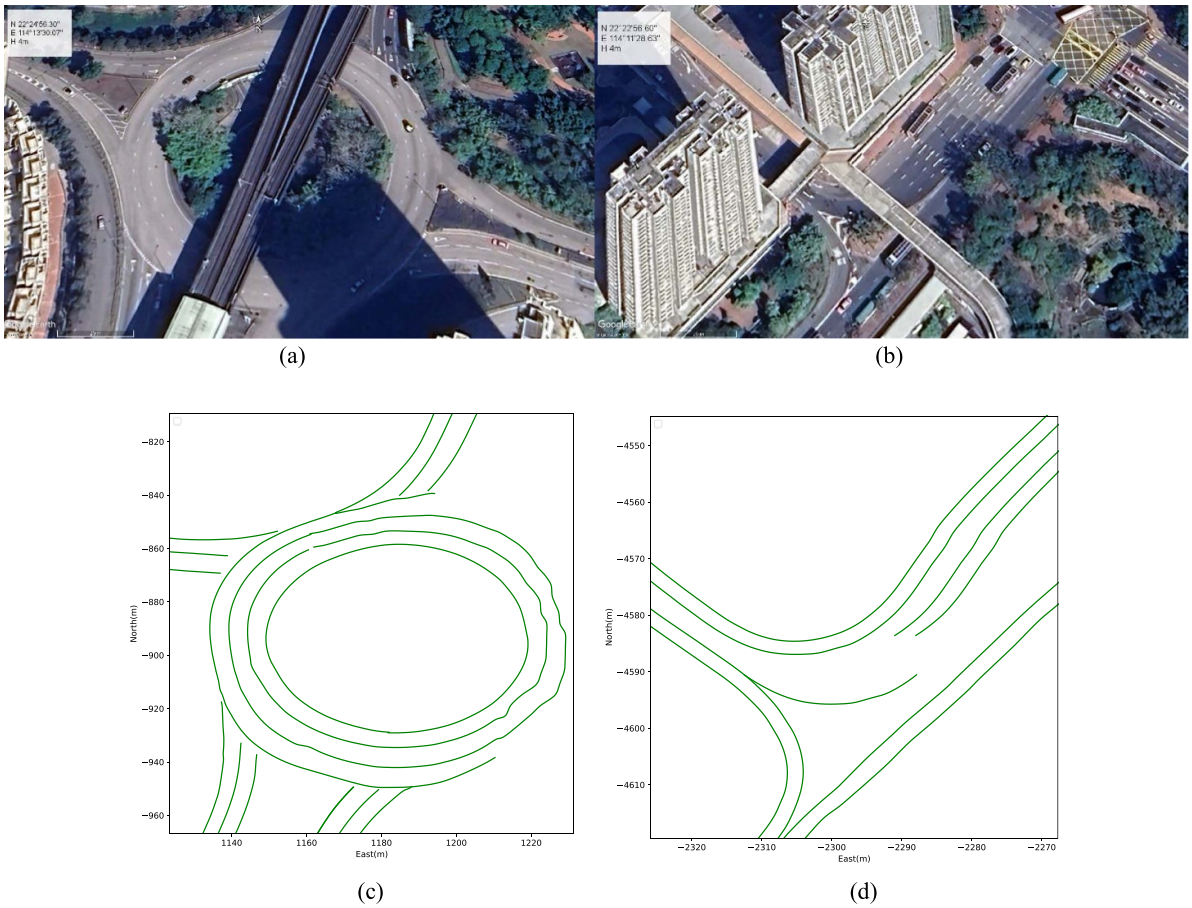

Figure 10. (a) Satellite map of scenario A. (b) Satellite map of scenario B. (c) Vector lane map of scenario A. (d) Vector lane map of scenario B.

Table 5. Running time of GVINS and our method.

	FT	LD	LM	Total
GVINS	21.91	N/A	N/A	22.68
Ours	22.54	3.22	6.47	30.40

modules compared to GVINS, whereas it still maintains the capability for real-time operation. It should be noted that since these modules run in parallel, the total runtime of the entire system is not simply the sum of runtime of each individual module.

4.4. Ablation study

To further validate the impact of the proposed modules on performance improvement, we conduct ablation studies. As shown in table 6, the ablation study is evaluated in the sports field scenario. The baseline refers to a loosely coupled multi-sensor SLAM system, where only the GNSS-derived coordinate values are used in the nonlinear optimization. The tightly coupled system, on the other hand, uses raw GNSS measurements for optimization. Our system uses both the ALIKE network and a tightly coupled approach. Through the comparison, it can be found that the modules used in our system indeed contribute to the improvement of localization accuracy.

Table 6. Comparison of the effects of different modules on localization performance.

Method	RMSE (m)
Baseline	1.32
Baseline + ALIKE	1.01
Baseline + tight coupling	0.59
Ours	**0.50**

Note: The bold is used to highlight better experimental results.

5. Conclusions

Our proposed system enhances the robustness in complex environments by integrating the neural network ALIKE,

thereby mitigating the impact of visual interference on localization. The improved localization precision leads to more accurate mapping results. A vector map is generated by fitting lanes with Catmull–Rom splines and combining them with global positioning. The historical information association capability of SLAM and Catmull–Rom helps to construct more consistent and accurate lanes. However, the system's performance relies heavily on the accuracy of lane detection. Noise in the network output will affect mapping stability. In our next work plan, we will further refine the lane detection network to address the said accuracy problem and explore transformer-based lane detection methods [41]. Besides, we will also manage to project the lane mapping via head-up displays [42, 43] for the advanced driver-assistance system.

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Funding

This work is supported by Ministry of Industry and Information Technology of China (GO0300164/001) and Natural Science Foundation of Chongqing Municipality (cstb2023nscq-msx0465).

Conflict of interest

The authors declare no conflicts of interest.

ORCID iDs

Tianmu Gao https://orcid.org/0009-0005-4760-4930
Chao Ping Chen https://orcid.org/0000-0002-4366-4304

References

[1] Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I and Leonard J J 2016 Past, present, and future of simultaneous localization and mapping: toward the robust-perception age *IEEE Trans. Robot.* **32** 1309–32
[2] Qin T, Li P and Shen S 2018 VINS-Mono: a robust and versatile monocular visual-inertial state estimator *IEEE Trans. Robot.* **34** 1004–20
[3] Cao S, Lu X and Shen S 2022 GVINS: tightly coupled GNSS–visual–inertial fusion for smooth and consistent state estimation *IEEE Trans. Robot.* **38** 2004–21
[4] Catmull E 1974 A Class of Local Interpolating Splines *Computer Aided Geometric Design/Academic Press* 317–26
[5] Klein G and Murray D 2007 Parallel tracking and mapping for small AR workspaces *Int. Symp. on Mixed and Augmented Reality (ISMAR)* pp 225–34
[6] Forster C, Pizzoli M and Scaramuzza D 2014 SVO: fast semi-direct monocular visual odometry *2014 Int. Conf. on Robotics and Automation (ICRA)* pp 15–22
[7] Mur-Artal R, Montiel J M and Tardos J D 2015 ORB-SLAM: a versatile and accurate monocular SLAM system *IEEE Trans. Robot.* **31** 1147–63
[8] Sun K, Mohta K, Pfrommer B, Watterson M, Liu S, Mulgaonkar Y, Taylor C J and Kumar V 2018 Robust stereo visual inertial odometry for fast autonomous flight *IEEE Robot. Autom. Lett.* **3** 965–72
[9] Huang G, Kaess M and Leonard J J 2014 Towards consistent visual-inertial navigation *Int. Conf. on Robotics and Automation (ICRA)* pp 4926–33
[10] Qin T *et al* 2019 A general optimization-based framework for Global pose estimation with multiple sensors (arXiv:1901.03642)
[11] Liu C, Jiang C and Wang H 2023 InGVIO: a consistent invariant filter for fast and high-accuracy GNSS-visual-inertial odometry *IEEE Robot. Autom. Lett.* **8** 1850–7
[12] Li T, Pei L, Xiang Y, Yu W and Truong T K 2022 P3-VINS: tightly-coupled PPP/INS/visual SLAM based on optimization approach *IEEE Robot. Autom. Lett.* **7** 7021–7
[13] Shan T, Englot B, Meyers D, Wang W, Ratti C and Rus D 2020 LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping *Int. Conf. on Intelligent Robots and Systems (IROS)* pp 5135–42
[14] Li D, Shi X, Long Q, Liu S, Yang W, Wang F, Wei Q and Qiao F 2020 Dxslam: a robust and efficient visual slam system with deep features *Int. Conf. on Intelligent Robots and Systems (IROS)* pp 4958–65
[15] DeTone D, Malisiewicz T and Rabinovich A 2018 SuperPoint: self-supervised interest point detection and description *Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)* pp 224–36
[16] Li Y, Chen C P, Maitlo N, Mi L, Zhang W and Chen J 2020 Deep neural network-based loop detection for visual simultaneous localization and mapping featuring both points and lines *Adv. Intell. Syst.* **2** 1900107
[17] Kang Y, Xu Y, Chen C P, Li G and Cheng Z 2021 Simultaneous tracking, tagging and mapping for augmented reality *Int. Conf. on Display Technology (ICDT)* vol 52 pp 31–33
[18] Wang S, Clark R, Wen H and Trigoni N 2017 Deepvo: towards end-to-end visual odometry with deep recurrent convolutional neural networks *Int. Conf. on Robotics and Automation (ICRA)* pp 2043–50
[19] Almalioglu Y, Turan M, Saputra M R U, de Gusmão P P B, Markham A and Trigoni N 2022 SelfVIO: self-supervised deep monocular visual–inertial odometry and depth estimation *Neural Netw.* **150** 119–36
[20] Shamwell E J, Lindgren K, Leung S and Nothwang W D 2020 Unsupervised deep visual-inertial odometry with online error correction for RGB-D imagery *IEEE Trans. Pattern Anal. Mach. Intell.* **42** 2478–93
[21] Mildenhall B *et al* 2020 NeRF: representing scenes as neural radiance fields for view synthesis *European Conf. on Computer Vision (ECCV)* 405–21
[22] Kerbl B, Kopanas G, Leimkühler T and Drettakis G 2023 3D Gaussian splatting for real-time radiance field rendering *ACM Trans. Graph.* **42** 1–14
[23] Qin T, Chen T and Su Q 2020 AVP-SLAM: semantic visual mapping and localization for autonomous vehicles in the parking lot *Int. Conf. on Intelligent Robots and Systems (IROS)* pp 5939–45
[24] Qin T, Zheng Y, Chen T and Su Q 2021 A light-weight semantic map for visual localization towards autonomous

driving *Int. Conf. on Robotics and Automation (ICRA)* pp 11248–54
[25] Zhou Y, Li X, Li S and Wang X 2022 Visual mapping and localization system based on compact instance-level road markings with spatial uncertainty *IEEE Robot. Autom. Lett.* **7** 10802–9
[26] Qiao Z, Yu Z, Yin H and Shen S 2023 Online monocular lane mapping using Catmull-Rom spline *Int. Conf. on Intelligent Robots and Systems (IROS)* pp 7179–86
[27] Liao B *et al* 2023 Maptr: structured modeling and learning for online vectorized hd map construction *Int. Conf. on Learning Representations (ICLR)*
[28] Qiao L, Ding W, Qiu X and Zhang C 2023 End-to-end vectorized HD-map construction with piecewise Bézier curve *Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 13218–28
[29] Zhao X, Wu X, Miao J, Chen W, Chen P C Y and Li Z 2023 Accurate and lightweight keypoint detection and descriptor extraction *IEEE Trans. Multimedia* **25** 3101–12
[30] Hartley R and Zisserman A 2003 *Multiple View Geometry in Computer Vision* (Cambridge University Press)
[31] Agarwal S *et al* 2012 Ceres solver: tutorial & reference *Google Inc.* **2** 8
[32] Sibley G, Matthies L and Sukhatme G 2010 Sliding window filter with application to planetary landing *J. Field Robot.* **27** 587–608
[33] Qin Z, Wang H and Li X 2020 Ultra fast structure-aware deep lane detection *European Conf. on Computer Vision (ECCV)* pp 1–16
[34] Kuhn H W 1955 The Hungarian method for the assignment problem *Nav. Res. Logist. Q.* **2** 83–97
[35] Munkres J 1957 Algorithms for the assignment and transportation problems *J. Soc. Ind. Appl. Math.* **5** 32–38
[36] Chen L *et al* 2022 Persformer: 3d lane detection via perspective transformer and the openlane benchmark *European Conf. on Computer Vision (ECCV)* pp 550–67
[37] Garnett N, Cohen R, Pe'er T, Lahav R and Levi D 2019 3d-lanenet: end-to-end 3d multiple lane detection *Proc. IEEE/CVF Int. Conf. on Computer Vision* pp 2921–30
[38] Huang S, Shen Z, Huang Z, Ding Z H, Dai J, Han J, Wang N and Liu S 2023 Anchor3dlane: learning to regress 3d anchors for monocular 3d lane detection *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 17451–60
[39] Wang R, Qin J, Li K, Li Y, Cao D and Xu J 2023 BEV-LaneDET: an efficient 3d lane detection based on virtual camera via key-points *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 1002–11
[40] Yan F *et al* 2022 Once-3dlanes: building monocular 3d lane detection *Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 17143–52
[41] Zhuang L, Jiang T, Qiu M, Wang A and Huang Z 2024 Transformer generates conditional convolution kernels for end-to-end lane detection *IEEE Sens. J.* **24** 28383–96
[42] Chen C P, Cui Y, Ye Y, Yin F, Shao H, Lu Y and Li G 2021 Wide-field-of-view near-eye display with dual-channel waveguide *Photonics* **8** 557
[43] Chen C P, Ma X, Zou S P, Liu T, Chu Q, Hu H and Cui Y 2023 Quad-channel waveguide-based near-eye display for metaverse *Displays* **81** 102582