Enhanced Indoor Visual Navigation Using Sensor Fusion and Semantic Information

by

Ahmed G. Mahmoud

A dissertation submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

Accurate and robust indoor navigation systems are crucial in fields like robotics and autonomous vehicles. In the absence of an absolute positioning system like GPS, there is no single sensor that can provide an accurate and robust indoor navigation solution. The presented thesis tackles the indoor navigation challenge using two approaches; multi-sensor fusion and semantic information. In the first approach, visual odometry is enhanced by the fusion of inertial sensors and wireless ranging measurements. The fusion filter is based on Extended Kalman Filter (EKF). Stereo vision can provide 3D positioning by triangulating visual features. However, depth estimation errors and expensive computation are key challenges. The developed multi-sensor system has dual-mode where stereo vision is applied first to estimate inertial sensor biases. Once converged, the estimated biases help the system to switch to a monocular mode which reduces the system complexity and enables the tracking of faster movements with higher frame rates. As both visual and inertial tracking are drifting solutions, wireless ranging/positioning is integrated into the system to provide absolute global positioning and ensure overall accuracy. In the second approach, an improved Visual Simultaneous Localization and Mapping (VSLAM) solution using semantic segmentation and layout estimation is developed. The system utilizes advanced semantic segmentation and indoor layout estimation to optimize map representation and increase positioning accuracy. A testbed has been developed to collect indoor multi-sensor data and to perform experiments and analysis. Out of this thesis work, three conference papers were published, one journal paper was published, in addition to one journal paper and one conference to be submitted.

Acknowledgements

As the end of my Ph.D. journey nears, I'd like to thank my supervisor, Professor Mohamed Atia, whose vast knowledge, professional guidance, and patience helped me build and finish my research smoothly. Professor Atia was highly supportive and provided sincere, insightful suggestions that profoundly impacted my professional and personal life. I would also like to thank my colleagues at the Embedded Multi-Sensor Systems Lab at Carleton University for their valuable inspiration and continued assistance.

The completion of my Ph.D. was a dream that my late father dreamt for me. His dream always had been my compass and gave me the strength I needed to overcome all the challenges I faced. And, of course, I can not forget the uncountable sacrifices and the unconditional love of my mother, who is and always has been my safe space. Finally, I will always be in debt to my wife and children for giving me purpose and for their love and support.

Table of Contents

List of Tables

List of Illustrations

List of Acronyms

Acronym	Definition
AHRS	Attitude Heading-Reference System
AoA	Angle of Arrival
BA	Bundle Adjustment
BOW	Bag Of Words
CNN	Convolutional Neural Network
DS-TWR	Double-Sided Two-Way-Range
DSO	Direct Sparse Odometry
EKF	Extended Kalman Filter
ESM	Efficient Second-Order Minimization
FAST	Features from accelerated segment test
FCN	Fully Convolutional Network
GM	Gauss-Markov
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
INS	Inertial Navigation System

JRD	Judgments of Relative Direction
KF	Kalman Filter
LDSO	Direct Sparse Odometry with Loop Closure
LSD-SLAM	Large Scale Direct SLAM
MAV	Micro Arial vehicle
MTL	Medial Temporal Lobe
NLOS	Non Line Of Sight
NED	North East Down
ORB	Oriented FAST and Rotated BRIEF
PDR	Pedestrian Dead-Reckoning
RANSAC	Random Sample Consensus
RCS	Remote Control Software
RMSE	Root Mean Square Error
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
SLERP	Spherical Linear Interpolation
SIFT	Scale Invariant Feature Transform
SOP	Scene and Orientation-dependent Pointing test
SURF	Speeded Up Robust Features

TDoA	Time Difference of Arrival
ToA	Time of Arrival
ToF	Time of Flight
UWB	Ultra-Wideband Technology
V-SLAM	Visual Simultaneous Localization and Mapping
VIN	Visual Inertial Navigation
VIO	Visual Inertial Odometry
VO	Visual Odometry

Chapter 1: Introduction

This chapter will provide brief background information about indoor and visual navigation systems, followed by a discussion of the common challenges in the field. Then, the motivation and objectives of the thesis work are presented. Subsequently, the contributions of the thesis work are discussed, and a list of publications is provided. Lastly, the chapter is concluded with the structure and organization of the thesis chapters.

1.1 Background

Indoor positioning and navigation are extremely important given the absence of Global Navigation Satellite Systems (GNSS) such as GPS indoors. Indoor navigation can be addressed using different sensor technologies such as vision, inertial, and range sensors. Vision sensors have attracted researchers' attention due to their lightweight and cost-effective characteristics [1]. In addition, they provide rich information about the environment. Therefore, visual navigation and localization have become fundamental capabilities of autonomous mobile agents [2]. In an ideal scenario, a map of the environment where the autonomous agent is operating is available. The moving agent uses its camera feed to accurately track its motion and observe the visual points and landmarks to localize itself. However, the localization task is expected to be performed without a map in an unknown environment in many applications. In the absence of a map, the problem is redefined to either a Visual Odometry (VO) problem that requires only tracking the agent's location or a Visual Simultaneous Localization and Mapping (VSLAM) [3] problem where joint estimation of the trajectory and the environment model is calculated.

Visual navigation research has been undertaken since almost 1980. Although the first two decades witnessed many offline implementations, real-time working systems

flourished only in the third decade, which has led visual navigation systems to be used on another planet by two Mars-exploration rovers for the first time [4]. Visual navigation can be performed using a single camera or multiple cameras. Stereo cameras allow a straightforward motion scale estimation through feature point triangulation; however, processing two camera streams require higher processing power limiting the overall frame rate. On the other hand, monocular cameras are incapable of observing the true scale of the world and can only estimate the map and camera trajectory up to a scale. In addition, the scale can drift, resulting in dissimilar scales for distant portions of the map. Additional data sources such as IMU or known distances on the map are required to scale the estimated solution properly.

Visual navigation tasks are performed accurately in a real-time fashion by the human brain. Whenever we are in motion, we use our visuals alongside the motor and inertial sensors to either localize ourselves or build spatial memories while navigating to exploit the future [5]. Also, our brain performs semantical and geometrical analysis of the environment to optimize the map representation and compensate for the errors in its position estimation.

Many visual localization (direct and feature-based odometry) [6] and SLAM (filtering and optimization) solutions have been presented in the literature. All available building on the mathematical SLAM problem formulation was laid in the late ’80s when the available technologies limited behavioural neuroscience and spatial memories studies. With the recent revolutionary finding in machine learning and virtual reality alongside neuroscientific studies, the SLAM problem needs to be revisited with the human brain navigational solution in mind.

1.2 Motivation

While the visual navigation problem could be considered solved from the theoretical and mathematical point of view, it is still an active field of research in its practical aspects. Robustness and accuracy solutions are still open challenges. The term accuracy describes how close the estimated trajectory is to the ground truth, while the system's robustness shows if the system is capable of tracking its position in all timesteps and the system's ability to recover from tracking loss. Tracking scattered feature points without relating them geometrically or semantically decreases the chances of track loss. Additionally, visual SLAM systems are affected by the repetitive indoor structures, which could severely confuse the system into reporting a wrong loop. Moreover, in a real-world scenario, sensory measurements are contaminated by different sources of error and noise, which, if not compensated and complemented, significantly degrade the accuracy of localization. Hence, the general problem that this Ph.D. research targeted is to utilize sensor fusion and machine learning approaches to build robust and accurate indoor visual navigation system systems.

1.3 Objectives

The main objective of this Ph.D. study is to apply sensor fusion and machine learning techniques to achieve the following goals:

- Enhancing the performance of visual odometry during partial occlusions.
- Reducing the complexity of stereo platforms using an inertial-aided single camera.
- Using inertial sensors to increase the processing rate of vision navigation systems.
- Minimizing global drifts due to scale inaccuracies and visual features tracking errors.

- Enhancing robustness and handling loop closures inaccuracies of VSLAM systems.

1.4 Contributions

To achieve the thesis goals, the following contributions have been achieved:

- The thesis developed and tested a hybrid vision-inertial fusion scheme that applies Kalman filtering to enhance the performance of indoor visual navigation systems against partial occlusions. The proposed system maintains the stereo vision system-level accuracy using a single camera aided by inertial sensors while achieving higher frame rates.
- The developed fusion filter is further enhanced by integrating measurements from UWB positioning observations. The filter design considers the platform motion physical limits as a non-holonomic constraint, further enhancing overall accuracy and robustness.
- Inspired by neuroscience observations of how humans naturally navigate indoors, an improved visual SLAM system was developed using semantic segmentation and indoor layout estimation technologies to optimize the map representation and increase the positioning accuracy by imitating the human brain navigational and spatial representation approaches.
- An embedded platform was developed to perform real-time multi-sensor synchronized logging and visualization of multiple navigational sensors' data. The platform is a testbed that can support future research in the indoor navigation field.

1.5 List of Publications

In the pursuit of the thesis research, the following papers have been published/submitted:

[1] **A. Mahmoud** and M. M. Atia, "Hybrid IMU-Aided Approach for Optimized Visual Odometry," in 2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Ottawa, ON, Canada, Nov. 2019, pp. 1–5. doi: 10.1109/GlobalSIP45357.2019.8969460. **(Published).**

[2] H. Sadruddin, **A. Mahmoud**, and M. M. Atia, "Enhancing Body-Mounted LiDAR SLAM using an IMU-based Pedestrian Dead Reckoning (PDR) Model," in 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), Springfield, MA, USA, Aug. 2020, pp. 901–904. doi: 10.1109/MWSCAS48704.2020.9184561. **(Published).**

[3] H. Sadruddin, **A. Mahmoud**, and M. Atia, "An Indoor Navigation System using Stereo Vision, IMU and UWB Sensor Fusion," in 2019 IEEE SENSORS, Montreal, QC, Canada, Oct. 2019, pp. 1–4. doi: 10.1109/SENSORS43011.2019.8956942. **(Published).**

[4] **A. Mahmoud**, H. Sadruddin, P. Coser, and M. Atia, "Integration of Wearable Sensors Measurements for Indoor Pedestrian Tracking," *IEEE Instrum. Meas. Mag.*, vol. 25, no. 1, pp. 46–54, Feb. 2022, doi: 10.1109/MIM.2022.9693454. **(Published).**

[5] **A. Mahmoud**, H. Sadruddin, P. Coser, and M. Atia, "Ultra Wide-Band Automatic Grid Expansion for Indoor Path Tracking," in 2022 IEEE SENSORS, Dallas, Texas, USA, Nov. 2022. **(To be submitted).**

[6] **A. Mahmoud**, and M. Atia, "An Improved VSLAM System Using Semantic Segmentation and Layout Estimation," in Electronic Letters on Computer Vision and Image Analysis. **(To be submitted).**

1.6 Outline

The Thesis organization is as follows: Chapter 2 is devoted to background and literature review, including visual odometry, visual SLAM problem, inertial navigation and UWB positioning. Chapter 2 also includes a comparison and performance analysis of direct SLAM and feature-based SLAM approaches, which discusses the robustness and real-time issues of the tested algorithms. The chapter also discusses the robustness and real-time issues of SLAM algorithms. Chapter 3 illustrates the design and implementation of a multi-sensor navigation/logger embedded platform. The platform was developed to collect synchronized data indoors. Chapter 4 demonstrates a hybrid visual-inertial odometry system that combines the benefits of stereo and monocular vision positioning systems. The chapter then presents the sensor fusion system that integrates UWB-ranging sensors with IMU and stereo vision. Chapter 5 discusses the neuroscientific finding in navigational behavioural and spatial memory formation and then presents the design and the implementation of an improved VSLAM system using semantic segmentation and layout estimation. Finally, the thesis is concluded, and the future work directions are presented in Chapter 6.

Chapter 2: Background and Related Work

Indoor navigation systems are increasingly needed for different applications such as asset/people tracking and robotics. Although Global Positioning System (GPS) is widely used for positioning in numerous applications, it cannot be used for indoor navigation systems as the walls of buildings block satellite signals. As a result, other positioning sensors/systems are commonly used. Common sensors include inertial, visual, and local wireless ranging sensors. Individual sensors have limitations that make it impossible to rely on one system or sensor to provide indoor navigation. This chapter discusses the various current approaches used to find the location of a moving platform in an indoor environment. In this work, we have used the vision sensor as the main technology and other sensors (i.e., IMU and UWB) as aiding sources. Therefore, the chapter will start with an extensive background on visual navigation systems

2.1 Visual Navigation Systems

Visual navigation is accurately tracking the camera's ego-motion using the image sequence it receives as input. The camera as a sensor is cost-effective, lightweight, and provides rich information, making it suitable for navigation, mapping, obstacle detection, and path planning applications. Vision sensors are influenced by lighting conditions, the distortion caused by rapid motion, and their limited field of view. The initial objective of the visual navigation research was to develop visual odometry (VO) solutions that accurately estimate the camera trajectory by finding the relative transformation between two consecutive camera frames. Nevertheless, visual odometry methods were susceptible to the unbounded drifts inherent to all odometry solutions. These unbounded errors led researchers to construct environment models concurrently with trajectory estimation,

resulting in visual simultaneous localization and mapping systems (VSLAM). In this section, the primary steps of visual odometry and the fundamentals of VSLAM are described.

2.1.1 Visual Odometry

In VO solutions, motion tracking is achieved by tracking visual feature points in the environment. Observing how the relative pose between the camera and the tracked points evolves from one frame to the next incrementally builds the camera trajectory. Tracked points should be distinctive and reliably and repeatedly detected in images of the same scene, ideally under different viewpoints and illumination conditions. Once features have been extracted, the image can be discarded as feature-based methods only operate on these features.

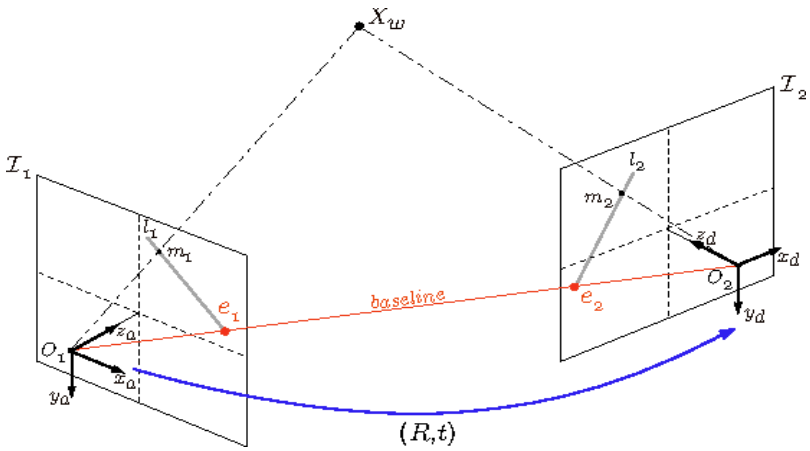


Figure 2.1 Epipolar Geometry of Stereo Vision.

Working on only features is faster and more accurate than direct methods. Therefore, most VO implementations rely on feature points. The main limitation of these approaches is that they can only work in texture-rich environments. Lack of texture can make feature-based methods perform very poorly or lose track. Using the Epipolar

constraint - illustrated in Figure 2.1- a stereo system could directly estimate the depth of any point in the stereo camera view by triangulation [7]:

$$Z = f.B/(x_l - x_r), \qquad (2.1)$$

Where Z is the point depth, f is the focal length B is the baseline, and $(x_l - x_r)$ is the disparity between the two images. The basic components of any VO solution are shown in Figure 2.2:

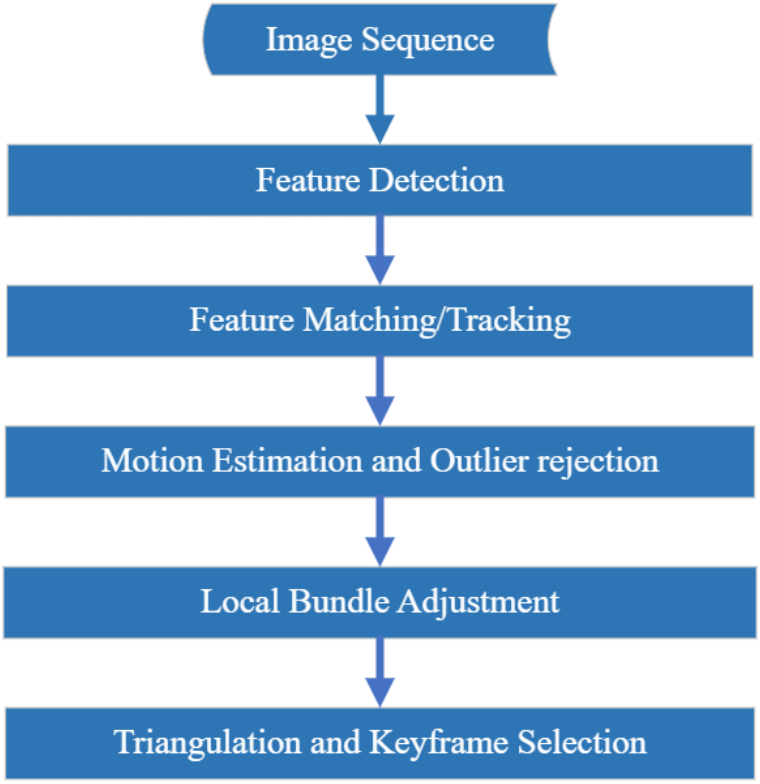


Figure 2.2. Visual Odometry Block diagram.

As illustrated in Figure 2.2, feature-based VO solutions have five main steps; each has been considered an independent research problem and has many solutions in the literature; each of these steps could be briefly summarized as follows:

Feature detection: The image is searched for salient keypoints that are likely to match well with other images. A local feature is an image pattern that differs from its

immediate neighbourhood in intensity, colour, and texture. The machine vision literature offers many corner point detectors such as Moravec [8], Forstner [9], Harris [10], Shi-Tomasi [11], and FAST [12], or blob detectors such as SIFT [13], SURF [14], and CENSURE [15]. For each keypoint, a descriptor is computed by operating on a patch of pixels around it; that descriptor is typically a vector of binary or real values of a certain length. Descriptors enable key-point matching across images simply by comparing their descriptors.

Feature Matching/Tracking: In this step, the extracted features from one image are searched for in another image. Feature matching detects features independently in the images and matches them based on some similarity metrics. In contrast, feature tracking consists of finding features in one image and then tracking them in the following images using a local search technique. Also, feature correspondences should not contain wrong data associations, namely outliers, for an accurate motion computation. Therefore, for the camera motion to be estimated accurately, outliers must be removed.

The solution to outlier removal is taking advantage of the geometric constraints imposed by the motion model. Robust estimation methods can be used, such as M-estimation [16], case deletion, and explicitly fitting and removing outliers [17]. In the presence of many outliers, RANSAC [18] has become the standard method for model estimation. RANSAC is a probabilistic and nondeterministic method, as previously stated [19]. As a result, it produces a different solution on each run; however, the solution becomes more stable as the number of iterations increases.

Motion Estimation: The camera motion between the current and previous images

is computed in this step. By concatenating all these single movements, the entire trajectory of the camera and the agent (assuming that the camera is rigidly mounted) can be recovered.

Transformation T_k between two images I_{k-1} and I_k can be computed from two sets of corresponding features f_{k-1} , f_k at time instants $k-1$ and k, respectively. Three different methods depending on whether the feature correspondences are specified in two or three dimensions.

- 2-D-to-2-D: In this case, both f_{k-1} and f_k are specified in 2-D image coordinates.
- 3-D-to-3-D: In this case, both f_{k-1} and f_k are specified in 3-D. It is necessary to triangulate 3-D points at each instant to acquire the 3-D point representations.
- 3-D-to-2-D: In this case, f_{k-1} are specified in 3-D and f_k are their corresponding 2-D reprojections on the image I_k.

Nister et al. [20] pointed out that motion estimation from 3-D-to-2-D correspondences is more accurate than 3-D-to-3-D correspondences. It minimizes the image reprojection error instead of the 3-D-to-3-D feature position error, eliminating the errors generated from wrong depth estimations. The transformation T_k is computed from the 3-D-to-2-D correspondences X_{k-1} and p_k : X_{k-1} can be estimated from stereo data. The general formulation, in such a case, is to find T_k that minimizes the image reprojection error.

$$\arg\min_{T_k} \sum_i \left\| p_k^i - \hat{p}_{k-1}^i \right\|^2, \tag{2.2}$$

where $\hat{p}_{k-1}^i$ is the reprojection of the 3-D point X_{k-1}^i into image I_k according to the transformation T_k . This problem is known as perspective n points (PnP) (or resection), and there are many different solutions to it in the literature [21].

Local optimization and bundle adjustment: Since VO works by computing the camera path incrementally (pose after pose), the errors introduced by each new frame-to-frame motion accumulate over time; this generates a drift of the estimated trajectory from the actual path. Therefore, it is essential to keep drift as small as possible, which can be done via local optimization over the last m camera poses. This approach, called sliding window bundle adjustment or windowed bundle adjustment, has been used in several works, such as [22], [23], [24], [25]. In particular, in a 10-km VO experiment, Konolige et al. [24] demonstrated that windowed-bundle adjustment could decrease the final position error by a factor of 2–5.

This iterative refinement works by minimizing the sum of the squared reprojection errors of the reconstructed 3-D points over the last m images (i.e. local map). The 3-D points are obtained by triangulation of the image points. Given a correspondence between a 3D point in world coordinates X_W and a 2D key-point x_C in a monocular camera, the reprojection error e_{proj} is computed as follows:

$$e_{proj} = x_C - \pi_m(\boldsymbol{R}_{Cw}\boldsymbol{X}_w + {}_c\boldsymbol{P}_w), \tag{2.3}$$

where $\boldsymbol{R}_{CW} \in SO(3)$ and ${}_C\boldsymbol{P}_W$ are the rotation and translation of the inverse of the camera pose, which transforms points from world to camera coordinates. The optimization of the positions of a set of points $\mathcal{P}$ and the poses of a set of cameras C, minimizing the reprojection error, is called Bundle Adjustment (BA) [26]:

$$\left\{\boldsymbol{X}_w^j, \boldsymbol{R}_{cw}^i, {}_c\boldsymbol{P}_w^i | \forall j \in \mathcal{P}, \forall i \in C\right\}$$
$$= \underset{\boldsymbol{X}_w^j \boldsymbol{R}_{cw}^j {}_c\boldsymbol{P}_w^i}{\operatorname{argmin}} \sum_{ij} \rho\left(\left\| x_i^j - \pi_m\left(\boldsymbol{R}_{cw}^i \boldsymbol{X}_w^j + {}_c\boldsymbol{P}_w^i\right)\right\|_{\boldsymbol{\Sigma}_i^j}^2\right), \tag{2.4}$$

where x_i^j is the key-point associated with the 3D point $\boldsymbol{X}_w^j$ in camera i, $\Sigma_i^j i$s the covariance of the location of keypoint x_i^j on the image of camera i, $\|.\|_\Sigma$ is the Mahalanobis distance, and ρ is a robust cost function to down-weight outlier correspondences.

Triangulation and Keyframe Selection: Motion estimation and bundle adjustment steps require triangulating 3-D points from 2-Ds image correspondences. Triangulated 3-D points are determined by intersecting back-projected rays from 2-D image correspondences of at least two image frames. In perfect conditions, these rays would intersect in a single 3-D point. However, image noise, camera model and calibration errors, and feature matching uncertainty never intersect. As a result, the point at the shortest distance from all rays can be used. When frames are taken at nearby positions, 3-D points exhibit very large uncertainty. A common way to avoid this is to skip frames until the average uncertainty of the 3-D projections decreases below a certain threshold. The selected frames are called keyframes. Keyframe selection is essential in VO and should always be done before updating the motion.

2.1.2 Visual Simultaneous Localization and Mapping

In Simultaneous Localization and Mapping (SLAM), the system is incrementally building a map representation while tracking its position [27]. In this section, the SLAM problem formulation is discussed then we introduce the graph SLAM approach.

SLAM Problem Formulation:

The accuracy of the trajectory estimation and map formulation is mutually dependent. SLAM solutions use the constructed map to allow localization in the same environment without continually accumulating drift. Also, accurate position estimation is crucial as an inaccurate localization will produce an erroneous map, making inferring

precise future positions even harder [28]. Figure 2.3 is a simple illustration of SLAM unknowns.

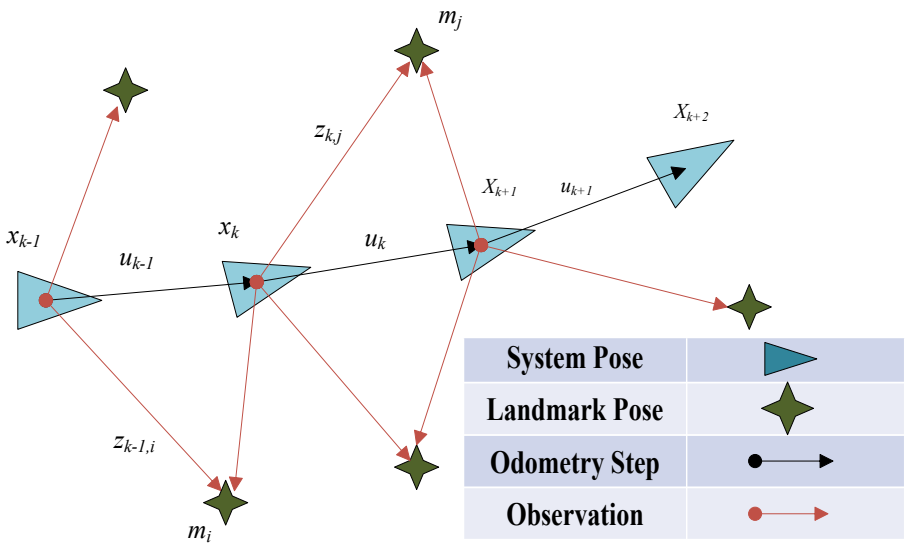
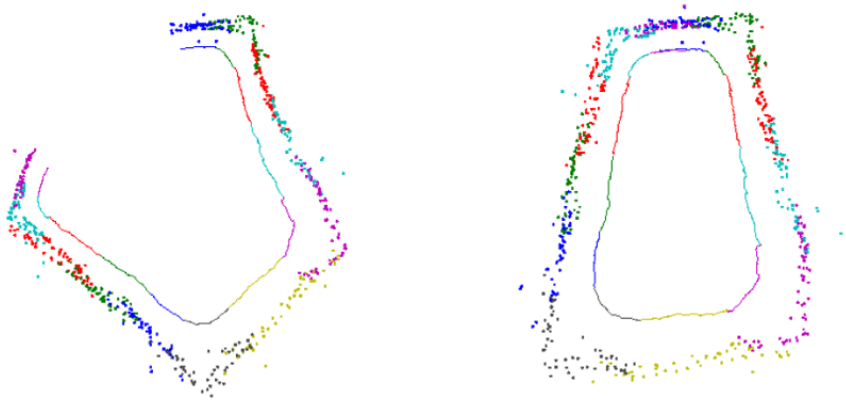


Figure 2.3 Localization and Mapping Unknowns.

SLAM solutions correct VO drifts by recognizing previously visited places and then eliminating the drifts accumulated between the two visits. This process in SLAM is called a loop closure, illustrated in Figure 2.4.



(a) Estimated Trajectory Before Loop Closure

(b) Optimized Trajectory After Loop Closure

Figure 2.4 Loop Closure Illustration.

However, recognizing a previously visited place requires the observed landmarks to be associated with one of the landmarks on the map [29]; this process is called data association. A false-positive data association could result in an unusable map and the system losing track of its position. The complexity of the SLAM problem comes from the noise in sensor measurements. These erroneous measurements make neither the pose estimation nor the perceived environment structure estimated accurately. SLAM solutions are usually approached using probabilistic approaches to cope with uncertainties.

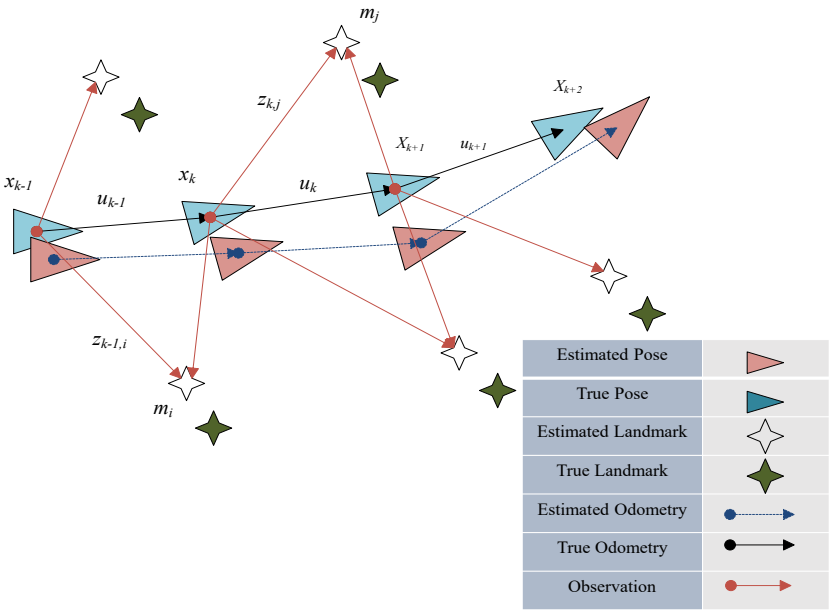


Figure 2. 5. Essential SLAM Problem Formulation.

Consider a mobile platform moving through an environment taking relative observations of several unknown landmarks using a sensor located on the system, as shown in Figure 2. **5**.

At any time instance k, the following mathematical quantities should be estimated:

- x_t : The state vector that describes the location and orientation of the vehicle.

- u_t : The odometry vector, at time $t-1$ that drives the system to the state x_t at time t.
- m_i : A vector describing the location of the i^{th} landmark whose actual location is assumed time-invariant.
- z_{it} : the observation was taken to the relative pose of the i^{th} landmark at time t.

In probabilistic form, the online SLAM problem could be expressed as:

$$P(x_t, m | z_{1:t}, u_{1:t}), \tag{2.5}$$

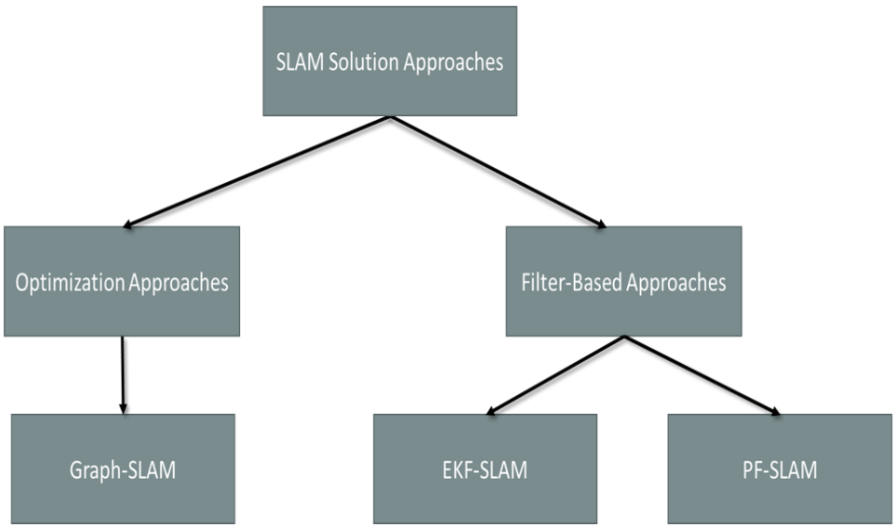


Figure 2.6 SLAM Solution Taxonomy

For many years, filter-based methods, such as extended Kalman filters or particle filters, have been mainstream SLAM solutions. However, the past decade witnessed technological and mathematical evolution, which led the focus of recent developments away from filtering toward nonlinear least-squares optimization approaches that exploit the sparsity inherent in the SLAM problem. Figure 2.6 shows the taxonomy of the approaches that SLAM solutions use to estimate the trajectory and the environment model jointly.

Graph-Based SLAM:

State-of-the-art SLAM solutions reformulate the SLAM problem as a least-squares optimization problem and use nonlinear optimization approaches to solve it. Such SLAM solutions usually have a typical structure that divides the system into two parts, a so-called front-end and a back end.

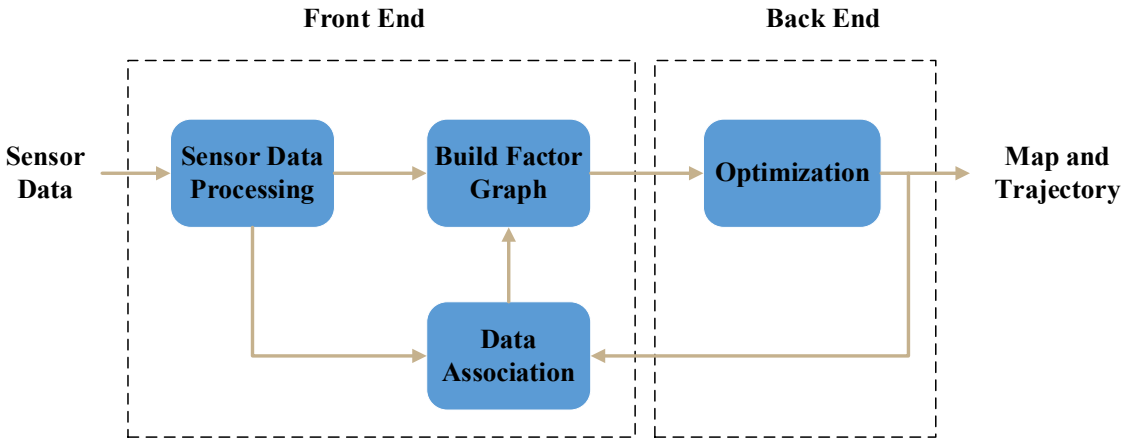


Figure 2.7. Graph SLAM Framework

Figure 2.7 illustrates the flow of information in a graph SLAM system. The front-end performs the typical visual odometry to estimate the localization priori for the optimization problem and creates a factor graph [30]. The second part of a graph SLAM system is called the back end, which solves the optimization problem encoded in the graph to gain a map and a position estimation. Then, the back end solves the optimization problem by finding the graph nodes' configurations that best align with the received measurements. Factor graphs are a powerful probabilistic tool for factorization over probability distributions.

The current standard formulation of Graph-SLAM has its origins in the work of F. Lu et al. [31], followed by [32]. Since then, numerous approaches have improved the efficiency and robustness of the optimization underlying the problem [33], [34], [35], [36],

[37], [38]. These approaches treat the SLAM task as a maximum a posteriori estimation (MAP) problem, and they frequently employ the formalism of factor graphs graphs [30] to reason about variable interdependence. Large-scale SLAM has been demonstrated to scale well with these graph-based representations [39], [40], [41].

Unlike Kalman filtering, MAP estimation does not require a clear distinction between observation and motion models; both are treated as factors and seamlessly integrated into the estimation process. Graph-based SLAM solutions have many advantages over filter-based approaches:

- Graphs allow for a clear visual representation of the problem. Figure 2.8 depicts the factor graph of a typical SLAM problem. The sequence of camera nodes, $x_1, x_1, \ldots$ are linked to odometry commands, $u_1, u_2, \ldots$ that constrain the camera's relative poses, $C_1, C_2, \ldots$ camera calibration parameters K could also play a role.

- Simple incorporation of the loop-closure factors as relative pose constraints by registering previously visited landmark observations with current views.

- Furthermore, factor graphs can incorporate additional sensory data, such as inertial measurement units (IMU), to improve camera trajectory and mapping accuracy.

- The sparsity of the factor graphs enables fast linear solvers [41], [42]. Moreover, it allows the implementation of online solvers, which update the estimate of X as new observations are acquired [36].

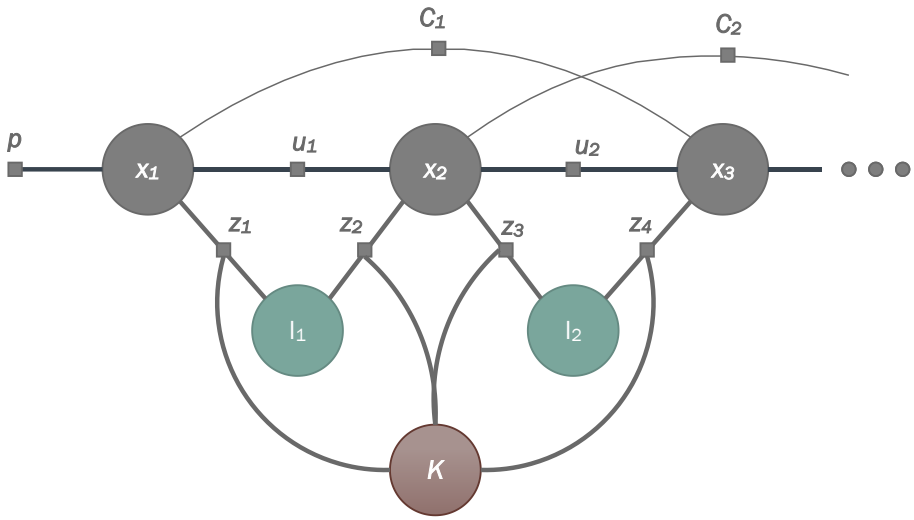


Figure 2.8. Illustration of Pose SLAM problem

Current SLAM libraries (e.g., GTSAM [40], g2o [43], Ceres [44], iSAM [42], and SLAM++ [45]) are able to solve problems involving tens of thousands of variables in a matter of seconds. Noticeably, EKF SLAM and Graph-SLAM estimation performance get comparable when the linearization point for the EKF is accurate, which happens in visual-inertial navigation problems [46],[47], [48].

2.1.3 Performance Analysis of Common VSLAM Frameworks

This section conducts a performance analysis of two existing open-source V-SLAM solutions, ORB-SLAM2 [49] and LDSO-SLAM [50]. These two solutions are considered the de facto V-SLAM approaches. Multiple SLAM solutions comparisons are presented in the literature. In [51], a theoretical comparison of many monocular VSLAMs, including ORB-SLAM and direct sparse odometry (DSO). Additionally, [50], [52] presented accuracy and robustness analysis between different V-SLAM solutions on publicly available datasets. This section performed further analysis to study the real-time performance and accuracy bottlenecks of common VSLAM frameworks. It is worth mentioning that we have tested more recent feature-based open-source solutions such as

SOFT-SLAM [53], OV2-SLAM [54], OpenV-SLAM [55], Kimera [56] and ORB-SLAM3 [57]; however, these solutions are based on the original ORB-SLAM2 work and with comparable or similar performances and weaknesses.

ORB-SLAM2

Mur-Artal et al. build on their previous work ORB-SLAM [49] by presenting ORB-SLAM2 [58]. ORB-SLAM2 is a feature-based SLAM system that extracts FAST features and calculates its ORB feature descriptors. The detected features are used throughout all the ORB-SLAM2 threads because ORB features are robust against feature rotation. Additionally, ORB descriptors have a lower time complexity than other popular descriptors (SIFT and SURF) while maintaining a high matching accuracy.

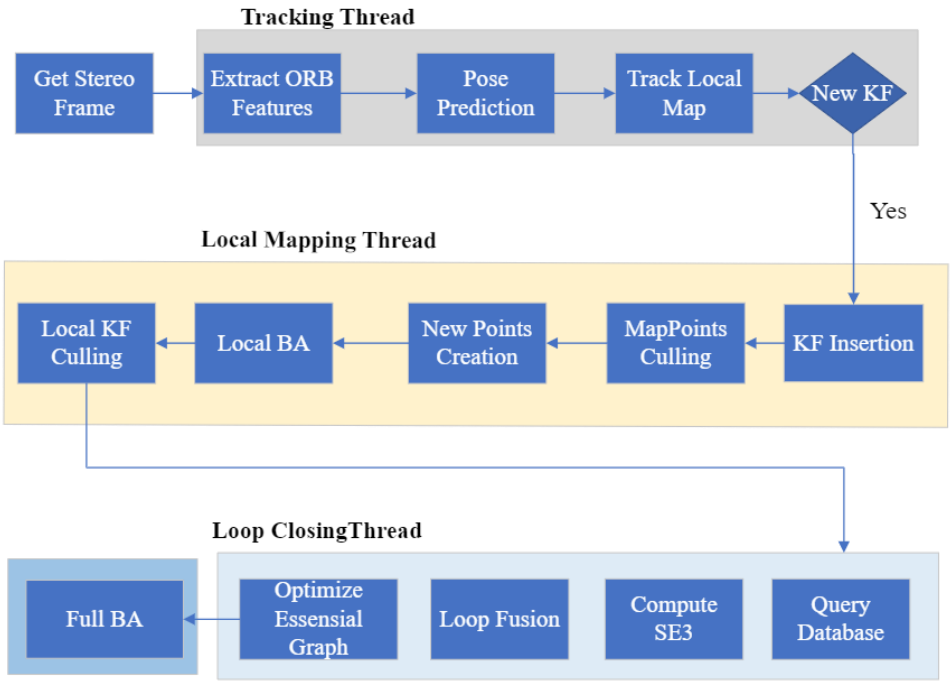


Figure 2.9 ORB-SLAM2 Flow Diagram.

ORB-SLAM2 can be used with a monocular camera as an input, similar to the original ORB-SLAM, or stereo or depth cameras to increase accuracy by removing scale ambiguity. The system has three threads: tracking, local mapping, and optimization. Figure

2.9 shows the ORB-SLAM2 flow diagram. Although stereo and depth cameras are used to reduce scale error, the back-end tracking and optimization process for these two sensors are identical to that for the monocular camera. The system begins by extracting keypoint features and calculating the descriptor associated with them. The image frame is fed into the tracking thread. Once the system has been initialized, it uses Random Sample Consensus (RANSAC) [18] to find enough matches between the current and previous frames'. It then uses these matches to estimate the current camera pose based on the previous frame's camera pose prediction and then tracks the local map. Finally, the tracking thread checks if the local map needs a new keyframe to be inserted.

ORB-SLAM2 processes only keyframe to generate new map points and optimization because most frames contain redundant information that is not worth maintaining. The Bag-of-Words (BOW) technique accelerates the ORB descriptor matching. BoW [59] is a method for visual location recognition. The descriptor matching speed can be increased by constructing a vocabulary tree and discretizing a binary descriptor space.

The redundant and invalid keypoints and keyframes are deleted to reduce memory and computing costs. The number of keyframes would not continue to grow indefinitely if the scene remained static. ORB-SLAM2 reduces the computing cost of the optimization thread by utilizing two new graph structure concepts: visibility graph and essential graph. Otherwise, redundant keyframes would increase the number of the optimization graph edges, making the optimization task more complex. The Covisibility graph is a weighted undirected graph. Each node represents a keyframe, and an edge exists between two keyframes if they share a threshold number of observations of the same map points. The

edge's weight is equal to the number of shared map points. The essential graph is the visibility graph's minimum spanning tree.

ORB-SLAM2 continuously optimizes map points and keyframes throughout its runtime. Once the pose of the new frame is approximated in the tracking thread, the system optimizes the local map using bundle adjustment (BA) to obtain a more accurate pose estimate [26]. When a new keyframe is created, the system optimizes the current keyframe and the frames connected to it in the co-visibility graph using local BA. Additionally, global BA is used to optimize the essential graph when the system detects a loop. The pose of map points is refined as new keyframes arrive, as they would be seen from various perspectives.

ORB-SLAM2's superior performance has been demonstrated on various publicly available datasets [58], [49]. However, it has several flaws. To begin, the initialization step is time-consuming. A valid initial map requires a large number of keypoints. As a result, initializing the system typically takes some time, even in an environment rich in features.

LDSO-SLAM

As a large-scale, direct monocular SLAM solution, Engel et al. presented LDSO-SLAM [60]. While ORB-SLAM2 adopts the traditional VO steps presented in section 2.1.1, LDSO tracks the camera position by operating directly on the pixel’s intensity. LDSO is a sparse version of the dense LSD-SLAM [61] and follows its general design and flow diagram. Tracking, depth map estimation, and map optimization are the three main threads of LDSO-SLAM. The solution presented a novel scale-aware image alignment algorithm that directly estimates the transformation matrix between two frames. In

addition, LDSO-SLAM estimates depth using probabilistically consistent incorporation in tracking to reduce scale drift. The LDSO-SLAM flow diagram is presented in Figure 2.10.

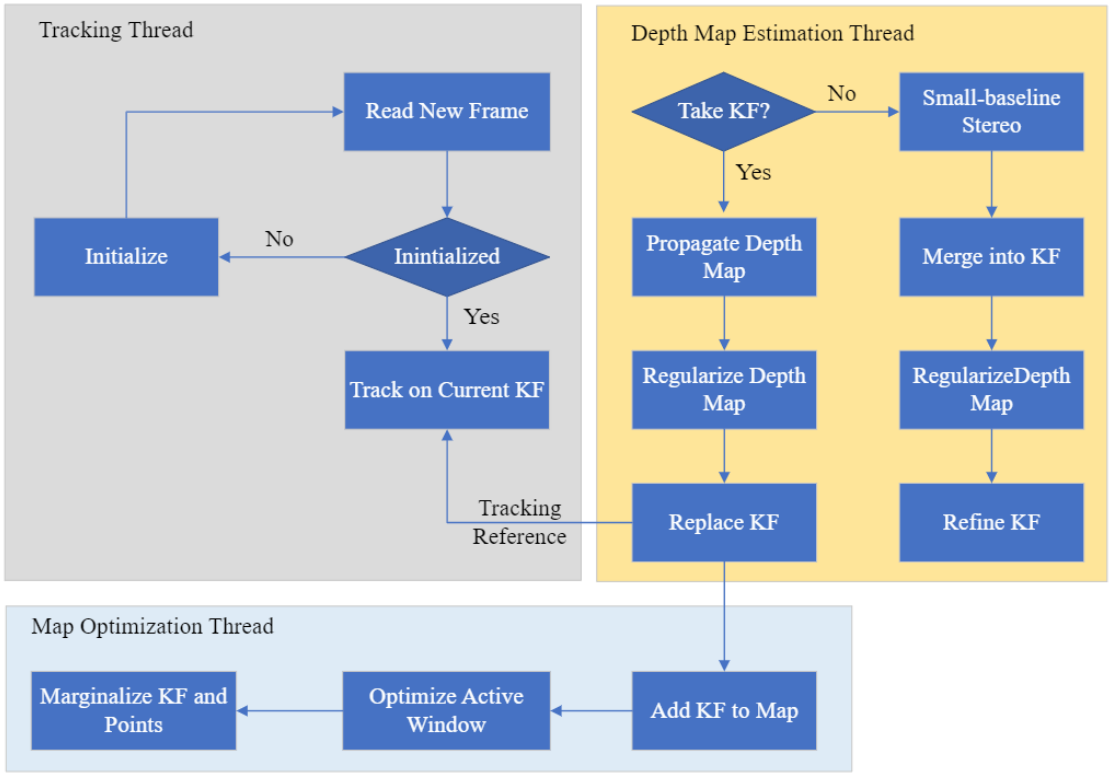


Figure 2.10 LDSO-SLAM Flow Diagram.

An initial map is created with a random depth map and a significant variance from the first keyframe during the initialization step. When the system receives a new image, the tracking thread minimizes the variance-normalized photometric error to estimate the transformation matrix between the current and reference keyframes. Photometric error measures the difference in intensity between two frames based on the assumption that a point's projections in different frames have the same intensity. Let the intensity of point p in frame i with the pose, T_i is represented by $I\ (p, T_i\ )$. The photometric error between frames i and j is:

$$e_{ij} = \sum_{p \in P} (I\ (p, Ti)\ -\ I\ (p, Tj\ )), \qquad (2.6)$$

P stands for all of the points in the frame. By minimizing the photometric error, e_{ij}, the relevant alignment similarity transform matrix between frame i and frame j, $T_{ij} = T_j T_i^{-1}$ can be estimated. The system creates a new keyframe from the current frame if the camera moves too far away from the existing map. Otherwise, the map is refined according to the current frame by the system. Each keyframe created is saved in the map structure, which triggers the optimization thread.

LSD-SLAM algorithm proposed a method for performing direct, scale-drift aware image alignment on transformation matrix by using the photometric residual e_{ij}. Furthermore, all keyframes in the map are scaled to have a mean inverse depth of one. As a result, because all keyframes are optimized simultaneously on the same scale, LDSO-SLAM performs well in a large-scale environment. To detect large-scale loop closures, like ORB-SLAM2, LDSO-SLAM uses a feature-based mapping algorithm. The closest keyframes are then chosen to use reciprocal tracking checks to prevent the insertion of false or incorrectly tracked loop closures. Finally, the system optimizes the map using g2o [62], [63].

LSD-SLAM and LDSO-SLAM have some flaws as well. As mentioned in the original paper on LSD-SLAM, it necessitates a very precise initialization. If the tracking is affected by a texture-less area or variations in lighting, pose estimation accuracy decreases [61]. LDSO-SLAM will not work in a dynamic environment because objects interfere with image alignment. LDSO-SLAM and direct SLAM, in general, are sensitive to noise in the frame because it affects depth estimation.

Datasets

The tests were performed on data collected at the Minto Case and the Mackenzie Buildings at Carleton University in Ottawa. The data collection stage used two different stereo cameras, Intel Realsense T265 [64] and StereoLab ZED2 [65]. Realsense T265 is equipped with fisheye lenses and provides gray-scale images. StereoLab ZED2 has standard lenses, and both provide position estimation using built-in tracking solutions. Seven different image sequences were collected in different scenarios, four ZED2 cameras and three T265 sequences. Due to the absence of a ground truth indoors, we compared the estimated trajectory of the proposed solution to the built-in Intel Realsense trajectory solution.

Lab sequence is a short sequence collected by the ZED2 camera in a laboratory area. The environment is very well lit with natural light and highly textured, making this sequence perfect for visual tracking applications.

Square sequences*:* are two short sequences with each camera in a square-shaped corridor. The place was artificially lit and had a moderate number of visual features.

Stairs sequences*:* we recorded two image sequences in the same environmental conditions as the square sequences. However, the sequences were more complicated, traversing between two floors using the stairs and moving under different light sources in two different building sections.

Finally, ***Eight-shape sequences****:* again, both cameras were used in an eight-shape trajectory on a low-textured floor with very repetitive scene appearances and combined normal and artificial light sources.

Evaluation Metrics

In this performance analysis, three metrics have been used; root mean squared error, frame rate, and memory usage. The three metrics are described as follows:

Root Mean Squared Error (RMSE): The root mean squared error is a common metric used to indicate the accuracy of the estimated trajectory. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_{pred}^{i} - P_{GT}^{i})^2}, \tag{2.7}$$

where N is the total number of time instances in which we have a position estimation, P_{pred} is the predicted trajectory and P_{GT} is the Ground Truth (GT) at the same point in time. As there is no ground truth available for our tests, we used the Intel Realsense T256 solution as a reference. This decision is based on many experiments. The Intel camera proved that its position estimation (no map provided) is fairly accurate with limited scenarios in which its tracking fails.

Frame rate and real-time tendency: We compare the output frequency and perform CPU profiling on both algorithms using Google gperf-tools [66]. The CPU profiler in gperf tools is an efficient tool that provides the CPU load of every single function in a software system. In addition, it interrupts the processor periodically and checks the processor context. It points out portions of the code that consume the CPU the most and must be optimized and sped up.

Memory usage and solution embeddability: We calculated each algorithm's memory requirements using another Google tool, the TCMalloc library [67]. TCMalloc library tracks each defined variable in the program, counting the used memory bytes and

identifying the memory leaks. This memory profiling gives the necessary information to analyze the feasibility of running a specific algorithm on a specific embedded platform. For example, it shows the required memory for the program initialization and the memory overhead of adding a new frame to the estimated environment map. Finally, using these calculations alongside the power requirement for each embedded platform tells how long the system could operate on a single run.

Results and Analysis

This section presents our comparative test results for ORB-SLAM and the LDSO-SLAM. Our analysis covers the robustness and accuracy of the solution and the embeddability and real-time tendency.

Robustness and Accuracy:

Overall results are given in Table 2.1. As we can conclude from Table 2.1, ORB-Stereo gives the lowest RMSE in all scenarios, implying the best available accuracy; however, the proper judgement cannot be made without reviewing the produced trajectories as it completes the overall picture. This section uses "-t" to identify the Realsense T265 sequences and "-z" for the ZED2 sequences.

Table 2.1 RMSE for ORB-Stereo, ORB-Mono and LDSO.

Sequence	Trajectory	ORB-Stereo	ORB-Mono	LDSO-Mono
LAB-z	62.8 m	2.9 m (4%)	6.04 m (9%)	5.25 m (8.3%)
Square-t	55.0 m	14.03 m (25.5%)	-	-
Square-z	103.5 m	10.3 m (10%)	16.9 m (16.4%)	21.5 m (20.7%)
Eight-t	111.0 m	10.19 m (9.1%)	-	-
Eight-z	221.85 m	18.08 m (8.1%)	19.3 m (8.7%)	18.9 m (8.5%)
Stairs-t	148.6 m	21.27 m (14.3%)	-	-

ORB-Stereo successfully tracked the complete inputted sequences and provided a trajectory in all tested scenarios with the best RMSE compared to the ORB-Mono and the LDSO-mono. In contrast, the ORB-Mono and LDSO-Mono could not track the camera's position in all the fisheye sequences. The reason for the failure of the monocular solutions is the complicated process of triangulating feature points. A feature point can be triangulated directly using the epipolar constraint given a single stereo frame. However, to triangulate a feature using a single camera feed, the feature needs to be tracked for at least three consecutive frames. The results of the monocular solutions were greatly affected by the scale ambiguity in the single-camera approaches in both the sparse direct and feature-based methods.

At the same time, the ORB-Stereo managed to provide more accurate trajectories even when it could not detect the loop closures. The Eight-shaped sequences have a similar repetitive structure which misled both ORB-SLAM2 and LDSO-SLAM to accept a wrong loop closure. Repetitive geometrical structures are common in indoor environments, and without proper geometrical and geographical validation could easily lead to mutated trajectory estimation.

Table 2. 2 Loop closing and relocalization capabilities of visual navigation solutions.

	Type	Mono	Stereo	RGB-D	Loop Closing	Relocalization
DSO	VO	✓	✓	-	-	-
LDSO	SLAM	✓	-	-	Feature points	Feature points
ORB-SLAM2	SLAM	✓	✓	✓	Feature points	Feature points
Kimera	VIO/SLAM	✓	✓	-	Feature points	-
OpenVSLAM	SLAM	✓	✓	✓	Feature points	Feature points

Finally, Table 2. 2 shows the existing visual navigation solutions' loop closing and relocalization capabilities. Both loop closing and relocalization capabilities are commonly implemented in the existing visual SLAM solutions to correct positional drifts and recover from any tracking loss; however, these capabilities still depend on feature points which cause a low rate of relocalization success and a high rate of wrong loop closures.

Real-time performance:

As aforementioned, we created both CPU and memory profiles to determine whether it is feasible to run each tested algorithm in real-time and how much memory is needed in a given embedded platform for the algorithm to run on it. As presented in Table 2.3, ORB-Mono has the highest frame rate per second (FPS) while maintaining the lowest memory needs from the hardware platform. However, 1400 MB for 2 minutes sequence is not commonly available in most commercial embedded platforms. A more intelligent scheme for loading/unloading portions of the created maps to/from the main memory is to solve this problem.

Table 2.3 Real-time and memory analysis for ORB-Stereo, ORB-Mono and LDSO-Mono.

	FPS	Init-Mem	1K Frames	4K frames
ORB-Stereo	10	635 MB	200 MB	1760 MB
ORB-Mono	29	635 MB	80 MB	1400 MB
LDSO-Mono	9	920 MB	250 MB	2000 MB

Noticeably, LDSO-Mono has high memory requirements and a low frame rate, which should not be the case, given that it only processes a monocular camera stream. These high memory and processing requirements can be explained because LDSO-SLAM

creates frames containing sparse, raw pixel values for the direct methods and the ORB features used in loop detection.

Finally, ORB-Stereo does not have a high frame rate; however, based on the CPU profile, around 50% of the CPU load is consumed to detect the ORB features in the stereo pairs and find the stereo matches. Having these bottlenecks spotted, we can solve the problem by offloading these functions from the CPU to a GPU to parallelize the execution and get a higher frame rate.

Table 2.4 Memory requirement distribution for SLAM threads.

	Tracking	Local Mapping	Loop Closure
LDSO-Mono	39.4%	29.7%	15.1%
ORB-Mono	43.2%	33.4%	17.3%
ORB-Stereo	60.4%	25.2%	9.8%

Table 2.4 illustrates the memory requirement distribution between different SLAM threads, showing the increased amount of memory needed to track stereo stream rather than monocular. Processing power distribution between different threads is presented in Table 2.5. It is worth mentioning that loading and decoding images take 15-34% of the system's total processing power.

Table 2.5 Processing power distribution for SLAM threads.

	Tracking	Local Mapping	Loop Closure	Image loading / decoding
LDSO-Mono	50.1%	6.6%	2.5%	24.2%
ORB-Mono	25.2%	24.4%	2.7%	34.1%
ORB-Stereo	63.1%	17.8%	2.0%	15.6%

2.1.4 Semantic SLAM

Purely geometry-based maps could not provide conceptual knowledge of the surroundings to facilitate complex tasks; as a result, associating semantic concepts with geometry entities in the environment has recently become a popular research domain. Semantic SLAM incorporates semantic information into the SLAM process to improve overall performance by providing task-driven perception, resilience, and high-level understanding. Incorporating semantic data into a SLAM pipeline has been around for a long time. Nonetheless, the implementation was not achievable until recently, at least in a meaningful fashion, because extracting semantic information from visual data was nearly impossible until the epoch of deep learning. When fast and accurate semantic segmentation DNN architectures existed, the semantic SLAM drew much attention. However, because this occurred less than a decade ago and despite several contributions, the topic of integrating semantics in SLAM is still far from mature [68]. This section identifies and summarizes the primary topics of interest in semantic SLAM solutions.

Semantics for loop closing:

Since it is challenging to detect a loop closing purely based on geometric features, robust loop closing is an open problem in SLAM. Changes in the observed environment, such as moving objects or changes in illumination, can cause dramatic changes in the geometry of a scene. When semantics are used, the problem becomes much easier to solve. Some researchers have concentrated their efforts on mathematically formulating the problem of combining geometric and semantic information into a single optimization framework for semantically associating observations and performing robust loop closing [69]. Others used semantic information to train a neural network to extract 3D descriptors

of the scene[70]. To detect loop closures, 3D descriptors are compared rather than words in a bag of binary words setting [71].

Semantics for handling dynamic environments:

Traditional SLAM pipelines have a critical flaw: they operate under the assumption that the environment is static. The RANSAC algorithm is commonly used in SLAM to eliminate erroneous feature correspondences. However, when a significant portion of the view is occupied by moving objects, this approach fails, causing the estimation process to diverge. Semantics can be used in various ways to deal with this situation. Semantic Segmentation is used to detect potentially moving parts of the image and exclude them from the tracking process entirely in a simple but effective approach. Mask-SLAM [72], which uses DeepLabv2 [73] to perform precise semantic scene segmentation, is one approach. Mask-SLAM discards ORB features detected within regions occupied by vehicles or the sky because vehicles move and the sky is too far away to show any parallax.

Before discarding potentially moving objects found using semantic Segmentation, more complicated methods check if they move. DS-SLAM [74], for example, examines whether the matched features are close to the epipolar line between frames. If multiple matched features belonging to a potentially moving object are far from the epipolar line, the object is considered moving. Li et al. [75] proposed a more complicated approach to avoid moving objects when estimating camera motion, but instead of discarding them, try to track them independently through time. They use an off-the-shelf deep network to get 2D bounding boxes of vehicles, then train their own CNN to create 3D bounding boxes based on the 2D ones. Based on this information, they can then track each moving vehicle

separately using a motion model and an object bundle adjustment to the 3D bounding boxes camera static.

Semantic reasoning within an unknown environment:

Semantic reasoning appears to be the most researched among the methods listed above. Since this is probably the simplest way of integrating semantics in SLAM, much research has created a dense or sparse semantic map of the environment. In this case, the SLAM framework assigns a semantic class to each map point produced by a semantic segmentation method. Even though the semantic maps generated by such methods can enable advanced interactions between the system and the outside world, most do not use them to improve the SLAM's robustness.

Building semantic maps with objects rather than 3D points is a unique approach. Object detectors have been used in place of feature detectors in QuadricSLAM [76],[77]. The detected objects are inserted into the map in a dual quadric representation. The dual quadrics positions and the camera pose are then estimated using a bundle adjustment. Because ignoring traditional features reduces SLAM's accuracy, Hosseinzadeh et al. [78] proposed integrating dual quadric representations of objects and traditional 3D points. Instead of abstract object representations, a more accurate but computationally expensive and complicated solution integrates detailed volumetric object reconstructions in the semantic map. By fusing the geometry of objects in successive planes, the object reconstructions are gradually refined over time. Similar concepts are implemented in a few published works, such as MaskFusion [79] and Fusion++ [80].

A more straightforward concept has been developed by Wang et al. [81]. As a starting point, ORB-SLAM2 [49] is employed. Then, the YOLO object detector [82] is

used to classify the frame's keypoints on each frame semantically. A voting system then propagates semantic information from keypoints to corresponding map points. All keypoints associated with the map point "vote" once for their semantic class. After integrating semantic information, semantics are used in feature matching, tracking, and loop closing. More specifically, associations between keypoints and map points of different classes are prohibited or reduced based on an adjustable ratio.

2.2 Inertial Navigation

The operating principle of inertial navigation is based upon Newton's first law of motion, which states that an object at rest will remain at rest. An object in uniform motion will continue to exhibit that motion unless an external force is external acts upon it.

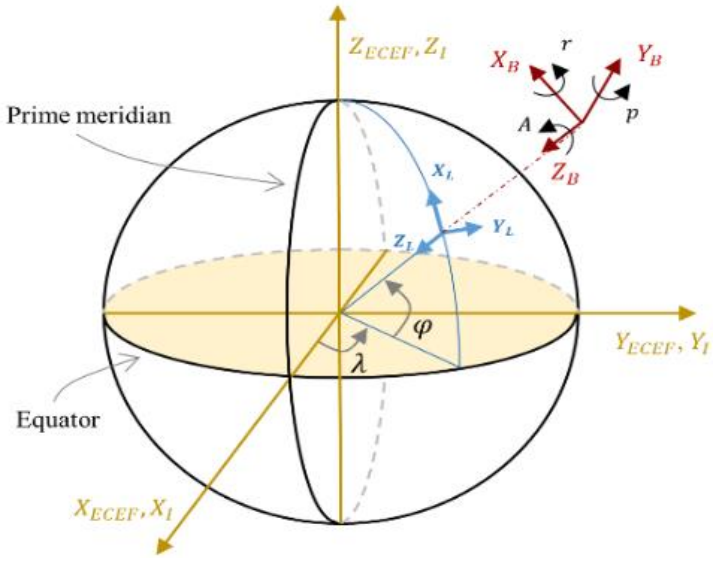


Figure 2.11 INS System of Coordinate Frames

The external force is directly proportional to the body's inertia (linear acceleration and rotation rate), which can be sensed by an Inertial Measurement Unit (IMU). IMU is a self-

reliant device that belongs to a class of proprioceptive sensors which do not need information from an external source.

Table 2.6 The axes and origin of coordinate frames in INS solutions

Frame	Definition
B: Body Frame	Origin: Platform center of mass X: Longitudinal (forward) direction Y: Transversal (lateral) direction Z: Down (vertical) direction
L: Local Level navigation frame	Levelled with Earth's ellipsoid surface Origin: Vehicle location X: True north direction Y: East direction Z: Down vertical direction
E: Earth-Centered Earth-Fixed (ECEF)	Rotates with Earth Origin: Center of the Earth Z: Extends through the North Pole. X: Passes through the intersection of the equatorial plane and the prime meridian. Y: completes the right-hand coordinate system in the equatorial plane.
Earth-Centered Inertial Frame (I)	Coinciding with ECEF at a specific initial time and it does not rotate with Earth

An Inertial Navigation System (INS) is a navigation method that provides positioning information by processing raw IMU data. INS is a common component of aircraft, ships, automobiles, and cellphones [83]. A typical IMU comprises three mutually orthogonal accelerometers and three orthogonal gyroscopes. Accelerometers measure the body's acceleration in a body frame B. However, to provide meaningful navigation data, acceleration measurements are transformed into a local level frame L, commonly in the North-East-Down (NED) representation [84]. Gyroscope (and possibly magnetometer)

measurements are used to calculate the object's orientation to perform this transformation. The definition of the coordinate frames [85] used in INS navigation is summarized in Table 2.6, and it is visually illustrated in Figure 2.11.

2.2.1 INS Motion Equations

The dynamic model of an Inertial Navigation System (INS) defines how a rigid body's position, velocity, and attitude evolve dynamically through time. The rigid body fundamental rotation equation is defined as follows [86]:

$$\dot{q}_B^L = \frac{1}{2}\Omega(\omega_{IB}^B)q_B^L + \frac{1}{2}\begin{bmatrix} 0 & -\omega_{IB}^{B\,T} \\ \omega_{IB}^B & -[\omega_{IB}^B]_\times \end{bmatrix}\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{2}\begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix}\begin{bmatrix} 0 \\ \omega_{IB_x}^B \\ \omega_{IB_y}^B \\ \omega_{IB_z}^B \end{bmatrix}, \tag{2.8}$$

where ω_{IB}^B is the corrected rotation rate of the B frame with respect to the I frame as represented in the B frame, q_B^L is the attitude of the body frame with respect to the L frame, and $[\,]_\times$ is the skew operator defined as:

$$[\omega]_\times \triangleq \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{2.9}$$

The velocity of the moving platform in the L frame changes according to [86]:

$$\dot{v}_{LB}^L = a_{iB}^L + g^L - (2\omega_{iE}^L + \omega_{EL}^L) \times v_{LB}^L, \tag{2.10}$$

where $v_{LB}^L = [v_n, v_e, v_d]$ is the velocity vector in the L frame along the NED axes, a_{IB}^L is the corrected acceleration measurement in the L frame, g^L is the gravity vector in the L frame, and the operator $\times$ is the cross product. In the above equation, ω_{IE}^L and ω_{EL}^L are, respectively, the Earth's rotation rate and transportation rate calculated as follows [86]:

$$\omega_{IE}^{L} = [\omega_{Ie}\cos(\varphi)\,, \quad 0, \quad -\omega_{Ie}\sin(\varphi)\,], \tag{2.11}$$

$$\omega_{EL}^{L} = \left[\frac{v_e}{(R_n + h)}, \quad \frac{-v_n}{(R_m + h)}, \quad \frac{-v_e}{(R_n + h)}\tan(\varphi)\right], \tag{2.12}$$

where $\omega_{Ie} = \left(2\pi/(24 \times 60 \times 60)\right) rad/s$ is the Earth's rotation rate represented in the I frame, and φ, λ, h are, respectively, the latitude, longitude, and altitude. R_m and R_n are, the meridian radius of curvature and normal radius of curvature of the Earth's ellipsoid model. The position of the B frame in the L frame evolves according to the following equation [86]:

$$\dot{P}_{LB}^{L} = v_{LB}^{L} - \omega_{EL}^{L} \times P_{LB}^{L}, \tag{2.13}$$

where P_{LB}^{L} is the position of the B in the L frame.

2.2.2 IMU Measurement Errors

Primary sources of error in IMU's measurements are bias, scale factor, and random walk noise. Figure 2.12 shows how these errors affect the IMU measurements. The error model for IMU's measurements is defined as follows:

$$a_{iB}^{B} = R_{I}^{B}(s_a \circ a_{iB}^{-I} - b_a) + n_a, \tag{2.14}$$

$$\omega_{iB}^{B} = R_{I}^{B}\left(s_g \circ \omega_{iB}^{-I} - b_g\right) + n_g, \tag{2.15}$$

Where a_{iB}^{B} and a_{iB}^{-I} are, the corrected and raw accelerometer's measurements, ω_{iB}^{B} and ω_{iB}^{-I} are, the corrected and raw gyroscope measurements, $R_{I}^{B} = R\{q_{B}^{L}\}$ is the rotation matrix that corrects the B and I frames misalignment. s_a and s_g are, respectively, accelerometer's and gyroscope's scale factor, b_a and b_g are, respectively, accelerometer's and gyroscope's biases, and n_a and b_g are zero-mean Gaussian noise.

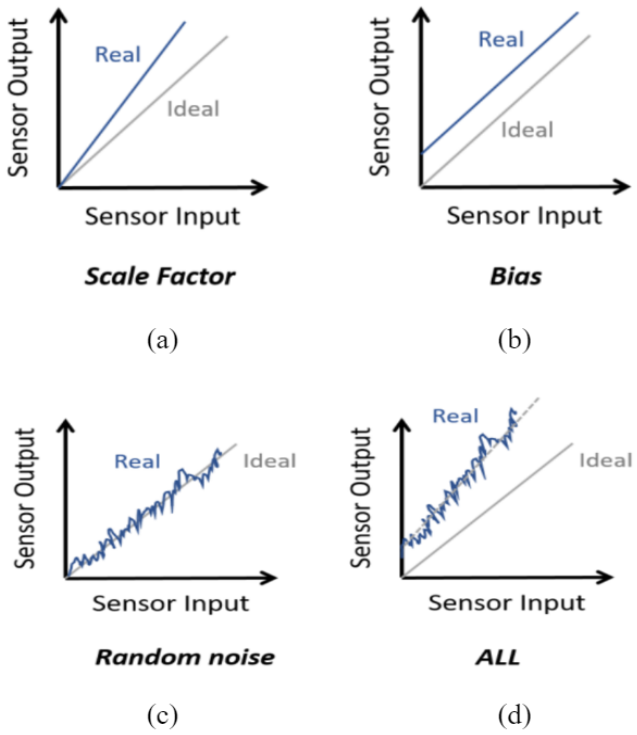


Figure 2.12 Sources of IMU measurement errors.

Commercial IMUs often suffer from stochastic behaviour in their measurement errors. To model the stochastic behaviour of scale factor and bias errors of the IMU, the first-order Gauss-Markov (GM) random processes are commonly used as follows [87]:

$$\dot{x} = -\frac{1}{\tau_x}x + \sqrt{\frac{2}{\tau_x}\sigma_x{}^2 n_x}, \tag{2.16}$$

where x is the IMU error, τ_x is a time constant, n_x is zero-mean Gaussian noise, and σ_x is the standard deviation of the GM process.

2.3 Wireless Positioning using Ultra-wideband Technology

Recently, ultra-wideband technology (UWB) has been introduced as an indoor positioning technology using the concept of wireless ranging and localization, a technique similar to GNSS but applied indoors. The system consists of fixed anchors and mobile

antennas (TAGs), and it uses active measurements for distance estimation based on Time of Flight (ToF), Time of arrival (ToA, TDoA), and angle of arrival (AoA) principles [88]. Using UWB technology in localization approaches yields accurate estimations under clear line-of-sight (LOS) conditions as distance estimation is directly correlated to the signal bandwidth. Knowing the UWB uses large frequency bandwidth, allowing high resolution in time and range. This technology is quite suitable for precise indoor positioning. UWB is especially useful for an environment where multipath is high because the wide bandwidth facilitates the detection of multiple Time-delayed versions of signal sequences.

2.3.1 Two-way Round-Trip ToA Ranging

UWB Transceivers can be grouped into anchors with known locations and tags with unknown locations. Once range measurement is available between a tag and at least three anchors, the tag location can be estimated using trilateration. The movable device gets this information by messages via a UWB network; the ranges are computed by a Two-Way-Round-Trip (ToA) Range method. Double-Sided Two-Way-Range (DS-TWR) [89] estimates the distance based on the time of flight (ToF) of messages exchanged between two devices. Since the speed of the propagation of radio waves can be considered constant in a particular environment, the range can be obtained if the measure of the two-way time-of-flight is accurate enough.

Suppose (x_i, y_i) denotes the x and y coordinates of n anchors where $(i = 1,2,..,n)$ and $T = (t_x, t_y)^T$ represents the unknown tag coordinates, then the distance between an anchor and tag is calculated as follows:

$$d_i = \sqrt{(t_x - x_i)^2 + (t_y - y_i)^2}, \tag{2.17}$$

2.3.2 UWB Positioning using Least-Squares

Given the availability of ranges from at least three different anchors, a dataset containing the range information is created. Then, to obtain the tag position, the least square-based trilateration method is used as follows [90]:

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = (A^T A)^{-1} A^T b \quad , \tag{2.18}$$

Where,

$$A = \begin{pmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ \vdots & \vdots \\ 2(x_1 - x_n) & 2(y_1 - y_n) \end{pmatrix} \quad , \tag{2.19}$$

And

$$b = \begin{pmatrix} d_2^2 - d_1^2 + x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ \vdots \\ d_n^2 - d_1^2 + x_1^2 - x_n^2 + y_1^2 - y_2^2 \end{pmatrix} \quad , \tag{2.20}$$

Secondly, the resulted distances are verified, and errors are calculated. Finally, the positions with the highest errors are removed, and the best estimate is considered incumbent. Also, a quality factor based on the calculated errors is reported.

It is worth highlighting the availability of high accuracy anchors' positions is a key factor in tag position trilateration which means UWB positioning is only possible in a structured environment in which the anchors are placed, and their positions are measured accurately.

2.3.3 UWB Positioning errors

Although UWB has many advantages for a variety of indoor navigation applications, it also has some weaknesses. For instance, misconfiguration may cause interference with nearby systems that operate in the ultra-wide spectrum [91]. Also, while it is known that UWB systems are resistant to multipath issues, they are not completely immune to its effects [92]. Additionally, designing and implementing antennas for UWB systems can be

more difficult than the bandwidth and variable operating conditions. This may impose some restrictions on UWB systems compared to conventional RF.

2.4 Summary

This chapter provided background about commonly used sensors for indoor navigation, namely, vision, inertial, and wireless ranging sensors. The next chapter will demonstrate the multi-sensor logging/navigation system that was developed during this thesis to collect indoor data and perform all tests.

Chapter 3: Multi-sensor Testbed for Indoor Environments

Our work on sensor fusion to improve indoor navigation solutions faced the challenge of the unavailability of suitable multi-sensors datasets. Publicly available datasets are application-specific (designed specifically for an application such as autonomous vehicles and collected mostly outdoors under open-sky clear conditions). One example of such is an open-source KITTI dataset [93]. They also lack important modern tracking technologies, such as local wireless ranging technologies like ultrawideband (UWB). Another option is to collect the needed data for the application at hand. A major problem when collecting datasets for sensor fusion is accurate data acquisition, as sensor fusion algorithms require accurate sensor data time-synchronization. Moreover, it is hard to acquire an accurate ground truth.

This chapter demonstrates the development of an efficient multisensor indoor navigation system composed of off-the-shelf, low-cost commercial sensors needed for almost every autonomous navigation and SLAM research integrated with multiple connected embedded platforms. The developed system adopts client-server architecture where multiple embedded platforms are reported to a central server. The server is dedicated to real-time synchronization, visualization, and storage management. The embedded platforms control and interface with the body-mounted sensors. The entire system is implemented and organized under the umbrella of the Robot Operating System (ROS). Thus, all the implemented software programs are encapsulated within ROS Nodes.

The proposed system comprises two main sub-systems; Data Acquisition and Processing System (DAPS) module and the Remote-Controlled System (RCS) module. All the sensors are integrated on embedded platforms using the DAPS module, while the RCS

module monitors and controls the DAPS module's status and operations. The integrated sensors include a stereo vision camera system, multiple Inertial Measurement Units (IMUs), a Global Navigation Satellite System (GNSS) receiver, UWB wireless system (consists of a tag mobile receiver and multiple anchors) in addition to 3D LiDAR sensor. The presented system in this chapter offers the following novelties:

- A multi-sensor data collection platform for indoor/outdoor navigation provides an accurate ground truth.
- The system comprises a stereo camera, multiple IMUs, GNSS receiver, UWB positioning network, and LIDAR. These sensors span a wide range of navigation and positioning technologies.
- The system utilizes multiple processing units to interface and logs the sensor's data and dedicates a separate network for the bandwidth-demanding LIDAR sensor allowing the datasets to be uninterrupted and accurately synchronized.
- Finally, the system provides redundant IMUs to capture the full aspects of the motions in human/humanoid applications.

3.1 Hardware Architecture

Figure 3.1 shows the hardware configuration of the developed system. The system has a central server, and it incorporates state-of-art sensors that span visual, inertial, laser, local wireless networks, and global satellite positioning technologies. The sensors are configured and controlled by multiple computing platforms that communicate with each other wirelessly and exchange ROS messages. This section describes the individual sensors and the computing platforms.

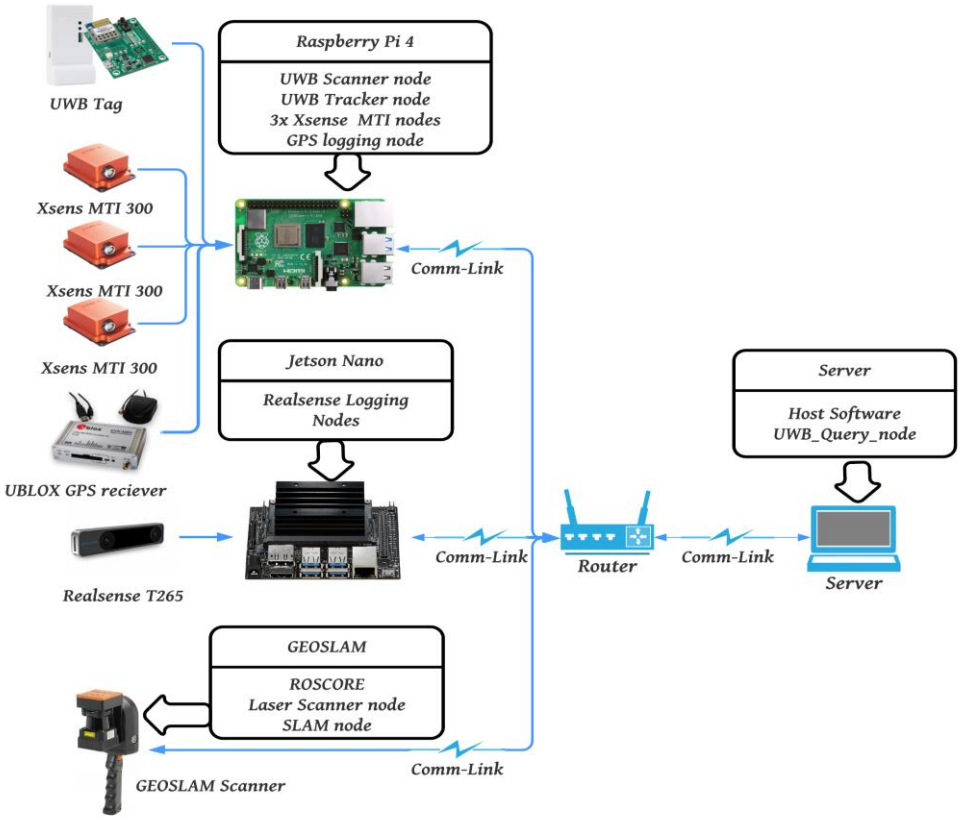


Figure 3.1 System Configuration (Node Distribution View)

3.1.1 Sensors Overview

This subsection summarizes the specifications of the utilized sensors, including the hardware and the logged sensory data.

Stereovision Tracking Module:

We use the Intel Realsense T265 Tracking Stereovsion camera module [64]. This stereovision tracking camera module outputs 3D pose data by processing inputs from dual fisheye cameras and IMU using a vision processing unit (VPU) from Movidius MA215x ASIC. The IMU is a system-in-package for detecting acceleration in 3 dimensions and rotations in 3 dimensions. The fisheye images are used to produce 6DoF data streamed to

the host platform. The imagers provide monochrome images at a high frame rate of 30 fps and 800x800 active pixels; the images are in 8bit, 10-bit RAW format.

The stereo camera field of view is close to the hemispherical shape with a 163±5° field of view to sustain robust tracking even with fast motion. 3D pose data are transmitted to the host platform at a sample rate of 200Hz. Also, raw images from left and right cameras are serialized at a real-time rate to the server through the GPU-enabled platform Jetson Nano board from NVIDIA.

Handheld 3D LiDAR Scanner System:

We used a "ZEB-Revo" portable 3D laser scanner from "GeoSLAM"[94] that provides accurate 3D point cloud and pose data. The maximum range of the laser scanner is 30m, and the data acquisition rate is 43,200 points/sec with a resolution of 0.625° horizontal, 1.8° vertical and an angular field of view of 270° x 360°. As the user walks through the area of interest, the data is captured on a body-mounted Remote Terminal (RT) that transmits pose, point-cloud, raw IMU, and raw LiDAR scans in real-time to the server. The RT performs a post-processing trajectory and point-cloud optimization when the data logging is concluded. In this stage, the system detects any loops and reduces the overall drift by applying loop closures. This post-processing optimization leads to a high-end optimized trajectory. The GeoSLAM 3D LiDAR SLAM system post-processed solution's accuracy was verified through loop errors and visual overlay of the results on floor maps of the area.

High accuracy IMU and AHRS

To provide precise spatial alignment that enables accurate transformation between different sensor frames, we have used three motion tracker MTI-300 [95] from Xsens. The

Xsens MTi IMU includes a MEMS-based IMU combined with a triad of magnetometers in a lightweight, low-power small package. The Xsens MTi IMU unit has an onboard sensor fusion engine designed for high performance under vibrations and magnetic distortions. The attitude heading & reference system (AHRS) unit [96] outputs stabilized magnetically referenced heading along with accurate roll and pitch estimates.

GNSS Receiver

We used a Global Navigation Satellite System (GNSS) receiver M8T EVK from U-Blox [97]. The GNSS receiver is serially connected to the logging system through a Raspberry Pi 4 Model B board. The system can save both the position fixation and the raw pseudorange and pseudorange rates data at 1Hz.

Decawave Ultrawide Band (UWB) Kit

We incorporated an ultrawideband (UWB) wireless setup into the developed system to evaluate wireless indoor positioning technology. We used the MDEK UWB kit from Decawave based on the DWM1001 module [98]. The kit consists of 12 nodes; each node has a Bluetooth communication module and UWB ranging/communication module. The node can be configured either as an anchor or as a tag. The developed system serializes both position and raw wireless range data to the server.

3.1.2 Computing Platforms

The implemented system runs on three different computing platforms, which exchange ROS messages over wireless media. Each computing platform constitutes a ROS node. Having multiple small computing platforms makes the system more resistant to single-point failures and allows a more flexible workload distribution that exploits the

computing platforms' diverse capabilities. The overall system's tasks can be classified into three categories:

1. User interface (visualization and control).
2. Sensors' interface.
3. Sensors' data processing (pose and map estimation).

All these tasks have different memory and processing requirements, which require different computing capabilities. As Figure 3.1 shows, three main computing platforms are used, a Linux-based server running ROS, Raspberry Pi 4 board, and Jetson Nano GPU-enabled platform. In addition to storage management, the server is used for visualization and user interface tasks. The Raspberry Pi 4 [99] has a 4GB memory with a sufficiently capable processor to process low bandwidth sensor data. Raspberry Pi runs six different nodes in the presented system controlling three Xsens MTi IMUs, Decawave UWB tag and UBlox GNSS receiver and communicating their data to the server.

Moreover, we dedicated 'NVIDIA's Jetson Nano [100] for the Intel Realsense T265 tracking stereovision camera node and having the onboard GPU dedicated for the camera with its parallel computing and machine learning capabilities. Having this kind of flexibility and diversity in the system makes it easier to scale and add more features. Finally, it is worth mentioning that the GeoSLAM 3D LiDAR SLAM system has its processing unit encapsulated in the remote terminal (RT). It publishes all its sensor readings directly to the server.

3.1.3 Wireless Connection Bridge

The communication between the different computing platforms is implemented wirelessly through a locally connected Wi-Fi-TCP communication module. ROS provides

all the necessary tools to use Wi-Fi-TCP communications. The ZEB-Revo GeoSLAM 3D LiDAR SLAM system provides its own local portable wireless network and ROSCore node. The two separate networks can handle high bandwidth streaming from camera feeds and the 3D LiDAR platform. A TP-Link AC1900 wireless router was used to augment the GeoSLAM 3D LiDAR SLAM system local network. Statistic IPs were assigned to all the computing platforms in the developed system to communicate effectively over the ROS communication bridge.

3.2 Software Architecture

The software architecture of the system was based on the distributed ROS environment. The implemented software comprises two main interactive software packages: the data acquisition and processing software (DAPS) and the remote-control software (RSC). Figure 3.2 depicts the tasks' distribution of the implemented software.

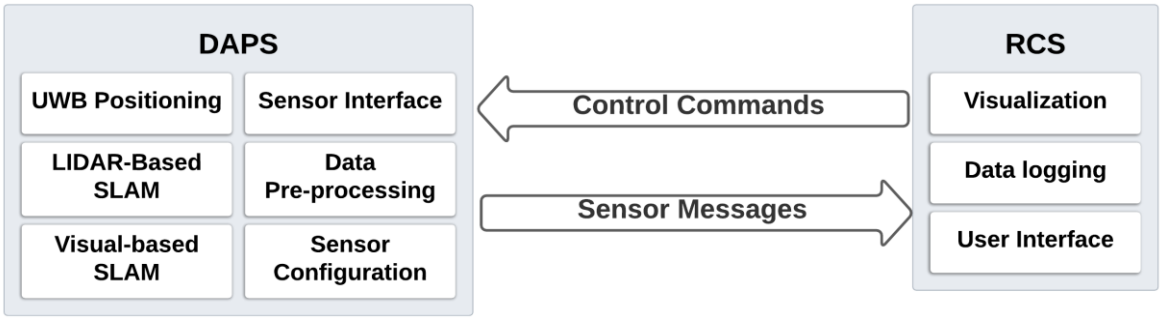


Figure 3.2 Software Architecture (Task Distribution)

3.2.1 Data Acquisition and Processing Software (DAPS)

The DAPS combines the ROS nodes, which run on the embedded platforms that interface with and control the sensors and prepare the sensory data to be transmitted to the RCS. DAPS package is also designed to perform pose estimation based on the logged sensory data, offloading any extensive optimization calculations to the remote server. The

modules running within the DAPS package are shown in Figure 3.2 on the left side. For each sensor, one or more ROS node was developed in C++ under Linux to configure and control the sensor, then encapsulate each data frame into the proper ROS message and then transmit it to the RSC. This distributed processing framework guaranteed real-time performance.

3.2.2 Remote Controlled Software (RCS)

As shown in Figure 3.2 (the right side), the RCS is running on the server and managed by the Qt framework [101], which is known for its real-time efficiency and suitability for embedded systems with a graphical user interface (GUI). The general software tasks implemented by the RCS are summarized as follows:

1) It provides an easy-to-use GUI.
2) It Initiates, control, and monitor the system modules.
3) Logging received ROS messages into a ROS bag file.
4) It provides real-time 3D visualization.

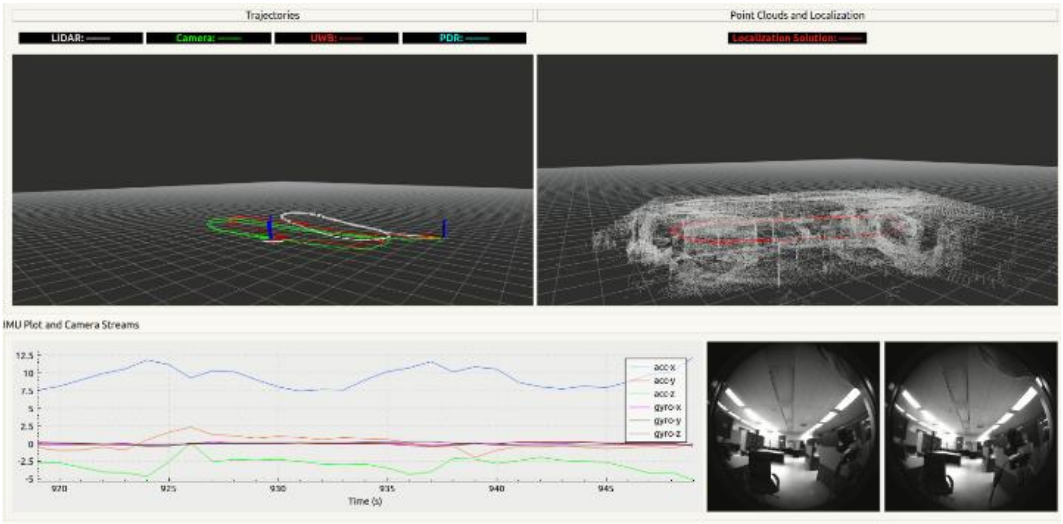


Figure 3.3 RCS – Visualization

Figure 3.3 shows a screenshot displaying the localization results, 3D point cloud map, camera frames and the raw IMU (accelerometer and gyroscope) data.

3.3 Tracking Algorithms

3.3.1 Pedestrian Dead Reckoning (PDR)

IMU was used for step counting in the developed system to measure forward motion. The person's step length is then projected in the direction of motion to estimate the person's new position. This technique is called Pedestrian Dead Reckoning (PDR). PDR uses acceleration to detect steps by applying a peak detection algorithm [84]. The peak detection is applied to the vertically projected acceleration data, which shows strong peaks corresponding to human walking steps. Then an attitude and heading reference system (AHRS) is used to project the user's step in the motion direction.

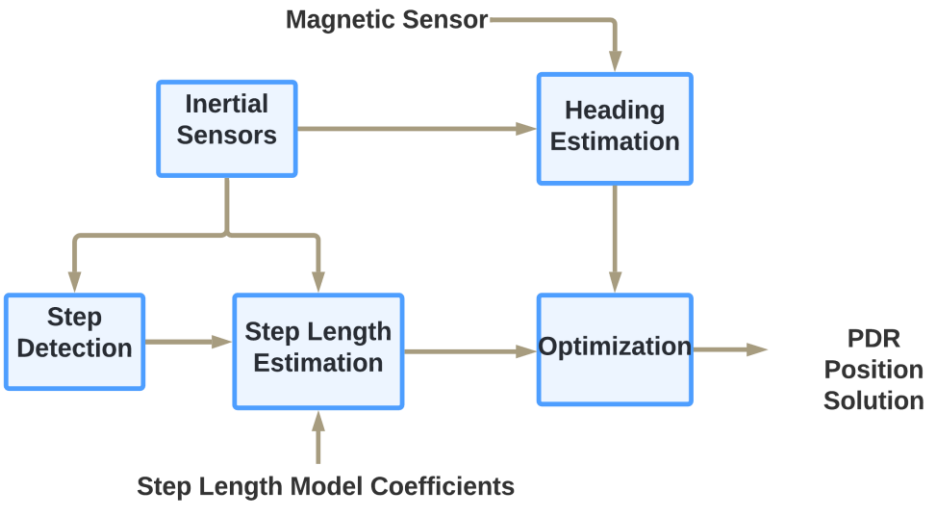


Figure 3.4 PDR- General Block Diagram

By concatenating the estimated relative motion steps, the whole trajectory is constructed. The step length is estimated using different techniques such as empirical models or machine learning methods in some algorithms. We kept the algorithm simple by using a

fixed empirically estimated step length in our implementation. A general block diagram of PDR is shown in Figure 3.4.

3.3.2 LIDAR Odometry and Simultaneous Localization and Mapping (SLAM):

A LiDAR scan can be projected as a point cloud where each beam is represented by a cartesian coordinate of a point in 3D space. A relative pose between two successive scans can be estimated using scan-matching algorithms. Scan-matching is the process of finding the rigid body transformation (a combination of translation and rotation) that optimally aligns two point-cloud scans. One popular algorithm that performs scan-matching is the Iterative Closest Point (ICP) approach [80], shown in Figure 3.5. ICP algorithm iteratively revises the transformation needed to minimize an error metric, usually a distance from the source to the reference point cloud, such as the sum of squared differences between the matched pairs' coordinates. This odometry process results in a trajectory construction that is vulnerable to drifts

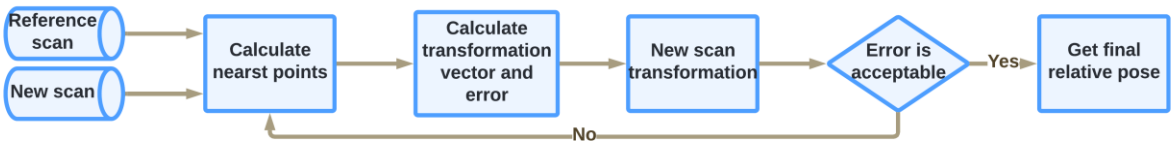


Figure 3.5 Iterative Closest Point (ICP) Algorithm

3.3.3 UWB Wireless Ranging and Positioning System

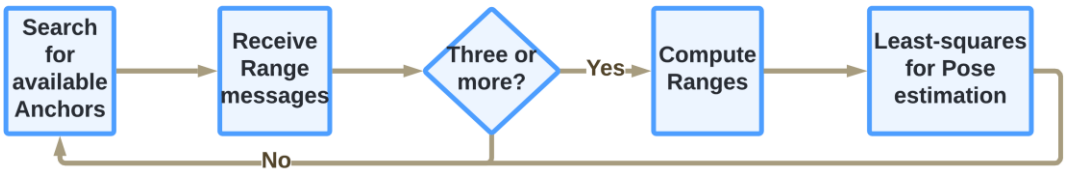


Figure 3.6 UWB Tag pose estimation

In this work, we used the Decwave MDEK1001 wireless module where radio-ranging is implemented using round-trip Symmetrical Double-Sided Two-Way Ranging (SDS-TWR) [102]. Figure 3.6 shows the basic UWB tag pose estimation process. UWB Transceivers can be grouped into anchors with known locations and tags with unknown locations. Once range measurement is available between a tag and at least three anchors, the tag location can be estimated using trilateration [90]. For a more detailed discussion on the UWB positioning, refer to section 2.3.

3.4 Experiment and Results

3.4.1 Experiment Setup

The system was tested, and data was collected at Carleton University (CU) campus in Ottawa, ON, Canada. The full test trajectory overlayed on the Carleton University campus map is shown in Figure 3. 7; the green portion of the trajectory is the outdoor portion, while the blue is the indoor section. The indoor section included multi-floor walks by stairs to add versatility to the dataset. The logging started indoors in the Minto Building lab area, where all the UWB Anchors were mounted at pre-defined locations, as seen in Figure 3. 8. Upon exiting the lab office, the test navigated different indoor places in the Minto Building, including four floors of stairs. The testing trajectory continues outdoors, exiting the building from the southwest side, entering the building again from the northeast side, and returning to the lab office area to the same point we started performing several loops.

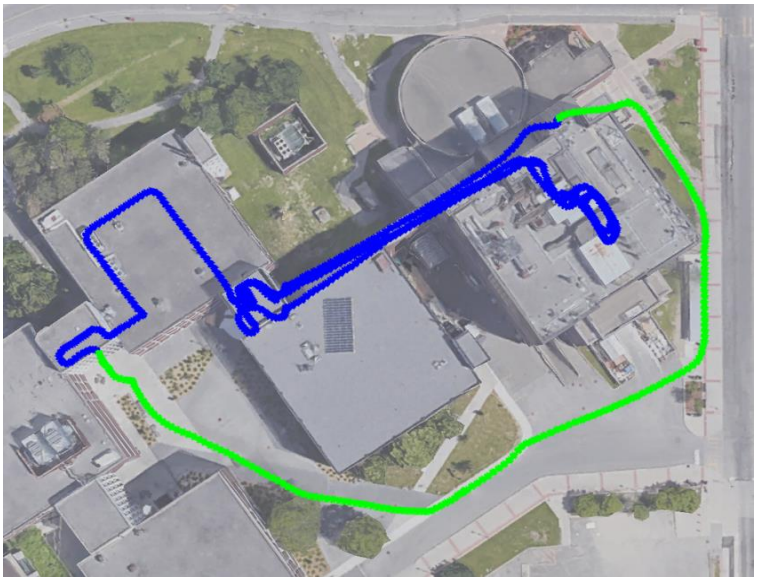


Figure 3. 7 Full test trajectory: blue is indoors, and green is outdoors – Minto Building, Carleton University.

The test trajectory included several loop closures to enable SLAM loop closures to reduce errors and drifts. Each sensor provides data in its body frame of reference. However, a high-accuracy AHRS MTi unit was rigidly attached to each sensor (LiDAR scanner, Stereo vision platform, and chest-mount IMU). Therefore, the spatial transformation between individual sensors can be obtained from the orientation estimated by the Xsens MTi IMU AHRS modules.

3.4.2 Results and Analysis

This section demonstrates sample results and trajectory comparisons between the different sensors' solutions. Some segments that provide different test scenarios, such as indoor/outdoor segments and climbing stairs segments, are demonstrated. The Decawave UWB Anchors were installed only in the office in the reported experiment.

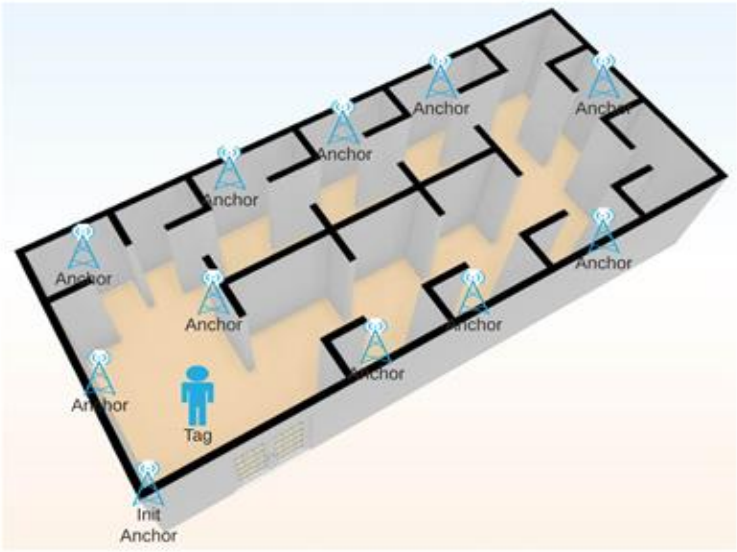


Figure 3. 8 UWB anchors' distribution in the lab area.

Figure 3. 9 shows all the GeoSLAM 3D LiDAR SLAM system trajectory estimation stages showing the iterative loop-closure effects. It shows multiple copies of the trajectory in red. Loop closures iteratively enhance the trajectory. The final post-processed optimized trajectory is shown in Figure 3. 7.

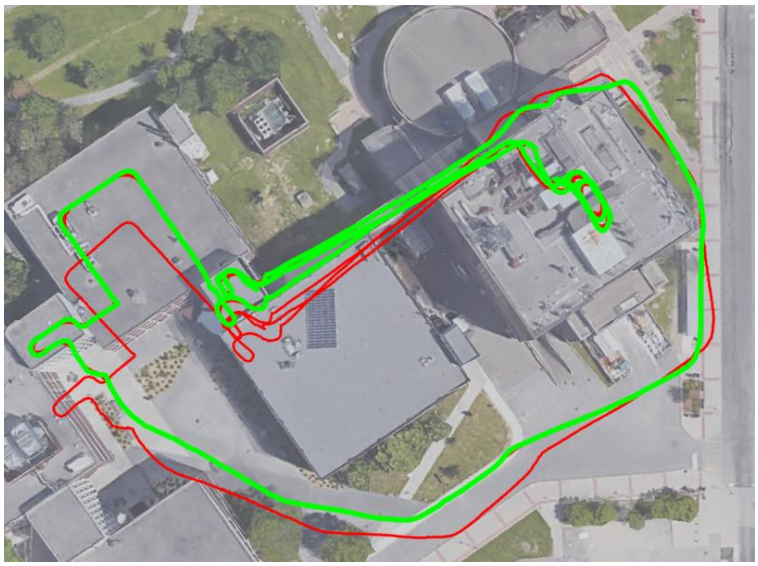


Figure 3. 9 GeoSLAM estimated trajectory before (Red) and after optimization (Green) - Carleton University.

The optimized post-processed 3D point-cloud map generated by GeoSLAM software is shown in Figure 3. 10. The lab is a typical indoor area with a short loop and rich visual and spatial features. Therefore, the stereovision results are consistent with the LIDAR solution with a small scale difference, as shown in Figure 3.11. Figure 3.12 shows the UWB solution demonstrating great decimeter-level accuracy thanks to the dense and elevated distribution of the UWB anchors.

Figure 3.13 shows the results of the IMU-based PDR solution showing moderate deviation from the reference. The results show that the performance of the stereovision tracking solution is comparatively smoother than UWB. The path from the office-area exit to the outdoor south-east side of the building shows transiting from indoors to outdoors, as demonstrated in Figure 3. 7. This section included 4-floors stairs. Figure 3.14 shows the stereovision solution against the LIDAR solution.

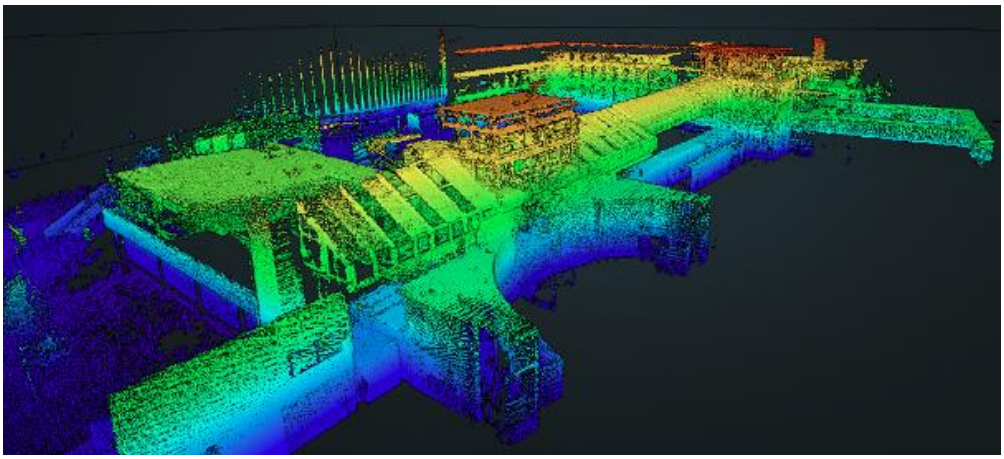


Figure 3. 10 LiDAR Point Cloud Map Solution (3D View)

After we exited the indoor area, a clear drift happened between the stereovision and the LIDAR solutions. In the outdoor space, the stereovision generated longitudinal drifts due to the short baseline between the right and left cameras, which leads to a lack of close, evenly spatially distributed visual features, which is pertinent for cameras to generate good

pose information. The PDR solution showed good performance within the building even during the stairs but gradually drifted later, as Figure 3.15 shows the typical behaviour of PDR technology. The performance in the multi-level stairs section is demonstrated in Figure 3.16 Experiment Path Segment - Stairs section.

The implemented PDR solution does not provide vertical positional estimation, so we display only the GeoSLAM SLAM and the stereovision solutions. For a complete video of the experiment, readers are referred to the video link [103].

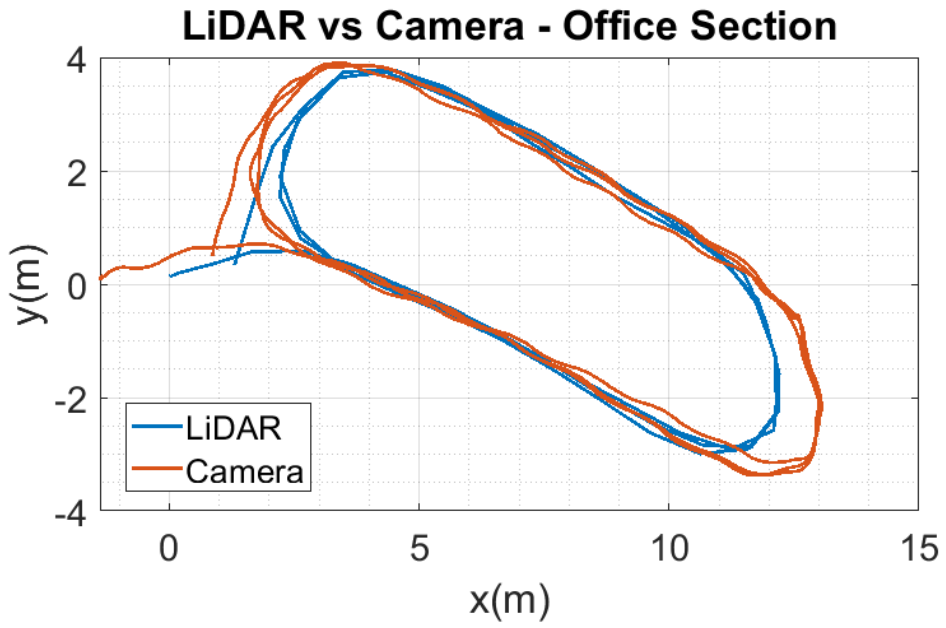


Figure 3.11 Stereovision vs LIDAR - Indoor Segment - Office.

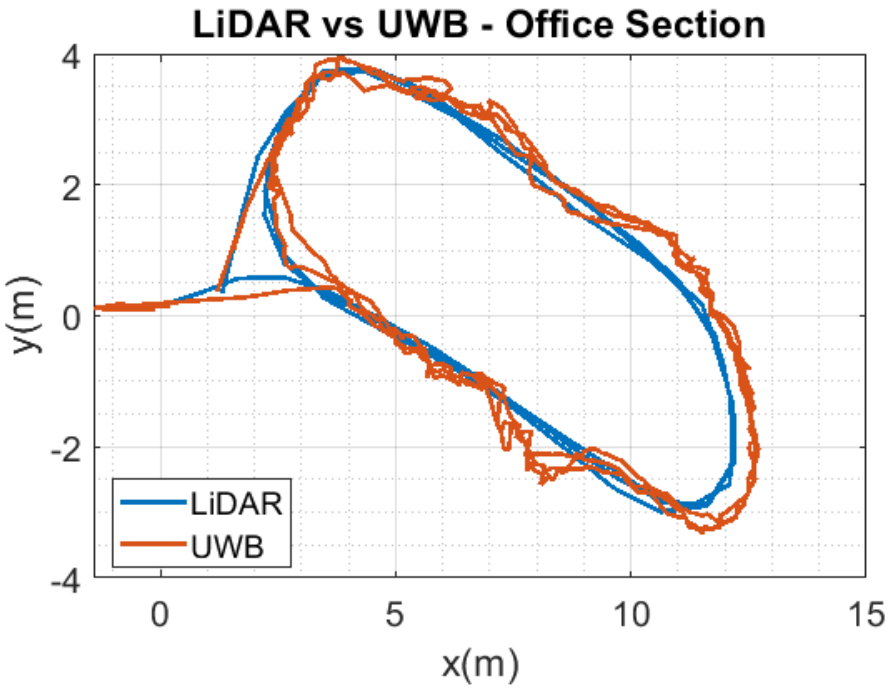


Figure 3.12 UWB vs. LIDAR- Indoor Segment – Office.

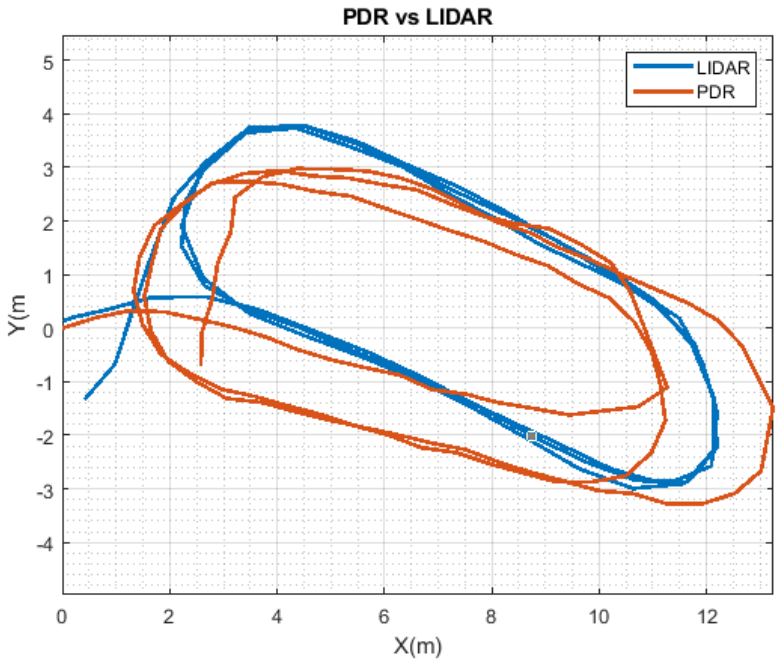


Figure 3.13 PDR vs. LIDAR- Indoor Segment - Office.

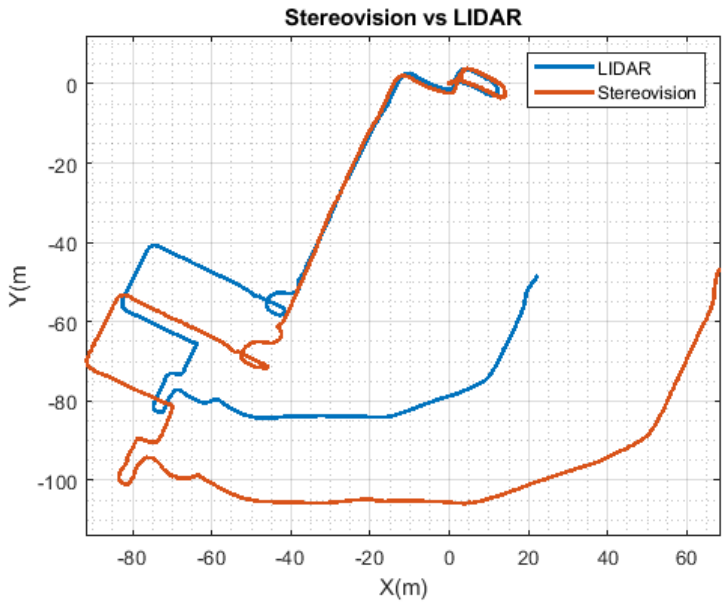


Figure 3.14 Stereovision vs LIDAR– Indoor/Outdoor section.

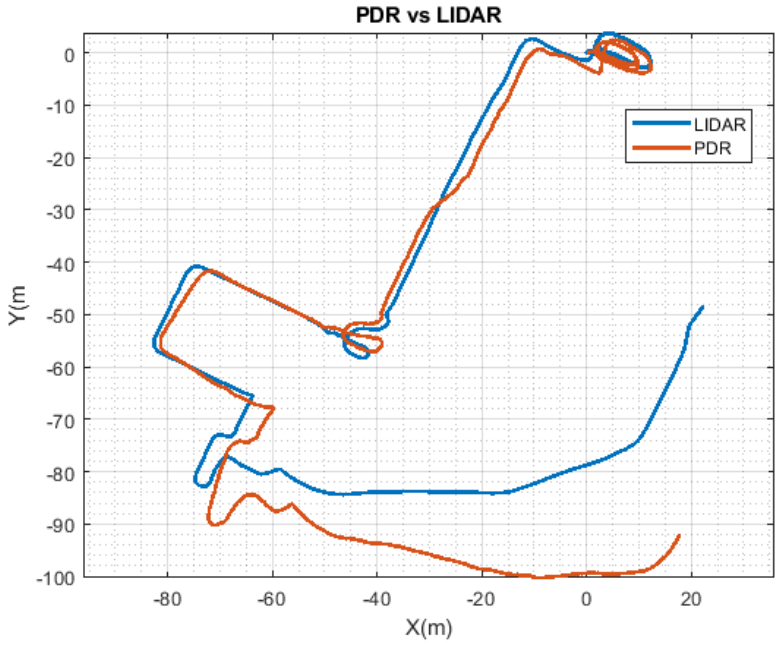


Figure 3.15 PDR vs LIDAR– Indoor/Outdoor section.

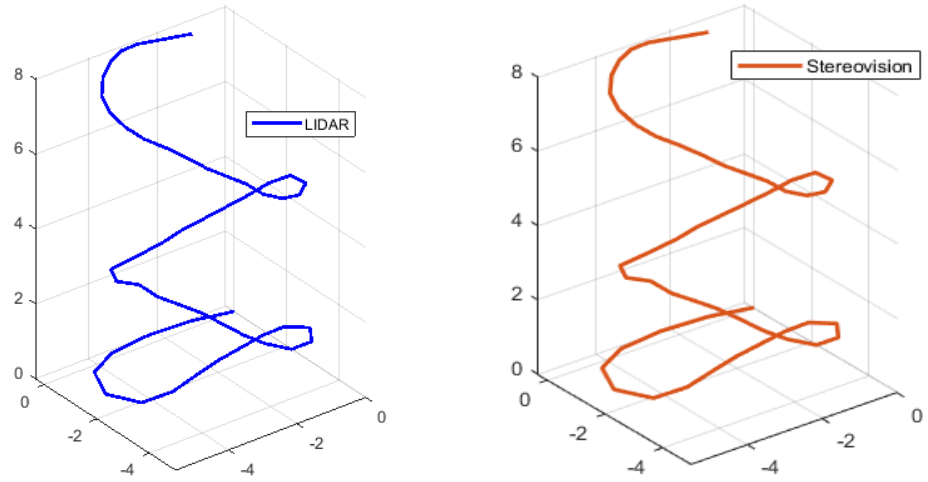


Figure 3.16 Experiment Path Segment - Stairs section.

3.5 Summary

This chapter discussed the design and implementation details of multisensor indoor pedestrian navigation using body-mounted sensors. The hardware and software architecture of the system was described in detail. The presented system adopts distributed multi-embedded platform architecture connected wirelessly using a high-speed portable wireless local network. The software architecture utilizes a client-server scheme using the Robot Operating System (ROS) framework. The system incorporated state-of-art tracking sensors and technologies, including stereovision, UWB wireless ranging and positioning, LiDAR and IMU. UWB showed decimeter-level accuracy when high-elevation anchors properly covered the area. Stereovision showed meter-level accuracy provided that rich features with proximity suitable for the stereo baseline are available. PDR showed reliable performance for approximately seven minutes before starting to drift. The concluding remark is that accurate long-term pedestrian tracking in complex GNSS-denied environments is feasible by combining these different technologies in one system. The experiment was performed in a complex environment incorporated indoors, outdoors, and

stairs sections with several loops. Results and analysis were presented, and a link to the experiment video was provided. The collected dataset was publicly released to the research community upon the publication of this work.

Chapter 4: Hybrid IMU/UWB-Aided Visual Odometry

This chapter presents two approaches to aid visual odometry solutions using sensor fusion. The first approach is a hybrid IMU-aided visual odometry, which aims to increase the system's robustness and frame rate while maintaining the solution's accuracy. To further increase the accuracy of visual-inertial odometry solutions, we introduce another solution which integrates a UWB radio positioning network to limit the trajectory estimation drift. The novelties presented in the chapter could be summarized as follows:

- Present a novel approach that enables switching between stereo and monocular camera odometry. The monocular visual odometry utilizes a motion scale estimation from an INS solution.
- Provide an implementation of a loosely coupled EKF for camera-IMU integration benefits from the motion constraints as non-holonomic constraints.
- Enhance the accuracy of the implemented visual-inertial odometry system using external updates from a UWB network.

4.1 Hybrid IMU-Aided Approach for Optimized Visual Odometry

Visual Odometry (VO) solutions can be classified into two main groups [4]; monocular and stereo. In monocular visual odometry algorithms [104], a single camera feed is used to estimate the new camera position and orientation. The weakness of this class of solutions is that a single camera can only estimate the translation and rotation between two image frames up to an unknown scale. This scale is necessary to transform the estimated camera motion into real-world physical motion. Consequently, monocular visual odometry is commonly integrated with other sensors like IMU [105]. This integration aims

to estimate this unknown scale and provide real-world positioning. However, uncalibrated IMU with large stochastic noise and errors accurately estimates these unknown scale challenges.

In stereo visual odometry algorithms [106], the visual feature's physical depth in the real world is directly measured by triangulation using images from two or more cameras and then constructing the camera motion in the physical, real-world domain. Although stereo vision does not suffer from scale ambiguity, stereo vision processing is more computationally expensive. It processes two input video streams that consume larger memory and power, leading to a shorter battery lifetime and limited operating range. Moreover, either approach loses track of the visual features with high dynamics and poor illumination.

Even with stereo cameras on board, having a monocular vision-based navigation solution is more suitable for small autonomous vehicles' restrictions imposed by the characteristics. The smaller the vehicle, the less the processing power available and the shorter the baseline distance between cameras which causes degeneration to a monocular mode whenever the visual features are at a distance that is significantly larger than the baseline between stereo cameras. Therefore, having a dynamic switch between a monocular and stereo mode could greatly benefit autonomous vehicle navigation. This work integrates visual odometry with INS using Extended Kalman Filter (EKF), which is widely used for its simple implementation and suitability for real-time applications [107], [108].

The proposed solution addresses visual-inertial odometry's accuracy and computation challenges by proposing a hybrid and adaptive fusion scheme that minimizes

the processing requirements by dividing the navigation mission into two interchangeable phases. Firstly, A stereo vision-based navigation phase is performed in which a loosely coupled integration between the camera and IMU solution. The vision solution is used to update the EKF to estimate the INS errors. The second phase is the monocular phase. We downgrade our vision system into only one camera-based system with minimum computations and get the motion scale from the calibrated INS solution. Results showed that the proposed hybrid adaptive fusion scheme could provide accuracy comparable to full stereo vision tracking with the advantage of fewer computations and less tendency to lose track.

4.1.1 Extended Kalman Filter (EKF)

EKF is an extension of the Kalman Filter (KF) that has been proposed for state estimation in nonlinear dynamic systems [96]. The EKF linearizes a nonlinear system about its current estimate of the state's mean. Consequently, the system is represented linearly in terms of the state error, allowing the Kalman filter to be applied to the linearized system. Even though the optimal mean-square error is not guaranteed in the EKF case, EKF provides an accurate approximation of it in many applications [109]. The definition of a nonlinear dynamic system [75] is given by:

$$\dot{x}(t) = f\big(x(t), u(t), w(t)\big), \tag{4.1}$$

$$y(t) = h\big(x(t)\big) + v(t), \tag{4.2}$$

Where $u(t) \in \mathbb{R}^m$ is the m-dimension control input, $x(t) \in \mathbb{R}^n$ is the n-dimension state vector and $w(t)$ is the process noise model. $f(.)$ is a nonlinear function defining the systems' prediction model. $y(t) \in \mathbb{R}^d$ represents the d-dimension measurement vector. $h(.)$ is the measurement model, while $v(t) \in \mathbb{R}^d$ is the model of the measurements

additive noise. By expanding equations (4.1) and (4.2) with Taylor series approximation, a linear system in terms of states' error can be obtained as follows:

$$\delta\dot{x}(t) = F(t)\delta x(t) + G(t)w(t), \tag{4.3}$$

$$\delta y(t) = H(t)\delta x(t) + v(t), \tag{4.4}$$

where

$$F_{n\times n}(t) = \frac{\delta f(x(t), u(t), w(t))}{\delta x}, \tag{4.5}$$

$$G_{n\times n}(t) = \frac{\delta f(x(t), u(t), w(t))}{\delta w}, \tag{4.6}$$

$$H_{n\times n}(t) = \frac{\delta h(x(t))}{\delta x}, \tag{4.7}$$

where F, G, and H matrices are the Jacobians of the system, input, and measurement models, respectively. System states error, $\delta x = \tilde{x} - x$, is the difference between the actual state value and the estimated value of the state vector. Similar to KF, the system noises (process and measurement) are considered Gaussian noise with zero-mean. The process and measurement covariance matrices $Q(t) \in \mathbb{R}^{n\times n}$ and $R(t) \in \mathbb{R}^{d\times d}$ are defined as follows:

$$Q(t) = \langle w(t)w^T(t)\rangle, \tag{4.8}$$

$$R(t) = \langle v(t)v^T(t)\rangle, \tag{4.9}$$

where $\langle\,\rangle$ is the mathematical exception operator. The covariance of the state error estimation can be calculated as:

$$P(t) = \langle \delta x(t)\delta x^T(t)\rangle. \tag{4. 10}$$

Equations (4.3) and (4.4) in discrete space are redefined as follows:

$$\delta x_k = (I + F_k T_s)\delta x_{k-1} + G_k w_{k-1}, \tag{4.11}$$

$$\delta y_k = \mathrm{H}_k \delta x_k + v_k, \tag{4.12}$$

where T_s is the sampling period of the discretization, and k denotes the discrete time step. Finally, with the measurement y_k, the EKF process can be described as two recursive prediction and update steps defined as follows:

1- Prediction Step:

   a. Calculate the full state $\tilde{x}$ which is the operating state that we linearize the system around.

$$x_{k+1} = f(x_k, u_k), \tag{4.13}$$

   b. Predict the error state covariance.

$$P_{k+1} = (I + F_{k+1}T_s)P_k(I + F_{k+1}T_s)^T + G_{k+1}QG_{k+1}^T T_s^2, \tag{4.14}$$

2- Update Step: whenever an update y_{k+1} is available calculate the following:

   a. Update error state covariance.

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1}, \tag{4.15}$$

   b. Calculate the Kalman gain.

$$K_{k+1} = P_{k+1}H_{k+1}^T(H_{k+1}P_{k+1}H_{k+1}^T + R_{k+1})^{-1}, \tag{4.16}$$

   c. Correct the states.

$$x_{k+1} = x_{k+1} + K_{k+1}[y_{k+1} - h(x_{k+1})], \tag{4.17}$$

Equations (4.13) and (4.14) propagate the state and covariance to the following time step. Once a new measurement y_{k+1} is obtained, we update the state covariance in equation (4.15), then the Kalman gain, K is calculated from equation (4.16), and then the state is corrected in equation (4.17). The error state covariance P_k is a positive definite and symmetric matrix, it is reported that update step in equation (4.15) violates the

characteristics of P_k due to the smallest error in the estimation of the Kalman gain K. To ensure the positive semi-definiteness of P_k the following equation is used:

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}{K_{k+1}}^T, \quad (4.18)$$

To maintain the semi-posiiveness of the covariance matrix P_k, it can be normalized after each processing epoch. For the complete details of EKF estimation equations, refer to [75].

4.1.2 System Description

We propose a loosely coupled visual-inertial odometry (VIO) system. The proposed fusion scheme benefits from the stereo vision-based system's accuracy and scale-free positioning estimate by updating a full 3D IMU positioning solution within an EKF module that considers the vehicle's non-holonomic constraints. Stereo mode is activated in the beginning until a reliable, accurate estimate of the IMU errors is obtained, and stable EKF convergence is achieved. Then, the system switches to IMU-aided monocular mode benefitting from its lower memory and computation requirements. The motion scale generated by the IMU is tested every few seconds. When the error in the motion scale goes beyond a certain threshold, the system switches back to the stereo mode and feeds its estimation to the EKF to re-estimate IMU biases.

The implemented EKF uses a full 3D IMU dynamic model [96] to generate its predicted trajectory estimation of the moving platform's position, velocity, and orientation at a rate of 100Hz. At a lower rate (15 Hz), a stereo vision system that processes images sequence from a couple of cameras is developed. The developed stereo vision uses a SOFT SLAM library [110]. While using the SOFT estimation as a position update, EKF convergence is watched. Once converged, the system automatically switches to the monocular mode using the calibrated IMU position prediction, aided by non-holonomic

constraints. This way, the system can estimate the motion scale leading to a full 3D monocular visual-inertial solution scale-free. Switching to monocular mode frees extra resources for other tasks like motion planning or mapping. The Flowchart of the proposed VIO system is shown in Figure 4.1.

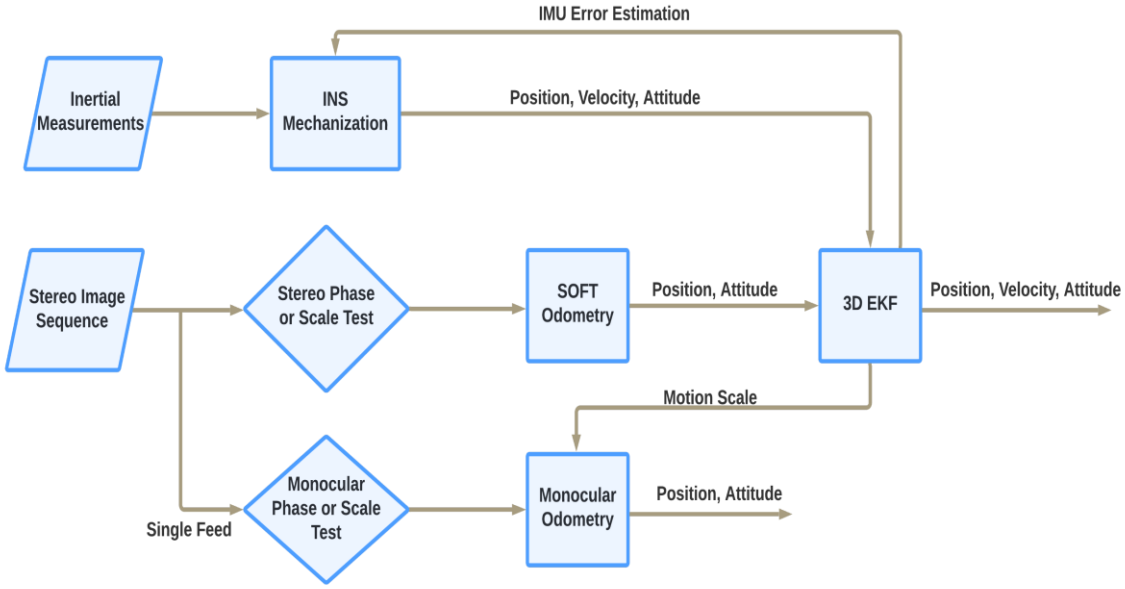


Figure 4.1 Flowchart of the proposed VIO system.

Referring to equations (4.1) and (4.2), the EKF nonlinear dynamic system *f(.)* is given by the IMU dynamic system model provided in section 2.2.1. The stereo vision-based position updates provide the measurements incorporated into the system by the measurement model h(.). In our loosely coupled system, given our state vector $x(t) = [P^L \quad q_B^L \quad v^L \quad b_g \quad b_a]$, the design matrix H relates the visual states' updates to the system states as follows:

$$\delta y = \begin{bmatrix} P_{vision} - P_{INS} \\ R_{vision} - R_{INS} \\ v_{vision} - v_{INS} \end{bmatrix} = H\delta x(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta P^L \\ \delta q_B^L \\ \delta v^L \\ \delta b_g \\ \delta b_a \end{bmatrix}, \qquad (4.19)$$

Where δy is the difference between the VO estimation and the INS prediction states, $P^L, q_B^L \; and \; v^L$ are the system position, rotation and velocity, respectively and b_g and b_a are the IMU estimated biases. The process noise covariance matrix Q and the measurement noise covariance matrix R were tuned to ensure the robustness and accuracy of the EKF fusion.

To sustain the system's accuracy, the following mechanism could be implemented. Every few seconds, we switch on the stereo SOFT odometry solution and test the EKF estimated motion scale against the SOFT odometry solution for one iteration. Suppose there is a large discrepancy between the two estimations. In that case, the system automatically switches off the monocular estimation and feeds the SOFT odometry updates to the EKF to re-converge and re-estimate the IMU biases. The system also can rely on the IMU readings to adjust for high dynamics. This adjustment could be made by switching from monocular mode to stereo mode when the sensed acceleration and rotation rates are within a certain low limit. Additionally, switching from stereo to monocular in case of high-speed motions or fast rotations. These fast motions require processing over 25 fps to estimate the system's trajectory. The remainder of this section is devoted to describing both SOFT stereo VO and the monocular VO. The proposed solution pseudo-code is as follows:

Algorithm: Hybrid IMU-Aided Approach for Optimized Visual Odometry

Inputs:

Stereo pair: $\{\boldsymbol{I_l}, \boldsymbol{I_r}\}$ // left and right images

IMU measurements: $\{\boldsymbol{a_x}, \boldsymbol{a_y}, \boldsymbol{a_z}, \boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}\}$ // IMU raw data

Output:

Platform Pose: $\{\boldsymbol{P_x}, \boldsymbol{P_y}, \boldsymbol{P_z}, \boldsymbol{R_x}, \boldsymbol{R_y}, \boldsymbol{R_z}\}$ // The estimated 3D pose

```
set_scale_error_limit(scale_error_limt)                  // set allowed scale error
set_test_count_limit (test_count_limit)                  //define scale test frequency
IMU_biases_estimated=false                               // IMU not calibrated
Count=0

while "new input data available" do                      // new data frame
    if "IMU_biases_estimated is true" then               // if IMU is calibrated
        scale = INS_Scale_prediction(a_x, a_y, a_z, ω_x, ω_y, ω_z)   // get scale prediction
        {P_x, P_y, P_z, R_x, R_y, R_z}=Monocular_VO(I_l,scale)       // run monocular VO
        count+=1
        if count >= test_count_limit then                // if test is needed
            scale_error = perform_scale_test(scale)      // perform scale test
            if scale_error > scale_error_limt then       // IMU not calibrated
                IMU_biases_estimated=false
                count=0                                  // reset counter
            end
        end
    else                                                 // if IMU is not calibrated
        {P_x, P_y, P_z, R_x, R_y, R_z}=SOFT_VO(I_l, I_r)  // run Stereo VO
        estimate_INS_biases(P_x, P_y, P_z, R_x, R_y, R_z)  // calibrate IMU
        if "INS_biases_converged" then
            IMU_biases_estimated=true                    // IMU calibration is done
        end
    end
end
```

SOFT Visual Odometry System:

We used the SOFT odometry method for the stereo visual odometry stage [53], which ranks as one of the highest accuracy visual odometry solutions on the KITTI vision benchmark suite [93]. The name SOFT stands for Stereo Odometry algorithm relying on Feature Tracking [111]. As can be seen in the proposed solution in Figure 4.2, the estimated change of position and orientation estimated from the SOFT is fed to the EKF as an update to estimate the IMU biases and errors optimally. The main drive that motivated every decision to design the SOFT odometry algorithm is to enable real-time performance and optimize the processing requirements. The algorithm comprises four primary stages, as shown in Figure 4.3. The first stage is the feature detection and selection in which the idea of bucketing is applied. Bucketing technique divides the image into 50x50 pixel-sized buckets. Figure 4.2 shows visual features and visual tracking solution sample.

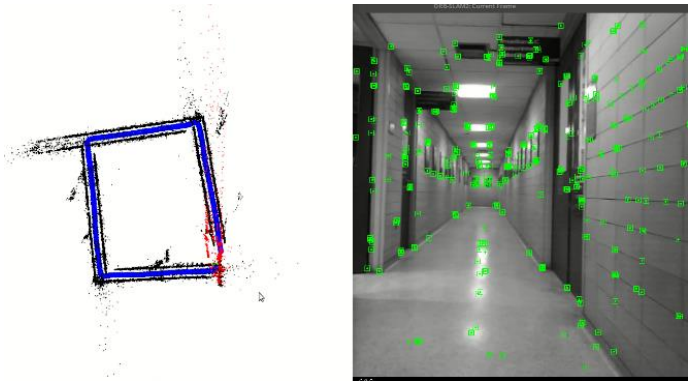


Figure 4.2 Sample of Visual Features and visual tracking solution

In every bucket, corner and blob masks are applied, followed by a non-maximum suppression step to have four class lists of carefully selected strong features. Then, these features go through a circular matching process in which the features are tested if they are matched in two consecutive frames. Only the features that pass this test are taken into

account. Using this small subset of the image features reduces the computational time for all the following stages.

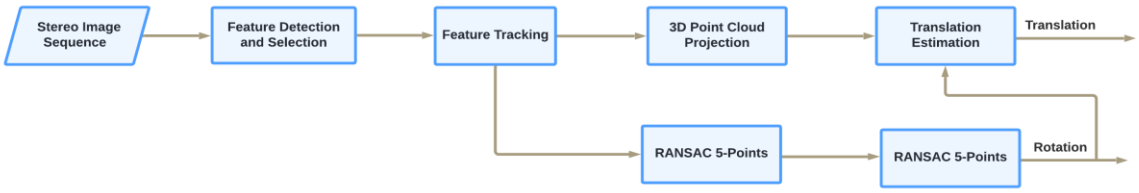


Figure 4.3 SOFT Visual odometry flowchart.

The second stage is the feature tracking stage. Each feature point has the following associated properties:

1. Unique identifier.
2. Age.
3. Refined current position in the image.
4. Feature strength.
5. Belonging class.
6. Initial descriptor.

A unique identifier is assigned for it with the feature's first appearance, and the age is initialized with zero. In each iteration, the position of the feature point is refined on a subpixel level with respect to the initial feature position; using the same descriptor during the entire feature lifetime reduces the drift. The third stage is the rotation estimation. In this stage, the feed from the left camera is used as an input to the RANSAC 5 points module [110], [4] to estimate the rotation matrix and then apply the spherical linear interpolation (SLERP) step that refines the rotation estimation. Finally, the translation estimation stage is implemented. Having all the correspondences determined, the 3D point cloud estimation

of the previous view is projected into the image plane of the current view. Then the translation is calculated iteratively by minimizing a cost function.

Monocular Odometry System:

In monocular mode, we implemented the basic components of the visual odometry algorithm as described in [112], [4]; the monocular solution flowchart is shown in Figure 4.4. The monocular mode solution satisfies the most basic implementation of the visual odometry solution. We applied the ORB feature detector [113], and then we tracked the features using Kanade-Lucas-Tomasi [114] sparse optical flow technique.

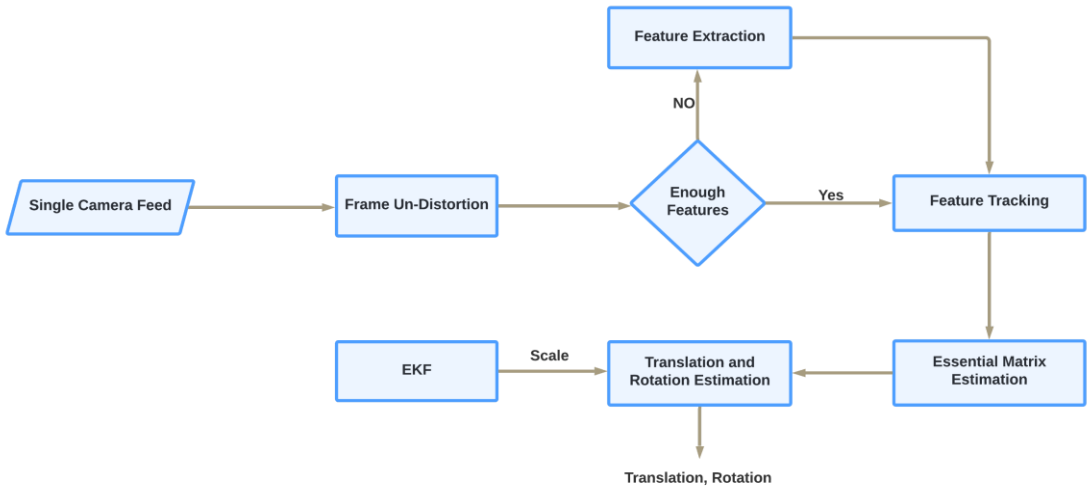


Figure 4.4 Proposed Monocular VO.

The choice of optical flow came from its ability to track features between frames without the need to detect the features with each coming frame. A threshold of the minimum features was tuned to trigger a new feature detection search. The sparse technique was selected because it is faster than the dense optical flow. Once we have point correspondences, we used RANSAC 5-points [115], a standard technique to handle outliers' existence and remove them and then compute the essential matrix using the

Epipolar geometry constraints. The epipolar constraint between two views of a calibrated camera can be expressed as:

$$I'EI = 0, \tag{4. 20}$$

Where I and I' are the corresponding features image coordinates between two views, and E is the essential matrix. Since the scale in monocular vision is unobservable, we get the motion scale from the IMU prediction model. Having the essential matrix and the motion scale, we estimate the rotation R and translation T. Finally. We construct the final position T_f and rotation matrix R_f:

$$R_f = R\, R_f\,, \tag{4. 21}$$

$$T_f = T_f + S\, R_f T, \tag{4. 22}$$

S is the motion scale. In the proposed system, the motion scale S is calculated from the calibrated IMU compensated for biases estimated by the EKF during the stereo mode phase.

4.1.3 Experimental Work and Results

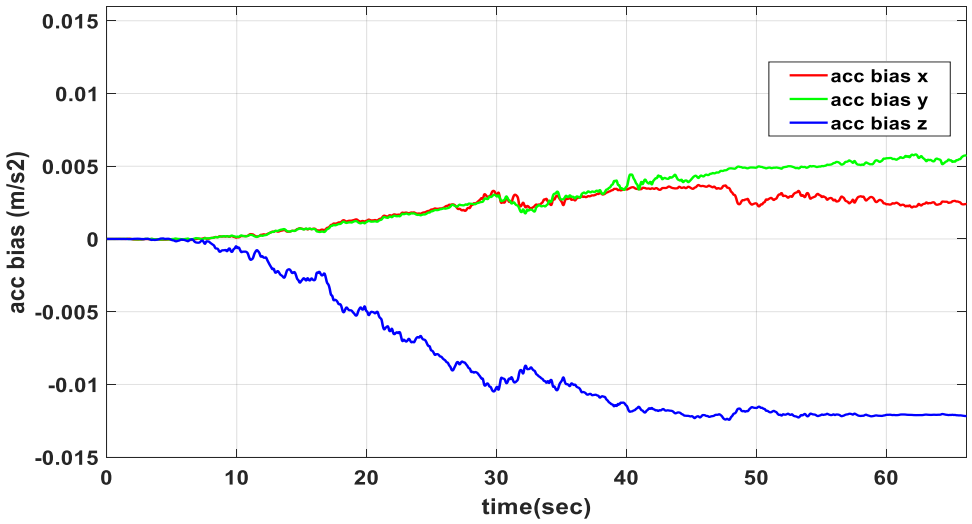


Figure 4.5 Accelerometer estimated biases.

The experiments were performed on the 30mx30m 5th floor of the Mackenzie Building at Carleton University in Ottawa. We measured the final position loop error, which is the positional difference between the starting and ending coordinate when we close the loop in the testing area. In the beginning, the system enabled the stereo mode by default. Figure 4.5 and Figure 4.6 show that the EKF needed less than 50 seconds to converge to the optimum estimate of the accelerometers and gyroscopes biases.

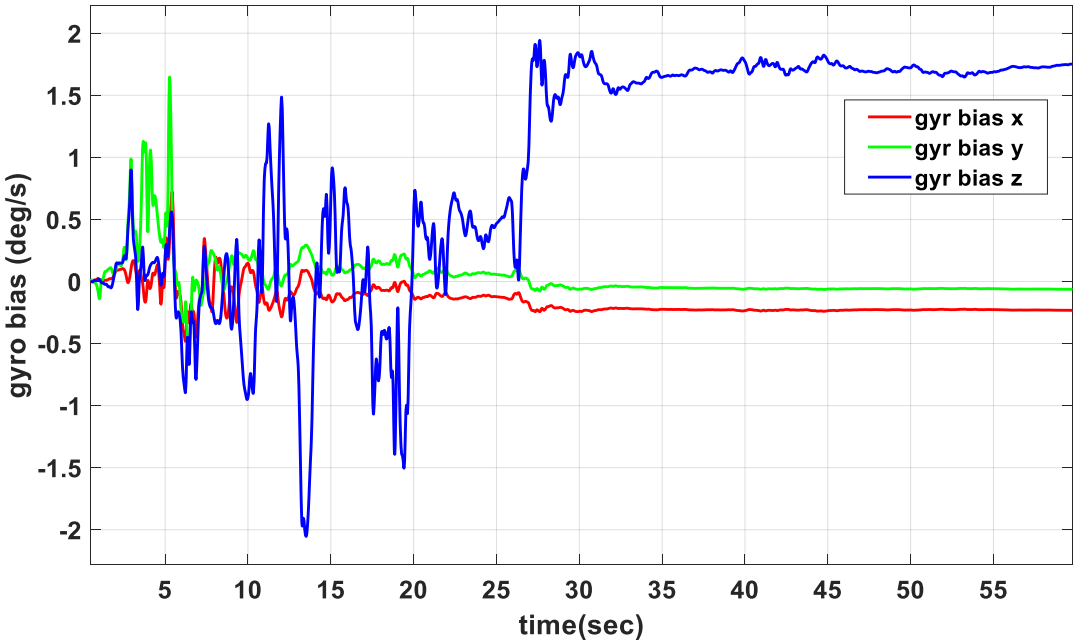


Figure 4.6 Gyroscopes estimated biases.

A comparison between the estimated trajectory of the ZED mini (ZED), RealSense (RS) and the SOFT odometry is illustrated in Figure 4.7. The presented figure suggests that SOFT results are comparable to the RS position estimation in our test conditions while outperforming the ZED Mini position estimation with less than 20 cm end-of-loop position error after a complete loop (23.5mX18.5m) which took approximately 90 seconds. Using the SOFT estimated trajectory as a position update to the EKF for 50 seconds, the EKF has converged, as shown in Figure 4.5 and Figure 4.6. An accurate estimation of the IMU

biases has been obtained. Then the system automatically switched to the monocular mode visual odometry solution and continued for 40 seconds, resulting in the blue trajectory illustrated in Figure 4.8. The red trajectory in Figure 4.7 is the full SOFT stereo solution with a 20 cm end-of-loop position error.

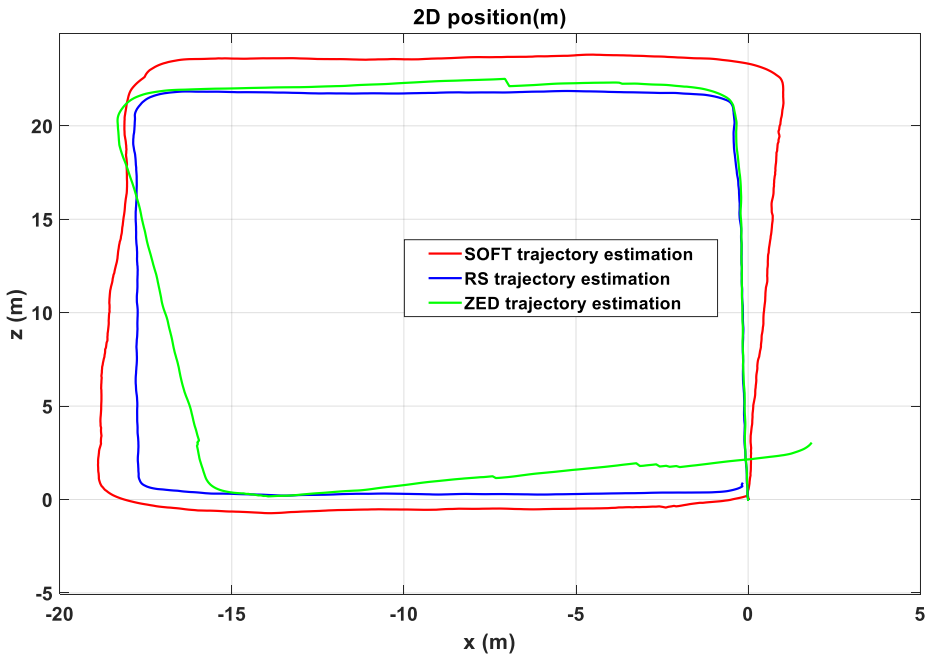


Figure 4.7 SOFT, Realsense and ZED mini estimated trajectories.

Five accuracy tests were performed on the IMU-generated scale during the experiment to verify its consistency with that estimated scale from stereo vision. No discrepancy has been detected as the error was smaller than the selected threshold, and the overall error at the end of the tested loop was 0.5 meters. Additionally, the output frame rate is compared to judge the proposed system's complexity. While The SOFT stereo solution operated at 10 fps and the monocular solution operated at 30 fps, the proposed hybrid solution worked at 19 fps. Although the calibrated IMU-aided monocular vision solution is slightly less accurate than the SOFT stereo vision solution, given the huge

saving in computation by switching to the monocular solution instead of a stereo solution, the advantage of the proposed hybrid scheme is clear.

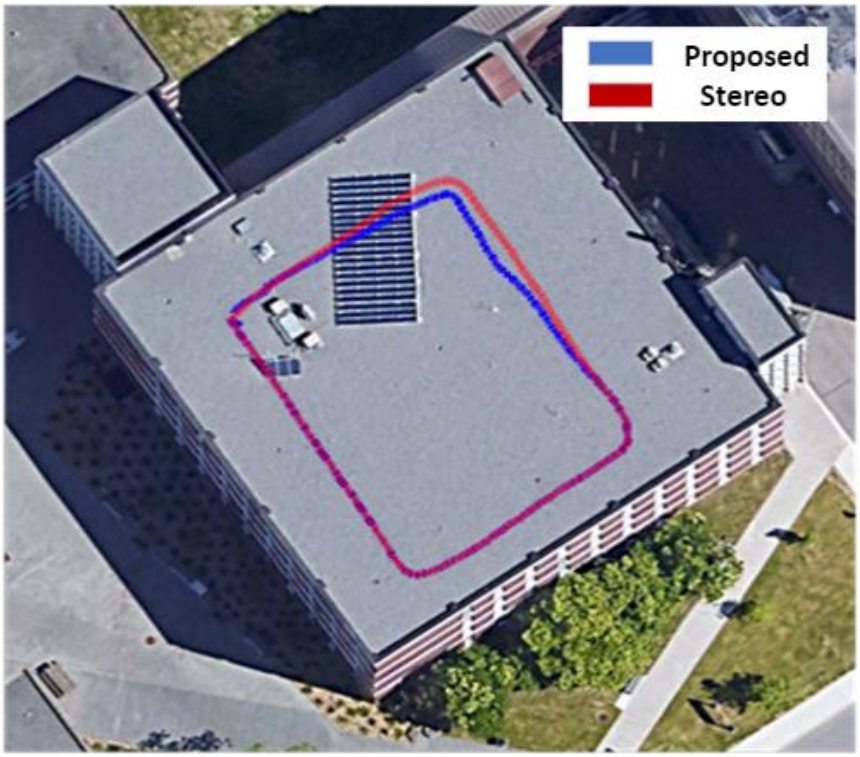


Figure 4.8 Stereo (Red) vs. IMU-aided monocular vision (Blue) solutions.

4.2 Stereo Visual odometry aided by IMU and UWB Sensor Fusion

To further enhance the visual-inertial odometry accuracy [116], ranging technologies such as UWB have been introduced as an indoor replacement for GNSS. However, UWB has some drawbacks, including reflections, multi-path and Non-line-of-sight (NLOS) conditions [88]. This section presents a robust fusion framework based on (EKF) [117] to accurately and efficiently fuse these sensors. The proposed design applies non-holonomic speed-aided motion constraints to minimize positional drifts in lateral and vertical directions induced by IMU sensors' biases integration [118], [119]. A non-holonomic constraint is a condition that is derived from the characteristics of the motion

prediction model or the measurement model, which could reduce the system's degrees of freedom.

Using non-holonomic constraints can decrease the system's complexity and increase the solution's accuracy. In contrast to some existing fusion schemes, such as those in [120], the non-holonomic motion constraints enabled the fusion of speed information obtained from processing stereo vision sensors. Using vision-estimated single-dimensional forward speed instead of a 3D position has shown more robustness against visual positioning errors. To further control positional error drifts, whenever reliable, UWB ranging and positioning have been used as an update to the EKF. The developed system has been demonstrated in a real indoor environment using a stereo-camera sensor, a built-in IMU sensor, and a local wireless infrastructure of the UWB radio network.

4.2.1 System Description

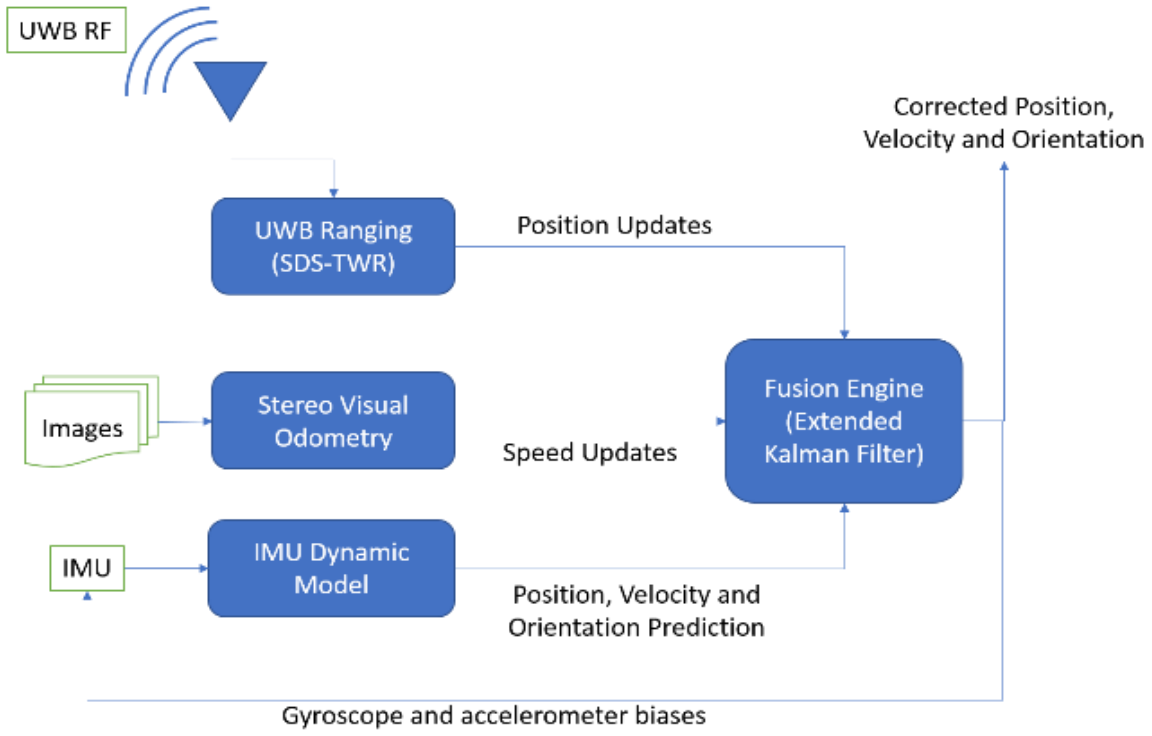


Figure 4.9 UWB-Vision-IMU System Block diagram.

In previous work [89], [88], conventional EKF designs and updates are taken as absolute positions calculated by stereo vision [121], [120]. However, this technique is not robust against visual tracking errors. The rigid body's motion is mainly concentrated in the forward direction (x-axis in *B* frame). Using this fact as a non-holonomic constraint could greatly enhance the trajectory accuracy. Our proposed system estimates the platform speed from visual tracking and then projects it into the platform orientation. The speed value is a single value that is more immune to orientation errors introduced by the stereo vision projection. The proposed system overall block diagram is shown in Figure 4.9. In the presented loosely coupled fusion, given our state vector $x(t) = \begin{bmatrix} P^L & q_B^L & v^L & b_g & b_a \end{bmatrix}$. The design matrix H relates the UWB position updates to the system states as follows:

$$\delta y = [P_{UWB} - P_{VIO}] = H\delta x(t) = [1 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} \delta P^L \\ \delta q_B^L \\ \delta v^L \\ \delta b_g \\ \delta b_a \end{bmatrix}, \qquad (4.23)$$

Where δy is the difference between the UWB estimation and the VIO prediction states, $P^L, q_B^L \; and \; v^L$ are the system position, rotation and velocity, respectively and b_g and b_a are the IMU estimated biases.

4.2.2 Experimental Work and Results

The experiments were performed in a 30mx30m area on the 5th floor of the Mackenzie Building at Carleton University in Ottawa (see Figure 4.11). We installed 11 battery-enabled UWB nodes as anchors in known locations and programmed one node as a tag. Figure 4.10 shows the UWB estimated trajectory against the Realsense T265 solution. The UWB solution did not accumulate drifts with time; however, it is very noisy compared to the RS solution. Figure 4.11 compares the visual tracking solution (red), the

EKF fusion (yellow) and the ground truth (blue). To avoid confusion, we did not display the UWB solution on the same 2D plot. However, Figure 4.12 shows all solutions in terms of cumulative error percentage.

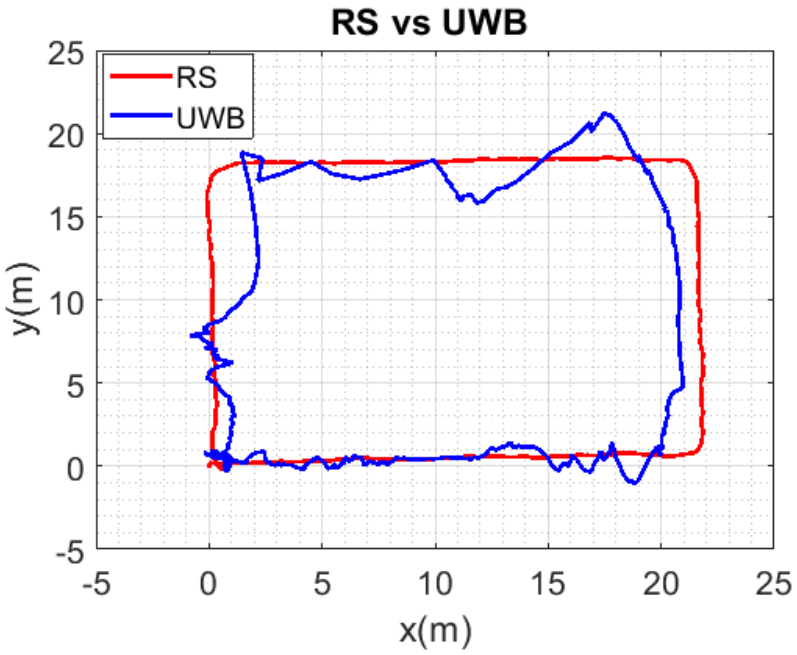


Figure 4.10 UWB vs RealSense T265 solution.

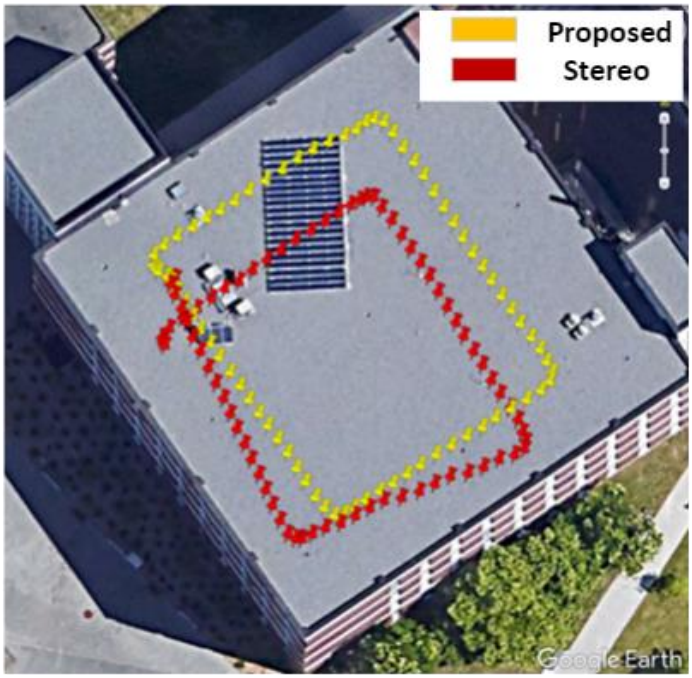


Figure 4.11 Stereo vision solution (Red) vs. Proposed Fusion Solution (Yello).

To test the developed EKF against outages (failures of UWB anchors or the presence of an insufficient number of anchors), we simulated five UWB outages for 10 seconds each at different locations and computed the EKF solution error. As seen in Figure 4.13, the error was in sub-meter accuracy (maximum is 57 cm), which shows the robustness and accuracy of the developed EKF fusion framework.

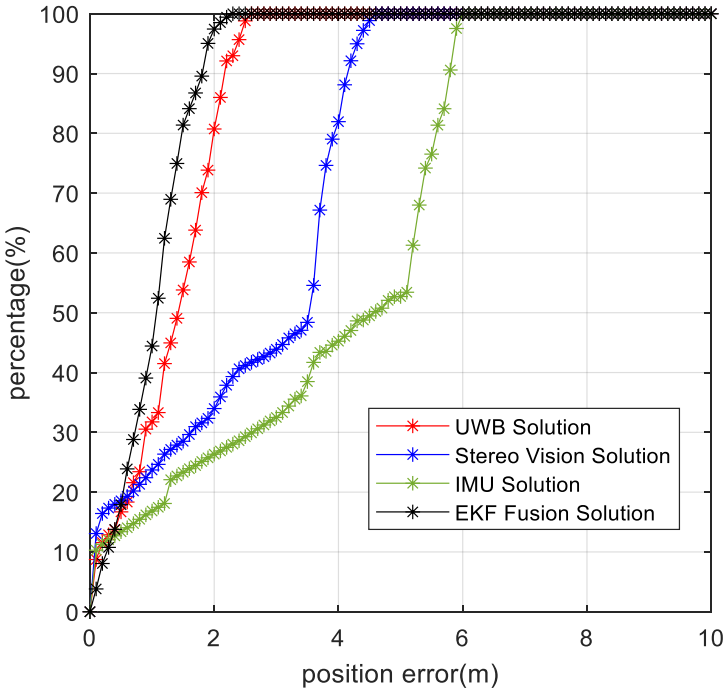


Figure 4.12 Cumulative Error Percentage(m)

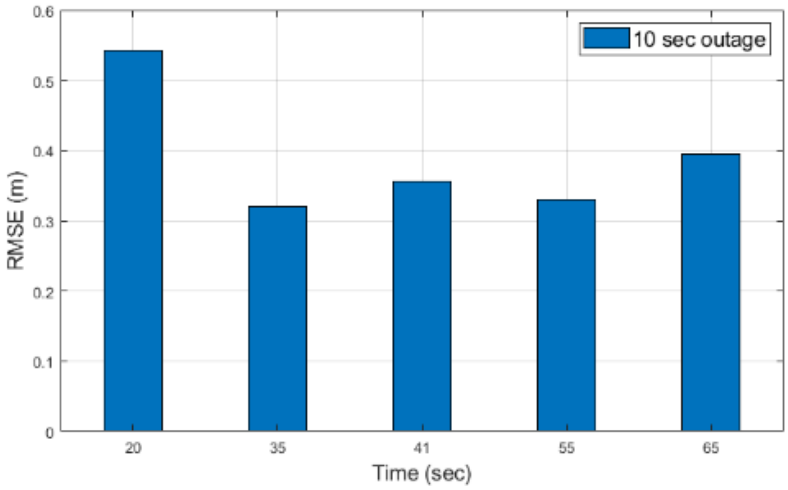


Figure 4.13 Positional Errors during Outages.

4.3 Summary

This chapter presented two visual odometry systems that use sensor fusion to increase their robustness, speed and accuracy. First, a hybrid IMU-aided visual odometry solution fuses both visual and inertial sensors in a hybrid and adaptive scheme. The solution uses a high-precision visual odometry algorithm (SOFT) to update the EKF to estimate IMU biases. Then the system minimizes the processing power requirements by switching to a monocular visual odometry mode, which frees resources for the additional tasks. The presented solution could help navigate unknown environments, and the obtained accuracy encourages an embedded implementation targeting small autonomous battery-operated vehicles. For 50 seconds update outages, the overall position estimation error was less than 0.5m.

Second, the chapter illustrated a robust fusion framework for indoor navigation using IMU, stereo vision, and UWB. The developed EKF benefits from non-holonomic constraints limiting drifts and resisting visual tracking orientation errors. Experimental results revealed that the proposed solution outperformed individual sensors having a

position error of 84cm (root mean square). We further showed that our system achieved an average accuracy of 30cm under outage of 10 seconds, even under temporal outages.

Chapter 5: Improved Visual SLAM Using Semantic Segmentation and Layout Estimation

The technological advances in computational systems enabled very complex computer vision and machine learning approaches to perform efficiently and accurately. These new approaches can be considered a new set of tools to reshape the visual SLAM solutions. We present an investigation of the latest neuroscientific research that explains how the human brain can accurately navigate and map unknown environments. The accuracy suggests that human navigation is not affected by traditional visual odometry drifts resulting from tracking visual features. It utilizes the geometrical structures of the surrounding objects within the navigated space. The identified objects and space geometrical shapes anchor the estimated space representation and mitigate the overall drift. This chapter presents our efforts to incorporate two machine learning techniques into the VSLAM solution: semantic segmentation and layout estimation to imitate human abilities to map new environments.

5.1 Background

As a human or an animal navigates through an unfamiliar environment, some form of spatial memory is formed. This memory creates a cognitive map representing a spatial environment and contains information about metric and directional relationships of objects in that environment. Humans are highly visual creatures [122], and under typical situations, vision forms a critical foundation for representing space [123]. While extensive research on navigational behaviour and spatial memory has been carried out in rodents, memory research in humans has traditionally focused on more abstract, language-based tasks [122]. Recent studies have begun to address the gap in human navigation research benefiting from

the Virtual Reality evolution that enabled virtual-navigation simulations combined with human electrophysiological recordings [124]. These studies suggest that the medial temporal lobe [5] (MTL) is equipped with a population of the place and grid cells similar to that previously observed in the rodent brain. We believe that recent neuroscience discoveries related to cognitive memory could revolutionize VSLAM solutions. In this section, we discuss the neuroscientific answers to the following questions:

- Does the Human brain perform SLAM?
- How does the human brain solve navigation tasks?
- What landmarks and semantic information could form human brain spatial maps?

5.1.1 Brain spatial mapping process

John O'Keefe, Moser and Edvard were awarded the Nobel Prize for Medicine in 2014. They won the prize for discovering two types of cells in the hippocampus: place cells and grid cells. These findings significantly impacted cognitive neuroscience, particularly spatial cognition. Further research found other cell types in the hippocampus, boundary vector cells, and head direction cells [125]–[127].

Place cells are activated when an animal is in a specific position in an environment, regardless of the aspects of the job at hand [128]; they include data about a specific area. Grid cells fire at regular intervals and represent the layout of an environment. When an animal is near a given edge of an environment, boundary vector cells show activity, providing additional information about the animal's relative position in its surroundings. The head direction cells signal the animal's head orientation. Combining these cells that code for position, layout, borders and head orientation creates a mental map of the

environment in a useful state for navigation. This mental map depicts the spatial organization of an environment without regard to a particular point of view.

Finally, it is essential to acknowledge the brain motor sensor that provides the brain with an initial guess of the body motion that helps the brain control the eye movements to prevent disorientation and always anticipate where surrounding objects should be [5], [129], [130]. Also, our brain optimizes memory and calculations needed to perform its navigation tasks by abstracting the environment to the space boundaries and objects referenced to these boundaries [131]–[133].

5.1.2 Human Navigation Strategies

We use three basic ways to plan our trajectory to reach our goal: allocentric, egocentric, and beacon [134].

The allocentric or spatial memory strategy is a navigational strategy that involves intricate geometric calculations [135]. These calculations take distance and directional information into account [136]. An environmental representation (map) is formed and referenced outside of one's current body position [137], [138]. The navigator explores the environment by establishing relationships between various landmarks and orienting oneself with respect to those landmarks [139]. Creating external maps is part of human navigation skills [140]–[142]. Still, they are helped by a variety of more complicated cognitive activities that extract abstract semantic information from landmarks [143], routes taken, and the environment's architecture [144]–[147].

Recent work in human spatial navigation has shown that the environmental boundaries provide a way to establish an allocentric coordinate system [137], [148]. The surrounding spatial geometry, such as the square or rectangle shape defined by an

environment's boundaries, can be a powerful cue for organizing externally referenced knowledge. Several investigations [149]–[152] have reported that aligning the objects with the environment axis increases the accuracy of the brain's awareness of the locations of the objects.

Egocentric representation [148] is a strategy that is more typically utilized in everyday scenarios like reaching for a pen or remembering where the key chain is in the room. Our current body position is the reference in egocentric representations. The pen is on the table in front of us, about 30 degrees around our current facing orientation. We frequently use this type of representation to avoid collisions with objects and traverse our immediate, peripersonal space, as demonstrated by various studies of human spatial cognition [153]–[155].

Several studies imply that egocentric representations are high-resolution visual snapshots linked to our current head direction. We may create a single coherent egocentric representation related to our current location in space by taking a succession of these high-resolution, static, body-referenced images [156]. These representations can then be updated as we walk through an environment, constituting the basis for path integration [143], [157]. However, these representations diminish during disorientation [154]–[156] or in large-scale environments [158].

A beacon or reaction method [128], [159]–[161] is the third navigation strategy. It requires learning a series of behavioural actions from specific environment points that act as stimuli. With a succession of stimulus-response associations, one can learn to navigate from home to work in an automated manner. These responses may involve turning at a specific corner or building, with the corner and building acting as stimuli and the

response involving turning [161]–[163]. The reaction approach is inflexible when the remembered route is unavailable since it can not create a novel way to the target place.

As a result, navigation tasks are as simple as moving closer to or further away from a specific object. Its position on the retina grows or shrinks, giving an important clue for locating the object. When combined with egocentric codes like "right" and "left," Beacon navigation is a replica of how we travel using mobile devices like GPS on our phones. Like GPS, beacon strategies decrease the navigator's job to simply search for a specific landmark and link it with a response.

5.1.3 Mapping and Navigation Tasks in Human Brain:

We presented neuroscience and behavioural analysis findings describing how our brain solves navigation tasks and forms a surroundings model. These findings could be summarized as follows:

1. The human brain builds short-term, high-resolution body-referenced visual maps (egocentric) to be used with the motor sensors' motion predictions to solve the immediate navigational tasks; however, these maps fade with time and when navigating in large spaces (environmental spaces). These maps are the equivalent of the local maps that VO and VSLAM systems use in pose estimation.
2. Alongside the egocentric maps, the human brain forms a more general representation that integrates visual and motor sensors over time, generating allocentric environmental models. The allocentric maps are referenced outside the human body to a distinct point or a landmark in the environment. This type of model is equivalent to the global maps in SLAM systems.

3. Unlike SLAM systems in the literature, the human brain depends on the environment's geometry and layout to recognize and map the visited places.
4. The brain mainly tracks the surrounding objects for pose estimation. It does not depend on feature points unless they are essential for describing the space layout or belong to a distinct landmark. Alternatively, the brain abstracts all the background points into colour and shade information.

These navigational guidelines resulted from generations of training and evolution of the human brain, the most sophisticated computer ever existed. Designing a SLAM solution based on these findings could yield scalable, more robust and optimized solutions.

5.2 Layout Estimation

Layout estimation research considers the task of estimating the spatial layout of an indoor scene from single or multiple images. Layout estimation solutions aim to delineate the walls, ground, and ceiling boundaries [164]. Early layout estimation solutions approached the problem intuitively as a semantic segmentation problem, assigning geometric context classes (floor, walls, and ceiling) to each pixel and then trying to obtain room layout keypoints and boundaries based on the pixel-wised labels [165]. However, it is non-trivial to identify layout keypoints and boundaries from the raw pixel output. Furthermore, the traditional pixel-based representation created ambiguity as researchers have shown that CNNs often have difficulty distinguishing between different surface identities [166]. This phenomenon largely undermines the overall room layout estimation performance.

RoomNet [167] reformulated the task of room layout estimation as an ordered room layout keypoint localization problem which can be directly addressed using CNNs as a

trainable end-to-end problem. RoomNet considered a list of different layout possibilities with their respective keypoint definition that could be inferred from a single image; these possibilities are first introduced by [168] and are listed in Figure 5.1. These 11 layouts cover most possible situations under typical camera poses and common cuboid representations under the "Manhattan world assumption" [169]. Once the trained model estimates correct keypoint locations with an associated room type, we can simply connect these points in a specific order to produce a boxy room layout representation.

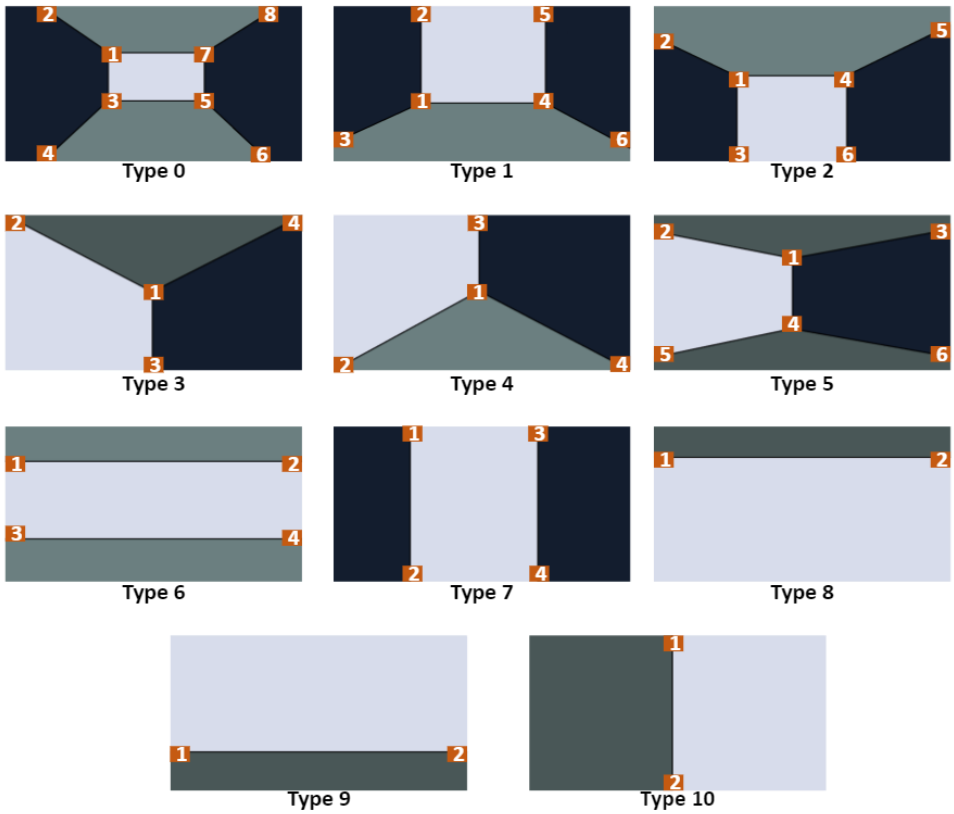


Figure 5.1 Definition of scene layout types.

RoomNet adopts the SegNet architecture proposed in [170] with modifications. The base architecture of RoomNet adopts the same convolutional encoder-decoder network as SegNet. Using an image of an indoor scene, the system can directly identify a set of 2D room layout keypoints to recover the room layout. Each keypoint is represented by a 2D

Gaussian heatmap centred at the true keypoint location. The encoder-decoder architecture processes the information flow through bottleneck layers, enforcing it to implicitly model the relationship among the keypoints that encode the 2D structure of the room layout.

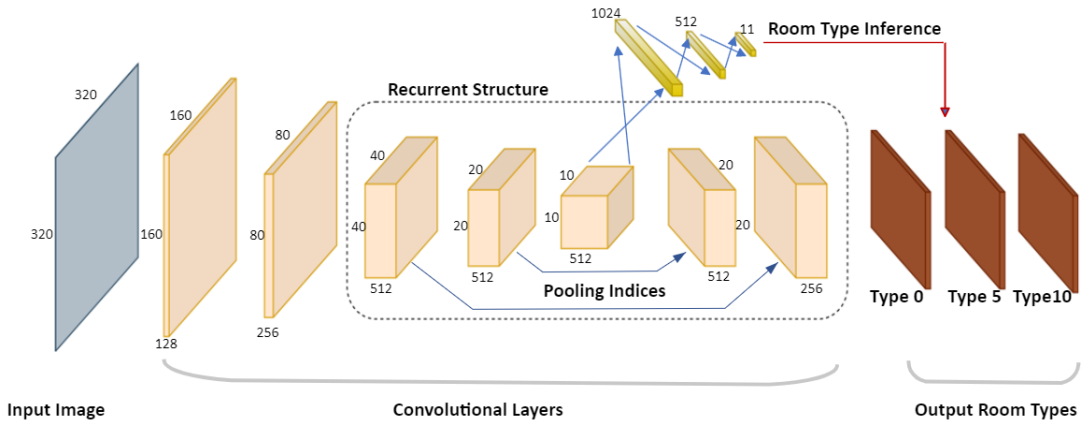


Figure 5.2 RoomNet base architecture.

RoomNet predicts room layout keypoints and the associated room type with respect to the input image in one forward pass. To achieve this goal, the channels in the output layer are 11 as the number of the layout scenarios shown in Figure 5.1. The output channel is selected by the side head with fully connected layers to the bottleneck [171]–[174], as shown in Figure 5.2.

5.3 Proposed System

We propose a semantically improved visual SLAM solution inspired by human reasoning in solving navigational tasks. The proposed SLAM builds an accurate joint map for sparse feature models for foreground objects, the environment's geometric bounds and detected sparse feature points to represent the background. The system depends on a RoomNet-trained network to estimate the layout keypoints in the input image sequences. RoomNet considers a Cuboid shape of the indoor spaces and infers its cuboid corners. The proposed system then incorporates the layout constraints into the global optimization

process to enhance the overall trajectory accuracy. The proposed system uses ORBSLAM2 as its SLAM backbone and propagates the semantic and geometric inferences across the tracking, mapping and loop closure tasks with the following novelties:

1- A complete visual SLAM system tracks and maps geometric and semantic structures in indoor environments. The system was built on ORB-SLAM2 as its SLAM backbone and redesigned its threads to utilize semantic observations throughout the tracking, mapping and loop closing tasks.
2- The system also uses an offline created object model database to provide a point of view independent object tracking and provide the physical relationship between the object points as a set of constraints that anchor the odometry positional drifts.
3- Because background feature points are hard to precisely re-detected and matched, the proposed system is configured to allow object points to be more influential in pose optimizations.
4- With most of the images in the tracked sequences containing at least one cuboid corner, tracking and mapping threads are modified to detect and track cuboid corners and include them in the map optimization process.
5- The proposed system introduces the idea of slicing the map into multiple geometrical links. Each link represents keyframes connecting two environment corners and exploits the geometric relationship between these corners as map optimization constraints.
6- Two modifications are introduced to the loop closure thread: First, the local map is queried for objects and cuboid points to verify loop detections before accepting

it. The second modification introduces a new loop closure approach by detecting all the cuboid corners of the traversed space. Four cuboid corners allow the system to optimize the four edges defining the space limits.

5.4 System Description

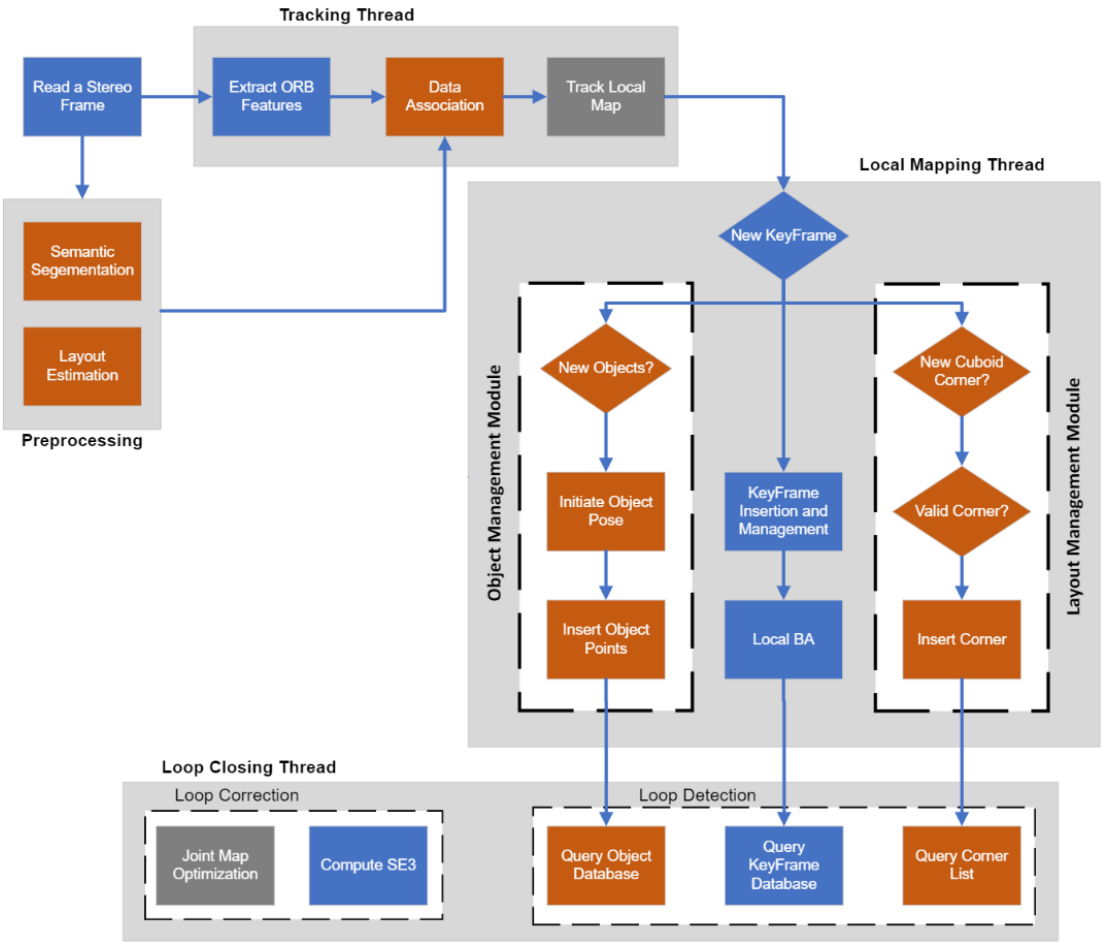


Figure 5.3 Proposed Solution Flow Diagram.

Figure 5.3 shows an overview of the proposed system. Like ORB-SLAM2, the developed system consists of three threads: tracking, local mapping and Loop closing. Each thread was redesigned to exploit the semantic and geometric information to accurately estimate the trajectory and environment map. The system embedded an offline learnt object model database for the common objects in AI2Thor [175] scenes. This section presents a

detailed description of the system's threads, highlighting the modified (grey) and the new (orange) components introduced to the ORB-SLAM2.

5.4.1 Object Model Database:

One of the system contributions is creating a sparse 3D point model database for common objects in AI2Thor simulator scenes. Figure 5.4 shows the model database creation and query flow diagram. To create an object model, 12 images of the object are taken 30 degrees apart from the same distance. Then ORB feature points are extracted and matched using the same ORB extractor and matcher from ORB-SLAM2. The matched points are then triangulated to create a point cloud of the features.

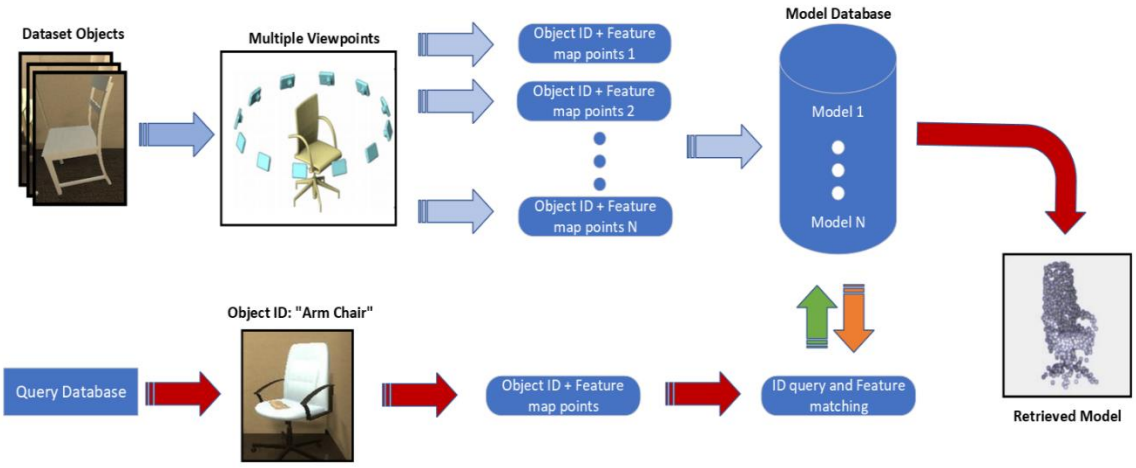


Figure 5.4 Object Model Database Flow Diagram.

For each point, the model saves the BRIEF descriptor and the transformation T_{op} from the center point of the point cloud. Finally, each point also contains the mean viewing angle at which it could be detected. To query the database for a new detection, the system uses the object ID provided by the semantic segmentation node to retrieve all models of the same kind (Chair, Sofa, …). Then perform feature matching on the retrieved models to select the right model and a rough estimate of its relative pose with respect to the camera

frame. The feature matching process considers the relative pose of the features and whether the detected object is cropped by the image pounds.

5.4.2 Frame Preprocessing:

While AI2Thor provides a semantic segmentation solution for each frame, the layout estimation solution is still needed. The solution should be able to identify and track the cuboid vertices of the surrounding space. RoomNet was utilized to perform this task for its compatible and efficient representation. We trained the RoomNet network on an AI2Thor-generated dataset. The dataset was composed of 10k images divided into 8k for training and 2K for the test data. To avoid overfitting, the image sequences used for the training were different from the test and trajectory sequences. The room type and layout keypoints indices are generated for the trajectory sequences. To evaluate the accuracy of the trained network, two standard room layout estimation evaluation metrics were used:

(1) Pixel error: is the pixel-wise error between the predicted surface labels and ground truth labels.

(2) Keypoint error: is the average Euclidean distance between the predicted keypoint and annotated keypoint locations, normalized by the image-diagonal length.

The model is trained with pixel-wise cross-entropy loss, optimized with Stochastic Gradient Descent (SGD), with a patch size of 32 and a 0.001 learning rate. The trained RoomNet network achieved 7.01% keypoint error and 10.9% pixel error. This high accuracy is due to the clean images generated from the AI2Thor simulator. The term clean means that most of the presented corners are un-occluded and visible in the images. Finally, our presented SLAM system only considers room types 0 – 5, as types 6 – 10 provide no real room corners (refer to Figure 5.1).

5.4.3 System Threads:

Tracking Thread:

The tracking thread localizes the camera with every input frame by finding feature matches to the local map and minimizing the reprojection error by applying motion-only BA. The flow of the proposed system's tracking thread is as follows:

1- After extracting ORB features from the current frame, all detected features are annotated using the semantic segmentation input data.

2- The previous frame is searched for matches; the encoded semantic annotations guide the search limiting the mismatches and the number of match candidates. An initial estimate of the pose is then optimized using the found correspondences.

3- Once an initial pose is acquired and we have an initial set of matches, we project the local mappoints into the current frame; this includes observed feature points, object points and layout corner points. If the point projection lies inside the image bounds and with a viewing angle less than 60°, we search the still unmatched features in the current frame and associate the mappoint to the best match.

4- Finally, the camera pose is optimized with Motion-only bundle adjustment, a variant of the generic BA described in Chapter 2, where the camera pose $\boldsymbol{T}_{cw} = [\boldsymbol{R}_{cw}|\boldsymbol{P}_{cw}]$ is optimized to minimize the reprojection error between the 3D map points $\boldsymbol{X}_w$ matched to the current frame's 2D keypoints x_c:

$$\boldsymbol{T}_{cw} = \underset{\boldsymbol{R}_{cw},\boldsymbol{P}_{cw}}{\operatorname{argmin}} \sum_j \rho\left(\left\|x_c^j - \pi_m\left(\boldsymbol{R}_{cw}\boldsymbol{X}_w^j + \boldsymbol{P}_{\boldsymbol{cw}}\right)\right\|_{\Sigma_j}^2\right), \tag{5.1}$$

All map points included in the pose optimization are fixed; only the camera pose is optimized. Our system's optimization process is modified to respect the physical

relationship governing the map points belonging to the same object. The system considers each matched mappoint an observation of either an object or the background. Background points are usually mappoints that lie on surfaces like floors, walls and ceilings. These surfaces are generally homogeneous, making it hard to track background points accurately, yet background points are usually the majority of the tracked mappoints. We propose performing the pose optimization with multiple subsets of the matched points. Each subset is composed of the mappoints belonging to the same object instance to remedy the effect of erroneous background matches. The final estimate is the mean of the subset estimates as shown in the equation:

$$\boldsymbol{T}_{cw} = \frac{1}{N}\sum_{N} \underset{\boldsymbol{R}_{cw},\boldsymbol{P}_{cw}}{\operatorname{argmin}} \sum_{j} \rho\left(\left\| x_c^j - \pi_m\left(\boldsymbol{R}_{cw}\boldsymbol{X}_w^j + \boldsymbol{P}_{\boldsymbol{cw}}\right)\right\|_{\Sigma_j}^2\right), \tag{5.2}$$

Where N is the number of tracked object instances, our experiments on the generated AI2Thor dataset showed that tracking objects improve trajectory accuracy.

Local Mapping Thread:

In ORB-SLAM2, the Local Mapping thread manages the local map and optimizes it by performing local BA. The map management includes adding new keyframes and removing redundant keyframes and map points. The proposed system added object management and map layout management tasks to the local mapping thread.

Object management module: Each object detection in newly created keyframes that does not exist in the local map is considered a new object. The object model database is queried for the detected object model. New objects are then aligned by computing the position of the object center point and the object orientation using the matched points. After performing any map optimization (global/local), the objects within the optimized map are

realigned. After object alignment, the local mapping creates object points for all the object model points regardless of their existence in the view of the keyframe. All points are then added to the object instance.

Layout management module: Indoor environments are characterized by limited size, which enforces low-speed movements and many pure rotations to correct directions. In our collected and tested datasets (real and simulated) for cuboid spaces, we noticed that over 80% of the images contain at least one cuboid corner. In the remaining 20%, the images are either occluded by close objects or walls, suggesting that the following motion would be a large rotation to correct the direction and avoid a collision. Large pure rotations are usually miscalculated or even could cause the tracking thread to lose track. Tracking cuboid corners can restore lost tracking and correct wrongly estimated rotations. Detected corners are kept in an ordered stack in the same order of corner observations. The corner stack can only contain four corners at max. The layout management module shown in Figure 5.3 as a part of the local mapping thread limits the solution drifts as follows:

1- Each corner instance should contain a list of all object instances in the local map when the corner was observed with the relative transformation between the object instance and the corner point. Object instance lists are used in corner matching and are updated with each new corner observation.

2- Each cuboid corner detected in newly created keyframes that do not have matches in the local map is required to pass two tests. First, the corner should be cross-examined with the previously detected corners geometrically and through object list validation to ensure it is a new corner. Then the distance d_{c_i,c_m} between the

detected corner c_i and the center point of the map c_m, which is updated with each new keyframe, should meet the following condition:

$$d_{c_i,c_m} = d_{mean} \pm 10\%, \quad (5.3)$$

Where d_{mean} is the mean distance between all the corners in the corner stack.

3- The first and second cuboid corners are directly inserted into the corner list. Now, the system can correct every newly detected corner by projecting the distance between the newly detected corner and the last inserted corner in the corner list in the direction of the perpendicular unit basis of the last link.

4- With 3rd and 4th corner detection, the estimated map is optimized using the corrected corner point position in edge optimizations. Edge optimization is the process in which the geometric constraints between the different corners are enforced. The optimizations are performed whether the corner is new or already included in the list as long as it is its first appearance on the current local map. An edge e_i is all keyframes that connect two subsequent corners. In other words, all keyframes inserted in the map from the beginning of the observation of the corner i to the end of the observation of corner $i + 1$. Edge optimization is similar to local map optimization, with one key difference, corner points are fixed.

With four corners already in the corner's list and the oldest corner is redetected, a full loop has already been navigated. In this case, we perform four edge optimizations which is equivalent to a loop closure. As presented in the analysis section, this process should yield a more accurate trajectory shape and scale. Also, edge optimization gives a much more accurate initial state to the full graph bundle adjustment, which decreases the number of iterations needed to find the final state.

Loop Closure Thread:

The Loop Closing thread is assigned to detect large loops and correct the accumulated drift by performing a pose-graph optimization. After accepting a loop closure candidate, this thread launches a fourth thread to perform full BA for the pose-graph optimization to compute the optimal map structure and motion solution. Originally, the ORBSLAM2 detects if a loop is closed by recognizing the visual bag of words of a previously visited scene. The proposed system implements an enhanced loop closure detection approach based on the layout corner detection. The new approach is performed by the layout management module in the local mapping when detecting four different and valid cuboid corners. The proposed system added a layer of verification for loop closure candidates, which query the objects and cuboid corners at both sides of the loop to validate its correctness.

5.5 Experiments and Analysis

In this section, the experiments are discussed, and a full accuracy analysis of the proposed system against ORB-SLAM2 is presented. As ORB has become the standard VSLAM in the literature as can be seen in [49] and [50], ORB-SLAM2 were considered as the baseline of the comparisons to evaluate the enhancements introduced by the proposed modifications.

5.5.1 Test Data:

The developed system was tested on multiple sequences generated from the AI2Thor environment simulator embodied AI agents [176]. AI2Thor was created by Allen Institute for AI in 2016 to facilitate research in many domains, including deep reinforcement learning, object detection and Segmentation, motion planning and visual

SLAM. AI2-THOR provides near photo-realistic 3D indoor scenes, where AI agents can navigate the scenes and interact with objects to perform tasks. The simulator offers RGB and depth images, annotated semantic Segmentation, and navigational ground truth [170]. To generate stereo frames from the simulator, a second view camera was added to the scenes, and the position and orientation of the camera were updated with each time step. The experiments were performed on five stereo/RGBD image sequences generated from the AI2Thor simulator. The sequences represent a full loop in the scenes with the following names: TrainFloor10_1, TrainFloor10_2, TrainFloor10_3, TrainFloor10_4 and TrainFloor10_5. The trajectories experienced many large pure rotations (with no combined translation) to test the robustness and accuracy of the systems. The length of the exported sequences varies from 1.3K to 3.6K steps per sequence.

5.5.2 Evaluation Metrics

To evaluate the proposed SLAM system against ORB-SLAM2, the estimated camera motion for both systems is compared against the true trajectory using two frequently utilized methods: the relative pose error (RPE) and the absolute trajectory error (ATE). RPE measures the difference between the estimated motion and the true motion. It can evaluate the drift of a visual odometry system [177] which is especially useful if only sparse and relative relations are available as ground truth. Instead of judging the drift of relative poses, the ATE tests global trajectory consistency. ATE aligns the two trajectories and then directly evaluates the absolute pose differences. This method is well suited for assessing visual SLAM systems [178] but requires that absolute ground truth poses are available. Finally, we use the frame rate to judge the systems' complexity.

5.5.3 Results and Analysis:

The quantitative comparison results are shown in Table 5.1 to Table 5.3. RMSE, Mean Error, Median Error and Standard Deviation (SD) values are presented, while RMSE and SD are more concerned because they can better indicate the robustness and stability of the system. We also show the values of improvement compared to the original ORB-SLAM2. The improvement values in the tables are calculated as follows:

$$\eta = \frac{o - b}{o} \times 100\%, \tag{5.4}$$

Where η represents the value of improvement, o represents the value of ORB-SLAM2 and b represents the value of the proposed system.

Table 5.1 Results of Metric Rotational Drift (RPE).

Sequences	ORB-SLAM2				Proposed				Improvements			
	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.
Floor10_1	4.299	3.472	3.662	2.537	2.567	2.086	2.208	1.496	%40.2	%39.9	%39.71	%41.0
Floor10_2	1.501	1.246	1.367	0.839	0.753	0.609	0.630	0.443	%49.8	%51.1	%53.89	%47.2
Floor10_3	1.395	0.936	0.728	1.035	0.798	0.648	0.639	0.467	%75.8	%24.9	%53.66	%62.6
Floor10_4	3.604	2.570	1.493	2.528	2.107	1.533	0.979	1.446	%41.5	%40.3	%34.41	%42.7
Floor10_5	3.699	3.231	3.092	1.799	2.431	2.126	2.128	1.179	%34.2	%34.2	%31.19	%34.4

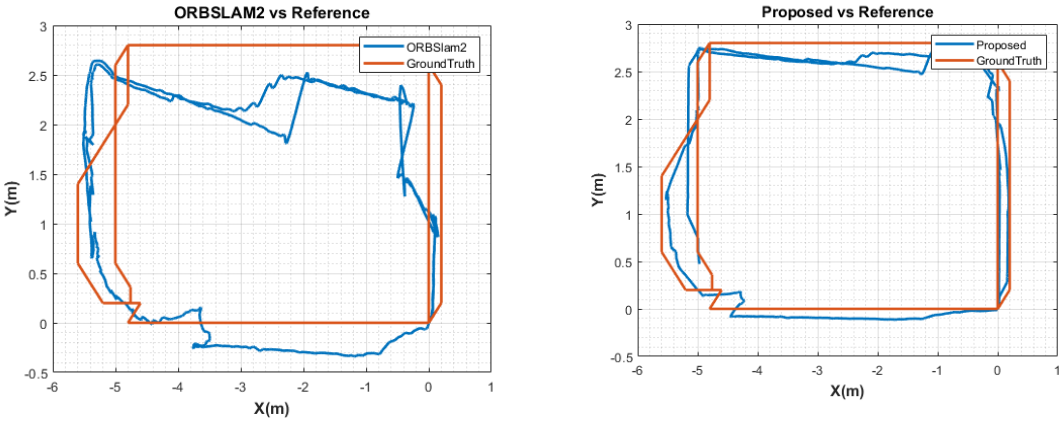
Table 5.2 Results of Metric Translational Drift (RPE).

Sequences	ORB-SLAM2				Proposed				Improvements			
	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.
Floor10_1	0.0036	0.095	0.064	0.164	0.0004	0.032	0.020	0.058	%87.77	%68.55	%66.28	%64.62
Floor10_2	0.0035	0.153	0.144	0.109	0.0009	0.063	0.038	0.072	%73.79	%58.52	%73.42	%33.74
Floor10_3	0.0014	0.058	0.035	0.103	0.0003	0.043	0.016	0.038	%75.88	%24.94	%53.66	%62.66
Floor10_4	0.0038	0.105	0.076	0.165	0.0016	0.098	0.075	0.083	%56.89	%7.193	%1.594	%49.48
Floor10_5	0.0004	0.045	0.030	0.047	0.0003	0.041	0.012	0.045	%11.26	%7.752	%60.60	%4.074

Table 5.3 Results of Metric Absolute Trajectory Error (ATE).

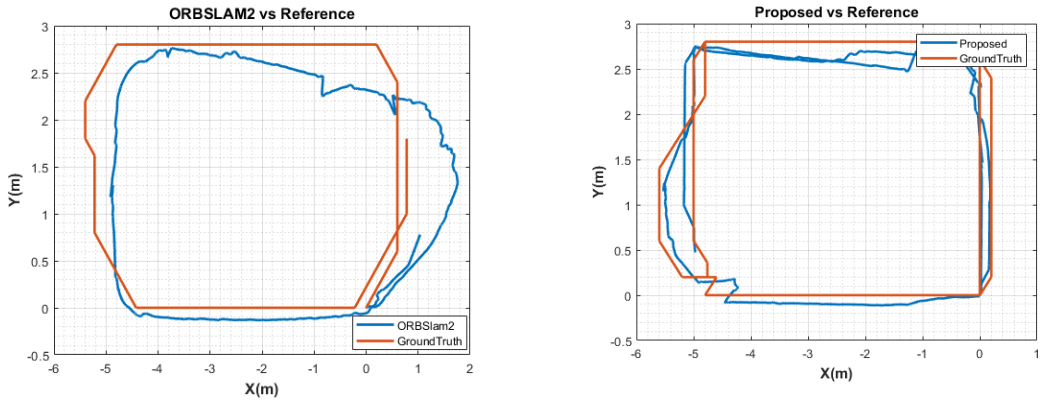
Sequences	ORB-SLAM2				Proposed				Improvements			
	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.	RMSE	MEAN	Median	S.D.
Floor10_1	0.696	0.552	0.504	0.424	0.23	0.181	0.158	0.142	%66.96	%67.25	%68.66	%66.47
Floor10_2	0.367	0.334	0.353	0.152	0.15	0.138	0.145	0.061	%58.97	%58.74	%58.92	%60.08
Floor10_3	1.170	0.188	0.157	1.156	0.36	0.013	0.079	0.359	%69.36	%92.96	%49.72	%68.98
Floor10_4	0.752	0.655	0.643	0.369	0.15	0.132	0.130	0.075	%79.83	%79.85	%79.79	%79.76
Floor10_5	0.717	0.514	0.478	0.499	0.261	0.176	0.162	0.193	%63.53	%65.77	%66.11	%61.29

Table 5.1 to Table 5.3 illustrate that by tracking objects and geometrical structures in indoor environments, the proposed SLAM solution outperforms ORB-SLAM2 by order of magnitude. The RMSE and SD improvement values can reach 79% in terms of ATE. The results indicate that the proposed approach can significantly improve the robustness and stability of the SLAM system in indoor environments.



(a) (b)

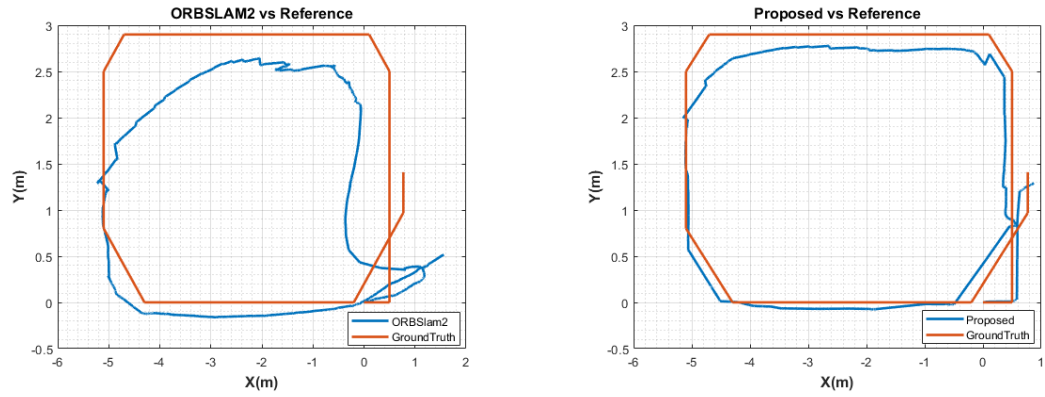
Figure 5. 5 Estimated Trajectories for Floor10_1 Sequence.



(a) (b)

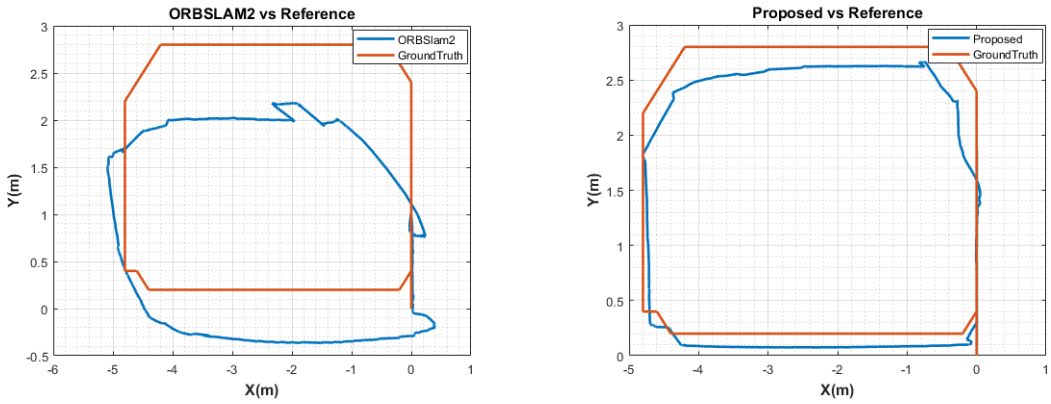
Figure 5.6 Estimated Trajectories for Floor10_2 Sequence.

Figure 5. 5 to Figure 5.9 show the estimated trajectory of the proposed SLAM system and ORB-SLAM2 against the ground truth. We can notice how starting the sequence by large pure rotation severely affects ORB-SLAM2 trajectory estimation in sequences 3 and 4, which also suffered large inaccuracy in the first sequence due to multiple large pure rotations. Our solution recovered from these rotations by tracking the environment boundaries in all cases. Finally, while the ORB-SLAM2 ran with 10 fps on stereo frames, the proposed system ran with 8 fps.



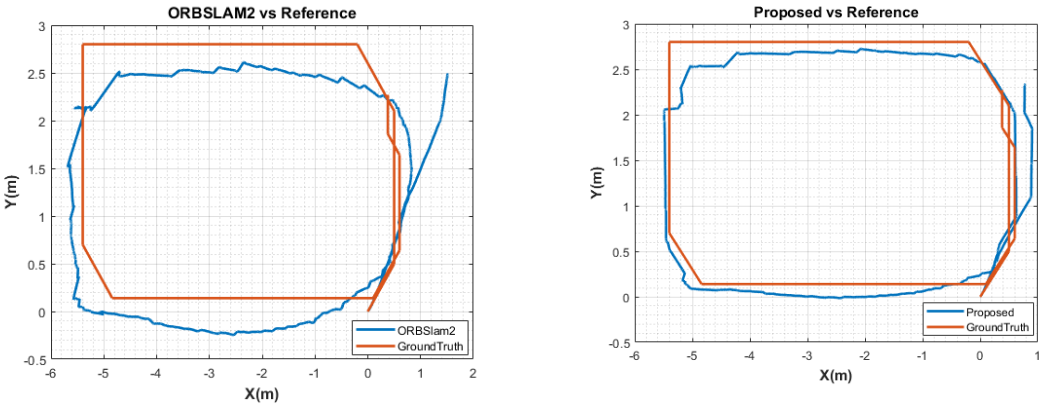
(a) (b)

Figure 5.7 Estimated Trajectories for Floor10_3 Sequence.



(a) (b)

Figure 5.8 Estimated Trajectories for Floor10_4 Sequence.



(a) (b)

Figure 5.9 Estimated Trajectories for Floor10_5 Sequence.

5.6 Summary

The human brain can perform navigation-related tasks flexibly and with very low accumulated errors. In this chapter, an improved visual SLAM system was presented. A new indoor SLAM system is inspired by the human ability to understand and relate semantic and geometrical information and then exploit it to navigate the environment accurately. The proposed system incorporated RoomNet, a CNN network designed to detect geometric properties of the image scenes and successfully used this information to

enhance its trajectory estimation. The proposed SLAM solution utilized the geometric constrained between the environment corners to correct the system drifts and create an improved loop closure approach. The presented work included the creation of an object model database to provide more robust object tracking. The proposed system showed robustness and accuracy in all tested sequences and outperformed its backbone SLAM engine, ORB-SLAM2, especially against pure rotations.

Chapter 6: Conclusions and Future work

6.1 Conclusion

This thesis presented sensor fusion and machine learning approaches to aided visual navigation systems in indoor environments. A thorough background review, detailed problem statement, in-depth explanation of the proposed solutions and experimental results on real and simulated datasets were discussed. Chapter 2 presented a detailed discussion about indoor navigation solutions. The chapter also investigated the bottlenecks of the existing SLAM solutions. The reported limitations span both robustness and accuracy of the trajectory estimations. The experimental results showed the following problems: losing track of the position, large drifts in the estimated trajectory and wrong loop closure detections.

A full design and implementation of a multi-sensor indoor tracking system were discussed in Chapter 3. The system incorporated many navigation sensors and accurately collected massive data in ROSbag format. The system utilized the robot operating system (ROS), and tasks were distributed on three different computing platforms communicating wirelessly. Each runs tailored software to successfully control the sensors and interact with each other in real-time.

In Chapter 4, a novel IMU-aided visual odometry system was presented. The system efficiently switches between stereo and monocular visual odometry schemes. Experimental results on a real dataset showed that the proposed approach added robustness to the system against rapid movements and single-camera occlusions without sacrificing the system's accuracy. Chapter 4 also illustrated yet another sensor fusion approach. The presented system integrated two navigation solutions with the visual odometry, UWB

positioning and an INS-based navigation solution. The fusion EKF was designed to utilize the platform motion constraints as a non-holonomic constraint, limiting the overall solution drift. Compared to the single sensors solutions, the system showed higher accuracy even with up to 10 seconds of UWB outages.

Chapter 5 presented an improved visual SLAM system using semantic segmentation and layout estimation. The system design was inspired by the human ability to estimate its position accurately by tracking the semantic and geometric structures in the traversed environment. By integrating the mature semantic segmentation and layout estimation approaches, the solution imitated the human brain's awareness of the objects and the space limits. It improved the accuracy and robustness of the estimated trajectory.

6.2 Potential Direction for Future Research

The research works presented in this thesis have opened new opportunities for further investigation in the following potential directions:

Optimized Semantic SLAM for real-time indoor navigational tasks: An intuitive future step is to optimize the presented visual SLAM implementation for real-time embedded applications. The optimization should aim to reduce the memory and processing power requirements of the solution to be suitable for the limited resources of the commercial embedded platforms. The suggested optimization is twofold: 1) to optimize the neural network's size of the semantic segmentation and layout estimation modules. However, this research direction should be guided by the fact that reducing the neural network size has a trade-off with its generalizability and accuracy. 2) to optimize the SLAM implementation. The current implementation of the proposed SLAM solution considers both feature points and object instances in the tracking and local mapping

threads. A good potential optimization practice is to consider only the detected object instances and neglect the corresponding feature points entirely whenever enough objects are present. Treating object instances as visual features increases tracking accuracy and reduce map size allowing for faster BA optimizations.

Multi-Modal layout estimation for generalized Semantic SLAM system: This Ph.D. thesis illustrated the power of exploiting environments' layouts as a geometrical constraint for more accurate trajectory estimations. An encouraging research direction is to generalize the layout estimation module in our proposed SLAM to include more environment models. For indoor environments, these layout models could include solutions for stairs, hallways and non-cuboid spaces. The system should be trained to recognize and switch between these space portions to estimate longer trajectories and model larger environments. Another benefit of pursuing this research direction is allowing the system to operate beyond indoor applications. The improved SLAM system can be used outdoors, in tunnels, or even in agricultural fields by redefining the environment's boundaries.

Explore different approaches for sensor fusion: In this thesis, the developed sensor fusion solutions used EKF to estimate the systems' states optimally. However, EKF requires a robust formulation of the error models and a long iterative calibration process to estimate sensors' error parameters. The higher the number of integrated sensors, the longer and more complicated the calibration process is. Additionally, the slightest drifts in the estimated error model compromise the solution's accuracy. A benifitial research trend is to explore different sensor fusion approaches to shorten the calibration process or enhance the overall system accuracy and realtime performance. Examples of such approaches are

the Particle Filters (PF) [179], Dynamic Data Reconciliation (DDR) [180] and the Random forest filter (RFF) [181].

References

[1] Achtelik, Markus, et al. “Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-Denied Indoor Environments.” *Unmanned Systems Technology XI*, vol. 7332, 2009, doi:10.1117/12.819082.

[2] Rubio, Francisco, et al., “A Review of Mobile Robots: Concepts, Methods, Theoretical Framework, and Applications.” *International Journal of Advanced Robotic Systems*, vol. 16, 2019, doi:10.1177/1729881419839596.

[3] Cadena, Cesar, et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age.” *IEEE Transactions on Robotics*, vol. 32, no. 6, 2016, doi:10.1109/TRO.2016.2624754.

[4] Scaramuzza, Davide, and Friedrich Fraundorfer, “Tutorial: Visual Odometry.” *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, 2011, doi:10.1109/MRA.2011.943233.

[5] Ekstrom, Arne D., and Eve A. Isham, “Human Spatial Navigation: Representations across Dimensions and Scales.” *Current Opinion in Behavioral Sciences*, vol. 17, 2017, doi:10.1016/j.cobeha.2017.06.005.

[6] Sualeh, M. and Kim, ‘Simultaneous Localization and Mapping in the Epoch of Semantics: A Survey’, *International Journal of Control, Automation and Systems*, 17(3), pp. 729–742. doi:10.1007/s12555-018-0130-x.

[7] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, “Rover navigation using stereo ego-motion,” *Robot. Auton. Syst.*, vol. 43, no. 4, pp. 215–229, Jun. 2003, doi: 10.1016/S0921-8890(03)00004-6.

[8] H. Moravec, “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover,” in *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University \& doctoral dissertation, Stanford University*, 1980.

[9] Forstner, W. “A Feature Based Correspondence Algorithm for Image Matching.” *From Analytical to Digital. Proc. Symposium, ISPRS, Commission III, Rovaniemi, 1986, Vol. 3*, 1987.

[10] Harris, C. G., and J. M. Pike. “3D Positional Integration from Image Sequences.” *Image and Vision Computing*, vol. 6, no. 2, 1988, doi:10.1016/0262-8856(88)90003-0.

[11] Jianbo Shi and Tomasi, "Good features to track," *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* 1994, pp. 593-600, doi: 10.1109/CVPR.1994.323794.

[12] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 3951 LNCS. doi: 10.1007/11744023_34.

[13] D. G. Low, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, 2004., doi: 10.1023/B:VISI.0000029664.99615.94.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, vol. 110, no. 3, 2008, doi: 10.1016/j.cviu.2007.09.014.

[15] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5305 LNCS. doi: 10.1007/978-3-540-88693-8_8.

[16] Torr, P. H. S., and Murray, D. W. "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *International Journal of Computer Vision*, V. 24, No. 3, 1997, pp. 271–300.

[17] Kristy Sim, and Hartley, R. "Recovering Camera Motion Using L\infty Minimization." 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06), vol. 1. New York, NY, USA, IEEE, 2006. pp. 1230–7.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, 1981, doi: 10.1145/358669.358692.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. 2004. doi: 10.1017/cbo9780511811685.

[20] Nister, D., Naroditsky, O., and Bergen, J. "Visual odometry." Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., vol. 1. Washington, DC, USA, IEEE, 2004. pp. 652–9.

[21] Moreno-Noguer, Francesc, et al. “Accurate Non-Iterative O(n) Solution to the PnP Problem.” *Proceedings of the IEEE International Conference on Computer Vision*, 2007, doi:10.1109/ICCV.2007.4409116.

[22] Fraundorfer, Friedrich, et al. “A Constricted Bundle Adjustment Parameterization for Relative Scale Estimation in Visual Odometry.” *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, doi:10.1109/ROBOT.2010.5509733.

[23] Sünderhauf, Niko, et al. “Visual Odometry Using Sparse Bündle Adjustment on an Autonomous Outdoor Vehicle.” *Informatik Aktuell*, 2006, doi:10.1007/3-540-30292-1_20.

[24] Konolige, Kurt, et al. “Large-Scale Visual Odometry for Rough Terrain.” *Springer Tracts in Advanced Robotics*, vol. 66, 2010, doi:10.1007/978-3-642-14743-2_18.

[25] Tardif, Jean Philippe, et al. “A New Approach to Vision-Aided Inertial Navigation.” *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, doi:10.1109/IROS.2010.5651059.

[26] Triggs, Bill, et al. “Bundle Adjustment – a Modern Synthesis.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1883, 2000, doi:10.1007/3-540-44480-7_21.

[27] Durrant-Whyte, Hugh, and Tim Bailey. “Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms.” *Robotics & Automation Magazine*, vol. 2, 2006, doi:10.1109/MRA.2006.1638022.

[28] Thrun, Sebastian. “Robotic Mapping: A Survey.” *Science*, vol. 298, no. February, 2002.

[29] Mahon, Ian, et al. “Efficient View-Based SLAM Using Visual Loop Closures.” *IEEE Transactions on Robotics*, vol. 24, no. 5, 2008, doi:10.1109/TRO.2008.2004888.

[30] Kschischang, Frank R., et al. “Factor Graphs and the Sum-Product Algorithm.” *IEEE Transactions on Information Theory*, vol. 47, no. 2, 2001, doi:10.1109/18.910572.

[31] Lu, F., and E. Milios. “Globally Consistent Range Scan Alignment for Environment Mapping.” *Autonomous Robots*, vol. 4, no. 4, 1997, doi:10.1023/A:1008854305733.

[32] Gutmann, Jens Steffen, and Kurt Konolige. “Incremental Mapping of Large Cyclic Environments.” *Proceedings - 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 1999*, 1999, doi:10.1109/cira.1999.810068.

[33] Grisetti, Giorgio, et al. “A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps Using Gradient Descent.” *Robotics: Science and Systems*, vol. 3, 2008, doi:10.15607/rss.2007.iii.009.

[34] Dellaert, Frank, and Michael Kaess. “Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing.” *International Journal of Robotics Research*, vol. 25, 2006, doi:10.1177/0278364906072768.

[35] Folkesson, John, and Henrik I. Christensen. “Graphical SLAM for Outdoor Applications.” *Journal of Field Robotics*, vol. 24, no. 1–2, 2007, doi:10.1002/rob.20174.

[36] Kaess, Michael, et al. “ISAM2: Incremental Smoothing and Mapping Using the Bayes Tree.” *International Journal of Robotics Research*, vol. 31, no. 2, 2012, doi:10.1177/0278364911430419.

[37] Olson, Edwin, et al. “Fast Iterative Alignment of Pose Graphs with Poor Initial Estimates.” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, 2006, doi:10.1109/ROBOT.2006.1642040.

[38] Thrun, Sebastian, and Michael Montemerlo. “The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures.” *International Journal of Robotics Research*, vol. 25, 2006, doi:10.1177/0278364906065387.

[39] Kaess, Michael, et al. “ISAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering.” *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, doi:10.1109/ICRA.2011.5979641.

[40] Dellaert, Frank. “Factor Graphs and {GTSAM}.” *Technical Report*, 2012.

[41] Dellaert, Frank, and Michael Kaess. “Factor Graphs for Robot Perception.” *Foundations and Trends in Robotics*, vol. 6, no. 1–2, 2017, doi:10.1561/2300000043.

[42] Kaess, Michael, et al. “ISAM: Incremental Smoothing and Mapping.” IEEE Transactions on Robotics, vol. 24, no. 6, 2008, doi:10.1109/TRO.2008.2006706.

[43] Kümmerle, Rainer, et al. “G2o: A General Framework for Graph Optimization.” *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, doi:10.1109/ICRA.2011.5979949.

[44] “Ceres Solver — A Large Scale Non-linear Optimization Library.” http://ceres-solver.org/ (accessed Jun. 09, 2021).

[45] Polok, Lukas, et al. Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics. 2016, doi:10.15607/rss.2013.ix.042.

[46] Hesch, Joel A., et al. “Camera-IMU-Based Localization: Observability Analysis and Consistency Improvement.” *International Journal of Robotics Research*, vol. 33, no. 1, 2014, doi:10.1177/0278364913509675.

[47] Huang, Guoquan P., et al. “An Observability-Constrained Sliding Window Filter for SLAM.” *IEEE International Conference on Intelligent Robots and Systems*, 2011, doi:10.1109/IROS.2011.6048760.

[48] Kottas, Dimitrios G., et al. “On the Consistency of Vision-Aided Inertial Navigation.” *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, edited by Jaydev P. Desai et al., Springer International Publishing, 2013, pp. 303–17, doi:10.1007/978-3-319-00065-7_22.

[49] Mur-Artal, Raul, and Juan D. Tardos. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras.” *IEEE Transactions on Robotics*, vol. 33, no. 5, 2017, doi:10.1109/TRO.2017.2705103.

[50] Gao, Xiang, et al. “LDSO: Direct Sparse Odometry with Loop Closure.” *IEEE International Conference on Intelligent Robots and Systems*, 2018, doi:10.1109/IROS.2018.8593376.

[51] Younes, Georges, et al. “Keyframe-Based Monocular SLAM: Design, Survey, and Future Directions.” *Robotics and Autonomous Systems*, vol. 98, 2017, doi:10.1016/j.robot.2017.09.010.

[52] Yang, Nan, et al. “Challenges in Monocular Visual Odometry: Photometric Calibration, Motion Bias, and Rolling Shutter Effect.” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018, doi:10.1109/LRA.2018.2846813.

[53] Cvišić, Igor, et al. “SOFT-SLAM: Computationally Efficient Stereo Visual Simultaneous Localization and Mapping for Autonomous Unmanned Aerial Vehicles.” *Journal of Field Robotics*, vol. 35, no. 4, 2018, doi:10.1002/rob.21762.

[54] Ferrera, Maxime, et al. “OV2SLAM: A Fully Online and Versatile Visual SLAM for Real-Time Applications.” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2021, doi:10.1109/LRA.2021.3058069.

[55] Sumikura, Shinya, et al. “OpenVSLAM: A Versatile Visual SLAM Framework.” *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, 2019, doi:10.1145/3343031.3350539.

[56] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, May 2020, pp. 1689–1696. doi: 10.1109/ICRA40945.2020.9196885.

[57] Campos, Carlos, et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM.” *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021, doi:10.1109/TRO.2021.3075644.

[58] Mur-Artal, Raul, et al. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System.” *IEEE Transactions on Robotics*, vol. 31, no. 5, 2015, doi:10.1109/TRO.2015.2463671.

[59] Gálvez-López, Dorian, and Juan D. Tardós. “Real-Time Loop Detection with Bags of Binary Words.” *IEEE International Conference on Intelligent Robots and Systems*, 2011, doi:10.1109/IROS.2011.6048525.

[60] Engel, Jakob, et al. “Large-Scale Direct SLAM with Stereo Cameras.” *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, 2015, doi:10.1109/IROS.2015.7353631.

[61] Engel, Jakob, et al. “LSD-SLAM: Large-Scale Direct Monocular SLAM.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8690 LNCS, 2014, doi:10.1007/978-3-319-10605-2_54.

[62] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011, doi: 10.1109/ICRA.2011.5979949.

[63] C. H. FITTS, “Graph slam tutorial : the use of g2o,” *The Medical journal of Australia*, vol. 2, no. 26, pp. 901–902, 1952, doi: 10.1038/143391a0.

[64] “Tracking camera T265 – Intel RealSense Depth and Tracking Cameras.” https://www.intelrealsense.com/tracking-camera-t265/, (accessed Feb. 27, 2021).

[65] Stereolabs, “Stereolabs - Capture the World in 3D,” https://www.*stereolabs.com*, (accessed Feb. 27, 2021).

[66] google, “gperftools,” https://www.*gperftools/gperftools*, (accessed Feb. 27, 2021).

[67] “TCMalloc Overview,” *tcmalloc*. https://google.github.io/tcmalloc/overview.html (accessed May 31, 2022).

[68] *Sualeh, Muhammad, and Gon Woo Kim. “Simultaneous Localization and Mapping in the Epoch of Semantics: A Survey.” International Journal of Control, Automation and Systems, vol. 17, no. 3, 2019, doi:10.1007/s12555-018-0130-x.*

[69] Bowman, Sean L., et al. “Probabilistic Data Association for Semantic SLAM.” *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, doi:10.1109/ICRA.2017.7989203.

[70] Schonberger, Johannes L., et al. “Semantic Visual Localization.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, doi:10.1109/CVPR.2018.00721.

[71] Gálvez-López, Dorian, and Juan D. Tardós. “Bags of Binary Words for Fast Place Recognition in Image Sequences.” *IEEE Transactions on Robotics*, vol. 28, no. 5, 2012, doi:10.1109/TRO.2012.2197158.

[72] Kaneko, Masaya, et al. “Mask-SLAM: Robust Feature-Based Monocular SLAM by Masking Using Semantic Segmentation.” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2018-June, 2018, doi:10.1109/CVPRW.2018.00063.

[73] Chen, Liang Chieh, et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.*” IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, 2018, doi:10.1109/TPAMI.2017.2699184.

[74] *Yu, Chao, et al. "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments." IEEE International Conference on Intelligent Robots and Systems, 2018, doi:10.1109/IROS.2018.8593691.*

[75] S. Li, G. Li, L. Wang, and Y. Qin, "SLAM integrated mobile mapping system in complex urban environments," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 316–332, Aug. 2020, doi: 10.1016/j.isprsjprs.2020.05.012.

[76] Casser, Vincent, et al. "Depth Prediction without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos." *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[77] Nicholson, Lachlan, et al. "QuadricSLAM: Dual Quadrics from Object Detections as Landmarks in Object-Oriented SLAM." *IEEE Robotics and Automation Letters*, vol. 4, no. 1, 2019, doi:10.1109/LRA.2018.2866205.

[78] Hosseinzadeh, Mehdi, et al. "Structure Aware SLAM Using Quadrics and Planes." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11363 LNCS, 2019, doi:10.1007/978-3-030-20893-6_26.

[79] Runz, Martin, et al. "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects." *Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2018, 2019, doi:10.1109/ISMAR.2018.00024.

[80] McCormac, John, et al. "Fusion++: Volumetric Object-Level SLAM." *Proceedings - 2018 International Conference on 3D Vision*, 3DV 2018, 2018, doi:10.1109/3DV.2018.00015.

[81] Wang, Ya, and Andreas Zell. “Improving Feature-Based Visual SLAM by Semantics.” *IEEE 3rd International Conference on Image Processing, Applications and Systems*, IPAS 2018, 2018, doi:10.1109/IPAS.2018.8708875.

[82] Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, doi:10.1109/CVPR.2016.91.

[83] Siegwart, Roland, et al. “Introduction to Autonomous Mobile Robots, Second Edition.” *MIT Press*, vol. 23, 2011.

[84] Groves, Paul D. “Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, 2nd Edition [Book Review].” *IEEE Aerospace and Electronic Systems Magazine*, vol. 30, no. 2, 2015, doi:10.1109/maes.2014.14110.

[85] Grubin, Carl. “Derivation of the Quaternion Scheme via the Euler Axis and Angle.” *Journal of Spacecraft and Rockets*, vol. 7, no. 10, 1970, doi:10.2514/3.30149.

[86] Farell, Jay A. “Aided Navigation: GPS with High Rate Sensors.” *CEUR Workshop Proceedings*, vol. 1542, 2015.

[87] El-Diasty, Mohammed, and Spiros Pagiatakis. “Calibration and Stochastic Modelling of Inertial Navigation Sensor Erros.” *Journal of Global Positioning Systems*, vol. 7, no. 2, 2008, doi:10.5081/jgps.7.2.170.

[88] Alarifi, Abdulrahman, et al. “Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances.” *Sensors (Switzerland)*, vol. 16, 2016, doi:10.3390/s16050707.

[89] Silva, Bruno, et al. “Experimental Study of UWB-Based High Precision Localization for Industrial Applications.” *Proceedings - IEEE International Conference on Ultra-Wideband*, 2014, doi:10.1109/ICUWB.2014.6958993.

[90] Commission, Federal Communication. “Revision of Part 15 of the Commission’s Rules Regarding Ultra-Wideband Transmission Systems.” *First Report and Order in ET ...*, no. FCC02-48, 2002.

[91] B. Allen and M. Ghavami, “Editorial: Ultra wideband systems, technologies and applications,” *IEE Proc., Commun.*, vol. 153, no. 1, p. 81, 2006, doi: 10.1049/ip-com:20069003.

[92] Porcino, Domenico, and Walter Hirt. “Ultra-Wideband Radio Technology: Potential and Challenges Ahead.” *IEEE Communications Magazine*, vol. 41, 2003, doi:10.1109/MCOM.2003.1215641.

[93] Geiger, A., et al. “Vision Meets Robotics: The KITTI Dataset.” *International Journal of Robotics Research*, vol. 32, no. 11, 2013, doi:10.1177/0278364913491297.

[94] “ZEB Revo RT - GeoSLAM.” https://geoslam.com/solutions/zeb-revo-rt/ (accessed Feb. 27, 2021).

[95] “MTi 100-series.” https://www.xsens.com/products/mti-100-series (accessed Feb. 27, 2021).

[96] Atia, Mohamed. “Design and Simulation of Sensor Fusion Using Symbolic Engines.” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 25, no. 1, 2019, doi:10.1080/13873954.2019.1566266.

[97] "EVK-8/EVK-M8 | u-blox." https://www.u-blox.com/en/product/evk-8evk-m8 (accessed Feb. 27, 2021).

[98] "MDEK1001 Development Kit - Decawave." https://www.decawave.com/product/mdek1001-deployment-kit/ (accessed Feb. 27, 2021).

[99] Pi, Raspberry. "Raspberry Pi 4 Model B Specifications – Raspberry Pi." *Raspberry Pi Foundation*, 2020.

[100] "NVIDIA Jetson Nano Developer Kit | NVIDIA Developer." https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed Feb. 27, 2021).

[101] "Foundations of Qt Development." Foundations of Qt Development, 2007, doi:10.1007/978-1-4302-0251-6.

[102] Silva, Bruno, et al. "Experimental Study of UWB-Based High Precision Localization for Industrial Applications." *Proceedings - IEEE International Conference on Ultra-Wideband*, 2014, doi:10.1109/ICUWB.2014.6958993.

[103] Embedded and Multisensor Systems Research Group, *Multi-sensor Indoor Positioning/Mapping System for Pedestrian using 3D LiDAR, Stereo Cam, and IMU.*, (Jan. 04, 2021). Accessed: Oct. 07, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=0GBWQKFurn4

[104] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018, doi: 10.1109/TPAMI.2017.2658577.

[105] Wang, Wei, and Dan Wang. "Land Vehicle Navigation Using Odometry/INS/Vision Integrated System." *2008 IEEE International Conference*

on Cybernetics and Intelligent Systems, CIS 2008, 2008, doi:10.1109/ICCIS.2008.4670839.

[106] Olson, Clark F., et al. "Robust Stereo Ego-Motion for Long Distance Navigation." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, doi:10.1109/CVPR.2000.854879.

[107] Mahmood, Azhar, et al. "Real Time Localization of Mobile Robotic Platform via Fusion of Inertial and Visual Navigation System." *2012 International Conference on Robotics and Artificial Intelligence, ICRAI 2012*, 2012, doi:10.1109/ICRAI.2012.6413407.

[108] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation." *Proceedings - IEEE International Conference on Robotics and Automation*, 2007, doi:10.1109/ROBOT.2007.364024.

[109] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *In Pract.*, vol. 7, no. 1, 2006, doi: 10.1.1.117.6808.

[110] Cvišić, Igor, et al. "SOFT-SLAM: Computationally Efficient Stereo Visual Simultaneous Localization and Mapping for Autonomous Unmanned Aerial Vehicles." *Journal of Field Robotics*, vol. 35, no. 4, 2018, doi:10.1002/rob.21762.

[111] Abdelkrim, Nemra, et al. "Robust Nonlinear Filtering for INS/GPS UAV Localization." *2008 Mediterranean Conference on Control and Automation - Conference Proceedings, MED'08*, 2008, doi:10.1109/MED.2008.4602149.

[112] D. Nister, O. Naroditsky and J. Bergen, "Visual odometry," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004. CVPR 2004., *2004, pp. I-I, doi: 10.1109/CVPR.2004.1315094.*

[113] Rublee, Ethan, et al. "ORB: An Efficient Alternative to SIFT or SURF." *Proceedings of the IEEE International Conference on Computer Vision*, 2011, doi:10.1109/ICCV.2011.6126544.

[114] Jabar, Falah, et al. *Object Tracking Using SIFT and KLT Tracker for UAV-Based Applications*. 2016, doi:10.1109/iris.2015.7451588.

[115] Nistér, David. "An Efficient Solution to the Five-Point Relative Pose Problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, 2004, doi:10.1109/TPAMI.2004.17.

[116] Rady, S., et al. "A Hybrid Localization Approach for UAV in GPS Denied Areas." *2011 IEEE/SICE International Symposium on System Integration, SII 2011*, 2011, doi:10.1109/SII.2011.6147631.

[117] M. Kam, Xiaoxun Zhu and P. Kalata, "Sensor fusion for mobile robot navigation," in *Proceedings of the IEEE*, vol. 85, no. 1, pp. 108-119, Jan. 1997, doi: 10.1109/JPROC.1997.554212.

[118] Mehra, Raman K. "On the Identification of Variances and Adaptive Kalman Filtering." *IEEE Transactions on Automatic Control*, vol. AC-15, no. 2, 1970, doi:10.1109/TAC.1970.1099422.

[119] Heffes, H. "The Effect of Erroneous Models on the Kalman Filter Response." *IEEE Transactions on Automatic Control*, vol. 11, no. 3, 1966, doi:10.1109/TAC.1966.1098392.

[120] Zhang, Yinlong, et al. “Robust Vehicle and Surrounding Environment Dynamic Analysis for Assistive Driving Using Visual-Inertial Measurements.” *IEEE Access*, vol. 7, 2019, doi:10.1109/ACCESS.2018.2889320.

[121] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, “Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments,” Orlando, Florida, USA, May 2009, p. 733219. doi: 10.1117/12.819082.

[122] Herweg, Nora A., and Michael J. Kahana. “Spatial Representations in the Human Brain.” *Frontiers in Human Neuroscience*, vol. 12, 2018, doi:10.3389/fnhum.2018.00297.

[123] Ekstrom, Arne D. “Why Vision Is Important to How We Navigate.” *Hippocampus*, vol. 25, 2015, doi:10.1002/hipo.22449.

[124] Widrow, Bernard, and Juan Carlos Aragon. “Cognitive Memory.” *Neural Networks*, vol. 41, 2013, doi:10.1016/j.neunet.2013.01.016.

[125] Burgess, Neil, et al. “A Model of Hippocampal Function.” *Neural Networks*, vol. 7, no. 6–7, 1994, doi:10.1016/S0893-6080(05)80159-5.

[126] Fyhn, Marianne, et al. “Spatial Representation in the Entorhinal Cortex.” *Science*, vol. 305, no. 5688, 2004, doi:10.1126/science.1099901.

[127] Sargolini, Francesca, et al. “Conjunctive Representation of Position, Direction, and Velocity in Entorhinal Cortex.” *Science*, vol. 312, no. 5774, 2006, doi:10.1126/science.1125572.

[128] Morris, R. G. M., et al. “Place Navigation Impaired in Rats with Hippocampal Lesions.” *Nature*, vol. 297, no. 5868, 1982, doi:10.1038/297681a0.

[129] Burgess, Neil, et al. “Predictions Derived from Modelling the Hippocampal Role in Navigation.” *Biological Cybernetics*, vol. 83, no. 3, 2000, doi:10.1007/s004220000172.

[130] Maguire, Eleanor A., et al. “Human Spatial Navigation: Cognitive Maps, Sexual Dimorphism, and Neural Substrates.” *Current Opinion in Neurobiology*, vol. 9, no. 2, 1999, doi:10.1016/S0959-4388(99)80023-3.

[131] Török, Ágoston, et al. “Reference Frames in Virtual Spatial Navigation Are Viewpoint Dependent.” *Frontiers in Human Neuroscience*, vol. 8, no. SEP, 2014, doi:10.3389/fnhum.2014.00646.

[132] Gramann, Klaus, et al. “Evidence of Separable Spatial Representations in a Virtual Navigation Task.” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 31, no. 6, 2005, doi:10.1037/0096-1523.31.6.1199.

[133] Li, Xiaoou, et al. “Retrieving Enduring Spatial Representations after Disorientation.” *Cognition, vol. 124, no. 2, 2012, doi:10.1016/j.cognition.2012.05.006.*

[134] Siegel, Alexander W., and Sheldon H. White. “The Development of Spatial Representations of Large-Scale Environments.” *Advances in Child Development and Behavior*, vol. 10, no. C, 1975, doi:10.1016/S0065-2407(08)60007-5.

[135] Tolman, Edward C. “Cognitive Maps in Rats and Men.” *Psychological Review*, vol. 55, no. 4, 1948, doi:10.1037/h0061626.

[136] McNaughton, B. L., et al. “”Dead Reckoning,” Landmark Learning, and the Sense of Direction: A Neurophysiological and Computational Hypothesis.” *Journal of Cognitive Neuroscience*, vol. 3, 1991, doi:10.1162/jocn.1991.3.2.190.

[137] O'keefe, John, and Lynn Nadel. "Précis of O'Keefe & Nadel's The Hippocampus as a Cognitive Map." *Behavioral and Brain Sciences*, vol. 2, no. 4, 1979, doi:10.1017/S0140525X00063949.

[138] Meilinger, Tobias, and Gottfried Vosgerau. "Putting Egocentric and Allocentric into Perspective." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6222 LNAI, 2010, doi:10.1007/978-3-642-14749-4_19.

[139] Worsley, Claire L., et al. "Path Integration Following Temporal Lobectomy in Humans." *Neuropsychologia*, vol. 39, no. 5, 2001, doi:10.1016/S0028-3932(00)00140-8.

[140] Appleyard, Donald. "Styles and Methods of Structuring a City." *Environment and Behavior*, vol. 2, no. 1, 1970, doi:10.1177/001391657000200106.

[141] Filomena, Gabriele, et al. "A Computational Approach to 'The Image of the City.'" *Cities*, vol. 89, 2019, doi:10.1016/j.cities.2019.01.006.

[142] Zhang, Hui, et al. "Different 'Routes' to a Cognitive Map: Dissociable Forms of Spatial Knowledge Derived from Route and Cartographic Map Learning." *Memory and Cognition*, vol. 42, no. 7, 2014, doi:10.3758/s13421-014-0418-x.

[143] Ekstrom, Arne D., et al. "A Critical Review of the Allocentric Spatial Representation and Its Neural Underpinnings: Toward a Network-Based Perspective." *Frontiers in Human Neuroscience*, vol. 8, no. OCT, 2014, doi:10.3389/fnhum.2014.00803.

[144] Thorndyke, Perry W., and Barbara Hayes-Roth. “Differences in Spatial Knowledge Acquired from Maps and Navigation.” *Cognitive Psychology*, vol. 14, no. 4, 1982, doi:10.1016/0010-0285(82)90019-6.

[145] Rieser, John J. “Access to Knowledge of Spatial Structure at Novel Points of Observation.” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 15, no. 6, 1989, doi:10.1037/0278-7393.15.6.1157.

[146] Shelton, Amy L., and Timothy P. McNamara. “Systems of Spatial Reference in Human Memory.” *Cognitive Psychology*, vol. 43, no. 4, 2001, doi:10.1006/cogp.2001.0758.

[147] Waller, David, and Eric Hodgson. “Transient and Enduring Spatial Representations under Disorientation and Self-Rotation.” *Journal of Experimental Psychology: Learning Memory and Cognition*, vol. 32, no. 4, 2006, doi:10.1037/0278-7393.32.4.867.

[148] Klatzky, Roberta L. “Allocentric and Egocentric Spatial Representations: Definitions, Distinctions, and Interconnections.” *Spatial Cognition: An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, edited by Christian Freksa et al., Springer Berlin Heidelberg, 1998, pp. 1–17, doi:10.1007/3-540-69342-4_1.

[149] L. Richard and D. Waller, “Toward a definition of intrinsic axes: The effect of orthogonality and symmetry on the preferred direction of spatial memory,” *J. Exp. Psychol. Learn. Mem. Cogn.*, vol. 39, no. 6, 2013, doi: 10.1037/a0032995.

[150] Richard, Laurence, and David Waller. “Toward a Definition of Intrinsic Axes: The Effect of Orthogonality and Symmetry on the Preferred Direction of Spatial

Memory." *Journal of Experimental Psychology: Learning Memory and Cognition*, vol. 39, no. 6, 2013, doi:10.1037/a0032995.

[151] Mou, Weimin, et al. "Layout Geometry in the Selection of Intrinsic Frames of Reference from Multiple Viewpoints." *Journal of Experimental Psychology: Learning Memory and Cognition*, vol. 33, no. 1, 2007, doi:10.1037/0278-7393.33.1.145.

[152] Chan, Edgar, et al. "Reference Frames in Allocentric Representations Are Invariant across Static and Active Encoding." *Frontiers in Psychology*, vol. 4, no. AUG, 2013, doi:10.3389/fpsyg.2013.00565.

[153] Frankenstein, Julia, et al. "Is the Map in Our Head Oriented North?" Psychological Science, vol. 23, no. 2, 2012, doi:10.1177/0956797611429467.

[154] Wang, Ranxiao Frances, and Elizabeth S. Spelke. "Updating Egocentric Representations in Human Navigation." *Cognition*, vol. 77, no. 3, 2000, doi:10.1016/S0010-0277(00)00105-0.

[155] Diwadkar, Vaibhav A., and Timothy P. McNamara. "Viewpoint Dependence in Scene Recognition." *Psychological Science*, vol. 8, no. 4, 1997, doi:10.1111/j.1467-9280.1997.tb00442.x.

[156] Holmes, Corinne A., et al. "Multiple Views of Space: Continuous Visual Flow Enhances Small-Scale Spatial Learning." *Journal of Experimental Psychology: Learning Memory and Cognition*, vol. 43, no. 6, 2017, doi:10.1037/xlm0000346.

[157] Mittelstaedt, M. L., and H. Mittelstaedt. "Homing by Path Integration in a Mammal." Naturwissenschaften, vol. 67, no. 11, 1980, doi:10.1007/BF00450672.

[158] Souman, Jan L., et al. “Walking Straight into Circles.” *Current Biology*, vol. 19, no. 18, 2009, doi:10.1016/j.cub.2009.07.053.

[159] Morris, R. G. M., et al. “Allocentric Spatial Learning by Hippocampectomised Rats: A Further Test of the ‘Spatial Mapping’ and ‘Working Memory’ Theories of Hippocampal Function.” *The Quarterly Journal of Experimental Psychology Section B*, vol. 38, no. 4, 1986, doi:10.1080/14640748608402242.

[160] Waller, David, and Yvonne Lippa. “Landmarks as Beacons and Associative Cues: Their Role in Route Learning.” *Memory and Cognition*, vol. 35, no. 5, 2007, doi:10.3758/BF03193465.

[161] Packard, M. G., et al. “Differential Effects of Fornix and Caudate Nucleus Lesions on Two Radial Maze Tasks: Evidence for Multiple Memory Systems.” *Journal of Neuroscience*, vol. 9, no. 5, 1989, doi:10.1523/jneurosci.09-05-01465.1989.

[162] Packard, Mark G., and Barbara J. Knowlton. “Learning and Memory Functions of the Basal Ganglia.” *Annual Review of Neuroscience*, vol. 25, 2002, doi:10.1146/annurev.neuro.25.112701.142937.

[163] White, Norman M., and Robert J. McDonald. “Multiple Parallel Memory Systems in the Brain of the Rat.” *Neurobiology of Learning and Memory*, vol. 77, 2002, doi:10.1006/nlme.2001.4008.

[164] Dasgupta, Saumitro, et al. “DeLay: Robust Spatial Layout Estimation for Cluttered Indoor Scenes.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, doi:10.1109/CVPR.2016.73.

[165] Mathew, Bincy. “Review on Room Layout Estimation from a Single Image.” *International Journal of Engineering Research and Technology*, vol. V9, June 2020, doi:10.17577/IJERTV9IS060820.

[166] Mohan, Narendra, and Manoj Kumar. “Room Layout Estimation in Indoor Environment: A Review.” *Multimedia Tools and Applications*, vol. 81, no. 2, 2022, doi:10.1007/s11042-021-11358-1.

[167] Lee, Chen Yu, et al. “RoomNet: End-to-End Room Layout Estimation.” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, 2017, doi:10.1109/ICCV.2017.521.

[168] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop.” arXiv, Jun. 04, 2016. Accessed: May 27, 2022. [Online]. Available: http://arxiv.org/abs/1506.03365

[169] Coughlan, James M., and A. L. Yuille. “The Manhattan World Assumption: Regularities in Scene Statistics Which Enable Bayesian Inference.” *Advances in Neural Information Processing Systems*, 2001.

[170] Badrinarayanan, Vijay, et al. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, 2017, doi:10.1109/TPAMI.2016.2644615.

[171] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, “Human pose estimation with iterative error feedback,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December. doi: 10.1109/CVPR.2016.512.

[172] Pfister, Tomas, et al. “Flowing ConvNets for Human Pose Estimation in Videos.” *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2015, pp. 1913–21, doi:10.1109/ICCV.2015.222.

[173] Tompson, Jonathan, et al. “Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation.” *Advances in Neural Information Processing Systems*, vol. 2, 2014.

[174] Wu, Jiajun, et al. “Single Image 3D Interpreter Network.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9910 LNCS, 2016, doi:10.1007/978-3-319-46466-4_22.

[175] M. Deitke *et al*., "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3161-3171, doi: 10.1109/CVPR42600.2020.00323.

[176] Ehsani, Kiana, et al. “ManipulaTHOR: A Framework for Visual Object Manipulation.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, doi:10.1109/CVPR46437.2021.00447.

[177] R. Kümmerle *et al.*, “On measuring the accuracy of SLAM algorithms,” *Auton. Robots*, vol. 27, no. 4, pp. 387–407, Nov. 2009, doi: 10.1007/s10514-009-9155-6.

[178] A. Kelly, “Linearized Error Propagation in Odometry,” *Int. J. Robot. Res.*, vol. 23, no. 2, pp. 179–218, Feb. 2004, doi: 10.1177/0278364904041326.

[179] J. Luo, Z. Wang, Y. Chen, M. Wu, and Y. Yang, “An Improved Unscented Particle Filter Approach for Multi-Sensor Fusion Target Tracking,” *Sensors*, vol. 20, no. 23, p. 6842, Nov. 2020, doi: 10.3390/s20236842.

[180] S. Bai, J. Thibault, and D. D. McLean, “Dynamic data reconciliation: Alternative to Kalman filter,” *J. Process Control*, vol. 16, no. 5, pp. 485–498, Jun. 2006, doi: 10.1016/j.jprocont.2005.08.002.

[181] C. Li, F. Zhong, X. Ma, and X. Qin, “Real-Time Head Pose Estimation Based on Kalman Filter and Random Regression Forest,” *J. Comput.-Aided Des. Comput. Graph.*, vol. 29, no. 12, p. 2309, 2017, doi: 10.3724/SP.J.1089.2017.16521.