2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341387

GR-SLAM: Vision-Based Sensor Fusion SLAM for Ground Robots on Complex Terrain

Yun Su, Ting Wang, Chen Yao, Shiliang Shao, Zhidong Wang

***Abstract*—In recent years, many excellent SLAM methods based on cameras, especially the camera-IMU fusion (VIO), have emerged, which has greatly improved the accuracy and robustness of SLAM. However, we find through experiments that most of the existing VIO methods perform well on drones or drone datasets, but for ground robots on complex terrain, they cannot continuously provide accurate and robust localization results. Some researchers have proposed methods for ground robots, but most of them have limited applications due to the assumption of plane motion. Therefore, this paper proposes GR-SLAM for the localization of ground robots on complex terrain, which can fuse camera, IMU, and encoder data in a tightly coupled scheme to provide accurate and robust state estimation for robots. First, an odometer increment model is proposed, which can fuse the encoder and IMU data to calculate the robot pose increment on manifold, and calculate the frame constraints through the pre-integrated increment. Then we propose an evaluation algorithm for multi-sensor measurements, which can detect abnormal data and adjust its optimization weight. Finally, we implement a complete factor graph optimization framework based on sliding window, which can tightly couple camera, IMU, and encoder data to perform state estimation. Extensive experiments are conducted based on a real ground robot and the results show that GR-SLAM can provide accurate and robust state estimation for ground robots.**

I. Introduction

SLAM is a basic and crucial task for ground robots to navigate in unknown environments. In the absence of GPS signals, SLAM can be accomplished via many types of sensors [1] [2] [3] [4] such as Lidar, camera, ultrasound, UWB, etc. However, applications such as service robots, sweeping robots, or logistics robots are relatively price sensitive. Although Lidar can achieve very good results, its expensive price limits its large-scale application on products. By contrast, cameras feature small size, low price, and easy installation, so they have become the most popular and most promising sensor for SLAM tasks. In recent years, many researchers have done a lot of excellent work in this area [5] [6] [7] [8]. Moreover, the researchers further fused the IMU with the camera [9] [10] [11] [12], which greatly improved the accuracy and robustness of visual SLAM.

*Research supported by National Key R&D Program of China (2017YFC0822201) and (2017YFC0822203).

Y. Su is with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China and Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, China and University of Chinese Academy of Sciences, Beijing, China. (Corresponding author, e-mail: robosu12@gmail.com).

T. Wang, C. Yao and S. L. Shao are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, China. (e-mail: {wangting, cyao, shaoshiliang}@sia.cn).

Z. D. Wang is with the Department of Advanced Robotics, Chiba Institute of Technology, Chiba, Japan. (e-mail: zhidong.wang@it-chiba.ac.jp).

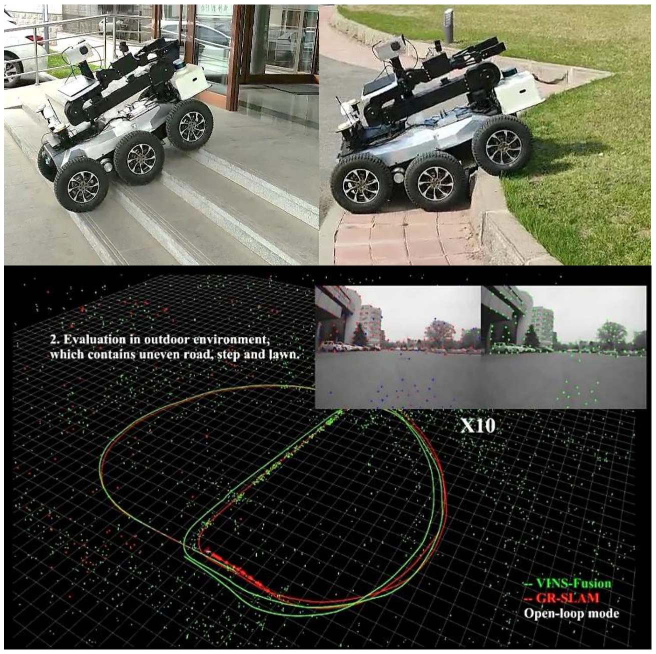

2. Evaluation in outdoor environment, which contains uneven road, step and lawn.
X10
-- VINS-Fusion
-- GR-SLAM
Open-loop mode

Figure 1. Ground robot with GR-SLAM running on Complex Terrain. Top: The robot is crossing stairs and steps. Bottom: Results visualization of VINS-Fusion and GR-SLAM in outdoot large-scale environment, which GR-SLAM only produces small error when the robot returns to starting point.

Although the VIO algorithm already has very good performance, we find through experiments that the existing algorithm performs well on drones and public data sets, but the performance declines when the algorithm is applied to ground robots. We summarize the following reasons:

- Ground robots vibrate when running on the road, and the vibration differs on different roads. The vibration will generate random noise and reduce the accuracy of the VIO, and even state mutation.
- Ground robots often have some special motions, such as constant-speed linear motion, start-stop operation, etc., which causes errors in the VIO [13] [14].
- When the image information has been lost for a short time (less than 1s[1]), the IMU can be used to propagate robot state. However, for long-term image data loss, the VIO algorithm will quickly produce large error.

The above reasons cause the performance of the existing VIO algorithm to decline or even diverge, and to fail to provide continuously accurate and robust localization results. In practical applications, even if an unexpected situation

[1] The length of the propagation time depends on the quality of the IMU. A high-quality IMU can still maintain a certain accuracy for a long period of time, but a low-cost IMU will cause a larger error if it exceeds 3s.

occurs, the algorithm should still be able to provide an acceptable state estimation result for the robot. Because in practice, the continuity of the robot state is more important than the local accuracy.

Based on the above problems, many researchers have proposed SLAM methods for ground robots [13] [14] [15] [16] [17]. For example, [13] and others have proven that degraded movements will bring errors to the VIO system, and proposed to add wheel odometer measurement information to the VIO system to improve localization accuracy. [16] designed a complete SLAM framework, which can fuse the measurements of cameras, IMU and odometer. [15] combined visual measurement and motion beyond SE(2) to propose a novel SE(2)-XYZ constraint to improve the localization accuracy and robustness of ground robots. However, the above studies are based on the assumption of plane motion and are only applicable to scenes with flat ground such as factories. To solve the problem of robot movement in non-planar scenes, [14] proposed to use the parameterization of polynomial to approximate the motion flow pattern, but it can only be applied to scenes with small changes in road gradients, such as shopping malls and streets and not suitable for more general scenes with elements such as slopes, stairs, etc.

Therefore, we propose GR-SLAM for the localization of ground robots on complex terrain, which can fuse camera, IMU, and encoder data in a tightly coupled scheme to provide accurate and robust state estimation for robots. We summarize the contributions of this paper as follows:

- An odometer increment model is proposed, which can fuse encoder and IMU data to calculate robot pose increment on manifold to fit complex terrain in large scale 3D environments.
- A complete factor graph optimization framework is proposed, which can tightly couple camera, IMU, and encoder data to perform state estimation. Also it can detect abnormal data and adjust the optimization weight of each sensor.
- Extensive experiments based on a real ground robot in indoor and outdoor environments demonstrate the accuracy and robustness of the proposed GR-SLAM for ground robots.

The remainder of this paper is organized as follows. Section II and III introduce the proposed system in the preliminary theory and the proposed method. The experimental setup and results are reported and analyzed in Section IV. Finally, we draw conclusions in Section V.

II. Preliminaries

A. Frames and Notation

The coordinate systems to implement the proposed GR-LOAM are shown in Fig. 2. We assume that a ground robot navigates in a reference world coordinate system $\{W\}$. The origin of the encoder coordinate system is the geometric center of the robot chassis, and the robot moves along the positive x axis. Let $\mathbf{p}_{AB_k}$ and $\mathbf{R}_{AB_k}$ represent the position and orientation of coordinate system $\{B\}$ with respect to $\{A\}$ at time k, respectively. We define the state vector as follows:

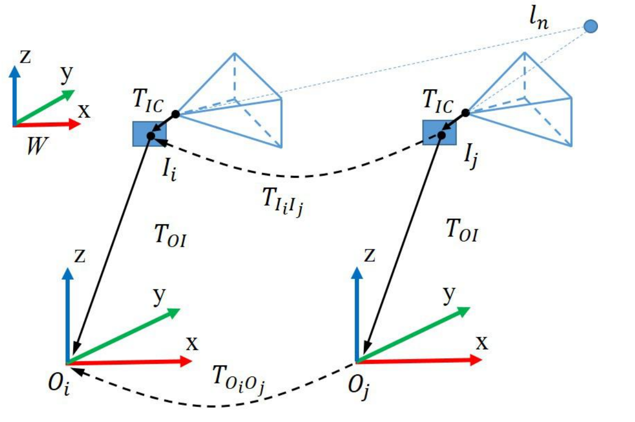


Figure 2. Coordinate definitions and rigid transformations from timestep i to timestep j. $\mathbf{T}_{IC}$ and $\mathbf{T}_{OI}$ are extrinsic parameters for IMU–camera and robot–IMU, respectively. $\mathbf{T}_{I_iI_j}$ and $\mathbf{T}_{O_iO_j}$ are transformations in IMU and robot coordinate, respectively. l_n is the n-th feature point observed.

$$\begin{aligned}
&\boldsymbol{\chi} = \left[\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_n, \mathbf{T}_{IC}, \lambda_0, \lambda_0, \dots \lambda_m\right] \\
&\mathbf{x}_k = \left[\mathbf{p}_{wI_k}, \mathbf{q}_{wI_k}, \mathbf{v}_{wI_k}, \mathbf{b}_a, \mathbf{b}_g\right], k \in [0, n] \\
&\mathbf{T}_{IC} = \left[\mathbf{p}_{IC}, \mathbf{q}_{IC}\right]
\end{aligned} \tag{1}$$

Where $\mathbf{x}_k$ is the state of the k-th keyframe in the sliding window, $\mathbf{T}_{IC}$ is the extrinsic parameter of IMU and the camera, and is calibrated in advance by kalibr [19].

B. Parameters Calibration

1) Wheel odometry intrinsic parameters calibration:

The encoder can measure the rotation angle of the motor φ, and then it is divide by the gear ratio of the reducer R to get the rotation angle of the wheel. If we want to integrate the encoder data into the pose estimation of the robot, then we also need to know the radiuses of the left and right wheels r_L, r_R [2], and the distance between the two wheels b [20], which is the internal parameter of the wheel odometer. Assuming the robot is moving in the plane, we can calculate robot motion information from the encoder data:

$$\begin{bmatrix} d_{ij} \\ \varphi_{ij} \end{bmatrix} = \begin{bmatrix} \frac{r_L}{2R} & \frac{r_R}{2R} \\ -\frac{r_L}{bR} & \frac{r_R}{bR} \end{bmatrix} \begin{bmatrix} \omega_{ij}^L \\ \omega_{ij}^R \end{bmatrix} = \mathbf{J} \begin{bmatrix} \omega_{ij}^L \\ \omega_{ij}^R \end{bmatrix} \tag{2}$$

Where ω_{ij}^L, ω_{ij}^R is the angle that the left and right wheel encoders rotate from time i to time j. d_{ij} is the movement distance of the robot. φ_{ij} is the rotation angle of the robot, and $\mathbf{J}$ is the Jacobian matrix of the odometer intrinsic parameters.

We control the robot to follow a rectangular trajectory on a flat ground. The closed-loop results of VINS-Fusion [21] are used as ground truth, a series of encoder angle data and robot

[2] The robot we use has six wheels, and we model the robot as a two-wheeled robot. We also have tested on four-wheeled robots and crawler robots, and this model is still applicable.

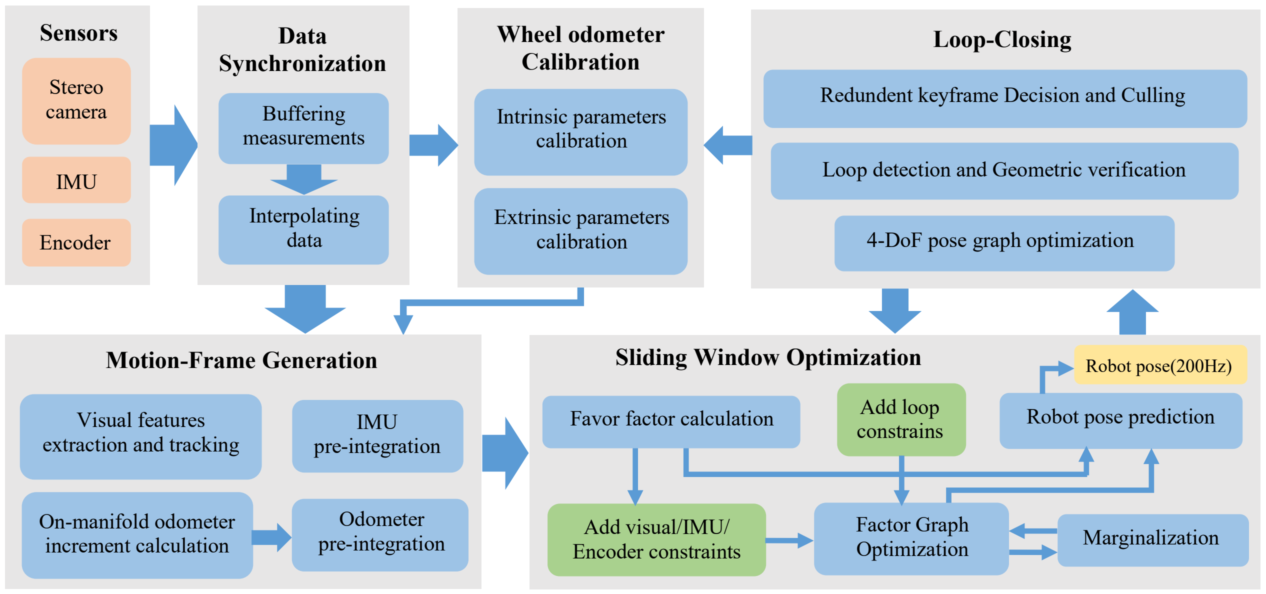

Sensors
Stereo camera
IMU
Encoder
Data Synchronization
Buffering measurements
Interpolating data
Wheel odometer Calibration
Intrinsic parameters calibration
Extrinsic parameters calibration
Loop-Closing
Redundent keyframe Decision and Culling
Loop detection and Geometric verification
4-DoF pose graph optimization
Motion-Frame Generation
Visual features extraction and tracking
IMU pre-integration
On-manifold odometer increment calculation
Odometer pre-integration
Sliding Window Optimization
Robot pose(200Hz)
Favor factor calculation
Add loop constrains
Robot pose prediction
Add visual/IMU/ Encoder constraints
Factor Graph Optimization
Marginalization

Figure 3. Pipeline of the proposed GR-SLAM. The system starts with measurements data synchronization. The frame generation module process all measurements and creates Motion-Frame with multiple information. The optimization thread tightly couples feature points, pre-integrated IMU and encoder data, and constrains from loop-closing. Finally, the loop-closing thread performs global optimization to eliminate drift and achieve map reuse.

motion data are collected, and then a batch of least squares is constructed to solve $\mathbf{J}$.

2) Extrinsic parameters calibration:

If the measured value $\mathbf{T}_{I_iI_j}$ of VIO and the measured value $\mathbf{T}_{O_iO_j}$ of the odometer are known, then we can get according to the external parameter transformation [22]:

$$\mathbf{T}_{O_iO_j}\mathbf{T}_{OI} = \mathbf{T}_{OI}\mathbf{T}_{I_iI_j} \tag{3}$$

From (3), we can use the measurement of the wheel odometer to predict the measurement in the IMU coordinate:

$$\mathbf{T}'_{I_iI_j} = \mathbf{T}_{OI}^{-1}\mathbf{T}_{O_iO_j}\mathbf{T}_{OI} \tag{4}$$

We can get the measurement error between two sensors:

$$e_{ij}(\mathbf{T}_{OI}) = \log(\mathbf{T}'_{I_iI_j}\mathbf{T}_{I_iI_j}^{-1}) \tag{5}$$

We define the covariance of the odometer measurement as $\mathbf{\Sigma}_{ij}$ and take the VINS-Fusion result as the reference value, then we can estimate the extrinsic parameters matrix $\mathbf{T}_{OI}$ of the odometer and IMU by minimizing the following cost function:

$$\hat{\mathbf{T}}_{OI} = \arg\min_{\mathbf{T}_{OI}} \sum_{<i,j>\in S} e_{ij}^T(\mathbf{T}_{OI})\mathbf{\Sigma}_{ij}^{-1}e_{ij}(\mathbf{T}_{OI}) \tag{6}$$

Where S represents a series of measurements.

III. Multi-Sensor Fusion and Graph Optimization

The GR-SLAM algorithm framework is shown in Fig. 3. It includes 4 threads. Front-end threads are formed by sensor data synchronization and motion frame generation. The wheel odometer calibration thread is started only when the robot platform runs for the first time, and parameters are fixed after the fine calibration is made. The core thread is sliding window optimization, which tightly coupled multi-sensors are used to localize the robot. The loop-closing thread receives the key frames of the optimization thread and maintains the database.

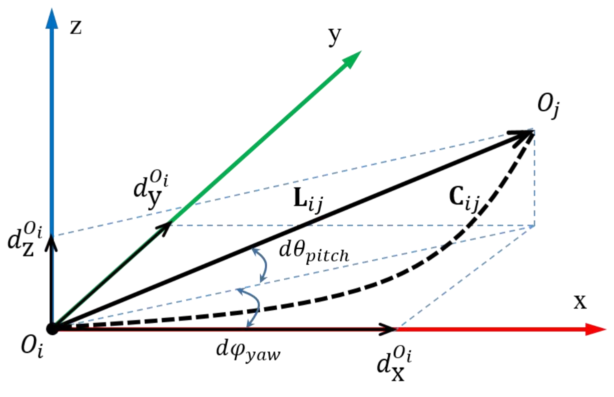


Figure 4. Illustration of odometer increment model on manifold ($\mathbf{C}_{ij}$ is the robot trajectory from timestep *i* to timestep *j* and $\mathbf{L}_{ij}$ is the motion vector).

At the same time, it performs loop detection and optimization, and outputs long-term drift-free robot pose.

A. On-Manifold Odometer increment model

We focus on SLAM for ground robots on complex terrain with elements such as large gradient changes (e.g., slopes and stairs). Most existing methods are based on absolute or random plane assumptions [22] [23] [24], limiting their applicability in complex 3D environments. Instead, we propose a pose increment model on manifold by combining the IMU and robot wheel encoder data to more suitably fit complex rough terrain.

The frequency of the encoder and IMU is 200Hz. During the time cycle measured by an encoder, the pose of the ground robot at timestep i is used as reference coordinate. We assume that the robot displacement at the next timestep, j, is described as shown in Fig. 4, where curve $\mathbf{C}_{ij}$ is the robot trajectory, vector $\mathbf{L}_{ij}$ is the chord length, and the motion vector is parameterized as the following form:

$$\mathbf{L}_{ij}=\begin{bmatrix} l_{ij} & d\varphi_{yaw} & d\theta_{pitch} \end{bmatrix}^{\mathrm{T}} \tag{7}$$

Where l_{ij} is the vector length, and $d\theta_{pitch}$ and $d\varphi_{yaw}$ are the changes of the robot's pitch and yaw, respectively.

We can calculate $d\varphi_{yaw}$ and $d\theta_{pitch}$ based on the current IMU measurement and bias. After calibrating the intrinsic parameter matrix $\mathbf{J}$ of the odometer, we can use the encoder angle to calculate the length of the trajectory curve c_{ij} by using (1). Since the rotation angle of the robot is small within 5ms, we can calculate length of L_{ij} through geometric derivation and approximation:

$$l_{ij}=c_{ij}\cos(d\varphi_{yaw}/3) \tag{8}$$

Then we project the motion vector to O_i coordinate axis to get the position increment at time *j* relative to time *i*:

$$dp_{O_iO_j}=\begin{bmatrix} c_{ij}\cos(d\varphi_{yaw}/3)\cos(d\theta_{pitch})\cos(d\varphi_{yaw}) \\ c_{ij}\cos(d\varphi_{yaw}/3)\cos(d\theta_{pitch})\sin(d\varphi_{yaw}) \\ -c_{ij}\cos(d\varphi_{yaw}/3)\sin(d\theta_{pitch}) \end{bmatrix} \tag{9}$$

B. Sensor factors

1) Camera Factor:

GR-SLAM supports monocular and binocular cameras set. We extract corner features for each frame and track them with optical flow. Then reprojection are made according to the matched features to construct a visual constraint factor. When the *l*-th feature of the *i*-th image is observed by the *j*-th image, the reprojection error is defined as follows:

$$\begin{aligned} \mathbf{r}_C(\mathbf{z}_l^{C_j},\boldsymbol{\chi}) &= \mathbf{z}_l^{C_j} - h_l^{C_j}(\mathbf{p}_i,\mathbf{R}_i,\mathbf{p}_j,\mathbf{R}_j,\lambda_l) \\ &= \begin{bmatrix} u_l^{C_j} \\ v_l^{C_j} \end{bmatrix} - \pi_c(T_{IC}^{-1}T_j^{-1}T_iT_{IC}\pi_c^{-1}(\lambda_l,\begin{bmatrix} u_l^{C_i} \\ v_l^{C_i} \end{bmatrix})) \end{aligned} \tag{10}$$

Where $\begin{bmatrix} u_i^l & v_i^l \end{bmatrix}^{\mathrm{T}}$ is the coordinate where the *l*-th feature is first observed by the *i*-th frame image, and $\begin{bmatrix} u_j^l & v_j^l \end{bmatrix}^{\mathrm{T}}$ is the coordinate of the feature in the *j*-th frame. π_c and π_c^{-1} are the projection and back-projection functions of the projection model of the camera, respectively. λ_l is the inverse depth of the feature point.

In binocular mode, we use left-right camera matching to initialize the inverse depth of the feature points. Moreover, added is the reprojection factor of the left camera in the time space, as well as the reprojection factor of the left and right camera in the spatial space.

2) IMU Factor:

Because the frequency of IMU is higher than camera, we use pre-integration method to integrate multiple IMU measurements between motion frames to construct an IMU constraint factor. For two consecutive motion frames *k* and *k*+1, we define the pre-integration residual of the IMU as follows:

$$\begin{aligned} \mathbf{r}_I(\mathbf{z}_{I_kI_{k+1}}^I,\boldsymbol{\chi}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}_{I_kI_{k+1}}^I & \delta\boldsymbol{\beta}_{I_kI_{k+1}}^I & \delta\boldsymbol{\theta}_{I_kI_{k+1}}^I & \delta\mathbf{b}_a & \delta\mathbf{b}_g \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \mathbf{R}_{I_kw}(\mathbf{p}_{wI_{k+1}} - \mathbf{p}_{wI_k} + \frac{1}{2}\mathbf{g}_w\Delta t_k^2 - \mathbf{v}_{wI_k}\Delta t_k) - \boldsymbol{\alpha}_{I_kI_{k+1}}^I \\ \mathbf{R}_{I_kw}(\mathbf{v}_{wI_{k+1}} + \mathbf{g}_w\Delta t_k - \mathbf{v}_{wI_k}) - \boldsymbol{\beta}_{I_kI_{k+1}}^I \\ 2\left[(\mathbf{q}_{wI_k})^{-1}\otimes\mathbf{q}_{wI_{k+1}}\otimes(\boldsymbol{\theta}_{I_kI_{k+1}}^I)^{-1}\right]_{xyz} \\ \mathbf{b}_{ak+1} - \mathbf{b}_{ak} \\ \mathbf{b}_{gk+1} - \mathbf{b}_{gk} \end{bmatrix} \end{aligned} \tag{11}$$

Where $[\bullet]_{xyz}$ represents the vector part of the quaternion. $\begin{bmatrix} \boldsymbol{\alpha}_{I_kI_{k+1}}^I & \boldsymbol{\beta}_{I_kI_{k+1}}^I & \boldsymbol{\theta}_{I_kI_{k+1}}^I \end{bmatrix}^{\mathrm{T}}$ is the pre-integration value of the IMU, which represents the changes in position, velocity, and attitude caused by linear acceleration and angular velocity, respectively. At the same time, the pre-integration process will also propagate the covariance.

3) Encoder Factor:

According to the incremental model of the wheel odometer, we conduct pre-integration on the increments between motion frames to construct the encoder constraint factor. For two consecutive motion frames, we define the pre-integration residual of the encoder as follows:

$$\begin{aligned} \mathbf{r}_O(\mathbf{z}_{I_kI_{k+1}}^O,\boldsymbol{\chi}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}_{I_kI_{k+1}}^O & \delta\boldsymbol{\beta}_{I_kI_{k+1}}^O \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \mathbf{R}_{I_kw}(\mathbf{p}_{wI_{k+1}} - \mathbf{p}_{wI_k}) - \boldsymbol{\alpha}_{I_kI_{k+1}}^O \\ \mathbf{R}_{I_kw}(\mathbf{v}_{wI_{k+1}} - \mathbf{v}_{wI_k}) - \boldsymbol{\beta}_{I_kI_{k+1}}^O \end{bmatrix} \end{aligned} \tag{12}$$

Where is $\begin{bmatrix} \boldsymbol{\alpha}_{I_kI_{k+1}}^O & \boldsymbol{\beta}_{I_kI_{k+1}}^O \end{bmatrix}^{\mathrm{T}}$ the pre-integration value of position and velocity of the encoder, which have been transformed to the IMU coordinate through extrinsic parameter $\mathbf{T}_{OI}$.

One problem that deserves attention is the propagation of the encoder's pre-integration error, which is mainly composed of the error at the previous moment, the current noise measurement, and the rotation error. We set the position and velocity noise of the encoder increment as $\boldsymbol{\eta}_d$ and $\boldsymbol{\eta}_v$, and its iterative propagation form is:

$$\begin{bmatrix} \delta\mathbf{p}_{k+1}^i \\ \delta\mathbf{v}_{k+1}^i \end{bmatrix} = \begin{bmatrix} \delta\mathbf{p}_k^i + \mathbf{R}_{I_k}^{I_i}\boldsymbol{\eta}_d + \mathbf{R}_{I_k}^{I_i}\delta\mathbf{R}_{I_k}^{I_i}\mathbf{p}_k^i \\ \delta\mathbf{v}_k^i + \mathbf{R}_{I_k}^{I_i}\boldsymbol{\eta}_v + \mathbf{R}_{I_k}^{I_i}\delta\mathbf{R}_{I_k}^{I_i}\mathbf{v}_k^i \end{bmatrix} \tag{13}$$

Where $\mathbf{R}_{I_k}^{I_i}$ is the IMU pre-integration rotation component and $\delta\mathbf{R}_{I_k}^{I_i}$ is the IMU pre-integration rotation error.

C. Favor Factor Calculation and Graph Optimization

When multiple sensor measurements are combined for estimation, there may be abnormal data due to robot wheel slip, vision loss, etc. If the wrong data is added to the optimization, it will bring errors or even divergence. Therefore, we propose a multi-source measurement evaluation algorithm that can detect abnormal data and adjust its optimization weight.

We use the measurements of IMU and encoder to calculate the position increment between the motion frame $\mathbf{p}_{I_kI_{k+1}}^I$

and $\mathbf{p}^{O}_{I_kI_{k+1}}$ respectively, and at the same time, PNP algorithm is used to calculate the position increment $\mathbf{p}^{C}_{I_kI_{k+1}}$ through the visual matching information. Compared to wheel encoders, IMUs are self-closing sensors that are less susceptible to external interference, and the short-term estimation results are more accurate. Therefore, the IMU increment value is used as a reference to calculate the favor factor:

$$\begin{cases} f_{pk} = \left(\dfrac{\left\|\mathbf{p}^{I}_{I_kI_{k+1}}\right\|}{\left\|\mathbf{p}^{O}_{I_kI_{k+1}}\right\|}\right)^2, \ 0 \le f_{pk} \le 1 \\ f_{rk} = \dfrac{\left\|\boldsymbol{\theta}^{I}_{I_kI_{k+1}}\right\|}{T_\theta}, \ 0 \le f_{rk} \le 0.5 \\ f_{ck} = \left(\dfrac{\left\|\mathbf{p}^{C}_{I_kI_{k+1}}\right\| - \left\|\mathbf{p}^{I}_{I_kI_{k+1}}\right\|}{\left\|\mathbf{p}^{I}_{I_kI_{k+1}}\right\|}\right)^2, \ 0 \le f_{ck} \le 1 \\ F^{O}_{k} = f_{pk}(1-f_{rk}), \ 0 \le F^{O}_{k} \le 1 \\ F^{I}_{k} = f_{rk}(1-f_{pk}), \ 0.5 \le F^{I}_{k} \le 1 \\ F^{C}_{k} = 1 - f_{ck}, \ 0 \le F^{C}_{k} \le 1 \end{cases} \tag{14}$$

Where F^{O}_{k} is the favor factor of the encoder, F^{I}_{k} is the favor factor of IMU, and F^{C}_{k} is the visual favor factor. $\boldsymbol{\theta}^{I}_{I_kI_{k+1}}$ is the pose increment between motion frames, with the unit being degree. T_θ is the threshold of the normalized angle, this paper takes $T_\theta = 2.5$.

For the calculation of the favor factor, the following need to be explained. First, in order to make the encoder more sensitive to abnormalities, the position factor f_{pk} is squared. Secondly, after a lot of tests, because the robot needs sliding steering, and the encoder's position accuracy is higher when the robot is moving in a straight line than turning, we introduce an angle factor to adjust the encoder weight. In addition, it should be noted that the weight of the IMU cannot be zero, because the IMU contains the constraint of gravity. If its weight is zero, the observability of the system will change.

Finally, we use the beam adjustment to minimize the residuals of all observations and obtain the maximum posterior estimate of the robot state:

$$\boldsymbol{\chi}^* = \arg\min \Big\{ (\mathbf{H}_p\delta\boldsymbol{\chi} - \mathbf{b}_p) + \sum_{k\in I}\left\|F^{I}_{k}\mathbf{r}_I(\mathbf{z}^{I}_{I_kI_{k+1}},\boldsymbol{\chi})\right\|^2_{\boldsymbol{\Sigma}^{I_k}_{I_{k+1}}} + \\ \sum_{k\in O}\left\|F^{O}_{k}\mathbf{r}_O(\mathbf{z}^{O}_{I_kI_{k+1}},\boldsymbol{\chi})\right\|^2_{\boldsymbol{\Sigma}^{O_k}_{O_{k+1}}} + \sum_{(l,j)\in C}\rho\left(\left\|F^{C}_{k}\mathbf{r}_C(\mathbf{z}^{C_j}_{l},\boldsymbol{\chi})\right\|^2_{\boldsymbol{\Sigma}^{C_j}_{l}}\right)\Big\} \tag{15}$$

Where $(\mathbf{H}_p, \mathbf{b}_p)$ is the prior information obtained by marginalizing the old variable. We define the marginalized variable as $\boldsymbol{\chi}_m$ and the remaining variables as $\boldsymbol{\chi}_r$. By reordering the variables, we can get:

$$\begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\chi}_m \\ \delta\boldsymbol{\chi}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix} \tag{16}$$

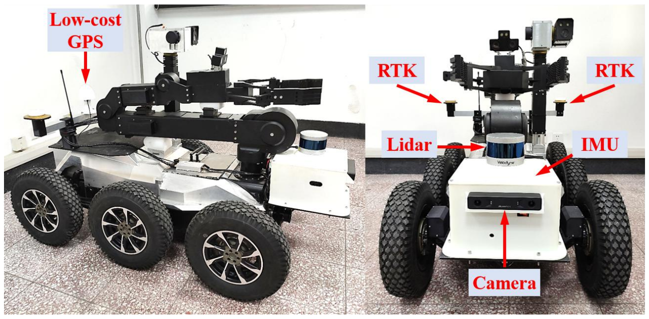


Figure 5. Experimental setup using a six-wheeled ground robot. The sensing box are installed in the front of the robot.

Then we use Shure complement to calculate the prior information:

$$\underbrace{(\mathbf{H}_{rr} - \mathbf{H}_{rm}\mathbf{H}^{-1}_{mm}\mathbf{H}_{mr})}_{\mathbf{H}_p}\delta\boldsymbol{\chi}_r = \underbrace{\mathbf{b}_r - \mathbf{H}_{rm}\mathbf{H}^{-1}_{mm}\mathbf{b}_m}_{\mathbf{b}_p} \tag{17}$$

D. Loop-Closing and Map Reuse

The method described above implements an open-loop localization algorithm and cumulative drift will occur over a long period of operation. Therefore, we have added online loop detection and optimization to eliminate cumulative drift. At the same time, for the long-term application of the robot in a fixed area, in order to prevent the infinite increase in the size of the database, we add a mechanism of redundant key frame detection and removal when maintaining the database. With each key frame as the center, a local geometric window is established, and feature matching is performed with each key frame in the window, and the relative pose is calculated. If the relative pose is less than a certain threshold and the number of matching features is greater than a certain threshold, then calculated is the relative poses of two key frames in the front and back of the key frame, and the key frames with large differences in relative pose will be removed. This strategy guarantees the even distribution of key frames, and ensures that the size of key frames is always within a certain range. In addition, added is the function of saving and loading the pose map in the loop-closing thread. Combined with loop optimization part, the function can help realize drift-free localization over long period when robots navigate in the same area, which is very useful in actual robot products.

IV. Experiments

To the best of the authors' knowledge, there is no public data set of original cameras, IMUs, and encoders for ground robots on the large-scale complex terrain. Therefore, we collected 10 datasets in different environments based on the ground rescue robot, including indoor flat environment, outdoor roads, slopes, stairs and other elements.

A. Evaluation Setup

We conducted experiments using a six-wheeled ground robot shown in Fig. 5. The motor of each wheel is equipped with a J52XU9735 encoder with a resolution of 4096 levels. We built a sensing box integrating sensing and computing and comprising a VLP-16 LiDAR, a D1000-IR-120 stereo camera, a low-cost IMU built into the camera, a NUC8I7HVK onboard computer, a wireless communication system, and a power supply system. The onboard computer communicates with the

TABLE I. RMSE OF POSITION AND ROTATION IN 2D ENVIRONMENTS

	GR-SLAM	VINS-Fusion	Se2lam
Dataset-01			
pos. err. (m)	**0.322**	0.515	0.355
rot. err. (deg.)	0.602	**0.598**	0.737
Dataset-02			
pos. err. (m)	**0.283**	0.453	0.294
rot. err. (deg.)	0.451	1.124	**0.349**
Dataset-03			
pos. err. (m)	0.486	0.778	**0.451**
rot. err. (deg.)	**1.204**	1.942	1.487
Dataset-04			
pos. err. (m)	**0.519**	0.595	0.625
rot. err. (deg.)	1.423	**1.248**	1.933

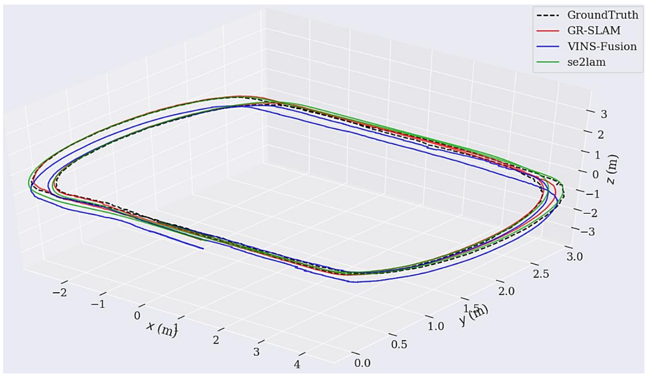


Figure 6. 2D environment evaluation: estimated trajectories on Dataset-01 by VINS-Fusion, se2lam, our method, as well as the ground truth. The Dataset-01 trajectory length is 51m.

robot controller through a serial port to acquire the robot wheel angle data and control data from the robot controller and send velocity commands to the robot controller at a frequency of 200 Hz.

In order to evaluate the performance of GR-SLAM, we use the closed-loop result of LeGO-LOAM as the ground truth in indoor environments. The robot is equipped with high-precision real-time kinematics (RTK) to collect the ground truth of the robot in outdoor environments. It should be noted that in the following experiments, we configured GR-SLAM in the open-loop mode.

B. Localization Performance in 2D Environments

First, we test GR-SLAM in 2D environments, and compare it with VINS-Fusion [21] and se2lam [15] algorithms, and VINS-Fusion employs a binocular and IMU configuration. Table Ⅰ shows the localization error on four 2D datasets. The trajectory on Dataset-01 is shown in Fig. 6. From Table Ⅰ, it can be seen that in the 2D environments, the localization error of GR-SLAM is basically equivalent to se2lam, and is smaller than VINS-Fusion.

Se2lam and GR-SLAM can both use pre-integration of encoder data to predict the robot state and tightly couple camera and encoder data for state optimization, and hence their accuracy is higher than VINS-Fusion. But se2lam is based on plane assumptions and is only applicable to scenes with flat ground such as factories. In actual tests, VINS-Fusion generally have a large drift on the z-axis. Se2lam performs pose estimation in SE(2), which is consistent with the 2D environment, and its drift on the z-axis is the smallest.

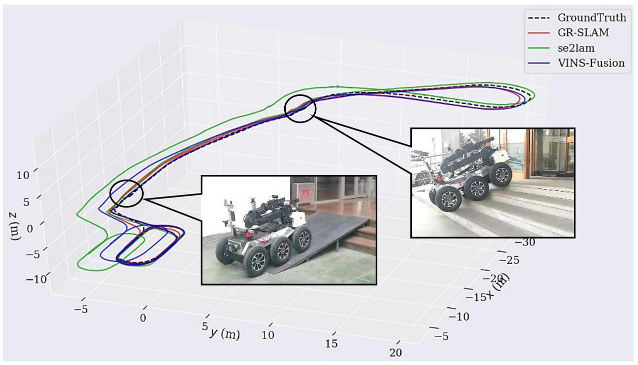
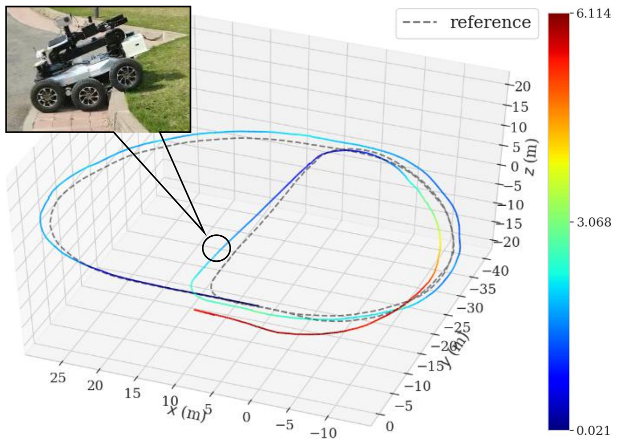
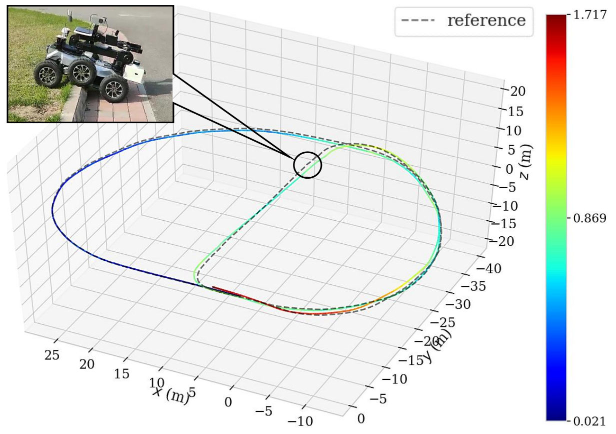


Figure 7. 3D environment evaluation: estimated trajectories on Dataset-08 by VINS-Fusion, se2lam, our method, as well as the ground truth. The trajectory length of Dataset-08 is 206m and contains elements such as slopes, stairs and uneven roads.



(a) VINS-Fusion



(b) GR-SLAM

Figure 8. 3D environment evaluation of VINS-Fusion and GR-SLAM on Dataset-10. The trajectory length of Dataset-10 is 268m and contains elements such as uneven roads, steps and lawn.

C. Localization Performance in 3D Environments

Then we perform experiments in 3D environments. The experimental scene contains slopes, stairs, uneven roads and lawn. We test on six representative datasets, and the results are shown in Fig. 7, Fig. 8 and Table Ⅱ. Among them, the Dataset-09 and Dataset-10 are collected in outdoor scene and the ground truth is collected through RTK, only with position

TABLE II. RMSE OF POSITION AND ROTATION IN 3D ENVIRONMENTS

	GR-SLAM	VINS-Fusion	Se2lam
Dataset-05			
pos. err. (m)	**0.627**	1.363	1.944
rot. err. (deg.)	**1.132**	2.540	1.357
Dataset-06			
pos. err. (m)	**2.094**	3.269	4.811
rot. err. (deg.)	**3.725**	4.351	5.228
Dataset-07			
pos. err. (m)	**0.273**	0.312	—
rot. err. (deg.)	0.683	**0.573**	—
Dataset-08			
pos. err. (m)	**1.661**	3.584	3.743
rot. err. (deg.)	**2.219**	2.865	3.259
Dataset-09			
pos. err. (m)	**2.147**	6.035	4.526
rot. err. (deg.)	—	—	—
Dataset-10			
pos. err. (m)	**0.928**	3.146	—
rot. err. (deg.)	—	—	—

TABLE III. MEAN COMPUTATION TIME OF ALGORITHMS EXECUTED.

Algorithm	**Computation time (ms)**		
	Prediction	**Front-End**	**Back-End**
VINS-Fusion	0.1	21.7	51.4
GR-SLAM	0.1	26.3	65.3

but no rotation. From this results, we can see that in the 3D environments, the localization error of GR-SLAM is significantly smaller than VINS-Fusion. We also find that the localization accuracy of VINS-Fusion in outdoor is lower than indoor environments.

When the robot moves along a straight line at a constant speed, the acceleration and angular velocity of the robot both tend to zero. At this time, the IMU cannot provide good constraints for the robot state. On the contrary, the signal-to-noise ratio of the IMU measurements is relatively low, and its noise will bring errors to the state estimation. Moreover, in outdoor environments, the vibration of the robot is more severe than indoor environments. The proposed GR-SLAM fully leverages camera, IMU, encoder data, and thoroughly processes multi-sensor measurements and tightly couples camera, IMU, encoder data for state optimization. All these characteristics render GR-SLAM more accurate than the other evaluated algorithms.

The mean computation time of the algorithms is listed in Table Ⅲ. Due to the additional calculations, GR-SLAM consumes more time than VINS-Fusion, but it can run in real time on the onboard computer and satisfy the robot navigation.

In addition, we test the robot wheel slipping and visual loss respectively to verify the robustness of the algorithm when abnormal data appear. During the operation of the robot, we dragged the robot to cause its wheels to slip, and then released it to continue normal operation. The experimental results are shown in Fig. 9. If there is no favor factor in the optimization, the pose of the robot will jump when the wheel is slipping, resulting in a large error. In contrast, with the adding of the favor factor, the abnormal data can be quickly isolated from optimization to ensure the accuracy of the estimate.

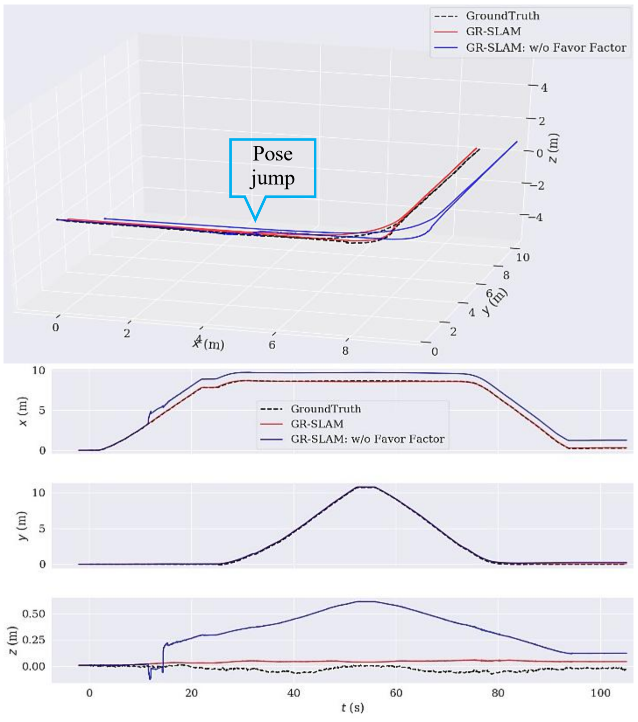


Figure 9. Evaluation shen robot wheels are slipping. Top: Estimated trajectories. Bottom: Estimated position curve of axes x, y and z.

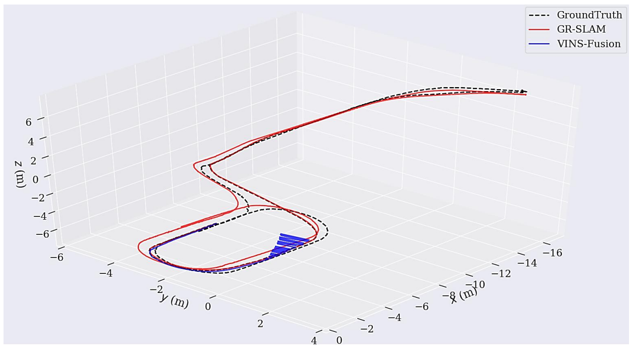


Figure 10. Evaluation when visual information is lost.

Finally, we block the camera for 30s and make vision lose for a long time to verify the continuity of the algorithm in state estimation. The experimental results are shown in Fig. 10. When the vision is lost, the results of VINS-Fusion diverge quickly, while GR-SLAM continues to perform state estimation and guarantee a certain accuracy. When the vision is restored, the visual information will be automatically added to the optimization, and no pose jump occurs during the entire process, and the continuity of the state estimation for ground robot is guaranteed.

V. CONCLUSION

We propose GR-SLAM for ground robots on complex terrain. This framework can estimate the robot state by fusing camera, IMU, and encoder measurements in a tightly coupled scheme. By adequate processing and use of multi-sensor measurements, GR-SLAM achieves high performance even in challenging environments. The odometer increment model that fuses IMU and encoder measurements can predict the robot state and provide robust constraints for estimation even under low-quality camera measurements. To detect abnormal data during fusion, we propose a multi-sensor measurement evaluation algorithm that can adjust optimization weights. Finally, we implement a complete factor graph optimization framework and perform extensive experiments based on real ground robot in 2D and 3D environments. The results show that GR-SLAM can accurately and robustly provide pose estimation for robots even if wheels slipping and vision is lost.

REFERENCES

[1] Ji Zhang and Sanjiv Singh, "Low-drift and real-time lidar odometry and mapping," Autonomous Robots, 41(2):401–416, 2017.

[2] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 2174–2181.

[3] Shan, Tixiao, and Brendan Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," IEEE International Conference on Intelligent Robots and Systems (IROS), 2018.

[4] R. Mur-Artal and J. D. Tard´os, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," IEEE Trans. Robot., 2017.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," IEEE Trans. Pattern Anal. Mach. Intell., 2017.

[6] C.Forster,M.Pizzoli,andD.Scaramuzza,"SVO:Fastsemi-directmonocul arvisualodometry,"inProc.IEEEInt.Conf.Robot.Autom.,HongKong, May 2014, pp. 15–22.

[7] J. Engel, T. Sch¨ops, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in Proc. Eur. Conf. Comput. Vis., 2014, pp. 834–849.

[8] I. Cviši´c, J. ´Cesi´c, I. Markovi´c, and I. Petrovi´c, "Soft-slam: Computationally efficient stereo visual slam for autonomous uavs," Journal of field robotics, 2017.

[9] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odometry. The International Journal of Robotics Research, 32(6):690–711, 2013.

[10] S. Leutenegger et al., "Keyframe-based visual–inertial odometry using nonlinear optimization," Int. J. Robot. Res., vol. 34, no. 3, pp. 314– 334, 2015.

[11] R. Mur-Artal and J. D. Tard´os, "Visual-inertial monocular slam with map reuse," IEEE Robot. Autom. Lett., vol. 2, no. 2, pp. 796–803, 2017.

[12] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," IEEE Trans. Robot., vol. 34, no. 4, pp. 1004–1020, Aug 2018.

[13] K. J. Wu, C. X. Guo, G. Georgiou, and S. I. Roumeliotis, "Vins on wheels," in Proc. IEEE Int. Conf. Robot. Autom, May 2017, pp. 5155–5162.

[14] Zhang, Mingming, Yiming Chen, and Mingyang Li, "Vision-aided localization for ground robots," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[15] Zheng, Fan, and Yun-Hui Liu, "Visual-Odometric Localization and Mapping for Ground Vehicles Using SE(2)-XYZ Constraints," International Conference on Robotics and Automation (ICRA), IEEE, 2019.

[16] M.Quan, S.Piao,M.Tan, andS.-S.Huang,"Tightly-coupled Monocular Visual-odometric SLAM using Wheels and a MEMS Gyroscope," ArXiv e-prints, Apr. 2018.

[17] L. Li et al., "Estimating position of mobile robots from omnidirectional vision using an adaptive algorithm," IEEE Trans. Cybern., vol. 45, no. 8, pp. 1633–1646, 2015.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," IEEE Trans. Robot., vol. 33, no. 1, pp. 1–21, 2017.

[19] P. Furgale, H. Sommer, J. Maye and others, "kalibr," https://github.com/ethz-asl/kalibr.

[20] Censi, Andrea, et al, "Simultaneous calibration of odometry and sensor parameters for mobile robots," IEEE Transactions on Robotics 29.2 (2013): 475-492.

[21] Qin T, Pan J, Cao S, et al, "A general optimization-based framework for local odometry estimation with multiple sensors," arXiv preprint arXiv:1901.03638 (2019).

[22] He Y, Guo Y, Ye A, et al, "Camera-odometer calibration and fusion using graph based optimization," 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, 2017.