Acknowledge

The work is conducted within the corporate MSc program supported by Sber. I would like to Aleksei Burkov for hosting us at Sber Robotics Lab during the time we came there to conduct our research and his brilliant comments, suggestions.

I would like to acknowledge and give my warmest thanks to my supervisor Prof. Sergey Kolyubin who made this work possible, his supporting and guidance through all my MSc years, thanks to him I am a better students. I would like to send my best regard to my friend Jaafar Mahmoud, who is my best friend and my first mentor when I started to step into research field, undoubtedly without him I can not finish this research successfully.

My student life will be more effective but way less fun without my colleague at Bio-mechatronics and Energy-Efficient Robotics Lab, thanks to all you guys.

Last but not least, I would like to thanks my parents and my family for their continuous, unconditionally supporting.

Contents

List of Figures

List of Tables

Acronyms and Abbreviation

BA Bundle Adjustment 9, 11–13

CPU Central Processing Unit 9, 14

DNN Deep Neural Network 16
DOF Degree of Freedom 8, 12, 14

EKF Extended Kalman Filter 13

IMU Inertial Measurement Unit 8, 9, 17, 18, 20, 21

KLT Kanade Lucas Tomasi 13

MSCKF Multi-State Constraint Kalman Filter 13

OKVIS Open Keyframe-based Visual-Inertial SLAM 14

RGB-D Red Green Blue and Depth 12
ROVIO Robust Visual Inertial Odometry 13

SLAM Simultaneous Localization and Mapping 8–15, 17, 48

TBB Thread Building Block 14

VINS Visual Inertial Navigation System 8, 13, 14
VIO Visual Inertial Odometry 8, 13

1 Introduction

1.1 Motivation

VIO (Visual Inertial Odometry) algorithms that fuse the optical information from the images of a camera with the accelerometer and gyroscope readings from an IMU (Inertial Measurement Unit) to achieve accurate state estimates of the agents. A camera and an IMU are especially interesting sensors in combination as they have complementary features.

An IMU provides high-frequency motion information, which is accurate at short intervals of time but drifts over time. Instead, a camera provides images at smaller frequencies but allows for accurate estimations of the motion long term. In the last decade, multiple real-time and highly- accurate VIO algorithms have been developed such as. [13], [9]. Unfortunately, these systems have nowadays two limitations that we want to address in this thesis, which we detail below.

1.1.1 Improve Unobservability of Inertial

As proved in [54], VINS (Visual Inertial Navigation System) has two more unobservable DOF (Degree of Freedom) besides 3 DOFs translation and the rotation around the gravity direction. These two new unobservable directions are scale direction and all 3 DOF of orientation if and only if VINS is under restricted movements i.e. constant acceleration and no rotation. These types of movements are typical on the ground vehicle, which leads to a significant reduction regarding accuracy in localization systems using IMU in this type of robot. To this limitation, we want to introduce the new approach of initialization and fusing scheme in which we can resolve the unobservability of IMU in VINS of the ground vehicles in general and mobile robots in particular.

1.1.2 Enhance in Dynamic Environment

Most of the Visual SLAM (Simultaneous Localization and Mapping) is based on the static world assumption where all objects are static. However, the robot has faced with high intensity dynamic objects around it;

consequently the static world assumption is no longer held. The dynamic environment affects badly to the accuracy of the SLAM. To overcome this limitation, we introduced a new thread in our system that detects moving objects by semantics segmentation and removes them from the map.

1.2 Approach

In the aspect to address these limitations, we introduce the new multi--sensor modular SLAM system that cooperates tightly-coupled three typical sensors existing in the mobile robot which are the camera, IMU and wheel odometry to achieve accuracy robot state estimation using the restricted resource CPU (Central Processing Unit).

To achieve real-time performance, the proposed system includes the front-end and back-end parts. The front-end part is used to handle the coming sensors data and predict the current state of the robot. We first extract the features in the new receiving frame, match, and triangulate them with features extracted from the last frame. We then perform the factor-graph optimization along with the preintegration measurement from IMU and wheel odometry. Due to the restriction on time-processing, the optimization in the front-end will only try to refine the robot pose with fixed landmarks position, also known as Motion-only BA (Bundle Adjustment) approach. Additionally, we implemented the well-parallelized scheme in the front-end so that the system will take advantage of multi-core CPU processing and therefore optimize the time-consuming. The front-end also decided to create a new keyframe based on optical principles and time differences with the last keyframe. Back-end will be run in a parallel thread with front-end, as soon as, back-end receives new keyframe, the factor graph is generated base on projection factor and preintegration factors from IMU, wheel odometry measurements. The new keyframe and old keyframe that sharing common view will be refined use built factor graph. More detail about the theoretical and mathematical of the system is described in next chapter.

To improve the initialization of the IMU approach, we adopt the stationary initialized process. Mobile robots commonly start at a static state, we implemented a Zero-Velocity Detector using the mean of matched features disparity between two consecutive frames, if the robot is not moving, from that we can estimate the bias of accelerator and gyroscope. The scale ambiguity can be recovered by the measurement from wheel odometry. Since we can assume that the robot is moving on the flat in the beginning, the system can easily align the gravity direction.

The dynamic environment can be addressed by cooperating with semantic segmentation. For every new keyframe, we will detect the semantic objects in the original image and register it for the corresponding landmark and delete it from the map.

1.3 Thesis Outline

Concerning the structure of the thesis, to get an overview of the current research state, we will first demonstrate a review of the related work in the field of Visual SLAM, the fusing technique with other types of sensors for state estimate. Next, we will present the theoretical and mathematical background of our approach along with the detailed implementation of our SLAM system. Then, we will illustrate the evaluation of our approach in terms of accuracy, processing time and robustness and compare them with the state-of-the-art. The final chapter dedicates to discussion, future improvements and the conclusion of our thesis.

1.4 Literature Review

1.4.1 Visual Odometry/SLAM

In this section, we focus on visual odometry/SLAM works, that can be segmented by its way of process input image from camera: *indirect* or *direct* method. The indirect method is the method that invokes several steps, first extracting features such as points, lines or even objects in the image; then matching or tracking with the corresponding sequential of images, from which we can estimate the camera motion. While the tracking step maintains the local map consistency, loop-closure detection and correction provide the global map consistency. However, the complexity of the optimization problem significantly increases as the number of optimization variables is unbounded. The way to tackle this challenge is *keyframe-based approach* [28], [20]. As proven by Strasdat et al., keeping as many landmarks as possible and continuously removing removes redundancy frames is the most efficient approach for large scale Visual SLAM. It means that the system only builds a map out of some selective frames - keyframes, to avoid redundancy data; moreover, by optimizing keyframes, the optimisation problem is more generalized, and produces a more consistent map as the solution of optimization. From the early era of Visual SLAM, the indirect method with a keyframe approach has shown its advantage over the other method.

PTAM [20] is one of the earliest, by not only using an indirect method but also introducing the parallel threads for camera tracking and local mapping, which contributes to the real-time AR application in the small room. Later, in the improving version [21], the point feature cooperates with the line feature and rotation estimation are implemented and show improvement in term of accuracy. PTAM uses FAST [40] corner detection for feature extraction and matching by patch correlation, however, by this matching method feature can not be used for loop-closure detection, it is stated to the fact that PTAM does not support loop-closure. Strasdat et al. proposes large scale monocular SLAM where front-end based on optical flow with feature matching, back-end performs sliding-window BA, the main contributes of this work is the loop-closure is solved by pose graph optimization with similarity factor - 7 DOF, allows to recover the scale factor and reduce drift significantly when perform the loop correction. This contribution is adopted by most of later work and appears in very *state-of-the-art* system. Mur-Artal et al. presents the most complete and reliable monocular SLAM systems - ORB-SLAM that shows exceptional result. ORB-SLAM [32] incorporates three threads that run parallel: tracking, local mapping, and loop closing. The system chooses ORB [41] because of its efficient extraction time and high quality of place recognition. The tracking thread is in charge of localizing the camera with every frame and deciding when to insert a new keyframe. ORB-SLAM initializes system by *8-point* algorithm [18], if bootstrap successful, this thread performs feature matching with the previous frame and optimizes the pose using motion-only BA. If the tracking is lost (e.g., due to occlusions or abrupt movements), *bag-of-words* [15] library is used to perform relocalization in the global map. Local mapping processes new keyframes and performs local BA to achieve an optimal reconstruction in the camera pose's surroundings. Difference from the previous works, ORB-SLAM build the local BA based on neighbour covisibility of new keyframe rather than sliding-window keyframes, this helps local BA can maximize the optimizing candidates on every BA. Local mapping creates new map points from unmatched features that later based on new information from the tracking thread those map point can be culled to maintain high-quality measurements, it is also in charge of culling redundant keyframes. The loop closing searches for loops with every new keyframe. If a loop is detected, we compute a similarity transformation that informs about the drift accumulated in the loop. Then the loop will be corrected and duplicated landmarks are fused. In the improvement version namely ORB-SLAM2 [30] supports stereo and RGB-D camera with more robustness and suitable for large scale application.

Beside indirect method, visual odometry/SLAM can also base on direct method which monitors the changes of pixel intensity in the image, so that it skips the extract feature step. It leads to the advantage of direct method is very fast, can over the real-time threshold (30 Fps). However, also because of rely on pixel intensity, direct method can be sensitive with illumination changing and due to no features using, it needs a new approach for place recognition and extra work on camera calibration. The noticeable direct method is DSO [11] which performs photometric BA to optimize camera intrinsic and inverse depths of sparse landmarks in a sliding window.

In the middle ways of the direct and indirect method is the semi-direct method, which collaborates advantages of both in the system and most know by project SVO [12]. It performs direct sparse image alignment to estimate the camera pose and only extracts features correspondences to the initial guess. Back-end performs geometric BA instead of photometric to refine the camera pose and map.

1.4.2 Visual-Inertial Odometry/SLAM

In this section, we discuss about VIO/SLAM that categorizes by the type of back-end: *filtering-based* or *optimization-based* method. Filtering-based method is one of the earliest method for state estimation, initial work is base on EKF (Extended Kalman Filter) [45]. To avoid the the quadratic complexity of EKF method, the MSCKF (Multi-State Constraint Kalman Filter) is introduced by Mourikis and Roumeliotis, in which avoids to include landmark measurement into state, consequently, it contributes to the significantly decrease number of state variable. ROVIO (Robust Visual Inertial Odometry) [4], [5] open-source framework is VINS framework with front-end direct method and back-end implements MSCKF, have achieved the high accuracy and robust in localization result. The excellent open-source OpenVINS [16] code-base is one most complete work of VINS filter-based MSCKF method with front-end uses KLT (Kanade Lucas Tomasi) feature tracker [27], [50] or descriptor-based matching, in addition, OpenVINS supports many usefull and reusable features that help leveraging performance in real-world application: static/dynamic initialization, camera-imu intrinsic/extrinsic calibration, zero-velocity detection/update, etc. The main contribution to research community of OpenVINS is comprehensive documentation[1], derivation and highly reusable code-base[2].

[1]`https://docs.openvins.com/pages.html`

[2]`https://github.com/rpng/open_vins`

Optimization-based method formulates the problem in the form of a non-linear energy function, which is continuously optimized in the back--end. To facilitate this, it usually go with keyframe selection scheme that constraint the complexity of optimization problem. The first tightly-couple visual and inertial odometry non-linear optimization method is OKVIS (Open Keyframe-based Visual-Inertial SLAM) [23], [24], [25]. Inspired by OKVIS, VINS-Mono [33], [34] is develop a robust and versatile monocular VINS uses an optimization-based sliding window fashion for perform high--accuracy visual-inertial odometry. It features efficient IMU pre-integration by Forster et al. with bias correction, automatic estimator initialization, online extrinsic calibration, failure detection and recovery, loop detection using bag-of-words [15], and global 4 DOF pose graph optimization, map merge, pose graph reuse, online temporal calibration, rolling shutter support. The extend work is VINS-Fusion [35], [36] supporting stereo camera, achieves the state-of-the-art accuracy and becomes standard for evaluating VINS framework. Basalt [51] is stereo-inertial framework, exploits the non-linear factor from visual-inertial odometry and adds into factor graph optimization. For marginalization, instead of using Schur complement, Usenko et al. implemented the square root marginalization [10] assured numerical stability and faster than baseline, features with multi-core CPU running uses TBB (Thread Building Block) library [38], Basalt reaches up-to 100 Fps, as knowledge of us it is one of the fastest open-source feature-based SLAM. Kimera [39] is an excellent open-source metric-semantic SLAM framework, in which Rosinol et al. focuses on building a semantic map that allows robot understands and interprets human-level command in the task of navigation while rendering the similar localization accuracy to VINS-Fusion. Both Basalt and Kimera base on fairy naive inertial initialization strategy; they assume robot starts at stationary state and perform static initialization, along with gravity alignment. In constrast, ORB-SLAM3 [9] implements the novel, fast optimization-based inertial initialization proposing by Campos et al. [8] that tightly couple initializes bias and gravity direction with visual odometry, as result it obtain 5% error after only 2 seconds and up-to only 1% after 15 seconds. Beside the novel inertial initialization, inheriting front-end from previous work [30], [32] and reuse map from [31], ORB-SLAM3 presents one of the most complete Visual-Inertial SLAM framework that is able to perform full-term data associate, including: short-term (map points matching with new receiving frame), mid-term (map points matching and optimizing in covisibility graph) and long-term (relocalization, loop-closure and map merging). These features all contribute to the current state-of-the-art Visual-Inertial SLAM, our work relies on ORB-SLAM3 due to its comprehensive. The most recent noticeable work is DM-VIO [52] is a monocular inertial SLAM with direct-method front-end from [11], von Stumberg and Cremers proposed delay-marginalization strategy allows to continuously optimize the scale and gravity direction even after IMU initialization, DM-VIO yields scale error smaller than 1% in EuRoC [7], TUM-VI [43] datasets, moreover, DM-VIO even out-performs ORB-SLAM3 and Basalt in 4season dataset [53], by used only monocular inertial sensors and without loop-closure.

1.4.3 Visual SLAM in Dynamic Environment

The main idea for tackling dynamic environments in visual SLAM is using semantic segmentation or object detectors to label dynamic objects like cars, people, bikes, etc. We then remove the extracted points that belong to the region labeled. Research related to the problem is significantly rising in recent years, there are vast semantic segmentation models that can be implemented in visual SLAM systems. For instance, Mask R-CNN [19] is used in [42], [2], [57]. Although the high-weighted models like Mask R-CNN gave excellent results in labelling and removing dynamic objects, they have a high impact on the system processing time; consequently, visual SLAM can not achieve the real-time operation. On the other hand, SegNet [1] is a lightweight convolutional network for image semantic segmentation and achieves processing of 33 (ms) per frame in our experiments so that it can guarantee real-time operation application. DS-SLAM [56] uses semantic segmentation model (SegNet [1]), along with moving consistency check and is based on ORB-SLAM2 [30]. Dre-SLAM [55] employs an object detection model (YOLO [37]) along with the K-means clustering method for segmentation over the depth data from the RGB-D sensor. Detect-SLAM [59] uses a

DNN-based object detector, along with propagating probabilities of features (probability of detecting this feature on a moving object).

2 Problem Statement

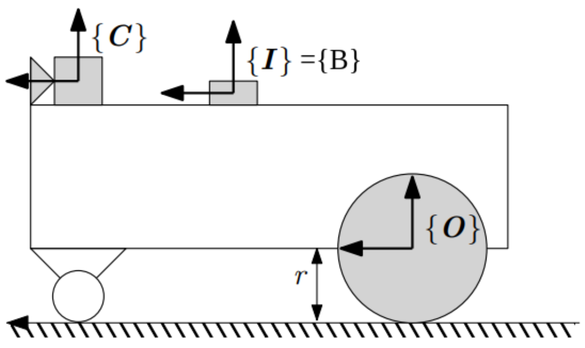
2.1 Robot State Vector

In this thesis, we develop the SLAM system for the mobile robot which contains three common sensors: optical, inertial and wheel odometry, so we formulate our problem as the estimating the state of an agent (e.g a mobile robot), equipped with an IMU (frame “I”), a monocular camera (frame “C”) and a wheel odometer (frame “O”).

An IMU produces smooth measurements with noise but little outliers which can make its estimate very accurate as long as the bias drift can be well-constrained by other sensors. Therefore, our system is designed around the IMU as the primary sensor; it means that we consider frame “I” to coincide with the body frame “B” of the agent, which is the desired state frame that we want to estimate, illustrated in Fig. 2.1. These three frame is fixed in the robot body, all the transformations between these trio-sensors are assumed to be constant and known from prior calibration and are denoted as:

— $\mathbf{T}_{\mathrm{BC}} = [\mathrm{R}_{\mathrm{BC}}|\mathbf{p}_{\mathrm{BC}}]$ - The transformation from camera “C” to body frame “B”.

— $\mathbf{T}_{\mathrm{BO}} = [\mathrm{R}_{\mathrm{BO}}|\mathbf{p}_{\mathrm{BO}}]$ - The transformation from wheel odometer “O” to body frame “B”.



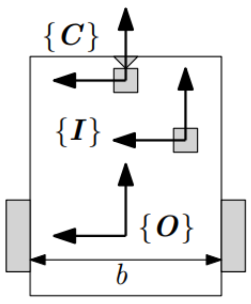


Figure 2.1 — The frame system uses in the thesis

These transformation is estimated by SE(3) Umeyama alignment method with given set of trajectory that estimated from each sensor individually, more detail and work-through calibration will be described in later sections. The ultimate goal of Visual-Inertial-Wheel State Estimation is to estimate the state of body frame w.r.t to world frame "W" - $\mathbf{T}_{\mathrm{WB}} = [\mathrm{R}_{\mathrm{WB}}|\mathbf{p}_{\mathrm{WB}}]$ from the sensors measurement. At the time i, the state of the robot $\mathbf{x}_i$ is described in the IMU frame

$$\mathbf{x}_i = [\mathrm{R}_i, \mathbf{p}_i] \tag{2.1}$$

Where:

— Pose $[\mathrm{R}_i|\mathbf{p}_i]$ is presented as SE(3) in which $\mathrm{R}_i \in \mathrm{SO}(3)$ is the rotation matrix and $\mathbf{p}_i \in \mathbb{R}^3$ is linear translation at time i.

2.2 Sensor Model

We denote the IMU measurement as $\mathcal{I}_{ij}$. The IMU measurement $\mathcal{I}_{ij}$ is the set of measurement between keyframe $\mathcal{K}_i$ and $\mathcal{K}_j$ and the measurement between two consecutive i; $i+1$ determined by the preintegrated on-manifold technique in [13]. The wheel odometer provides the absolute robot's pose at time i and is denoted by $\mathcal{O}_i$, we also perform preintegration to determined wheel odometric measurement out of two consecutive $\mathcal{K}_i$ and $\mathcal{K}_j$.

By using Pose Graph Optimization, we optimize the residual error and update the robot state with given the optical, IMU and wheel odometer measurement.

2.2.1 Optical Sensor

An optical sensor is usually a monocular camera or stereo camera. We assume that camera can be correctly model by pinhole model, the optical sensor should calibrate to correct distortion and estimate its intrinsic parameters before using it for further application. Undistortion lens and intrinsic parameters are computed by optimizing reprojection errors on several images with known calibration patterns. With respect to camera coordinate C, Z-axis is forward, X-axis is horizontal to right side and Y-axis is vertical downward. Each optical frame observes multi-landmarks l; the measurement of the optical sensor formulates as the pixel coordinate of the observed landmarks on 2D image plane coordinate P. The mapping formulation 3D landmark from world coordinate to camera coordinate

$$l_i^c = \mathrm{R}^T l_i - \mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

where $[\mathrm{R}|\mathbf{p}] \in \mathrm{SE}(3)$ is the camera pose in world coordinate. The formulation reprojects 3D landmarks in world coordinates into 2D pixel in camera coordinate, is difference between monocular and stereo camera. Give position of 3D landmark i in world coordinate as $l_i \in \mathbb{R}^3$, we denote its position in camera coordinate as $l_i^c \in \mathbb{R}^3$. Reprojection formulation π_m of monocular camera that mapping 3D map points l_i to 2D pixel coordinates $\mathbf{x} \in \mathbb{R}^2$ on the image plane $\mathbb{R}^3 \mapsto \mathbb{R}^2$:

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi_m\left(l_i^c\right) + \eta_{\mathbf{x}} = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} + \eta_{\mathbf{x}} \tag{2.3}$$

where:

— $[u \quad v]^T$ is horizontal and vertical coordinate in image plane.

— f_x, f_y are camera focal length in X and Y-axis.

— c_x, c_y are coordinate of camera principle point, which are the point on the image plane onto which the perspective center is projected.

— $\eta_{\mathbf{x}} \sim \mathcal{N}\left(\mathbf{0}, \sigma_u^2 \mathbf{I}_2\right)$ is the projection uncertainty (σ_u is the covariance from ORB pyramid level on which the feature is located)

In stereo cases, it consists of two rigidly attached frames left and right. One frame of stereo can estimate depth by finding a correspond between the left and right images. Calibration performs to estimate not only intrinsic parameters but also rectify parameters. Rectification process aligns left, right images so that the epipolar line is horizontal. Once the rectified, the corresponds features in the left image are on the same row in the right image. Moreover, stereo-rectified has left and right camera with the same focal length and principal point. The reprojection formulation π_s of rectified stereo camera $\mathbb{R}^3 \mapsto \mathbb{R}^3$:

$$\mathbf{x} = \begin{bmatrix} u_L \\ v_L \\ u_R \end{bmatrix} = \pi_s\left(l_i^c\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{2.4}$$

where

— $[u_L \quad v_L]^T$ is horizontal and vertical coordinate in left image plane.

— u_R is horizontal coordinate in right image plane, vertical v_R should equal to v_L after rectified.

— b is baseline - horizontal distance between left and right camera.

— f_x, f_y are camera focal length in X and Y-axis after rectified.

— c_x, c_y are coordinate of camera principle point after rectified.

— $\eta_{\mathbf{x}} \sim \mathcal{N}\left(\mathbf{0}, \sigma_u^2 \mathbf{I}_3\right)$ is the projection uncertainty (σ_u is the covariance from ORB pyramid level on which the feature is located)

Henceforth, the symbol stands for the reprojection of 3D landmarks onto the image plane for both monocular and stereo camera cases.

2.2.2 IMU

IMU provides measurement model as linear acceleration $\tilde{a}_B$ and angular velocity $\tilde{\omega}_B$ at high-frequency approximate hundreds of Hezts. However, these measurements are disturbed by the bias of the accelerometer - b_a and gyrocopter - b_g. They are the combination of static and dynamic bias which is produced by temperature variation, vibration, pressure etc. The raw measurement model at time t formulates as:

$$\begin{aligned} \tilde{\boldsymbol{\omega}}_B(t) &= \boldsymbol{\omega}_B(t) + \mathbf{b}^g(t) \\ \tilde{\mathbf{a}}_B(t) &= \mathrm{R}_{\mathrm{WB}}^{\top}(t)\left(\mathbf{a}_B(t) - \mathbf{g}_{\mathrm{w}}\right) + \mathbf{b}^a(t) \end{aligned} \tag{2.5}$$

Where

— $[\mathrm{R}_{\mathrm{WB}} | \mathbf{p}_{\mathrm{WB}}]$ - the transformation from sensor to world coordinates at time i.

— $\boldsymbol{\omega}_B(t)$ is the actual angular velocity measurement.

— $\boldsymbol{a}_B(t)$ is the actual linear acceleration measurement.

— $\mathbf{g}_\mathrm{W}$ is the gravity vector in world coordinates.

The sate of IMU sensor $\mathcal{I}$ at time i is described

$$\mathcal{I}_i = [\mathrm{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i] \tag{2.6}$$

where

— $[\mathrm{R}_i|\mathbf{p}_i] \in \mathrm{SE}(3)$ is the pose of sensor.

— $v_i \in \mathbb{R}^3$ is linear velocity.

— $b \in \mathbb{R}^6$ is the bias for accelerometer and gyrocopter is 3D vector for each.

Integrating formula 2.5, we get the state vector of the IMU in the discrete time domain; more specifically at the $k + 1$ measurement with interval Δt from previous measurement k, the state vector is calculated according to the following formula

$$\begin{aligned}
\mathrm{R}_{\mathrm{WB}}^{k+1} &= \mathrm{R}_{\mathrm{WB}}^{k} \operatorname{Exp}\left(\left(\boldsymbol{\omega}_{\mathrm{B}}^{k} - \boldsymbol{b}_{g}^{k}\right) \Delta t\right) \\
\mathbf{v}_{\mathrm{B}}^{k+1} &= \mathbf{v}_{\mathrm{B}}^{k} + \mathbf{g}_{\mathrm{W}} \Delta t + \mathrm{R}_{\mathrm{WB}}^{k} \left(\boldsymbol{a}_{\mathrm{B}}^{k} - \boldsymbol{b}_{a}^{k}\right) \Delta t \\
\mathrm{p}_{\mathrm{WB}}^{k+1} &= \mathbf{p}_{\mathrm{WB}}^{k} + \mathbf{v}_{\mathrm{B}}^{k} \Delta t + \frac{1}{2} \mathbf{g}_{\mathrm{W}} \Delta t^{2} + \frac{1}{2} \mathrm{R}_{\mathrm{WB}}^{k} \left(\boldsymbol{a}_{\mathrm{B}}^{k} - \boldsymbol{b}_{a}^{k}\right) \Delta t^{2}
\end{aligned} \tag{2.7}$$

2.2.3 Wheel Odometry

The wheel odometry measurement model is simpler than the optical sensor and IMU. The modern wheel odometry on a mobile robot usually provides the odometry directly. In addition, the white noise - $\boldsymbol{\eta}$ introduced in the measurement due to slippage, hardware error etc. The raw angle rotation k measurement $\tilde{\phi}_k \in \mathbb{R}$ and translation $\tilde{\mathbf{p}}_k \in \mathbb{R}^2$

$$\begin{aligned}
\tilde{\phi}_k &= \phi_k + \boldsymbol{\eta}_k^{\phi} \\
\tilde{\mathbf{p}}_k &= \mathbf{p}_k + \boldsymbol{\eta}_k^{\mathbf{p}}
\end{aligned} \tag{2.8}$$

3 Background Theory

3.1 Lie Group Preliminary

In this section, we briefly provide some essential Lie Group and Lie Algebra properties that will useful later on.

Special Orthogonal Group - SO(3) is the group of 3-D rotation with formally defined as

$$\mathrm{SO}(3) = \left\{\mathrm{R} \in \mathbb{R}^{3\times 3}, \mathrm{R}^\top \mathrm{R} = \mathbf{I}, \det(R) = 1\right\} \tag{3.1}$$

The tangent space of to the SO(3) (at the identity) is called as *Lie Algebra*, denoted as $\mathfrak{so}(3)$ and can be found by taking the time derivative of $\mathrm{R}^\top \mathrm{R} = \mathbf{I}$, yield: $\mathrm{R}^\top \dot{\mathrm{R}} = -\left(\mathrm{R}^\top \dot{\mathrm{R}}\right)^\top$. It reveals that $\mathrm{R}^\top \dot{\mathrm{R}}$ is the skew-symmetric matrix, which is defined as:

$$[\boldsymbol{\omega}]_\times = \boldsymbol{\omega}^\wedge = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3) \tag{3.2}$$

At the identity ($\mathrm{R} = \mathbf{I}$), we obtain $\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times$, it proofs that $[\boldsymbol{\omega}]_\times$ is in $\mathfrak{so}(3)$. We map from $\mathfrak{so}(3)$ to $\mathbb{R}^3$ and vice versa by linear operator *hat* and *vee*:

$$\begin{aligned} &\text{Hat: } \mathbb{R}^3 \rightarrow \mathfrak{so}(3); \quad \boldsymbol{\omega} \mapsto \boldsymbol{\omega}^\wedge = [\boldsymbol{\omega}]_\times \\ &\text{Vee: } \mathfrak{so}(3) \rightarrow \mathbb{R}^3 \;; \quad [\boldsymbol{\omega}]_\times \mapsto [\boldsymbol{\omega}]_\times^\vee = \boldsymbol{\omega} \end{aligned}$$

The property of skew-symmetric matrices that can be useful later is

$$\mathbf{a}^\wedge \mathbf{b} = -\mathbf{b}^\wedge \mathbf{a} \quad \forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^3 \tag{3.3}$$

The *exponential map* is $\exp : \mathfrak{so}(3) \mapsto \mathrm{SO}(3)$ converts elements of the Lie Algebra into elements of the group, according to Rodrigues formula:

$$\exp\left(\phi^\wedge\right) = \mathbf{I} + \frac{\sin(\|\phi\|)}{\|\phi\|}\boldsymbol{\phi}^\wedge + \frac{1-\cos(\|\phi\|)}{\|\boldsymbol{\phi}\|^2}\left(\boldsymbol{\phi}^\wedge\right)^2 \tag{3.4}$$

The first-order approximation of exponential map:

$$\exp\left(\phi^\wedge\right) \simeq \mathbf{I} + \phi^\wedge \tag{3.5}$$

The *logarithm map* is $\log : \mathrm{SO}(3) \mapsto \mathfrak{so}(3)$ exactly the invert of the exponential map, convert the elements of the group into Lie Algebra

$$\log(\mathrm{R}) = \frac{\varphi \cdot \left(\mathrm{R} - \mathrm{R}^\top\right)}{2\sin(\varphi)} \text{ with } \varphi = \cos^{-1}\left(\frac{\mathrm{tr}(\mathrm{R}) - 1}{2}\right) \tag{3.6}$$

We adopted the “vectorized" notation version of exp and log in [13] for the sake of simplicity:

$$\begin{aligned} \mathrm{Exp} : \mathbb{R}^3 \quad &\to \mathrm{SO}(3); \quad \boldsymbol{\phi} \mapsto \exp\left(\boldsymbol{\phi}^\wedge\right) \\ \mathrm{Log} : \mathrm{SO}(3) &\to \mathbb{R}^3 \quad ; \quad \mathrm{R} \mapsto \log(\mathrm{R})^\vee \end{aligned}$$

A useful first-order approximation of Exp:

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\,\mathrm{Exp}\left(\mathrm{J}_r(\phi)\delta\phi\right) \tag{3.7}$$

where $\mathrm{J}_r(\phi)$ is the Right Jacobian on Lie Group

$$\mathrm{J}_r(\phi) = \mathbf{I} - \frac{1 - \cos(\|\phi\|)}{\|\phi\|^2}\phi^\wedge + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi^3\|}\left(\phi^\wedge\right)^2 \tag{3.8}$$

A similar approximation is also held for Log

$$\mathrm{Log}(\mathrm{Exp}(\phi)\,\mathrm{Exp}(\delta\phi)) \approx \phi + \mathrm{J}_r^{-1}(\phi)\delta\phi \tag{3.9}$$

where $\mathrm{J}_r^{-1}(\phi)$ is the inverse of Right Jacobian

$$\mathrm{J}_r^{-1}(\phi) = \mathbf{I} + \frac{1}{2}\phi^\wedge + \left(\frac{1}{\|\phi\|^2} + \frac{1 + \cos(\|\phi\|)}{2\|\phi\|\sin(\|\phi\|)}\right)\left(\phi^\wedge\right)^2 \tag{3.10}$$

Another useful property of Exp

$$\begin{aligned} &\mathrm{R}\,\mathrm{Exp}(\phi)\mathrm{R}^\top = \mathrm{Exp}\left(\mathrm{R}\phi^\wedge\mathrm{R}^\top\right) = \mathrm{Exp}(\mathrm{R}\phi) \\ \Leftrightarrow \quad &\mathrm{Exp}(\phi)\mathrm{R} = \mathrm{R}\,\mathrm{Exp}\left(\mathrm{R}^\top\phi\right) \end{aligned} \tag{3.11}$$

3.2 Visual-Inertial Fusion

3.2.1 IMU Preintegration Factor

Residual Errors

The IMU preintegration measurement $\Delta\mathcal{I}_{\mathrm{ij}} = \left[\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}, \Delta\tilde{\mathbf{v}}_{\mathrm{ij}}^{\top}, \Delta\tilde{\mathbf{p}}_{\mathrm{ij}}^{\top}\right]^{\top} \in \mathbb{R}^9$ between keyframe i and j is determined as

$$\begin{aligned}
\Delta\tilde{\mathrm{R}}_{\mathrm{ij}} &= \tilde{\mathrm{R}}_i^{\top}\tilde{\mathrm{R}}_j \operatorname{Exp}\left(\delta\phi_{\mathrm{ij}}\right) \\
\Delta\tilde{\mathbf{v}}_{\mathrm{ij}} &= \tilde{\mathrm{R}}_i^{\top}\left(\tilde{\mathbf{v}}_j - \tilde{\mathbf{v}}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) + \delta\mathbf{v}_{\mathrm{ij}} \\
\Delta\tilde{\mathbf{p}}_{\mathrm{ij}} &= \tilde{\mathrm{R}}_i^{\top}\left(\tilde{\mathbf{p}}_j - \tilde{\mathbf{p}}_i - \tilde{\mathbf{v}}_i\Delta t_{\mathrm{ij}} - \frac{1}{2}\mathbf{g}\Delta t_{\mathrm{ij}}^2\right) + \delta\mathbf{p}_{\mathrm{ij}}
\end{aligned} \tag{3.12}$$

where

— $\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}$,$\Delta\tilde{\mathbf{v}}_{\mathrm{ij}}$,$\Delta\tilde{\mathbf{p}}_{\mathrm{ij}}$ are the preintegrated rotation, velocity and position measurement.

— $\tilde{\mathrm{R}}_i$,$\tilde{\mathrm{R}}_j$ $\in$ SO(3) are the rotation measured from IMU at keyframe i and j.

— $\tilde{\mathbf{v}}_i$,$\tilde{\mathbf{v}}_j \in \mathbb{R}^3$ are the velocity measured from IMU at keyframe i and j.

— $\tilde{\mathbf{p}}_i$,$\tilde{\mathbf{p}}_j \in \mathbb{R}^3$ are the position measured from IMU at keyframe i and j.

— $\delta\phi_{\mathrm{ij}}, \delta\mathbf{v}_{\mathrm{ij}}, \delta\mathbf{p}_{\mathrm{ij}} \in \mathbb{R}^3$ are the “random" noise of preintegrated rotation, velocity and position measurement.

— $\mathbf{g} = [0,0,9.81]^{\top}$ is the gravity vector in world frame.

— Δt_{ij} is the time difference between keyframe i and j.

In general, the IMU preintegration are written as a function of the (to-be-estimated) state “plus” a random noise, described by $[\delta\phi_{\mathrm{ij}}, \delta\mathbf{v}_{\mathrm{ij}}, \delta\mathbf{p}_{\mathrm{ij}}]^{\top}$. For more details of mathematical underneath, please reference to original paper [13].

Given the IMU preintegration measurement (3.12), we can compute the IMU factor residual error $\mathbf{r}_{\mathcal{I}_{\mathrm{ij}}} = \left[\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}^{\top}, \mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}^{\top}, \mathbf{r}_{\Delta\mathbf{p}_{\mathrm{ij}}}^{\top}\right]^{\top} \in \mathbb{R}^9$ between two consecutive keyframe i and j

$$\begin{aligned}
\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}} &= \operatorname{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_i^{\top}\mathrm{R}_j\right) \\
\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}} &= \mathrm{R}_i^{\top}\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) - \Delta\tilde{\mathbf{v}}_{\mathrm{ij}} \\
\mathbf{r}_{\Delta\mathbf{p}_{\mathrm{ij}}} &= \mathrm{R}_i^{\top}\left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{\mathrm{ij}} - \frac{1}{2}\mathbf{g}\Delta t_{\mathrm{ij}}^2\right) - \Delta\tilde{\mathbf{p}}_{\mathrm{ij}}
\end{aligned} \tag{3.13}$$

where

— $\mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}, \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}, \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$ are the residual error vector of rotation, velocity and position.

— $\mathrm{R}_i, \mathrm{R}_j \in \mathrm{SO}(3)$ are the rotation from body frame "B" to world frame "W" of keyframe i and j.

— $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^3$ are the velocity from body frame "B" to world frame "W" of keyframe i and j.

— $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^3$ are the translation from body frame "B" to world frame "W" of keyframe i and j.

Jacobian of Residual Errors

In this section, we present the analytical expression for the Jacobian of the IMU Preintegrated Factor. All the derivative model in this work is computed by *Perturbation Model* where we add a small disturbance into residual error (3.13):

$$\begin{aligned} \mathrm{R}_i &\leftarrow \mathrm{R}_i \operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right), & \mathrm{R}_j &\leftarrow \mathrm{R}_j \operatorname{Exp}\left(\delta \boldsymbol{\phi}_j\right) \\ \mathbf{p}_i &\leftarrow \mathbf{p}_i + \mathrm{R}_i \delta \mathbf{p}_i, & \mathbf{p}_j &\leftarrow \mathbf{p}_j + \mathrm{R}_j \delta \mathbf{p}_j \\ \mathbf{v}_i &\leftarrow \mathbf{v}_i + \delta \mathbf{v}_i, & \mathbf{v}_j &\leftarrow \mathbf{v}_j + \delta \mathbf{v}_i \\ \delta \mathbf{b}_i^g &\leftarrow \delta \mathbf{b}_i^g + \tilde{\delta} \mathbf{b}_i^g, & \delta \mathbf{b}_i^a &\leftarrow \delta \mathbf{b}_i^a + \tilde{\delta} \mathbf{b}_i^a \end{aligned} \tag{3.14}$$

— Jacobian of $\mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}$: We note that $\mathbf{p}_i, \mathbf{p}_j, \mathbf{v}_i, \mathbf{v}_j, \delta \mathbf{b}_i^a$ do not appear in the expression of $\mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}$, hence, its correspond Jacobian equals to zero. The remaining Jacobian is computed as following:
We perturb R_{i} by $\operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right)$, the change of $\mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}$ relative to this disturbance is

$$\begin{aligned} \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_i \operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right) &= \operatorname{Log}\left(\Delta \tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\left(\mathrm{R}_i \operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right)^{\top} \mathrm{R}_j\right) \\ &= \operatorname{Log}\left(\Delta \tilde{\mathrm{R}}_{\mathrm{ij}}^{\top} \operatorname{Exp}\left(-\delta \boldsymbol{\phi}_i\right) \mathrm{R}_i^{\top} \mathrm{R}_j\right) \\ &\overset{(3.11)}{=} \operatorname{Log}\left(\Delta \tilde{\mathrm{R}}_{\mathrm{ij}}^{\top} \mathrm{R}_i^{\top} \mathrm{R}_j \operatorname{Exp}\left(-\mathrm{R}_j^{\top} \mathrm{R}_i \delta \boldsymbol{\phi}_i\right)\right) \\ &\overset{(3.9)}{\simeq} \mathbf{r}_{\Delta \mathrm{R}_{ij}}\left(\mathrm{R}_i\right) - \mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta \mathrm{R}}\left(\mathrm{R}_i\right)\right) \mathrm{R}_j^{\top} \mathrm{R}_i \delta \boldsymbol{\phi}_i \end{aligned} \tag{3.15}$$

From (3.15), we can compute the Jacobian w.r.t rotation $\boldsymbol{\phi}_i$ of keyframe $\mathcal{K}_i$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \boldsymbol{\phi}_i} &= \lim_{\delta\boldsymbol{\phi}_i \to 0} \frac{\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_i \operatorname{Exp}\left(\delta\boldsymbol{\phi}_i\right)\right) - \mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_i\right)}{\delta\boldsymbol{\phi}_i} \\ &\overset{(3.15)}{=} -\mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta\mathrm{R}_{ij}}\left(\mathrm{R}_i\right)\right)\mathrm{R}_j^\top \mathrm{R}_i\end{aligned} \tag{3.16}$$

We perturb R_j by $\operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)$, the change of $\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}$ relative to this disturbance is

$$\begin{aligned}\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_j \operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right) &= \operatorname{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^\top \mathrm{R}_i^\top \left(\mathrm{R}_j \operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right)\right) \\ &\overset{(3.9)}{\simeq} \mathbf{r}_{\Delta\mathrm{R}}\left(\mathrm{R}_j\right) + \mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta\mathrm{R}}\left(\mathrm{R}_j\right)\right)\delta\boldsymbol{\phi}_j\end{aligned} \tag{3.17}$$

Similar to (3.16), we obtain the Jacobian w.r.t rotation $\boldsymbol{\phi}_j$ of keyframe $\mathcal{K}_j$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \boldsymbol{\phi}_j} &= \lim_{\delta\boldsymbol{\phi}_j \to 0} \frac{\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_j \operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right) - \mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}\left(\mathrm{R}_j\right)}{\delta\boldsymbol{\phi}_j} \\ &\overset{(3.17)}{=} \mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta\mathrm{R}}\left(\mathrm{R}_j\right)\right)\end{aligned} \tag{3.18}$$

In summary, the Jacobian of $\mathbf{r}_{\Delta\mathrm{R}_{ij}}$ are

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_i} &= -\mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta\mathrm{R}}\left(\mathrm{R}_i\right)\right)\mathrm{R}_j^\mathrm{T}\mathrm{R}_i & \frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_j} &= \mathrm{J}_r^{-1}\left(\mathbf{r}_{\Delta\mathrm{R}}\left(\mathrm{R}_j\right)\right) \\ \frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \mathbf{v}_i} &= \mathbf{0} & \frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \mathbf{v}_j} &= \mathbf{0} \\ \frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \mathbf{p}_i} &= \mathbf{0} & \frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{\mathrm{ij}}}}{\partial \mathbf{p}_j} &= \mathbf{0}\end{aligned} \tag{3.19}$$

— Jacobian of $\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}$: As previous, the terms $\boldsymbol{\phi}_j$,$\mathbf{p}_i$,$\mathbf{p}_j$ are not appear in $\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}$ expression, so the Jacobian w.r.t to them equal to zero. We add the perturb the remaining terms and obtains:

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathbf{v}_i + \delta\mathbf{v}_i\right) &= \mathrm{R}_i^\top\left(\mathbf{v}_j - \mathbf{v}_i - \delta\mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) \\ &= \mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathbf{v}_i\right) - \mathrm{R}_i^\top\delta\mathbf{v}_i\end{aligned} \tag{3.20}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathbf{v}_j + \delta\mathbf{v}_j\right) &= \mathrm{R}_i^\top\left(\mathbf{v}_j + \delta\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) \\ &= \mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathbf{v}_j\right) + \mathrm{R}_i^\top\delta\mathbf{v}_j\end{aligned} \tag{3.21}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathrm{R}_i \operatorname{Exp}\left(\delta\phi_i\right)\right) &= \left(\mathrm{R}_i \operatorname{Exp}\left(\delta\boldsymbol{\phi}_i\right)\right)^\top\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) \\ &\overset{(3.5)}{\simeq} \left(\mathbf{I} - \delta\phi_i^\wedge\right)\mathrm{R}_i^\top\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right) \\ &\overset{(3.3)}{=} \mathbf{r}_{\Delta\mathbf{v}_{\mathrm{ij}}}\left(\mathrm{R}_i\right) + \left(\mathrm{R}_i^\top\left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}}\right)\right)^\wedge\delta\boldsymbol{\phi}_i\end{aligned} \tag{3.22}$$

We determine the Jacobian w.r.t to $\boldsymbol{\phi}_i$,$\mathbf{v}_i$,$\mathbf{v}_j$ as following

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \mathbf{v}_i} &= \lim_{\delta \mathbf{v}_i \to 0} \frac{\mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathbf{v}_i + \delta \mathbf{v}_i) - \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathbf{v}_i)}{\delta \mathbf{v}_i} \\ &\overset{(3.20)}{=} -\mathrm{R}_i^\top\end{aligned} \tag{3.23}$$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \mathbf{v}_j} &= \lim_{\delta \mathbf{v}_j \to 0} \frac{\mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathbf{v}_j + \delta \mathbf{v}_j) - \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathbf{v}_j)}{\delta \mathbf{v}_j} \\ &\overset{(3.21)}{=} \mathrm{R}_i^\top\end{aligned} \tag{3.24}$$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{ij}}}{\partial \boldsymbol{\phi}_i} &= \lim_{\delta \boldsymbol{\phi}_i \to 0} \frac{\mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathrm{R}_i \operatorname{Exp}(\delta \phi_i)) - \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}(\mathrm{R}_i)}{\delta \boldsymbol{\phi}_i} \\ &\overset{(3.22)}{=} \left(\mathrm{R}_i^\top (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}})\right)^\wedge\end{aligned} \tag{3.25}$$

In summary, the Jacobian of $\mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}$ are

$$\begin{aligned}&\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_i} = \left(\mathrm{R}_i^\top (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{\mathrm{ij}})\right)^\wedge && \frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_j} = \mathbf{0} \\ &\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \mathbf{p}_i} = \mathbf{0} && \frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \mathbf{p}_j} = \mathbf{0} \\ &\frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \mathbf{v}_i} = -\mathrm{R}_i^\top && \frac{\partial \mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}}{\partial \mathbf{v}_j} = \mathrm{R}_i^\top\end{aligned} \tag{3.26}$$

— Jacobian of $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$: R_j and $\mathbf{v}_j$ are not appear in $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$, hence, Jacobian w.r.t to $\delta\boldsymbol{\phi}_j$ and $\delta\mathbf{v}_j$ are zero. Similar to previous, we perturb $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$ by the remaining terms

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{p}_i + \mathrm{R}_i \delta \mathbf{p}_i) &= \mathrm{R}_i^\top \left(\mathbf{p}_j - \mathbf{p}_i - \mathrm{R}_i \delta \mathbf{p}_i - \mathbf{v}_i \Delta t_{\mathrm{ij}} - \frac{1}{2}\mathbf{g}\Delta t_{\mathrm{ij}}^2\right) \\ &= \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{p}_i) + (-\mathbf{I}_{3\times 1})\,\delta \mathbf{p}_i\end{aligned} \tag{3.27}$$

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{p}_j + \mathrm{R}_j \delta \mathbf{p}_j) &= \mathrm{R}_i^\top \left(\mathbf{p}_j + \mathrm{R}_j \delta \mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta t_{\mathrm{ij}} - \frac{1}{2}\mathbf{g}\Delta t_{\mathrm{ij}}^2\right) \\ &= \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{p}_j) + \left(\mathrm{R}_i^\top \mathrm{R}_j\right)\delta \mathbf{p}_j\end{aligned} \tag{3.28}$$

$$\begin{aligned}\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{v}_i + \delta \mathbf{v}_i) &= \mathrm{R}_i^\top \left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta t_{\mathrm{ij}} - \delta \mathbf{v}_i \Delta t_{\mathrm{ij}} - \frac{1}{2}\mathbf{g}\Delta t_{\mathrm{ij}}^2\right) \\ &= \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}(\mathbf{v}_i) + \left(-\mathrm{R}_i^\top \Delta t_{\mathrm{ij}}\right)\delta \mathbf{v}_i\end{aligned} \tag{3.29}$$

$$
\begin{aligned}
\mathbf{r}_{\Delta \mathrm{p}_{\mathrm{ij}}}\left(\mathrm{R}_i \operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right) &= \left(\mathrm{R}_i \operatorname{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right)^{\top}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i \Delta t_{\mathrm{ij}}-\frac{1}{2} \mathbf{g} \Delta t_{\mathrm{ij}}^2\right) \\
&\overset{(3.5)}{\simeq} \left(\mathbf{I}-\delta \boldsymbol{\phi}_i^{\wedge}\right) \mathrm{R}_i^{\top}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i \Delta t_{\mathrm{ij}}-\frac{1}{2} \mathbf{g} \Delta t_{\mathrm{ij}}^2\right) \\
&\overset{(3.3)}{=} \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right)+\left(\mathrm{R}_i^{\top}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i \Delta t_{\mathrm{ij}}-\frac{1}{2} \mathbf{g} \Delta t_{\mathrm{ij}}^2\right)\right)^{\wedge} \delta \boldsymbol{\phi}_i
\end{aligned} \tag{3.30}
$$

Then the Jacobian w.r.t to $\mathbf{p}_i$, $\mathbf{p}_j$, $\boldsymbol{\phi}_i$ can be found

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \mathbf{p}_i} &= \lim_{\delta \mathbf{p}_i \rightarrow 0} \frac{\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_i+\mathrm{R}_i \delta \mathbf{p}_i\right)-\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_i\right)}{\delta \mathbf{p}_i} \\
&\overset{(3.27)}{=} -\mathbf{I}_{3 \times 1}
\end{aligned} \tag{3.31}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \mathbf{p}_j} &= \lim_{\delta \mathbf{p}_j \rightarrow 0} \frac{\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_j+\mathrm{R}_j \delta \mathbf{p}_j\right)-\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_j\right)}{\delta \mathbf{p}_j} \\
&\overset{(3.28)}{=} \mathrm{R}_i^{\top} \mathrm{R}_j
\end{aligned} \tag{3.32}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \mathbf{v}_i} &= \lim_{\delta \mathbf{v}_i \rightarrow 0} \frac{\mathbf{r}_{\Delta \mathbf{v}_{\mathrm{ij}}}\left(\mathbf{v}_i+\delta \mathbf{v}_i\right)-\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{v}_i\right)}{\delta \mathbf{v}_i} \\
&\overset{(3.29)}{=} -\mathrm{R}_i^{\top} \Delta t_{\mathrm{ij}}
\end{aligned} \tag{3.33}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \boldsymbol{\phi}_i} &= \lim_{\delta \boldsymbol{\phi}_i \rightarrow 0} \frac{\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i \operatorname{Exp}(\delta \boldsymbol{\phi})\right)-\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right)}{\delta \boldsymbol{\phi}_i} \\
&\overset{(3.30)}{=} \left(\mathrm{R}_i^{\top}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i \Delta t_{\mathrm{ij}}-\frac{1}{2} \mathbf{g} \Delta t_{\mathrm{ij}}^2\right)\right)^{\wedge}
\end{aligned} \tag{3.34}
$$

In summary, the Jacobian of $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$ are

$$
\begin{aligned}
&\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_i}=\left(\mathrm{R}_i^{\top}\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i \Delta t_{\mathrm{ij}}-\frac{1}{2} \mathbf{g} \Delta t_{\mathrm{ij}}^2\right)\right)^{\wedge} && \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_j}=\mathbf{0} \\
&\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{v}_i}=-\mathrm{R}_i^{\top} \Delta t_{\mathrm{ij}} && \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{v}_j}=\mathbf{0} \\
&\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{p}_i}=-\mathbf{I}_{3 \times 1} && \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{p}_j}=\mathrm{R}_i^{\top} \mathrm{R}_j
\end{aligned} \tag{3.35}
$$

3.2.2 Visual Odometry Factor

Residual Errors

The 3-D coordinates of the landmark l in the world coordinate system ($\mathbf{p}_{\mathrm{W}}$) are converted to the 3-D position in camera coordinate ($\mathbf{p}_{\mathrm{C}}$):

$$\begin{aligned}\mathbf{p}_{\mathrm{C}} &= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathbf{p}_{\mathrm{B}} - \mathbf{t}_{\mathrm{BC}}\right] \\ &= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}} - \mathbf{p}_{\mathrm{WB}}\right) - \mathbf{t}_{\mathrm{BC}}\right]\end{aligned} \tag{3.36}$$

with $\mathbf{p}_{\mathrm{B}}$ is the landmark position in the body/IMU coordinate.

The 2-D position of landmark in image plane is obtain by the standard perspective projection

$$\begin{aligned}\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} &= \mathrm{K}\cdot\mathbf{p}_{\mathrm{C}} \\ &\overset{(3.36)}{=} \mathrm{K}\mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}} - \mathbf{p}_{\mathrm{WB}}\right) - \mathbf{t}_{\mathrm{BC}}\right]\end{aligned} \tag{3.37}$$

where $\mathrm{K} \in \mathbb{R}^{3\times 3}$ is the projection matrix which is depend on camera models.

The residual error of a single image measurement $\mathbf{z}_{\mathrm{il}}$ is the reprojection error of landmark l on keyframe $\mathcal{K}_i$

$$\begin{aligned}\mathbf{r}_{\mathcal{C}_{\mathrm{il}}} &= \mathbf{z}_{\mathrm{il}} - \mathbf{p} \\ &\overset{(3.37)}{=} \mathbf{z}_{\mathrm{il}} - \mathrm{K}\mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}} - \mathbf{p}_{\mathrm{WB}}\right) - \mathbf{t}_{\mathrm{BC}}\right]\end{aligned} \tag{3.38}$$

Jacobian of Residual Errors

— Jacobian w.r.t $\mathbf{p}_{\mathrm{W}}$: we apply *chain rule*

$$\begin{aligned}\frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{p}_{\mathrm{W}}} &= \frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{p}_{\mathrm{C}}}\cdot\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{p}_{\mathrm{W}}} \\ &= -\frac{\partial \mathbf{p}}{\partial \mathbf{p}_{\mathrm{C}}}\cdot\frac{\partial\left[\mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}} - \mathbf{p}_{\mathrm{WB}}\right) - \mathbf{t}_{\mathrm{BC}}\right]\right]}{\partial \mathbf{p}_{\mathrm{W}}} \\ &= -\pi\cdot\mathrm{R}_{\mathrm{BC}}^{\top}\mathrm{R}_{\mathrm{WB}}^{\top}\end{aligned} \tag{3.39}$$

where $\pi \in \mathbb{R}^{2\times 3}$ is the Perspective Projection Jacobian Matrix.

— Jacobian w.r.t Keyframe $\mathcal{K}_i$'s pose $\mathrm{T}_{\mathrm{WB}} = [\mathrm{R}_{\mathrm{WB}}, \mathbf{p}_{\mathrm{WB}}]$: we apply *chain rule*

$$\begin{aligned}\frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{T}_{\mathrm{WB}}} &= \frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{p}_{\mathrm{C}}}\cdot\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{T}_{\mathrm{WB}}} \\ &= -\pi\cdot\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{T}_{\mathrm{WB}}}\end{aligned} \tag{3.40}$$

We can rewrite $\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{T}_{\mathrm{WB}}}$ as the "vectorized" form $\left[\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \boldsymbol{\phi}_{\mathrm{WB}}} \quad \frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{p}_{\mathrm{WB}}}\right]$ and compute them separately.

Partial derivative w.r.t $\boldsymbol{\phi}_{\mathrm{WB}}$: First, we perturb $\boldsymbol{\phi}_{\mathrm{WB}}$, the change of $\mathbf{p}_{\mathrm{C}}$ relative to these disturbance

$$\begin{aligned}
\mathbf{p}_{\mathrm{C}}\left(\mathrm{R}_{\mathrm{WB}} \operatorname{Exp}\left(\delta \boldsymbol{\phi}_{\mathrm{WB}}\right)\right) &= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\left(\mathrm{R}_{\mathrm{WB}} \operatorname{Exp}\left(\delta \boldsymbol{\phi}_{\mathrm{WB}}\right)\right)^{\top}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)-\mathbf{t}_{\mathrm{BC}}\right] \\
&= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\operatorname{Exp}\left(\delta \boldsymbol{\phi}_{\mathrm{WB}}\right)^{\top} \mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)-\mathbf{t}_{\mathrm{BC}}\right] \\
&\overset{(3.5)}{\simeq} \mathrm{R}_{\mathrm{BC}}^{\top}\left[\left(\mathbf{I}-\delta \boldsymbol{\phi}_{\mathrm{WB}}^{\wedge}\right) \mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)-\mathbf{t}_{\mathrm{BC}}\right] \\
&= \mathbf{p}_{\mathrm{C}}\left(\mathrm{R}_{\mathrm{WB}}\right)-\mathrm{R}_{\mathrm{BC}}^{\top} \delta \boldsymbol{\phi}_{\mathrm{WB}}^{\wedge} \mathrm{R}_{\mathrm{WB}}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right) \\
&\overset{(3.3)}{=} \mathbf{p}_{\mathrm{C}}\left(\mathrm{R}_{\mathrm{WB}}\right)+\mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)\right]^{\wedge} \delta \boldsymbol{\phi}_{\mathrm{WB}}
\end{aligned} \tag{3.41}$$

We can compute Partial derivative w.r.t $\boldsymbol{\phi}_{\mathrm{WB}}$ as

$$\begin{aligned}
\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \boldsymbol{\phi}_{\mathrm{WB}}} &= \lim_{\delta \boldsymbol{\phi}_{\mathrm{WB}} \rightarrow 0} \frac{\mathbf{p}_{\mathrm{C}}\left(\mathrm{R}_{\mathrm{WB}} \operatorname{Exp}\left(\delta \boldsymbol{\phi}_{\mathrm{WB}}\right)\right) \mathbf{p}_{\mathrm{C}}\left(\mathrm{R}_{\mathrm{WB}}\right)}{\delta \boldsymbol{\phi}_{\mathrm{WB}}} \\
&\overset{(3.41)}{=} \lim_{\delta \boldsymbol{\phi} \rightarrow 0} \frac{-\mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)\right]^{\wedge} \delta \boldsymbol{\phi}_{\mathrm{WB}}}{\delta \boldsymbol{\phi}_{\mathrm{WB}}} \\
&= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}\right)\right]^{\wedge} \\
&= \mathrm{R}_{\mathrm{BC}}^{\top} \mathbf{p}_{\mathrm{B}}^{\wedge}
\end{aligned} \tag{3.42}$$

Partial derivative w.r.t $\mathbf{p}_{\mathrm{WB}}$: Similarly, we perturb $\mathbf{p}_{\mathrm{WB}}$

$$\begin{aligned}
\mathbf{p}_{\mathrm{C}}\left(\mathbf{p}_{\mathrm{WB}}+\mathrm{R}_{\mathrm{WB}} \delta \mathbf{p}_{\mathrm{WB}}\right) &= \mathrm{R}_{\mathrm{BC}}^{\top}\left[\mathrm{R}_{\mathrm{WB}}^{\top}\left(\mathbf{p}_{\mathrm{W}}-\mathbf{p}_{\mathrm{WB}}-\mathrm{R}_{\mathrm{WB}} \delta \mathbf{p}_{\mathrm{WB}}\right)-\mathbf{t}_{\mathrm{BC}}\right] \\
&= \mathbf{p}_{\mathrm{C}}\left(\mathbf{p}_{\mathrm{WB}}\right)-\mathrm{R}_{\mathrm{BC}}^{\top} \delta \mathbf{p}_{\mathrm{WB}}
\end{aligned} \tag{3.43}$$

We can compute Partial derivative w.r.t $\mathbf{p}_{\mathrm{WB}}$ as

$$\begin{aligned}
\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{p}_{\mathrm{WB}}} &= \lim_{\delta \mathbf{p}_{\mathrm{WB}} \rightarrow 0} \frac{\mathbf{p}_{\mathrm{C}}\left(\mathbf{p}_{\mathrm{WB}}+\mathrm{R}_{\mathrm{WB}} \delta \mathbf{p}_{\mathrm{WB}}\right)-\mathbf{p}_{\mathrm{C}}\left(\mathbf{p}_{\mathrm{WB}}\right)}{\delta \mathbf{p}_{\mathrm{WB}}} \\
&\overset{(3.43)}{=} \lim_{\delta \mathbf{p}_{\mathrm{WB}} \rightarrow 0} \frac{-\mathrm{R}_{\mathrm{BC}}^{\top} \delta \mathbf{p}_{\mathrm{WB}}}{\delta \mathbf{p}_{\mathrm{WB}}} \\
&= -\mathrm{R}_{\mathrm{BC}}^{\top}
\end{aligned} \tag{3.44}$$

Stack (3.42) and (3.44), we obtain the partial derivative of $\mathbf{p}_{\mathrm{C}}$ w.r.t $\mathbf{T}_{\mathrm{WB}}$

$$\frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \mathbf{T}_{\mathrm{WB}}}=\left[\mathrm{R}_{\mathrm{BC}}^{\top} \mathbf{p}_{\mathrm{B}}^{\wedge} \quad -\mathrm{R}_{\mathrm{BC}}^{\top}\right]=\mathrm{R}_{\mathrm{BC}}^{\top} \cdot\left[\mathbf{p}_{\mathrm{B}}^{\wedge} \quad -\mathbf{I}_{3 \times 3}\right] \tag{3.45}$$

Substitute (3.45) into (3.40), we receive the analytical Jacobian w.r.t Keyframe $\mathcal{K}_i$'s pose expression

$$\frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{T}_{\mathrm{WB}}} = -\pi \cdot \mathrm{R}_{\mathrm{BC}}^{\top} \cdot [\mathbf{p}_{\mathrm{B}}^{\wedge} \quad - \mathbf{I}_{3\times 3}] \tag{3.46}$$

In summary, Jacobian matrices of residual error in Visual Odometry factor are

$$\begin{aligned} \frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{p}_{\mathrm{W}}} &= -\pi \cdot \mathrm{R}_{\mathrm{BC}}^{\top} \mathrm{R}_{\mathrm{WB}}^{\top} \\ \frac{\partial \mathbf{r}_{\mathcal{C}_{\mathrm{il}}}}{\partial \mathbf{T}_{\mathrm{WB}}} &= -\pi \cdot \mathrm{R}_{\mathrm{BC}}^{\top} \cdot [\mathbf{p}_{\mathrm{B}}^{\wedge} \quad - \mathbf{I}_{3\times 3}] \end{aligned} \tag{3.47}$$

4 Visual Inertial Wheel Odometry

4.1 Wheel Odometry Factor in SE(3)

4.1.1 Residual Errors

The wheel odometer preintegration measurement $\Delta\mathcal{O}_{\mathrm{ij}} = \left[\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^\top, \Delta\tilde{\mathbf{p}}_{\mathrm{ij}}^\top\right]^\top$ between keyframe i and j is determined as

$$\begin{aligned} \Delta\tilde{\mathrm{R}}_{\mathrm{ij}} &= \tilde{\mathrm{R}}_i^\top \tilde{\mathrm{R}}_j \operatorname{Exp}\left(\delta\phi_{\mathrm{ij}}\right) \\ \Delta\tilde{\mathbf{p}}_{\mathrm{ij}} &= \tilde{\mathrm{R}}_i^\top \left(\tilde{\mathbf{p}}_j - \tilde{\mathbf{p}}_i\right) + \delta\mathbf{p}_{\mathrm{ij}} \end{aligned} \tag{4.1}$$

where

— $\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}$ and $\Delta\tilde{\mathbf{p}}_{\mathrm{ij}}$ are the preintegrated rotation and position measurement.

— $\tilde{\mathrm{R}}_i$,$\tilde{\mathrm{R}}_j \in \mathrm{SO}(3)$ are the rotation measured from wheel odometer at keyframe i and j.

— $\tilde{\mathbf{p}}_i$,$\tilde{\mathbf{p}}_j \in \mathbb{R}^3$ are the position measured from wheel odometer at keyframe i and j.

— $\delta\phi_{\mathrm{ij}}$ and $\delta\mathbf{p}_{\mathrm{ij}} \in \mathbb{R}^3$ are the “random" noise of preintegrated rotation and position measurement.

Given the wheel odometer preintegration measurement (4.1), we can compute the wheel odometry factor residual error $\mathbf{r}_{\mathcal{O}_{\mathrm{ij}}} = \left[\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}^\top, \mathbf{r}_{\Delta\mathbf{p}_{\mathrm{ij}}}^\top\right]^\top \in \mathbb{R}^3$ between two consecutive keyframe i and j

$$\begin{aligned} \mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}} &= \mathbf{e}_3^\top \operatorname{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^\top \left(\mathrm{R}_i\mathrm{R}_{\mathrm{BO}}\right)^\top \mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right) \\ &= \mathbf{e}_3^\top \operatorname{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^\top \mathrm{R}_{\mathrm{BO}}^\top \mathrm{R}_i^\top \mathrm{R}_j \mathrm{R}_{\mathrm{BO}}\right) \\ \mathbf{r}_{\Delta\mathbf{p}_{\mathrm{ij}}} &= \Lambda \mathrm{R}_{\mathrm{BO}}^\top \left[\mathrm{R}_i^\top \left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R}_{\mathrm{j}}\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] - \Delta\tilde{\mathbf{p}}_{\mathrm{ij}} \end{aligned} \tag{4.2}$$

where

— $\mathbf{r}_{\Delta\mathrm{R}_{\mathrm{ij}}}$ and $\mathbf{r}_{\Delta\mathbf{p}_{\mathrm{ij}}}$ are the residual error vector of rotation and position.

— R_i,$\mathrm{R}_j \in \mathrm{SO}(3)$ are the rotation from body frame “B” to world frame “W” of keyframe i and j.

— $\mathbf{p}_i$,$\mathbf{p}_j \in \mathbb{R}^3$ are the translation from body frame “B” to world frame “W” of keyframe i and j.

— $\Lambda = [\mathbf{e}_1 \quad \mathbf{e}_2]^\top$ with $\mathbf{e}_i$ is the i-th standard unit basis vector.

4.1.2 Jacobian of Residual Errors

For convenience, we denote

$$\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}} = \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)$$
$$\mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}} = \mathrm{R}_{\mathrm{BO}}^{\top}\left[\mathrm{R}_i^{\top}\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R}_{\mathrm{j}}\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right]$$

Instead of compute Jacobian directly on $\mathbf{r}_{\Delta \mathrm{R_{ij}}}$ and $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$, we will take the partial derivative of $\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}$ and $\mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}}$.

— Jacobian of $\mathbf{r}_{\Delta \mathrm{R_{ij}}}$: Notice that $\mathbf{p}_i$ and $\mathbf{p}_j$ are not appear in $\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}$, hence, their Jacobian are zero. We perturb the remaining term

$$\begin{aligned}\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_i\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_i\right)\right) &= \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\left(\mathrm{R}_i\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_i\right)\right)^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)\\ &= \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\,\mathrm{Exp}\left(-\delta\boldsymbol{\phi}_i\right)\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)\\ &\overset{(3.11)}{=} \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\,\mathrm{Exp}\left(-\left(\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)^{\top}\delta\boldsymbol{\phi}_i\right)\right)\\ &\overset{(3.9)}{\simeq} \mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_i\right) - \mathrm{J}_r^{-1}\left(\mathbf{r}^{\star}_{\Delta \mathrm{R}_{ij}}\right)\left(\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)^{\top}\delta\boldsymbol{\phi}_i\end{aligned} \tag{4.3}$$

$$\begin{aligned}\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_j\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right) &= \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\mathrm{R}_i^{\top}\mathrm{R}_j\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_j\right)\mathrm{R}_{\mathrm{BO}}\right)\\ &\overset{(3.11)}{=} \mathrm{Log}\left(\Delta\tilde{\mathrm{R}}_{\mathrm{ij}}^{\top}\mathrm{R}_{\mathrm{BO}}^{\top}\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\,\mathrm{Exp}\left(\mathrm{R}_{\mathrm{BO}}^{\top}\delta\boldsymbol{\phi}_j\right)\right)\\ &\overset{(3.9)}{\simeq} \mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_j\right) + \mathrm{J}_r^{-1}\left(\mathbf{r}^{\star}_{\Delta \mathrm{R}_{ij}}\right)\mathrm{R}_{\mathrm{BO}}^{\top}\delta\boldsymbol{\phi}_j\end{aligned} \tag{4.4}$$

We can compute the Jacobian w.r.t rotation $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \boldsymbol{\phi}_i} &= \lim_{\delta\phi_i \to 0}\frac{\mathbf{e}_3^{\top}\left[\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_i\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_i\right)\right) - \mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_i\right)\right]}{\delta\boldsymbol{\phi}_i}\\ &\overset{(4.3)}{=} -\mathbf{e}_3^{\top}\mathrm{J}_r^{-1}\left(\mathbf{r}^{\star}_{\Delta \mathrm{R}_{ij}}\right)\left(\mathrm{R}_i^{\top}\mathrm{R}_j\mathrm{R}_{\mathrm{BO}}\right)^{\top}\end{aligned} \tag{4.5}$$

$$\begin{aligned}\frac{\partial \mathbf{r}_{\Delta \mathrm{R}_{ij}}}{\partial \boldsymbol{\phi}_j} &= \lim_{\delta\phi_j \to 0}\frac{\mathbf{e}_3^{\top}\left[\mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_j\,\mathrm{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right) - \mathbf{r}^{\star}_{\Delta \mathrm{R_{ij}}}\left(\mathrm{R}_j\right)\right]}{\delta\boldsymbol{\phi}_i}\\ &\overset{(4.4)}{=} \mathbf{e}_3^{\top}\mathrm{J}_r^{-1}\left(\mathbf{r}^{\star}_{\Delta \mathrm{R}_{ij}}\right)\mathrm{R}_{\mathrm{BO}}^{\top}\end{aligned} \tag{4.6}$$

In summary, the Jacobian of $\mathbf{r}_{\Delta \mathrm{R_{ij}}}$ are

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathrm{R_{ij}}}}{\partial \boldsymbol{\phi}_i} &= -\mathbf{e}_3^\top \mathrm{J}_r^{-1}\left(\mathbf{r}^\star_{\Delta \mathrm{R}_{ij}}\right)\left(\mathrm{R}_i^\top \mathrm{R}_j \mathrm{R_{BO}}\right)^\top & \frac{\partial \mathbf{r}_{\Delta \mathrm{R_{ij}}}}{\partial \boldsymbol{\phi}_j} &= \mathbf{e}_3^\top \mathrm{J}_r^{-1}\left(\mathbf{r}^\star_{\Delta \mathrm{R}_{ij}}\right)\mathrm{R_{BO}^\top} \\
\frac{\partial \mathbf{r}_{\Delta \mathrm{R_{ij}}}}{\partial \mathbf{p}_i} &= \mathbf{0} & \frac{\partial \mathbf{r}_{\Delta \mathrm{R_{ij}}}}{\partial \mathbf{p}_j} &= \mathbf{0}
\end{aligned} \tag{4.7}$$

— Jacobian of $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$: Similar, we perturb $\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}$

$$\begin{aligned}
\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i \,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right) &= \mathrm{R_{BO}^\top}\left[\left(\mathrm{R}_i \,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right)^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&\overset{(3.5)}{\simeq} \mathrm{R_{BO}^\top}\left[\left(\mathbf{I} - \delta\phi_i^\wedge\right)\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&= \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right) - \mathrm{R_{BO}^\top}\delta\boldsymbol{\phi}_i^\wedge\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\mathbf{p}_{\mathrm{BO}}\right)\right] \\
&\overset{(3.3)}{=} \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right) + \mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\mathbf{p}_{\mathrm{BO}}\right)\right]^\wedge \delta\boldsymbol{\phi}_i
\end{aligned} \tag{4.8}$$

$$\begin{aligned}
\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_j \,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_j\right)\right) &= \mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_j\right)\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&\overset{(3.5)}{\simeq} \mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\left(\mathbf{I} + \delta\phi_j^\wedge\right)\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&= \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right) + \mathrm{R_{BO}^\top}\mathrm{R}_i^\top\mathrm{R_j}\delta\boldsymbol{\phi}_j^\wedge\mathbf{p}_{\mathrm{BO}} \\
&\overset{(3.3)}{=} \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right) - \mathrm{R_{BO}^\top}\mathrm{R}_i^\top\mathrm{R_j}\mathbf{p}_{\mathrm{BO}}^\wedge\delta\boldsymbol{\phi}_j
\end{aligned} \tag{4.9}$$

$$\begin{aligned}
\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_i + \mathrm{R}_i\delta\mathbf{p}_i\right) &= \mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i - \mathrm{R}_i\delta\mathbf{p}_i + \mathrm{R}_j\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&= \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_i\right) - \mathrm{R_{BO}^\top}\delta\mathbf{p}_i
\end{aligned} \tag{4.10}$$

$$\begin{aligned}
\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_j + \mathrm{R}_j\delta\mathbf{p}_j\right) &= \mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j + \mathrm{R}_j\delta\mathbf{p}_j - \mathbf{p}_i + \mathrm{R}_j\mathbf{p}_{\mathrm{BO}}\right) - \mathbf{p}_{\mathrm{BO}}\right] \\
&= \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathbf{p}_j\right) + \mathrm{R_{BO}^\top}\mathrm{R}_i^\top\mathrm{R}_j\delta\mathbf{p}_j
\end{aligned} \tag{4.11}$$

The Jacobian matrices of $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$ are computed as:

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \boldsymbol{\phi}_i} &= \lim_{\delta\boldsymbol{\phi}_i \to 0} \frac{\Lambda\left[\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i \,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_i\right)\right) - \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_i\right)\right]}{\delta\boldsymbol{\phi}_i} \\
&\overset{(4.8)}{=} \Lambda\mathrm{R_{BO}^\top}\left[\mathrm{R}_i^\top\left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R_j}\mathbf{p}_{\mathrm{BO}}\right)\right]^\wedge
\end{aligned} \tag{4.12}$$

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \boldsymbol{\phi}_j} &= \lim_{\delta\boldsymbol{\phi}_j \to 0} \frac{\Lambda\left[\mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_j \,\mathrm{Exp}\left(\delta \boldsymbol{\phi}_j\right)\right) - \mathbf{r}^\star_{\Delta \mathbf{p}_{\mathrm{ij}}}\left(\mathrm{R}_j\right)\right]}{\delta\boldsymbol{\phi}_i} \\
&\overset{(4.9)}{=} -\Lambda\mathrm{R_{BO}^\top}\mathrm{R}_i^\top\mathrm{R_j}\mathbf{p}_{\mathrm{BO}}^\wedge
\end{aligned} \tag{4.13}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \mathbf{p}_i} = \lim_{\delta \boldsymbol{\phi}_j \to 0} \frac{\Lambda \left[\mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}} \left(\mathbf{p}_i + \mathrm{R}_i \delta \mathbf{p}_i\right) - \mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}} \left(\mathbf{p}_i\right)\right]}{\mathbf{p}_i} \overset{(4.10)}{=} -\Lambda \mathrm{R}_{\mathrm{BO}}^{\top} \tag{4.14}$$

$$\frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{ij}}}{\partial \mathbf{p}_i} = \lim_{\delta \boldsymbol{\phi}_j \to 0} \frac{\Lambda \left[\mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}} \left(\mathbf{p}_j + \mathrm{R}_j \delta \mathbf{p}_j\right) - \mathbf{r}^{\star}_{\Delta \mathbf{p}_{\mathrm{ij}}} \left(\mathbf{p}_i\right)\right]}{\delta \mathbf{p}_i} \overset{(4.11)}{=} \Lambda \mathrm{R}_{\mathrm{BO}}^{\top} \mathrm{R}_i^{\top} \mathrm{R}_j \tag{4.15}$$

In summary, the Jacobian of $\mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}$ are

$$\begin{aligned} \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_i} &= \Lambda \mathrm{R}_{\mathrm{BO}}^{\top} \left[\mathrm{R}_i^{\top} \left(\mathbf{p}_j - \mathbf{p}_i + \mathrm{R}_{\mathrm{j}} \mathbf{p}_{\mathrm{BO}}\right)\right]^{\wedge} & \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \boldsymbol{\phi}_j} &= -\Lambda \mathrm{R}_{\mathrm{BO}}^{\top} \mathrm{R}_i^{\top} \mathrm{R}_{\mathrm{j}} \mathbf{p}_{\mathrm{BO}}^{\wedge} \\ \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{p}_i} &= -\Lambda \mathrm{R}_{\mathrm{BO}}^{\top} & \frac{\partial \mathbf{r}_{\Delta \mathbf{p}_{\mathrm{ij}}}}{\partial \mathbf{p}_j} &= \Lambda \mathrm{R}_{\mathrm{BO}}^{\top} \mathrm{R}_i^{\top} \mathrm{R}_j \end{aligned} \tag{4.16}$$

4.2 Visual-Wheel Odometry on SE(2)

For most of the time, mobile robot is moving in the flat environment where the movement in Z axis remains constant. We will perform the estimation states in SE2-XYZ [58] by fusing optical and wheel odometry sensor. We also detect planar movement then we should switch between performing sensor fusion in SE2-XYZ state or sensor fusion in SE(3).

4.2.1 Reprojection factor on SE(2)

This constraint represents the error between the observed 2D feature $\boldsymbol{x}_i$ and the 2D projected point $\mathbf{x}_i$ from the corresponding landmark 3D l_i w.r.t robot body coordinate system $[\mathrm{R}_i|\mathbf{p}_i] \in$ SE2 and then to the image plane P

$$\mathbf{x}_i = \pi(\mathrm{R}_{\mathrm{CB}} \mathrm{R}_i^T \left(l_i - \mathbf{p}_i\right) + \mathbf{p}_{CB}) + \eta_{\mathbf{x}} \tag{4.17}$$

The feature-based SE(2)-XYZ [58] benefits from encapsulating the out--of-SE(2) motion perturbation and directly parameterizing the robot's poses on SE(2). The out-of-SE(2) motion [58] includes two parts: the translation perturbation along z as $\eta_z \sim \mathcal{N}\left(\mathbf{0}, \sigma_z^2\right)$ and the rotation perturbation xy as

$\boldsymbol{\eta}_{xy} \sim \mathcal{N}\left(\mathbf{0}_{2\times1}, \boldsymbol{\Sigma}_{\theta_{xy}}\right)$. Therefore, the pose can be written as:

$$\mathrm{R}_i \leftarrow \mathrm{Exp}(\underbrace{\begin{bmatrix}\boldsymbol{\eta}_{\theta_{xy}}^T 0\end{bmatrix}^T}_{\boldsymbol{\eta}_\theta})\mathrm{R}_i, \quad \mathbf{p}_i \leftarrow \mathbf{p}_i + \underbrace{\begin{bmatrix} 0 & 0 & \eta_z \end{bmatrix}^T}_{\boldsymbol{\eta}_z} \tag{4.18}$$

then the projection equation (4.17) becomes

$$\begin{aligned}
&\mathbf{x}_i \\
=&\pi\left(\mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T \,\mathrm{Exp}\left(-\boldsymbol{\eta}_\theta\right)\left(\mathbf{l}_\ell - \mathbf{p}_i - \boldsymbol{\eta}_z\right) + \mathbf{p}_{\mathrm{CB}}\right) + \boldsymbol{\eta}_u \\
\approx&\pi\left({}_{\mathrm{C}}l_i\right) + \mathbf{J}_{\boldsymbol{\eta}_\theta}\mathbf{u}_{\boldsymbol{\theta}}\boldsymbol{\eta}_\theta + \mathbf{J}^{\mathbf{u}}_{\boldsymbol{\eta_z}}\boldsymbol{\eta}_z + \boldsymbol{\eta}_u \\
=&\pi\left({}_{\mathrm{C}}l_i\right) + \delta\boldsymbol{\eta}_u
\end{aligned} \tag{4.19}$$

where $\delta\boldsymbol{\eta}_u$ is a synthetic zero-mean noise and landmark position w.r.t camera coordinate ${}_{\mathrm{C}}l_i = \mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T\left(l_i - \mathbf{p}_i\right) + \mathbf{p}_{\mathrm{CB}}$. The noises $\boldsymbol{\eta}_\theta$, $\boldsymbol{\eta}_z$ and $\boldsymbol{\eta}_{\mathbf{x}}$ are independent. Hence we use first-order approximation to linearize the noise terms. The jacobians of $\mathbf{u}$ w.r.t. $(\boldsymbol{\eta}_\theta, \boldsymbol{\eta}_z)$ are computed as:

$$\begin{aligned}
\mathbf{J}^{\mathbf{u}}_{\boldsymbol{\eta}_\theta} &= \mathbf{J}^\pi\left({}_{\mathrm{C}}l_i\right)\mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T\left(\mathbf{l}_\ell - \mathbf{p}_i\right)^\wedge \\
\mathbf{J}^{\mathbf{u}}_{\boldsymbol{\eta}_z} &= -\mathbf{J}^\pi\left({}_{\mathrm{C}}l_i\right)\mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T
\end{aligned}$$

where $\mathbf{J}^\pi$ is the jacobian of the projection model function in terms of the point in the image plane. ORB-SLAM3 provides fish-eye and pinhole models. then we can compute the covariance matrix as:

$$\boldsymbol{\Sigma}_{\delta\boldsymbol{\eta}_u} = \mathbf{J}^{\mathbf{u}}_{\boldsymbol{\eta}_\theta}\Lambda_{12}^T\boldsymbol{\Sigma}_{\theta_{xy}}\Lambda_{12}\mathbf{J}^{\mathbf{u}T}_{\boldsymbol{\eta}_\theta} + \sigma_z^2\mathbf{J}_{\boldsymbol{\eta}_z}\mathbf{e}_3\mathbf{e}_3^T\mathbf{J}^{\mathbf{u}T}_{\boldsymbol{\eta}_z} + \sigma_u^2\mathbf{I}_2 \tag{4.20}$$

As the matching feature of the corresponding landmark is represented in pixels ${}^{i\ell}\boldsymbol{u}$, we can formulate our reprojection error as:

$$r_i = \pi({}_{\mathrm{C}}l_i) - {}^{i\ell}\boldsymbol{u} \tag{4.21}$$

Graph optimization method is used for the minimization of the error stated in (4.21). The information matrix for (4.21) is the inverse of covariance matrix $\boldsymbol{\Sigma}_{\delta\boldsymbol{\eta}_{\mathbf{x}}}$. The Jacobian matrix of r_i w.r.t robot pose are

$$\begin{aligned}
\mathbf{J}_i = &\begin{bmatrix}\frac{\partial r_i}{\partial \mathbf{p}_i} & \frac{\partial {}^{i\ell}\mathbf{e}}{\partial \phi_i}\end{bmatrix}, \\
&\frac{\partial r_i}{\partial \mathbf{p}_i} = -\mathbf{J}^\pi\left({}_{\mathrm{C}}l_i\right)\mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T\Lambda_{12} \\
&\frac{\partial r_i}{\partial \phi_i} = \mathbf{J}^\pi\left({}_{\mathrm{C}}l_i\right)\mathrm{R}_{\mathrm{CB}}\mathrm{R}_i^T\left(\mathbf{1}_\ell - \mathbf{p}_i\right)^\wedge\mathbf{e}_3
\end{aligned} \tag{4.22}$$

The Jacobian matrix of r_i w.r.t landmark l_i is

$$\mathbf{J}_{l_i} = \frac{\partial r_i}{\partial l_i} = \mathbf{J}^{\pi}\left({}_{\mathrm{C}}l_i\right) \mathrm{R}_{\mathrm{CB}} \mathrm{R}_i^T \tag{4.23}$$

4.2.2 Wheel Odometry Factor on SE(2)

Inspired by the work on preintegrated IMU on SE(3) by [13], Zheng and Liu [58] have formulated the preintegration of encoder's measurements on SE(2). From the motion model of the wheel encoder, we get the robot body poses $\boldsymbol{\nu}_i$ and $\boldsymbol{\nu}_j$ between two consecutive odometry readings *k, k+1*, respectively. The preintegrated measurement and the corresponding noises between key frames i, j are formulated as:

$$\begin{aligned} \phi_{ij} &:= \tilde{\phi}_{ij} - \delta\phi_{ij} \\ \mathbf{p}_{ij} &:= \tilde{\mathbf{p}}_{ij} - \delta\mathbf{p}_{ij} \end{aligned} \tag{4.24}$$

The propagation of the integrated noise $\delta\phi_{ij}$, $\delta\mathbf{r}_{ij}$ written in compact form as in [58]

$$\begin{aligned} &\begin{bmatrix} \delta\mathbf{r}_{k+1} \\ \delta\phi_{k+1} \end{bmatrix} := \delta\boldsymbol{\nu}_{k+1} = \mathbf{A}_k \delta\boldsymbol{\nu}_k + \mathbf{B}_k \boldsymbol{\eta}_{\nu k}, \\ &\mathbf{A}_k = \begin{bmatrix} \mathbf{I}_2 & \Phi\left(\tilde{\phi}_k\right) 1^{\times} \tilde{\mathbf{r}}_k \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{B}_k = \begin{bmatrix} \Phi\left(\tilde{\phi}_k\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \end{aligned} \tag{4.25}$$

Hence, the covariance of wheel odometry $\delta\boldsymbol{\nu}_k$ can be propagated at each steps:

$$\boldsymbol{\Sigma}_{\delta\nu_{k+1}} = \mathbf{A}_k \boldsymbol{\Sigma}_{\delta\boldsymbol{\nu}_k} \mathbf{A}_k^T + \mathbf{B}_k \boldsymbol{\Sigma}_{\boldsymbol{\nu}_k} \mathbf{B}_k^T \tag{4.26}$$

We can now formulate the residual error function of preintegrated wheel odometry as follows:

$$\mathbf{r_{ij}} = \begin{bmatrix} \Phi\left(-\phi_i\right)\left(\mathbf{r}_j - \mathbf{p}_i\right) \\ \phi_j - \phi_i \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{r}}_{ij} \\ \tilde{\phi}_{ij} \end{bmatrix} \tag{4.27}$$

where its information matrix is the inverse of the covariance term $\mathbf{\Sigma}_{\delta\nu_{k+1}}$. The Jacobian of error function is:

$$\begin{aligned} \mathbf{J}_i^{ij} &= \frac{\partial \mathbf{r_{ij}}}{\partial \boldsymbol{\nu}_i} = \begin{bmatrix} -\Phi\left(-\phi_i\right) & -\Phi\left(-\phi_i\right) 1^{\times}\left(\mathbf{r}_j - \mathbf{p}_i\right) \\ \mathbf{0} & -1 \end{bmatrix} \\ \mathbf{J}_j^{ij} &= \frac{\partial \mathbf{r_{ij}}}{\partial \boldsymbol{\nu}_j} = \begin{bmatrix} \Phi\left(-\phi_i\right) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \end{aligned} \tag{4.28}$$

4.3 Switching criteria between VIW & VW

After inertial initialization and determining the gravity direction in R-V(I)WO, we monitor the maximum change of movement along the Z-axis in the current local window of key frames.

$$\Delta\bar{Z} = \max \bar{Z}_i - \min \bar{Z}_i : \bar{Z}_i = e3^T * p_i \in local\_window \tag{4.29}$$

if $\Delta\bar{Z}$ < threshold then the planar assumption is activated in both pose optimization in tracking thread, and VW fusion is performed in bundle adjustment in local mapping thread. Otherwise, VIW fusion is activated and the fusion is performed as optimization in SE(3). After experiences, we chose the threshold is 15 cm for our robot and work best at most of the time.

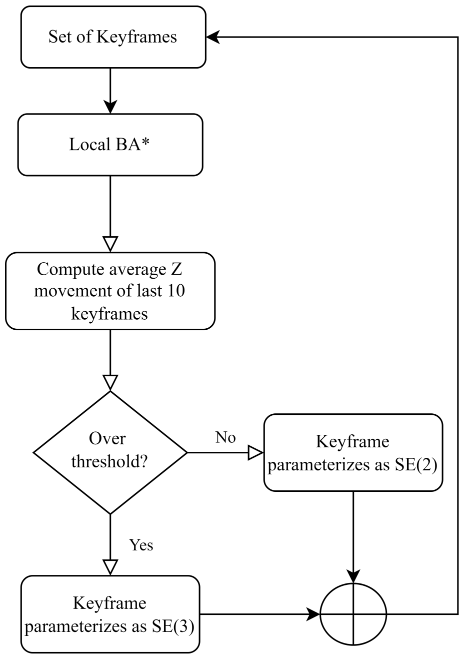

Set of Keyframes
Local BA*
Compute average Z movement of last 10 keyframes
Over threshold?
No
Keyframe parameterizes as SE(2)
Yes
Keyframe parameterizes as SE(3)

Figure 4.1 — Switching criteria between VIW & VW

4.4 Pose Graph Optimization for Sensor Fusion

Since we have three measurements, we will generate three types of factors in the pose graph optimization: IMU Preintegration Factor, Visual Odometry Factor and Wheel Odometry Factor. We phrase the problem as a Least Squares Optimization and use the Levenberg-Marquardt algorithm implemented in the g2o framework [22] to carry out all the optimization. The Levenberg-Marquardt algorithm is the iterative optimization technique to minimize the cost function, hence, the Jacobian Matrix is crucial. For each factor that we are using in this work, we provide the analytical expression to determine the cost function as residual error and its Jacobian. The overview of our graph optimization base is illustrated in Fig 4.2.

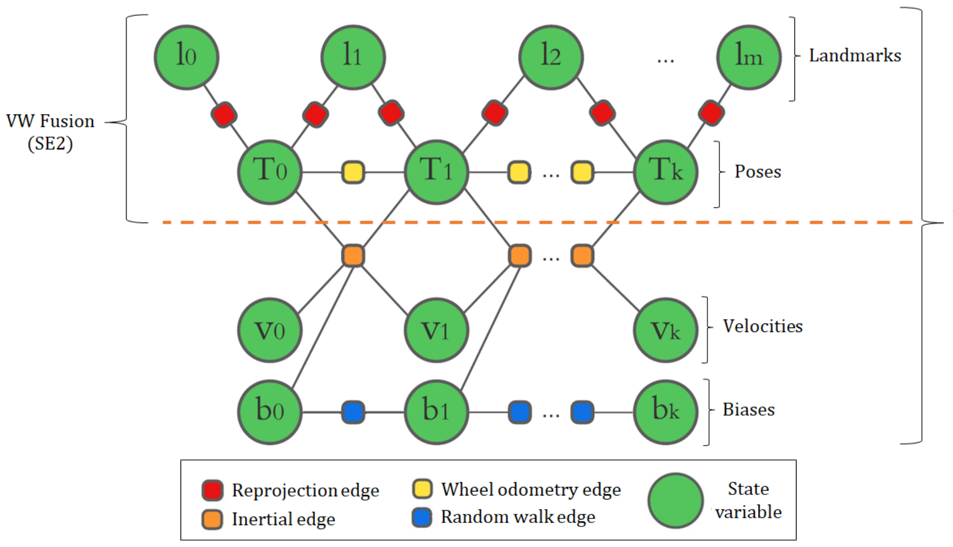


Figure 4.2 — Factor graph for fusing Visual-Inertial-Wheel Odometry sensor

4.5 Dynamic Visual SLAM

It is essential to design and tailor an Visual SLAM system that is compatible to work in high-dynamics enviroment. There are two sets of approach to address this problem, one is geometry-based and the other is machine learning-based.

4.5.1 Geometry-based

The main motive of this methods that we take advantage of the motion consistency to outlier and reject the moving map points. There are two geometry-based methods that is widely use - K-Means cluster and epipolar constraint.

K-Means Cluster

We first separate all the extracted features in each receiving new image by N clusters using K-Means algorithm. Each point groups is assumed that they are belong to the same surface of object. For each point groups c_j, the

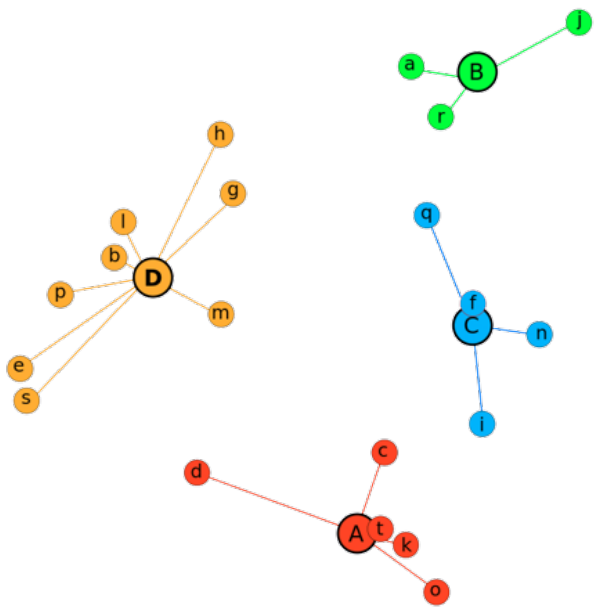


Figure 4.3 — K-Means Cluster Algorithm

average reprojection error r_j of all features $\boldsymbol{u}_i$ with corresponding matched features $\boldsymbol{P}_i$ is computed. When the average error is larger than the a certain threshold, we consider this cluster as moving object. All the features that belong to this cluster are removed and not be used to further tracking.

Epipolar Constraints

The ideal static feature must satisfy the epipolar constraints, on the other hand the moving features will violated the constraint. Fig. 4.4 shows that the static point $\boldsymbol{X}$, which is projected into two consecutive frames i

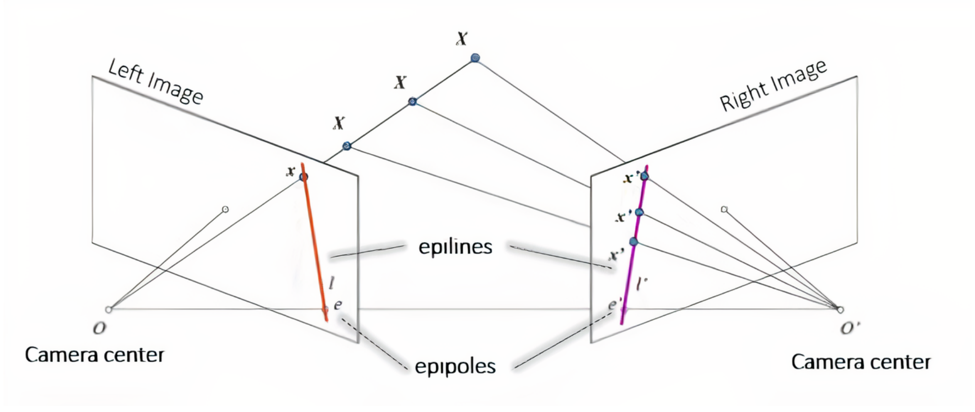


Figure 4.4 — The ideal epipolar constraint

and j, x is on frame i and $x^{'}$ is on frame j. Now, one can conclude that corresponding points seen by two views of the same 3D real point lie on conjugate epipolar lines. If the 3D real point $\boldsymbol{X}$ is static, it is satisfy the equation

$$p_2^T F p_1 = 0 \tag{4.30}$$

where F is fundamental matrix and p_1, p_2 are the the homogeneous coordinates of x and $x^{'}$ respectively. However, due to all the uncertainty of feature extraction and the estimation of F matrix, projected point $x^{'}$ will not always satisfy (4.30) but it will be very close to the epipolar line $l^{'}$. So that we can compute the distance $\boldsymbol{D}$ between $x^{'}$ and epipolar line $l^{'}$, if it greater than certain threshold then we mark it moving point and marginalize it.

4.5.2 Machine learning-based methods

Due to the real-time performance constraints, we have to choose the image semantic segmentation that light-weighted and meet this requirement, as mentioned in 1.4.3, SegNet is the valid candidate for this task. SegNet is the encoder-decoder semantic segmentation architecture illustrating in Fig. 4.5. The encoder uses a pre-trained VGG16 [26] model, including 13 convolution layers. After every 2 encoder layer, there exists a corresponding max-pooling to down-sample the image. The innovated of SegNet is in the decoder side, where instead of using deconvolution to upsample the image to original size, it stores the pooling indices in encoder side and recall it in decoder for upsampling. This helps reduce significantly trainable parameters

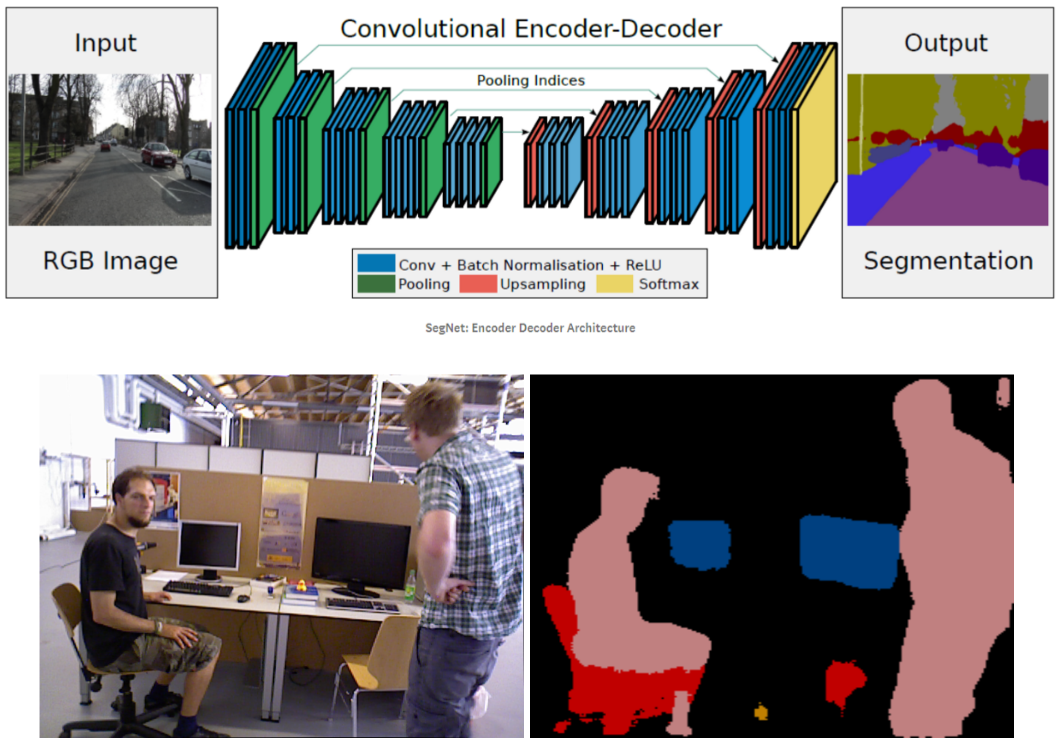

Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation
SegNet: Encoder Decoder Architecture

Figure 4.5 — SegNet - Encoder-Decoder semantic segmentation architecture

from 134M to 17.4M and less memory usages. In the following section we will discuss in more detail how we implement SegNet in our pipeline.

4.5.3 Pros and Cons

Geometry-based methods is only based on geometry constraints, so that it costs minor computation and allows the system work in real-time. Moreover, it can detected the moving objects that unable by Learning-based can not detect. However, it also have some limits

— They can not detect the potential moving objects that is currently not moving.

— They is not able to build that semantic maps which have rich information which can use for further application.

In this opposite, machine-learning method can detect the potential moving object and build meaningful map that is able to reuse for various tasks. However, on the downside, it takes a lot of time and memory to process and also requires the right hardware.

4.6 System Overview

The System Overview section aims to introduce our Visual-Inertial-Wheel Odometry system and implemented SegNet models to working in the high dynamic enviroment. Our system is built on ORB-SLAM3 [9], the system includes five main threads: Segmentation thread, Tracking thread, Local Mapping thread, Loop-closure & Map Merge thread and Atlas component.

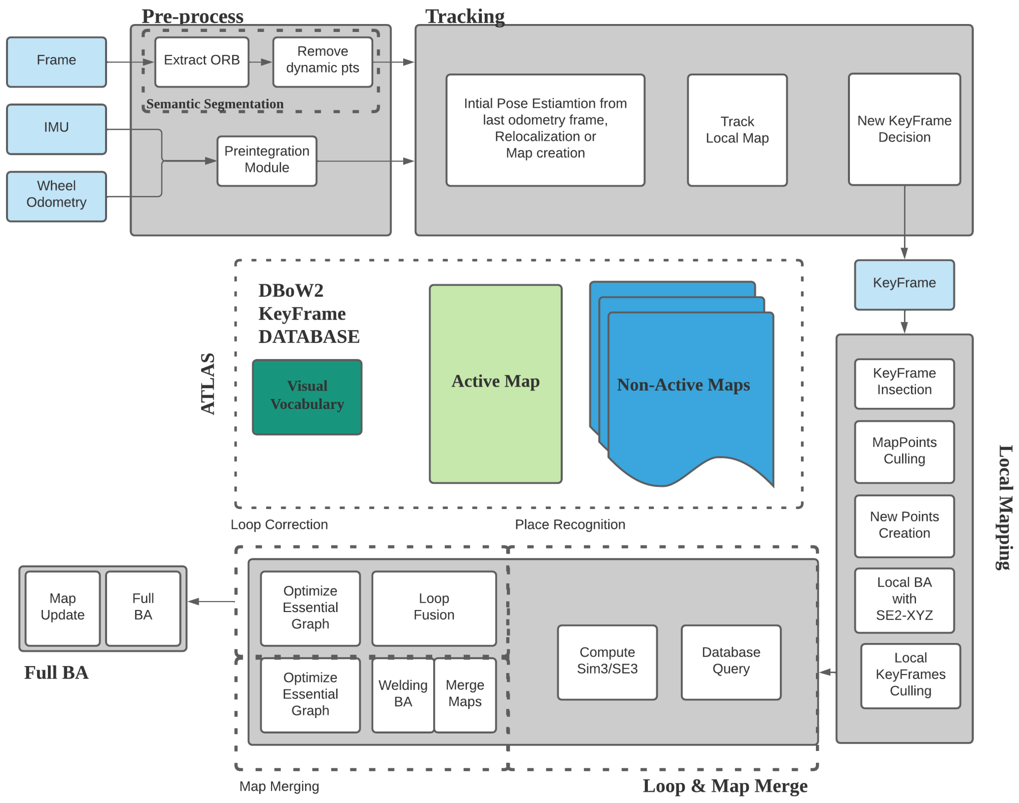


Figure 4.6 — The main components of our proposed system

4.6.1 Tracking thread

This is the main thread of the system; it is in charge of extracting ORB features in the new frame, but it will remove the features that lay on dynamic religion. Epipolar constraint also takes into account and removes violating features. In the monocular mode, we initialize odometric data to get the immediately correct scale; then we update the motion model of the body robot with preintegrated odometric. It also check the zero-velocity state, if current and last frame have enough matched and mean disparity is lower than threshold, it means that the robot is in static state, all next steps will be skipped, reference landmarks of current frame will be updated and system wait for new frame. After having the relative estimation of the robot pose, we optimize this estimation using the motion-only BA method to reduce the complexity of the problem to be optimized, i.e. in the optimal graph; the positions of the landmarks are fixed to reduce the complexity of optimize; we only optimize the robot pose and the state vector of the IMU. Indeed, the graph also includes factor IMU and Wheel Odometry preintegration. The tracking thread also plays the role of selecting the keyframe for the system. If only an optical sensor is used, the selection criteria are carefully considered through the tracking variables that have been presented in [30]. However, if using the IMU and wheel odometry sensors simultaneously, it is critical that we add the keyframe at small intervals to avoid erroneous estimation of motion sensor biases. Through the experiment, we can choose the interval to add keyframes when using the motion sensor, which is 0.5 seconds. These keyframes might be removed later if not needed in the following steps of the system. When the visual tracking lost, the system changes to RECENTLY_LOST-mode; in this mode, the system only use odometric to update camera pose and map points are projected in the estimated camera pose, then searched for matches within a large image window.

4.6.2 Segmentation thread

When this thread receives an incoming frame, it will frame the frame into regions, each of which belongs to an object; if this object has motion potential, all ORB features within this region will be marked as move and then discard. The Fig 4.7 shows how our segmentation thread works, where the green points indicate statics points using for tracking and the red points illustrate moving features points are removed by SegNet.

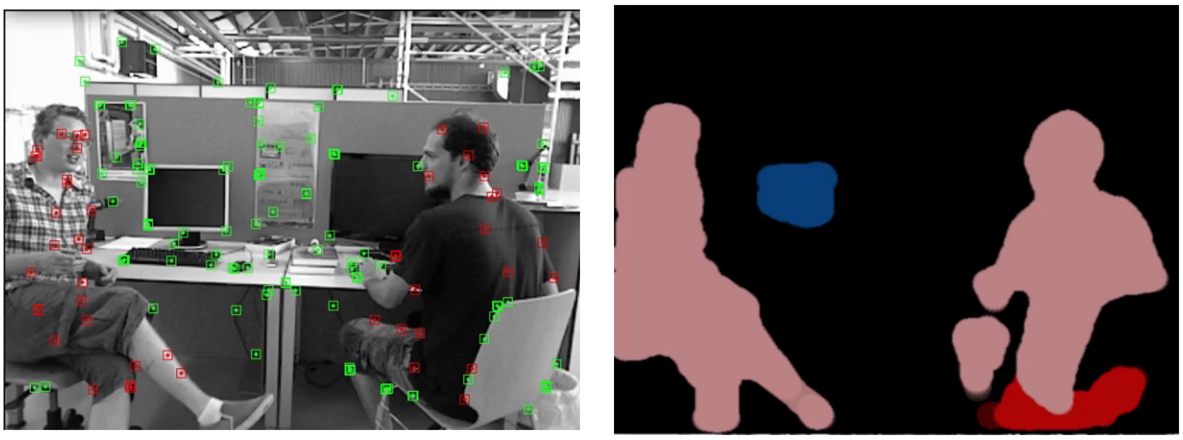


Figure 4.7 — Semantic Segmentation thread

4.6.3 Local Mapping thread

The Local Mapping thread assumes the role of handling the new keyframe created by the tracking thread. First, when a new keyframe is received, it will update the graphs related to covisibility and compute the bag of words. After keyframe preprocessing, this sequence performs the removal of outlier landmarks that are observed by less than 30% (used by [30]) of the frame in which they are intended to be observable it. At the same time, Local BA also marks any landmarks as outliers and they will also be removed in this step. Then new map point is created by triangulating matched between new keyframe and its neighbors, all the parameter e.g positive depths, reprojection error, epipolar constrains and sufficiency parallax have checked, if all have passed new map point will be created. The next step is to perform a Local BA. If only optical sensor is used, the local keyframe is built by the covisibility; this approach is superior than the sliding-window method since it helps to maximize the constancy and more information graph that leads to find the global minimize. When fusing with IMU and Wheel Odometry, the sliding-window method is used, since we need the consecutive keyframe for the preintegration factor. Difference with Motion-only BA in tracking thread; Local BA will try to optimize all the variables including robot pose, IMU's vector state and landmark position. This optimization is an expensive operation, but it runs in parallel threads so it will not affect real-time constrains in tracking thread. The keyframe culling step executes after Local BA, a keyframe that considers as redundant when 90% of the landmarks have been observed by it, also been seen in at least other three keyframes. If fusing IMU and wheel odometry, after keyframe is removed, we have to re-preintegration that from previous to the next keyframe of the keyframe that will be deleted. After this step local map will decide that should system uses VIW or VW fusion based on principle describing in 4.3.

4.6.4 Loop Closure & Map Merge thread

Loop closure is an essential part of the SLAM system. New keyframe after be processed in Local Mapping thread, be sent to this thread for updating the vocabulary tree in bag of words. This bag of words of new keyframe compares its neighbor and retains the most similar. Then we query the recognition database and discard all those keyframes whose score is lower than s_{min}. To accept a loop candidate, we must detect consecutively three loop candidates that are consistent (keyframes connected in the covisibility graph). There can be several loop candidates if there are several places with similar appearance to new keyframe. We then optimize the loop and fusing the duplicate landmark in $sim(3)$ [47]. Map merge works with same principle as loop-closure, it fusing two maps in Atlas component that has the appearance similarity.

4.6.5 Atlas Component

Atlas is a component inside the system that stores saved or building maps that the robot has visited. It contains active map which is the building map, in opposite of non-active maps. The Loop Closure & Map Merge thread actively searching for similarity between the active and non-active maps, if the merging candidate found between keyframe of active map and keyframe in one of non-active maps, map merging will be activated to merge and fused two map together, all duplicate landmark will be fused. Atlas allows system achieved the life-long mission of SLAM.

5 Evaluation

In this part, we are going to evaluation our propose system in the term of robustness in bad illumination condition and robustness in high dynamic enviroment. For the stated purpose, we use OpenLoris [44] datasets to evaluate the the robustness in bad illumination and TUM RGB-D [49] to evaluate the result when working in high dynamic environments. All the computation Absolute trajectory error (ATE), scale correction and illustration trajectory have done by [17]. We also test our system in real robot in laboratory's Youbot and Courier Robot at Sberbank Robotics Laboratory.

5.1 OpenLoris Datasets

The evaluation took part on OpenLoris [44], which was recorded from a wheeled robot moving at indoor environments. It contains a wide range of sensors, as showing in Table 5.1. The datasets provides five scenes:

Sensor	Data	Fps	resolution
D435i	color	30	848x480
D435i	depth	30	848x480
D435i	IMU	400	-
T265	Fisheye Stereo	30	848x480
Wheel encoder	odom	20	-

Table 5.1 — Sensors providing on OpenLoris datasets

— **Office** includes 7 sequences, recording in the same room, sequence 1 - 2 - 3 have potential merging to one maps, sequence 5 - 6 have low-light condition, sequence 7 introduces dynamic objects.

— **Corridor** contains 5 sequences, recording in corridor. This sense is most challenging, having not only change illumination, but also low-texture during the corridor, extreme high contrast, changing day-night time. The lobby in the middle of each sequence can potential use for map merge.

— **Cafe** has 2 sequences, taking from same public cafe shop with moving people and changing objects.

— **Home** records in 2 bedrooms apartment, which include may change objects arranges, it is a good challenging sequences for testing place recognition and map merging.

— **Market** includes 3 sequences in a supermarket with a long loop - 220 meters and high dynamics environments.

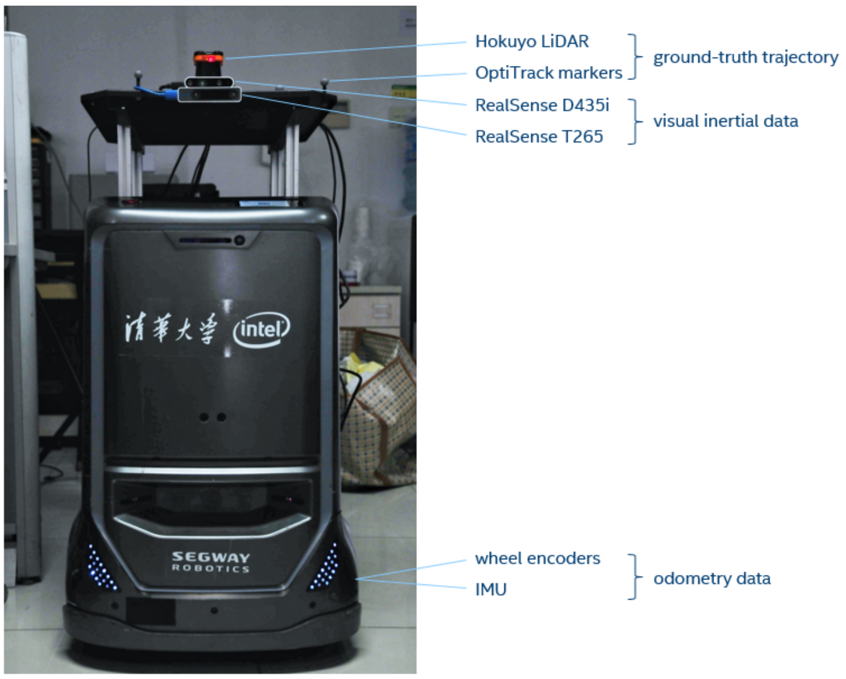


Figure 5.1 — Mobile Robot in OpenLoris datasets

The ground-truth from motion capture or offline Lidar SLAM is provided for all sequences.

In table 5.2, we see that the wheel odometry in the OpenLoris dataset is very good compared to the ground truth. In general, in most cases for wheeled mobile robots that operate indoors can be provided with very good wheel odometry.

For ORB-SLAM3, we show that purely optical SLAM is not suitable for real-life scenarios in terms of accuracy as well as robustness. It is clear that visual SLAM system fails in corridors sequences where visual conditions are hard (e.g. reflection, bad illumination) and in markets where there are moving objects in the scene.

Visual-Inertial ORB-SLAM3 shows better performance then visual one. ORB-SLAM3 shows stable and robust performance on different sequences except for market sequences. Usually, VINS systems are more robust

Sequence	Wheel Odometry	ORB-SLAM3 Visual	ORB-SLAM3 Visual Inertial	R-V(I)WO VIW (Ours) without switching	R-V(I)WO VW (ours) with switching	1st place
office1-1	0.040	0.132	0.109	0.081	**0.020**	0.172
office1-2	0.070	0.108	0.071	0.075	**0.026**	0.936
office1-3	0.043	0.195	-	0.137	**0.022**	0.732
office1-4	0.080	0.182	0.094	0.094	**0.048**	0.673
office1-5	0.090	0.234	0.235	0.233	**0.081**	0.515
office1-6	0.042	0.125	0.089	0.075	**0.024**	0.459
office1-7	0.072	0.093	0.069	0.087	**0.022**	0.854
Avg	0.062	0.153	0.111	0.112	**0.035**	0.620
home1-1	0.230	0.402	0.406	0.388	0.221	**0.172**
home1-2	0.314	0.345	0.363	0.350	0.299	**0.240**
home1-3	**0.110**	0.354	0.361	0.373	0.116	0.158
home1-4	0.095	0.310	0.351	0.341	**0.085**	0.171
home1-5	**0.066**	0.321	0.318	0.295	0.069	0.254
Avg	0.163	0.346	0.360	0.350	**0.158**	0.206
cafe1-1	0.236	0.177	0.116	**0.118**	0.213	0.232
cafe1-2	0.424	0.145	0.164	**0.194**	0.422	0.230
Avg	0.330	0.161	0.140	**0.156**	0.318	0.231
corridor1-1	1.948	-	1.817	1.647	1.831	**1.032**
corridor1-2	2.215	-	1.260	1.395	2.147	**0.675**
corridor1-3	0.222	-	0.990	0.919	**0.159**	1.320
corridor1-4	0.345	-	0.819	0.596	**0.235**	1.270
corridor1-5	0.640	2.109	1.131	0.705	**0.420**	0.964
Avg	1.074	2.109	1.204	**1.052**	1.139	**1.052**
market1-1	2.522	10.29	8.829	9.378	2.590	**1.073**
market1-2	5.398	-	11.86	12.378	5.351	**1.216**
market1-3	4.267	10.36	9.905	9.613	4.523	**1.432**
Avg	4.062	10.33	9.928	10.376	4.155	**1.240**

Table 5.2 — Performance comparison of the RMSE position (m) between different settings of ORB-SLAM3, R-V(I)WO with/without switching and the 1st place winner of the competition on OpenLoris Dataset

to moving objects, but the main problem is bad inertial initialization by planar movements. This affects the whole VINS system and its robustness.

We tested R-V(I)WO without the planar assumption switching technique on OpenLoris scenarios. The visual-inertial-wheel fusion on $SE3$ improves the accuracy of the system. It even outperforms the results of the competition's winner solution in several scenarios. On the other hand, we can see that the wheel odometry alone outperforms R-V(I)WO without switching. This is because of the bad effect of IMU initialization problems in planar motion, and because of complex optimization in $SE3$ space, which is over-parameterized.

After adding the switching ability based on the planar assumption detection, we show that in most cases R-V(I)WO with switching performs best. This proves how wheel odometry fusion and parameterizing the optimization in $SE2$ provide the wheeled ground mobile robot with accurate and robust localization and mapping system in different real-life scenarios.

We illustrated the quantity result comparison of R-V(I)WO with switching and ORB-SLAM3 in Fig. 5.2, in which we can clearly see the improvement and robustness of our system compare with ORB-SLAM3. Ad-

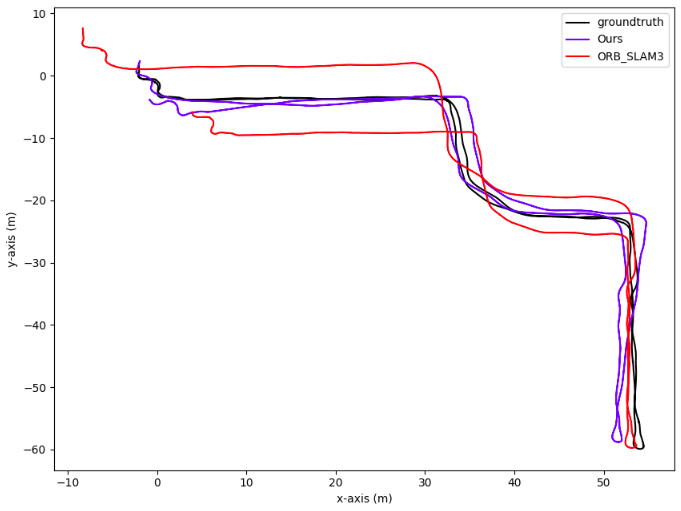


Figure 5.2 — Quantity comparison on corridor sequence

ditionally, we show that R-V(I)WO is optimized in terms of inference time. In table 5.3, we present the execution time of the tracking thread. It can run up to 50 fps on a moderate notebook, processing each frame in the tracking thread. For other threads, ORB-SLAM3 and R-V(I)WO execute them in parallel threads and only process the keyframes.

	ORB_SLAM3	R-V(I)WO
Features Extraction	21.76841± 7.62156	11.41350±2.92199
Stereo Matching	2.70659±0.59921	2.58476±0.54398
IMU Preintegration	0.13646±0.55224	0.11735±0.05763
Pose Prediction	0.13646±0.55224	0.09650±0.46915
Local Map Track	6.35867±2.39035	5.62523±1.59043
New KF decision	0.13486±0.26861	0.13499±0.25433
Total Tracking	34.34582±10.88638	22.97315±4.23357

Table 5.3 — Execution time (Milliseconds) comparison with ORB-SLAM3

5.2 TUM RGB-D Datasets

The TUM RGB-D is recorded in the typical office senses with people appear, during the dataset people are moving, walking, chatting and interacting with objects. It contains multiple sequences for each camera moving is moving in the typical direction: *xyz*, *static*, *rotation half sphere*, *roll-pitch-yaw*. By this character, this datasets is very useful for benchmarking dynamics SLAM system. TUM-RGBD is also suitable to compare with other models in the literature by robustness, accuracy, and execution cost.

We tested our modification of the ORB-SLAM3 system without the constrained optimization part, to see independently how well the model is performing.

This outliers' rejection model provides ORB-SLAM3 with robustness and accuracy in the case of tracking in a dynamic environment.

We show in table 5.4, the results of our model compared to ORB-SLAM3 and other works in the literature. It is shown that our model is robust over all the sequences we tested, and gets either the best or second best accuracy in most cases. We also show two trajectories of ORB-SLAM3 with/without our dynamic model (figure 5.3), and it is clear that ORB-S-

Sequence	Translation RPE (m/s)			
	ORB_SLAM3 (RGB-D)	DS-SLAM	Detect Slam	Ours (G+AI)
walking_xyz	1.251107	0.0333	**0.0241**	0.046710
walking_rpy	1.517069	0.1503	0.2959	**0.040565**
walking_half	1.055122	**0.0303**	0.0514	0.040534
walking_static	0.553423	**0.0102**	-	0.011880
sitting_xyz	**0.016737**	-	0.0201	**0.016970**
sitting_rpy	**0.031859**	-	-	0.032727
sitting_half	0.037480	-	**0.0231**	0.024585
sitting_static	0.015077	-	-	**0.009412**

Table 5.4 — Comparison of the RMS RPE in translation drift over TUM-RGBD dataset. Best results are highlighted in bold and the second-best are underlined.

LAM3 system is not robust to dynamic environments and it works better with our modifications.

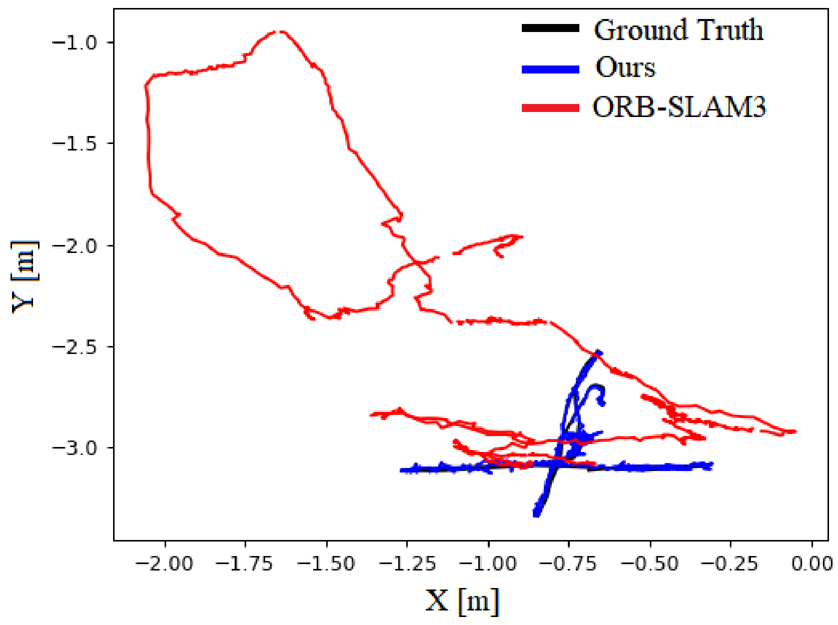


Figure 5.3 — Trajectory comparison on TUM-RGBD walking-XYZ sequence

We also show in table 5.5 that our modification preserve the real-time performance of the ORB-SLAM3 system and it runs as fast or faster than other models.

Methods	AI	Geometric	Tracking	Hardware
ORB SLAM3	-	-	**22.2322**	CPU only
DS-SLAM	75.64	47.38	148.53	Intel i7 CPU P4000 GPU
Dyna SLAM	884.24	589.72	1144.93	Titan X GPU
Detect_1 SLAM	310.0	**20.0**	-	Intel i7-470 GTX960M GPU
Ours	**69.3642**	26.6683	**22.837**	Intel i7 CPU RTX2060 GPU

Table 5.5 — Comparison of Computation Time [ms] on each module. The best results are highlighted in bold

5.3 Youbot Platform

5.3.1 Robot Platform

youBot [3] is a mobile robot designed and developed by German robot manufacturer Kuka. In addition to the movable body, youBot is coupled with a gripper with 5 degrees of freedom, but in our testing, this arm was not needed. The robot is actuated by 4 omnidirectional wheels. These special wheels have reels mounted around the circumference at a 45° angle to the plane of the wheel and allow youBot to combine any translational and rotational movements at any given time. This makes omnidirectional movement possible, including horizontal and diagonal movements.

Some general youBot specification:

— Overall Weight : $\sim$ 20 kg
— Permissible Payload : 20 kg
— Overall Length : 580 mm

— Overall Widht : 380 mm
— Overall Height : 140 mm
— Minimum velocity : 0.01 m/s
— Maximum velocity: 0.8 m/s
— Power supply: 24 V
— Communication: EtherCat

The robot's motor has a built-in encoder that will provide the robot position directly with a frequency of 300 Hz that we will use infusing with other sensors to estimate the position of the robot. The KUKA youBot is controlled by an onboard computer running a version of Ubuntu Linux. The youBot's onboard computer has many of the features of a standard computer, including a VGA port to connect an external monitor, several USB ports for connecting sensors and other peripherals, and an Ethernet port for connecting the youBot to a network.

In addition to the hardware manufactured and available at youBot, we need to design and manufacture specialized mounting components to place the necessary sensors for our navigation system. Aluminium industrial profiles are used to create a frame for the rack. The camera mount is custom

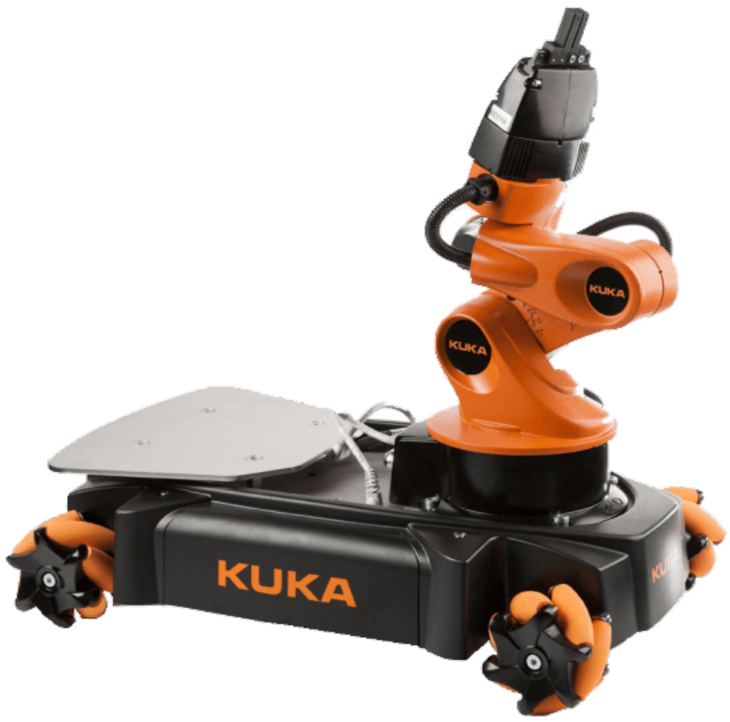
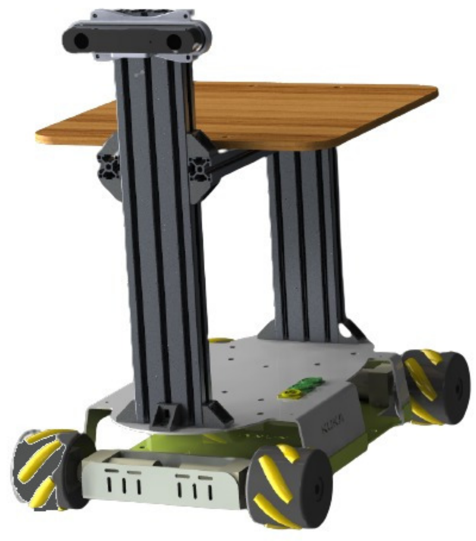


Figure 5.4 — Kuka youBot mobile robot

designed and manufactured using a 3D printer. The design and manufacture of the fixture are carried out at the ITMO Biomechatronics and Energy Efficient Robotics Laboratory.


(a) The designed youBot and mounted parts in Solidwork

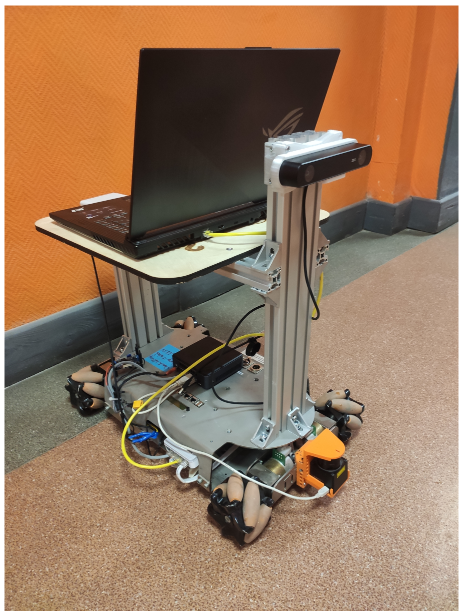

(b) The manufactured youBot with mounted camera and laptop

Figure 5.5 — The costumed youBot for our experiment

The Fig. 5.5a and Fig. 5.5b show the designed robot in Solidwork and the manufactured robot with mounting parts for the laptop and camera, respectively.

5.3.2 Visual-Inertial Sensor

ZED 2 AI Stereo Camera (Fig. 5.6) is our choice for optical sensors and inertial sensors. The choice is based on the fact that the ZED2 camera provides:

— Wide Field-of-Views stereo cameras: 120° Wide-Angle Field of View

— Neural Depth-Sensing: Bring the high-quality depth information by neural network

— Support: Spatial Object Detection opens new opportunities for improving our tracking and navigation system.

Moreover, the camera has done the factory optical calibration with the accurate optical parameters:

"All ZED 2, ZED and ZED Mini stereo cameras are calibrated at our factory using special equipment that guarantees highly accurate stereo and sensor calibration." [46]

Besides that, ZED 2 was also build-in with an Inertial Measurement Unit that gathers real-time synchronized inertial, elevation and magnetic field data along with image and depth, in addition, the measurement has been denoised and we can obtain a more accurate measurement from IMU in ZED 2 camera.

Figure 5.6 — ZED 2 AI Stereo Camera

Some general specifications about the optical sensor and IMU sensor in ZED 2 camera

— Stereo Field of View (Horizontal/Vertical): 120°.
— Maximum RGB Resolution: 4416 x 1242.
— RGB Fps: Up-to 100.
— Depth Technology: Neural Stereo Depth Sensing.
— Depth Resolution: Native video resolution (in Ultra mode).
— Depth Fps: Up to 100.
— Depth FOV: 110° (H) x 70° (V) x 120° (D) max.
— Inertial Unit: Gyroscope and Acceleration at 400 Hz.

5.3.3 Intrinsic Calibration

Optical Sensor Calibration

As stated in Section 5.3.2, we do not have to perform the optical calibration for ZED 2, instead of that, the parameters provided by company is used directly, those camera parameters (for 1280x720 res.) are following: Cameras model is *Pinhole* with intrinsic matrices K:

$$\text{Left Camera:} \quad \mathrm{K}_1 = \begin{bmatrix} 528.9899902343 & 0 & 633.989990234 \\ 0 & 528.510009765 & 347.914001464 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Right Camera:} \quad \mathrm{K}_2 = \begin{bmatrix} 529.6049804692 & 0 & 646.489990234 \\ 0 & 529.299987793 & 364.442504883 \\ 0 & 0 & 1 \end{bmatrix}$$

The distortion model is *Plumb Bob* [6] model: $D = [k_1\, k_2\, p_1\, p_2\, k_3]^\top$

$$\text{Left Camera:} \quad \mathrm{D}_1 = \begin{bmatrix} -0.0418732017279 \\ 0.0109836999327 \\ 0.000561256019864 \\ 2.17856995732\mathrm{e}-05 \\ -0.00519326981157 \end{bmatrix}$$

$$\text{Right Camera:} \quad \mathrm{D}_2 = \begin{bmatrix} -0.043696500361 \\ 0.0126504004002 \\ 0.000142247998156 \\ -0.00036192900734 \\ -0.00574736017734 \end{bmatrix}$$

Baseline between left and right sensor is $b = 120(mm)$

Inertial Intrinsic Calibration

In the task of IMU intrinsic Calibration, we try to estimate the *IMU Noise Model* which includes *Additive "White Noise"* and *Bias. IMU Noise Model* - The rapid fluctuations in the sensor signal are modelled heuristically with a zero-mean, independent, continuous-time white Gaussian noise process of strength σ_g for gyroscope and σ_a for acceleration. In other words, the higher is, the more "noisy" our measurements. In common IMU model, slow variations in the sensor bias are modelled with a "Brownian motion" process, also termed a "Wiener process", or "random walk" in discrete-time. Formally, this process is generated by integrating "white noise" of strength σ_{bg} (gyro) or σ_{ba} (accel). In general, there is 4 parameters that we need to estimate to model IMU noise (Table 5.6).

Parameter	Symbol	Units
Gyroscope "white noise"	σ_g	$\frac{\text{rad}}{s}\frac{1}{\sqrt{Hz}}$
Accelerometer "white noise"	σ_a	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$
Gyroscope "random walk"	σ_{bg}	$\frac{\text{rad}}{s^2}\frac{1}{\sqrt{Hz}}$
Accelerometer "random walk"	σ_{ba}	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$

Table 5.6 — Summary of the IMU noise model parameters

These parameters can be estimate by analysed *Allan standard deviation*. We used the open-source code project *kalibr* [1] [14] for estimate these parameters. In order to yield the good result, there are some requirement when we record the datasets:

— The datasets must contain at least 3 hours long of IMU data, the longer the more accurate result.

— During the recording dataset, IMU sensor must be static and parallel with flat surface.

— The enviroment should not have any significantly changing in term of temperature and pressure.

Following the instruction in the project page and recording the datasets that meet all above criteria, we are able to obtain all the necessary intrinsic pa-

[1] `https://github.com/ethz-asl/kalibr`

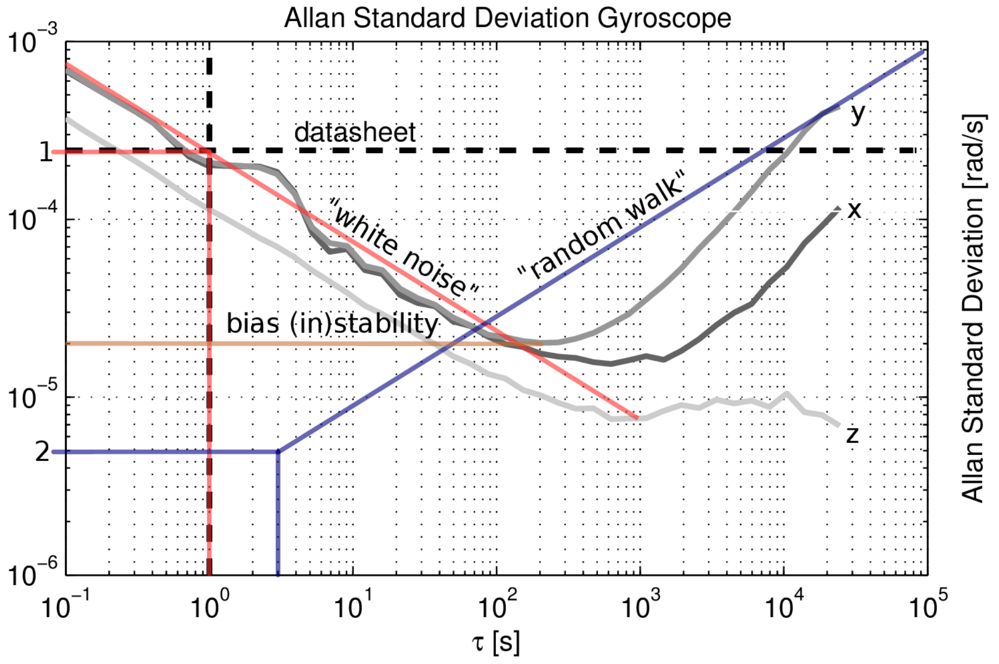


Figure 5.7 — Allan standard deviation of a MEMS gyro with manually identified noise processes

rameters for IMU that can work fine in our VIWO, the result as following

$$
\begin{aligned}
\sigma_g &= 0.000139626 \quad \left(\frac{\text{rad}}{s}\frac{1}{\sqrt{Hz}}\right)\\
\sigma_a &= 0.0016 \quad \left(\frac{m}{s^2}\frac{1}{\sqrt{Hz}}\right)\\
\sigma_{bg} &= 0.000033989 \quad \left(\frac{\text{rad}}{s^2}\frac{1}{\sqrt{Hz}}\right)\\
\sigma_{ba} &= 0.0002509 \quad \left(\frac{m}{s^3}\frac{1}{\sqrt{Hz}}\right)
\end{aligned}
$$

5.3.4 Extrinsic Calibration

Camera-IMU Extrinsic

We used directly the extrinsic parameters that provide in camera SDK:

$$
T_{bc} = \begin{bmatrix} 0.0052295 & 0.00167686 & 0.99998492 & 0.00187634 \\ -0.99998563 & -0.00119165 & 0.0052315 & 0.02301252 \\ 0.00120045 & -0.99999786 & 0.00167064 & 0.00197573 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

Wheel Odometric-IMU Extrinsic

We computed the transformation between wheel odometric-IMU and yields following results:

$$T_{bo} = \begin{bmatrix} 0.9998229886630 & -0.0129414007982680 & -0.0136569208171174 & -0.246310950318683 \\ 0.0129874331342 & 0.99991025598902 & 0.00328733152227729 & -0.0328630430685953 \\ 0.0136131525154 & -0.00346411797326174 & 0.999901336115347 & -0.530726569477215 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.3.5 Experiments

Sensor configuration

We configured the sensor properties so that it can provide sufficiently information for our system, as showing in Table 5.7 We will run the test on

Sensor	Data	Fps	resolution
RGB	color	30	1280x720
Inertial	gyroscope	400	-
Inertial	accelerometer	400	-
Wheel odometric	odometry	300	-

Table 5.7 — Sensors provided on our robot

two sequences

— **Office**: Robot operates on the office with good illumination condition, rich features.

— **Empty Room**: Robot run on the bad light condition, lack of feature.

The ground truth from lidar SLAM is provided for the empty room.

Evaluation on youBot

We will compare our works (Stereo-Inertial-Wheel Odometric) with the state-of-the-art visual-inertial ORB-SLAM3 [9] on the real-time running.

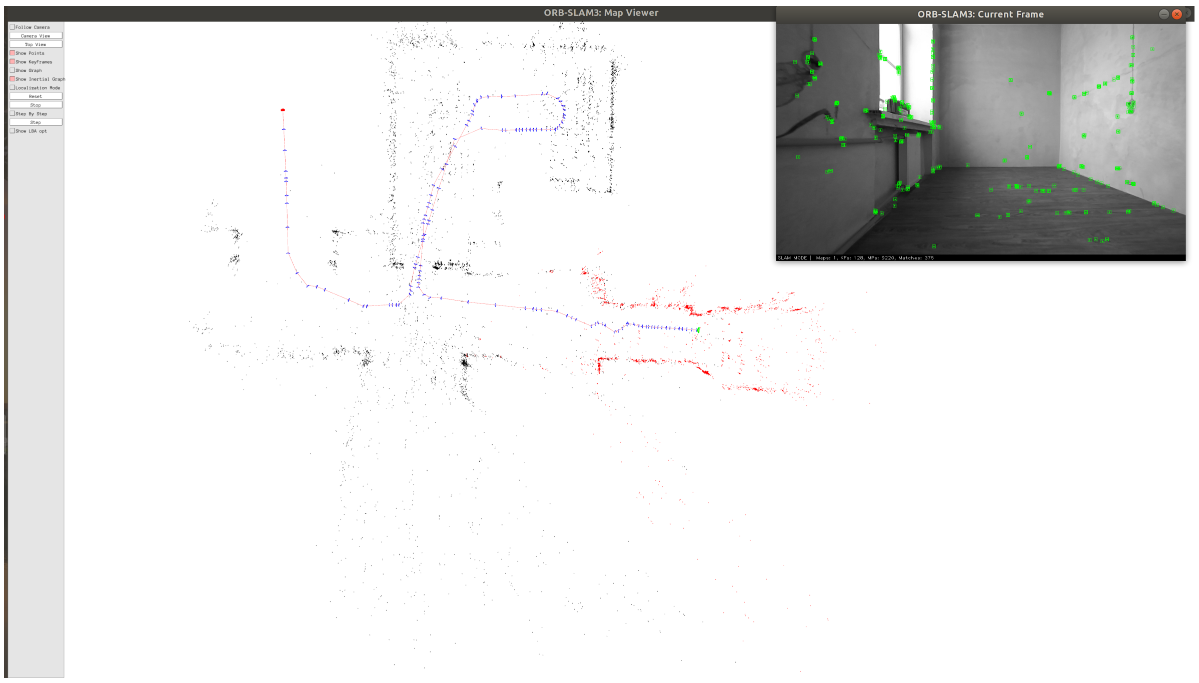


Figure 5.8 — Testing our navigation on real-equipment

The quantity comparison result is shown in Fig. 5.9. Observing at Fig. 5.9, we can see that the trajectory of our navigation system can locate the robot with the same accuracy as the lidar SLAM result, and the error over time is not as large as ORB-SLAM3 when observing at a separate XYZ graph. In particular, according to the Z axis, the error is very large over time even though the robot moves on the plane during the operation. However, the evaluation may contain errors because the accuracy of youBot's 2D lidar is not really high.

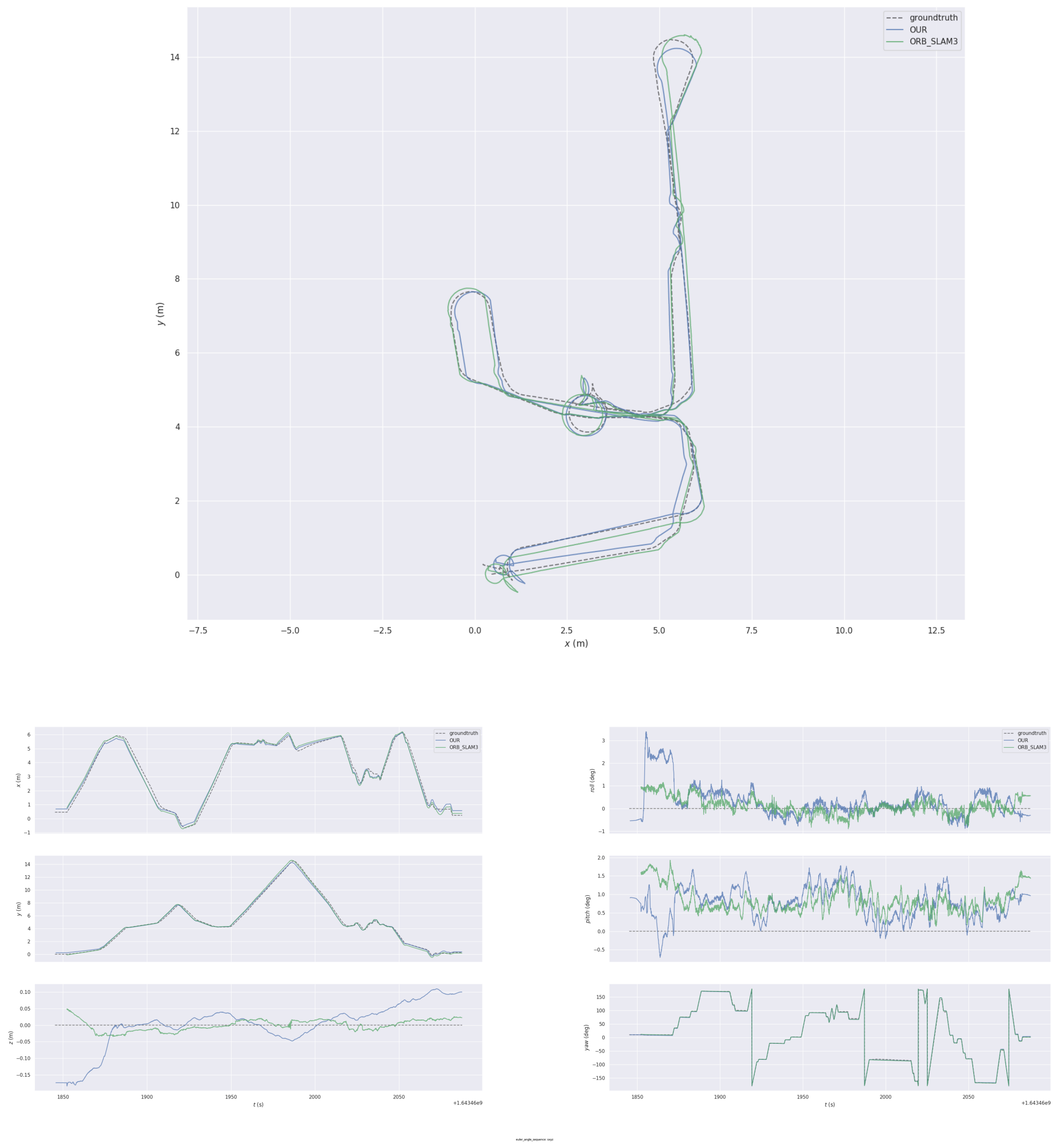


Figure 5.9 — Comparison trajectory with ORB-SLAM3

The Absolute Pose Error (APE) for ORB-SLAM3 is 35 cm and for OUR system is 30 cm.

5.4 Courier Robot

We also run the testing on real Courier in Fig 5.10.

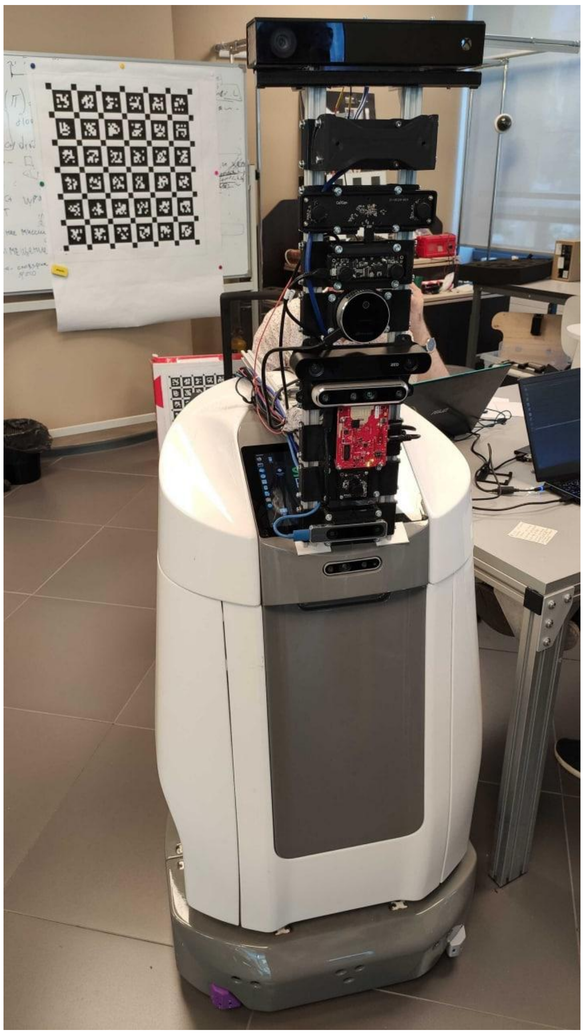


Figure 5.10 — Courier Mobile Robot in Sberbank Robotics Lab

The ground-truth is generated from ACML Localization on built map. We achieved the the Absolute Pose Difference with "ground-truth" is 25 cm. The trajectory comparision is presented in Fig. 5.11

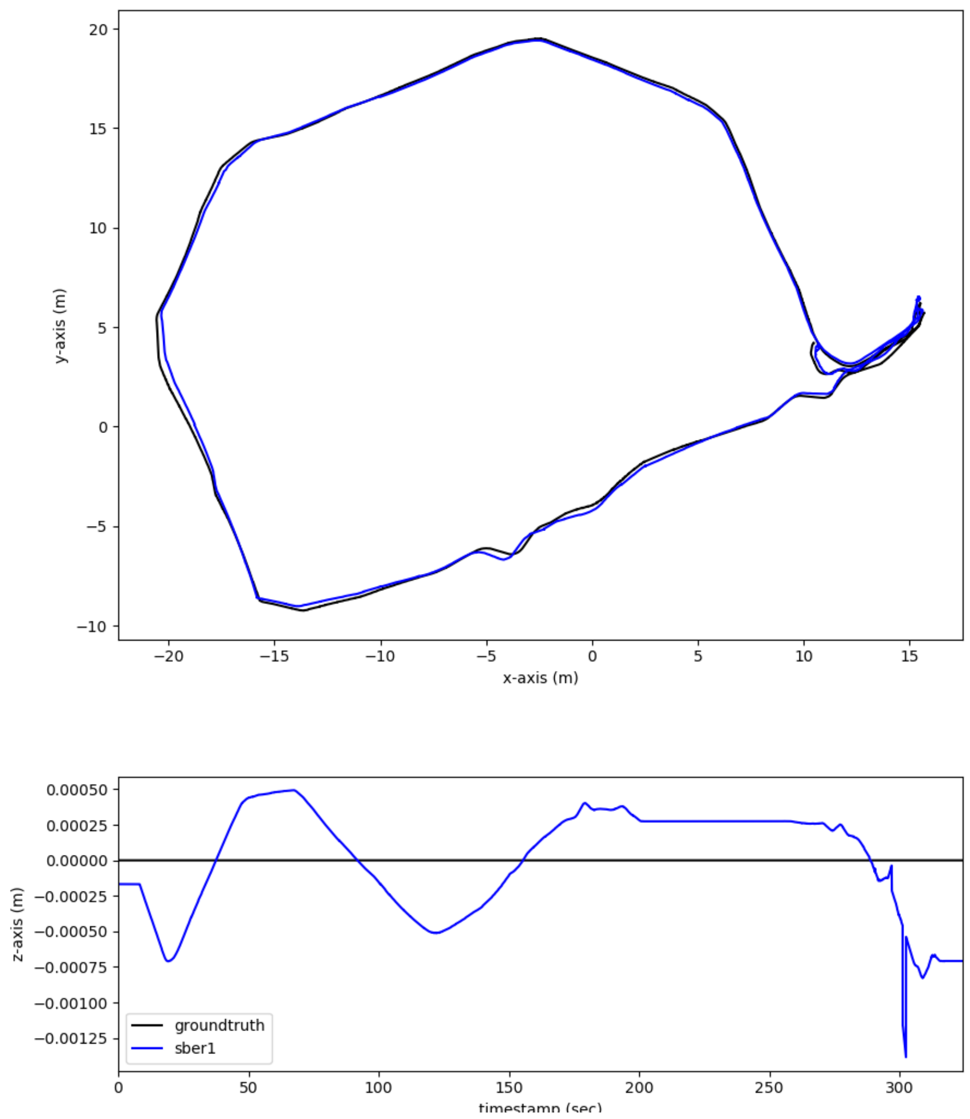


Figure 5.11 — Quantity comparison with AMCL Localization in Courier Robot

Conclusion

In this report, we presented in this work a SLAM system called R-V(I)WO, designed to work efficiently on real-life challenging scenarios for ground wheeled mobile robots. It is based on the ORB-SLAM3 system and uses a novel technique of switching between VIW and VW fusion modes. The experiment has shown that our system is superior performance in the task localization and mapping in dynamic and real-life scenario. The evaluation is conducted in public datasets and real robot platform in the laboratory. The system is built on the classical ORB feature that meet the real-time performance, however with the revolution of hardware, it opens many opportunity to implement more sophisticate type of feature and descriptor into SLAM system like handcraft-based SIFT, SURF or AI-based SuperPoint, etc. That can help to improve significantly the consistency and quality sparse map. Current system built sparse map which suitable for localization but indeed a dense map is more meaning full and can import to other works to build a dense and semantic map for high level and intelligence command navigation task. In the future works, we will focus to resolve the above remaining tasks and ready to bring visual SLAM into real world.

Bibliography

1. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: a deep convolutional encoder-decoder architecture for image segmentation, 2016.

2. Berta Bescos, Jose M. Facil, Javier Civera, and Jose Neira. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, Oct 2018. ISSN 2377-3774. doi: 10.1109/lra.2018.2860039. URL `http://dx.doi.org/10.1109/LRA.2018.2860039`.

3. Rainer Bischoff, Ulrich Huggenberger, and Erwin Prassler. Kuka youbot - a mobile manipulator for research and education. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011. doi: 10.1109/ICRA.2011.5980575.

4. Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Zürich, 2015. ETH-Zürich. doi: 10.3929/ethz-a-010566547.

5. Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017. doi: 10.1177/0278364917728574. URL `https://doi.org/10.1177/0278364917728574`.

6. Dean Brown. Decentering distortion of lenses. In *Photogrammetric Engineering*, volume 32, pages 444–462, 1966.

7. Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. doi: 10.1177/

0278364915620033. URL http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

8. Carlos Campos, J. M. M. Montiel, and Juan D. Tard'os. Inertial-only optimization for visual-inertial initialization. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 51–57, 2020.

9. Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M.M Montiel, and Juan D. Tardos. ORB-SLAM3: an accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 12 2021. ISSN 1941-0468. doi: 10.1109/tro.2021.3075644. URL http://dx.doi.org/10.1109/TRO.2021.3075644.

10. Nikolaus Demmel, David Schubert, Christiane Sommer, Daniel Cremers, and Vladyslav Usenko. Square root marginalization for sliding-window bundle adjustment. *CoRR*, abs/2109.02182, 2021. URL https://arxiv.org/abs/2109.02182.

11. J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP (99):1–1, 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2017.2658577.

12. Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 15–22, 2014. doi: 10.1109/ICRA.2014.6906584.

13. Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017. doi: 10.1109/TRO.2016.2597321.

14. Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286, 2013.

15. Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012. ISSN 1552-3098. doi: 10.1109/TRO.2012.2197158.

16. Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: a research platform for visual-inertial estimation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020. URL `https://github.com/rpng/open_vins`.

17. Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

18. R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. doi: 10.1109/34.601246.

19. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN, 2018.

20. Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. doi: 10.1109/ISMAR.2007.4538852.

21. Georg S. W. Klein and David William Murray. Improving the agility of keyframe-based SLAM. In *ECCV*, 2008.

22. Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: a general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi: 10.1109/ICRA.2011.5979949.

23. Stefan Leutenegger. *Unmanned Solar Airplanes. Design and Algorithms for Efficient and Robust Autonomous Operation.* PhD thesis, ETH Zurich, Zürich, 2014.

24. Stefan Leutenegger, Paul Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial SLAM using nonlinear optimization. *Proceedings of Robotics: Science and Systems*, June 2013. doi: 10.15607/RSS.2013.IX.037.

25. Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. doi: 10.1177/0278364914554813. URL `https://doi.org/10.1177/0278364914554813`.

26. Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015. doi: 10.1109/ACPR.2015.7486599.

27. Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

28. E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370, 2006. doi: 10.1109/CVPR.2006.236.

29. Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007. doi: 10.1109/ROBOT.2007.364024.

30. Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016. URL `http://arxiv.org/abs/1610.06475`.

31. Raul Mur-Artal and Juan D. Tardós. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters*, 2: 796–803, 2017.

32. Raul Mur-Artal, J. Montiel, and Juan Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147 – 1163, 10 2015. doi: 10.1109/TRO.2015.2463671.

33. Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669. IEEE, 2018.

34. Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: a robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

35. Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors, 2019.

36. Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

37. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

38. Arch D. Robison. *Intel® Threading Building Blocks (TBB)*, pages 955–964. Springer US, Boston, MA, 2011. ISBN 978-0-387-09766-4. doi: 10.1007/978-0-387-09766-4_51. URL `https://doi.org/10.1007/978-0-387-09766-4_51`.

39. Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization

and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. URL `https://github.com/MIT-SPARK/Kimera`.

40. Edward Rosten and Tom Drummond. Machine learning for high--speed corner detection. In *Comput Conf Comput Vis*, volume 3951, 07 2006. ISBN 978-3-540-33832-1. doi: 10.1007/11744023_34.

41. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

42. Martin Rünz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects, 2018.

43. D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. In *International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

44. Xuesong Shi, Dongjiang Li, Pengpeng Zhao, Qinbin Tian, Yuxin Tian, Qiwei Long, Chunhao Zhu, Jingwei Song, Fei Qiao, Le Song, Yangquan Guo, Zhigang Wang, Yimin Zhang, Baoxing Qin, Wei Yang, Fangshi Wang, Rosa H. M. Chan, and Qi She. Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM. In *2020 International Conference on Robotics and Automation (ICRA)*, pages 3139–3145, 2020.

45. R. Smith, M. Self, and P. Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics*, page 167–193. Springer-Verlag, Berlin, Heidelberg, 1990. ISBN 0387972404.

46. StereoLabs. How do i recalibrate my zed stereo camera?, 2021. URL `https://support.stereolabs.com/hc/en-us/articles/360011828773-How-do-I-recalibrate-my-ZED-stereo-camera-`.

47. Hauke Strasdat, J. Montiel, and Andrew Davison. Scale drift-aware large scale monocular slam. *Rob. Sci. Syst.*, 2, 06 2010. doi: 10.15607/RSS.2010.VI.010.

48. Hauke Strasdat, J. Montiel, and Andrew Davison. Real-time monocular slam: Why filter? *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2657–2664, 05 2010. doi: 10.1109/ROBOT.2010.5509636.

49. J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

50. C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. Shape and motion from image streams. School of Computer Science, Carnegie Mellon Univ., 1991. URL `https://books.google.ru/books?id=20wpSQAACAAJ`.

51. Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, 2020. doi: 10.1109/LRA.2019.2961227.

52. L. von Stumberg and D. Cremers. DM-VIO: Delayed marginalization visual-inertial odometry. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1408–1415, 2022. doi: 10.1109/LRA.2021.3140129.

53. P. Wenzel, R. Wang, N. Yang, Q. Cheng, Q. Khan, L. von Stumberg, N. Zeller, and D. Cremers. 4Seasons: A cross-season dataset for multi-weather SLAM in autonomous driving. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020.

54. Kejian Wu, Chao X. Guo, Georgios A. Georgiou, and Stergios I. Roumeliotis. VINS on wheels. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162, 2017.

55. Dongsheng Yang, Shusheng Bi, Wei Wang, Chang Yuan, Wei Wang, Xianyu Qi, and Yueri Cai. DRE-SLAM: dynamic rgb-d encoder slam for a differential-drive robot. *Remote Sensing*, 11(4), 2019. ISSN 2072-4292. doi: 10.3390/rs11040380. URL `https://www.mdpi.com/2072-4292/11/4/380`.

56. Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. DS-SLAM: A semantic visual SLAM towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2018. doi: 10.1109/iros.2018.8593691. URL `https://doi.org/10.1109%2Firos.2018.8593691`.

57. Liu Yubao and Jun Miura. RDS-SLAM: real-time dynamic slam using semantic segmentation methods. *IEEE Access*, PP:1–1, 01 2021. doi: 10.1109/ACCESS.2021.3050617.

58. Fan Zheng and Yun-Hui Liu. Visual-Odometric Localization and Mapping for Ground Vehicles Using SE(2)-XYZ Constraints. In *Proc. IEEE Int. Conf. Robot. Autom (ICRA)*, 2019.

59. Fangwei Zhong, Sheng Wang, Ziqi Zhang, China Chen, and Yizhou Wang. Detect-SLAM: Making object detection and slam mutually beneficial. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1001–1010, 2018. doi: 10.1109/WACV.2018.00115.