2023

Portable Robotic Navigation Aid for the Visually Impaired

Lingqiu Jin
Virginia Commonwealth University

PORTABLE ROBOTIC NAVIGATION AID FOR THE VISUALLY IMPAIRED

A submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

LINGQIU JIN

Virginia Commonwealth University - September 2017 to August 2023

Advisor: Dr. Cang Ye,

Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

August, 2023

Acknowledgements

First of all, I would like to thank Dr. Cang Ye, my Ph.D. advisor. I could not have completed my research in my Ph.D. program without his advice, dedication, and great patience. I would also like to thank my Ph.D. committee members: Dr. Changqin Luo, Dr. Dianne Pawluk, Dr. John Leonard, and Dr. Weijun Xiao.

I would like to thank Dr. He Zhang. He is more than a senior but a close friend and excellent teacher. He teaches me how to do research and be a good person.

I would like to thank my friends, Chunyi Guo, Guang Yang, Meihan Hu, Tianlan Cao, Wei Yang, and Yu'an Xiao. I cannot make the datasets and test the devices without their help.

I would like to thank my family. They continuously supported my life and encouraged me to complete the Ph.D. program.

I would like to thank my girlfriend, Wei Xu. She cheered me up during the most challenging time and brought light and happiness into my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

PORTABLE ROBOTIC NAVIGATION AID FOR THE VISUALLY IMPAIRED

By Lingqiu Jin

A submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2023.

Advisor: Dr. Cang Ye,

Professor, Department of Computer Science

This dissertation aims to address the limitations of existing visual-inertial (VI) SLAM methods - lack of needed robustness and accuracy - for assistive navigation in a large indoor space. Several improvements are made to existing SLAM technology, and the improved methods are used to enable two robotic assistive devices, a robot cane, and a robotic object manipulation aid, for the visually impaired for assistive wayfinding and object detection/grasping. First, depth measurements are incorporated into the optimization process for device pose estimation to improve the success rate of VI SLAM's initialization and reduce scale drift. The improved method, called depth-enhanced visual-inertial odometry (DVIO), initializes itself immediately as the environment's metric scale can be derived from the depth data. Second, a hybrid PnP (perspective n-point) method is introduced for a more accurate estimation of the pose change between two camera frames by using the 3D data from both frames. Third, to implement DVIO on a smartphone with variable camera intrinsic parameters (CIP), a method called CIP-VMobile is devised to simultaneously estimate the intrinsic parameters and motion states of the camera. CIP-VMobile estimates in real

time the CIP, which varies with the smartphone's pose due to the camera's optical image stabilization mechanism, resulting in more accurate device pose estimates. Various experiments are performed to validate the VI-SLAM methods with the two robotic assistive devices.

Beyond these primary objectives, SM-SLAM is proposed as a potential extension for the existing SLAM methods in dynamic environments. This forward-looking exploration is premised on the potential that incorporating dynamic object detection capabilities in the front-end could improve SLAM's overall accuracy and robustness. Various experiments have been conducted to validate the efficacy of this newly proposed method, using both public and self-collected datasets. The results obtained substantiate the viability of this innovation, leaving a deeper investigation for future work.

CHAPTER 1

INTRODUCTION

1.1 Background

There are approximately 253 million people with visual impairment worldwide [1]. Among them, about 36 million are blind. Vision loss limits their mobility and deteriorates their quality of life. Currently, white cane, a century-old tool, is the prevailing mobility aid that the blind and visually impaired (BVI) use for their daily travel. Only about 2% of the BVI use a guide dog for their mobility need. These people are also white cane users. Compared with the white cane, a guide dog is costly to train and own and requires the BVI to take care of it. On the other hand, a white cane is not an effective mobility tool in some cases because it cannot detect above-the-waist obstacles and lacks a positioning function (locating the user in the environment). The first shortcoming may cause safety problems for the BVI. The second shortcoming makes it impossible for the BVI to get to the destination when navigating indoors, as indoor space is GPS-denied.

As the population continues to age, the number of BVI people will grow [2]. Therefore, there is a dire need to develop new assistive wayfinding technology to help the BVI with their independent mobility and improve their quality of life. To satisfy this unmet need, this research aims to develop an assistive navigation tool called RoboCane (RC) and the enabling technology to allow it for assistive wayfinding.

1.2 Research Problems, Proposed Solutions, and Contributions

The problem of independent mobility (aka assistive navigation) of a BVI individual includes wayfinding and obstacle avoidance. Wayfinding is a global problem of planning and following a path toward the destination, while obstacle avoidance is a local problem of taking steps without colliding, tripping, or falling. To provide wayfinding and obstacle avoidance functions to a BVI traveler at the same time, the location of the traveler and the 3D map of the surroundings must be accurately acquired. The technique to address the problem is called simultaneous localization and mapping (SLAM). In the literature, Visual SLAM and visual-inertial SLAM techniques have been employed for assistive navigation. Visual SLAM estimates the poses of a moving camera by tracking image features or appearance across image frames. However, it only works well in a feature-rich environment. Visual-inertial SLAM overcomes this shortcoming by pairing an inertial measurement unit (IMU) with a visual camera and estimating the camera's pose by processing both the visual and inertial data. Numerous visual-inertial SLAM algorithms have been proposed in the past few years. Some of them have been validated with remarkable results on public datasets, such as TUM [3], KITTI [4], and EuRoC [5]. However, adapting these algorithms to assistive navigation is still an open question. This dissertation aims to improve the accuracy of visual-inertial SLAM under the condition of limited computing resources. Three state-of-the-art visual-inertial SLAM methods, OKVIS [6], VI-ORB [7], and VINS-Mono [8], were selected and compared by using data collected from typical wayfinding scenarios. The results revealed that VINS-Mono outperformed the other two. Therefore, it was chosen as the foundation of this dissertation, and two enhancements were made to improve its accuracy for assistive navigation.

The first is to use a depth sensor to improve visual-inertial SLAM. A visual-

inertial navigation system (VINS) must be sufficiently excited for its initialization. However, this requirement may not always be satisfied in a wayfinding scenario. In addition, a VINS may suffer from scale drift due to the lack of direct measurement of the scale (the size of the environment). To tackle these problems, a depth-enhanced visual-inertial odometry (DVIO) method was proposed. DVIO uses an RGB-D camera and an IMU for pose estimation. It uses the RGB-D camera's depth data to initialize the system and closely couples the depth data with the VINS-Mono framework to estimate the camera's pose, resulting in a more accurate and robust SLAM system. The proposed DVIO was extended into a visual-LiDAR-inertial odometry (VLIO) on an iPhone-based assistive navigation system, which eliminates the need for a dedicated sensing and computing platform. The much more accurate and reliable depth measurements of the iPhone's (12 Pro and later) LiDAR result in a better SLAM system. To support algorithm development, a dataset called "VCU-RVI Benchmark" [9] was collected and made available online. The indoor dataset is particularly designed for the evaluation of VIO and DVIO methods. It consists of thirty-nine data sequences covering indoor scenarios, such as laboratory, corridor, stairway, and atrium.

The second is to integrate online camera intrinsic parameters (CIP) estimation with VIO for a smartphone-based wayfinding system. Modern smartphones use an optical image stabilization (OIS) mechanism to reduce hand-tremor-induced image blur. The mechanism, however, does not respond to low-frequency motion and thus causes CIP variation, which may degrade pose estimation accuracy. To mitigate this effect, a method called CIP-VMobile was developed to simultaneously estimate CIP and VIO motion states. The method first uses a linear model that relates CIP with IMU-measured camera acceleration to determine the initial CIP and then incorporates the CIP into a graph structure to fine-tune the CIP values and estimate the

VIO motion state. CIP-VMobile was validated in terms of VIO (with a smartphone without a depth camera) and VLIO (with an iPhone with a LiDAR).

In this research, the RoboCane (RC) was used as a main platform to validate the proposed SLAM methods. Several versions of the RC were fabricated to support the work. In addition, a wearable device named Wearable Robotic Object Manipulation Aid (W-ROMA) was developed and used to test the proposed algorithms. W-ROMA can be viewed as a special assistive navigation device, which assists the user in finding a collision-free path for the hand to reach and take a hold of the target object.

The dissertation is organized as follows. Chapter 2 provides a thorough literature review on existing robotic navigation aid (RNA) and visual and visual-inertial SLAM technology. Chapter 3 introduces the RNAs proposed for this dissertation. Chapter 4 describes the sensors used on the RNAs. Chapter 5 presents pose change estimation methods based on visual data and that based on inertial data. Chapter 6 compares several state-of-art visual-inertial SLAM methods to justify the selection of the method for this research. Chapters 7, 8, and 9 present the proposed DVIO, VLIO, and CIP-VMobile algorithms, respectively. Finally, Chapter 10 concludes this dissertation and discusses future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature in Robotic Assistive Devices (RAD)

A navigation aid is able to keep the BVI user away from the obstacles in the path and at the same time make the user aware of their position and orientation. Ultrasonic sensors are widely used on RNA for obstacle detection. An ultrasonic sensor measures the distance to an object based on the time-of-flight of the ultrasonic wave it emitted. A number of cane-shaped RNAs [10], [11], [12], [13], [14], [15] have been developed using ultrasonic sensors. Other range sensors, including infrared [16], [17], and laser [18], [19] have also been used in this type of RNAs. Due to the limited sensing capability, these range sensors can only provide very limited object/obstacle information. More importantly, they cannot provide device pose information and thus the location of the BVI for wayfinding. While for most outdoor applications, GPS can be easily accessed and used for localization, navigation is still a challenging task for GPS-denied, especially indoor scenarios. Low-energy beacons [20], radio-frequency identifications [21], and near-field communication tags [22] have been employed for positioning to assist with wayfinding in indoor environments. However, the approach requires re-engineering the environment and is thus not practical for most assistive navigation scenarios.

Recently, cameras have been more commonly used for assistive navigation due to their low cost and being able to provide rich information about the environment. In the literature, several RNAs [23], [24], [25], [26], [27], [28], and [29] that use cameras for assistive navigation of the blind have been introduced. Among these

RNAs, [23] and [24] use a stereo camera, and [25], [26], [27], [28], and [29] use an RGB-D camera as the perception sensor. In either case, visual data (or visual + depth data) are processed to estimate the device's pose, based on which the user's location is determined for wayfinding. A 3D point cloud map can be generated and used for object/obstacle detection. The technology for pose estimation is called visual simultaneous localization and mapping (SLAM), or vSLAM in short.

2.2 Literature in vSLAM

In vSLAM, the camera pose change between two image frames can be estimated by analyzing the disparity between the frames. With a sequence of consecutive image frames, the estimated pose changes can be translated into camera pose estimates. The technique is also called visual odometry (VO). As an incremental method, VO accumulates pose estimation errors over time, and the error may cause the navigation system to malfunction. Therefore, it is crucial to develop methods to reduce/eliminate the accumulative pose estimation error.

Based on how the disparity between two images is analyzed, the vSLAM can be classified as one of the four categories: indirect sparse, direct sparse, indirect dense, or direct dense methods.

A sparse method extracts and tracks a subset of the image pixels (like corner points) called visual features (or feature points), while a dense method attempts to extract information from all image pixels. The former needs a visual feature detector, such as SIFT [30] or ORB [31], to determine the pixels that are going to be used for tracking. Feature extraction takes much more time than feature tracking.

Direct or indirect methods can be distinguished by their residual models. Direct approaches aim to minimize photometric errors, while indirect approaches attempt to minimize the geometric reprojection errors of visual features. Direct methods

track pixels across the frames based on intensity (color, brightness) gradients, while indirect approaches extract and track features based on their descriptors. In general, indirect methods are more robust to lighting change but consume more computational resources, while direct methods are more robust to fast camera motion but more likely to fail with illumination changes.

2.3 Literature in Visual-Inertial SLAM (VI-SLAM)

In the past decades, many vSLAM methods have been proposed, including ORB-SLAM (indirect sparse) [32], DSO (direct sparse) [33], Voldor (indirect dense) [34], and LSD-SLAM (direct dense) [35], each of which is the state-of-the-art technique of its category. Due to their incremental pose estimation nature, all of these methods accumulate pose error over time. Although a loop closure, if detected, can be used to eliminate the error, not all assistive navigation tasks incur a loop closure, or the accumulative pose estimation error may be too big by the time of a loop closure. Therefore, it is critical to reduce the speed of error growth in a vSLAM process. Also, a vSLAM method degrades its accuracy in cases where the environment is not feature-rich enough. It may fail if the environment is featureless.

To overcome the issue and improve the reliability/accuracy, an inertial measurement unit (IMU) is added to the SLAM system, and both the visual and IMU data are processed for pose estimation [7], [8], [36], [37], [38], [39]. As IMU can provide accurate short-term pose change estimation, such a visual-inertial navigation system (VINS) is more robust to a feature-sparse environment and drastic camera motion. The method for camera pose estimation of such a VINS is called Visual-Inertial SLAM (VI-SLAM) or Visual-Inertial Odometry (VIO), which is becoming increasingly popular in the robotics communities.

Existing VIO methods can be classified into two categories, namely loosely-

coupled [36], [37], and tightly-coupled [7], [8], [38], [39]. A loosely-coupled method usually has an independent vision-only pose estimator [32], [33], [34], [35], and a standalone filtering or batch optimization process to fuse IMU measurements. Weiss et al. [36] propose to integrate the visual-based and IMU-based pose estimations by using an extended Kalman filter (EKF), where the pose change estimated by the IMU is used for state prediction, and the vSLAM-estimated pose is used for state update. Indelman et al. [37] propose to use a factor graph to integrate the pose estimated by a vision-only estimator, with that estimated by integrating the IMU data, and estimate the pose by graph optimization. Such an approach ignores the correlations between the states of the camera and the IMU, yielding a sub-optimal solution. On the contrary, a tightly-coupled method jointly estimates the states of all sensors by using their raw measurements.

Related works on tightly-coupled VIO approaches can be further categorized into two categories, filter-based [36], [38] and optimization-based (also called smoothing-based) [7], [8], [39]. MSCKF [38] is an extended Kalman filter (EKF) based VI-SLAM method. It utilizes IMU measurements to predict the state and compute the predicted measurements of the visual features, and it uses the actual visual feature measurements to update the state vector. Unlike a traditional EKF, it simultaneously updates multiple camera poses (each of which is a part of the state vector) using a novel measurement model for the visual features. This model estimates a visual feature's 3D location by using its multi-view geometric constraint, computes the feature's reprojection residuals on multiple images, and uses the visual features residuals to update the state vector. The method adopts a delayed state update strategy to get the most from the multi-view constraint, i.e., a tracked visual feature is used to update the state vector only when it is no longer detected. In so doing, it uses much fewer visual features for state estimation as those features that are currently tracked are not used.

On the contrary, a smoothing-based VIO method [7], [8], [39] estimates all state vectors (within a sliding window) by using all visual and inertial measurements related to the states, achieving a more accurate result in general. For instance, OKVIS [39] finds an optimal set of state vectors that minimizes a cost function formulated as the weighted sum of the residuals of the visual features' reprojections and the inertial measurements. OKVIS performs well on a stereo-camera-based visual-inertial navigation system (VINS). However, its performance may significantly degrade in cases where a monocular camera is used. This is because the method lacks a reliable approach to accurately estimating the initial values of the state variables (e.g., gyroscope bias, metric scale, etc.). Due to the non-convexity of the cost function, an improper initial state estimate would most likely make the optimization process converge to a local minimum that may result in an unacceptable pose estimation error. To mitigate this issue, VI-ORB [7] implements a sophisticated sensor fusion procedure to bootstrap a monocular VINS with a more accurate initial state estimate, including the pose, velocity, 3D feature locations, gravity vector, metric scale, gyroscope bias, and accelerometer bias. However, it requires a 15-second process to collect visual-inertial data with adequate acceleration (IMU excitation). It is inappropriate for applications that require an accurate scale estimate right at the beginning. Qin et al. [8] discover that the metric scale error is linearly dependent upon the accelerometer bias, and it requires a long duration of sensor data collection to estimate the scale and the accelerometer bias simultaneously. To overcome the problem, they propose the VINS-Mono [8], where the initialization process is simplified by ignoring the accelerometer bias. The method uses a two-step approach to initializing the VINS' motion state. First, it builds a scale-dependent 3D structure with a visual-only structure-from-motion (SFM) method. Second, it aligns the IMU integration with the visual-only structure to recover the scale, gravity, velocity, and gyroscope

bias. This initialization approach converges much faster (in ∼100 ms) with a negligible accuracy loss, enabling VINS-Mono to perform well even if the VINS starts with smaller IMU excitation and/or non-trivial initial IMU bias. However, VINS-Mono [8] still falls short of real-time computing performance on a computer with limited computing power [40].

Although it was claimed that these state-of-the-art VI-SLAM methods achieved superior accuracy and robustness, their performances are case/application dependent. Therefore, it is necessary to investigate their performance in the context of assistive navigation. To this end, three state-of-the-art graph-based visual-inertial SLAM-OKVIS [39], VI-ORB [7], and VINS-Mono [8], were further analyzed and compared for assistive navigation with the RNA (detailed in Chapter 6). Based on the results, the best method was selected as the framework and was extended by improving its pose estimation accuracy to meet the need for assistive navigation.

CHAPTER 3

ROBOTIC NAVIGATION AID (RNA)

3.1 RoboCane (RC)

Several versions of RoboCane (RC) were fabricated and used during the project period developed for data collection and algorithm evaluation. In general, an RC consists of a cane-fitted computer vision system and an active rolling tip (ART) [41]. The former comprises a sensor suite (for SLAM and object/obstacle detection), a computer, and a power supply (battery), while the latter, if activated, may steer the RC into the desired direction of travel to guide the movement of the BVI traveler.

3.1.1 RoboCane (RC) using a UP Board

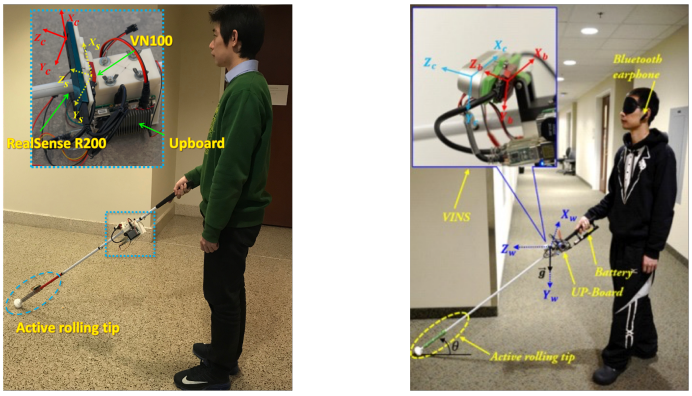


Fig. 1. RC prototype V1 (left) and V2 (right)

As depicted in Figure 1, the first two versions of RC prototypes use a sensor-suite consisting of an Intel RealSense RGB-D camera and an IMU (VN100 of VectorNav Technologies), and an Up Board computer to form its navigation system. The first

version uses a RealSense R200, and the second version uses a RealSense D435 (which is more accurate in depth measurement) is used. The camera is mounted on the cane with a 25° tilt-up angle to keep the cane's body out of the camera's field of view. This prototype and its successors use an ART [41] to steer the cane to the desired direction of travel to guide the BVI user to move toward the destination.

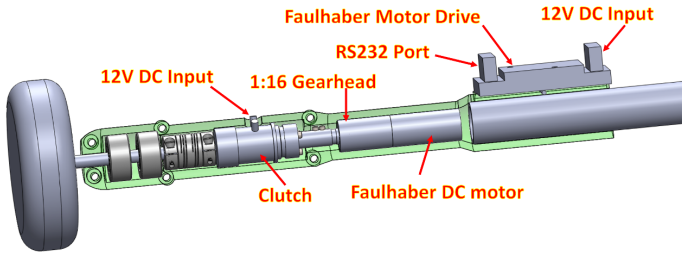


Fig. 2. Active Rolling Tip (ART)

As depicted in Figure 2, the ART consists of a rolling tip, a gear motor, a motor drive, and a clutch. A custom control board is built to engage and disengage the clutch. When the clutch is engaged, the motor drives the rolling tip and steers the cane. When it is disengaged, the rolling tip is disconnected from the gear motor, and the user can swing the RC just like using a white cane. A UP Board computer [42] is used to process the RGBD camera's and IMU's data and generate navigation commands to control the ART.

3.1.2 RoboCane (RC) using an iPhone

As depicted in Figure 3, the latest two versions of RCs use an iPhone as its sensing and computer platform and thus eliminate the need for a dedicated sensor suite and computer. Images from the rear camera and inertial data from the IMU of the iPhone are used for perception. Different from the previous versions, the ART is controlled by the phone via a Bluno Nano board. The iPhone is connected to

Fig. 3. RC prototype V3 (left) and V4 (right)

the Bluno Nano [43] through Bluetooth. Once Bluno Nano receives a navigation command from the iPhone, it will send out control signals to ART via its GPIO port and RS232 port.

3.2 Wearable Robotic Object Manipulation Aid (W-ROMA)

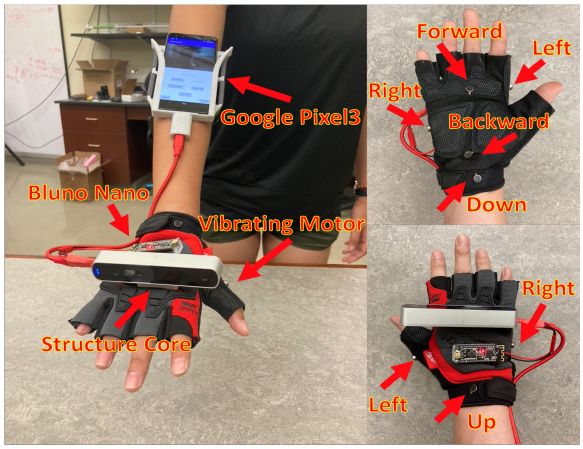


Fig. 4. W-ROMA hardware system

A wearable device named Wearable Robotic Object Manipulation Aid (W-ROMA) was developed to help the BVI with wayfinding, object detection, and grasping. As depicted in Figure 4, the W-ROMA comprises two main units: a sensing unit and a guiding unit. The sensing unit employs an Occipital Structure Core (SC) [44], which

is connected to and powered by a Google Pixel 3 via a USB-C cable. The SC has a built-in 6-axis IMU (Bosch BMI055), a color camera, and a stereo IR camera (for depth measurements from 0.3 to 5m). These sensors form an RGB-D VINS for device pose estimation, 3D mapping, and target-object detection. The computational tasks are all performed by the Pixel 3, which is equipped with a Snapdragon 845 processor and 4 GB RAM [45]. Also, the smartphone powers the whole W-ROMA prototype, including an SC sensor, a Bluno Nano board [43], and six vibrating motors.

A speech interface (using a Bluetooth headset) is used for human-device interaction. In addition, six vibrating motors, controlled by the Bluno Nano board, are used to indicate the direction of hand movement to align the hand with the target object. The guiding unit determines the desired hand movement and generates a proper vibration pattern to guide the user to move their hand to reach the target object. As shown in Figure 4, the vibrating motors are installed on the surface of W-ROMA and controlled by the Bluno Nano board (powered by Pixel 3 via a USB-C splitter cable and communicates with Pixel 3 via Bluetooth). Each vibrating motor indicates one of the six directions for the desired hand movement: up, down, left, right, backward, and forward. A combination of two of the four vibrating motors (up, down, left, right) indicates one of the four diagonal directions of movement: up-right, up-left, down-right, and down-left. Once Bluno Nano receives a command from Pixel 3, it generates the corresponding vibrating pattern and retains it for one second. Such tactile guidance is more natural to follow than the speech command and avoids the speech message delay.

CHAPTER 4

SENSORS FOR SLAM

A visual-inertial navigation system (VINS) consists of a visual camera and a 6-axis IMU. In this dissertation, a depth sensor is also used to enhance the VINS performance. The operating principles of these sensors and the characteristics that may affect SLAM will be detailed in this chapter. Some notions introduced in this chapter will be used throughout the dissertation.

4.1 Visual Camera

In the literature, two camera models have been introduced. They are the thin-lens model and the pin-hole model.

4.1.1 Camera Model

In the literature, two camera models have been introduced. They are the thin-lens model and the pin-hole model. The latter is an ideal form of the former and is more popularly used in visual SLAM. The pinhole model will be employed in this thesis.

4.1.1.1 Thin-lens model

Most modern cameras produce the image of a scene by using one or a set of lenses to focus the light onto the image plane. Rays of light from a surface point (Figure 5) travel along their paths through the lens and converge at the point on the view plane. In Figure 5, f denotes the focal length, and Z_1 denotes the distance from the center of the lens to a surface point on an object. The rays from the surface point $\boldsymbol{P}$ are in

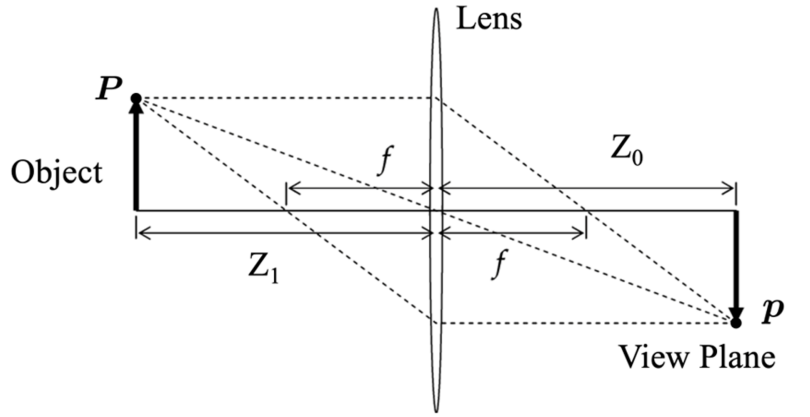


Fig. 5. Illustration of thin-lens model

focus (imaged) on the image plane as $\boldsymbol{p}$ with a distance Z_0 from the lens center. Z_0 and Z_1 satisfy the following thin lens equation:

$$\frac{1}{f} = \frac{1}{Z_0} + \frac{1}{Z_1} \tag{4.1}$$

4.1.1.2 Pin-hole model

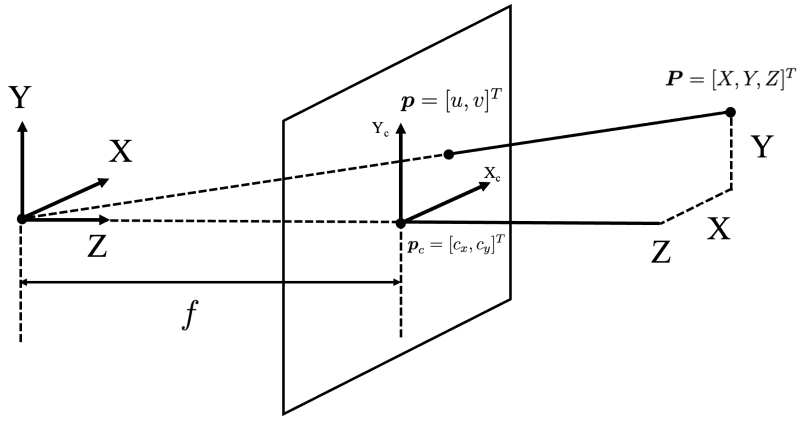


Fig. 6. Illustration of pin-hole model

A pin-hole camera is an idealization of a thin-lens camera as the aperture shrinks to zero and $f = Z_0$ in this case. For this model, a point in 3D space $\boldsymbol{P} = [X, Y, Z]^T$

is projected on image plane as a 2D point $\boldsymbol{p} = [u, v]^T$ with:

$$\begin{aligned} u &= \frac{X \cdot f}{Z} + c_x \\ v &= \frac{Y \cdot f}{Z} + c_y \end{aligned} \tag{4.2}$$

where f is the focal length, c_x and c_y are the coordinates of the principal point. The principal point $\boldsymbol{p}_c = [c_x, c_y]^T$ is perspective center on image plane. c_x, c_y and f form the intrinsic matrix K:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

And Equation 4.2 can be expressed by:

$$Z\boldsymbol{p} = K\boldsymbol{P} \tag{4.4}$$

Letting $\boldsymbol{p} = [u, v, 1]^T$ denotes the 3D projection on a virtual plane with $f = 1$ and $c_x = c_y = 0$, we have $Z\boldsymbol{p} = \boldsymbol{P}$.

In this dissertation, c_x, c_y, and f are collectively called the camera intrinsic parameters (CIP). As shown in Equation 4.2, the CIPs are essential in relating a 3D point to its 2D image on the image plane. In cases where CIPs are unknown or varying, an accurate CIP estimation is critical to the function of the vSLAM system.

The pin-hole camera model is an ideal form of the thin-lens camera model and is popularly used in visual SLAM. Therefore, it is adopted in this dissertation.

4.2 Depth Sensors

A depth sensor, as known as a range sensor, can provide depth information about the scene. The techniques for depth measurement can be categorized into

three types: stereovision, structure light ranging, and Time-of-Flight (ToF). Typical depth sensors include depth cameras and LiDAR. Depth cameras can be classified d into three classes: stereo, structured light, and ToF.

4.2.1 Stereo camera

Stereovision uses two cameras to mimic the binocular vision of humans. The two cameras are fixed rigidly with a known displacement. Depth is triangulated based on the disparity between the left and right images. Limited by its operating principle, stereovision does not work well when the environment is not visual-feature-rich, causing difficulty in stereo matching. Also, the measurement error of a stereo camera increases quadratically with increasing depth. As a result, it is most suitable for short-range depth measurement.

4.2.2 Structured light

A structured light depth sensor projects patterned infrared light onto the scene to ease stereo matching even in a featureless environment. Some structured light sensors also encode depth information in their lighting patterns, which changes with the depth. The encoded information facilitates the computation of the depth for each image pixel. Like stereovision, the depth accuracy of structured light depth sensors also degrades quadratically with increasing distance.

4.2.3 Time of flight (ToF)

ToF depth sensors (including ToF cameras and LiDAR) use indirect Time of Flight (iToF) or direct Time of Flight (dToF) ranging techniques to determine the depth. An iToF depth sensor illuminates the scene by modulated continuous wave IR light and determines the ToF by measuring the phase shift of the reflected light.

Thus, this technique incurs range ambiguity. To overcome the disadvantage, dToF illuminates the scene with a single laser pulse and determines the depth based on the time the laser pulse travels.

In this dissertation, a depth sensor is used to improve the performance of a vSLAM/VI-SLAM system, as it may help determine the metric scale. The related methods will be discussed in the next few chapters.

4.3 Inertial Measurement Unit (IMU)

The IMU used in this dissertation is MEMS (Micro-electromechanical systems) IMU, which is composed of a 3-axis gyroscope and a 3-axis accelerometer and can provide a 6-axis measurement of device movement. The gyroscope measures the angular rate, and the accelerometer measures the acceleration. As IMU provides measurements at a much higher rate (above 100 Hz) than a camera or depth sensor, the measurements must be integrated to produce an IMU-estimated pose change between two imaging/depth data frames. This means that the estimated pose change is sensitive to the noise and bias of IMU data, particularly the translation that it is double integrated from the accelerometer data.

The raw accelerometer measurements $\hat{a}$ and raw gyroscope measurement $\hat{\omega}$ can be expressed as:

$$\begin{aligned} \hat{a}_t &= a_t + \mathrm{b}_{a_t} + \mathbf{R}_w^t \mathrm{g}^w + \mathrm{n}_a \\ \hat{\omega}_t &= \omega_t + \mathrm{b}_{\omega_t} + \mathrm{n}_\omega \end{aligned} \tag{4.5}$$

where b_{a_t} and b_{ω_t} stand for the accelerometer and gyro biases, respectively; $\mathbf{R}_w^t$ denotes the rotation from the device coordinate system to the world coordinate system at time t, and g^w denotes the gravity in the world coordinate system $\{W\}$. The white noises in acceleration and rotation measurements are modeled as Gaussian, $\mathrm{n}_a = \mathcal{N}(0, \sigma_a^2)$ and $\mathrm{n}_\omega = \mathcal{N}(0, \sigma_\omega^2)$, respectively, and the derivatives of biases b_{ω_t} and

b_{ω_t} are Gaussian distributed with $\mathcal{N}\left(0, \sigma_{b_a}^2\right)$ and $\mathcal{N}\left(0, \sigma_{b_\omega}^2\right)$.

During the IMU propagation for pose estimation, the white noises of IMU are unknown and assumed to be 0. The biases are also unknown and will be estimated along with other state variables.

CHAPTER 5

POSE CHANGE ESTIMATION

5.1 Pose Change Estimation with Mono Cameras

The relative movement between two frames can be described as a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. Given two measurements ($\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ on image frames 1 and 2, respectively) of the same point $\boldsymbol{P}$ with the camera intrinsics K_1 and K_2, the spatial relationship can be built as:

$$\boldsymbol{P^{c_2}} = \mathbf{R}\boldsymbol{P^{c_1}} + \mathbf{t} \tag{5.1}$$

where $\boldsymbol{P^{c_1}}$ and $\boldsymbol{P^{c_2}}$ present the position of the 3D point $\boldsymbol{P}$ in the camera's coordinate system $\{C_1\}$ and $\{C_2\}$, respectively.

From Equation 4.4, we have:

$$Z_1\boldsymbol{p}^{c_1} = K_1\boldsymbol{P}^{c_1}$$

$$Z_2\boldsymbol{p}^{c_2} = K_2\boldsymbol{P}^{c_2}$$

where $\boldsymbol{p^{c_1}}$ and $\boldsymbol{p^{c_2}}$ represent the projection of $\boldsymbol{P}$ on a virtual image plane with $f = 1$ in $\{C_1\}$ and $\{C_2\}$, respectively. By substituting Equation 4.4 into Equation 5.1:

$$Z_2\boldsymbol{p^{c_2}} = K_2\boldsymbol{P^{c_2}} = K_2\left(\mathbf{R}\boldsymbol{P^{c_1}} + \mathbf{t}\right) \tag{5.2}$$

$$\boldsymbol{p^{c_2}} = K_2^{-1}Z_2\boldsymbol{P^{c_2}} = \mathbf{R}K_1^{-1}Z_1\boldsymbol{p^{c_1}} + Z_1\mathbf{t} \tag{5.3}$$

$$K_2^{-1}\boldsymbol{p^{c_2}} = \mathbf{R}K_1^{-1}\boldsymbol{p^{c_1}} + \frac{Z_1}{Z_2}\mathbf{t} = \mathbf{R}K_1^{-1}\boldsymbol{p^{c_1}} + s\mathbf{t} \tag{5.4}$$

where $\mathbf{s} = \frac{Z_1}{Z_2}$ is the scale. In the case of a monocular camera, Z_1 and Z_2 are unknown.

As a result, s is unknown. Let $s = 1$ for simplicity.

We define the operation $t^\wedge$ as follow:

$$t^\wedge = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{5.5}$$

By multiplying $t^\wedge$ on both side of Equation 5.4,

$$\mathbf{t}^\wedge K_2^{-1}\boldsymbol{p}^{\boldsymbol{c_2}} = \mathbf{t}^\wedge \mathbf{R} K_1^{-1}\boldsymbol{p}^{\boldsymbol{c_1}} + \mathbf{t}^\wedge \mathbf{t} \tag{5.6}$$

$$\mathbf{t}^\wedge K_2^{-1}\boldsymbol{p}^{\boldsymbol{c_2}} = \mathbf{t}^\wedge \mathbf{R} K_1^{-1}\boldsymbol{p}^{\boldsymbol{c_1}} + \mathbf{0} = \mathbf{t}^\wedge \mathbf{R} K_1^{-1}\boldsymbol{p}^{\boldsymbol{c_1}} \tag{5.7}$$

By multiplying $K_2^{-1}\boldsymbol{p}^{\boldsymbol{c_2}}$ on both sides of Equation 5.7, we have:

$$\left(K_2{}^{-1}\boldsymbol{p}^{\boldsymbol{c_2}}\right)^T \mathbf{t}^\wedge K_2{}^{-1}\boldsymbol{p}^{\boldsymbol{c_2}} = \mathbf{0} = \left(K_2{}^{-1}\boldsymbol{p}^{\boldsymbol{c_2}}\right)^T \mathbf{t}^\wedge \mathbf{R} K_1^{-1}\boldsymbol{p}^{\boldsymbol{c_1}} \tag{5.8}$$

$$\boldsymbol{p}^{\boldsymbol{c_2}T} K_2{}^{-1} \mathbf{t}^\wedge \mathbf{R} K_1{}^{-1}\boldsymbol{p}^{\boldsymbol{c_1}} = \mathbf{0} \tag{5.9}$$

Let $\mathbf{x_1} = K_1{}^{-1}\boldsymbol{p}^{\boldsymbol{c_1}}$ and $\mathbf{x_2} = K_2{}^{-1}\boldsymbol{p}^{\boldsymbol{c_2}}$, we have:

$$\mathbf{x_2}{}^T \mathbf{E} \mathbf{x_1} = \boldsymbol{p}^{\boldsymbol{c_2}T} \mathbf{F} \boldsymbol{p}^{\boldsymbol{c_1}} = \mathbf{0} \tag{5.10}$$

where $\mathbf{E} = \mathbf{t}^\wedge \mathbf{R}$ which is known as Essential matrix, and $\mathbf{F} = K_2{}^{-1}\mathbf{E}K_1$ which is known as Fundamental matrix [46].

$\boldsymbol{E}$ is a 3×3 matrix with 9 unknowns which can be solved by at least 8 points, any four of which are non-coplanar, with the Eight-point algorithm [47]. The rotation $\mathbf{R}$ and $\mathbf{t}$ can be recovered by singular value decomposition (SVD) [48].

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T \tag{5.11}$$

where a ($a > 0$) is the singular value of $\mathbf{E}$, $\mathbf{U}$ and $\mathbf{V}$ are two orthogonal matrixes.

There are two possible factorizations of $\mathbf{E}$,

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} 0 & -a & 0 \\ a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T \tag{5.12}$$

or

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} 0 & a & 0 \\ -a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T \tag{5.13}$$

The solution for Equation 5.12 is:

$$\mathbf{R_1} = \mathbf{U} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T, \mathbf{t_1} = \mathbf{U} \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} \tag{5.14}$$

The solution for Equation 5.13 is:

$$\mathbf{R_2} = \mathbf{U} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T, \mathbf{t_2} = \mathbf{U} \begin{bmatrix} 0 \\ 0 \\ -a \end{bmatrix} \tag{5.15}$$

Since $\mathbf{E} = -\mathbf{t}\hat{}\mathbf{R}$ also satisfies Equation 5.10, there are two more possible solutions:

$$\mathbf{R_3} = \mathbf{U} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T, \mathbf{t_3} = \mathbf{U} \begin{bmatrix} 0 \\ 0 \\ -a \end{bmatrix} \tag{5.16}$$

or

$$\mathbf{R_4} = \mathbf{U} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T, \mathbf{t_4} = \mathbf{U} \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} \tag{5.17}$$

Even though there are four possible solutions (Figure 7), only one solution ($\mathbf{R_1}$, $\mathbf{t_1}$) is valid because the object must be in front of the image plane.

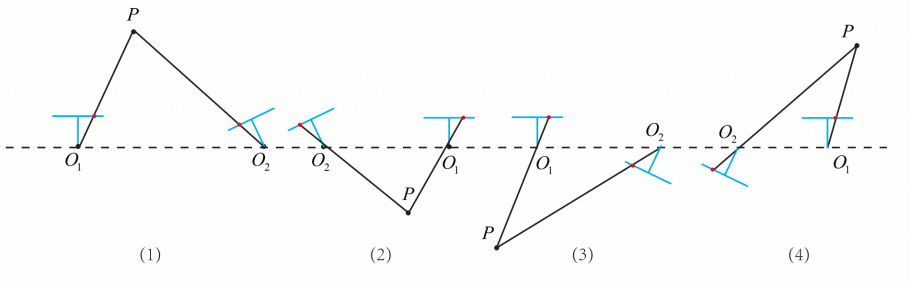


Fig. 7. Four possible solutions from SVD of **E**. The blue lines indicate the cameras with different points of view. The red dots indicate the projections on the image plane.

According to Equation 5.10, the estimation of the Fundamental matrix **F** only depends on the coordinates of the image points. Therefore, this pose estimation method is also called a 2D-2D method.

5.2 Pose Change Estimation with RGB-D Cameras

5.2.1 Pose Change Estimation with 2D-3D correspondences

Although the rotation between two camera image frames can be estimated with a single visual (monocular) camera, the translation can only be estimated up to the scale (i.e., by assuming an arbitrary scale). In other words, the translation is unknown. By using an RGB-D camera, the scale can be directly computed from the depth data. As a result, both rotation and translation (the full 6-degree-of-freedom (DoF) pose

change) can be estimated. An RGB-D camera, consisting of a visual camera and a depth sensor, produces both image and depth data. A depth data point Z^d can be associated with an image pixel $\boldsymbol{p^c}$ by:

$$\boldsymbol{P^c} = Z^c {K_c}^{-1} \boldsymbol{p^c} = \mathbf{R}_d^c \boldsymbol{P^d} + \mathbf{t}_d^c = \mathbf{R}_d^c Z^d {K_d}^{-1} \boldsymbol{p^d} + \mathbf{t}_d^c \tag{5.18}$$

where $\mathbf{R}_d^c$ and $\mathbf{t}_d^c$ are the extrinsics of the camera-depth-sensor suite. Most RGB-D cameras provide Z^c, which is registered with $\boldsymbol{p^c}$ and computed with the raw depth measurements Z^d by Equation 5.18. For simplicity, Z denotes the depth measurement for $\boldsymbol{p^c}$ by default.

For 3D point $\boldsymbol{P^{c_1}} = [X, Y, Z, 1]$ in the camera coordinate system $\{C_1\}$ and its 2D projection $\boldsymbol{p^{c_2}} = [u, v, 1]$ at camera coordinate system $\{C_2\}$, we have::

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.19}$$

With the depth information from the first frame, the scale (s) can be determined accordingly. And from Equation 5.19, we have:

$$\begin{aligned} u &= \frac{a_1 X + a_2 Y + a_3 Z + a_4}{a_9 X + a_{10} Y + a_{11} Z + a_{12}} \\ v &= \frac{a_5 X + a_6 Y + a_7 Z + a_8}{a_9 X + a_{10} Y + a_{11} Z + a_{12}} \end{aligned} \tag{5.20}$$

$[\mathbf{R} \mid \mathbf{t}] \in \mathbf{R}^{3\times4}$ can be solved by Direct Linear Transform (DLT) [49]. At least 6 pairs of 3D-2D associations (any four of them are non-planar) are required to solve the problem.

5.2.2 Pose Change Estimation with 3D-3D correspondences

For the feature points with a depth measurement on both frames, the 6-DoF pose can also be estimated by ICP [50]. $\mathbf{R}$ and $\mathbf{t}$ can be solved by minimizing the reprojection error J for a pair of corresponding 3D point-sets $\boldsymbol{P_i}^{\boldsymbol{c_1}}$ and $\boldsymbol{P_i}^{\boldsymbol{c_2}}$:

$$\min_{\mathbf{R},\mathbf{t}} J = \frac{1}{2}\sum_{i=1}^{n} \left\| (\boldsymbol{P_i}^{\boldsymbol{c_1}} - (\mathbf{R}\boldsymbol{P_i}^{\boldsymbol{c_2}} + \mathbf{t})) \right\|^2 \tag{5.21}$$

Let $\bar{\boldsymbol{P}}^{\boldsymbol{c_1}}$ and $\bar{\boldsymbol{P}}^{\boldsymbol{c_2}}$ denote the centroids of each point set:

$$\bar{\boldsymbol{P}}^{\boldsymbol{c_1}} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{P_i}^{\boldsymbol{c_1}}), \text{ and } \bar{\boldsymbol{P}}^{\boldsymbol{c_2}} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{P}_i^{c_2}) \tag{5.22}$$

J in Equation 5.21 can be derived as:

$$\begin{aligned}
J =& \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{P_i}^{\boldsymbol{c_1}} - \mathbf{R}\boldsymbol{P_i}^{\boldsymbol{c_2}} - \mathbf{t} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} + \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} + \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} \right\|^2 \\
=& \frac{1}{2}\sum_{i=1}^{n} \left\| \left(\boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right)\right) + \left(\bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} - \mathbf{t}\right) \right\|^2 \\
=& \frac{1}{2}\sum_{i=1}^{n} \Big(\left\| \boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right) \right\|^2 + \left\| \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} - \mathbf{t} \right\|^2 + \\
& 2\left(\boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right)\right)^T \left(\bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} - \mathbf{t}\right) \Big)
\end{aligned} \tag{5.23}$$

Since $\sum_{i=1}^{n} \left(\boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right)\right) = 0$, the equation can be simplified as:

$$\min_{\mathbf{R},\mathbf{t}} J = \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right) \right\|^2 + \left\| \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} - \mathbf{t} \right\|^2 \tag{5.24}$$

The $\mathbf{R}^*$ and $\mathbf{t}^*$ to minimize J can be obtained by solving:

$$\begin{aligned}
\mathbf{R}^* &= \arg\min_{\mathbf{R}} \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{P_i}^{\boldsymbol{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\left(\boldsymbol{P_i}^{\boldsymbol{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right) \right\|^2 \\
\mathbf{t}^* &= \bar{\boldsymbol{P}}^{\boldsymbol{c_1}} - \mathbf{R}\bar{\boldsymbol{P}}^{\boldsymbol{c_2}} - \mathbf{t}
\end{aligned} \tag{5.25}$$

Equation 5.25 can be solved by SVD:

$$\begin{aligned} \boldsymbol{W} &= \sum_{i=1}^{n} \left(\boldsymbol{P_i^{c_1}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_1}}\right)\left(\boldsymbol{P_i^{c_2}} - \bar{\boldsymbol{P}}^{\boldsymbol{c_2}}\right)^T \\ \boldsymbol{W} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ \mathbf{R} &= \boldsymbol{U}\boldsymbol{V}^T \end{aligned} \tag{5.26}$$

Each point set must contain at least three non-collinear points to solve the pose change estimation problem.

5.2.3 Hybrid Pose Change Estimation

Each of the above-mentioned methods has its advantages and disadvantage. A 2D-2D method uses all correspondences to compute pose estimate and thus may result in a smaller pose estimation error. However, it cannot determine the metric scale, i.e., the translation is estimated by assuming an arbitrary scale. A 3D-2D/3D-3D method, on the other hand, does not incur the scale problem. But it may not use all correspondences for pose computation because only some of them have a depth measurement. This may cause a larger pose estimation error. Usually, a 3D-2D method has better accuracy than a 3D-3D method because there are more 3D-2D correspondences than 3D-3D correspondences. As illustrated in Figure 8, there are 9 2D points that are associated between $\{C_1\}$ and $\{C_2\}$, while only 6 points have depth measurements on each frame. As a result, there are 6 associations that can be established if a 3D-2D method is used and only 3 associations if a 3D-3D method is used. Therefore, a 3D-2D method has better accuracy than a 3D-3D method as it uses more data points to estimate the pose change. A commercially available RGB-D camera may incur missing depth data on a dark/reflective surface. In a difficult case, only a few visual features have a depth measurement. In this scenario, existing

3D-2D methods may result in a large error in pose change estimation. To mitigate the issue, a hybrid-perspective-n-point (HPnP) method was proposed in this work. HPnP decouples the rotation and translation computation, i.e., first computes the rotation by using all 2D points (visual features) and then determines the translation using depth measurements from both frames. In this dissertation, the Eight-point algorithm [47]is used to compute the rotation.

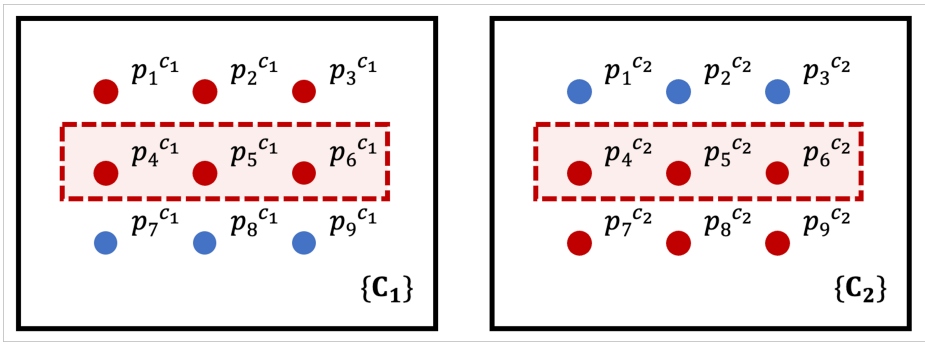


Fig. 8. Point associations between $\{C_1\}$ and $\{C_2\}$. The red points indicate the points with a depth measurement, and the blue points indicate the points without a depth measurement. The points with a depth measurement in both $\{C_1\}$ and $\{C_2\}$ are highlighted by the red box.

Assuming that there are K 3D points in $\{C_1\}$ and M 3D points in $\{C_2\}$ among the N associated visual features and that rotation $\mathbf{R}$ has been computed, HPnP solves the translation by using the projection equation. For a 3D point $\boldsymbol{P}^{\boldsymbol{c_1}}$ in $\{C_1\}$ and the corresponding 2D point $\boldsymbol{p}^{\boldsymbol{c_2}}$ in $\{C_2\}$:

$$\boldsymbol{P}^{\boldsymbol{c_2}} = Z^{c_2}\boldsymbol{p}^{\boldsymbol{c_2}} = \mathbf{R}\boldsymbol{P}^{\boldsymbol{c_1}} + \mathbf{t} \tag{5.27}$$

where $\boldsymbol{P}^{\boldsymbol{c_1}}$ can be calculated with $\boldsymbol{P}^{\boldsymbol{c_1}} = Z^{c_1}\boldsymbol{p}^{\boldsymbol{c_1}} = [X^{c_1}, Y^{c_1}, Z^{c_1}]^T$. Note that $\boldsymbol{p}^{\boldsymbol{c_2}}$ is the projection of $\boldsymbol{P}^{\boldsymbol{c_1}}$ onto the image plane with $f = 1$, i.e., $\boldsymbol{p}^{\boldsymbol{c_2}} = [u^{c_2}, v^{c_2}, 1]^T$ in C2.

i.e.,

$$\begin{aligned} \mathrm{row}_1(\mathbf{R})\boldsymbol{p}^{\boldsymbol{c_1}} + t_1 &= Z^{c_2}u^{c_2} \\ \mathrm{row}_2(\mathbf{R})\boldsymbol{p}^{\boldsymbol{c_1}} + t_2 &= Z^{c_2}v^{c_2} \\ \mathrm{row}_3(\mathbf{R})\boldsymbol{p}^{\boldsymbol{c_1}} + t_3 &= Z^{c_2} \end{aligned} \tag{5.28}$$

where $\mathrm{row}_i(\mathbf{R})$ (i = 1,2,3) are the i^{th} row of $\mathbf{R}$ and $\mathbf{t} = [t_1, t_2, t_3]^T$, respectively. By substituting Z^{c_2} in the 3^{rd} row of Equation 5.28 into the other two equations, we obtain:

$$\begin{aligned} t_1 - u^{c_2}t_3 &= u^{c_2}\,\mathrm{row}_3(\mathbf{R})\mathbf{p}^{c_1} - \mathrm{row}_1(\mathbf{R})\boldsymbol{p}^{\boldsymbol{c_1}} \\ t_2 - v^{c_2}t_3 &= v^{c_2}\,\mathrm{row}_3(\mathbf{R})\mathbf{p}^{c_1} - \mathrm{row}_1(\mathbf{R})\boldsymbol{p}^{\boldsymbol{c_1}} \end{aligned} \tag{5.29}$$

i.e.,

$$\begin{bmatrix} 1 & 0 & -u^{c_2} \\ 0 & 1 & -v^{c_2} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} (u^{c_2}\,\mathrm{row}_3(\mathbf{R}) - \mathrm{row}_1(\mathbf{R}))\,\boldsymbol{p}^{\boldsymbol{c_1}} \\ (v^{c_2}\,\mathrm{row}_3(\mathbf{R}) - \mathrm{row}_2(\mathbf{R}))\,\boldsymbol{p}^{\boldsymbol{c_1}} \end{bmatrix} \tag{5.30}$$

For the k^{th} ($k = 1, \cdots, K$) 3D-2D point-pair, we can re-write Equation 5.30 as $\mathbf{M_k^1 t} = \mathbf{b_k^1}$ for simplicity.

Similarly, for a 3D point $\boldsymbol{P}^{\boldsymbol{c_2}}$ in $\{C_2\}$ and its projection $\boldsymbol{p}^{\boldsymbol{c_1}}$in $\{C_1\}$, we have:

$$\begin{bmatrix} \mathrm{row}_1(\mathbf{R}^T) - u^{c_1}\,\mathrm{row}_3(\mathbf{R}^T) \\ \mathrm{row}_2(\mathbf{R}^T) - v^{c_1}\,\mathrm{row}_3(\mathbf{R}^T) \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} \left(\mathrm{row}_1(\mathbf{R}^T) - u^{c_1}\,\mathrm{row}_3(\mathbf{R}^T)\right)\boldsymbol{p}^{\boldsymbol{c_2}} \\ \left(\mathrm{row}_2(\mathbf{R}^T) - v^{c_1}\,\mathrm{row}_3(\mathbf{R}^T)\right)\boldsymbol{p}^{\boldsymbol{c_2}} \end{bmatrix} \tag{5.31}$$

For the m^{th} ($m = 1, \cdots, M$) 3D-2D point-pair, we can re-write it as $\mathbf{M_m^2 t} = \mathbf{b_m^2}$.

Combining the two point sets, we have:

$$\begin{bmatrix} \mathbf{M_1^1} \\ \cdots \\ \mathbf{M_K^1} \\ \mathbf{M_1^2} \\ \cdots \\ \mathbf{M_M^2} \end{bmatrix} \mathbf{t} = \begin{bmatrix} \mathbf{b_1^1} \\ \cdots \\ \mathbf{b_K^1} \\ \mathbf{b_1^2} \\ \cdots \\ \mathbf{b_M^2} \end{bmatrix} \tag{5.32}$$

For the above overdetermined system $\mathbf{Mt} = \mathbf{b}$, the solution $\mathbf{t}$ that minimizes $\|\mathbf{Mt} - \mathbf{b}\|^2$ is given by [51]:

$$\mathbf{t} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{b} \tag{5.33}$$

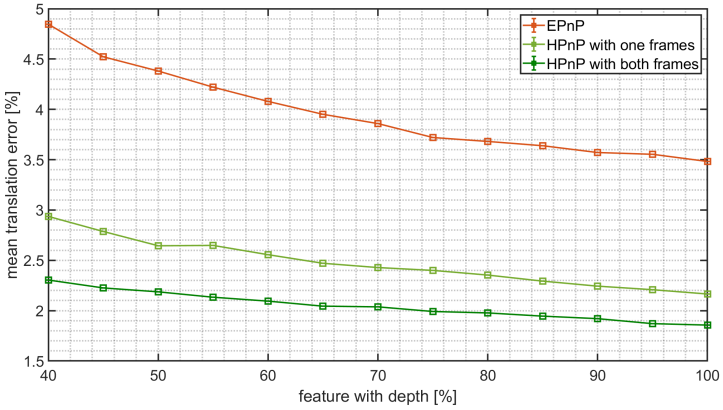
where $\mathbf{M}$ can be decomposed as $\mathbf{M} = \mathbf{U\Sigma V}^T$ by SVD [48].

Therefore,

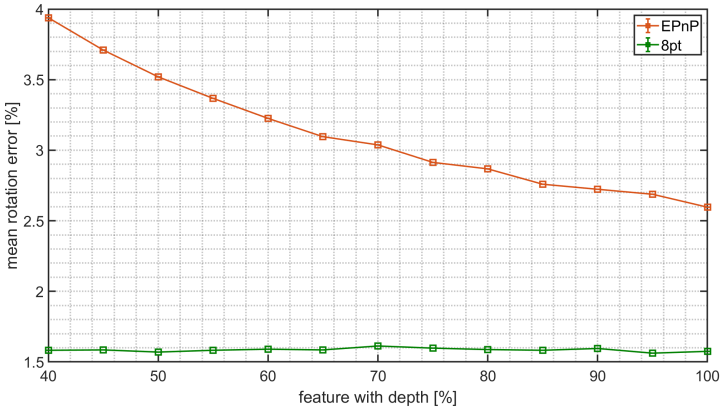
$$\mathbf{t} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T\mathbf{b} \tag{5.34}$$

To validate the proposed HPnP, simulation was performed by using a virtual stereo camera (camera intrinsic parameters: $c_x = 320$, $c_y = 240$, $f_x = f_y = 460$ pixels, stereo baseline: 50 mm). The camera was simulated with imagery noise $n_\tau \sim \mathcal{N}(0, \sigma_\tau^2)$ ($\sigma_\tau = 0.5$ pixel). The depth data of each visual feature was computed from the image disparity by triangulation.

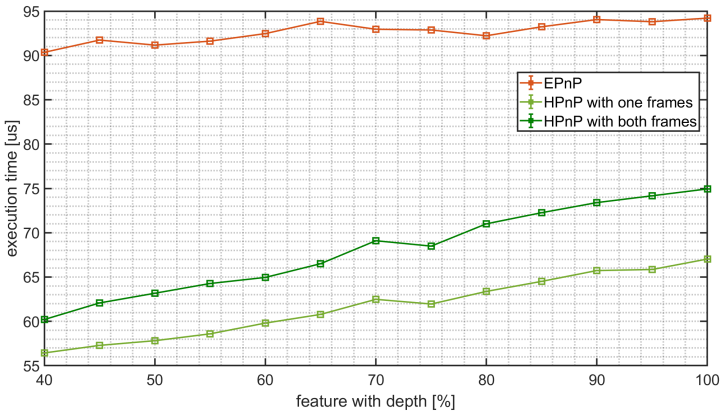
The simulation compares the performance of HPnP with that of EPnP [52] under different numbers of visual features with a depth measurement. The percentage of visual features with a depth measurement changes at an increment of 10%. 1000 iterations were performed for each increment. In each iteration, 40 visual feature correspondences were randomly created. The depth data for some (40% - 100%) of the visual features were computed by triangulation. The movement between the two camera poses was set as 10 degrees around an arbitrary direction and 20 cm along



(a) Mean translation error



(b) Mean rotation error



(c) Mean execution time

Fig. 9. Transformation error of estimating a random motion of 20 cm translation and 10-degree rotation. HPnP computes the translation using the rotation estimate from the Eight-point method [47].

an arbitrary direction. The proposed HPnP is compared with EPnP as EPnP is regarded as the state-of-the-art closed-form method [53]. The translation estimation results are depicted in Figure 9 (a). To demonstrate the effect of using depth data from both frames, HPnP using depth data from only one frame is also plotted. It can be seen that 1) the translation estimation error of HPnP is consistently lower and much better than that of EPnP; 2) translation estimation accuracy is improved by using depth measurements from both frames; 3) as the percentage of visual features with a depth measurement increase the differences between the three methods' estimation errors decrease. Figure 9 (b) compares the rotation estimation error of the eight-point method with that of EPnP. Apparently, the Eight-point method is much more accurate, and therefore, it is adopted by the proposed HPnP method as a smaller rotation estimation error help to improve the accuracy of translation estimation. In addition, although HPnP introduces more depth information which results in a larger matrix when solving SVD, the computation growth is relatively slow. The complexity of computing the SVD of a problem $Ax = b$, where A is a $N \times M$ matrix, is typically, $O(min(N^2M, NM^2)) + O(min(N, M)) + O(M^2N)$. For translation computation, $M = 3$, so, the overall complexity is $O(N)$. As a result, HPnP has an even smaller execution time than EPnP, as depicted in Figure 9 (c).

5.3 Pose Change Estimation with IMU

Unlike camera-based pose change estimation, IMU measurement can be used to calculate the device's 6-DoF poses in the world coordinate. By integrating the IMU data during time interval $[t_k, t_{k+1}]$, device's Position $p^w_{b_{k+1}}$, Velocity $v^w_{b_{k+1}}$, and

Quaternion $\mathbf{q}_{b_{k+1}}^{w}$ (PVQ) can be computed as:

$$\begin{aligned} p_{b_{k+1}}^{w} &= p_{b_k}^{w} + v_{b_k}^{w}\Delta t_k + \iint_{t\in[k,k+1]} \left[\mathbf{R}_t^{w} a_t - g^{w}\right] dt^2 \\ v_{b_{k+1}}^{w} &= v_{b_k}^{w} + \int_{t\in[k,k+1]} \left[\mathbf{R}_t^{w} a_t - g^{w}\right] dt \\ \mathbf{q}_{b_{k+1}}^{w} &= \mathbf{q}_{b_k}^{w} \otimes \int_{t\in[k,k+1]} \frac{1}{2}\Omega(\omega_t)\mathbf{q}_t^{b_k} dt \end{aligned} \tag{5.35}$$

where

$$\Omega(\omega) = \begin{bmatrix} -\omega^{\wedge} & \omega \\ -\omega^{T} & 0 \end{bmatrix}, \omega^{\wedge} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{5.36}$$

and g^{w} denotes the gravity in the world coordinates.

It can be seen that the IMU state propagation requires the position, velocity, and rotation of frame b_k, and it must be re-computed when the starting state of frame b_k changes. To avoid this state re-propagation and save computational cost, the pre-integration algorithm [54] is adopted. The method computes the IMU state in the local frame b_k (instead of the world frame), and thus IMU integration can be performed by:

$$\begin{aligned} \mathbf{R}_w^{b_k} p_{b_{k+1}}^{w} &= \mathbf{R}_w^{b_k}\left(p_{b_k}^{w} + v_{b_k}^{w}\Delta t_k - \frac{1}{2}g^{w}\Delta t_k^2\right) + \alpha_{b_{k+1}}^{b_k} \\ \mathbf{R}_w^{b_k} v_{b_{k+1}}^{w} &= \mathbf{R}_w^{b_k}\left(v_{b_k}^{w} - g^{w}\Delta t_k\right) + \beta_{b_{k+1}}^{b_k} \\ \mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^{w} &= \gamma_{b_{k+1}}^{b_k} \end{aligned} \tag{5.37}$$

where:

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} \left[\mathbf{R}_t^{b_k} a_t\right] dt^2 \\ \beta_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \left[\mathbf{R}_t^{b_k} a_t\right] dt \\ \gamma_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \frac{1}{2}\Omega(\omega_t)\gamma_t^{b_k} dt \end{aligned} \tag{5.38}$$

After considering the acceleration bias n_{b_a}, gyroscope bias n_{b_ω}, and additive noise n_a and n_ω, Equation 5.37 is changed to:

$$\begin{aligned}
\alpha_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} \left[\mathbf{R}_t^{b_k}\left(\hat{a}_t - b_{a_t} - n_a\right)\right] dt^2 \\
\beta_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \left[\mathbf{R}_t^{b_k}\left(\hat{a}_t - b_{a_t} - n_a\right)\right] dt \\
\gamma_{b_{k+1}}^{b_k} &= \int_{t\in[k,k+1]} \frac{1}{2}\Omega\left(\hat{\omega}_t - b_{\omega_t} - n_\omega\right)\gamma_t^{b_k} dt
\end{aligned} \tag{5.39}$$

$\alpha_{b_{k+1}}^{b_k}$, $\beta_{b_{k+1}}^{b_k}$ and $\gamma_{b_{k+1}}^{b_k}$ can be updated by their first-order approximations with respect to b_a and b_ω:

$$\begin{aligned}
\alpha_{b_{k+1}}^{b_k} &\approx \hat{\alpha}_{b_{k+1}}^{b_k} + \mathrm{J}_{b_a}^{\alpha}\delta\mathrm{b}_{a_k} + \mathrm{J}_{b_\omega}^{\alpha}\delta\mathrm{b}_{\omega_k} \\
\beta_{b_{k+1}}^{b_k} &\approx \hat{\beta}_{b_{k+1}}^{b_k} + \mathrm{J}_{b_a}^{\beta}\delta\mathrm{b}_{a_k} + \mathrm{J}_{b_\omega}^{\beta}\delta\mathrm{b}_{\omega_k} \\
\gamma_{b_{k+1}}^{b_k} &\approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} \frac{1}{2} \\ \mathrm{J}_{b_\omega}^{\gamma}\delta\mathrm{b}_{\omega_k} \end{bmatrix}
\end{aligned} \tag{5.40}$$

where $\mathrm{J}_{b_a}^{\alpha}$,$\mathrm{J}_{b_a}^{\beta}$, $\mathrm{J}_{b_a}^{\gamma}$,$\mathrm{J}_{\omega_a}^{\alpha}$,$\mathrm{J}_{\omega_a}^{\beta}$, $\mathrm{J}_{\omega_a}^{\gamma}$ at b_{k+1} can be calculated recursively from $\mathbf{J}_{b_k} = \mathbf{I}$. In this dissertation, Mid-point integration [55] is used for discrete-time calculation of PVQ:

$$\begin{aligned}
\hat{\alpha}_{b_{i+1}}^{b_k} &= \hat{\alpha}_i^{b_k} + \hat{\beta}_i^{b_k}\delta t + \frac{1}{2}\hat{\bar{a}}_i\delta t^2 \\
\hat{\beta}_{b_{i+1}}^{b_k} &= \hat{\beta}_i^{b_k} + \hat{\bar{a}}_i\delta t \\
\hat{\gamma}_{b_{i+1}}^{b_k} &= \hat{\gamma}_i^{b_k} \otimes \hat{\gamma}_{i+1}^{i} = \hat{\gamma}_i^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\hat{\bar{\omega}}_i\delta t \end{bmatrix}
\end{aligned} \tag{5.41}$$

where:

$$\begin{aligned}
\hat{\bar{a}}_i &= \frac{1}{2}\left[q_i\left(\hat{a}_i - b_{a_i}\right) + q_{i+1}\left(\hat{a}_{i+1} - b_{a_i}\right)\right] \\
\hat{\bar{\omega}}_i &= \frac{1}{2}\left(\hat{\omega}_i + \hat{\omega}_{i+1}\right) - b_{\omega_i}
\end{aligned} \tag{5.42}$$

δt is the time interval between the i^{th} and $(i+1)^{th}$ IMU measurements.

CHAPTER 6

COMPARISON AND ANALYSIS FOR THE VISUAL-INERTIAL SLAM METHODS

6.1 Operating Principle of Visual-inertial SLAM

A graph-based visual-inertial SLAM method consists of five essential components: initialization, feature tracking, loop detection (or re-localization), IMU pre-integration, and nonlinear optimization. Initialization strives to estimate the initial state of a VINS. Due to the nonlinearity, the quality of the initial state estimation may significantly affect the accuracy of the backend nonlinear optimizer. Feature tracking extracts visual features from the camera's current frame and associates them with that of the previous frames to generate constraints between the corresponding nodes of the graph for graph optimization. Loop detection identifies a loop closure when the VINS re-visits the same scene, and the loop-closure is then used to eliminate accumulated pose error and recover the VINS from tracking failure. IMU pre-integration integrates the readings of IMU between each two camera frames into state changes (including the changes in pose and velocity, as well as the IMU bias) to create inertial constraints between the two nodes. The nonlinear optimizer finds the estimate for the VINS's current state that minimizes the graph errors that factors the visual and the inertial constraints. In this Chapter, the operating principles and performances of several state-of-the-art visual-inertial SLAM methods, including OKVIS [6], VI-ORB [7], and VINS-Mono [8], are evaluated, and the best one is selected for RC application. Since these methods use a similar nonlinear optimization method, the strategies used in the other four components are compared.

6.1.1 Initialization

Initialization aims to compute the initial estimation for a VINS's parameters, such as the initial velocity, gravity direction, visual scale, and the biases of IMU (gyroscope and accelerometer). Among OKVIS [6], VI-ORB [7], and VINS-Mono [8], OKVIS adopts the simplest 2-step initialization procedure. First, it computes the VINS's attitude by aligning the IMU-measured acceleration with gravity. To make the gravitational acceleration observable, the initial acceleration and IMU biases are assumed to be zero. Second, it derives the visual scale by aligning the pose obtained by visual odometry (VO) and the pose obtained by IMU integration. To calculate the IMU translation, the initial velocity and the IMU's initial biases are assumed to be zero. Unfortunately, these assumptions are not applicable to the RC applications as the VI user may be walking and swinging the RC. Therefore, OKVIS may incur an inaccurate initialization, which may degrade the VIO performance.

VI-ORB [7] (Visual-Inertial ORB-SLAM) employs ORB-SLAM [32] to build a visual structure as well as a local map, which is used to establish visual constraints between keyframes. The use of a local map allows for re-establishing visual feature correspondence (i.e., visual constraint) between keyframes if these features move out of the camera's FOV and re-enter the FOV. This visual constraint re-establishment is called local loop-closure in this dissertation. VI-ORB obtains the initial state estimate, including the scale, IMU bias, and motion state, by aligning the visual structure with the structure obtained by integrating the IMU measurements. It takes the method time to build a local map of sufficient size and use it to perform the initialization process. In some cases, the initialization may take $\sim$15 seconds [56] to converge. This is too long for RC when re-localization is needed (e.g., for system recovery from a failure), making the method unsuitable for RC application.

Unlike VI-ORB [7] that simultaneously estimation of gyroscope and accelerometer biases, VINS-Mono [8] simplifies and thus speeds up computation by ignoring accelerometer bias during the initialization step. The rationale is that considering the large magnitude of the gravity vector, the effect of the accelerometer bias is quite limited due to the short duration of initialization and low VINS dynamics. Similar to VI-ORB [7], VINS-mono [8] takes a 2-step approach, building a visual structure by vision-only SfM (structure from motion) and then aligning this structure with the pose change estimated by integrating the IMU measurements, to recover the gyroscope bias, visual scale, velocity, and gravity direction. After the initialization, VINS-Mono boosts a tightly-coupled nonlinear optimization framework that estimates the state of the VINS, including the inverse depths of the features (when first observed) in camera coordinate systems, the position, velocity, and orientation of the IMU in the world coordinate system, and IMU bias in the IMU body coordinate system. VINS-Mono's initialization process converges within several camera frames and thus can be achieved in real-time.

6.1.2 Feature tracking

Feature tracking consists of three steps: first, track the features from the previous image frame onto the current frame; second, extract more features from the current frame; third, associate these new features with those from earlier frames (stored in a local feature map) to create constraints between the current and all previous frames. VI-ORB, OKVIS, and VINS-Mono have little difference in the first two steps. However, they use different strategies for the third step, resulting in different performances.

VI-ORB [7] enforces a strict feature tracking strategy of ORB-SLAM [32]. Specifically, it reprojects the features in a local map onto the current image frame to find

the matches. If the number of matched features is below a threshold, the tracking process fails. Otherwise, a local bundle adjustment is run to refine the matched features' locations. To keep the computational cost low, the method maintains a covisibility graph, a graph structure based on the field of view, and uses it to quickly identify feature matches. This tracking strategy may fail in a feature-sparse area.

Similar to VI-ORB [7], OKVIS [6] uses a local map to find visual feature correspondences between the current and previous keyframe. However, it employs IMU-integrated pose prediction to aid visual feature association, i.e., pose change predicted by using IMU data is used to project the keypoints in the local map onto the current image frame for visual feature association. As IMU data can be used to accurately estimate pose change in a short period, OKVIS may perform more reliable feature matching than VI-ORB, which is purely visual based, in some difficult cases (e.g., VINS is experiencing an abrupt motion [57]). This capability is essential to RC application because RC may incur abrupt motion when the blind user is swinging the device.

Unlike VI-ORB [7] and OKVIS [6], VINS-Mono [8] does not maintain a local map for feature associations. Instead, it tracks the visual features between the current and the previous image frame by simply using the KLT optical flow algorithm [58]. For simplicity, this type of feature tracking approach is called Simple Match (SM) strategy in this dissertation. Due to the use of the SM strategy, a visual feature that moves out of and re-enters the camera's FOV will create two tracks and thus disconnect the visual constraints between the nodes of the two separate tracks.

6.1.3 Loop closure detection

Loop closures in the case of RC can be classified into short-term loops (STLs), which are caused by the periodical swing motion of the cane, and long-term loops

(LTLs), where the RC revisits the same scene. VINS' pose uncertainty grows over time. An LTL, if detected, can be used to eliminate the accumulated pose error. An LTL can be detected by comparing the appearance of the scenes. VI-ORB [7] and VINS-Mono [8] detect an LTL by evaluating the similarity of the scenes based on DBoW [59] and bag-of-word, respectively. However, OKVIS [6] is incapable of detecting LTL. For STL detection, VI-ORB [7] and OKVIS [6] use a local map to create feature associations between the current frame and the previous keyframes. However, VINS-Mono ignores STLs. Since STLs periodically occur when using RC, the best strategy is to detect and use both short-term and long-term loops.

6.1.4 IMU pre-integration

IMU pre-integration was first proposed by Lupton and Sukkarieh in [60]. It computes IMU pose change between two image frames by integrating IMU measurements in the body/IMU frame. IMU pre-integration can be performed without the need for a known VINS state and therefore avoids repeated integration of IMU measurements as the IMU pose is iteratively updated during the process of VIO computation. Forster et al. [54] improve the pre-integration theory by using the SO(3) manifold structure to represent the rotation group and modeling the noise propagation and posterior bias correction. As compared in [54], the SO(3) manifold-based IMU pre-integration method outperforms the Euler-forward method. In VI-ORB [7] and VINS-Mono [8], Forster's method is used to process IMU's measurements, while in OKVIS, the Euler-forward method is used.

6.1.5 Summary

Table 1 summarizes the above comparison of the three VI-SLAM methods. VINS-Mono is adopted as the framework for this work as it has the best and fastest ini-

tialization and uses a manifold structure for IMU preintegration. However, its simple match strategy results in its incapability of SLC detection. Thus, VINS-Mono may be enhanced by performing feature matching between several consecutive keyframes so that it can detect SLCs and incorporate the related constraint into the graph to improve pose estimation accuracy.

Table 1. Comparison of the VI-SLAM methods.

VI- SLAM	**Initialization**	**Loop Detection**	**Feature Tracking**	**IMU Pre-integration**
VIORB	Slow, whole state estimation	**Long-term, Short-term**	Local map	**Manifold**
OKVIS	Fast, only estimate gravity direction	Short-term	**Local map, IMU-aided**	Euler Forward
VINS- Mono	**Fast, whole state estimation (except for acceleration bias)**	Long-term	Optical Flow	**Manifold**

For each column, the best strategy is bolded.

6.2 Experiments using TUM VI benchmark

To quantitatively compare the performances of the three VI-SLAM methods, experiments were conducted by running these methods on the TUM VI benchmark [61]. TUM VI benchmark [61] contains five diverse sets of sequences in various indoor/outdoor scenarios for evaluating VIO/VI-SLAM. Four indoor settings, including room, corridor, magistrale, and slides, were used for comparison. For trajectory evaluation, the ground truth poses at the start and end points of each sequence are provided. For the room set, the ground truth is given throughout each trajectory. VI-ORB [7], OKVIS [6], and VINS-Mono [8] were run on each dataset five times. For the sake of a fair comparison, all methods were set to monocular (inertial) mode with

their long-term loop detection modules disabled when running the datasets. Each time, the estimated trajectory was aligned with the ground truth trajectory by minimizing the absolute trajectory error (ATE) in the same way as [61] to compute the root mean squared error (RMSE). The averaged RMSE of the five runs was then computed and used as the RMSE of the method for the dataset. The position error (in the percentage of the trajectory length) was computed as the ratio of the RMSE to the trajectory length and used as the performance index for comparison. The results are tabulated in Table 2. It is noted that VI-ORB [7] frequently failed on these datasets. Its results are therefore not included. It can be observed that VINS-Mono [8] has a smaller mean position error (over all sequences) than OKVIS [61]. The error reduction is 45.74%. The experimental results support the selection of VINS-Mono [8] as the framework for this dissertation.

Table 2. Position error of the VI-SLAM methods on TUM VI dataset

Sequence Name	OKVIS	VINS-Mono	Length [m]
corridor1	0.227	**0.193**	305
corridor2	**0.273**	0.295	322
corridor3	**0.199**	0.445	300
corridor4	**0.257**	0.272	114
corridor5	**0.181**	0.267	270
magistrale1	0.518	**0.258**	918
magistrale2	1.437	**0.558**	561
magistrale3	0.269	**0.073**	566
magistrale4	1.190	**0.623**	688
magistrale5	0.204	**0.175**	458
magistrale6	0.934	**0.032**	771
room1	**0.049**	0.0507	146
room2	0.0787	**0.046**	142
room3	**0.049**	0.090	135
room4	**0.058**	0.062	68
room5	**0.044**	0.162	131
room6	**0.101**	0.104	67
slides1	0.658	**0.171**	289
slides2	1.525	**0.291**	299
slides3	0.638	**0.385**	383
Mean	0.445	**0.241**	347

Result of the TUM VI benchmark: RMSE in percentage (%) of trajectory length of the VI-SLAM methods. For each row, the best result is bolded.

CHAPTER 7

DEPTH ENHANCED VISUAL INERTIAL ODOMETRY

While VINS-Mono [8] is one of the state-of-the-art VI-SLAM methods, it has three disadvantages that must be addressed before it can be deployed for assistive navigation of the BVI. First, its initialization process requires sufficient VINS excitation (to estimate the metric scale and IMU biases), a condition that might not always be satisfied. Second, the metric scale estimated by the method may drift over time. Third, VINS-Mono requires a large visual parallax for SFM computation, as the visual features' depths are determined by triangulation. To overcome these problems, a new VIO method, called depth-enhanced visual-inertial odometry (DVIO), is proposed for a more accurate and robust SLAM in this Chapter. DVIO is intended for a VINS that uses a sensor suite consisting of an RGB-D camera and an IMU. It employs HPnP (see Chapter 5.2.3) to build the visual structure (by using the RGB-D camera's depth) in its state initialization and state estimation processes. As the visual features' depth data can be provided by the depth camera, and the scale can be easily determined by the depth measurements, the abovementioned problems are effectively resolved. The proposed DVIO is employed by the RC and W-ROMA for device pose estimation. Experimental results with the two assistive devices (using different R-GDB cameras RealSense D435 and Structure Core) in both laboratory and real-world scenarios validate the method's efficacy.

The remainder of this chapter is organized as the following. The first section gives a brief literature review on the related VIO algorithms. The second section details the proposed DVIO algorithm. And the third section describes the navigation

systems of the RC and W-ROMA and presents the experimental results with the two devices.

7.1 Related Depth-Visual-Inertial Odometry (DVIO) Algorithms

Early research efforts on VIO have been devoted to monocular VINS [8], [62], [63] and stereo VINS [39], [64], which use a visual (monocular or stereo) camera. RGB-D-camera-based VIO remains a relatively less-explored area. Existing RGB-D-camera-based VIO methods can be classified into two types, filtering-based and smoothing-based. For the first category, an EKF-based VIO method is introduced in [65] for pose estimation of an RGB-D-camera-based VINS. The method uses the pose change estimated by IMU preintegration to predict the state and observation and takes the pose estimated by using the visual-depth data as the actual observation to update the state. With regard to the second category, dense RGB-D-Inertial SLAM [66](by Laidlow et al.) estimates the optimal pose by minimizing an error sum consisting of the errors of photometric per-pixel alignment, geometric (point-to-plane) alignment, IMU pose measurement. The method is robust to aggressive motions as well as scenes with low photometric and/or geometric variations. As the method is computationally expensive, GPU speedup is required. In the case of a platform with limited computing power, a sparse-feature-based VIO method is preferred due to its computational efficacy. Taking the method in [67] (by Lin et al.) as an example, it first computes the camera pose from the 3D point data of ORB features [31] by using the least square method [50], then computes the VINS's initial state (including the IMU's poses, velocities, bias, and gravity direction) by aligning the visual-odometry-estimated camera poses with the IMU-estimated poses (i.e., IMU preintegration), and finally estimates the system's pose by minimizing the cost that factors in the residuals of the visual and inertial measurements. This method is extended to an

RGB-D-camera-based VINS in [68] where a 3D-2D PnP [52] method is employed to build the visual structure by using corner points [69] as visual features, and the RGB-D camera's depth data are used to estimate the motion state via a nonlinear optimization process (after the system's initialization). For the work in [67] and [68], each visual feature tracked over two image frames was assigned, at the time it was first observed, an inverse depth (obtained from the RGB-D camera), which remains constant during the iterative pose estimation process. This depth data treatment scheme ignores the RGB-D camera's depth measurement characteristics and may introduce unwanted errors to the VIO computation. As indicated in Chapter 4.2, the depth measurement error of the RGB-D camera increases with the distance. Certainly, using inaccurate depth measurements in VIO may degrade the pose change estimation result. One may prohibit the use of depth measurements beyond a threshold in VIO computation. However, this reduces the number of feature correspondences with 3D data, resulting in degraded pose accuracy. To overcome these problems, the proposed DVIO allows the state estimator to update the inverse depths for the visual features with depth measurements throughout the optimization process.

7.2 DVIO Algorithm

The proposed DVIO method consists of two components, frontend feature tracking, and backend state estimation. The feature tracking component extracts corner features [69] from an image and tracks them across images by using KLT [58]. A fundamental-matrix-based RANSAC process is implemented to remove the outliers. Keyframes are selected based on the average parallax difference and the number of tracked features, and managed by using a sliding window. The tracked features in all keyframes within the sliding window are passed to the backend process to estimate the VINS's motion state. The backend state estimator starts with a sophisticated ini-

tialization process and then proceeds with a nonlinear optimization process for state estimation.

7.2.1 Initialization

The initialization procedure consists of two stages. In the first stage, a visual structure (including the camera's poses and the positions of the tracked features) is built by a vision-only SFM process. To keep the computational cost low, only keyframes within the sliding window are used, and sparse features extracted from and tracked over these keyframes are used to build the visual structure. First, the pose change between each pair of keyframes is computed by using a Perspective-n-Point (PnP) method. Second, the depths of those visual features (on the two keyframes) that have no depth data from the RGB-D camera are computed by triangulation (using the estimated pose change). Third, the camera pose is estimated by using all visual features from the keyframes within the sliding window. Finally, a global Bundle Adjustment method is used to compute the poses for all keyframes and the features' positions by minimizing the total feature reprojection error. Since depth data are available from the RGB-D camera, a visual structure with a known scale (compared to an arbitrary scale in [8]) can be obtained. In the second stage, a visual-inertial alignment pipeline [68] is employed to estimate the VINS's initial state, including the IMU's poses, velocities, and biases.

The pose change estimation (PCE) accuracy in the first stage determines the position accuracy of the triangulated visual feature points, which affects the visual-inertial alignment. In other words, the result of the initialization process highly depends on the PnP method. VINS-RGBD [68] employs the 3D-2D-PnP method [52] to compute the PCE since depth data are available. This method, however, may result in inaccurate PCE if the number of visual features with a depth measurement

is low. To mitigate the issue, HPnP (see Chapter 5.2.3) is used in this work.

7.2.2 State Estimator

The state estimation problem can be illustrated by a factor graph model [37]. A factor graph is a bipartite graph consisting of nodes and edges. There are two types of nodes: variable nodes and factor nodes. A variable node represents the random variable to be estimated, while a factor node encodes a measurement model defined by a probabilistic distribution function (PDF) of the variable. Let the set of variables up to m nodes denoted by Θ_m. The factor graph can be denoted by $G_m = (\mathcal{F}_m, \Theta_m, \mathcal{E}_m)$, where a variable node $\theta_i \in \Theta_m$ represents an unknown random variable to be estimated; a factor node $f_i \in \mathcal{F}_m$ represents the variable's probabilistic distribution function; an edge $\varepsilon_{ij} \in \mathcal{E}_m$ indicates the connection/relation between nodes f_i and θ_j. The joint PDF of the graph G_m is factorized by:

$$\text{pdf}\left(G_m\right) = \prod_i f_i\left(\theta_i\right) \tag{7.1}$$

Assuming a Gaussian measurement model, f_i can be computed by:

$$f_i\left(\theta_i\right) \propto \exp\left(-\frac{1}{2}\left\|\mathbf{r}_i\right\|^2\right) = \exp\left(-\frac{1}{2}\mathbf{e}_i^T\boldsymbol{\Sigma}_i^{-\mathbf{1}}\mathbf{e}_i\right) \tag{7.2}$$

Here, $\left\|\mathbf{r}_i\right\|^2$ is the squared Mahalanobis distance of vector $\mathbf{r}_i$; $\mathbf{e}_i = h_i\left(\theta_i\right) - \mathbf{z}_i$ is the residual vector, representing the difference between the estimated measurement $h_i\left(\theta_i\right)$ and the actual measurement $\mathbf{z}_i$; and $\boldsymbol{\Sigma}_i$ is the covariance matrix. $\mathbf{r_i} = \boldsymbol{\Sigma}_i^{-\frac{1}{2}}\mathbf{e}_i$ is called the normalized residual vector (residual hereafter for simplicity). The measurement model f_i represents a constraint for the estimation of θ_i. The solution to the state estimation problem is to find the optimal value θ_m^* that maximizes pdf(G_m):

$$\theta_m^* = \underset{\theta_m}{\text{argmax}} \prod_i f_i\left(\theta_i\right) \tag{7.3}$$

It is equivalent to the following nonlinear least-square (LS) solution:

$$\theta_m^* = \underset{\Theta_m}{\operatorname{argmax}} \left(-\sum \log f\left(\theta_m\right) \right) = \underset{\theta_m}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left\| \mathbf{r}_i \right\|^2 \right) \tag{7.4}$$

In the proposed DVIO, this state estimation problem is solved by using the visual, IMU, and depth data associated with the keyframes within a sliding window. The state vector is defined as $\boldsymbol{\chi} = \left\{ \mathbf{x}_{b_1}^w, \mathbf{x}_{b_2}^w, \ldots, \mathbf{x}_{b_n}^w, \lambda_1, \lambda_2, \ldots, \lambda_m \right\}$. Here, $\mathbf{x}_{b_i}^w = \left\{ \mathbf{t}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{q}_{b_i}^w, \mathbf{b}_a, \mathbf{b}_g \right\}$ is the IMU's motion state consisting of the translation, velocity, rotation, accelerometer bias, and gyroscope bias for the i^{th} keyframe. n is the size of the sliding window, and m is the total number of features inside the sliding window. It is noted that the latest image frame may not satisfy the keyframe criterion, but it still is treated as if it were a keyframe and placed into the sliding window as the n^{th} keyframe. $\lambda_k \, (k = 1, \ldots, m)$ denotes the estimated inverse depth of the k^{th} visual feature. In the context of this work, visual features are used for graph construction only if they are tracked across at least two keyframes. The variable nodes of the factor graph for this problem include the pose node $\mathbf{P_i} = \left\{ \mathbf{t}_{b_i}^w, \mathbf{q}_{b_i}^w \right\}$, inverse depth node λ_k, IMU bias node $\mathbf{b_i} = \{ \mathbf{b}_a, \mathbf{b}_g \}$, and velocity node $\mathbf{v_i} = \left\{ \mathbf{v}_{b_i}^w \right\}$; and the factor nodes include the marginalization factor, preintegrated IMU factor, visual factor for a feature with no depth measurement, depth factor for a feature with the depth measurement at its first observation, depth factor for the feature with the depth measurement on a subsequent frame. Figure 10 depicts the DVIO's factor graph.

The residuals related to the factors of marginalization, IMU pre-integration, and feature reprojection (FR) [65] are denoted by ${}^{\mathbf{0}}\mathbf{r}$, ${}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}}$, and ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}}$, respectively, and the residual for the k^{th} visual feature's depth measurement at the i^{th} keyframe (where it was first observed) and that at the j^{th} (subsequent) keyframe are denoted as ${}^{\mathbf{D^i}}\mathbf{r}_{\mathbf{k}}^{\mathbf{i}}$ and ${}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}}$, respectively. The optimal state vector $\boldsymbol{\chi}^*$ for such a state estimation

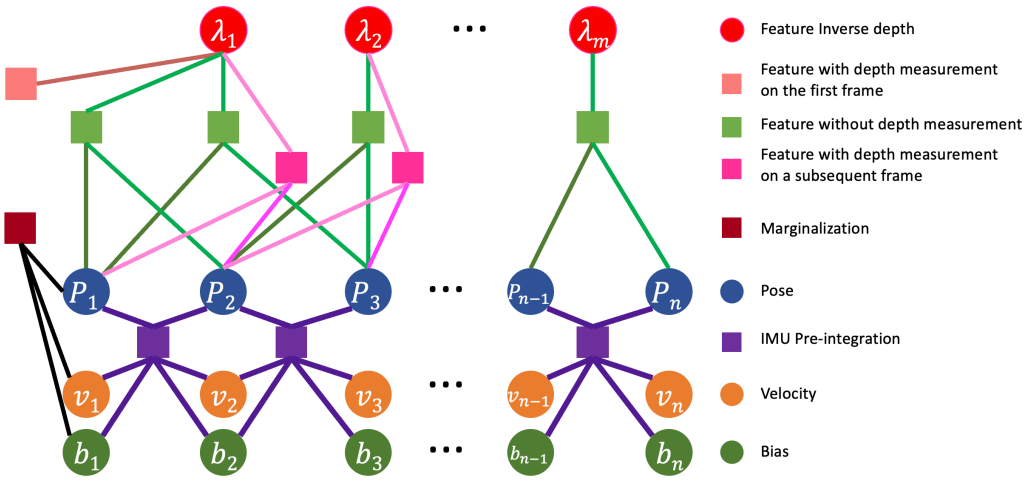


Fig. 10. Factor graph structure of DVIO: circle and square stand for variable node and factor node, respectively.

problem can be obtained by solving the least-square problem:

$$\begin{aligned}\boldsymbol{\chi}^{*} = \underset{\boldsymbol{\chi}}{\operatorname{argmin}} \Bigg(& \left\| {}^{\mathbf{0}}\mathbf{r} \right\|^{2} + \sum_{i} \left\| {}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}} \right\|^{2} \\ & + \sum_{ij} \rho \left(\left\| {}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}} \right\|^{2} \right) + \sum_{i} \left\| {}^{\mathbf{D^i}}\mathbf{r}^{\mathbf{i}}_{\mathbf{k}} \right\|^{2} + \sum_{ij} \rho \left(\left\| {}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}} \right\|^{2} \right) \Bigg)\end{aligned} \tag{7.5}$$

where $\rho(\cdot)$ is the Cauchy loss function [70].

For the k^{th} visual feature (w/o depth) that was first observed on the i^{th} keyframe $\boldsymbol{p}_{\boldsymbol{k}}^{c_i} = [u_k^{c_i}, v_k^{c_i}, 1]^T$ and then tracked onto the j^{th} frame as $\boldsymbol{p}_{\boldsymbol{k}}^{c_j} = \left[u_k^{c_j}, v_k^{c_j}, 1\right]^T$ with depth measurement $Z_k^{c_j}$, the estimated 3D coordinate can be computed by:

$$\hat{\boldsymbol{P}}_{\boldsymbol{k}}^{c_j} = \left[\hat{X}_k^{c_j}, \hat{Y}_k^{c_j}, \hat{Z}_k^{c_j}\right]^T = \mathbf{R}_{c_i}^{c_j} \boldsymbol{p}_{\boldsymbol{k}}^{c_i} / \hat{\lambda}_k^{c_i} + \mathbf{t}_{c_i}^{c_j} \tag{7.6}$$

where $\mathbf{R}_{c_i}^{c_j}$ and $\mathbf{t}_{c_i}^{c_j}$ are the rotation matrix and translation vector from $\{C_j\}$ to $\{C_i\}$, respectively.

For the k^{th} visual feature on the i^{th} frame, the feature reprojection residual is

given by:

$$^{\mathbf{FR}}\mathbf{r_k} = \Sigma_{\mathrm{c}}^{-\frac{1}{2}} \left[u_k{}^{c_j} - \frac{\widehat{X}_k^{c_j}}{\hat{Z}_k^{c_j}}, v_k{}^{c_j} - \frac{\widehat{Y}_k^{c_j}}{\hat{Z}_k^{c_j}} \right]^T \tag{7.7}$$

where Σ_c is the imagery measurement covariance, and it is given by:

$$\Sigma_c = \mathrm{diag}\left(\sigma_X^2, \sigma_Y^2\right) \tag{7.8}$$

with

$$\sigma_X = \sigma_Y = \sigma_\tau / f \tag{7.9}$$

where f is the focal length of the visual camera and σ_τ is the imagery noise. Both RealSense D435 and Structure Core (used in this work) use "active" stereo vision technology [71]. Their inverse depth noise obeys Gaussian distribution $\mathcal{N}\left(0, \sigma_\lambda^2\right)$, σ_λ^2 is a constant related to the baseline l, IR camera imagery noise σ_γ, and IR camera's focal length f_{IR}.

$$\sigma_\lambda = \sigma_\gamma / \left(f_{IR} \cdot l\right) \tag{7.10}$$

For the k^{th} visual feature on the i^{th} frame, the inverse depth residual vector $^{\mathbf{D^i}}\mathbf{r^i_k}$ is defined as:

$$^{\mathbf{D^i}}\mathbf{r^i_k} = \frac{^{\mathbf{i}}\mathbf{e_k}}{\sigma_\lambda} = \frac{1}{\sigma_\lambda}\left(\frac{1}{Z_k^{c_i}} - \hat{\lambda}_k^{c_i}\right) \tag{7.11}$$

where $\frac{1}{Z_k^{c_i}}$ is the inverse depth measurement at the first observed frame $\{C_i\}$, and $\hat{\lambda}_k^{c_i}$ is the estimated inverse-depth.

Similarly, for the k^{th} visual feature on the j^{th} frame, the residual vector $^{\mathbf{D^{ij}}}\mathbf{r^j_k}$ is defined as:

$$^{\mathbf{D^{ij}}}\mathbf{r^j_k} = \frac{^{\mathbf{j}}\mathbf{e_k}}{\sigma_\lambda} = \frac{1}{\sigma_\lambda}\left(\frac{1}{Z_k^{c_j}} - \frac{1}{\hat{Z}_k^{c_j}}\right) \tag{7.12}$$

where $\frac{1}{Z_k^{c_j}}$ is the inverse depth measurement at the frame $\{C_j\}$, and $\hat{\lambda}_k^{c_i}$ is the estimated inverse-depth computed from Equation 7.6.

From Equation 7.5, it can be seen that DVIO automatically degrades itself into

VIO if depth data are unavailable.

7.2.3 Marginalization

To restrain the computational complexity, DVIO optimizes the state variables by maintaining a sliding window with a fixed size, where the same criteria in [8] are used to determine which nodes are needed or to marginalize. To hand over the uncertainty of the marginalized node, a marginalization scheme based on Schur complement [72] is used. For a given information matrix $\mathbf{H} = \begin{bmatrix} \mathbf{H_{mm}} & \mathbf{H_{mr}} \\ \mathbf{H_{rm}} & \mathbf{H_{rr}} \end{bmatrix}$ and the corresponding residual vector $\mathbf{b} = \begin{bmatrix} \mathbf{b_m} \\ \mathbf{b_r} \end{bmatrix}$, r is the number of remaining nodes, and m is the number of marginalized nodes.

The marginalization process computes the new information and residuals for the remaining nodes:

$$\begin{aligned} \mathbf{H}_{\text{new}} &= \mathbf{H}_{rr} - \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{H}_{mr} \\ \mathbf{b}_{\text{new}} &= \mathbf{b}_{\boldsymbol{r}} - \mathbf{H}_{rm}\mathbf{H}_{mm}^{-1}\mathbf{b_m} \end{aligned} \tag{7.13}$$

The new prior is constructed based on the existing prior and all marginalized measurements related to the removed nodes. And the marginalized residual ${}^{\mathbf{0}}\mathbf{r}$ can be computed respectively.

The residual vector to be marginalized in this work include ${}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}}$ and ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}}$ as well as ${}^{\mathbf{D^i}}\mathbf{r}^{\mathbf{i}}_{\mathbf{k}}$ and ${}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}}$ if the related depth measurement(s) exists. For ${}^{\mathbf{D^i}}\mathbf{r}^{\mathbf{i}}_{\mathbf{k}}$, information matrix $\mathbf{H}$ is a diagonal matrix in which the diagonal entries as $1/\sigma_\lambda^2$. For ${}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}}$, the corresponding $\mathbf{H}$ and $\mathbf{b}$ can be computed from the Jacobian $\mathbf{J}$ by:

$$\mathbf{H} = {}^{\top}\boldsymbol{\Sigma}^{-1}\mathbf{J}\delta\chi, \mathbf{b} = -\mathbf{J}^{\top}\boldsymbol{\Sigma}^{-1}\mathbf{r} \tag{7.14}$$

where,

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \chi} = \frac{\partial \mathbf{r}}{\partial \boldsymbol{P}_{\boldsymbol{j}}^{\boldsymbol{c_j}}} \cdot \frac{\partial \boldsymbol{P}_{\boldsymbol{j}}^{\boldsymbol{c_j}}}{\partial \chi}, \tag{7.15}$$

For ${}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}}$,

$$\begin{aligned} \frac{\partial {}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}_{\mathbf{k}}}{\partial \boldsymbol{P}_{\boldsymbol{j}}^{\boldsymbol{c_j}}} &= \begin{bmatrix} 0 & 0 & -\frac{1}{(Z_k{}^{c_j})^2} \end{bmatrix} \\ \frac{\partial \boldsymbol{P}_{\boldsymbol{i}}^{\boldsymbol{c_j}}}{\partial \lambda} &= -\mathbf{R}_b^c \mathbf{R}_w^{b_j} \mathbf{R}_{b_i}^w \mathbf{R}_c^b \frac{P^{c_i}}{\lambda^2} \end{aligned} \tag{7.16}$$

where subscript/superscript w, b, and c denote the world, IMU, and camera coordinate systems, respectively.

7.2.4 DVIO Performance Evaluation

7.2.4.1 DVIO Performance Evaluation by using SC

To demonstrate its effectiveness, DVIO was compared with two state-of-the-art RGB-D-camera-based VIO methods, VINS-Fusion (the RGB-D version) [73] and VINS-RGBD [68], by using a Structure Core (SC) sensor. The experimental data were collected by handholding the SC while walking in the laboratory. The ground truth trajectories of the experiments were obtained by using the OptiTrack Motion Capture System (MCS) [74]. As shown in Figure 11, the SC sensor was placed into a 3D-printed bracket with six infrared LEDs (powered by a battery). The MCS uses these active IR markers to determine the SC's pose, which is treated as the ground truth. Five datasets (three short and two long trajectories) were acquired. The root mean square error (RMSE) of an estimated trajectory was used as the performance metric. The results are tabulated in Table 4. In each row, the smallest RMSE is bolded. The results in the table show that DVIO has the smallest RMSE in four of the five experiments, and its mean RMSE (0.269m) is smaller than that of VINS-Fusion [73] (0.359m) and VINS-RGBD [68] (0.360m). On average, DVIO reduces the

RMSE by 25.2% and 25.4%, respectively, when compared to VINS-Fusion [73] and VINS-RGBD [68]. This demonstrates that DVIO has a more accurate pose estimation accuracy than the other two methods. In addition, a qualitative result is shown in Figure 12, where the point cloud map built for dataset 3 by using the DVIO-estimated poses is rendered, and the trajectories estimated by the three methods are plotted. The quality of the map reflects the DVIO's good performance in pose estimation.

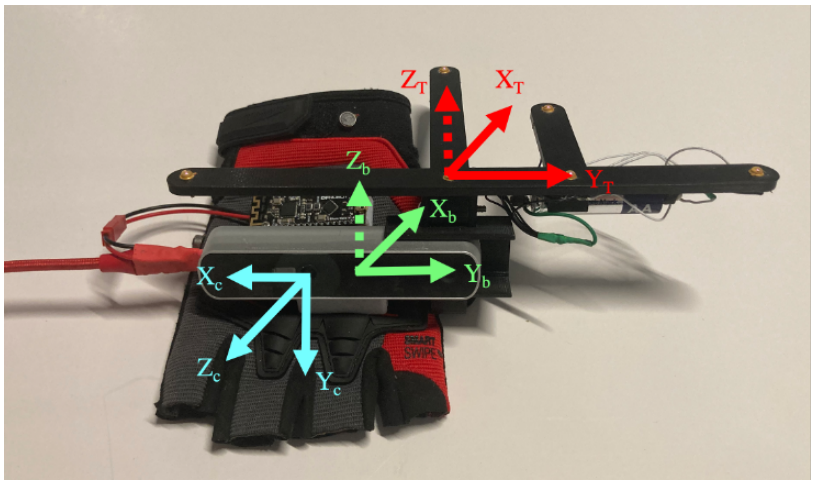


Fig. 11. Structure sensor and IR LEDs used for data collection

7.2.4.2 DVIO Performance Evaluation by using the RC

The performance of DVIO is compared with that of VINS-Mono [8] by experimentation with the RC (see Figure 1 right) that uses a sensor suite consisting of an intel RealSense D435 (camera) and a VN100 (IMU). Datasets were collected by holding the RC and walking at a speed of ∼0.7 m/s. During each data collection session, the user swung RC just like using a white cane. The ground truth positions of the start point and endpoint are $[0, 0, 0]$ and $[0, 0, 20m]$, respectively (see Figure 13). The endpoint position error norm (EPEN) is used as the metric to evaluate pose estimation accuracy. DVIO's pose estimation accuracy and computational cost can

Table 3. RMSE of the estimated trajectory of each method using SC hand-held data

Dataset	**TL (m)**	**DVIO (m)**	**VINS-Fusion (m)**	**VINS-RGBD (m)**
1	15.3	**0.134**	0.178	0.137
2	23.53	**0.12**	0.193	0.153
3	22.73	0.174	**0.166**	0.206
4	65.6	**0.43**	0.583	0.581
5	84	**0.486**	0.677	0.725
Mean	42.232	**0.269**	0.359	0.360

In each row, the best result is bolded. TL - Trajectory Length.

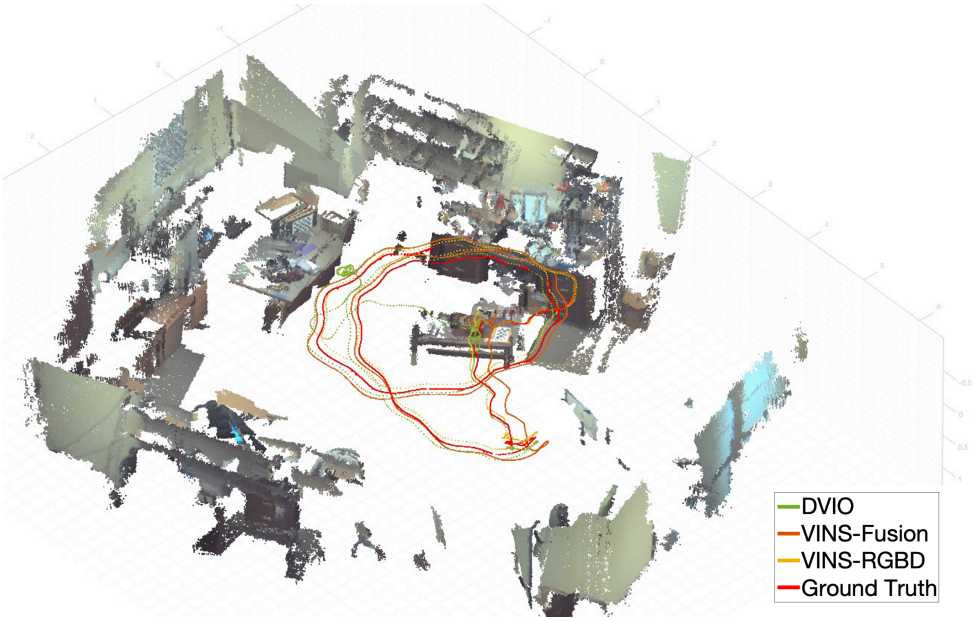


Fig. 12. The point cloud and estimated trajectories for dataset 2 by SC.

be tuned by adjusting the size of the sliding window. A small window consisting of 4 pose-nodes is used for the sake of real-time computation. And for a fair comparison, VINS-Mono also uses a 4-node sliding window, and its loop closure function is disabled. To demonstrate that the use of depth data improves pose estimation accuracy, the pose estimation results of DVIO are compared with that of VINS-Mono in Table 4. On average, it reduced the EPEN by 19%.

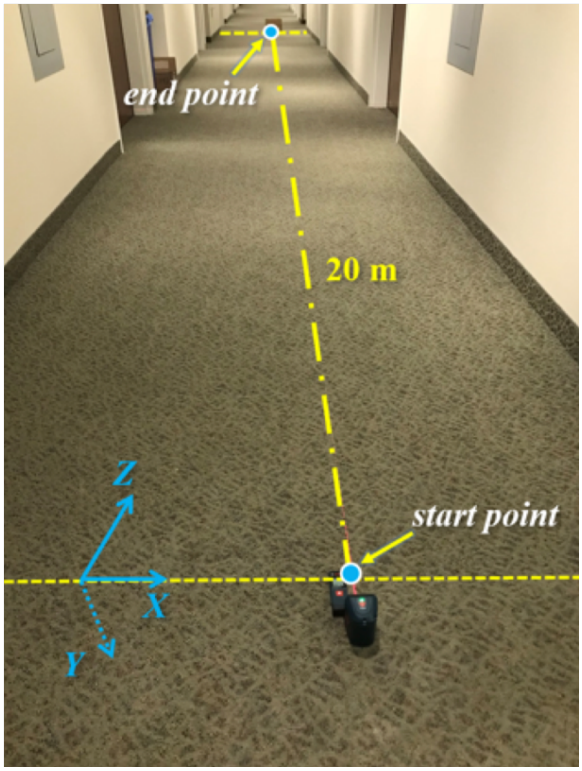


Fig. 13. Experiment settings for comparing EPENs of VINS-Mono and DVIO.

Table 4. Comparison of EPEPs of VINS-Mono and DVIO

Dataset	**VINS-Mono**		**DVIO**	
	(%)	(m)	(%)	(m)
D1	1.34	6.7	**0.87**	**4.35**
D2	1.16	5.8	**0.44**	**2.2**
D3	0.76	3.8	**0.57**	**2.85**
D4	**1.34**	**6.7**	1.74	8.7
D5	1.21	6.05	**1.09**	**5.45**
Mean	1.162	5.81	**0.942**	**4.71**

In each row, the best result is bolded.

7.3 Visual Positioning System

7.3.1 Visual Positioning System for RoboCane

7.3.1.1 RoboCane System

A visual positioning system (as depicted in Figure 1) for the RoboCane (RC) is developed to validate the proposed methods in real-world assistive navigation scenarios. The DVIO-estimated pose is used to: 1) generate a 3D point cloud map for obstacle avoidance and 2) obtain a refined 2D pose on a floor plan map by using the Particle Filter Localization (PFL) module for wayfinding. DVIO and PFL form a visual positioning system, based on which an assistive navigation system is created. The system (as depicted in Figure 14) was developed based on the robot operating system (ROS) framework. Each ROS node is an independent functional module that communicates with the others through a messaging mechanism. The Data Acquisition node acquires and publishes the camera's and the IMU's data, which are subscribed by the

DVIO node for pose estimation. The Terrain Mapping node registers the depth data captured with different camera poses to form a 3D point cloud map, which is then reprojected onto the floor plane to create a 2D local grid map for obstacle avoidance and localization of RC in the 2D floor plan. Based on the RC's location in the floor plan, the Path Planning module [27] determines the desired heading to direct RC toward the next point of Interest (PoI). This information is passed to the Obstacle Avoidance module [75] to compute the Desired Direction of Travel (DDT) that will move RC towards the PoI without colliding with the surrounding obstacle(s). Based on the DDT, the ART Controller steers RC into the DDT, and the speech interface sends audio navigation messages to the blind traveler via the Bluetooth headset. Both the tactile and audio information will guide the blind traveler to move along the planned path. The details of the major modules, such as PFL, Path Planning, Obstacle Avoidance, and ART Control, are described below.

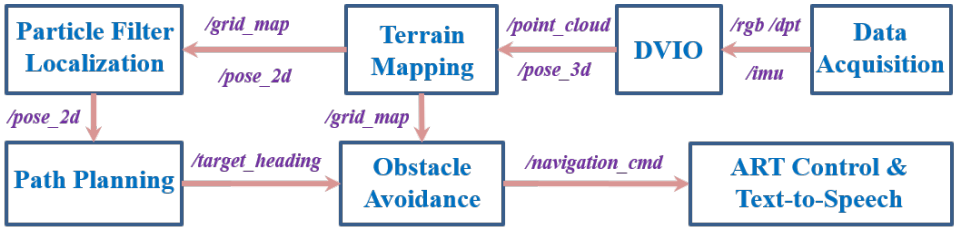


Fig. 14. Software pipeline for the assistive navigation software

7.3.1.2 Particle Filter based Localization

As an incremental state estimation method, DVIO accrues pose errors over time. When using VIO for navigation in a large space, a loop closure can be used to eliminate the accumulated pose error. However, if a loop closure cannot be detected or it is not detected in a timely fashion, the accrued pose error may become big enough to make the navigation system malfunction. To address the problem, the floor plan

of the operating environment has been used to reduce accumulative pose errors in the robotics community. The multimodal Particle Filter (PF) [76], a well-developed method, has been used for robust pose tracking with a pre-built map. Winterhalter et al. employ a 6-DOF PFL approach [77] to track the camera pose for a Google Tango tablet in an indoor environment by using the data from the device's RGB-D camera and IMU. The method utilizes the VIO-estimated motion to predict the pose for each particle. It computes an importance weight for each particle, which is proportional to the observation likelihood of the measurement given the particle's state. The likelihood value is estimated by using the difference between the actual and expected depth measurement on the floor plan given the predicted pose. A particle survives with a probability proportional to its importance weight in the re-sampling step. To adapt this idea to the application with RC, a simplified 3-DOF PFL method with a reduced particle number is proposed to estimate the RC's position and orientation on a 2D floor plan for real-time assistive navigation. The proposed method creates a local submap by registering 5 frames of depth data and aligns this map with the floor plan to determine the device pose with respect to the floor plan. As the local submap was created using multi-frame depth data, it has a higher success rate of detecting geometrical features, making the localization method more robust to depth data noise.

The proposed PFL consists of three steps, motion prediction, weight update, and resampling. First, the RC's pose change is computed from the DVIO-estimated poses at time steps $t-1$ and t, and it is used to predict the RC's pose. At time step t, the predicted pose for particle i is given by $\boldsymbol{x}_t^i = \boldsymbol{x}_{t-1}^i + \Delta\xi_t^{w'} + \mathbf{n_0}$, where $\Delta\xi_t^{w'}$ is the projected RC pose change on $X_w - Z_w$ plane and $\mathbf{n}_0 \sim \mathcal{N}(0, \mathbf{\Lambda}_0)$ is the pose noise with $\mathbf{\Lambda}_0 = \text{diag}\left(\sigma_x^2, \sigma_y^2, \sigma_\psi^2\right)$, $\sigma_x = \sigma_z = 0.03m$ and $\sigma_\psi = 3°$. Second, given the pose $\boldsymbol{x_t^i}$ and the floor plan map $\mathbf{m}$, the likelihood of making the measurement $\mathbf{z_t}$ is computed by

the sensor model $p(\mathbf{z_t} \mid \boldsymbol{x}_t^i, \mathbf{m}) \propto \prod_{\beta=-45}^{225} p(z_{t,\beta} \mid \boldsymbol{x}_t^i, \mathbf{m})$, where $p(z_{t,\beta} \mid \boldsymbol{x}_t^i, \mathbf{m})$ with $z_{t,\beta} \sim \mathcal{N}(\hat{z}_{t,\beta}, \sigma_d^2), \sigma_d = 0.2m$ is the measurement model for the pseudo laser scanner which produces 2D range measurements on $\mathbf{m}$ from $-45°$ to $225°$ (with $1°$ interval) at the current RC pose x_t^i using depth measurements. The importance weight of the i^{th} particle is then updated by $w_t^i = \eta w_{t-1}^i p(\mathbf{z}_t \mid \boldsymbol{x}_t^i, \mathbf{m})$, where η is a normalizer. Third, the adaptive strategy [78] is employed for resampling. The effective sample size $N_{\text{eff}} = 1/\sum_{i=1}^{N} (w_t^i)^2$ is used to evaluate how well the $N-$particle (N=100) set represents the target posterior. If $N_{\text{eff}} < 0.8$, a resampling operation is performed. At time step t, the output of the PF is given by $\boldsymbol{x_t} = \sum_{i=1}^{N} w_t^i \boldsymbol{x_t^i}$.

7.3.1.3 Path Planning

The Point of Interest (PoI) graph method [79] is used for path planning. The graph's nodes are the PoIs (hallway junctions, elevators, etc.), and each edge between two nodes has a weight equal to the distance between them. The A* algorithm is used to find the shortest path from the starting point to the destination. At each PoI along the path, a navigational message is generated based on the next PoI. This message is conveyed to the user by the speech interface. In addition, at each junction PoI where a turn is required, the needed heading angle change is computed as the difference between the current heading angle and the angle required to move toward the next PoI.

7.3.1.4 Obstacle Avoidance

Traversability Field Histogram (TFH) [75] method is employed to determine an obstacle-free direction for RC. First, a local terrain map surrounding RC is converted into a Traversability Map (TM). Then, a Polar Traversability Index (PTI) is computed for each $5°$ sector of the TM. The smaller the PTI, the more traversable the

direction. The PTIs are structured in the form of a histogram. Consecutive sectors with a low PTI form a histogram valley, indicating a walkable direction to RC. The valley closest to the RC's target direction is selected, and the DDT for RC is thus determined. The steering angle for RC is calculated based on the DDT and the current RC heading. The steering angle is then used to control the ART. In addition, a navigational message is generated based on the next PoI. This message is conveyed to the user via the speech interface.

7.3.1.5 ART Control

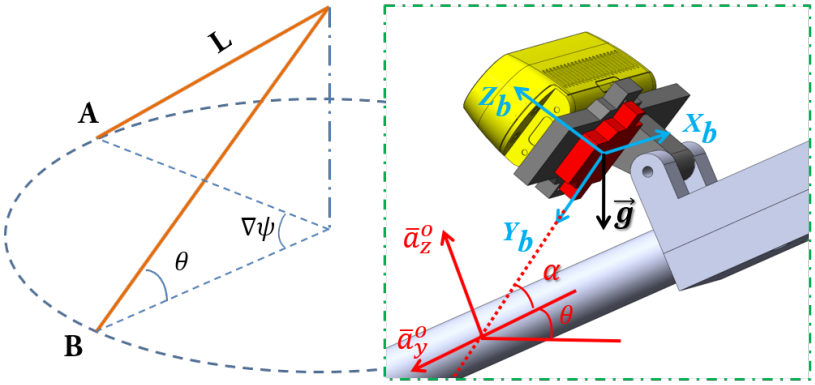


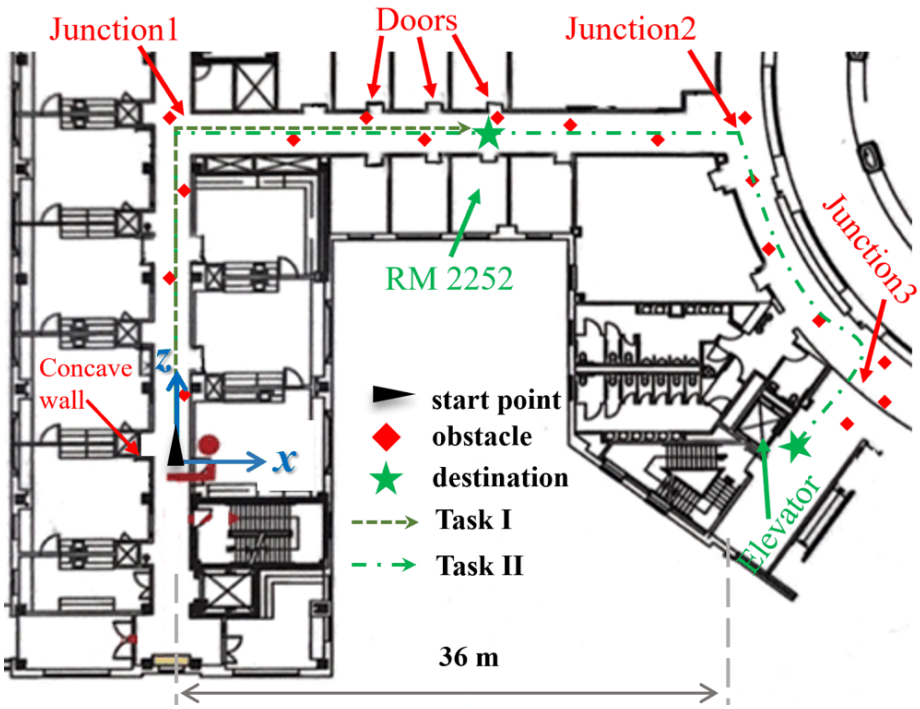
Fig. 15. left: RC swings from A to B, right: computation of θ from the accelerometer data.

To steer the rolling tip of RC from position A to B and make a heading angle change $\Delta\psi$ (see Figure 15), the required rotation of the motor is computed by $\Delta\mu = CL\Delta\psi\cos(\theta)/r$, where C, L, θ, and r are the gearhead reduction ratio, the RC's length, the RC's tilt angle, and the rolling tip's radius, respectively. This means that the RC's turning angle can be accurately controlled by the motor. In other words, RC may use its motor control system to steer itself in the desired direction for the user to follow. In this work, $C = 16$, $L = 1.47m$, and $r = 0.04m$. The tilt angle θ (see Figure 15) is mainly determined by the user's height. It may undergo a

small change when the user is walking. An initial value of θ can be estimated at the beginning of each navigation task (when the user holds the cane steadily) based on the accelerometer reading. The averaged value of the first 100 IMU readings, denoted $\bar{\boldsymbol{a}}^{\boldsymbol{b}} = \{\bar{a}_x^b, \bar{a}_y^b, \bar{a}_z^b\}$, is used to estimate the tilt angle by $\theta = \left|\arctan\left(\bar{a}_z^o/\bar{a}_y^o\right)\right|$, where $\bar{a}_y^o = \bar{a}_y^b \cos\alpha + \bar{a}_z^b \sin\alpha$ and $\bar{a}_z^o = -\bar{a}_y^b \sin\alpha + \bar{a}_z^b \cos\alpha$. α is the angle between Y_b and the cane body, and it is known a priori.

7.3.1.6 PFL Performance Evaluation with RC

To evaluate RC's localization performance, experiments are carried out by holding RC and walking along several different paths on the second floor of the Engineering East Hall of Virginia Commonwealth University. The floor plan map (as shown in Figure 16 a) is created from the architectural floor plan drawing after performing necessary editing to the doors (to show the geometric shapes of the closed doors along the paths). The distinctive geometric shapes of the areas around the doors, junctions, and corners will be used by PFL for RC localization in the floor plan. For each experiment, the target and actual Endpoints of RC were recorded, and their difference is calculated as the EPEN for performance evaluation. Table 5 summarizes the EPENs of the experiments. The trajectories estimated by PFL (i.e., DVIO + PF) and that by DVIO only are compared in Figure 17 to demonstrate the improved localization accuracy. It can be seen that PFL has a smaller EPEN for each experiment. Its mean EPEN over all experiments is 0.58%, i.e., 82.5% smaller than that of DVIO, meaning that the particle filter reduces the DVIO-accrued pose error by 82.5% on average. It is noted that the use of EPEN in the percentage of path length allows us to compute the mean value over experiments with different paths for overall performance comparison. In principle, PFL eliminates DVIO-accrued pose error whenever RC "sees" a geometrically featured region. When RC moves in a corridor (between two featured



(a) Locations of obstacles and task destinations in the floor plan

(b) Snapshots of the scenes at the start point and Junctions 1, 2

Fig. 16. Experimental settings for localization/wayfinding experiments

regions), PFL can eliminate the lateral but not the longitudinal position error. As a result, PFL's pose error is the PF alignment error plus the uncorrected DVIO pose error since the last alignment (which occurred at the last geometrically featured region). This means that the path length does not affect the EPEN of the PFL method. One can see from Table 5 that the EPEN of data sequence D6/D7 is much smaller than that of D4 even if its path length is much longer. This is because the endpoint of D6 locates at junction 1, and the last concave wall of D7 that RC "saw" is very close to the endpoint, while the elevator (endpoint for D4) is much farther from junction 3 (the last-seen feature). From the trajectory plots (Figure 17), it can be seen that the trajectories estimated by DVIO (blue lines) intersect with the walls or doors as the result caused by the accrued pose error. But the PFL method eliminated the pose errors from time to time, resulting in much more accurate trajectories (red lines).

Table 5. Comparison of EPENs: meters (% of path-length)

Data Sequence	Trajectory Length	DVIO	DVIO + PF
D1	80 m	3.42(4.28%)	**0.45(0.56%)**
D2	80 m	2.78(3.48%)	**0.85(1.06%)**
D3	80 m	1.71(2.14%)	**0.78(0.98%)**
D4	80 m	3.99(4.99%)	**0.50(0.63%)**
D5	120 m	3.72(3.10%)	**0.58(0.48%)**
D6	110 m	1.58(1.44%)	**0.17(0.15%)**
D7	190 m	7.20(3.79%)	**0.32(0.17%)**
Mean		3.20%	**0.58%**

In each row, the best result is bolded.

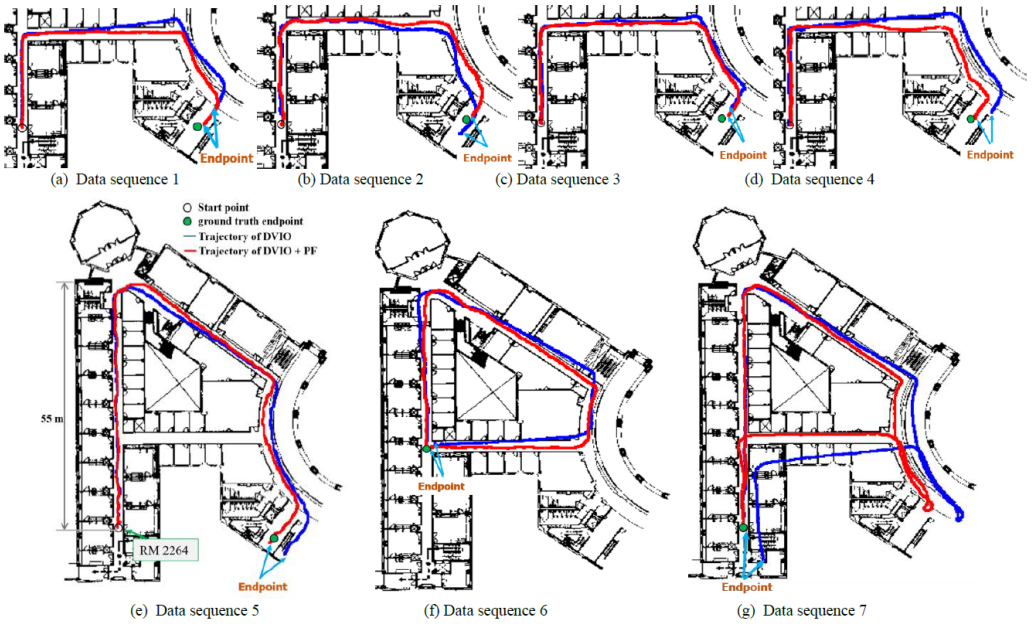


Fig. 17. Trajectories estimated by DVIO and PFL (DVIO+PF) over the architectural floor plan of the Engineering East Hall. The start point and the endpoint for D6/D7 are the same.

7.3.1.7 Wayfinding Experiments with RC

The practicality of the visual positioning system is tested by performing two navigation tasks in the Engineering East Hall. Task I is from RM 2264 to RM 2252 (path length: $\sim$ 35 meters), and task II is from RM 2264 to the elevator (path length: $\sim$ 80 meters). Two sighted persons (blindfolded) performed these tasks. Each person conducted two experiments for each task and stopped at the point when RC indicated that the destination had been reached. The EPENs (in meters) for the experiments are tabulated in Table 6. Absolute EPEN is used as the performance metric for each task. The average EPEN for tasks I and II are 0.20 m and 0.45 m, respectively. Due to the small error, RC successfully guided the users to get to the destinations in all experiments. Mean EPENs over persons and that over experiments for each task is close to the overall averaged value (0.20 m or 0.45 m), indicating a consistent

localization performance.

Table 6. EPENs of Wayfinding Experiments

Task \ Person	1	2	Mean
I (35m)	0.20 m	0.30 m	0.25 m
	0.10 m	0.20 m	0.15 m
Mean	0.15 m	0.25 m	**0.20 m**
II (80m)	0.70 m	0.50 m	0.60 m
	0.40 m	0.20 m	0.30 m
Mean	0.55 m	0.35 m	**0.45 m**

In these wayfinding experiments, numerous obstacles are placed along the paths to test the assistive navigation system's obstacle avoidance function. The results show that the obstacle avoidance module functioned well, and the ART successfully steered RC into an obstacle-free direction toward the destination. Successful obstacle avoidance reflects accurate pose estimation of PFL from a different aspect.

7.3.2 Visual Positioning System for WROMA

7.3.2.1 W-ROMA System

The proposed DVIO method was validated with W-ROMA (presented in Figure 4). The coordinate systems of W-ROMA are depicted in Figure 18. The IMU (body) and camera coordinate systems are denoted by $\{B\}$ /$(X_bY_bZ_b)$ and $\{C\}$ /$(X_cY_cZ_c)$, respectively. The initial $\{B\}$ is taken as the world coordinate system $\{W\}$. The superscripts b and c indicate a variable in $\{B\}$ and $\{C\}$, respectively. The transformation matrix from $\{B\}$ and $\{C\}$ is pre-calibrated and denoted $\mathbf{T}_c^b = \left[\mathbf{R}_c^b; \quad \mathbf{t}_c^b\right]$,

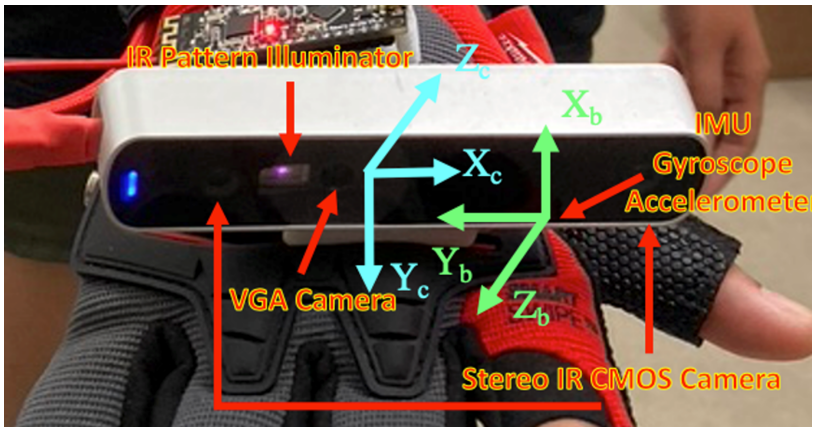


Fig. 18. IMU and camera coordinate systems for the W-ROMA

where $\mathbf{R}_c^b$ stands for the rotation and $\mathbf{t}_c^b$ the translation. Using SC's depth data, we can construct the 3D point cloud, which is denoted by $\boldsymbol{P}^{c_k}$ for the k^{th} camera frame. It can be transformed and described in C_n (i.e., the camera's coordinate system when the nth camera frame was captured) by $\boldsymbol{P}^{c_n} = \mathbf{T}_{c_k}^{c_n} \boldsymbol{P}^{c_k}$.

The hand coordinate system is denoted by $\{H\}$ $/(X_h Y_h Z_h)$ and its origin O_h is located at the image center as shown in Figure 19. The center point of the target object expressed in $\{H\}$ is denoted by $\boldsymbol{P} = [X_h, Y_h, Z_h]^T$. If $Z_h \leq 0$, the "backward" command will be generated, indicating that the user should move the hand backward. Otherwise, the system reports the distance Z_h to the user. The projection of $\boldsymbol{P}$ on the virtual image plane $(X_h O_h Y_h)$, denoted $\tilde{p}$, is used to determine the required lateral hand movement for alignment with the target object and thus the proper navigational command. For example, if $\tilde{p}$ is in the right green region, the command "right" will be generated. If $\tilde{p}$ is in the down-left blue region ([-157.5°, -112.5°]), the command will be "down left".

In the real world, object manipulation scenarios can be classified into three categories: 1) Case I: the target object is close and inside the views of both the color and depth cameras; 2) Case II: the target object is beyond the SC's depth range but

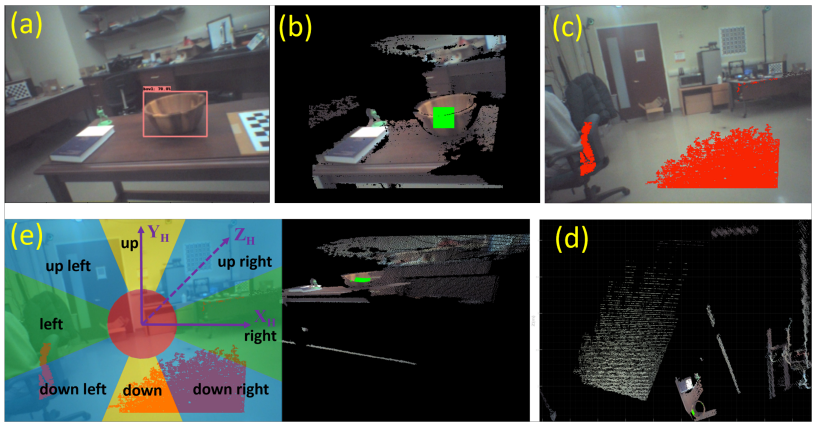


Fig. 19. W-ROMA workflow: (a) Target object (a bowl) detected on an image; (b) Depth image (texture-mapped); (c) The bowl is not visible on the image (pixels with depth data are shown in red); (d) Top (XZ) view of the point cloud data corresponding to (c); (e) The target object's center projected onto the virtual image plane of current (the entire point cloud is transformed for visualization).

inside the color camera's view; 3) Case III: the target object is outside of both color cameras' views, but it was observed and detected earlier. The guidance scheme is designed and explained as follows.

Case I: At the k^{th} camera frame, the object detection module detects the target object on the color image and determines the center of the object's bounding box denoted $\boldsymbol{P}^{c_k}$. $\boldsymbol{P}^{c_k}$ can be transformed into $\{H\}$ by $\boldsymbol{P}^{H_k} = \boldsymbol{T}_c^H \ \boldsymbol{P}^{c_k}$. Figure 19 shows an example of this scenario. First, the object detection module recognizes the bowl and returns the coordinates of a bounding box (Figure 19 (a)). Then, it is checked on the depth image (Figure 19 (b)) if there is a sufficient number of data points around the center of the bounding box (i.e., within the square image patch in green as shown in Figure 19 (b)). If yes, the center point of the target object is obtained by using the data points within the patch. The center point is then transformed into $\{H\}$ to determine the hand-object misalignment and thus the desired hand movement and

the navigation command.

Case II: W-ROMA detects the target object on the k^{th} camera image frame. The desired hand movement is computed based on the location of the object bounding box's center point. Since the object is beyond the depth camera's range, the device reports to the user that the target object is found but is in a distant place. In this case, the device will signal the user to walk toward the object. The needed command to retain hand-object alignment will be computed and conveyed to the user while s/he is walking toward the target object.

Case III: As the target object was observed and detected earlier, its point cloud is transformed into the current hand coordinate system to determine the required hand movement. For example, the user walks away from the target object (bowl) after it was observed and detected (in Figure 19 (a)), making it afar and become not visible on the camera's current image as shown in 19 (c). The center point (i.e., the center of the green square in Figure 19 (b)) that was computed earlier is transformed to the current camera coordinate system (see the green dot in Figure 19 (d)). By re-projecting the transformed center point onto the virtual image plane (as shown in Figure 19 (e)), it can be found that the bowl lies in the right sector. Therefore, the navigation command is "move right", and the right vibrating motor will vibrate for one second, signaling the hand to move right.

In summary, as long as the target has been observed and its point cloud has been generated, the system can accurately estimate the relative 3D distance and the misalignment between the target and the hand. The information will then guide the hand movement to reach the object. W-ROMA's pose must be accurately and reliably tracked to achieve this function. This will be achieved by the proposed DVIO method.

7.3.2.2 Wayfinding Experiments with W-ROMA

Five sighted volunteers were recruited to test the W-ROMA prototype for object manipulation. Each of them was asked to perform an object manipulation task five times with and without the device. The time taken to complete the task was recorded for each experiment. If a volunteer could not grasp the target object in 2 minutes or s/he picked up the wrong object, the task was terminated, and this test was considered unsuccessful. Otherwise, it is a successful one. The number of successful trials (NST) for each subject was also recorded for comparison.

The volunteers were blindfolded during the five trials. The target object was a wooden bowl (Figure 19 (a)). At the beginning of each trial, the volunteer stood in front of the bowl and was asked to move W-ROMA around slowly. This way, the system can successfully detect and locate the bowl and initialize DVIO for pose tracking. Then the volunteer was accompanied by a sighted person and walked to another place. By making the target object out of the camera's view, it was intended to test W-ROMA's three functions: wayfinding: guiding the user to walk to the vicinity of the target object; object detection: detecting the target object once it appears in the camera's view, and motion guidance: generating effective motion commands to guide the user's hand to grasp the target object. If any of these functions fails, the user may fail to grab the bowl or need more time searching for it. When a subject started to search the bowl by following the instructions from W-ROMA, the timer was set to start. W-ROMA sends the volunteers a voice command every two seconds and a vibration pattern every one second. At the beginning of each W-ROMA-aided test, the volunteer was guided by a sighted person to touch the bowl and then escorted to the navigation starting point. Then, s/he started searching for the bowl, and the timer was set to start.

Table 7. W-ROMA Humans Subject Test Results

	w/ W-ROMA		w/o W-ROMA	
Subject	NST	Mean Time (s)	NST	Mean Time (s)
1	**5**	**15.2**	3	39
2	**5**	13	2	12
3	**4**	**14**	2	31
4	**5**	**20**	1	30
5	**5**	**12**	0	X
Mean	4.8	**15.6**	1.6	29.1

NST: Number of Successful Trials, X : Failed in all five tests. In each row, the best result is bolded.

Table 7 summarizes the experimental results. It can be seen that with the assistance of W-ROMA, the total success rate of trials was improved three times, from 32% to 96%. And the average time for task completion was halved, from 29.1s to 15.6 s. The results demonstrate that our W-ROMA can effectively help the person in wayfinding and object manipulation.

CHAPTER 8

VISUAL-LIDAR-INERTIAL ODOMETRY: A NEW VISUAL-INERTIAL SLAM METHOD BASED ON AN IPHONE 12 PRO

Recently, smartphone manufacturers, such as Huawei, Samsung, Xiaomi, and Apple, integrated depth cameras into their high-end smartphones. The depth cameras are used to provide 3D point cloud data for facial recognition, enhanced autofocus, and creating artificial depth of field. In 2014, Google started Project Tango, a prototype smartphone called Peanut, which became the first one equipped with a built-in depth camera. In the same year, HTC released the first commercial smartphone, One M8, with depth-sensing capability. However, the depth sensors on these devices are mostly stereovison-based, which can only measure accurately at a very close distance. In 2018, the iPhone 12 Pro was launched with a LiDAR scanner (LiDAR hereafter) capable of measuring depth up to 5 meters with a small and relatively constant depth error. The iPhone 12 Pro's exceptional computational power and depth-sensing capability make it possible to form a phone-based DVIO system. In this chapter, a thorough characterization analysis of the LiDAR on an iPhone was conducted to evaluate the depth measurement quality. A new Visual-LiDAR-Inertial Odometry method adapted from DVIO, named VLIO, was proposed to utilize the characterization results. To the best of my knowledge, the proposed VLIO is the first of its kind in the literature. To validate the efficacy of VLIO, five experiments were conducted, and the pose estimation result of VLIO was compared with the results from Apple ARKit 4 [80]. The proposed VLIO produces more accurate pose estimations due to

its better incorporation with LiDAR's depth data.

The remaining of this chapter is constructed as the following, the first section will detail the characterization of the iPhone 12 Pro's IMU and LiDAR, the second section will present the egomotion analysis, the third section will introduce the VLIO algorithm, and the last section will present the experimental results. For the rest of this chapter, "iPhone" is referred to "iPhone 12 Pro" for simplicity.

8.1 Characterization of the iPhone's IMU and LiDAR

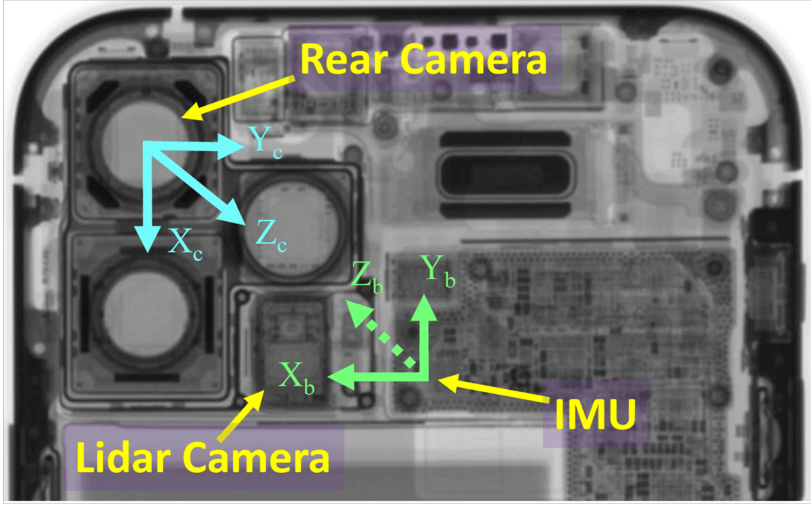
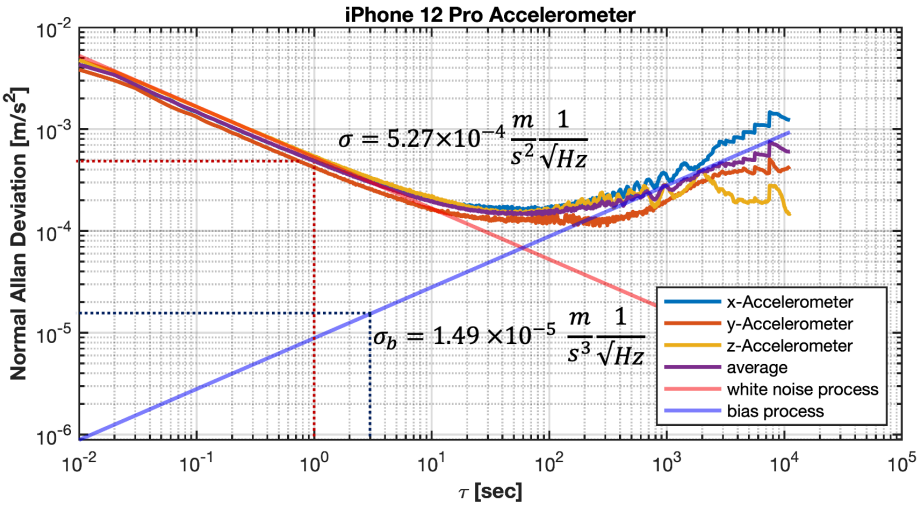


Fig. 20. X-ray image of an iPhone 12 Pro

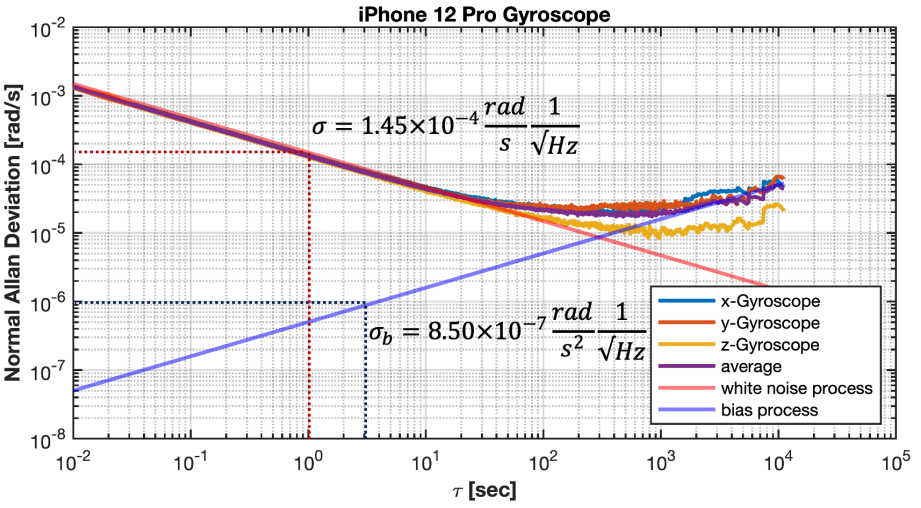
Figure 20 shows an X-ray image [81] of the iPhone. The wide-angle camera located in the top left corner is used by VLIO. The camera's and the IMU's coordinate systems are $\{C\}/(X_cY_cZ_c)$ and $\{B\}/(X_bY_bZ_b)$, respectively. The image frame rates of the color camera and LiDAR are 30 Hz and the data update of IMU is 100 Hz.

8.1.1 IMU

The Camera-IMU calibration estimation methods [82], [8] can be used to automate the estimation of camera-IMU extrinsics. However, the estimation results,



(a) Allan variance analysis on iPhone 12 Pro Accelerometer



(b) Allan variance analysis on iPhone 12 Pro Gyroscope

Fig. 21. Allan variance analysis on iPhone 12 Pro IMU

especially translation, may be quite inaccurate due to the interplay between the extrinsics and other parameters, such as IMU noise density, random walk, and camera intrinsics. In this work, we chose to directly measure the camera-IMU displacement from the X-ray image. We employed the Allan variance analysis to estimate the statistical properties of the iPhone's IMU. The results are reported in Figure 21 and tabulated in Table 8.

Table 8. Noise and Random Walk Bias of the IMU in iPhone 12 Pro

	Noise density	Random walk
Accelerometer	$5.27 \times 10^{-4} \frac{m}{s^2} \frac{1}{\sqrt{Hz}}$	$1.49 \times 10^{-5} \frac{m}{s^3 \sqrt{Hz}}$
Gyroscope	$1.45 \times 10^{-4} \frac{rad}{s}$	$8.50 \times 10^{-7} \frac{rad}{s^2} \frac{1}{\sqrt{Hz}}$

8.1.2 Visual Camera

The visual camera in use is labeled in Figure 20. It is a rolling-shutter camera with a $48^\circ \times 60^\circ$ field of view and it streams color images with a 1440×1920-pixel resolution at 30 Hz. The images have been undistorted. The Pin-hole model was used for this camera.

8.1.3 LiDAR

The LiDAR on the iPhone uses the dToF ranging technique that provides depth data ranging from 0.2 m to 5 m at a resolution of 192×256 points. The depth data fill rate is $> 95\%$. To model the depth measurement error, the camera(iPhone) was placed in front of a wall and 400 consecutive data frames were taken at each distance. The mean error and its standard deviation against the distance (ranging 0.5m -4m) are plotted in Figure 22 and Figure 23, respectively. The mean is < 2 cm, and the standard deviation is <1.5 mm.

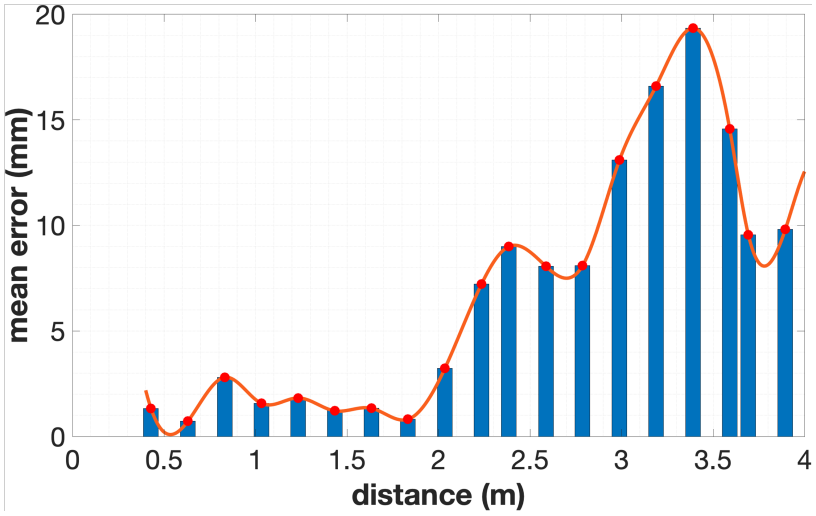


Fig. 22. Mean error of depth measurement from iPhone 12 Pro LiDAR vs. true depth

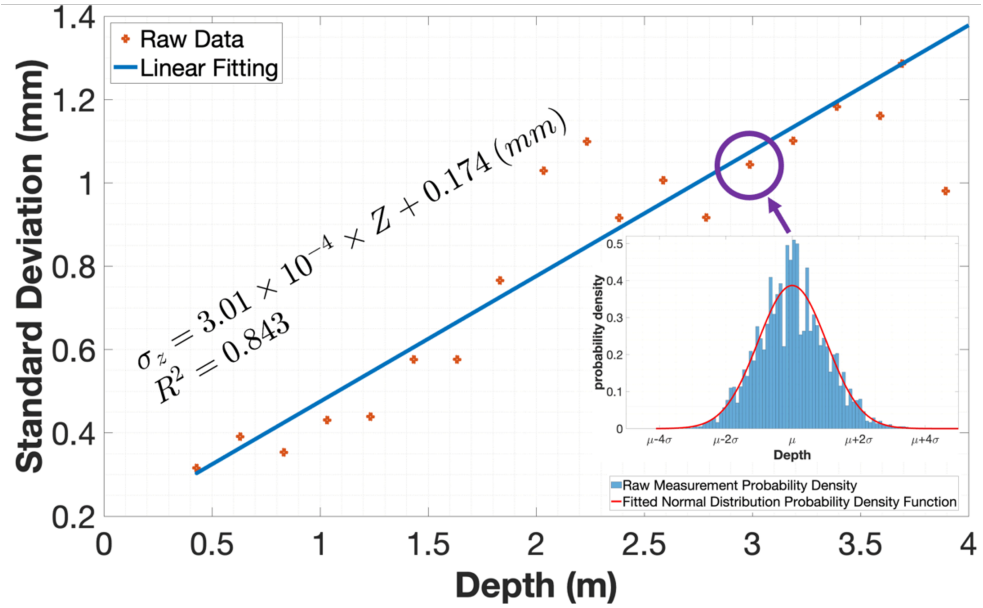


Fig. 23. Standard deviation of depth measurement error from iPhone 12 Pro LiDAR vs. true depth

A cubic spline to the data points in Figure 23 was fitted. The resulted spline function will be used to estimate the mean error for a given depth value. Linear fitting was used to approximate the relationship between the standard deviation and the depth. The following equation is the fitting result with the coefficient of determination $R^2 = 0.843$:

$$\sigma_Z = 3.01 \times 10^{-4} \times Z + 0.174 \tag{8.1}$$

Equation 23 will be used to estimate the standard deviation of depth error for a specific depth Z. It is important to note that the depth measurement error is approximately Gaussian distributed as the insertion in Figure 23. The Gaussianity determines the new depth residual formulation, which will be discussed later in this chapter.

8.2 Egomotion Analysis

HPnP was used to compute the camera egomotion with the raw LiDAR-visual data. The data were collected from the iPhone installed on a motion table that produced successive translational (X/Y/Z) movement with a 100-mm step-size or rotational (roll/pitch/yaw) movement with a 3° step-size. The environment consists of objects at various distances from 0.5 to 4 meters. 400 pairs of data frames were collected before and after each movement. HPnP extracted corner features[69] from the 1^{st} image and tracked them onto the 2^{nd} image by using the KLT tracker[58]. The mean and standard deviation of egomotion estimation error are tabulated in Tables 9 and 10. It can be seen that the rotation estimation has: 1) an extremely high repeatability ($\sigma \leq 0.05°$), owing to the LiDAR's high repeatability in depth measurement, and 2) a decent accuracy ($\mu \leq 0.5°$) except for the yaw estimates in the first group of Table 9. The error in yaw estimation can be further reduced if the

Table 9. Measurement Accuracy of Rotation

MV : (μ, σ) / TV : (ϕ, θ, ψ)	Roll $\phi\,(^\circ)$	Pitch $\theta\,(^\circ)$	Yaw $\psi\,(^\circ)$	Reprojection Error (pixels)
$(3, 0, 0)$	$(0.08, 0.01)$	$(0.01, 0.00)$	$(0.27, 0.00)$	$(0.53, 0.04)$
$(6, 0, 0)$	$(0.05, 0.01)$	$(0.03, 0.00)$	$(0.44, 0.01)$	$(0.83, 0.06)$
$(9, 0, 0)$	$(0.05, 0.01)$	$(0.04, 0.01)$	$(0.61, 0.01)$	$(1.14, 0.09)$
$(12, 0, 0)$	$(0.12, 0.01)$	$(0.04, 0.01)$	$(0.80, 0.01)$	$(1.46, 0.11)$
$(15, 0, 0)$	$(0.21, 0.02)$	$(0.03, 0.01)$	$(0.97, 0.01)$	$(1.74, 0.15)$
$(0, 3, 0)$	$(0.04, 0.01)$	$(0.03, 0.00)$	$(0.06, 0.01)$	$(0.83, 0.06)$
$(0, 6, 0)$	$(0.05, 0.01)$	$(0.01, 0.01)$	$(0.14, 0.01)$	$(1.14, 0.07)$
$(0, 9, 0)$	$(0.04, 0.02)$	$(0.13, 0.01)$	$(0.21, 0.01)$	$(1.52, 0.11)$
$(0, 12, 0)$	$(0.03, 0.03)$	$(0.12, 0.02)$	$(0.30, 0.02)$	$(1.95, 0.25)$
$(0, 15, 0)$	$(0.03, 0.04)$	$(0.07, 0.02)$	$(0.39, 0.02)$	$(2.31, 0.34)$
$(0, 0, 3)$	$(0.09, 0.01)$	$(0.05, 0.00)$	$(0.23, 0.00)$	$(0.67, 0.11)$
$(0, 0, 6)$	$(0.16, 0.01)$	$(0.03, 0.01)$	$(0.25, 0.01)$	$(1.05, 0.12)$
$(0, 0, 9)$	$(0.21, 0.02)$	$(0.04, 0.01)$	$(0.20, 0.01)$	$(1.40, 0.12)$
$(0, 0, 12)$	$(0.28, 0.03)$	$(0.09, 0.01)$	$(0.24, 0.02)$	$(1.73, 0.13)$
$(0, 0, 15)$	$(0.33, 0.04)$	$(0.10, 0.02)$	$(0.32, 0.02)$	$(1.93, 0.18)$

MV: Measurement Values, TV: True Values, μ: Mean, σ: Standard Deviation. LiDAR's angular resolution $0.27^\circ \times 0.26^\circ$

Table 10. Measurement Accuracy of Translation

MV : (μ, σ) / TV : (X, Y, Z)	X (mm)	Y (mm)	Z (mm)	Reprojection Error (pixels)
(100,0,0)	(2.31,3.47)	(2.64,1.67)	(4.37,2.75)	(2.26,0.38)
(200,0,0)	(3.64,6.21)	(3.90,3.76)	(5.76,5.64)	(4.08,0.57)
(300,0,0)	(3.91,4.41)	(4.84,4.48)	(6.73,4.99)	(5.72,0.71)
(400,0,0)	(6.00,5.86)	(6.34,4.80)	(8.13,6.12)	(7.58,1.14)
(0,100,0)	(3.99,0.37)	(0.90,0.22)	(1.66,0.43)	(0.44,0.10)
(0,200,0)	(7.81,0.64)	(1.76,0.44)	(1.23,0.86)	(0.89,0.15)
(0,300,0)	(11.32,1.01)	(2.60,0.78)	(3.61,1.90)	(1.88,0.22)
(0,400,0)	(14.29,1.63)	(3.28,1.14)	(4.62,3.27)	(3.2,0.26)

MV: Measurement Values, TV: True Values, μ: Mean, σ: Standard Deviation. LiDAR's distance accuracy: <20mm

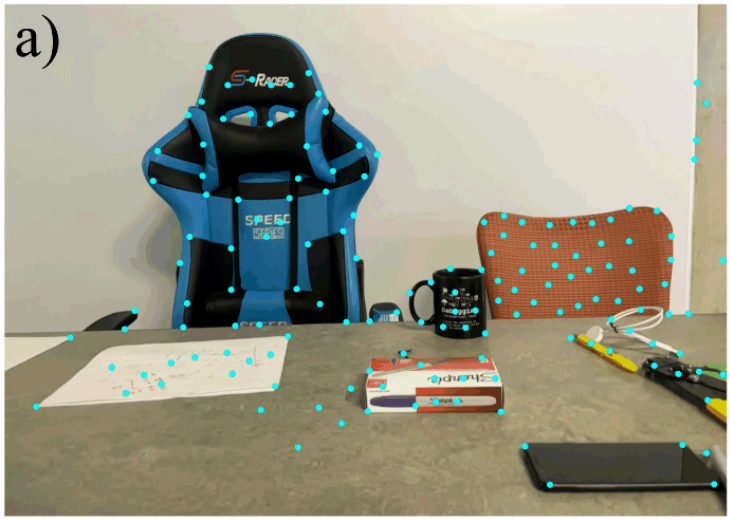
depth bias is compensated by using the mean error and more accurate camera intrinsic parameters (on both frames) are applied. It can also be observed that both σ and μ for translation estimations are several millimeters, which conforms to the LiDAR's small mean error (as shown in Figure 22). Similar results for movements with combined rotation and translation were observed. Finally, the smaller the phone movement, the smaller the egomotion estimation error. Since VLIO uses the reprojection error of each visual feature in its cost function, the re-projection error was characterized, and the results were shown in the 5^{th} column of Table 10. It is noted that each value represents the average overall matched visual features of an image pair. The characterization results are also useful to some SLAM methods that require computing the egomotion between each pair of the pose nodes.

8.3 Visual-LiDAR-Inertial Odometry (VLIO)

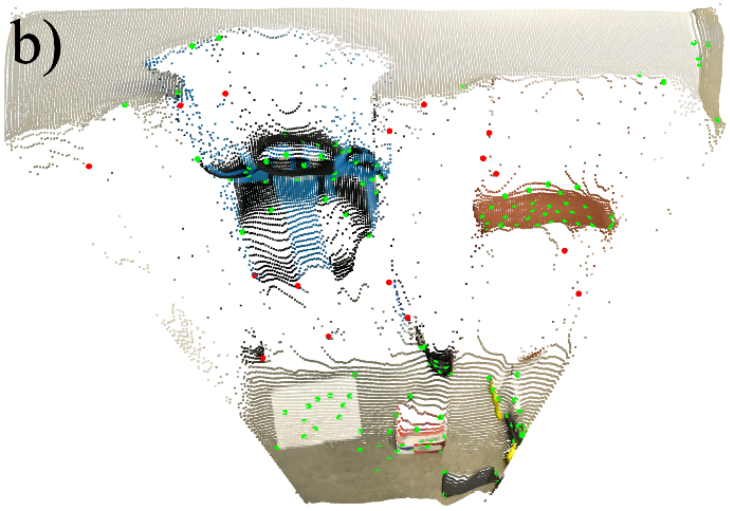
VLIO consists of two components, front-end visual odometry, and back-end state estimation. The front-end computes the camera's egomotion and tracks the visual features for the backend, which closely couples the visual, LiDAR, and IMU data to determine the optimal motion state that minimizes a cost function. In the front-end visual odometry, HPnP is used for initial pose estimation. Due to the working principle of LiDAR, there are mixed pixels that degrade the depth-related pose estimation results. Thus, a mixed pixels detection and removal module was applied to the raw depth data before HPnP computation.

8.3.1 Mixed pixels detection and removal

The iPhone's LiDAR data entail mixed measurements when a data point locates at the edge of an object. Mixed measurements, also known as mixed pixels, are a well-known issue with LiDARs in the robotics literature [83], [84]. As shown in Figure 24, a mixed pixel occurs at an object's edge because part of the light is reflected from the front object and part from the background, resulting in a mixed depth measurement in-between [85]. As a mixed pixel is along the light ray, it satisfies epipolar constraint and thus cannot be removed by the RANSAC process in HPnP. To deal with the problem, we developed a filter for mixed pixel removal. The filter first uses a Sobel edge detector [86] to locate objects' edges in the depth image. It then detects mixed pixels along each edge by comparing each edge-point's depth with that of its lateral neighbors (i.e. one on each side of the edge). A substantial depth difference with the neighbors indicates a mixed pixel. The depth value is then replaced by the depth of the neighbor that is closer. The filtered LiDAR data, after subtracted by the bias (mean error as shown in Figure 22), are then used by HPnP for egomotion estimation.



(a) Detected visual features: The visual features detected on an image are illustrated as cyan dots.



(b) The texture-mapped point cloud: red and green dots represent the feature points w/ and w/o a mixed measurement.

Fig. 24. Mixed pixels in the iPhone's LiDAR data

8.3.2 Graph Optimization

Camera egomotion between two consecutive frames is computed by HPnP. An image frame with sufficient parallax is identified as a keyframe. In this work, ten keyframes are used to initialize VLIO. A global Bundle Adjustment (BA) method [87] is used to refine the keyframes' poses and the related features' positions (obtained from the LiDAR data when they were observed the first time) by minimizing the total feature re-projection error. The visual structure optimized by BA is then aligned with the IMU-estimated structure [68] to determine the VINS' initial state. Due to the small egomotion error and σ_z values, the initialization process results in a very accurate initial state in a very short time.

After the initialization, an iterative optimization process is employed to solve the nonlinear state estimation problem of the VINS by using the keyframes and the associated IMU data within a sliding window. Similar to DVIO, the state vector of VLIO is defined as $\boldsymbol{\chi} = \left\{\mathbf{x}_{b_1}^w, \mathbf{x}_{b_2}^w, \ldots, \mathbf{x}_{b_n}^w, \lambda_1, \lambda_2, \ldots, \lambda_m\right\}$. Here, $\mathbf{x}_{b_i}^w = \left\{\mathbf{t}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{q}_{b_i}^w, \mathbf{b}_a, \mathbf{b}_g\right\}$ is the IMU's motion state consisting of the translation, velocity, rotation, accelerometer bias, and gyroscope bias for the i^{th} keyframe. n is the size of the sliding window, and m is the total number of features inside the sliding window. $\lambda_k \, (k = 1 \ldots m)$ denotes the estimated inverse depth of the k_{th} visual feature. $\mathbf{x}_{b_i}^w$ consists of three variable nodes: $\mathbf{P_i} = \left\{\mathbf{t}_{b_i}^w, \mathbf{q}_{b_i}^w\right\}$, $\boldsymbol{v_i} = \left\{\mathbf{v}_{b_i}^w\right\}$, and $\boldsymbol{b_i} = \{\mathbf{b}_a, \mathbf{b}_g\}$ in the factor graph.

The optimal state vector $\boldsymbol{\chi}$ can be solved by the least-squares problem:

$$\begin{aligned}\boldsymbol{\chi}^* = \underset{\boldsymbol{\chi}}{\operatorname{argmin}} \Bigg(& \left\| {}^{\mathbf{0}}\mathbf{r} \right\|^2 + \sum_i \left\| {}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}} \right\|^2 + \sum_{ij} \rho \left(\left\| {}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}} \right\|^2 \right) \\ & + \sum_i \left\| {}^{\mathbf{D^i}}\mathbf{r}_{\mathbf{k}}^{\mathbf{i}} \right\|^2 + \sum_{ij} \rho \left(\left\| {}^{\mathbf{D^{ij}}}\mathbf{r}^{\mathbf{j}}{}_{\mathbf{k}} \right\|^2 \right) \Bigg)\end{aligned} \tag{8.2}$$

where ${}^{\mathbf{0}}\mathbf{r}$, ${}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}}$and ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}}$ are the normalized residual vectors related to the factors

of marginalization, pre-integrated IMU, and FR respectively [8]. Subscripts i and j represent the i^{th} and j^{th} keyframes, respectively. $^{\mathbf{D^i}}\mathbf{r^i_k}$ and $^{\mathbf{D^{ij}}}\mathbf{r^j_k}$ are the residual of the k_{th} visual feature's depth at the keyframe where it was first observed and the residual of its 3D position in a subsequent keyframe, respectively. $\rho(\cdot)$ is the Cauchy loss function [70].

For the k^{th} visual feature on the i^{th} keyframe, the residual $^{\mathbf{D^i}}\mathbf{r_k}$ is given by:

$$^{\mathbf{D^i}}\mathbf{r_k} = \frac{^{\mathrm{i}}\mathrm{e}_k}{\sigma_z} = \frac{1}{\sigma_z}(Z_k - \frac{1}{\hat{\lambda}_k}) \tag{8.3}$$

where Z_k is the actual depth measurement provided by the iPhone's LiDAR, and $\hat{\lambda}_k$ is the estimated inverse-depth. As the LiDAR's depth error follows Gaussian distribution $\mathcal{N}(0, \sigma_z^2)$, where σ_z is estimated by using Equation 8.1. For the k^{th} visual feature that was first observed on the i^{th} frame as $\boldsymbol{p}_{\boldsymbol{k}}^{c_i} = [u_k^i, v_k^i, 1]^T$ and then tracked onto the j^{th} frame as $\boldsymbol{p}_{\boldsymbol{k}}^{c_j} = \left[u_k^j, v_k^j, 1\right]^T$ with depth measurement $Z_k^{c_j}$, the estimated 3D coordinate can be computed by:

$$\widehat{\boldsymbol{P}}_{\boldsymbol{k}}^{j} = \left[\hat{X}^{c_j}, \hat{Y}^{c_j}, \hat{Z}^{c_j}\right]^T = \mathbf{R}_{c_i}^{c_j}\boldsymbol{p}_{\boldsymbol{k}}^{c_i}/\hat{\lambda}_k^{c_i} + \mathbf{t}_{c_i}^{c_j} \tag{8.4}$$

where $\mathbf{R}_{c_i}^{c_j}$ and $\mathbf{t}_{c_i}^{c_j}$ are the rotation matrix and translation from $\{C_j\}$ to $\{C_i\}$.

With depth estimate $\hat{Z}_k^{c_j}$ at $\{C_j\}$, the residual vector $^{\mathbf{D^{ij}}}\mathbf{r^j_k}$ can be computed accordingly:

$$^{\mathbf{D^{ij}}}\mathbf{r^j_k} = \frac{1}{\sigma_z}(Z_k^{c_j} - \hat{Z}_k^{c_j}) \tag{8.5}$$

8.4 Experiments

The pose estimation results of VLIO under several conditions were compared to show the usefulness of the characterization results and the mixed pixel removal method. These conditions include: 1) raw (μ, σ): raw LiDAR data after subtraction of μ with σ determined by Equation 8.1; 2) filtered (μ): LiDAR data after removal

of mixed pixels and subtraction of μ, with an arbitrary constant σ ($\sigma = 10cm$) ; 3) filtered (μ, σ): the same data as 2 with σ determined by Equation 8.1. The experimental results (for type 3 data) were also compared with Apple ARkit 4, the only available benchmark at the time that uses LiDAR data.

The iPhone's image, LiDAR, and inertial data were collected by an operator hand-holding it when walking along different trajectories in the laboratory. The ground truth trajectories of the iPhone were obtained by the OptiTrack Motion Capture System (MCS)[74]. The Root Mean Square Error (RMSE) of the estimated trajectories was used to compare the VLIO performance under different conditions. The results on five of the datasets are shown in Table 11. In each row, the smallest RMSE is bolded.

Table 11. Means and RMSE of the estimated trajectories under different conditions

Dataset	**Unit(m)**	**Raw**(μ, σ)	**Filtered**(μ)	**Filtered**(μ, σ)	**ARkit**	**VINS-RGBD**
Data 1	Mean	0.107	0.107	**0.090**	0.187	0.171
18m	RMSE	0.117	0.116	**0.101**	0.204	0.193
Data 2	Mean	0.111	0.111	**0.110**	0.367	0.228
18m	RMSE	0.123	0.124	**0.122**	0.399	0.252
Data 3	Mean	0.113	0.113	**0.112**	0.309	0.358
28m	RMSE	0.138	0.139	**0.138**	0.343	0.340
Data 4	Mean	0.111	0.115	**0.110**	0.137	0.258
37cm	RMSE	0.118	0.123	**0.118**	0.149	0.275
Data 5	Mean	0.230	0.230	**0.179**	0.332	0.574
39m	RMSE	0.297	0.297	**0.248**	0.344	0.615
Average	Mean	0.50%	0.50%	**0.45%**	1.07%	1.12%
	RMSE	0.57%	0.58%	**0.53%**	1.16%	1.23%

In each row, the best result is bolded.

As can be seen, 1) the mixed pixels removal method consistently improves the result: 6% and 3.5% reduction in mean error and RMSE, respectively (the 5^{th} column versus the 3^{rd} column); 2) the use of σ computed by Equation 8.1 results in a similar improvement than the use of an over-estimated σ (σ=10 mm) (the 5^{th} column versus the 4^{th} column). VLIO performed much better than ARkit 4 in terms of pose estimation accuracy. Its average mean error and RMSE are less than half of that of the ARkit 4, and VINS-RGBD [68]. Figure 25 shows the point cloud map built for dataset 5 and the estimated trajectory by using the VLIO-estimated poses. The high-quality map and the very small estimation error of the endpoint indicates that VLIO is very accurate in pose estimation. Among all three trajectories, the one produced by VLIO is much closer to the ground truth compared to the others.

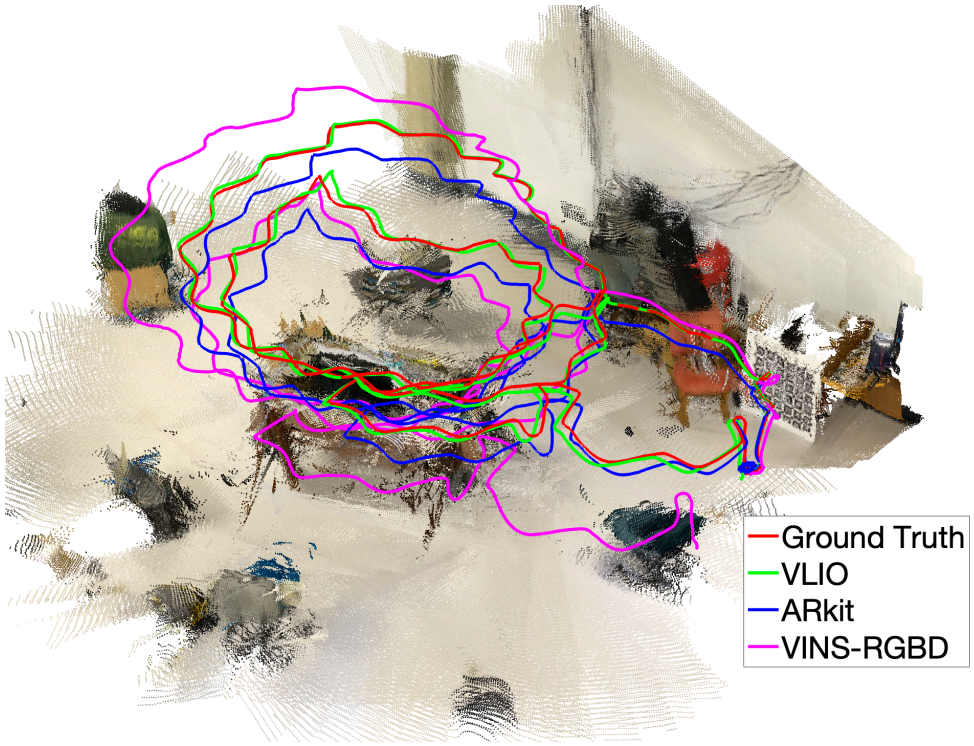


Fig. 25. The point cloud and estimated trajectories for dataset 5. The ground truth trajectory begins and ends at the same point.

CHAPTER 9

CAMERA INTRINSIC PARAMETERS ESTIMATION AIDED VISUAL-INERTIAL ODOMETRY

In the last chapter, a smartphone-based VLIO was proposed as an extension of DVIO. Using a smartphone as both the sensing and computing platforms may help to reduce the size of VINS and result in a more portable navigation aid for the BVI. However, a modern smartphone's camera uses optical image stabilization (OIS) mechanism in its lens to stabilize the image and reduce hand tremor induced image blur. The OIS mechanism results in varying camera intrinsic parameters (CIP), which, if not estimated online, may affect pose estimation accuracy. To address this issue, a linear model is first developed to relate the CIP with the IMU-measured device acceleration. Based on the model, a new VIO method, called CIP-VMobile, is introduced. CIP-VMobile treats CIP as the state variables and tightly couples them with other state variables in a graph optimization process to estimate the optimal state. The method uses the values determined by the CIP-acceleration model as an initial CIP to speed up the VIO computation. To simplify the case, the method uses only visual and inertial data (i.e., no depth data is used). A RoboCane (RC) was fabricated by using an iPhone 7 (see Figure 3 left) and used for experimental validation of the proposed method in real-world navigation scenarios.

9.1 Optical Image Stabilization (OIS)

A smartphone's camera uses a small imaging sensor and therefore requires a long exposure time. Hand tremors during the exposure time can alter the optical path of

the object being imaged and results in a blurred image. The OIS mechanism stabilizes the image by using an actuator to shift the lens barrel to counteract the optical path movement. Currently, the most widespread actuator is based on the voice coil motor (VCM)[88], [89], which produces a force by running a current through the coil winding amid the magnetic field. While it improves the image and video quality, the OIS mechanism results in varying CIP, which may result in unwanted pose estimation error if not considered by a SLAM method.

For a VCM-based OIS smartphone camera, the lens is connected to a mechanical support that is anchored to the chassis by springs. The springs allow for the lens's translation and/or rotation, resulting in varying CIP. According to Hooke's law, the extension/compression of the springs is linearly proportional to the exerted force. As the force is linearly related to the acceleration that the accelerometers can measure, we use a linear model to estimate the CIP based on the accelerometer data. As a result, the CIP for the k^{th} keyframe is given by:

$$\eta_{\mathbf{k}} = L\left(\mathbf{a_k}\right) = \langle \boldsymbol{\alpha}, \mathbf{a_k} \rangle + \boldsymbol{\beta} \tag{9.1}$$

where $\mathbf{a_k}$ is the accelerometer reading, $\langle \cdot, \cdot \rangle$ stands for element-wise multiplication, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the coefficients whose values are determined by experiment.

9.2 CIP-VMobile Algorithm

The CIP-VMobile method is developed using the framework of VINS-Mobile, extending it by adding CIP into the state vector for online estimation. Moreover, a linear model was built to relate the CIP to the accelerations and used to constrain the solution to the optimization problem. CIP-VMobile has two major modules: front-end feature tracking and back-end state estimation. The front-end module is the same as that of VINS-Mobile, while the back-end module is different, and it is

described as follows.

9.2.1 Factor Graph

CIP-VMobile uses keyframes to estimate the poses by optimizing a factor graph. At the time when the i^{th} keyframe is captured, the state vector of the VIO problem is defined as $\Theta_k = \left\{\mathbf{x}_{b_1}^w, \mathbf{x}_{b_2}^w, \ldots, \mathbf{x}_{b_n}^w, \zeta_1, \zeta_2, \ldots, \zeta_n, \lambda_1, \lambda_2, \ldots, \lambda_m\right\}$. n is number of keyframes, and m is the total number of features with the sliding window. Here, $\mathbf{x}_{b_i}^w = \left\{\mathbf{t}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{q}_{b_i}^w, \mathbf{b}_a, \mathbf{b}_g\right\}$ is the IMU's motion state consisting of the translation, velocity, rotation, accelerometer bias, and gyroscope bias. $\zeta_k = \left\{f_c^{c_i}, c_x^{c_i}, c_y^{c_i}\right\}$ is the camera intrinsic vector including the focal length and principle point for the i^{th} keyframe, and $\lambda_k\,(k = 1, \ldots, m)$ denotes the estimated inverse depth of the k^{th} visual feature. $\mathbf{x}_{b_k}^w$ consists of four variable nodes: $\mathbf{P_i} = \left\{\mathbf{t}_{b_i}^w, \mathbf{q}_{b_i}^w\right\}$, $\zeta_i = \left\{f_c^{c_i}, c_x^{c_i}, c_y^{c_i}\right\}$, $\boldsymbol{v_i} = \left\{\mathbf{v}_{b_i}^w\right\}$, and $\boldsymbol{b_i} = \{\mathbf{b}_a, \mathbf{b}_g\}$ in the factor graph.

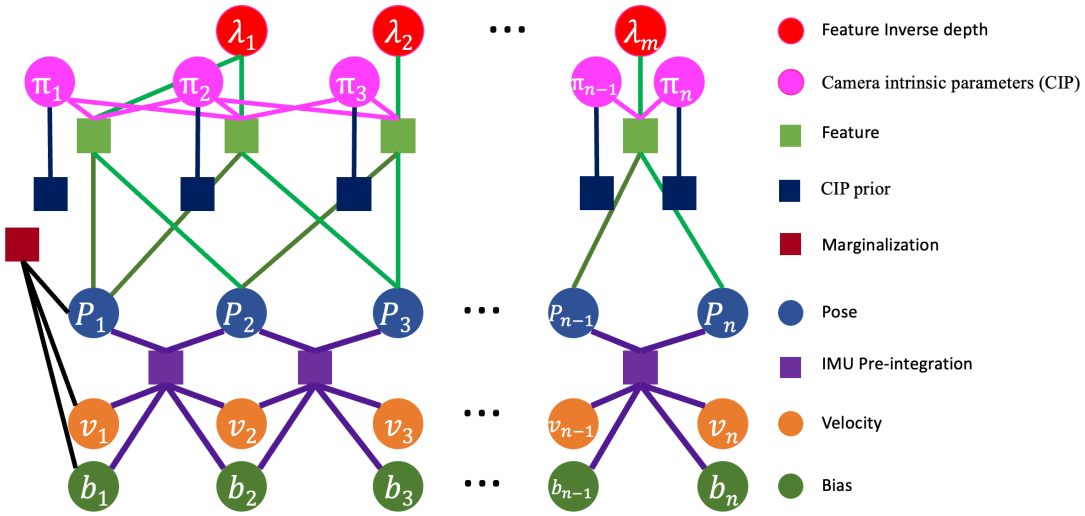


Fig. 26. Factor graph structure of CIP-VMobile: circle and square stand for variable node and factor node, respectively

Figure 26 depicts the factor graph, and the optimal pose can be obtained by

minimizing the residual error of the factor graph:

$$\Theta_m^* = \underset{\Theta_m}{\operatorname{argmin}} \left(\left\| {}^{\mathbf{0}}\mathbf{r} \right\|^2 + \sum_i \left\| {}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}} \right\|^2 + \sum_{ij} \left\| {}^{\mathbf{FR}}\mathbf{r}_{\mathbf{ij}}{}^2 \right\| + \sum_i \left\| {}^{\mathbf{CIP}}\mathbf{r}_{\mathbf{ij}} \right\|^2 \right) \tag{9.2}$$

where ${}^{\mathbf{0}}\mathbf{r}$, ${}^{\mathbf{IMU}}\mathbf{r}_{\mathbf{i,i+1}}$, ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{ij}}$ and ${}^{\mathbf{CIP}}\mathbf{r}_{\mathbf{ij}}$ are the normalized residual vectors related to the factors of marginalization, pre-integrated IMU, feature reprojection (FR), and CIP prior, respectively.

9.3 Feature reprojection Factor

Let the k^{th} feature point that was first observed at the i^{th} keyframe be denoted as $\boldsymbol{p}^{\boldsymbol{c_i}} = [u^{c_i}, v^{c_i}, 1]^T$, where (u^{c_i}, v^{c_i}) are the feature coordinates in $\{C_i\}$. The estimated inverse depth for the feature point is λ. If the feature point is tracked onto the j^{th} keyframe as $\boldsymbol{p}^{\boldsymbol{c_j}} = [u^{c_j}, v^{c_j}, 1]^T$, the tracked visual feature from $\{C_i\}$ to $\{C_j\}$ is computed as:

$$\hat{\boldsymbol{P}}^{\boldsymbol{c_j}} = \left[\hat{X}^{c_j}, \hat{Y}^{c_j}, \hat{Z}^{c_j} \right]^T = \mathbf{R}_{c_i}^{c_j} \frac{\pi_i^{-1}\left(\boldsymbol{p}^{\boldsymbol{c_i}}\right)}{\lambda} + \mathbf{t}_{c_i}^{c_j} \tag{9.3}$$

where

$$\mathbf{R}_{c_i}^{c_j} = \left(\mathbf{R}_{b_j}^{w} \mathbf{R}_{c}^{b} \right)^T \mathbf{R}_{b_i}^{w} \mathbf{R}_{c}^{b}, \text{ and } \mathbf{t}_{c_i}^{c_j} = \left(\mathbf{R}_{b_j}^{w} \mathbf{R}_{c}^{b} \right)^T \left(\mathbf{R}_{b_i}^{w} \mathbf{t}_{c}^{\mathrm{b}} + \mathbf{t}_{b_i}^{w} - \mathbf{R}_{b_j}^{w} \mathbf{t}_{\mathrm{c}}^{\mathrm{b}} - \mathbf{t}_{b_j}^{w} \right) \tag{9.4}$$

$\pi_i^{-1}\left(\boldsymbol{p}^{\boldsymbol{c_i}}\right)$ denotes the inverse camera perspective transformation of $\boldsymbol{p}^{\boldsymbol{c_i}}$ that is given by:

$$\pi_i^{-1}\left(\boldsymbol{p}^{\boldsymbol{c_i}}\right) = \left[\left(u^{c_i} - c_x^{c_i}\right) / f_c^{c_i}, \left(v^{c_i} - c_y^{c_i}\right) / f_c^{c_i}, 1 \right]^T \tag{9.5}$$

Then, the residual vector of the FR factor ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}}$, can be defined by ${}^{\mathbf{FR}}\mathbf{r}_{\mathbf{k}} = \Sigma_{\mathrm{ij}}^{-\frac{1}{2}} \mathbf{e}_{\mathbf{ij}}$. where:

$$\mathbf{e}_{\mathbf{ij}} = \left(\pi_j^{-1}\left(\boldsymbol{p}^{\boldsymbol{c_j}}\right) - \frac{\hat{\boldsymbol{P}}^{\boldsymbol{c_j}}}{\hat{Z}^{c_j}} \right)_2 \tag{9.6}$$

$\mathrm{w}(\cdot)_2$ represents the first two entries of the vector. The covariance matrix is defined as a diagonal matrix:

$$\mathbf{\Sigma_{ij}} = \mathrm{diag}\left[\sigma_\gamma^2/\left(f_c^{c_j}\right)^2, \sigma_\gamma^2/\left(f_c^{c_j}\right)^2\right] \tag{9.7}$$

where σ_γ is the imagery noise. The Jacobians of $\mathbf{e_{ij}}$ with respect to pose variables $\mathbf{P_i}$, $\mathbf{P_j}$, and inverse depth λ_k, $(k = 1, .., n)$ are defined in [8], and the Jacobians with respect to CIP ζ_i and ζ_j are given by:

$$\frac{\partial \mathbf{e_{ij}}}{\partial \zeta_i} = \frac{\partial \mathbf{e_{ij}}}{\partial \hat{\boldsymbol{P}}^{\boldsymbol{c_j}}} \cdot \frac{\partial \hat{\boldsymbol{P}}^{\boldsymbol{c_j}}}{\partial (\pi_i^{-1}(\boldsymbol{p}^{\boldsymbol{c_i}}))} \cdot \frac{\partial (\pi_i^{-1}(\boldsymbol{p}^{\boldsymbol{c_i}}))}{\partial \zeta_i} \tag{9.8}$$

where

$$\begin{aligned}
\frac{\partial \mathrm{e_{ij}}}{\partial \zeta_j} &= -\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{c_x^{c_j} - u^{c_j}}{\left(f_c^{c_j}\right)^2} & -\frac{1}{f_c^{c_j}} & 0 \\ \frac{c_y^{c_j} - v^{c_j}}{\left(f_c^{c_j}\right)^2} & 0 & -\frac{1}{f_c^{c_j}} \\ 0 & 0 & 0 \end{bmatrix} \\
\frac{\partial \mathbf{e_{ij}}}{\partial \boldsymbol{p}'_j} &= \begin{bmatrix} \frac{1}{\hat{Z}^{c_j}} & 0 & -\frac{\hat{X}^{c_j}}{\left(\hat{Z}^{c_j}\right)^2} \\ 0 & \frac{1}{\hat{Z}^{c_j}} & -\frac{\hat{Y}^{c_j}}{\left(\hat{Z}^{c_j}\right)^2} \end{bmatrix} \\
\frac{\partial \hat{\boldsymbol{P}}^{\boldsymbol{c_j}}}{\partial (\pi_i^{-1}(\boldsymbol{p}^{\boldsymbol{c_i}}))} &= \frac{\mathbf{R}_{c_i}^{c_j}}{\lambda} \\
\frac{\partial (\pi_i^{-1}(\boldsymbol{p}^{\boldsymbol{c_i}}))}{\partial \zeta_i} &= \begin{bmatrix} \frac{c_x^{c_i} - u^{c_i}}{\left(f_c^{c_i}\right)^2} & -\frac{1}{f_c^{c_i}} & 0 \\ \frac{c_y^{c_i} - v^{c_i}}{\left(f_c^{c_i}\right)^2} & 0 & -\frac{1}{f_c^{c_i}} \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{9.9}$$

9.4 CIP prior factor

The residual vector of the CIP prior factor is then given by ${}^{\mathrm{CIP}}\mathbf{r}_{\mathrm{ij}} = \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{e}_k$, where:

$$\mathbf{e}_k = \eta_k - \hat{\eta}_k \tag{9.10}$$

The covariance of the CIP prior factor $\mathbf{\Sigma}$ is a diagonal matrix defined as:

$$\mathbf{\Sigma} = \text{diag}\left(\frac{1}{n-1}\sum_{i=1}^{n-1}\langle\delta_i, \delta_i\rangle\right) \tag{9.11}$$

where n is the total number of the data points and δ_i is the fitting error for the i^{th} data point.

9.5 Experimental results

The camera and IMU of the iPhone 7 were first characterized. Based on the experimental data, a linear model that relates the CIP to the IMU-measured acceleration of the camera was derived. Both simulations and experiments were conducted to compare the pose estimation performance of CIP-VMobile with that of VINS-Mobile. The performance of CIP-VMobile was also evaluated in assisted wayfinding applications by using the RC (see Figure 3 left) as an experimental platform in real-world environments.

9.5.1 Calibration

As shown in Figure 27, the iPhone 7 was mounted on a Dynamixel EX-106 servo actuator that rotated the phone around its x, y, and z axes, respectively, from 0 to 210° with a step-size of 3°. At each step, the 3-axis accelerometer reading $\mathbf{a} = [a_x, a_y, a_z]^T$ and the camera's CIP values are obtained. The CIP values were determined by camera calibration [90]. The acquired data are plotted in Figure 28, which clearly indicates that the CIP values are linearly related to the acceleration.

During the experiments, the focal lengths along the x axis and y axis are found to be close to each other. Therefore, the difference in-between was ignored, letting $f_c = f_x = f_y$. The coefficients $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the linear CIP-Acceleration (CIPA) model

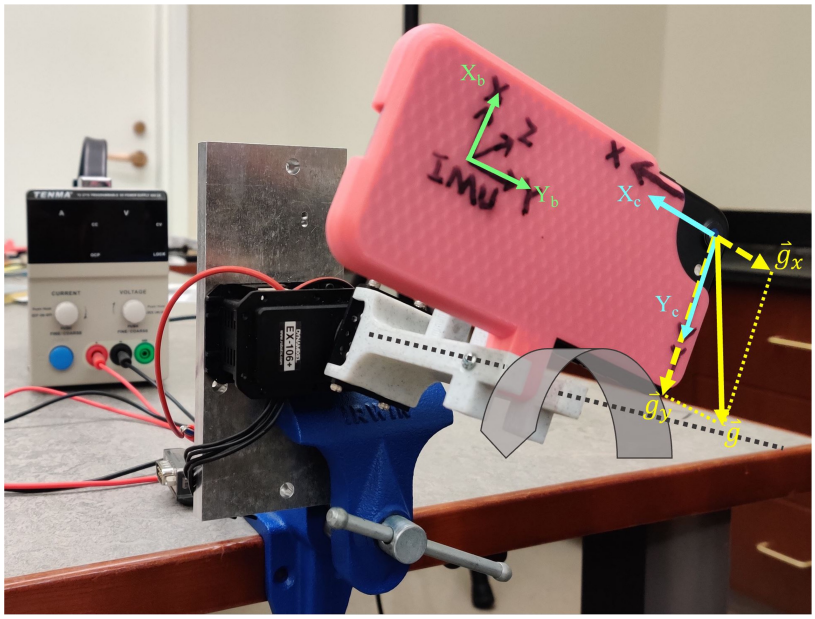


Fig. 27. Data collection setup for iPhone 7

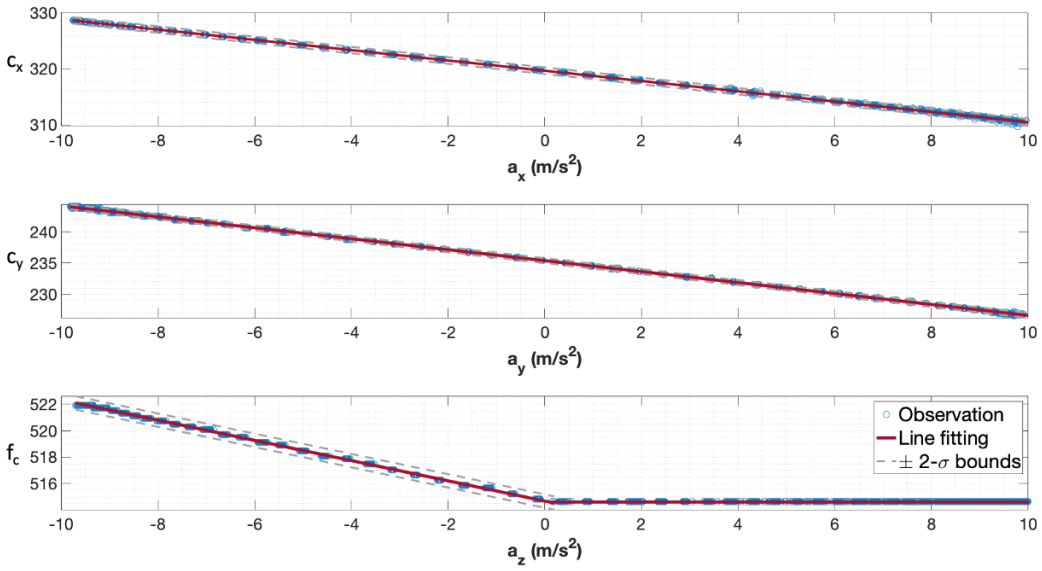


Fig. 28. Parametric fitting for the CIP's linear model. c_x, c_y, f_c unit: pixel.

denoted $L(\mathbf{a})$ can be obtained by line-fitting. The resulted model is given by

$$\begin{aligned} f_c &= \begin{cases} -0.6806 \cdot a_z + 514.7183, & a_z < 0 \\ 514.7183, & a_z \geq 0 \end{cases} \\ c_x &= -0.8549 \cdot a_x + 319.6476 \\ c_y &= -0.8817 \cdot a_y + 235.2553 \end{aligned} \tag{9.12}$$

The line-fitting result also produces the numerical covariance of the CIP prior factor (see Equation 9.11):

$$\mathbf{\Sigma} = diag\,(0.08585,\ 0.0392,\ 0.064) \tag{9.13}$$

9.5.2 Simulation

The open-source code [91] was employed to generate simulated visual-inertial data for a simulated run of moving the iPhone in an arbitrary trajectory (about 120 meters). The statistical properties of IMU and the values of the CIP are generated based on the calibration results. Visual features were projected by using a virtual camera with the corresponding CIP. The standard deviation of a visual feature measurement σ_γ is set to 1.5 pixels. We ran CIP-VMobile and VINS-Mobile on the simulated data. The estimated trajectories are compared against the ground truth in Figure 29. Clearly, CIP-VMobile demonstrates a superior pose estimation performance over VINS-Mobile: its trajectory closely tracks the ground truth while VINS-Mobile diverges quickly from the ground truth.

To demonstrate that the graph optimization process can effectively refine the CIPA-computed CIP, the CIPA model was purposefully degraded by increasing the noise 10 times (from 0.3 pixels to 3 pixels) and the accelerometer noise by 3 times. This significantly decreased the accuracy of the CIP data. CIP-VMobile with/without CIP optimization was then tested on the simulated data. Figure 30 compares the

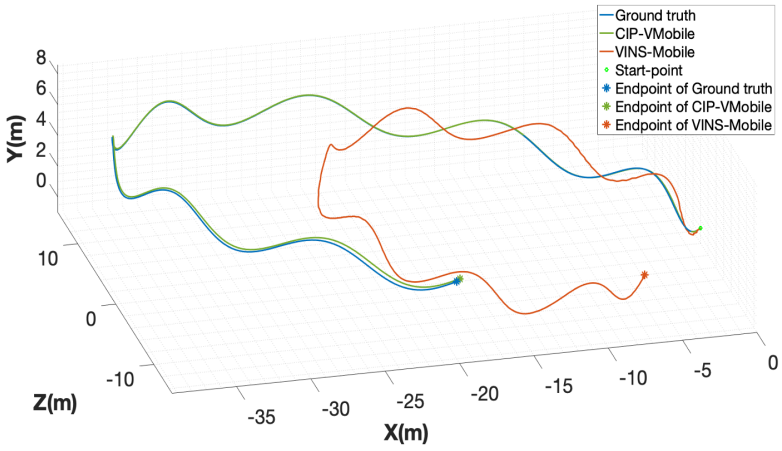


Fig. 29. Trajectory Comparison using simulated data

trajectories generated by the two simulation runs to the ground truth. The CIP-VMobile-generated trajectory is much closer to the ground truth. This was because the CIP optimization procedure of CIP-VMobile refined the CIP (as shown in Figure 31) for each keyframe during the graph optimization process, resulting in a more accurate camera model and thus a more accurate pose estimation result.

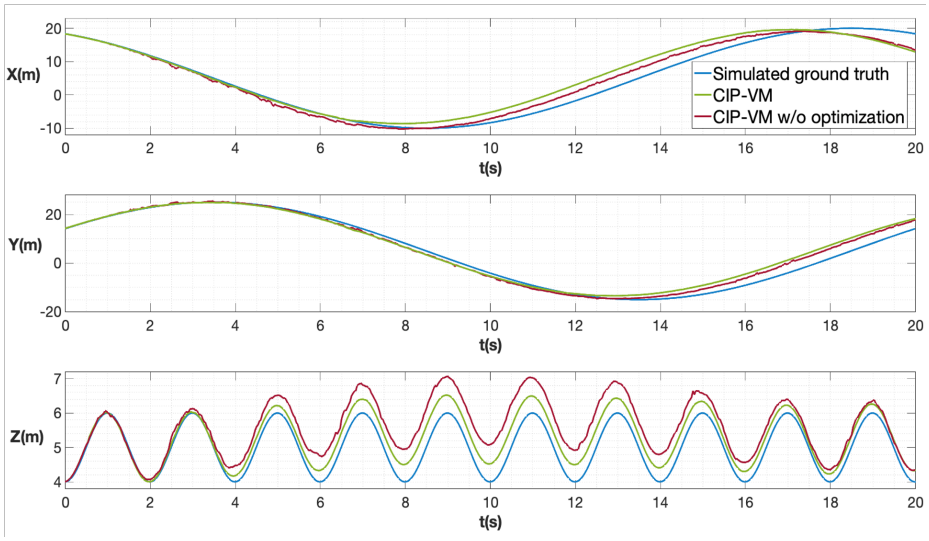


Fig. 30. Estimated Translation on x, y and z axis

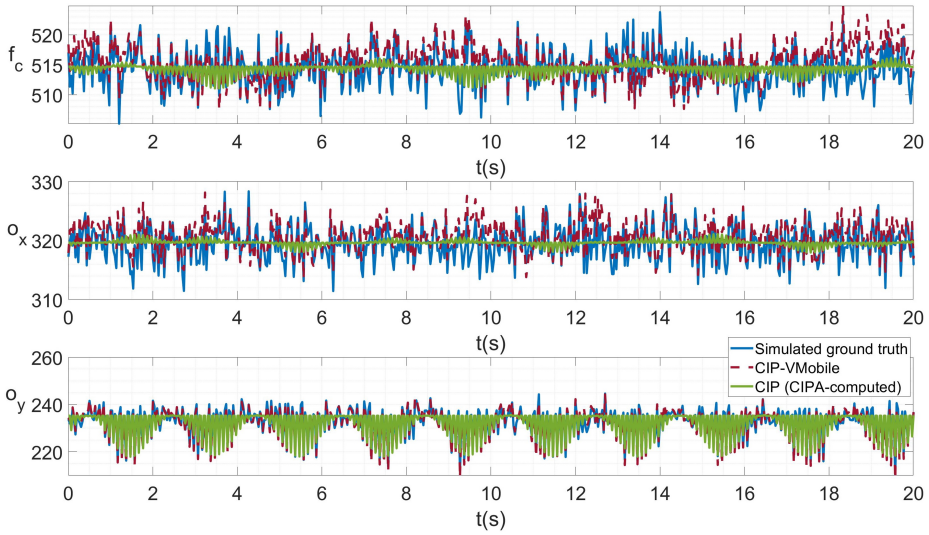


Fig. 31. CIP estimation using simulated data

In addition, the simulation results showed that CIP-VMobile resulted in more significant errors in tracking the intrinsic parameters when the CIPA model was not used to produce the initial CIP estimates. This was because without using the model, the method started with some bad CIP values, which increased the chance for the method to get stuck at the local minimum. The use of the CIPA model can effectively alleviate the effect of the local minimum and reduce CIP estimation error. It also reduces the iteration number of the graph optimization procedure and speeds up the computation.

9.5.3 Experimental Results with Hand-held iPhone

Ten experiments were conducted in the laboratory by an operator hand-holding the iPhone 7 when walking in a looped trajectory (with the same starting point and endpoint) at an average walking speed ($\sim$ 0.6 m/s). The length of the trajectory for each experiment is about 20 meters. At the beginning of each experiment, the iPhone was rotated significantly to excite the visual-inertial system to allow for a good

system initialization. The Endpoint Position Error Norm (EPEN) in a percentage of the path length is used as the metric of pose estimation accuracy. CIP-VMobile consistently outperformed VINS-Mobile. The results of the experiments are tabulated in Table 12. On average, CIP-VMobile reduces the EPEN error by 34.6%. Figure 32 compares the trajectories estimated by the two methods against that of the ground truth trajectory for experiment 1. The CIP-VMobile generated trajectory tracks the ground truth better than that of VINS-Mobile. The root mean squares of the point-to-point position errors of CIP-VMobile and VINS-Mobile are 0.109 meters and 0.148 meters, respectively, indicating that CIP-VMobile has an overall better pose estimation accuracy.

Table 12. EPENs for experiments with a handheld iPhone

Exp	1	2	3	4	5	6	7	8	9	10	Avg
C	0.89	1.34	0.89	1.11	1.30	0.95	0.53	0.73	1.95	0.89	1.06
V	1.05	3.41	1.37	1.26	1.90	1.29	1.05	0.84	2.42	1.34	1.62

C: CIP-VMobile, V: VINS-Mobile, Exp: Experiment, Avg: Average

9.5.4 Runtime Analysis

CIP-VMobile and VINS-Mobile were deployed on a laptop computer (Intel Core i7-8550U, 16 GB memory, Ubuntu 16.04 LTS 64-bit OS) for runtime analysis. The results showed that both methods could compute pose in real-time. Taking the 1^{st} experiment in Table 12 for instance, the runtimes of the two methods are compared in Figure 33. It can be seen that the runtime for CIP-VMobile to compute a pose is larger than that of VINS-Mobile. On average, the runtimes for CIP-VMobile and VINS-Mobile are 44.0 ms and 32.4 ms, respectively. With respect to the imple-

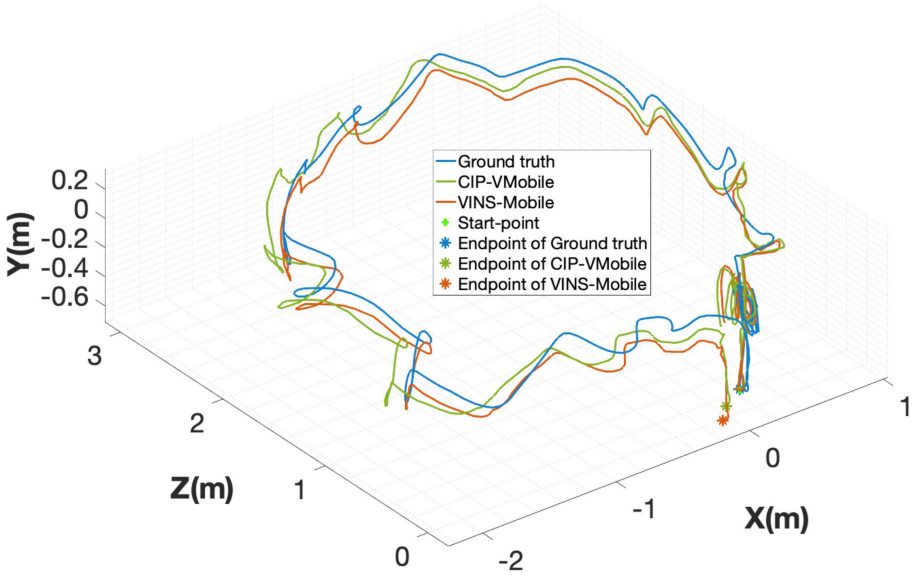


Fig. 32. Trajectory comparison using data by hand-holding an iPhone 7

mentation with a smartphone, VINS-Mobile achieved a real-time pose computation performance (∼23 per pose computation) on an iPhone 7. Therefore, it is anticipated that CIP-VMobile can run in real-time on the same smartphone platform. It is noted that VINS-Mobile uses simplified linear algebra libraries to save computational cost when implemented with an iPhone 7, resulting in a faster speed than the laptop implementation.

9.5.5 Experimental Results with the RC

To validate the CIP-VMobile method in the real world, experiments with the RC prototype were conducted in the hallways of the Engineering East Building at VCU. In each experiment, the RC user walked from Room 2264 to the elevator and returned to the starting point. The user swung the RC when walking to mimic how a blind person uses a traditional white cane. The results are tabulated in Table 13. It can be seen that CIP-VMobile has a smaller EPEN than VINS-Mobile in all four experiments.

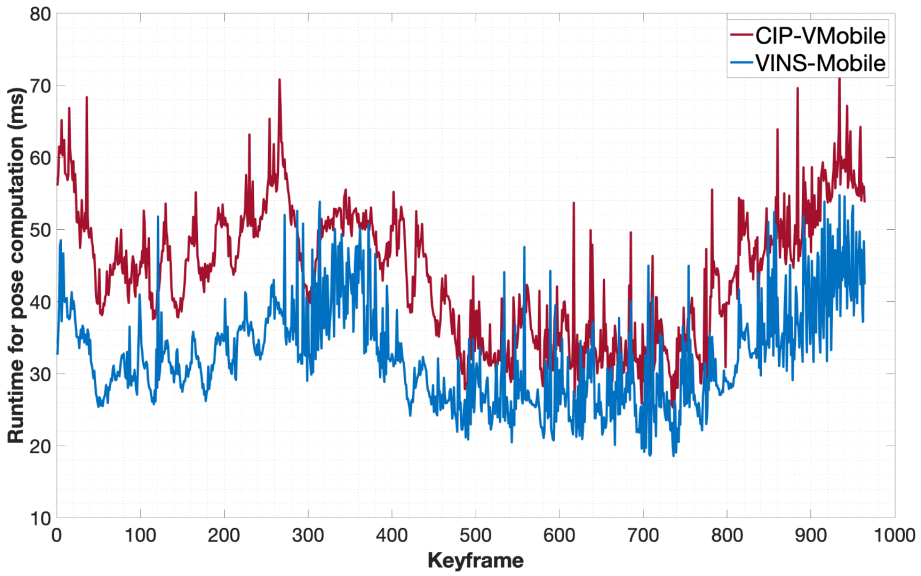


Fig. 33. Comparison of the runtimes for CIP-VMobile and VINS-Mobile.

On average, CIP-VMobile reduces the EPEN by $\sim 11\%$. The trajectories estimated by the two methods for experiment 3 are compared in Figure 34, from which it can be observed that CIP-VMobile resulted in a more accurate trajectory than VINS-Mobile. This is evidenced by that the trajectory of VINS-Mobile collides with the walls at several locations, but no collision occurs along the CIP-VMobile estimated trajectory.

Table 13. EPENs for experiments with the RC prototype

Experiment	1	2	3	4	Avg
CIP-VMobile	1.07%	1.57%	1.21%	1.59%	1.46%
VINS-Mobile	1.31%	1.69%	1.38%	1.85%	1.64%

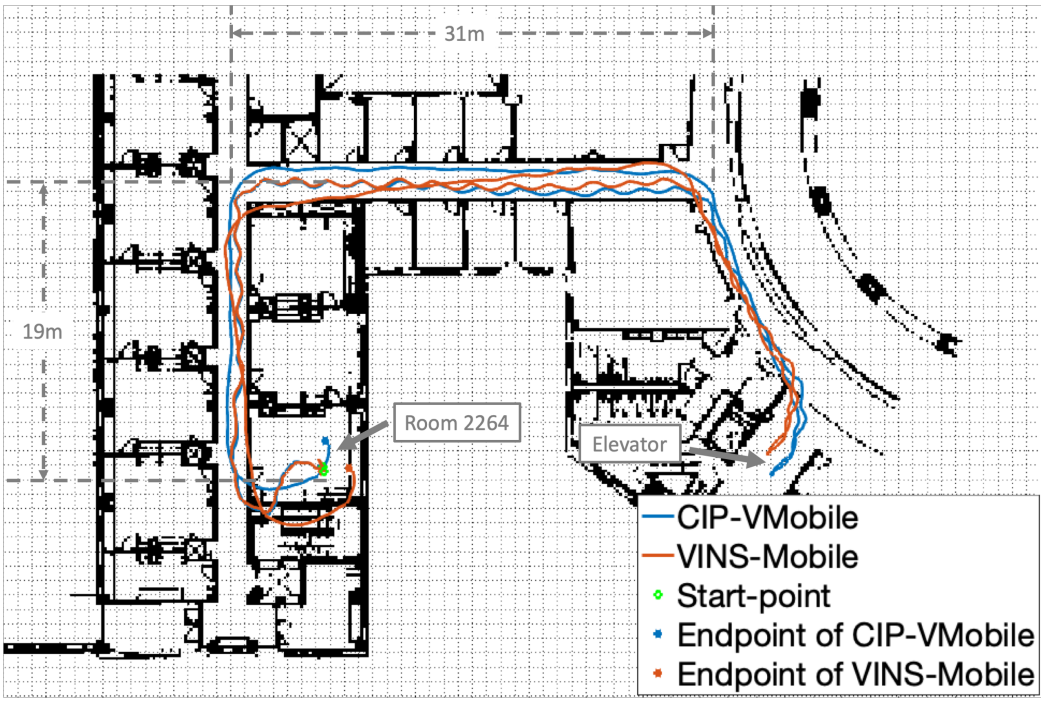


Fig. 34. Trajectory comparison using data from RC

9.6 Application in VLIO

The proposed online CIP estimation method was incorporated with VLIO and tested by using the same dataset in Chapter 8.4. The results are tabulated in Table 14. As can be seen, with online CIP estimation the mean and RMSE are reduced by 28% and 27%, respectively. CIP-VMobile and VINS-Mobile are also compared in this table. CIP-VMobile reduces the RMSE by 21%.

9.7 Summary

A new VIO method, called CIP-VMobile, was introduced for pose estimation of a modern smartphone. The proposed method treats the CIPs of the phone's camera as state variables. It tightly couples them with the other state variables (including the camera poses, velocity, and IMU bias) in a graph optimization process

Table 14. Means and RMSE of the estimated trajectories under different conditions

Dataset	Unit(m)	VLIO	VLIO w/ CIP	VINS-Mobile	CIP-VMobile
Data1	Mean	0.090	**0.079**	0.086	**0.068**
18m	RMSE	0.101	**0.093**	0.100	**0.079**
Data2	Mean	0.110	**0.084**	**0.078**	0.085
18m	RMSE	0.122	**0.104**	**0.098**	0.102
Data3	Mean	0.112	**0.093**	0.138	**0.110**
28m	RMSE	0.138	**0.121**	0.150	**0.135**
Data4	Mean	0.110	**0.065**	0.135	**0.134**
37cm	RMSE	0.118	**0.072**	0.148	**0.145**
Data5	Mean	0.230	**0.117**	0.178	**0.073**
39m	RMSE	0.296	**0.122**	0.239	**0.078**
Average	Mean	0.47%	**0.34%**	0.44%	0.36%
	RMSE	0.55%	**0.40%**	0.52%	0.41%

The best results are highlighted in blue, and the better result between the methods with and without CIP estimation is highlighted in bold.

to solve the state estimation problem. As part of the state variables, the CIP values are re-estimated at each iteration of the VIO process, resulting in a more accurate pose estimate. A linear model relating the IMU-measured acceleration to the CIP is created and used to initialize the CIP values for all keyframes. The use of the initial CIP values speeds up the VIO computation and improves the pose estimation accuracy. The method was validated to be effective with a robotic navigation aid in real-world assistive navigation cases. The proposed method is the first of its kind in the literature.

CHAPTER 10

SLAM IN DYNAMIC ENVIRONMENTS

This dissertation aims to develop new SLAM techniques that are suitable for implementation on an edge device and can be used to create smartphone applications that can help the visually impaired in their everyday lives. The depth-enhanced VIO method and the VIO method capable of online camera intrinsic parameter estimation can effectively reduce accumulative estimation error of device pose. However, they are based on an assumption that the scene is static. Such an assumption usually does not hold in real-world scenarios. For example, indoor environments often have moving objects (e.g., pedestrians), which will likely cause the VIO methods as well as existing SLAM methods to malfunction.

Existing SLAM methods [8], [32], [92] use a VO front-end to estimate the pose change between two image frames. The VO usually uses a PnP RANSAC [93] process to find a motion consensus for the tracked visual features and remove outliers. It may obtain an accurate pose estimate when static features in the view are dominant, and their number is adequate. In this case, the minority dynamic features are identified as outliers (and removed) and thus have little effect on the pose change estimation result. However, when the dynamic features (1) are the majority or (2) have a consensus in motion, existing VIO methods may incur a long RANSAC process to identify inliers or even fail. In the former case, the low inlier ratio may dramatically increase the number of iterations of the RANSAC process and thus result in a long computational time. And in the latter case, the motion consensus falls on the moving object, and the estimated pose change is therefore incorrect.

In recent years, researchers have attempted to use deep-learning methods to segment the scene and use the segmentation result to alleviate the negative impact of moving objects on the performance of a SLAM method. The most representative works include DynaSLAM [94] and DS-SLAM [95]. DynaSLAM [94] treats all movable objects (e.g., people, cars) as moving ones regardless of their actual motion states and only detects visual features from the regions containing static object(s) and the background. DynaSLAM achieved good results on TUM dynamic dataset [3], where two people are moving around in the scene. But its performance on some other datasets (e.g., KITTI dataset [4]) is even worse than that of the original ORB-SLAM2 [92]. This is because it eliminates visual features on any movable object even if it is static (e.g., a parked car). These features could otherwise be used for SLAM computation. DS-SLAM [95], on the other hand, determines the moving features on each segmented object by a RANSAC process based on the fundamental matrix (i.e., by checking if each feature satisfies the epipolar constraint). If the number of moving features is above a threshold, the object is treated as a moving one, and all visual features on this object are excluded from SLAM computation. In essence, this approach checks motion consistency between the parts of an object and uses the level of consistency to speculate if it is moving. It makes sense for certain objects. For instance, the moving lower-limbs of a person indicate he/she is in motion. Due to its incapability in detecting the object's movement relative to the static background, the result can be erroneous, causing over-removal (false positive) or under-removal (false negative) of moving features. Neither DynaSLAM [94] nor DS-SLAM [95] can effectively solve SLAM with dynamic objects because they cannot reliably identify static visual features in all cases.

To address the problem, I will wrap up this dissertation work by proposing a semantic-segmentation-aided VIO method, named SM-SLAM, for dynamic environ-

ments. It combines a deep learning object detection and segmentation method and a popular SLAM framework (ORB-SLAM2 [92] or VINS-Mono [8]). The proposed method first detects objects and semantically classifies their motion statuses into three categories - likely moving (LM), likely static (LS), and dynamic. Then, it attempts to identify any LS features out of the LM features and use them to increase the inlier of the LS features, by applying RANSAC to one object after another. The outliers generated by the RANSAC process are marked as dynamic features. SM-SLAM uses only static features to estimate the pose change between two image frames.

10.1 Deep-Learning based Semantic Segmentation

In the field of computer vision, semantic segmentation plays a critical role in understanding the content of images. Several representative approaches to semantic segmentation are illustrated in Figure 35. The terminologies, Semantic Segmentation, Instance Segmentation, Panoptic Segmentation, and object detection, are often used interchangeably in the literature. However, each of them has its unique characteristics and suitable application domains despite their common ground.

Semantic segmentation classifies each pixel in an image into predefined classes, without distinguishing different objects of the same class. Instance segmentation, on the other hand, not only classifies each pixel but also identifies distinct objects of the same class. Panoptic segmentation combines the characteristics of both semantic and instance segmentation, providing a comprehensive perspective by delineating every pixel while also identifying individual objects. Object detection, distinct from the above-mentioned, locates and identifies objects within an image without classifying every pixel. Due to the difference in the operating principles, these methods incur different computational costs. Table 15 shows the computational times of some common object segmentation/detection methods that I implemented with an iPhone 12

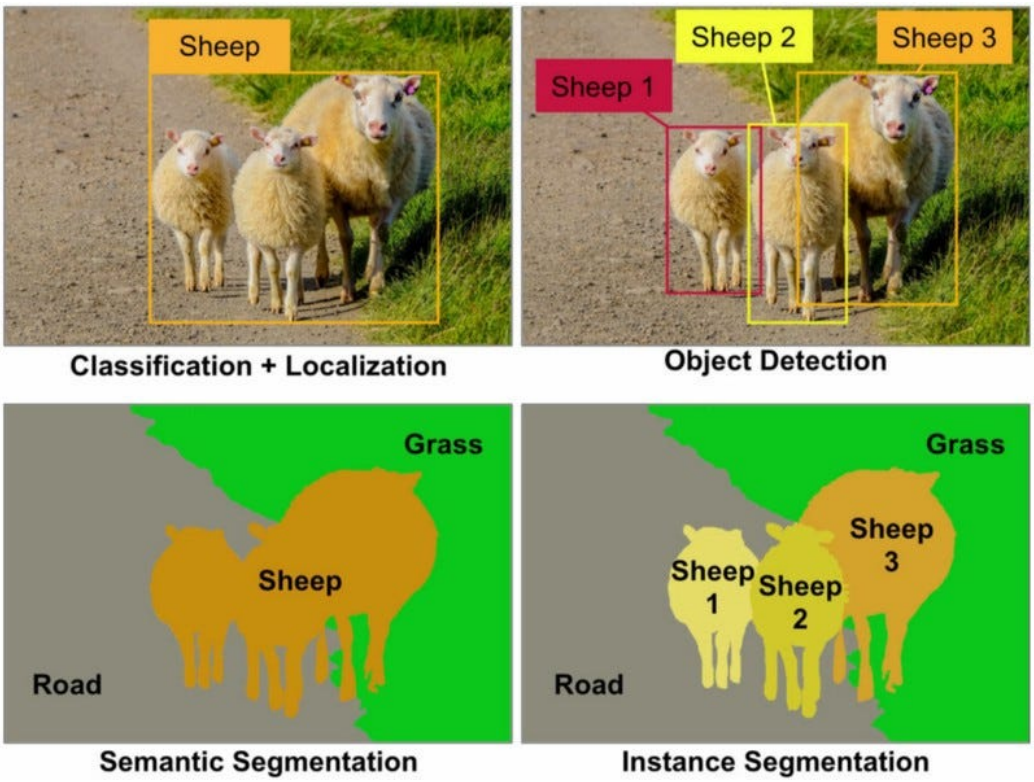


Fig. 35. Four main classes of problems in detection and segmentation

Pro smartphone. In these cases, the object detection models have been converted to the Apple Core ML Model except for Mask-RCNN, which uses an older segmentation model that cannot be specifically optimized for Apple devices.

Table 15. Computation time of object detection and segmentation methods

Name	Task	Speed
YOLOv8n-obj	Object Detection	7.46 ms
YOLOv3	Object Detection	29.94 ms
DeepLabV3	Semantic segmentation	29.07 ms
SegNet	Semantic segmentation	31.22 ms
Mask-RCNN	Instance Segmentation	610.6 ms
YOLOv8x-inst	Panoptic Segmentation	66.53 ms
Panoptic-FPN	Panoptic Segmentation	71.88 ms

YOLOv8n-obj is the smallest and fastest variant of YOLOv8, while YOLOv8x-inst is the full panoptic segmentation model of YOLOv8.

Recently, substantial advancements have been made in object segmentation and object detection. The representative works include SegNet [96] and DeepLabV3 [97] for semantic segmentation, Mask R-CNN [98] for instance segmentation, Panoptic-FPN [99]and UPSNet [100] for panoptic segmentation, and YOLO [101] for object detection. There have been attempts to integrate object segmentation/detection with existing SLAM frameworks as this may allow a SLAM method to understand the scene semantically and use the semantic object information to improve SLAM performance in the presence of dynamic objects. For example, DynaSLAM [94] uses Mask R-CNN [98] to provide instance labels with object semantic information. DS-SLAM [95] employs SegNet [96] to serve a similar purpose (without instance information). While these object segmentation methods provide highly accurate and comprehensive

segmentation results, they do this at a significant computational overhead. This poses challenges to their real-time implementation on a mobile device.

In contrast, object detection methods such as YOLO [101] can effectively detect many common indoor objects (e.g., people, furniture, etc.) with a substantially lower computational cost. Instead of segmenting objects in a scene, these methods identify objects in terms of class and instance and label each of them by a bounding box. Even though no object boundaries information is provided, the object detection result can be used to analyze each object's motion for dynamic SLAM.

Recently, YOLOv8 [102] is proposed as a state-of-the-art (SOTA) model for object detection. It builds upon the success of previous YOLO versions [101] and includes some new functions, such as instance segmentation. In this work, YOLOv8 is used to extract semantic information (including instance ID, object class, object region, and detection probability) for the proposed SM-SLAM methods. In the rest of this chapter, YOLOv8-obj stands for the YOLOv8 variant with object detection function, while YOLOv8-inst refers to the YOLOv8 variant with both object detection and object segmentation functions. As can be seen from Table 15 that YOLOv8-obj is very computationally efficient in object detection. Despite object regions (given by the bounding boxes) that it produces being less accurate than that of a segmentation method (e.g., YOLOv8-inst), it provides a more time-efficient solution for dynamic SLAM. It is noted that given the much lower number of visual features in each object region (compared to the entire image) the inaccurate object boundary will not increase the computational cost in object motion analysis too much. In this Chapter, the results of both SM-SLAM using YOLOv8-obj($mAP^{val} = 37.3\%$) and SM-SLAM using YOLOv8-inst($mAP^{val} = 53.9\%$) are reported for comparative analysis.

10.2 Extended Semantic Information of Object/Feature

A segmented object's semantic information is extended by adding its motion status to form a so-called Extended Semantic Information (ESI). Harris corners are extracted from the scene's image and clustered using the segmentation result. Each feature on the same object inherits the ESI of that object. When an object is first detected/segmented, its initial motion status is determined as Likely Moving (LM) or Likely Static (LS) based on prior knowledge about the object (e.g., a piece of furniture is LS while a person is LM). Its subsequent motion status will be determined based on the ratio of its LS features. The background is regarded as a special object with LS as its initial motion status.

10.3 Object Association and Semantic Information Update of Visual Features

Visual features are tracked across image frames by using optical flow. If IMU data is available, IMU-measured camera velocity is then used to facilitate feature tracking. Otherwise, a constant camera velocity is used. Feature tracking result is used to determine if an object in the previous frame is tracked onto the current frame. If the majority of features (60% in this work) are tracked, then an object association is established, i.e., the object on the previous frame is tracked onto the current frame. For a tracked object, its semantic information is determined/updated as follows:

1. If the associated object in the current frame is a front object that is detected as one of a different class, the object detection probability scores on both frames are then compared. The object with the higher score is the winner, and its semantic information will be used to replace that of the object with the lower score. The associated visual features' semantic information is then updated

accordingly.

2. If the associated object in the current frame is a background object (i.e., a previously detected object is not detected as a front object in the current frame), object association fails. In this case, the tracked features in the current frame will inherit the semantic information from those features in the previous frame.

10.4 Dynamic Feature Identification and Rejection

Assuming that the majority of features are static, the VO solution is to find a consensus of motion model (static in this case), with which the majority of features satisfy. This is performed by a RANSAC process based on the fundamental matrix: Eight pairs of features are randomly selected and used to compute the fundamental matrix F (by using the Eight-point method [47]) and the epipolar distance for each feature pair ($p_i^{\text{prev}}, p_i^{\text{cur}}$) on the current frame is computed as $d_i^{ep} = \frac{\left|p_i^{\text{cur}} F p_i^{\text{prev}}\right|}{\sqrt{a^2+b^2}}$, based on which ($p_i^{\text{prev}}, p_i^{\text{cur}}$) is identified as an inlier if d_i^{ep} is below a threshold, or outlier otherwise. The total number of inliers is recorded for this F. To find the maximum number of inliers, this process needs to be iterated k times:

$$k = \frac{\log(1-p)}{\log\left(1-w^n\right)} \tag{10.1}$$

where w is the inlier ratio, n = 8, and p = 95% in this work. When there are dynamic objects, the inlier ratio can be very low (i.e., w is very small), resulting in a very large k and prolonged computation time. To avoid this, k is estimated by using an estimated value of w, which is computed as the ratio of the total LS features to all visual features in the entire image. This is equivalent to assuming that all LS features are static. If k is smaller than 100 (i.e., $w \geq 65\%$), the above-mentioned RANSAC process will be applied to the entire image to identify the total static features, which

are then used to compute the pose change between the two frames. Otherwise, the following steps will be performed:

1. For each LS object, a RANSAC using only its LS features is applied to find the inliers. If the inlier ratio is below a threshold value, this object is determined as a dynamic one, and all of its features are excluded from further processing. Otherwise, the inliers are added into an LS feature set. This step repeats until all LS objects are processed.

2. Merge the resulted set of LS features with the features of an LM object one by one, starting from the one with the smallest number of LM features. Apply a RANSAC process to the new set of features to find the inliers.

   (a) If the number of inliers does not increase, the LM object is determined as a dynamic one. All features on the dynamic objects will be excluded for the next step.

   (b) Otherwise, check the LS features ratio of this LM object. If the ratio is above a threshold, all LS features of this object are then merged with the LS feature set. Otherwise, all features on this object will be excluded for the next step. If the LS feature ratio is above the threshold for three consecutive frames, this object's motion status is then changed to LS. Otherwise, it remains LM.

   Repeat step 2 until all LM objects are processed.

After the above processing, the resulted LS features are forwarded to the SLAM backend for pose estimation.

10.5 Experimental Results: SM-DVO on TUM Dataset

The TUM dataset is used to evaluate the performance of the proposal SM-SLAM approach. Since the TUM dataset was collected by using an RGB-D camera, a RGB-D-camera-based SM-SLAM variant, called SM-DVO, was developed by using the framework of ORB-SLAM2 [92]. Two object detection methods, namely YOLOv8-obj (YOLOv8 with object detection function) and YOLOv8-inst (YOLOv8 with both object detection and segmentation functions), were employed to investigate the impact of object segmentation (i.e., how much performance improvement can be made in dynamic SLAM by having accurate object boundary). The results of the SM-SLAM methods are compared with that of DynaSLAM [94] and DS-SLAM [95].

Table 16. RMSE of the Estimated Trajectories on TUM Dataset

	ORB-SLAM2	DynaSLAM	DS-SLAM	SM-DVO-inst	SM-DVO-obj
fr3_walking_static	0.390 m	**0.006 m**	0.008 m	0.011 m	0.007 m
fr3_walking_rpy	0.871 m	**0.035 m**	0.444 m	0.061 m	0.076 m
fr3_walking_xyz	0.752 m	0.015 m	0.025 m	**0.010 m**	0.011 m
fr3_walking_half	0.486 m	0.025 m	0.030 m	**0.017 m**	0.021 m
fr3_sitting_half	0.020 m	0.017 m	0.015 m	0.014 m	**0.013 m**
fr3_sitting_xyz	0.009 m	0.015 m	0.010 m	**0.007 m**	0.008 m

The best results are highlighted in bold.

As shown in 16, SM-DVO outperforms DS-SLAM[95] in most sequences due to its proficiency in incorporating additional static features from LM objects. Despite this, DynaSLAM[94] remains superior in terms of accuracy for $fr3\_walking\_static$ and $fr3\_walking\_rpy$, both representing imaging data sequences with high dynamics. This is because that DynaSLAM [94] uses an aggressive approach to removing dynamic features (removing any features fall within a movable object regardless of

its actual motion status), of which some are challenging for SM-DVO to identify by using epipolar constraint (e.g., if they are moving along the epipolar line). In other words, SM-DVO [95] may potentially have more unidentified dynamic features than DynaSLAM [94] if the detected objects are moving. resulting in a larger pose estimation error. However, if the detected objects are actually static, DynaSLAM's aggressive approach may cause over removal of the statics visual features on these objects, resulting in a subpar performance in pose estimation for environments with low dynamics. Taking dataset *fr3_sitting_xyz* for instance, DynaSLAM [94] overlooks the visual features of a seated person even though most of them are static.

10.6 Experimental Results: SM-DVIO on Datasets

Table 17. RMSE of the Estimated Trajectories on VCU-RVI Dataset

Data Sequence	Mono	RGBD	DVIO	SM-DVIO-obj	SM-DVIO-inst
lab-dynamic1	0.39 m	0.22 m	0.21 m	0.14 m	**0.12 m**
lab-dynamic2	0.62 m	X	0.54 m	0.19 m	**0.18 m**
lab-dynamic3	X	0.56 m	0.47 m	0.35 m	**0.35 m**
lab-dynamic4	X	X	X	0.58m	**0.55 m**
lab-dynamic5	0.75 m	X	0.31 m	0.15 m	**0.14 m**

The best results are highlighted in bold.

X means that the method fails to generate a trajectory.

To test the proposed SM-SLAM in scenarios where inertial (IMU) data are available, SM-DVO is extended by using the same backend methodology in Chapter 7. The extended variant is called SM-DVIO. A dataset called the VCU-RVI dataset, collected at VCU by our team is used to evaluate the SM-SLAM approach with the availability of IMU data. The VCU-RVI dataset, comprising dynamic sequences, was

acquired by using a hand-held Structure Core sensor in a lab environment where the WROMA was mostly used. The dynamic elements include people, chairs, a wheeled robot, and a rollator. SM-DVIO was run on the datasets, and the results are compared with that of other methods including VINS-Mono, VINS-RGBD, and DVIO, and tabulated in Tables 17. As can be seen, SM-DVIO outperforms the other methods and is the only method that produces a correct trajectory on sequence 4.

CHAPTER 11

CONCLUSION AND FUTURE WORK

This dissertation work aims to improve SLAM techniques for assistive navigation. HPnP was first proposed as a pose change estimation method for RGB-D cameras. It decouples the estimation of rotation and translation and improves translation estimation accuracy by using visual features with a depth measurement from both frames. With HPnP in the visual front-end, DVIO was introduced as a depth-enhanced visual-inertial odometry to overcome the drawbacks of existing VIO methods in initialization and scale estimation. Experiments demonstrate that DVIO produces a smaller accumulative error and can be used for accurate pose estimation for two robotics assistive devices, RoboCane and W-ROMA. To migrate DVIO from a dedicated onboard computer to a mobile device, CIP-VMobile and VLIO were developed. CIP-VMobile incorporates online CIP estimation into its VIO process while VLIO focuses itself on the usage of iPhone's Lidar data. Finally, SM-SLAM is attempted as a possible solution for SLAM in dynamic environments. It extends existing SLAM methods, ORB-SLAM2 and DVIO, by incorporating dynamic object detection capabilities in the front-end. Initial experiments with the public TUM Dataset and our VCU-RVI dataset have confirmed its viability. The findings reveal that SM-SLAM outperforms several state-of-the-art techniques in most sequence comparisons. Nonetheless, I acknowledge that there is room for refining SM-SLAM. Future research will therefore aim at further optimizing SM-SLAM performance and exploring potential areas of improvement, with the ultimate goal of advancing the robustness and efficiency of SLAM technologies.

Appendix A

ABBREVIATIONS

AR	Augmented Reality
ART	Active Rolling Tip
BLE	Bluetooth Low Energy
BVI	Visually Impaired
CIP	Camera Intrinsic Parameters
DDT	Desired Direction of Travel
DoF	Degree of Freedom
EKF	Extended Kalman Filter
EPEN	Endpoint Position Error Norm
GPS	Global Positioning System
HPnP	Hybrid Perspective-n-Point
IB	Initial Background
IF	Initial Feature
II	Initial Inlier
IMU	Inertial Measurement unit
LM	Likely Moving
LS	Likely Static
MCS	Motion Capture System
NST	Number of Successful Trials
OIS	Optical Image Stabilization
PCE	Pose Change Estimation

PAD	Portable Assistive Device
PND	Portable Navigation Device
PnP	Perspective-n-Point
PVQ	Position, Velocity, and Quaternion
RAD	Robotic Assistive Devices
RANSAC	Random Sample Consensus
RCC	RNA Control circuit
RNA	Robotic Navigation Aid
RVA	Richmond Virginia
SC	Structure Core
SLAM	Simultaneous Localization and Mapping
ToF	Time-of-Flight
VCM	Voice Coil Motor
VCU	Virginia Commonwealth University
VINS	Visual-Inertial Navigation System
VIO	Visual Inertial Odometry
VLIO	Visual-LiDAR-Inertial Odometry
VO	Visual Odometry
VR	Virtual Reality
W-ROMA	Wearable Robotic Object Manipulation Aid

REFERENCES

[1] Rupert RA Bourne et al. "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis". In: *The Lancet Global Health* 5.9 (2017), e888–e897.

[2] Sylvie Treuillet et al. "Body Mounted Vision System for Visually Impaired Outdoor and Indoor Wayfinding Assistance." In: *CVHI*. 2007.

[3] Jürgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 573–580.

[4] Andreas Geiger et al. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[5] Michael Burri et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163.

[6] Stefan Leutenegger et al. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[7] Raúl Mur-Artal and Juan D Tardós. "Visual-inertial monocular SLAM with map reuse". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803.

[8] Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[9] He Zhang, Lingqiu Jin, and Cang Ye. "The VCU-RVI Benchmark: Evaluating Visual Inertial Odometry for Indoor Navigation Applications with an RGB-D Camera". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 6209–6214.

[10] Ming Yu Fan, Jia Tong Bao, and Hong Ru Tang. "A guide cane system for assisting the blind in travelling in outdoor environments". In: *Applied Mechanics and Materials*. Vol. 631. Trans Tech Publ. 2014, pp. 568–571.

[11] Dileeka Dias et al. "A sensor platform for the visually impaired to walk straight avoiding obstacles". In: *2015 9th International Conference on Sensing Technology (ICST)*. IEEE. 2015, pp. 838–843.

[12] Ramesh Saptpute et al. "Smart cane for Visually impaired person by using Arduino". In: *IJIR* 3.5 (2017), pp. 1104–1108.

[13] Dada Emmanuel Gbenga, Arhyel Ibrahim Shani, and Adebimpe Lateef Adekunle. "Smart walking stick for visually impaired people using ultrasonic sensors and Arduino". In: *International journal of engineering and technology* 9.5 (2017), pp. 3435–3447.

[14] Sharang Sharma et al. "Multiple distance sensors based smart stick for visually impaired people". In: *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2017, pp. 1–5.

[15] Krishna Kumar et al. "Development of an ultrasonic cane as a navigation aid for the blind people". In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE. 2014, pp. 475–479.

[16] Yuichi Tsuboi et al. "Detection of Obstacles and Steps by a White Cane Device for Visually Impaired People". In: *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2019, pp. 1491–1496.

[17] Nur Syazreen Ahmad, Ng Lai Boon, and Patrick Goh. "Multi-sensor obstacle detection system via model-based state-feedback control in smart cane design for the visually challenged". In: *IEEE Access* 6 (2018), pp. 64182–64192.

[18] Quoc Khanh Dang et al. "A virtual blind cane using a line laser-based vision system and an inertial measurement unit". In: *Sensors* 16.1 (2016), p. 95.

[19] J Malvern Benjamin Jr and Nazir A Ali. "An improved laser cane for the blind". In: *Quantitative Imagery in the Biomedical Sciences II*. Vol. 40. SPIE. 1974, pp. 101–104.

[20] Dragan Ahmetovic et al. "NavCog: a navigational cognitive assistant for the blind". In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2016, pp. 90–99.

[21] Aura Ganz et al. "PERCEPT indoor navigation system for the blind and visually impaired: architecture and experimentation". In: *International journal of telemedicine and applications* 2012 (2012).

[22] Aura Ganz et al. "PERCEPT-II: Smartphone based indoor navigation system for the blind". In: *2014 36th annual international conference of the IEEE engineering in medicine and biology society*. IEEE. 2014, pp. 3662–3665.

[23] J Manuel Saez, Francisco Escolano, and Antonio Penalver. "First steps towards stereo-based 6DOF SLAM for the visually impaired". In: *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops. IEEE. 2005, pp. 23–23.

[24] Vivek Pradeep, Gerard Medioni, and James Weiland. "Robot vision for the visually impaired". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE. 2010, pp. 15–22.

[25] Young Hoon Lee and Gérard Medioni. "RGB-D camera based navigation for the visually impaired". In: *Proceedings of the RSS*. Vol. 2. Citeseer. 2011.

[26] Cang Ye et al. "Co-robotic cane: A new robotic navigation aid for the visually impaired". In: *IEEE Systems, Man, and Cybernetics Magazine* 2.2 (2016), pp. 33–42.

[27] He Zhang and Cang Ye. "An indoor navigation aid for the visually impaired". In: *2016 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE. 2016, pp. 467–472.

[28] Bing Li et al. "Vision-based mobile indoor assistive navigation aid for blind people". In: *IEEE transactions on mobile computing* 18.3 (2018), pp. 702–714.

[29] He Zhang, Lingqiu Jin, and Cang Ye. "A depth-enhanced visual inertial odometry for a robotic navigation aid for blind people". In: *Proc. Visual-Inertial Navigation: Challenges and Applications Workshop at 2019 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*. 2019.

[30] Pauline C Ng and Steven Henikoff. "SIFT: Predicting amino acid changes that affect protein function". In: *Nucleic acids research* 31.13 (2003), pp. 3812–3814.

[31] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[32] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[33] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[34] Zhixiang Min, Yiding Yang, and Enrique Dunn. "Voldor: Visual odometry from log-logistic dense optical flow residuals". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4898–4909.

[35] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM". In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[36] Stephan Weiss et al. "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments". In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, pp. 957–964.

[37] Vadim Indelman et al. "Information fusion in navigation systems via factor graph based incremental smoothing". In: *Robotics and Autonomous Systems* 61.8 (2013), pp. 721–738.

[38] Anastasios I Mourikis, Stergios I Roumeliotis, et al. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." In: *ICRA*. Vol. 2. 2007, p. 6.

[39] Stefan Leutenegger et al. "Keyframe-based visual-inertial slam using nonlinear optimization". In: *Proceedings of Robotis Science and Systems (RSS) 2013* (2013).

[40] Jeffrey Delmerico and Davide Scaramuzza. "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 2502–2509.

[41] He Zhang and Cang Ye. "Human-robot interaction for assisted wayfinding of a robotic navigation aid for the blind". In: *2019 12th International Conference on Human System Interaction (HSI)*. IEEE. 2019, pp. 137–142.

[42] *Up Board Power Up Your Ideas*. URL: `https://up-board.org`.

[43] *Bluno Nano SDK*. URL: `https://wiki.dfrobot.com/Bluno_Nano_SKU_DFR0296`.

[44] *Structure core - specs data.* URL: `https://structure.io/structure-core/specs`.

[45] Gack Davidson. *Google Pixel: The Complete Beginner's Guide*. Vol. 1. Van Helostein, 2017.

[46] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[47] Richard I Hartley. "In defense of the eight-point algorithm". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.6 (1997), pp. 580–593.

[48] Virginia Klema and Alan Laub. "The singular value decomposition: Its computation and some applications". In: *IEEE Transactions on automatic control* 25.2 (1980), pp. 164–176.

[49] Yousset I Abdel-Aziz, Hauck Michael Karara, and Michael Hauck. "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry". In: *Photogrammetric engineering & remote sensing* 81.2 (2015), pp. 103–107.

[50] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets". In: *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), pp. 698–700.

[51] Jean Gallier. *Geometric methods and applications: for computer science and engineering*. Vol. 38. Springer Science & Business Media, 2011.

[52] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem". In: *International journal of computer vision* 81.2 (2009), pp. 155–166.

[53] Laurent Kneip and Paul Furgale. "OpenGV: A unified and generalized approach to real-time calibrated geometric vision". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 1–8.

[54] Christian Forster et al. "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation". In: Georgia Institute of Technology. 2015.

[55] Preston C Hammer. "The midpoint method of numerical integration". In: *Mathematics Magazine* 31.4 (1958), pp. 193–195.

[56] Carlos Campos, Montiel Jose M.M., and Juan D. Tardos. "Fast and Robust Initialization for Visual-Inertial SLAM". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.

[57] Yonggen Ling and Shaojie Shen. "Dense visual-inertial odometry for tracking of aggressive motions". In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2015, pp. 576–583.

[58] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: Vancouver. 1981.

[59] Dorian Gálvez-López and Juan D Tardos. "Bags of binary words for fast place recognition in image sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[60] Todd Lupton and Salah Sukkarieh. "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions". In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 61–76.

[61] David Schubert et al. "The TUM VI benchmark for evaluating visual-inertial odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1680–1687.

[62] Laurent Kneip, Margarita Chli, and Roland Siegwart. "Robust real-time visual odometry with a single camera and an IMU". In: *Proceedings of the British Machine Vision Conference 2011*. British Machine Vision Association. 2011.

[63] Shaojie Shen, Nathan Michael, and Vijay Kumar. "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs". In: *2015*

IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2015, pp. 5303–5310.

[64] Ke Sun et al. "Robust stereo visual-inertial odometry for fast autonomous flight". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 965–972.

[65] Nicholas Brunetto et al. "Fusion of inertial and visual measurements for rgb-d slam on mobile devices". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 1–9.

[66] Tristan Laidlow et al. "Dense RGB-D-inertial SLAM with map deformations". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6741–6748.

[67] Yang Ling et al. "RGB-D inertial odometry for indoor robot via Keyframe-based nonlinear optimization". In: *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE. 2018, pp. 973–979.

[68] Zeyong Shan, Ruijian Li, and Sören Schwertfeger. "Rgbd-inertial trajectory estimation and mapping for ground robots". In: *Sensors* 19.10 (2019), p. 2251.

[69] Jianbo Shi et al. "Good features to track". In: *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE. 1994, pp. 593–600.

[70] Xuelong Li et al. "Robust subspace clustering by cauchy loss function". In: *IEEE transactions on neural networks and learning systems* 30.7 (2018), pp. 2067–2078.

[71] Anders Grunnet-Jepsen et al. "Projectors for intel® realsense depth cameras d4xx". In: *Intel Support, Interl Corporation: Santa Clara, CA, USA* (2018).

[72] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. "Sliding window filter with application to planetary landing". In: *Journal of Field Robotics* 27.5 (2010), pp. 587–608.

[73] Manii Xu. *VINS-Fusion RGBD*. 2022. URL: https://github.com/ManiiXu/VINS-Fusion-RGBD.

[74] *OptiTrack - Motion Capture Systems*. URL: https://optitrack.com.

[75] C Ye and J Borenstein. "T-transformation: a new traversability analysis method for terrain navigation". In: *Proc. of the SPIE Defense and Security Symposium, Orlando, USA*. 2004.

[76] Sepideh Seifzadeh, Bahador Khaleghi, and Fakhri Karray. "Distributed soft-data-constrained multi-model particle filter". In: *IEEE Transactions on Cybernetics* 45.3 (2014), pp. 384–394.

[77] Wera Winterhalter et al. "Accurate indoor localization for RGB-D smartphones and tablets given 2D floor plans". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 3138–3143.

[78] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters". In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46.

[79] He Zhang and Cang Ye. "An indoor wayfinding system based on geometric features aided graph SLAM for the visually impaired". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.9 (2017), pp. 1592–1604.

[80] *ARKit 4*. URL: https://developer.apple.com/augmented-reality/arkit/.

[81] *iPhone 12 Teardown X-ray Images*. URL: https://creativeelectron.com/apple-iphone-12-teardown-x-ray-images/.

[82] Joern Rehder et al. "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4304–4311.

[83] Cang Ye and Johann Borenstein. "Characterization of a 2D laser scanner for mobile robot obstacle negotiation". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 3. IEEE. 2002, pp. 2512–2518.

[84] Yoichi Okubo, Cang Ye, and Johann Borenstein. "Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation". In: *Unmanned Systems Technology XI*. Vol. 7332. SPIE. 2009, pp. 279–288.

[85] Cang Ye. "Mixed pixels removal of a laser rangefinder for mobile robot 3-d terrain mapping". In: *2008 International Conference on Information and Automation*. IEEE. 2008, pp. 1153–1158.

[86] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of solid-state circuits* 23.2 (1988), pp. 358–367.

[87] Bill Triggs et al. "Bundle adjustmenta modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[88] Richard J Topliss. *VCM OIS actuator module*. US Patent 9,134,503. Sept. 2015.

[89] Richard J Topliss et al. *Voice coil motor optical image stabilization wires*. US Patent 10,063,752. Aug. 2018.

[90] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[91] Yijia He. *vio data simulation*. 2022. URL: `https://github.com/HeYijia/vio_data_simulation`.

[92] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[93] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[94] Berta Bescos et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083.

[95] Chao Yu et al. "DS-SLAM: A semantic visual SLAM towards dynamic environments". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1168–1174.

[96] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[97] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE*

transactions on pattern analysis and machine intelligence 40.4 (2017), pp. 834–848.

[98] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[99] Alexander Kirillov et al. "Panoptic feature pyramid networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6399–6408.

[100] Yuwen Xiong et al. "Upsnet: A unified panoptic segmentation network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8818–8826.

[101] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[102] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: `https://github.com/ultralytics/ultralytics`.

VITA

Lingqiu Jin was born on September 6, 1992, in Wuxi, China. He received his Bachelor of Science in Electrical Engineering from The Pennsylvania State University in 2015. He received a Master of Science in the Department of Electrical Engineering from Columbia University in 2016. He started the doctoral program at the University of Arkansas and joined the Department of Computer Science at Virginia Commonwealth University in 2017.