2023-10-04

Shaped-based IMU/Camera Tightly Coupled Object-level SLAM using Rao-Blackwellized Particle Filtering

Ilyar Asl Sabbaghian Hokmabadi

UNIVERSITY OF CALGARY

Shaped-based IMU/Camera Tightly Coupled Object-level SLAM using Rao-Blackwellized Particle Filtering

by

Ilyar Asl Sabbaghian Hokmabadi

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN GEOMATICS ENGINEERING

CALGARY, ALBERTA

OCTOBER, 2023

Abstract

Simultaneous Localization and Mapping (SLAM) is a decades-old problem. The classical solution to this problem utilizes entities such as feature points that cannot facilitate the interactions between a robot and its environment (e.g., grabbing objects). Recent advances in deep learning have paved the way to accurately detect objects in the image under various illumination conditions and occlusions. This led to the emergence of object-level solutions to the SLAM problem.

Current object-level methods depend on an initial solution using classical approaches and assume that errors are Gaussian. This research develops a standalone solution to object-level SLAM that integrates the data from a monocular camera and an IMU (available in low-end devices) using Rao Blackwellized Particle Filter (RBPF). RBPF does not assume Gaussian distribution for the error; thus, it can handle a variety of scenarios (such as when a symmetrical object with pose ambiguities is encountered). The developed method utilizes shape instead of texture; therefore, texture-less objects can be incorporated into the solution. In the particle weighing process, a new method is developed that utilizes the Intersection over the Union (IoU) area of the observed and projected boundaries of the object that does not require point-to-point correspondence. Thus, it is not prone to false data correspondences. Landmark initialization is another important challenge for object-level SLAM. In the state-of-the-art delayed initialization, the trajectory estimation only relies on the motion model provided by IMU mechanization (during the initialization), leading to large errors. In this thesis, two novel undelayed initializations are developed. One relies only on a monocular camera and IMU, and the other utilizes an ultrasonic rangefinder as well.

The developed object-level SLAM is tested using wheeled robots and handheld devices, and an error (in the position) of 4.1 to 13.1 cm (0.005 to 0.028 of the total path length) has been obtained through extensive experiments using only a single object. These experiments are conducted in different indoor

environments under different conditions (e.g. illumination). Further, it is shown that undelayed initialization using an ultrasonic sensor can reduce the algorithm's runtime by half.

Acknowledgements

This thesis is the fruit of many discussions with many people. While it is very difficult to quantify the individual contributions to developed ideas in this thesis, I would like to acknowledge the people who had the most significant impact on shaping my journey during this research.

I would like to express my thanks, first and foremost, to my supervisor, Dr. El-Sheimy. Without his guidance throughout this time, I would never be able to accomplish what I had in this thesis. A paragraph or two cannot suffice to contain how much I am indebted to his patience with me through this time.

Next, I would like to thank my cousin, Sina Taghvakish. During my Ph.D. and especially during the pandemic era, I had many conservations with him regarding my thesis. It was through these iterative rounds of discussion that some of the trajectories I took in this research became clear to me. I would only hope one day I will be able to return the favour he has provided me, if not in whole but in parts.

I want to thank my dear Dulcie Foofat. Her company was perhaps not the most technical, but it helped me grow as a person during the last three years. She also helped me a lot in editing this thesis. Perhaps most importantly, she helped me understand that I should be able to explain my ideas in simple terms to the general audience, no matter how difficult they might be.

I would like to thank my friend Puya Latafat. He has extended his kind helping hand many times during this period. I want to thank him for answering all my questions about writing the thesis, copyright and other stuff. Alas, I am also regretful for not listening to him in some respects, which I should have.

I would like to thank the staff in the library of the University of Calgary. Throughout time, winter or summer, night or day, I have been a resident of their place. I only hope that this place stays as helpful as it is now for the students in the future with their research.

I would like to thank all my other friends in MMSS, including my dear colleague, friend and co-author Mengchi Ai.

Dedication

“I dedicate this dissertation to friends and family.”

Table of Contents

List of Tables

List of Figures and Illustrations

List of Abbreviations

Abbreviation	Definition
2D	Two dimensional
3D	Three dimensional
6DoF	Six Degrees of Freedom
CAD	Computer-Aided Design
cm	Centimeter
CMOS	Complementary Metal-Oxide Semiconductor
DBN	Dynamic Bayesian Network
DL	Deep Learning
DNN	Deep Neural Network
EKF	Extended Kalman Filter
EV	Expected Value
FG	Factor Graph
GBA	Global Bundle Adjustment
GNSS	Global Navigation Satellite Systems
HTTPS	Hypertext Transfer Protocol Secure
HVP	Horizontal Vanishing Point
Hz	Hertz
IBOB	Infrared Bearing-Only Beacons
IC	Integrated Circuit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit

IoU	Intersection over Union
IR	Infrared
k	Kilo
Kd-tree	K-dimensional tree
KF	Kalman Filter
LBA	Local Bundle Adjustment
LED	Light Emitting Diode
LiDAR	Light Detection and Ranging
LoS	Line of Sight
MAP	Maximum A Posteriori
MEMS	Micro-Electrical Mechanical System
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
MW	Manhattan World
MWC	Manhattan World Constraint
NNS	Nearest Neighbour Search
PF	Particle Filter
R/G/B	Red, Green, Blue
RANSAC	Random Sampling Consensus
RBPF	Rao-Blackwellized Particle Filter
RPV2	Raspberry Pi Camera Module V2
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transformation

SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
UART	Universal Asynchronous Receiver Transmitter
UTC	Coordinated Universal Time
VP	Vanishing Point
VVP	Vertical Vanishing Point

List of Symbols

Symbol	Definitions
u	Odometry inputs
x	Robot's state (position, velocity and orientation)
z	Observations
m	Landmark state (position and orientation)
a	Data association
$\hat{a}$	Data association (maximum likelihood estimate)
w	Particle weight
R_e^i	Rotation from the earth to the inertial frame
R_b^i	Rotation from the body to the inertial frame
Ω_{ib}^i	Skew-symmetric matrix of angular rates in the inertial frame
ω	Angular rates
f^b	Accelerometer readings in the body frame
f^i	Accelerometer readings in the inertial frame
v_k^i	Velocity at epoch k (in the inertial frame)
$\|\theta\|$	Magnitude of angular change
r_{k+1}^i	Position of the robot at epoch $k+1$ (in the inertial frame)
H_b^i	Homogenous transformation from the body to the inertial frame
$\widetilde{H}_c^i$	Homogenous transformation from the camera to the inertial frame (predicted)
H_i^o	Homogenous transformation from the inertial to the object's frame
$\widetilde{H}_c^o$	Homogenous transformation from the camera to the object frame (predicted)

$e_{k+1}^{[n]}$	Error in the position of particle n at epoch $k+1$
$\Sigma_{\Delta e}$	The covariance matrix of the errors (Δe)
$IoU(\tilde{z}_{k+1}^{[n]}, z_{k+1})$	The intersection over the union of the observed and predicted contours
P	The coordinate center of the robot's body frame
l	The vector from the coordinate center (P) to the center of the wheel
ξ^b	The robot's state vector in the body frame
α	The angle between vector l and the x-axis of the robot's body frame
β	The angle between vector l and the perpendicular vector to the axis of a wheel
D	Wheel diameter
$\dot{\varphi}$	The angular velocity of the wheel
A_c	The state matrix
B_c	The input matrix
ψ	The beam angle of an ultrasonic sensor
s	The radius of uncertainty region of an ultrasonic distance measurement
$c(.)$	B-spline curve
B_n	B-spline basis
c_n	Control points (of a B-spline curve)
N_B	The number of basis (of a B-spline curve)
Q	B-spline coefficients
$B(s)$	B-spline basis matrix
q	Quadrants
T_{IoU}	The sum of IoU of the shapes in the quadrant

q^*	The quadrant with the highest T_{IoU}
S^r	Ranked shape list
S^c	Tested shape list
κ	The curvature of a point
p	Pixel coordinates
K	Intrinsic calibration matrix
v_j^c	Vector corresponding to a vanishing point
C^b	The covariance matrix of the accelerometer readings
C^r	The covariance matrix of the rotation parameters
$\hat{P}$	Predicted position of a projected pixel
γ	Angle between a ray and z-axis of the map
α	The distance from the camera's center to a point on the floor
R_c^m	Rotation from camera to map frame
$\tilde{P}$	The estimated position of the back-projected points in the world frame
$e(\tilde{P},P)$	The error between the projected and the predicted points
F	Fundamental matrix
k_1, k_2	The radial distortion coefficients
o_x, o_y	The coordinates of the principal point of a camera

Chapter 1: Introduction

1.1 Object-level Simultaneous Localization and Mapping (SLAM)

Localization and mapping are among the most important aspects of successful navigation and mission planning for an autonomous agent. The known map of an environment helps an autonomous agent localize itself, while conversely, the known trajectory of an autonomous agent helps to build the map of an environment. Under certain circumstances, an agent's location and the map of an environment are not known simultaneously. This problem is known as Simultaneous Localization and Mapping (SLAM). Solving the SLAM problem is very important in many areas, such as autonomous industry (Badue et al., 2020), planetary exploration (Matthies et al., 2007) and search and rescue missions (Calais et al., 2007). Figure 1.1 illustrates some of the devices and robots where it is important to address the SLAM problem.

In the past, geometrical entities such as points were often used to successfully estimate the trajectory of an autonomous agent while simultaneously building sparse (Mur-Artal et al., 2015) or dense maps (Newcombe et al., 2011). More recently, there has been ongoing research in object-level SLAM (Nicholson et al., 2019; Qian et al., 2020). Figure 1.2 compares these three classes of solutions to address SLAM problem. Object-level mapping can facilitate the tasks that require the interaction of a robot with the environment. Such tasks can include picking up and placing objects from one location to another. Object-level localization also has certain advantages over the classical sparse and dense SLAM methods. For example, tracking a single object with 6 Degrees of Freedom (6DoF) is sufficient to localize a camera (Salas-Moreno et al., 2013). Further objects are more distinctive than simpler geometrical abstractions (Choudhary et al., 2014), which reduces the possibility of false data association. However, modelling, identifying, and tracking arbitrary objects in the images requires more sophisticated techniques than points, lines, and other simpler geometrical forms. Therefore, classical solutions to the SLAM problem might be inadequate for the objects level SLAM.

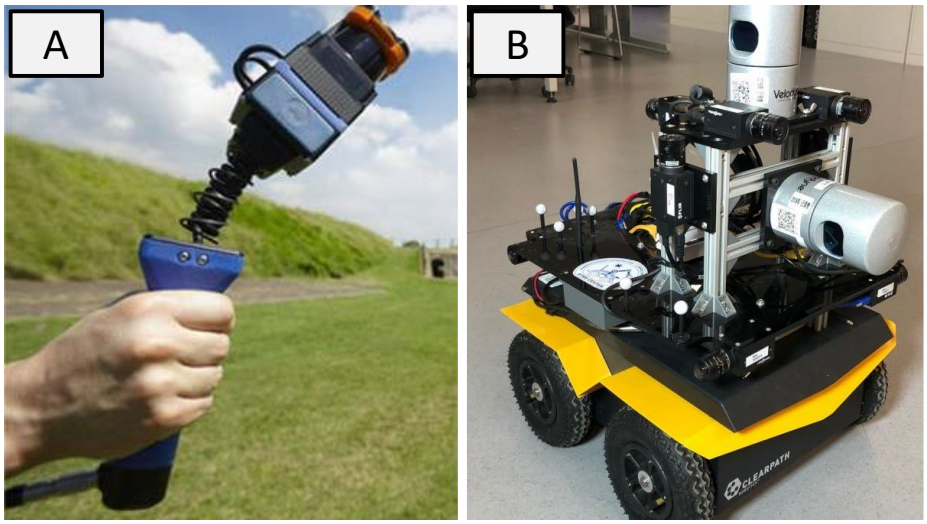


Figure 1.1: In this figure, two platforms that should solve the SLAM problem are shown. Figure A shows a handheld portable mapping device courtesy of (CSIRO research group, January 2013, Zebedee Fort Lytton, accessed May 2023). Figure B shows a robot designed to map indoor environments (image reference: (Chen et al. 2020)).

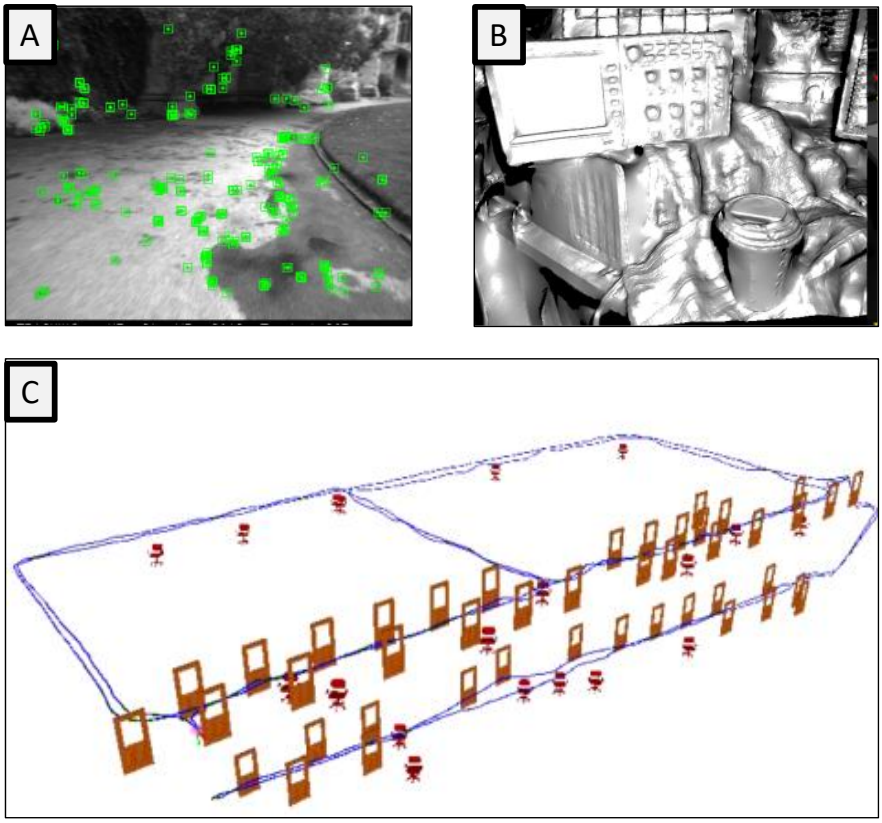


Figure 1.2: Solutions to the SLAM can be categorized into three major groups: Figure A shows sparse-SLAM (image reference:(Mur-Artal & Tardós, 2017)) where features (shown in green) are identified sparsely. Figure B shows dense SLAM (image reference: (Newcombe et al., 2011)) where the environment is reconstructed densely. Figure C shows object level-SLAM (image reference: (Bowman et al., 2017)) where building the map of the environment and estimating the robot's trajectory is achieved with the help of objects such as doors and chairs.

1.2 Current Gap and Research Motivations (Overview)

A solution to object-level SLAM includes many components that are integrated into a mapping and localization framework. Each component addresses one aspect of the problem of object-level SLAM. In this section, some of the most important components are introduced:

1. **Object detection/segmentation** refers to an algorithm or a set of algorithms used to identify the position of the objects in the image (or in a 3D scan). Object detection often refers to identifying a bounding box around the object of interest, while object segmentation requires pixel-level identification. Object segmentation/detection can be achieved using classical computer vision algorithms such as template matching (Bolme et al., 2009) and more advanced and modern approaches using Deep Learning (DL) (He et al., 2017). Object detection/segmentation must be robust to the variations in the illumination conditions in the environment, the sensor's viewpoint, and the sensor's intrinsic properties. Section 1.2.2 provides the current research gaps and motivation for the developed DL-based object segmentation.
2. **Object representation** refers to the stored a-priori information of the object's shape or appearance, and it is important for other components of an object-level SLAM (e.g., pose estimation). In the past, objects are represented using voxel grids (Gouiaa & Meunier, 2014), 3D features clusters (Gálvez-López et al., 2016), and more. Object representation should capture the object with sufficient details to be useful for object-level SLAM. Section 1.2.3 provides an overview of the current research gaps and the developed class of object representation.
3. **Object pose estimation** refers to estimating the object's orientation and translation in the camera's (or any other sensor's) coordinate frame. The objects detected/segmented in the images must be inserted into the map if they have been observed for the first time. The process is known as initialization, and to achieve it, the object's pose with respect to the camera should be estimated.

Therefore, pose estimation is another important component of an object-level SLAM. Pose estimation should be robust to errors occurring in object segmentation. Section 1.2.4 introduces the research motivations leading to the developed contour-based object-pose estimation.

4. Finally, the three abovementioned components should be integrated into an **object-level localization and mapping framework**. In the past, mapping and localization were achieved using multiple stages where often an initial solution is obtained with the help of Dynamic Bayes Networks (DBN) (e.g., Extended Kalman Filtering (EKF)), and later, it is optimized as more observations become available. Object-level mapping and localization framework refer to the process of estimating the poses of the sensor and the objects in an environment. A solution should be able to handle the situations where the errors cannot be approximated using Gaussian distributions, and the observations model is non-linear. Section 1.2.1 explains the challenges these approaches face and will briefly explain the developed solution.

Solutions to object-level SLAM can be obtained by fusing the observation from many different sensors. Data fusion is the process of achieving such a collaborative solution. With the help of data fusion, individual advantages of the sensors such as monocular cameras, Inertial Measurement Units (IMU) and rangefinders can be utilized to improve the accuracy of the solution to object-level SLAM. Section 1.3 will provide the motivations and an overview of the developed solution for data fusion.

1.2.1 Object-level Mapping and Localization Frameworks

Classical solutions to SLAM have been addressed using many different methods. DBN is amongst the most utilized solutions. A DBN includes two important stages: prediction and update. In the prediction stage, the robot or handheld device's pose is estimated with the help of a motion model. In the update stage, the observation model is used to correct the predictions and obtain a final estimate of the robot's pose at a given epoch. This process of predicting and updating is repeated as new observations become

available. Unfortunately, most utilized DBNs, such as EKF, require the parameters to have a Gaussian distribution. However, this restriction leads to challenges in object-level SLAM. For example, it is common to encounter symmetrical objects (objects with the same silhouettes or appearance in the images from different viewpoints) during the navigation. This leads to uncertainties in the object's pose that cannot be accurately approximated using a Gaussian distribution. Due to such a source of errors, it is more suitable to use DBN methods such as Particle Filtering (PF), which does not assume Gaussian distribution.

In this thesis, a PF-based solution for object-level SLAM is developed that relies on the fusion of IMU and monocular camera. This solution uses IMU mechanization to estimate the particle trajectories in a short time. The particle weights are updated using images from a monocular camera. This developed method uses a novel object initialization method. With the help of this initialization approach, objects are inserted with a pose uncertainty in the map once they are observed. Such an undelayed initialization allows updates (particle weighing and resampling) immediately. In contrast, the current state-of-the-art solutions are based on delayed initialization, where objects are only inserted into a map after being observed in many images. Such an approach can only rely on IMU mechanization during the initialization period and thus can suffer from the accumulation of errors in trajectory estimation.

Two different fusion methods of tightly and loosely coupled are developed in this thesis. Both approaches only rely on the shape of the object rather than texture (defined as variations in the colours or intensity of the object in an image). Thus, objects without texture can also be integrated into the solution. Developed tightly and loosely coupled methods have many significant differences. One such difference is that while the loosely coupled method depends on the pose estimation component, the tightly coupled method does not. The tightly coupled method is more advantageous since pose estimation can be erroneous (e.g. if the object is severely occluded). Detailed literature on object-level SLAM is provided in Section 2.5. The developed tightly and loosely coupled object-level solution is explained in Chapter 3.

1.2.2 Object Segmentation/detection

Object segmentation/detection is an important step of many solutions to object-level SLAM. In this thesis, it is the first step of pose estimation, initialization, and particle weighting processes. In the past, objects were segmented/detected using different approaches. The objects are often detected using feature points (Gálvez-López et al., 2016). However, detecting such features requires texture on the object's surface. Further, classical feature detectors exhibit low robustness to variations in the illumination conditions and camera viewpoints changes (which can cause severe perspective distortions). The presence of cluttered backgrounds and occlusions is yet another challenge that complicates the task of object segmentation. Therefore, the segmentation should be robust to such distortions and sources of error.

Recent advances in DL have provided novel opportunities for many problems, including SLAM. Unfortunately, training DL networks requires a massive amount of input data. In order to address the challenges associated with the massive input data requirements (and manual output labelling), synthesized data sets have been utilized in the past (Tremblay et al., 2018). This thesis uses a novel method based on a hybrid technique (Georgakis et al. 2017) to synthesize a large set of training images and the output masks in a short time. These synthesized images are used to train an object-segmentation DL network successfully. Section 2.2 investigates classical and more modern DL-based object detection/segmentation methods in detail. Further, different approaches for synthesizing training images are investigated in section 2.2. Section 4.2 will introduce the developed segmentation approach.

1.2.3 Object Representation

Geometrical entities such as points and lines have a simple parametric representation. Semantic Objects, in general, need a much larger number of parameters to be defined. Objects in the past were represented using simpler geometrical shapes (e.g., ellipsoids (Nicholson et al., 2019), cuboids (Yang & Scherer, 2019)) or detailed 3D models (Gálvez-López et al., 2016; Salas-Moreno et al., 2013). The

object's representation should account for possible sources of distortions in the appearance and the shape of objects due to variations in the camera's viewpoints and illumination conditions. Due to these challenges, in this thesis, an object representation is developed that can be matched directly to the segmented contour of an object in the image under the conditions mentioned above. In order to achieve this, the object of interest is represented as a set of 2D contours denoted as the shape-prior set. Finally, a fast-matching method is developed to find the closest shapes in the shape-prior set to the segmented object in the image. Section 2.3 briefly reviews different approaches for object representation and the current research gap. Section 4.3 will, in detail, explain the developed object representation using shape-prior sets.

1.2.4 Object Pose Estimation

Once the objects are segmented in the image and prior information about the shape is available (object representation), the next step is to find the relative pose of the object with respect to the camera. The capability of estimating the camera's pose (and trajectory) using only one object is one of the advantages of object-level SLAM over the classical sparse and dense SLAMs. The current pose-detection methods can be categorized into two groups. The first approach jointly solves object segmentation and pose estimation using an end-to-end Deep Neural Network (DNN). End-to-end deep pose estimation requires large training image sets (Sundermeyer et al., 2018). In contrast to the end-to-end method, a second paradigm is a two-step approach where in the first step, features are detected (using classical computer-vision methods or DL), and in the second step, the pose is estimated with the help of these features. In order to estimate the pose, a correspondence between the features in the image and the 3D model of the object is established. The most ubiquitous technique for such pose estimation is the Perspective-n-Points (PnP) algorithms which require the detection of salient features on the object. Such feature detection relies on the existence of the texture on the object. Further, the detection and matching are not completely robust to the influence of photometric and geometrical distortions (e.g., variations in

the scale). Finally, object segmentation usually does not directly provide features, and additional steps of feature detection should be included.

In order to address the abovementioned issues, a novel course-to-fine pose estimation method based on contours (and not the texture) is proposed in this thesis. The method relies on initially finding the object's pose by matching the contour of the segmented object to the shape-prior set. With the help of the initial pose estimation and the 3D model of the object, a PnP algorithm is utilized to refine the pose. Section 2.4 will introduce the state-of-the-art pose estimation algorithms and highlight the importance of developing a contour-based pose estimation rather than a texture-based approach. Sections 4.3 and 4.4 will provide the details of this coarse-to-fine pose estimation.

1.3 Data Fusion Towards Object-level SLAM

SLAM has been addressed in the past using many sensors. Single-sensor solutions often focus on reducing costs while trying to achieve a highly accurate solution. The multi-sensor solutions attempt to leverage a wider array of sensors to improve the accuracy of the solution. Currently, many types of sensors are utilized to solve the SLAM problem. Some of these sensors are designed only to sense manmade signals. These include Global Navigation Satellite Systems (GNSS), Bluetooth, Wi-Fi, and infrared receivers. On the other hand, sensors such as a monocular camera can detect manmade (a lamp light) and natural signals (sunlight). In a different categorization, the utilized sensors in the navigation can also be divided based on whether they are active or passive. Active sensing systems such as Light Detection and Ranging (LiDAR) send and receive the returning signal from the environment. In contrast, passive sensors such as an IMU detect the signals due to the motion of the robot (or a device) itself. These sensors are passive because they do not send signals into the environment. Since most sensors are not suffering from the same type of error, data fusion can help mitigate the individual errors of sensors and contribute to increasing the overall accuracy and robustness of the solution to the SLAM.

Among many sensors used to address the SLAM, cameras are one of the most important sensors in object segmentation since they capture rich radiometric information (e.g., colour). However, estimating the trajectory of the device's motion using only a camera is challenging. Therefore, it seems that the fusion of a camera with a sensor capable of estimating the motion more accurately (in a short period of time), such as an IMU or wheel odometer, would be advantageous. IMUs and cameras have been fused in the past for the purpose of trajectory estimation (Mourikis & Roumeliotis, 2007). In the context of object-level SLAM, a possible approach is to utilize IMU to predict the motion while utilizing segmented objects in the images to update the estimation (e.g., weighting and resampling of the particles). The developed IMU/ camera fusion in this thesis is based on such a paradigm (explained in Chapter 3).

A requirement for the developed fusion technique is to find the extrinsic calibration parameters between the two sensors. Extrinsic calibration refers to estimating two sensors' relative orientation and translation parameters. The estimation of the orientation parameters is known as boresight calibration, and the estimation of the translation parameters is known as lever arm calibration. In this thesis, the extrinsic calibration term refers to the scenario where both of these parameters are calibrated. The main challenge with most of the extrinsic calibration methods proposed in the past is that they depend either on ground control points (Pinto et al., 2002), accurate instruments such as a turntable (Lobo & Dias, 2007) or require an initial solution (Furgale et al., 2013). Accessing such instruments or an initial solution might not be possible in many scenarios. In this thesis, a novel extrinsic calibration method using DL-based floor segmentation in the indoor environment is developed. The method offers similar accuracy to the state of the art. A detailed review of the current methods for extrinsic calibration of IMU and camera is provided in Section 2.6. The developed method is explained in Section 5.3.

A second challenge for camera-based SLAM is the problem of landmark initialization. As mentioned, initialization is the process of inserting a landmark into the map. Due to the lack of scale in

the monocular camera observations, delayed initialization (Bailey, 2003; Davison et al., 2007) is used (which requires observing objects from multiple viewpoints). In the undelayed initialization (Solà et al. 2005), landmarks are immediately inserted into the map. If the distance to the object is unknown, the landmark must be initialized with larger uncertainty. See Figure 1.3, where this is explained schematically. One possible solution to address scale ambiguity is to use stereo image-pair. However, stereo image pair processing includes a number of steps with an overall high computational cost. These steps can include rectification of the images, feature detection/matching, and disparity estimation. Most software-based implementations of these processing steps are unsuitable for SLAM in real time (Lazaros et al., 2008). A second solution to resolve the scale ambiguity is a data fusion of a camera with a rangefinder, which can resolve the scale ambiguities related to the monocular camera and can help achieve undelayed initialization with smaller uncertainty. A drawback of such solutions is that projection of the point cloud (obtained by the rangefinder) onto the image often leads to a sparse set of points, where many pixels are left without any assigned depth (especially for 2D LiDAR and ultrasonic rangefinders, where they can provide only sparse point clouds). However, an important distinction between the classical and object-level SLAM problem is that unlike primitive geometrical points or abstract feature points (ORB points used in ORB-SLAM (Mur-Artal & Tardos, 2016)), objects are volumetric bodies and can occupy hundreds of pixels in an image. Therefore, low angular resolution rangefinders can still provide distance to some points on these objects. The approximate initialization obtained from these distance readings can be refined in the process of PF-based SLAM as more information becomes available. This is explained in detail in Section 3.6.

As with IMU/camera fusion, the extrinsic calibration parameters between an ultrasonic sensor and a camera should be estimated. These values can be obtained using Computer-Aided Design (CAD) models. However, better calibration is required in most scenarios. The developed method in this thesis

utilizes 2D line-segment maps built using an ultrasonic sensor (Abadi & El-Sheimy, 2022) and matches them to pixels corresponding to the boundary of the floor and the walls in the environment. Similar to the extrinsic calibration technique of IMU and monocular cameras, a DL-based floor segmentation approach is used to find the floor segment boundary in the image. Details about this method are provided in Section 5.4.

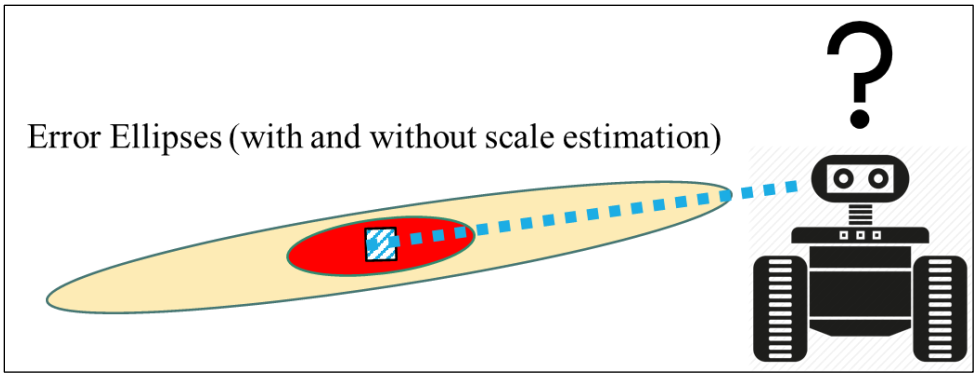


Figure 1.3: The initialization is a challenging task due to the scale's ambiguity using a monocular camera.

1.4 Summary of the Objectives

One of the key challenges of the state-of-the-art object-level solution to the SLAM is the assumption that the uncertainties in the pose of the object and the camera can be represented using a Gaussian distribution. The second challenge is that most object-level solutions do not take advantage of an accurate motion estimation. Further, object-level solutions depend on the detection of salient feature points on the object's surface; however, many objects lack such features. In order to address these challenges and limitations, in this thesis, a novel object-level PF-based solution using IMU and monocular camera fusion is developed. In the following, details of the objectives regarding the proposed solution are explained:

1. This thesis develops and investigates a tightly and loosely coupled fusion of IMU/monocular cameras. While in a tightly coupled solution, the particle's weight is updated directly by evaluating the observation likelihood, in the loosely coupled solution, the particle's weight is updated after

the pose of the device is estimated independently using each sensor (IMU and camera). The two implementations are compared to each other in terms of error in the trajectory estimation and the algorithm's runtime.

2. Standalone trajectory estimation using IMU mechanization can lead to a quick accumulation of errors if no measurement updates are available from other sensors (such as a camera). In this thesis, a novel undelayed initialization of the objects in the map is developed to address this problem. The developed method estimates the object pose up to an unknown scale using a single image. The initialized object can be subsequently used to provide updates to the particle filter. The advantage of such an approach is investigated.
3. The uncertainty in the pose of an object in the initialization is very large. Such a larger uncertainty requires a large number of particles. In order to reduce the computations, a possible solution is a fusion with a rangefinder. In order to achieve this, a camera /ultrasonic sensor fusion is developed in this thesis. Due to the low angular resolution of the ultrasonic sensor, the objective is to identify if the observations are returned from the foreground (the object of interest) or from the background (false positives). Further, improvement to the runtime and the accuracy of the solution with this fusion should be investigated.
4. As most objects lack texture, relying on the shape for the process object-level solution is important. In the thesis, the possibility of implementing an object-level SLAM using shape is investigated. In particular, the objective is to assess the robustness and the accuracy of the shape-based pose estimation under different conditions (such as illumination conditions and the distance of the camera from the object).
5. Object segmentation is an important step in the developed particle filter-based method. Training a DL-based segmentation requires a large image set. In this thesis, a hybrid method for synthesizing

images is utilized to produce a training set. The objective is to investigate the precision and recall of this segmentation method.

1.5 Summary of the Contributions

1.5.1 Major Contributions

1. The developed shape-based tightly and loosely coupled solutions are tested under different conditions (e.g., illumination of the scenes, severe presence of the occlusion). The results indicate that the developed method exhibits robustness to many sources of distortions. The tightly coupled solution can achieve an error of about 4.1 to 13.1 cm (0.005 to 0.028 of the total path length), and the loosely coupled method can achieve an error of 11.5 to 170.9 cm (0.024 to 0.426 of the total path length). Therefore, the tightly coupled method provides a more accurate solution. However, the loosely coupled method achieved lower runtime. The number of particles needed to obtain such accuracy is between 5,000 to 12,500 particles, and increasing the number above these limits did not achieve higher accuracy.
2. It is shown that the undelayed initialization can provide an immediate update possibility in the filtering process. Thus, this greatly reduces the possibility of error accumulation. Further, it is concluded that the initial uncertainty can be reduced significantly after one or two observations.
3. Camera/ultrasonic sensor fusion demonstrates that the algorithm's runtime can be reduced greatly (approximately 50%) with additional observation from an ultrasonic sensor. This is because distance measurement leads to the elimination of many particles without a requirement to evaluate the observation likelihood directly. Further, it is shown that it is possible to detect if the distance measurement is returned from the object of interest with approximately 100% precision and 80% recall. This is achieved without an extensive requirement for hyperparameter tuning.

4. In order to estimate an object's pose, a novel object representation and matching using a shape-prior set is developed. The segmented object in the images is matched to the shape prior to estimate the pose. This method can provide a list of possible poses for objects that are symmetrical, and an average error of 9.96° to 10.31° is achieved for the pose estimation.
5. A novel DL-based object segmentation is developed. The training image set is built using a hybrid approach with synthesized images, achieving a precision of 94% and recall of over 85% in different indoor scenes.

1.5.2. Minor Contributions

1. The IMU/camera boresight calibration achieved an error of 3.49° using 141 images. This result outperforms Kalibr, which is the standard benchmark IMU/Camera calibration in many studies.
2. The ultrasonic/camera boresight calibration achieves an error of approximately 1°.

Figure 1.4 illustrates an overview of the developed system and its contributions. The green boxes show the major components and the modules of the developed tightly coupled and loosely coupled methods. The blue arrow indicates that pose estimation is required for the loosely coupled each time an object is observed. For the tightly coupled method, however, pose estimation is only required once and in the initialization of the object in the map. A summary of the information and the advantages of the developed components are shown inside gray boxes. Finally, the extrinsic calibration (shown as the red box) is often only required to be performed once before each experiment. The developed solution is implemented on two systems. The first system is a handheld device that includes a monocular camera, IMU, and ultrasonic rangefinder. The second platform is a wheeled robot designed for the indoor environment. Instead of an IMU, this robot includes wheel odometry to predict the motion.

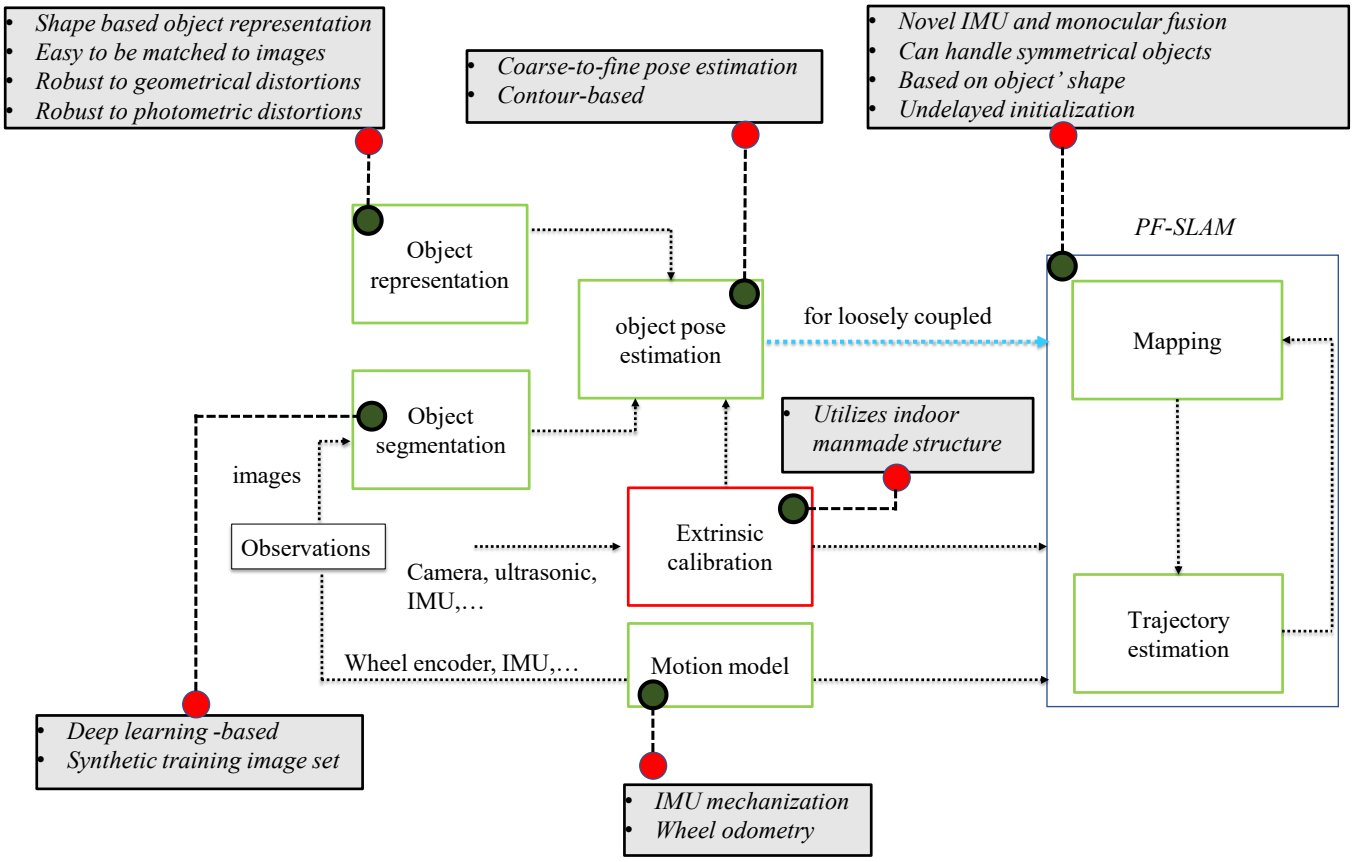


Figure 1.4: An overview of the developed tightly and loosely coupled object-level SLAM.

1.6 Chapter Organization

The remaining chapters of this dissertation are organized as follows. In Chapter 2, a literature review is provided. The state-of-the-art solutions for the SLAM are reviewed in detail with a focus on the individual components utilized in addressing the object segmentation, object representation, object-based pose estimation, as well as the overall mapping and localization frameworks.

Chapters 3 to 5 are dedicated to introducing the methodology of developed object-level SLAM. The framework of the developed PF-based solution using the fusion of IMU and monocular camera in two loosely and tightly coupled fashions is explained in Chapter 3. Chapter 4 introduces the developed crucial components to perform the object SLAM, including object segmentation and pose estimations. Chapter 5 explains the extrinsic calibration of the monocular camera and IMU, as well as the monocular camera and

ultrasonic rangefinder. The details of the experiments are provided in Chapter 6, which includes the platforms used to perform the experiments and how ground-truth values for the parameters of interest are estimated. Chapter 7 provides the results. The developed methods are assessed using many different metrics. These include the pose estimation accuracy, the reprojection error, the algorithm's runtime and more. The results are all conducted using real datasets and in many indoor environments. Finally, Chapter 8 concludes this thesis and provides directions for future research work in this newly emerging study area.

List of publications:

- Abadi, Ilyar, and Naser El-Sheimy. "Manhattan World Constraint for Indoor Line-based Mapping Using Ultrasonic Scans." In 2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-8. IEEE, 2022.
- Asl Sabbaghian Hokmabadi, I., and N. El-Sheimy. "Probabilistic Silhouette-Based Close-Range Photogrammetry Using a Novel 3d Occupancy-Based Reconstruction." The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 43 (2022): 343-350.
- Hokmabadi, Ilyar Asl Sabbaghian, Mengchi Ai, Chrysostomos Minaretzis, Michael Sideris, and Naser El-Sheimy. "Accurate and Scalable Contour-based Camera Pose Estimation Using Deep Learning with Synthetic Data." In 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 1385-1393. IEEE, 2023.
- Asl Sabbaghian Hokmabadi, Ilyar, Mengchi Ai, and Naser El-Sheimy. "Shaped-Based Tightly Coupled IMU/Camera Object-Level SLAM." Sensors 23, no. 18 (2023): 7958.
- Hokmabadi, Ilyar Asl Sabbaghian, Mengchi Ai, Naser El-Sheimy. " Deep Learning based Online Extrinsic Calibration of IMU and Monocular Camera Using Handheld Devices" IEEE Access, July (2023).(Review Process)

- Hokmabadi, Ilyar Asl Sabbaghian, Mengchi Ai, Naser El-Sheimy. " Low-cost and Accurate Infrared-based Angle of Arrival Beacons for Three Degrees of Freedom Mobile Robot Pose Estimation" in IEEE Transactions on Instrumentation and Measurement, July. 2023. (Review Process)
- Hokmabadi, Ilyar Asl Sabbaghian, Mengchi Ai, Naser El-Sheimy. " Shaped-based Loosely-Coupled IMU/Camera Object-level SLAM." July, Sensors (2023). (Review Process)

Chapter 2: A Review of the State of the Object-level SLAM

2.1 Overview

This chapter includes the main literature review and investigates the state-of-the-art solutions for object-level SLAM. These solutions can be categorized more specifically based on the type of method used in object detection/segmentation, object representation and object pose estimation components. The relevant literature is provided in Sections 2.2 to 2.4. Object-level solutions to SLAM problems in the past can also be categorized based on the method used to integrate these three components into a mapping and localization framework, which is covered in Section 2.5. In reviewing each topic, the focus was on the literature that implements an object-level solution. However, influential research works in the study areas of object segmentation, object representation, and object pose estimation are also reviewed at some level to provide background and increase clarity.

Data fusion is the most important process in SLAM, and almost every solution with a high level of accuracy utilizes more than one type of sensor. Individual sensors have many shortcomings. However, through the process of data fusion, the complementary observations of the sensors mitigate the disadvantages of each individual sensor. The sensors that are utilized to achieve an object-level solution in this thesis are IMU, wheel odometer, camera, and low-cost 2D ultrasonic rangefinder. In order to achieve this data fusion, extrinsic (boresight and lever arm) calibration of two sensors is performed. Therefore, the state-of-the-art extrinsic calibration methods of an IMU and a camera and extrinsic calibration of an ultrasonic rangefinder and a camera are reviewed in Section 2.6. Finally, Section 2.7 provides an overview of the proposed solutions to address the key research gaps.

2.2 Object Detection/Segmentation

Object detection and segmentation are important components in an object-level SLAM. Object detection refers to the task of locating objects of interest in the images. The output of such detection is

usually a bounding box around the object. Object segmentation, however, identifies the pixels that belong to the object. The output of object segmentation is often a binary mask where the foreground (object pixels) is labelled as 1, and the background pixels are labelled as 0. Object detection and segmentation methods can be classically categorized into template matching, feature detector/descriptors, and active contours (Treiber, 2010). With the advances in DL, a rapidly growing number of object detection/segmentation algorithms are also emerging and designated as the fourth category. In the following, each of these four categories is reviewed in detail.

Template matching is a conceptually simple method of addressing object detection. A template is often represented using a rectangle (Lewis, 2010) or an ellipse (Bolme et al., 2009), and it can be defined by the user or be learned from a set of training data (Avidan, 2004; Bolme et al., 2010). Template matching aims to minimize the distance (or maximize the similarity) defined between the template and the image. A simple matching technique involves convolving (or correlating) the template with the image and finding the region of the highest similarity (Felzenszwalb et al., 2009). However, this approach is computationally inefficient since templates of different scales and orientations (and possible other transformations) must be exhaustively matched to the image. An alternative approach to exhaustive matching is to use optical flow (Oron et al., 2014). These techniques can track the templates frame-by-frame, where the initial approximate location of the template is provided by the last frame.

Template matching is a computationally efficient method for the detection of planar objects (e.g., road layouts (Kyriacou et al., 2005) and simple box-shaped objects with planar surfaces (Jensfelt et al., 2006). The disadvantage of the template matching method is that it includes background pixels. These pixels can diverge the template from the object in the image (i.e., an increasing number of background pixels enter the template area eventually) (Matthews et al., 2004).

While template matching methods represent an object using a bounded region, the **detector/descriptor paradigm** uses salient feature points on the object's surface. The detector/descriptor paradigm is a three-step object detection process. In the first step, salient feature points (e.g., corner points (Harris & Stephens, 1988) and Blobs (Lindeberg, 1993)) are detected in the image. In the second step, the neighbouring pixel of the salient point is used to build a descriptor vector (a feature vector) (e.g., SIFT (Lowe, 2004)). In the third step, global matching (Bentley & Friedman, 1979; Silpa-Anan & Hartley, 2008) or local tracking (Zhang et al., 2014) is used to establish data correspondence. An object can be represented as a cluster of feature points (Castle et al., 2010; Civera et al., 2011; Gálvez-López et al., 2016). This cluster of feature points and their associated feature descriptors can be stored in a database in an offline phase. As new images are acquired (during navigation), the detected feature points can be matched to the object database.

An important advantage of the detector/descriptor paradigm is that objects are represented as sets of points; therefore, numerous geometrical constraints can be imposed to remove outlier data correspondence (e.g., epipolar geometry). Further, objects can still be detected under partial occlusion since some of the feature points are likely to be visible. The detector/descriptor paradigm's drawback is that objects should have texture on their surface.

Both template matching and detector/descriptor methods do not explicitly track the object's contour in the image. However, contours correspond to an object's shape, one of its important distinctive characteristics. In the past, object contours were detected and tracked using **active contours** (Cohen, 1991; Terzopoulos & Szeliski, 1993; Caselles et al., 1997; Chan & Vese, 2001)**.** An active contour is a parametric curve that iteratively converges to the object's boundary. This convergence is based on minimizing the contour's energy, defined as the summation of external and internal terms. The external term is derived from the image (often the image's gradient), and the internal term is derived from prior knowledge of the

object's shape or appearance. Prior knowledge of the shape of an object is known as shape-priors and is discussed in object representation in Section 2.3.

With the increasing processing and storing power of computers, a new class of machine learning methods, namely **DL**, has emerged. Two strategies based on DL are useful for object-level SLAM. The first strategy is based on object detection (Liu et al., 2016; Redmon et al., 2016). The output of this approach is a bounding box around the object. The second strategy is object segmentation (Long et al., 2015; Ronneberger et al., 2015). The segmentation designates the pixels belonging to the object in the image and, therefore, is often more advantageous. It is also required to correspond the detected/segmented objects in individual images with one another with the help of a tracking method. The most common approach is tracking by detection, which refers to the process of detecting objects in each image independently and subsequently matching these detections. As opposed to tracking by detection, in the direct method, the object of interest (which is defined or designated in the first frame) is tracked by maximizing the value of a similarity function between the tracked region and the image (Bertinetto et al., 2016; Bhat et al., 2018; Li et al., 2018).

One of the most restrictive shortcomings of DL-based methods is the requirement for large training data. This data set should include distortions due to geometrical and photometric variations. Furthermore, partial visibility (due to occlusion) should also be considered in the training set. The exclusion of such training data will severely reduce DL-based tracking accuracy (Emami et al., 2020).

Table 2.1 summarizes the advantages and disadvantages of the four categories used for object detection/segmentation and tracking. These methods are also categorized based on how frequently they have been utilized in object-level SLAM. Among these approaches, DL and detector/descriptor paradigms are most utilized in object-level SLAM. More details will be provided in Section 2.4.

Table 2.1: Summary of the methods that can be used for object detection and segmentation.

Method	**Information**		
	Advantages	*Disadvantages*	*Object-SLAM(Applied?)*
Templated matching	• Conceptual simplicity • Suitable for planer objects	• Computational cost • Divergence due to background clutter	Limited
Detector/ Descriptors	• Robust to occlusions • Robust to photometric distortions	• Unusable for texture-less objects. • A very large number of features	Frequently
Active contours	• Robust to geometrical distortions • Robust to photometric distortions • Usable for texture-less objects.	• Computational cost • Lacks robustness to occlusions • Lacks robustness to background clutter	Limited
Deep learning	• Robust to occlusions • Robust to geometrical distortions • Robust to photometric distortions • Usable for texture-less objects	• Large training data requirement	Frequently

In order to address the challenges associated with the massive training data requirements (and manual labelling) for DL, image-synthesizing was used in the past. Image-synthesizing can be divided into three approaches. The first approach is based on complete simulation, where the photo-realistic images are generated from a virtually designed indoor/outdoor environment (Tremblay et al., 2018; Rasmussen et al., 2022). This procedure relies on advanced computer graphic techniques such as raytracing, where generating a single high-resolution image might take up to minutes (Saleh et al., 2018).

The second image-synthesizing approach is based on domain adaptation using Generative Adversarial Networks (GAN) (MacKay & Moh, 2021). Domain adaptation refers to a method that transforms an input image from the source domain (the simulated environment) to the target domain (the real environment). With the help of domain adaptation, millions of synthesized images can be generated. Such images and their corresponding mask can be used to train a neural network to detect/segment objects

of interest. Similar to the first approach (complete simulation of the environment), the domain adaptation often requires simulating the indoor/outdoor environment of the target domain.

The third image-synthesizing method is the hybrid technique. In this technique, only certain aspects of the environment (such as illumination conditions and object's 3D models) are simulated (Dwibedi et al., 2017; Georgakis et al., 2017). The other aspects (such as the background clutter) are captured using real images. The advantage of the hybrid approach is that it can provide a means to reduce manual labour by only simulating some aspects of the images. Table 2.2 compares the labour required for different techniques to generate training images. The labour intensity refers to human resources required for preparing a training data set (e.g., labelling images). The labour intensity is broadly categorized into very high, high, and low requirements. In this thesis, a hybrid approach is used to generate a set of training images and output masks. This approach is briefly explained in Section 2.7

Table 2.2: Summary of different methods of generating training data for deep learning networks.

Method	**Information**	
	Task (labour intensity)	*Details*
Real images	• Capture images in different scenarios (high) • Label images (very high)	• Highest accuracy • Wide area of applicability
Simulation	• Design the background environment (high) • Build object's 3D model (low)	• Medium to high accuracy • Moderate area of applicability
GAN	• Capture target domain images(high) • Build images in the source domain (high)	• High accuracy • Moderate area of applicability
Hybrid	• Capture the background images (low) • Build object 3D model(low)	• Medium to low accuracy • Limited area of applicability

2.3 Object Representation

Object representation is the information about the shape and/or appearance of the object. This information can be obtained prior to or during the process of mapping and localization. Object representation is often used to estimate the pose of an object and thus is an important component for many object-level SLAM solutions. Once the objects are segmented in the images, they should be matched to object representation. The objects in the images often change in appearance and shape due to illumination conditions and distortions such as the perspective projection of a camera. Therefore, matching segmented objects to these representations can be a challenging task.

Object-level solutions to the SLAM problem in the past have often utilized salient points on the surface of an object for the representation. This type of representation requires textures on the surface of the object (Civera et al., 2011; Gálvez-López et al., 2016). However, most objects lack such textures. In order to address this challenge, other methods in the past have been proposed that represent the objects using simple geometrical shapes such as ellipsoids (Nicholson et al,. 2019; Ok et al., 2019) and cuboids (Yang & Scherer, 2019). These approaches take advantage of the well-defined perspective projection of these shapes. Therefore, it is possible to utilize simple observation models, as was the case for classical point-based (Davison et al., 2007), line-based (Smith et al., 2006), or plane-based (Kaess, 2015) methods. Unfortunately, representing every object using simple shapes such as ellipsoids might not be accurate. An alternative object representation approach is to rely on detailed object models. Object-level SLAM solutions that utilize such models can be categorized into two groups. In the first group, models are learned during the navigation period (Prisacariu et al., 2013; Caccamo et al., 2017) (*online modelling*) while in the second group, prior database of object models is used (Salas-Moreno et al., 2013; Joshi et al., 2018; Parkhiya et al., 2018; Wang et al., 2021) (*offline modelling*).

The first group often utilizes image segmentation to detect plausible hypotheses of objects in consecutive frames and eventually builds 3D models by tracking and evaluating hypotheses segments. The computational complexity of image segmentation is one of the biggest challenges of these approaches. In the second category of the methods, the model is built prior to the localization and mapping. The most common approach is to build a 3D shape prior of the objects. Unfortunately, matching these 3D shape-priors directly to the contour of the segmented object in the images is challenging, and most such solutions are only suitable for RGB-D cameras. Another possible solution is to represent an object as a set of 2D contours. Such 2D contours are built using the images of the object captured from different viewpoints (Hinterstoisser et al., 2013). These approaches represent the 2D contours often as images which are not robust to the variation in the scale, translation, and orientation. This is an important disadvantage as such geometrical variations between the shape-prior and the segmented objects in the image are expected to be encountered.

As opposed to image-based methods, contours can also be represented using explicit 2D parameterization (Cremers, 2002; Khalid, 2012) and implicit 2D parameterization (Tsai et al., 2003; Foulonneau et al., 2009; Tran et al., 2013). Explicit parameterization represents the contour of an object as parameterized curves, while implicit representation uses level sets. The two representations are shown in Figure 2.1. The boundary in the implicit representation is the zero-crossing of the level set function ($F(\emptyset) = 0$), while in the explicit representation, the boundary coordinates are parameterized with a variable (p). With the help of such parameterization, the scale, in-plane rotation and translation of parameterized shapes can be changed easily. A drawback for 2D parameterized shape prior set is that, unlike image-based contour representation, matching parameterized shapes is more challenging. The matching process requires finding the changes in all the geometrical parameters (such as scale), or the measured distance will be erroneous. Table 2.3 summarizes the advantages and disadvantages of object

representation methods. In this thesis, a novel object representation using parameterized 2D shape priors is introduced. The advantage of this solution is briefly summarised in Section 2.7.

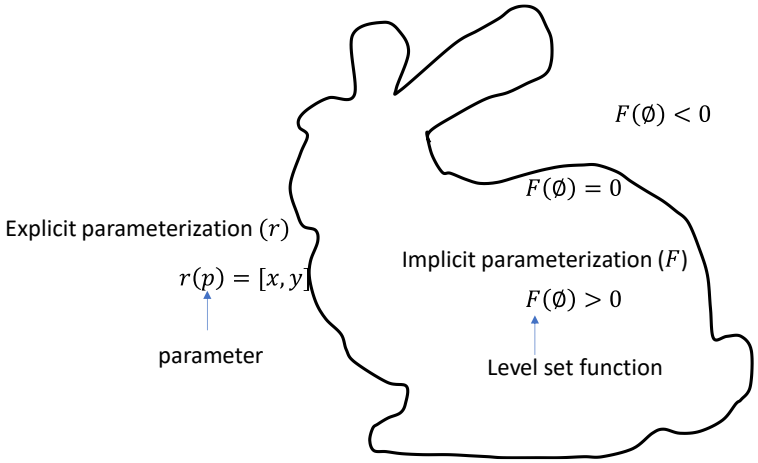


Figure 2.1: Illustration of implicit and explicit shape representation.

Table 2.3: This table summarizes the advantages and disadvantages of object representation methods.

Method	**Information**		
	Advantage	*Disadvantage*	*Details*
Simple geometry	• Simple perspective projection model	• Inaccurate representation	Objects are represented using primitives such as ellipsoids, cuboids
Feature point clusters	• Robust to occlusions	• Requires surface textures	Objects are represented as a collection of features points such as SIFT
3D shape-based models	• No requirement for surface texture	• Difficult 3d-2d matching	Objects are represented using 3D models such as volumetric voxel-based models
2D shape-based (Images contour)	• No requirement for surface texture • easy 2d-2d matching	• Not robust to geometrical variations	Objects are represented as a collection of 2D contour images
2D shape-based (Parameterized contour)	• No requirement for surface texture • easy 2d-2d matching	• Limited application in pose estimation	Objects are represented as a collection of 2D implicit or explicit parameterization of curves.
No model	• No requirement for prior models	• Computationally very costly • Mostly implemented using RGB-D	Objects models are built during navigation

2.4 Object Pose Estimation

Object pose estimation is an important module in an object-level SLAM. The classical solutions for the pose estimation are similar to the PnP algorithms. In this process, some feature points are detected on the object of interest. The detected feature points are matched to the object model. These 2D-to-3D correspondences are utilized to estimate the 6DoF pose of the object. Unfortunately, objects with repetitive or no texture are unsuitable for this approach. In order to address this issue, some pose estimation techniques have relied on the contours of the objects (Hinterstoisser et al., 2013) in the past. While the texture problem is addressed by utilizing the object contours, other challenges should also be addressed.

One of these challenges is the degradation of the accuracy of contour detection in the presence of occlusion and background clutter. In order to address such issues, DL-based solutions for pose estimation also has emerged. These solutions can be broadly categorized into one-step and two-step methods. In one-step or end-to-end methods (Xiang et al., 2017; Hu et al., 2020), the pose of an object is directly estimated using one DNN. However, this approach has several drawbacks. For the regression-based end-to-end DNN (where the input is an RGB image and the output is a vector representing translation and rotation), it is difficult to estimate the poses of the objects that are symmetrical (produce similar silhouettes/appearance from some viewpoints). Therefore, the conventional one-to-one pose estimation should be replaced with networks that can produce one-to-many estimation. Categorical DNN (where the input is an image and the output is a 2D or 3D matrix of discretized poses) can handle symmetrical situations. However, in order to achieve an accuracy of 5°, approximately $50{,}000$ training images are required to include one sample for one pose (Sundermeyer et al., 2018). This number of images should be increased to include different backgrounds and scenarios where the object is occluded.

In two (or more) step approach, the pose estimation task is divided into smaller modules. Each module is responsible for performing one task, and a DNN can be trained separately for them. However,

it is not required to rely on DL to implement every module. A common approach is first to detect features on the object of interest using DL and, in the second step, use these features to estimate the pose (Pavlakos et al., 2017; Rad & Lepetit, 2017; Zakharov et al., 2019).

In order to address the issues that arise in the case of symmetrical objects, several solutions have been proposed in the past. One solution is to detect and exclude the symmetrical object from the initial pose estimation. In this approach, the camera's pose and trajectory are estimated using other objects in the images. With the help of this initial estimation, the symmetrical objects are integrated into the solution in the second step (Merrill et al., 2022). Therefore, such methods are unsuitable if other objects are not observed in the image with the symmetrical objects. A second solution to address pose-ambiguities is to train a DNN for different ranges in which such ambiguities do not exist. In the pose estimation phase, the likely poses of the objects in each region are identified (Rad & Lepetit, 2017). A challenge with this approach is that it requires further manual labour during the time of training to identify regions where the object is not symmetrical.

In this thesis, two different techniques for pose estimation are developed. The first technique jointly estimates the object and camera pose in the SLAM framework. This framework will be explained in Section 3.4 under tightly coupled IMU/monocular object-level SLAM. The second approach estimates the pose using a coarse-to-fine pipeline. This approach addresses some of the key issues with the state of art methods and is summarized in Section 2.7.

2.5 Object-level Localization and Mapping Framework

In the past, different solutions to object-level SLAM were suggested. The earlier solutions are based on EKF (Ahn et al., 2006; Castle et al., 2010; Civera et al., 2011). Since in EKF, the observations are processed one at a time, the error propagation can cause a reduction in the accuracy of the final solution to the SLAM problem. In order to address this challenge, later object-level solutions include two important

stages: frontend and backend. Frontend refers to an initial estimation of the poses of the camera and the objects, which can be achieved using EKF (or other DBN such as PF) and Local Bundle Adjustment (LBA). LBA is performed using a subset of images captured by the camera (denoted as keyframes) in close vicinity, while Global Bundle Adjustment (GBA) is performed using all the keyframes (keyframe-based LBA is introduced in (Klein & Murray, 2007)). The frontend solution is then provided to the backend, where further optimization is performed using GBA or Factor Graphs (FG) (Dellaert & Kaess, 2006).

The solutions to object-level SLAM can also be categorized based on how objects are integrated into the method. In the first category, objects are used to add additional constraints in the backend (Pillai & Leonard, 2015; Dharmasiri et al., 2016; Bowman et al., 2017). These constraints can improve the accuracy of classical SLAM solutions such as ORB-SLAM2. In the second category, the map and the trajectory estimation are performed using objects only. These objects are initialized with 6DoF in the map by estimating their pose in the frontend. The poses and the trajectory of the camera are refined using GBA (Ok et al., 2019; Yang & Scherer, 2019; Qian et al., 2020).

Table 2.4 and Table 2.5 summarize some of the past solutions to object-level SLAM. Most of these methods utilize descriptor/detector paradigm and DL-based object detection. Object representation is also utilized by most of the solutions. Among these methods, objects are most represented as ellipsoids and feature point clusters (points clouds with assigned feature descriptors for each point). The methods that do not use any object representation also do not provide a map of the objects (Bernreiter et al., 2019; Bowman et al., 2017). For the pose estimation, classical PnP (Civera et al., 2011) and DL (Deng et al., 2021; Song et al., 2021) are used mostly. A solution that depends on ellipsoids (to represent an object) estimates the tangential planes encapsulating the object and subsequently fits an ellipsoid with correct orientation and position (Ok et al., 2019; Qian et al., 2020). Some of the solutions have decoupled the

mapping and trajectory estimation. (Joshi et al., 2018; Parkhiya et al., 2018). Such decoupling can result in a decrease in accuracy due to error propagation. Most of these solutions implement a frontend and a backend. It is important to note that most of the solutions in the past did not provide a standalone object-level SLAM and depended on low-level feature-based SLAM (ORB-SLAM or visual odometry (Nistér et al., 2004)) to find the initial solution.

The introduction of keyframe-based LBA has shifted the interest away from DBN methods in the last decades in classical and object-level SLAMs. This approach can often attain higher accuracy. However, one of the aspects of many keyframe-based methods is the reliance on standalone monocular camera solutions. Thus, no motion prediction beyond heuristics and simple models (e.g., constant velocity) is assumed to be available. A more accurate motion prediction can be achieved using IMU in shorter periods of time (and wheel odometry in short to medium periods), which can make DBN methods achieve comparable results to keyframe-based alternatives. The developed method in this thesis is based on sensor fusion of a monocular camera and IMU in a tightly and loosely coupled fashion (Chapter 3).

Another challenge with most of the current object-level SLAM frameworks is that the errors in the observation model and the trajectory of the camera are assumed to have a Gaussian distribution with only a few methods assuming non-gaussian errors (e.g., mixture Gaussian distribution (Doherty et al., 2019)). In practice, many objects cannot be initialized using a Gaussian distribution. For example, symmetrical objects will have ambiguous poses that can stay so throughout the navigation and should be accounted for appropriately. One possible solution is to use a DBN that does not assume a Gaussian error distribution. To this end, the earlier EKF-SLAM (Smith and Cheeseman, 1986) and Extended Information Filter (EIF-SLAM) (Nettleton et al., 2000) were replaced later by PF-SLAM (Thrun et al., 2001).

Table 2.4: Summary of the components and the framework used in different state-of-the-art object-level solutions.

Object-Level Solution	**Methodology**		
	Object Detection	*Object Representation*	*Frontend/backend*
(Bowman et al., 2017)	Detector/Descriptor+ DL	No 6DoF object models used	LBA/FG
(Bernreiter et al. 2019)	Detector/Descriptor+ DL	No 6DoF object models used	Not mentioned/FG
(Doherty et al. 2019)	DL	No 6DoF object models used	EKF/FG
(Qian et al. 2020)	Detector/Descriptor+ DL	Ellipsoids	LBA/FG
(Pillai & Leonard 2015)	Detector/Descriptor	Online (feature point clusters)	LBA/FG
(Civera et al. 2011)	Detector/Descriptor	Offline (feature point clusters)	EKF/ (no backend)
(Nicholson, et al. 2019)	DL	Ellipsoids	decoupled estimation/FG
(Ok et al., 2019)	DL	Ellipsoids	LBA/GBA
(Hosseinzadeh et al., 2018)	Detector/Descriptor+ DL	Ellipsoids	LBA/FG
(Joshi et al., 2018)	DL	Offline (category-level 3D models)	decoupled estimation/FG
(Parkhiya et al., 2018)	DL	Offline (category-level 3D models)	decoupled estimation/FG
(Choudhary et al., 2014)	Detector/Descriptor	Online (point cloud)	FG
(Ahn et al., 2006)	Detector/Descriptor	Offline (feature point clusters)	EKF
(Castle et al., 2010)	Detector/Descriptor	Offline (feature point clusters)	EKF
(Prisacariu et al., 2013)	Active contours	Online (3D level-sets)	Other optimization methods
(Ma & Sibley, 2014)	Active contours	Online (3D level-sets)	Other optimization methods
(Dharmasiri et al. 2016)	Detector/Descriptor	Online (feature point clusters)	LBA/GBA
(Yang & Scherer, 2019)	Detector/Descriptor+ DL	Cuboids	LBA/GBA
(Deng et al., 2021)	DL	3D models with RGB values	RBPF
(Song et al., 2021)	No object models are used	No object models are used	EKF

Table 2.5: Details of different state-of-the-art object-level solutions.

Object-Level Solution	Methodology
	Details
(Bowman et al., 2017)	• Classical point-based visual odometry is used for the initial solution
(Bernreiter et al. 2019)	• Visual/lidar odometry is used for initial solution
(Doherty et al. 2019)	• Classical point-based visual odometry is used for the initial solution
(Qian et al. 2020)	• ORB-SLAM is used for the initial solution. • Object ellipsoids are initialized with the help of bounding tangential planes.
(Pillai & Leonard 2015)	• ORB-SLAM is used as the initial solution. • The solution to SLAM is used to increase the accuracy of object detection.
(Civera et al. 2011)	• Objects are initialized using PnP
(Nicholson, et al. 2019)	• Object ellipsoids are initialized with the help of bounding tangential planes.
(Ok et al., 2019)	• Classical point-based visual odometry is used as the initial solution. • Object ellipsoids are initialized with the help of bounding tangential planes.
(Hosseinzadeh et al., 2018)	• ORB-SLAM2 is used to provide the initial solution. • Besides objects, planar surfaces are also used.
(Joshi et al., 2018)	• Visual odometry is used for camera pose estimation. • Object poses are estimated using very simple shape-prior.
(Parkhiya et al., 2018)	• Visual odometry is used for camera pose estimation. • Object poses are estimated using statistical shape-priors
(Choudhary et al., 2014)	• ICP and wheel odometry is used for sensor pose estimation. • As objects are modelled, the initial object pose is assumed.
(Ahn et al., 2006)	• Homography-based object pose estimation
(Castle et al., 2010)	• Homography-based object pose estimation
(Prisacariu et al., 2013)	• Single-object joint 3D reconstruction and SLAM
(Ma & Sibley, 2014)	• Multiple-objects joint 3D reconstruction and SLAM
(Dharmasiri et al., 2016)	• The initial solution is provided using classical monocular-based SLAM. • Object poses are only used to provide constraints in BA.
(Yang & Scherer, 2019)	• ORB-SLAM2 is used to provide the initial solution. • The pose is estimated using a single image.
(Deng et al., 2021)	• Only uses one object.
(Song et al., 2021)	• End-2-end pose estimation

PF-SLAM does not assume the parameters have a Gaussian distribution and is, therefore, more suitable for object-level SLAM due to the abovementioned issue. A special type of PF-SLAM is RBPF-SLAM (Montemerlo & Thrun, 2003; Murphy & Russell, 2001) which reduces unnecessary computations. Due to the structure of the SLAM problem, it is observed that for a given particle, the poses of the landmarks in the map are conditionally independent. Thus, storing or computing such information for a given particle is avoided using RBPF.

RBPF-SLAM has been applied in the past using simpler geometrical entities such as points (Eade & Drummond, 2006; Nguyen et al., 2008) and lines (Eade & Drummond, 2009), but only recently applied to objects (Deng et al., 2021). Figure 2.4 shows the evolution of DBN-based solutions to the SLAM problem. To the best of the author's knowledge, the framework explained in (Deng et al. 2021) is the first object-level RBPF-SLAM; however, it is only based on a single object.

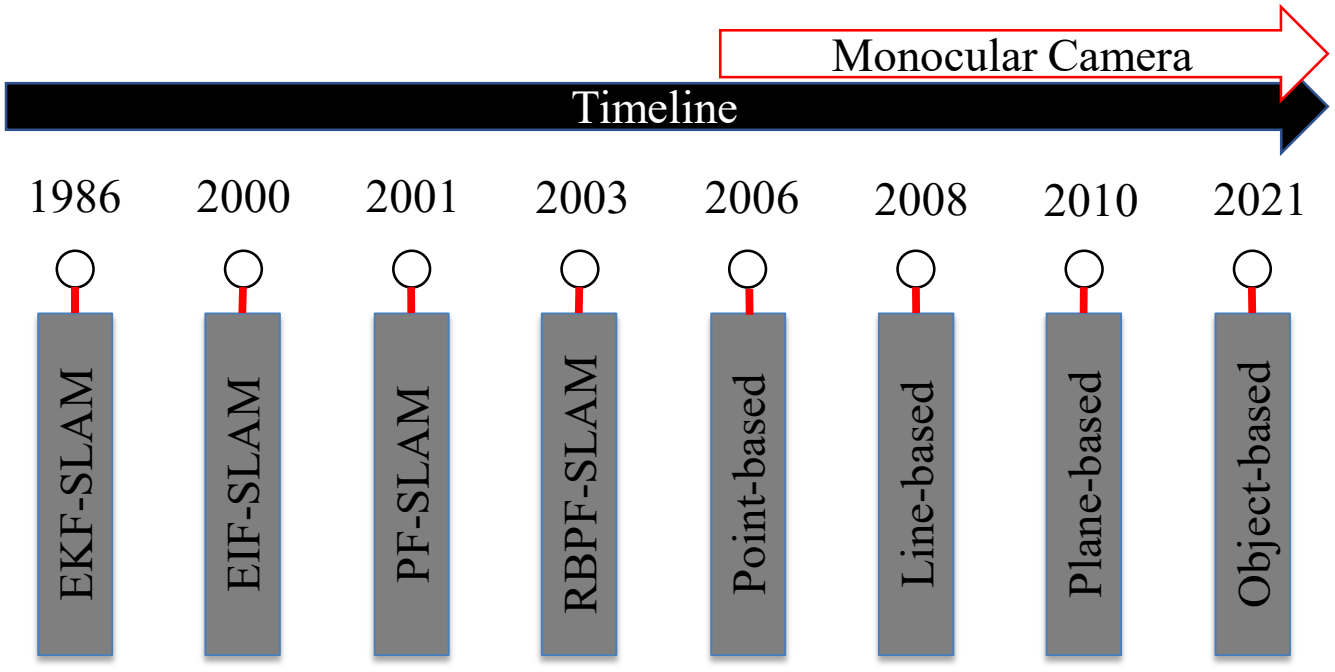


Figure 2.2: Timeline of the DBN-based solutions to the SLAM problem.

It is known that PF-based methods suffer from the Curse of Dimensionality (COD) (Daum & Huang, 2003). Defining COD is beyond the scope of this thesis, and other resources can be used for further information (Asl Sabbaghian Hokmabadi, 2018). However, due to the COD, the number of particles required to effectively sample the space of the solutions can grow exponentially as the number of dimensions grows. This leads to an increase in computational cost significantly. The dimensions of the particles in PF-SLAM are very large due to the number of features in the map.

In this thesis, a novel mapping and localization framework is developed that addresses some of the problems associated with the state-of-the-art object-level mapping and localization framework. The developed approach is summarized in section 2.7.

2.6 Data Fusion: Towards Motion Prediction and Object Initialization

Cameras provide an advantage for object detection/segmentation as the rich radiometric information acquired from the RGB channels of a camera can help detect the appearance and the shape of the objects in the environment. Cameras in the past have been used as standalone solutions to object-level SLAM; however, relying on only a camera is a challenging task. One of the challenges is that cameras depend on the observation of the objects in the environment to estimate the trajectory. This can lead to large errors in trajectory estimation when there are no or few objects in sight of the camera. In these circumstances, heuristic motion models should be used to estimate the camera's trajectory (such as the constant velocity). However, such motion models are reliable when the camera does not move fast. Therefore, fusing cameras with sensors such as IMU is more advantageous. IMU can predict the motion, while the camera observations can be used to weigh the particles in object-level RBPF-SLAM.

A key component of such data fusion is extrinsic calibration, which refers to finding the relative orientation (boresight) and the translation (lever arm) between two sensors. IMU and monocular camera extrinsic calibration in the past has been achieved using mainly two classes of methods. Methods that

depend on special equipment (Lobo & Dias, 2007) and methods that depend on independent trajectory estimation for each sensor (Kelly & Sukhatme, 2009). The first class of methods does not offer an on-site solution and is not useful when the extrinsic calibration parameters are changed during the navigation (for example, due to physical shocks to the device's body). The second class of methods requires an independent and accurate estimation of the trajectory for each sensor. However, this solution might be erroneous since low-cost IMU produces large errors in a short period of navigation. Similarly, accurate trajectory estimation with a monocular camera that is useful for extrinsic calibration requires special targets (such as checkerboards) and is challenging. In addition to these shortcomings, most previous solutions require an initial estimation of the calibration parameters (Furgale et al., 2013; Huang & Liu, 2018). In this thesis, a novel calibration method is developed that is not affected by the disadvantages mentioned before. The advantages of the developed method are explained in section 2.7

In addition to the issues related to trajectory estimation, other challenges also exist for a monocular camera. A monocular camera is a type of bearing-only sensor where only the direction of a point from the camera is known. Using a bearing-only sensor, it is not possible to insert the landmarks into the map with few observations. The problem of inserting landmarks into the map is known as initialization. In the past, two classes of methods were proposed to solve the initialization problem. The first approach is delayed initialization, where points are observed from different viewpoints, and therefore, the uncertainty of the position of a landmark can be reduced (Bailey 2003; Davison et al. 2007; Munguia & Grau 2012). Delayed initialization methods use techniques such as triangulation to reduce the uncertainty of the location of a landmark. Therefore, a landmark should be observed from different viewpoints with a sufficiently large baseline. Delayed initialization is suitable for landmarks close to the robot, while distant landmarks cannot be initialized using this approach. Further, during the delayed initialization, corrections to the robot's

trajectory are not possible. This will cause challenges in IMU/monocular-based solutions as relying only on an IMU for pose estimation for a longer period can deteriorate the solution significantly.

In order to address the abovementioned issues, undelayed initialization has been proposed in the past. In undelayed initialization, points are immediately inserted into the map with a large location uncertainty (Gordon et al., 1993; Kwok & Dissanayake, 2003, 2004; Solà et al., 2005; Mungúia et al., 2013). The accuracy of the trajectory estimation of a robot can be improved using undelayed initialization (Eade & Drummond, 2006). This improvement is because of the utilization of landmarks for bearing correction immediately after they are observed. However, due to the unknown position of a landmark at the initialization phase, undelayed methods use larger uncertainty. This uncertainty, in turn, will require more particles in RBPF-SLAM and, therefore, increase the computational cost. See Figure 2.3 for a schematic comparison of delayed and undelayed initialization. In delayed initialization, the landmark cannot be inserted into the map at the time t_k but as the robot moves and encounters the object, the uncertainty in the position of the object decreases. Finally, the object can be inserted into the map at the time t_{k+1}. During the motion, the robot can only rely on other sensors (such as wheel odometry) to estimate its position but not the camera. In contrast, in undelayed initialization, the landmark is instantly inserted into the map at the time t_k but with large uncertainty. During the motion, this uncertain landmark can still provide partial corrections to the pose of the robot.

With the help of 2D and 3D rangefinders, the distance to the object can also be directly observed. Unfortunately, the points clouds obtained using these sensors are sparse in comparison to the resolution of the camera and therefore, most pixels will lack an associating depth. However, unlike point-based SLAM methods, objects are volumetric entities, and object segments can occupy a large number of pixels in the image. A few or even one pixel with a depth value can be used for the initialization, making it

possible to use 2D sensors such as ultrasonic rangefinders. The initial pose of the object can improve during the filtering process as more observations become available.

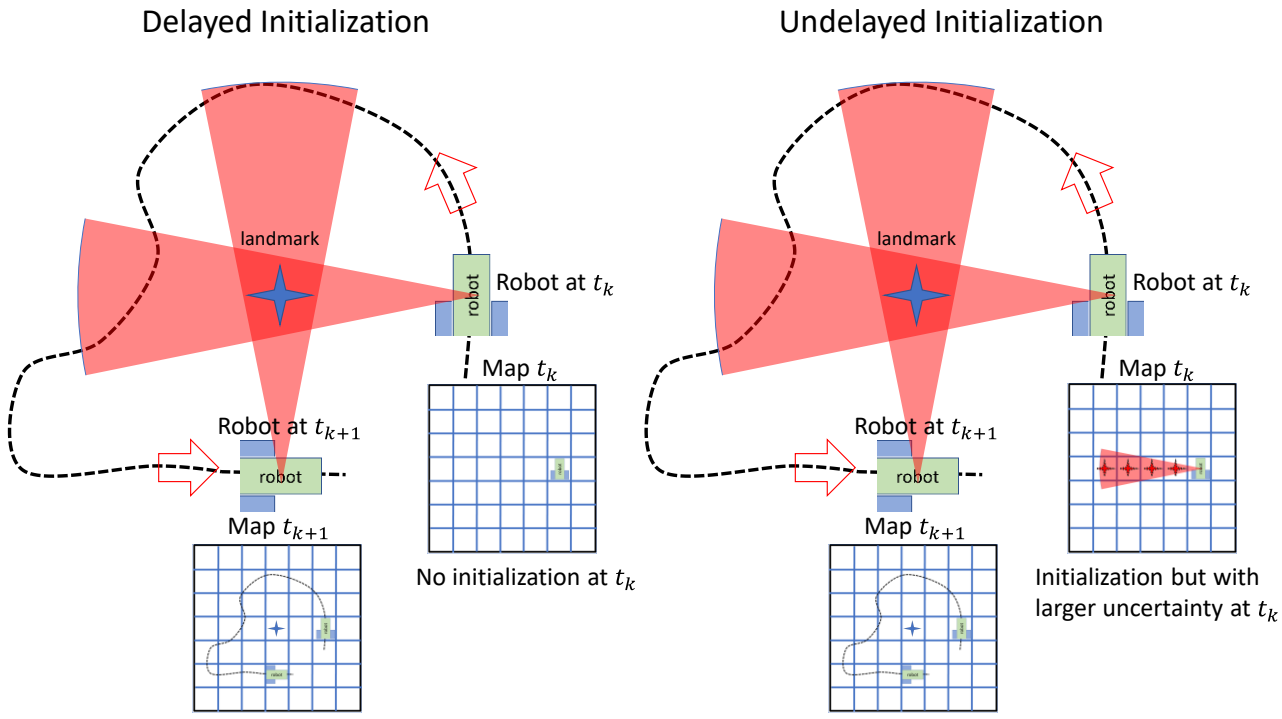


Figure 2.3: Schematic comparison of delayed and undelayed initializations.

In this thesis, a novel fusion method of an ultrasonic rangefinder and a monocular camera is developed. The developed solution projects ultrasonic observations onto the image. If the projected ultrasonic observation falls within the segmented area of the object, distance observation can be utilized. More details about this technique of fusion are provided in Section 3.6. Since the projection of the ultrasonic reading in the image requires accurate extrinsic calibration, the remainder of this section will briefly explain the methods of extrinsic calibration of a 2D rangefinder and a monocular in the past and underline the current gaps.

Extrinsic calibration methods between rangefinders and cameras could be divided into targetless and target-based methods. The target-based approach utilizes a special target, such as a checkerboard. The checkerboard is placed in a location that can be detected using the camera and the rangefinder. An extrinsic calibration method often matches detected lines on the checkerboard using a 2D rangefinder with the detected checkerboard's plane using a camera (Zhang & Pless, 2004; Unnikrishnan & Hebert, 2005; Kassir & Peynot, 2010; Gomez-Ojeda et al., 2015; Dong & Isler, 2018). The literature indicates that 20-30 (Unnikrishnan & Hebert, 2005) detected points on the checkboard are required. However, this number is large for the ultrasonic rangefinder with a low angular resolution. Therefore, the checkerboard size should be larger, substantially increasing the cost. In the target-less calibration approach, the structure of the surrounding environment has been utilized for the calibration. To the author's knowledge, the target-less calibration methods are developed for more accurate rangefinders (e.g., LiDAR). The developed method in this thesis is a targetless calibration method that utilizes the available structure of the indoor manmade environment to find a match between certain points observed by a monocular camera and a local map built by the ultrasonic rangefinder. Section 5.4 will provide a detailed methodology.

2.7 Chapter Summary

Object-level SLAM includes a number of components. These components are object detection/segmentation, object representation and object pose estimation. Finally, the components are integrated using a mapping and localization framework. Based on the literature review, a number of important research gaps are identified and addressed in this thesis. In the following, the research gaps and overview of the developed solution are summarized.

Current object-level SLAM mapping and localization rely on the classical feature point-based methods for the initial solution. These feature point-based methods (such as ORB-SLAM 2) themselves rely on the assumption that the objects of interest have texture on them. Further, most of the current

methods rely on the fact that the uncertainties in the camera pose and the objects can be represented using Gaussian distribution. In order to address these issues, a tightly coupled IMU/monocular camera object-level solution using RBPF-SLAM is developed. This developed approach can address the restriction of the current methods. Further, the proposed solution only relies on the shape of the object of interest.

One of the challenges with the fusion of a camera with low-cost IMUs is that the errors tend to accumulate very fast due to the dead-reckoning nature of an IMU mechanization. While in the outdoor environment, this can be addressed by providing frequent updates using GNSS signals, in the indoor environment, other sensors should provide such updates (here, camera). This problem is specifically very important when an object is encountered for the first time. In order to address this challenge, an undelayed initialization is utilized in the thesis. With the help of such initialization, the measurement updates can be provided to IMU immediately after the object is observed in the first image. Unfortunately, the undelayed initialization using a monocular camera results in large ambiguity in the distance of the camera from the object of interest. In order to address this issue, in this thesis, a novel method of fusion with low-cost ultrasonic sensors is proposed.

In order to achieve a fusion between an IMU and a monocular camera, the system should be calibrated. This calibration process includes estimating the boresight and lever arm parameters. In the past, most calibration methods required targets (often with known coordinates) and/or special equipment. Further, most solutions require an initial estimate of the parameters. However, such an initial estimate might not be available. In this thesis, a method of extrinsic calibration is developed which utilizes Manhattan World Constraint (MWC). MWC refers to the geometrical relationship of the planar surfaces (e.g., wall, floor, ceiling) in the indoor environment, where these surfaces are often parallel or orthogonal to each other. Such structures can be used to correct the heading of a device (Elloumi et al., 2014) or

provide constraints in the map-building process (Abadi & El-Sheimy, 2022). In Section 5.3, the development of the IMU/camera boresight calibration using MWC is explained.

While object detection and segmentation have been proposed in the past, a segmentation algorithm is often preferred to detection since it can identify the object of interest at the level of pixels. The state-of-the-art solutions for object segmentation rely mainly on DNNs. However, training such networks requires a large database. For most objects, it is difficult to find training images and their corresponding output masks in the available online databases. In these situations, synthesizing images can meet the large training set requirement. In this thesis, a hybrid approach to generate a set of synthetic training images and output masks is proposed. This approach relies on the fact that for the task of object segmentation using monocular cameras, it is sufficient to only segment the foreground (object) from the background clutter. In this method, the object's pose and the illumination conditions (such as shading and highlights) are simulated. For the background (the scene), real images are captured from multiple indoor environments. The simulated object is projected onto a virtual camera and is finally superimposed on a background image. The simulated setup is used to generate tens of thousands of virtual training images and output masks. With the help of these images, a U-Net (Ronneberger et al., 2015) model is trained. Section 4.2 will explain this in more detail.

The shape-based object representation in the past was achieved using 2D and 3D shape priors. The 3D shape-priors can be directly matched to the observation from the monocular cameras. Further, the 2D shape-priors are often built using a set of images captured from the object of interest from different viewpoints. Representing an object using images can lead to challenges in the matching step. Such challenges are due to the difficulty of changing the scale, in-plane rotation and translation for an image-based representation. In this thesis, the objects are represented using a collection of explicit 2D contours. Samples taken from this set for four objects are shown in Figure 2.4.

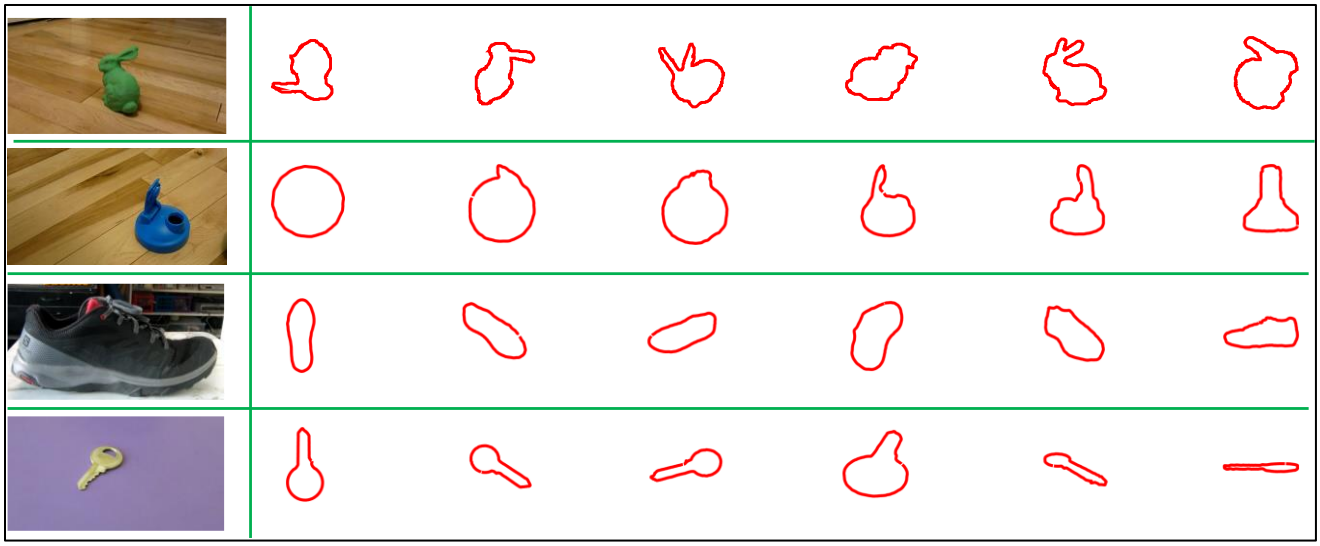


Figure 2.4: Some samples taken from the shape-prior set for four objects

The shape-prior set can provide a coarse estimation of the pose of the object with respect to the camera in the matching process. The developed method has robustness to geometrical and photometric distortions since it does not rely on textures. Sections 4.3 and 4.4 will explain the details of the developed approach.

The state-of-the-art end-to-end pose estimation often requires a large data set and can encounter difficulty addressing issues regarding symmetrical objects. On the other hand, most of the classical solutions to the pose estimation rely on the detection of salient feature points on the objects. In this thesis, a novel method of shape-based pose estimation is developed to address these challenges and limitations. For this approach, initially using a DNN, objects are segmented from background clutter, occlusions, and other sources of distortions. This segmented object is then matched to the shape-prior introduced in the previous paragraph. The output of this matching process is a *list of best poses (poses with the smallest errors)* instead of only one pose; therefore, it can also be used for symmetrical objects. Finally, the initial coarse pose estimation is refined using a PnP algorithm. This pose estimation method does not rely on the object's texture. Figure 2.5 illustrates three examples of estimated poses using the developed method. In

this figure, the object of interest is a blue water bottle cap. The 3D model of the object and the estimated viewpoints are shown in an object-centric coordinate frame (Figure 2.5 top row). For this example, hundreds of best poses are selected for clear visualization. The images are taken from different angles, and it can be seen that object symmetry will result in ambiguity in the pose. Pose estimation and pose refinement are important modules for the loosely coupled IMU/monocular camera approach, and it is explained in Sections 4.3 to 4.4 in detail.

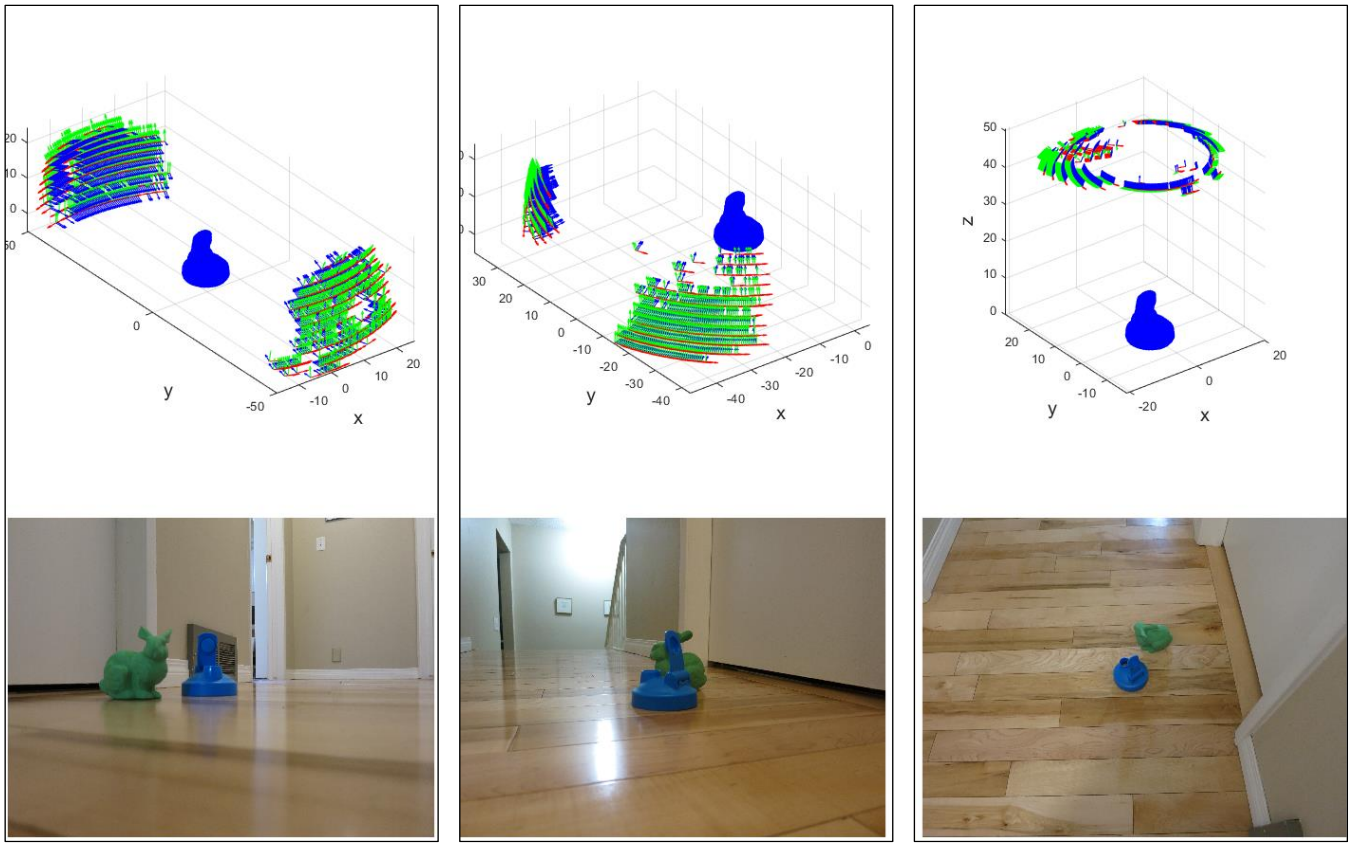


Figure 2.5: Samples of the developed contour-based pose estimation. This method can estimate ambiguous poses.

Chapter 3: Tightly/Loosely Coupled Object-level RBPF-SLAM

3.1 Overview

The Object-level SLAM problem includes many sources of errors. Among these, a type of error can occur due to the symmetrical shape or appearance of the object from different viewpoints, which results in ambiguity in the object's pose with respect to the camera. Another type of error can arise in the process of object segmentation. Such errors occur since the objects can be partially segmented due to occlusions (or background clutter) in the indoor environment. These two types of errors are amongst many that can result in inaccuracies or complete failure of the solution to the SLAM problem. In order to account for such errors, Gaussian distribution is unrealistic and classical DBN such as EKF cannot be utilized. For example, if symmetrical objects are encountered, camera poses are best represented by two or more Gaussian distributions around the symmetrical axes of the object. Therefore, utilizing other DBN methods that can use or approximate multimodal distributions, such as RBPF, is crucial.

In Section 3.2, an overview of the object-level RBPF-SLAM is provided. In Sections 3.3 and 3.4, the developed loosely and tightly coupled IMU/camera fusion methods are explained. In these methods, the acceleration and angular velocity readings from IMU are used to predict the trajectory of the particles and images from the camera are used to weigh the particles. Such fusion can be useful for handheld devices nowadays, as they include both of these sensors. For the wheeled robot (which is a widely used type of robot), IMUs can be replaced with wheel odometers, and tightly/loosely coupled methods can be adapted to these robots. The main difference is that instead of using IMU mechanization, the robot's kinematic model is used to predict the trajectory, which is explained in Section 3.5. In Section 3.6, it is explained how an ultrasonic rangefinder can also be integrated into the developed solution. Such fusion will provide distance observation to the object and reduce the pose of uncertainty. Finally, Section 3.7 discusses some of the challenges with the particle weighting process in RBPF-SLAM.

3.2 Object-level RBPF-SLAM

DBN has been utilized in the past to formulate a solution to the SLAM problem. These Bayesian networks include two main types of variables. The first type is the known variables. Examples of such variables can be the odometry inputs or the observations (e.g., images, point clouds). The second type of variable is the hidden variable. The hidden variables are not directly observed but can be estimated given the known variables. The hidden variables can be the robot's state (e.g., position, attitude) and the landmark's state (e.g., position). The landmarks can be points, lines as well as objects. One of the differences between the object-level and classical point-based SLAM is that the landmarks defined as objects have an orientation as well as a position. Figure 3.1 shows a simple schematic of DBN where hidden variables are shown in gray colour. In this figure, the inputs, the device's poses, the observations, and the landmarks are denoted as u, x, z, m, respectively. In a SLAM problem, the motion model predicts the next state, while the observation model is used to update the state of the robot and the landmarks. Particle filters are among the methods that can be used to implement a DBN.

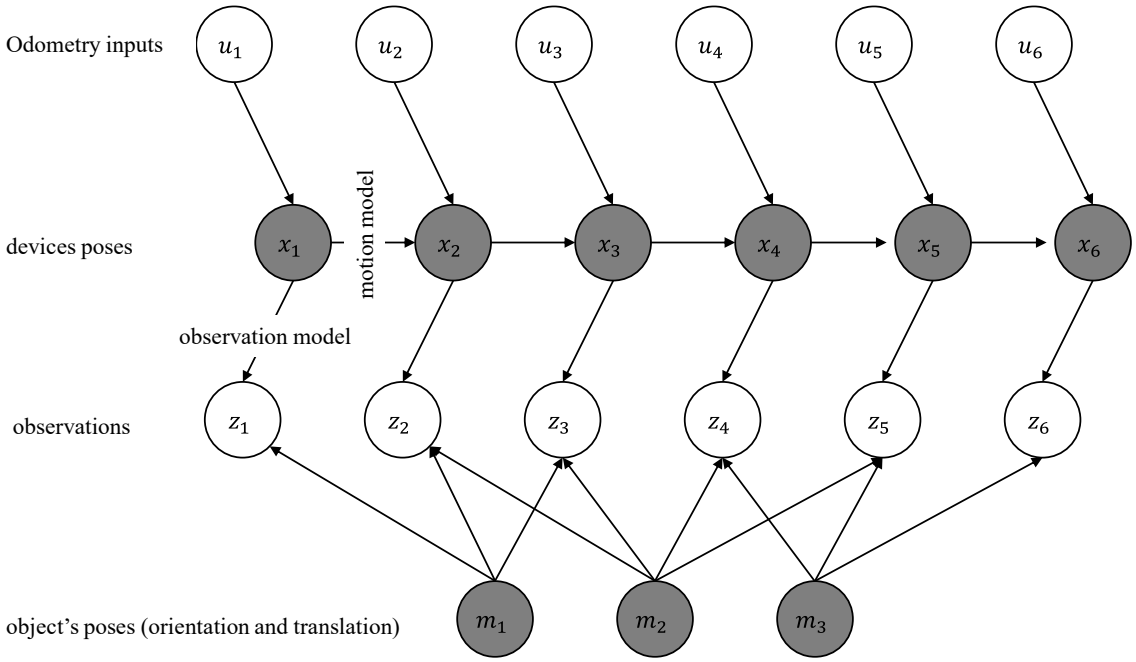


Figure 3.1: Illustration of the Dynamic Bayesian Network of the SLAM problem

The developed method in this thesis extends the methodology of RBPF-SLAM to objects. RBPF-SLAM is a special type of PF-SLAM, where each particle represents a possible trajectory of a robot (x) and the pose of the landmarks in the environment (m). Equation 3.1 shows the formalization of RBPF-SLAM (Montemerlo & Thrun, 2003) where subscript $1{:}\,t$ indicates epochs starting from the beginning of the navigation time ($x_{1:t} = \{x_1, x_2, \dots x_t\}$). The first term on the right-hand side of Equation 3.1 is the posterior of the robot's trajectory, and the second term is the posterior of the state of the landmarks in the map. A single particle is associated with one trajectory and one map, each consisting of multiple objects. In this equation, data association between landmarks and observations is denoted as a. Data association is the process of assigning observations to landmarks which can be a challenging problem when there are similar-looking objects in an environment. As suggested by (Montemerlo & Thrun, 2003), Maximum Likelihood Estimation (MLE) can be used to keep the most likely data association for each particle. Equation 3.2 shows this where $x_{1:t}^{[n]}$ denotes the trajectory of the n_{th} particle, a_t denotes a data association hypothesis, and $\hat{a}_t^{[n]}$ denotes the MLE for the data association. $\hat{a}_t^{[n]}$can be inserted back into Equation 3.1 once calculated.

$$p(x_{1:t}, m|a_{1:t}, z_{1:t}, u_{1:t}) = p(x_{1:t}\,|a_{1:t}, z_{1:t}, u_{1:t}) \prod_j p(m_j|x_{1:t}, a_{1:t}, z_{1:t}, u_{1:t}) \tag{3.1}$$

$$\hat{a}_t^{[n]} = argmax_{a_t} p(z_t|a_t, \hat{a}_{1:t-1}^{[n]}, x_{1:t}^{[n]}, z_{1:t-1}, u_{1:t}) \tag{3.2}$$

The update step in the RBPF-SLAM includes particle weighting and resampling. The weighting is proportionate to the observation likelihood, and it is shown in Equation 3.3 ($w_t^{[n]}$ denotes the weight). The right-hand side is only proportionate to the left-hand side since the weights have to be normalized to sum to one. The details about weighting will be provided in Sections 3.3. and 3.4.

$$w_t^{[n]} \propto p\left(z_t \middle| \hat{a}_t^{[n]}, x_{1:t}^{[n]}, z_{1:t-1}, u_{1:t}\right) \tag{3.3}$$

As an overview, the following modules should be defined in RBPF implementations (flowchart provided in Figure 3.2).

1. **Proposal distribution** is the predicted trajectory of particles, and it is obtained using a motion model. If IMU is available, mechanization can be used as the motion model. For wheeled robots, the kinematic model can be used to provide the motion model. Section 3.3 will discuss the IMU mechanization, and Section 3.5 will discuss the kinematic modelling of a differential-drive robot.
2. **Particle weighting** is performed using observation likelihood, which can be measured as the distance between actual and predicted observations. For tightly coupled and loosely coupled methods, different particle weighting methods are used. In Sections 3.3 and 3.4, these weighting methods are defined.
3. **Resampling** is used to duplicate the particles with higher weights and discard particles with lower weights. Resampling often is performed when most of the weights are concentrated in a few particles. In the following sections, the utilized resampling strategy is explained.
4. **Data association** is the process of assigning observations to landmarks. If observations are associated with the wrong landmarks, large errors will be introduced. Therefore, it is important to have a strategy that is robust to false data associations. The data association in this thesis is used to assign ultrasonic rangefinder observations to objects, which is explained in detail in Section 3.6.

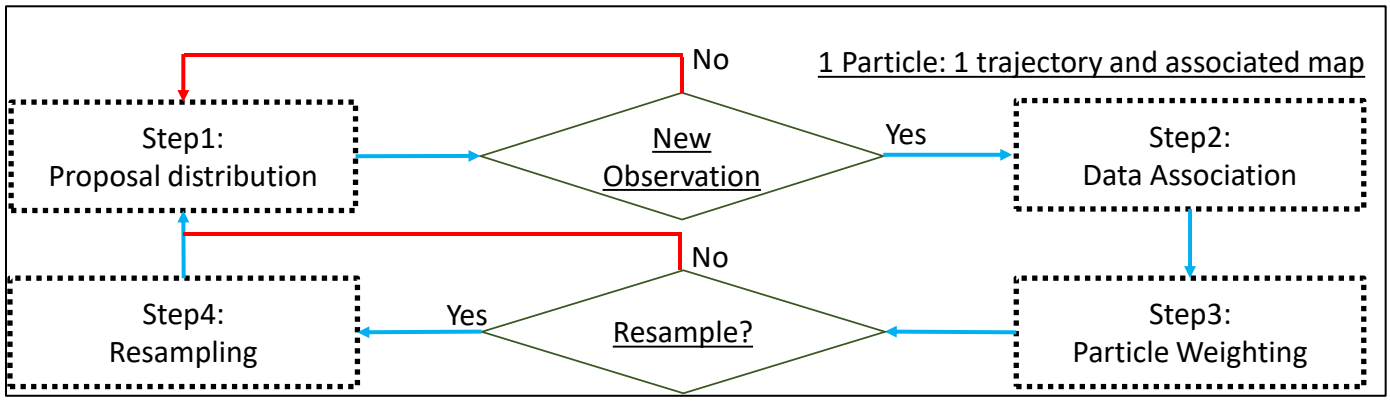


Figure 3.2: The flowchart of a RBPF algorithm.

3.3 Loosely Coupled IMU/Monocular Camera Object-level RBPF SLAM

In the following, an overview of the developed loosely coupled IMU/camera object-level SLAM is given. The details, such as mathematical derivations, are provided later in this section. See Figure 3.3 for the flowchart of the algorithm. The first step is the initialization of the particles, which can be achieved if an image containing at least one object is available. This object is segmented in the image, and the object's pose is estimated in the camera's coordinate frame (object segmentation and pose estimation is explained in Sections 4.2 and 4.4). The estimated pose is transformed to the inertial frame defined as the initial position of the IMU (will be explained later in this section). In order to generate the initial proposal distribution, particles are sampled around the estimated pose. The exact details of the initialization strategy differ if an ultrasonic rangefinder is available or not. Without such a sensor, samples should be taken with a larger uncertainty in the direction from the object toward the camera. However, if the ultrasonic sensor provides distance to the object, the uncertainty in all three directions can be assumed to be equal.

In the next step, the pose of the particles is predicted using IMU mechanization. In the particle proposal, noise is added to the output from mechanization. In order to weigh these particles, at least one object should be observed. Object segmentation is applied to the image, and if there is an object in the image, the camera's pose is estimated with respect to the object. This pose is denoted as the **observed pose**. If this object is already initialized in the map (its coordinates are known in the inertial frame), it is also possible to estimate the camera's pose in the object's frame independently using the particle proposals. This pose is denoted as the **predicted pose**. In order to weigh the particles, the distance between the observed and the predicted poses is measured. The loosely coupling of the IMU and monocular camera here refers to the fact that particle weighting happens only after an independent pose estimation using each of the sensors. If the object has not been initialized yet on the map, a different process should be followed. Similar to the initialization phase, the object's pose is estimated in the camera's frame, and samples are

drawn around this pose. The sampled object poses are added to the map. If the object is initialized in the current epoch (and no object is available to be segmented in the image), weighting and resampling cannot be performed. In these cases, since all particles are propagated to the next frame without weighting, the uncertainty increases. If no objects were observed for a period of time, the errors could accumulate as the solution only relies on the IMU.

Finally, resampling can be performed after every weight update. Alternatively, a test can be used to determine if resampling should be done. This approach follows research literature such as (Grisetti et al., 2005), where resampling is performed if the weights are accumulated on a small percentage of the particles.

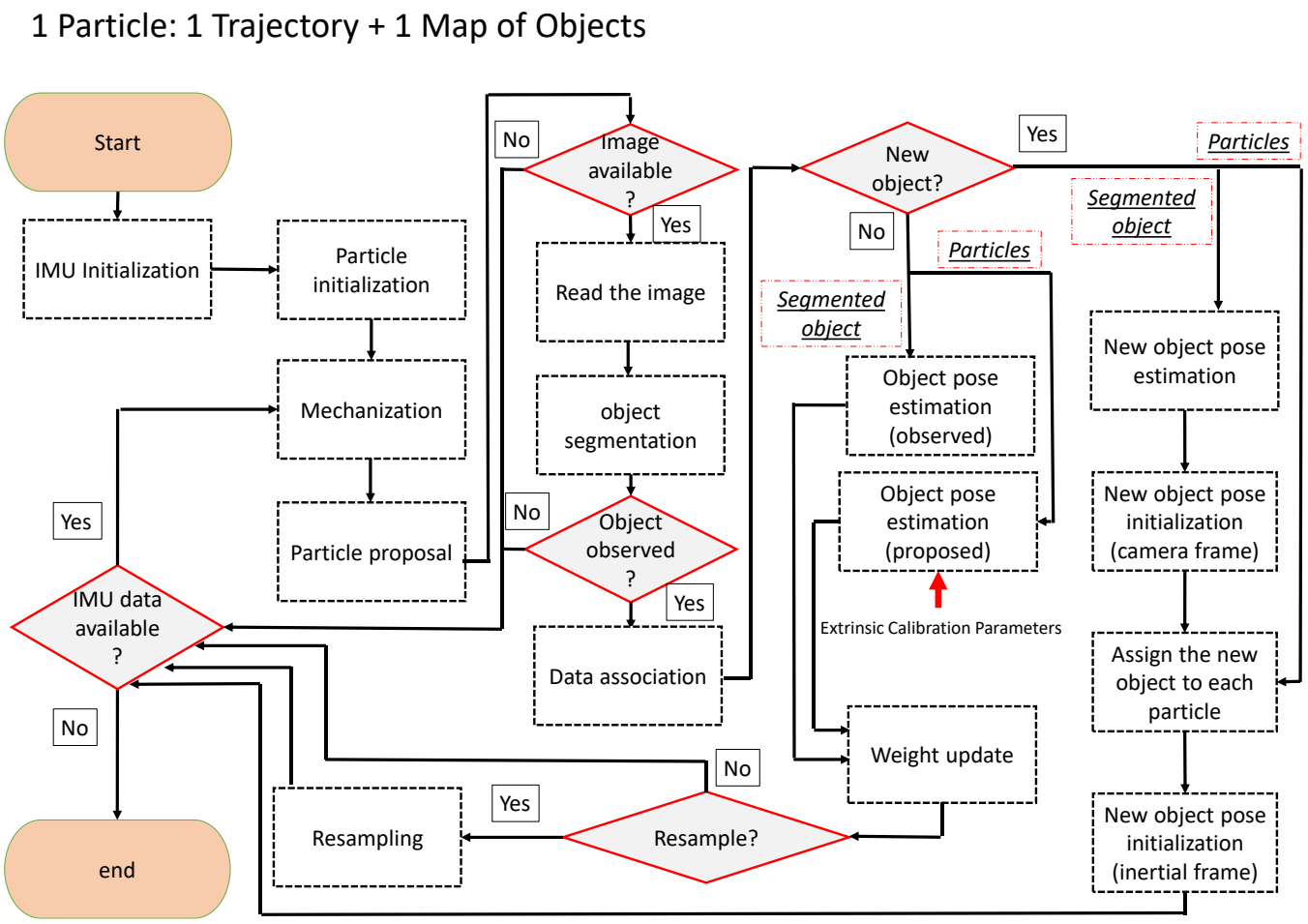


Figure 3.3: Flowchart of the proposed IMU/monocular loosely-coupled object-level SLAM

In order to explain the developed method mathematically, mechanization using IMU is explained first. The difference between the developed mechanization and the typical mechanization in the Local Level Frame (LLF) and Earth Centered Earth-Fixed frame (ECEF) (Noureldin et al., 2012) is that several simplifications can be considered since the device (or robot) navigates a small indoor environment. It is important to note that more elaborate mechanization methods can also be utilized if the robot is navigating in larger indoor or outdoor environments. Figure 3.4 shows the mechanization in the inertial frame (an inertial frame is a frame that can be assumed to have a constant velocity). The following assumptions and differences are made to simplify the mechanization process. The first assumption is that the inertial frame is the first frame (the frame before the devices are moved). This assumption is inaccurate since it neglects the earth's rotation. However, it is assumed that the earth's rotation cannot be sensed reliably by low-cost IMU sensors and, therefore, can be neglected in the developed solution. The second difference is in the estimation of the gravity vector in the inertial frame. In the developed approach, the gravity vector is assumed to be the only external force that applies to the device in the initial frame and is corrected in the following frames. The gravity in the initial frame is estimated by keeping the device still for 10-15 seconds and averaging the accelerometer's reading. The initial rotation is set to an identity matrix.

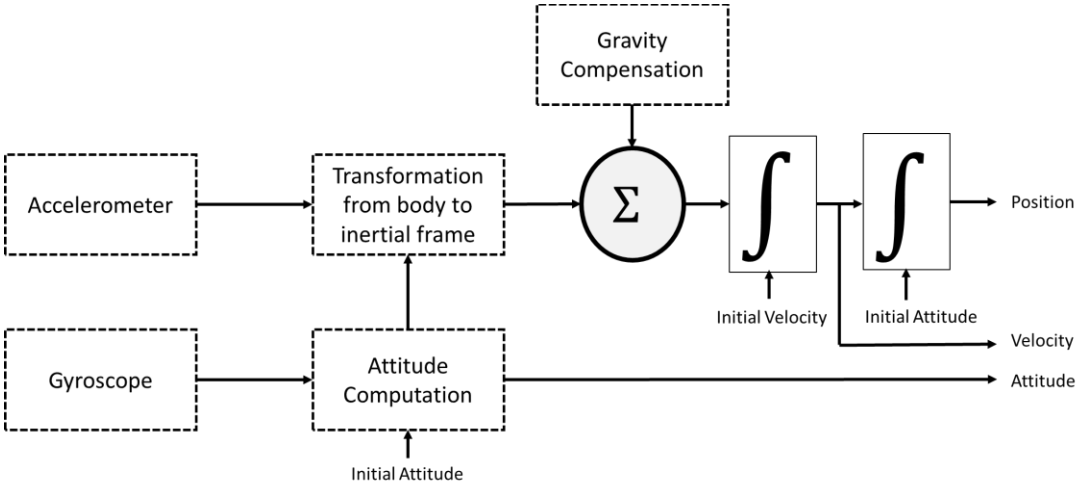


Figure 3.4: Flowchart of IMU mechanization in an inertial frame

Equation 3.4 shows the transformation of the accelerometer to the inertial frame. The symbol R_b^i denotes the transformation from the body frame (b) to the inertial frame (i). With the previous simplifications, in the initial frame, this matrix can be set to identity. In Equation 3.5, the velocities' rate can be computed by compensating for the gravity. The gravity vector g^e is not computed directly. Rather $R_e^i g^e$ is set to the measured gravity vector in the initial (inertial) frame. The rate of the change of the rotation matrix R_b^i is given in Equation 3.6 where Ω_{ib}^i is a skew-symmetric matrix that is built using the angular velocity reading provided by the gyroscopes. The angular rate is shown as the vector $[\omega_x, \omega_y, \omega_z]'$, denoting the rate of rotation around three axes of IMU (the symbol $(.)'$ denotes the transpose of a vector). The skew-symmetric of such a vector is shown in Equation 3.7.

$$f^i = R_b^i f^b \tag{3.4}$$

$$\dot{v}^i = R_b^i f^b + R_e^i g^e \tag{3.5}$$

$$\dot{R}_b^i = R_b^i \Omega_{ib}^i \tag{3.6}$$

$$\Omega_{ib}^i = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.7}$$

In order to discretize the equations above, the following steps are taken. For the velocity and the position, Equations 3.8 and 3.9 can be used. In these equations, the position vector in the inertial frame is shown as r_k^i (the subscript k denotes the epoch, and Δt denotes the time step). Similarly, the velocities are denoted as v_k^i. For the rotation matrix, Equation 3.10 can be used for discretization (where the S matrix is defined as in Equation 3.11). The magnitude of the rotation is denoted as $|\theta|$ in Equation 3.10 and is defined in Equation 3.12.

$$r_{k+1}^i = r_k^i + \Delta t((v_{k+1} + v_k)/2) \tag{3.8}$$

$$v_{k+1}^{i} = v_{k}^{i} + \Delta t \dot{v}^{i} \tag{3.9}$$

$$R_{b,k+1}^{i} = R_{b,k}^{i}(I + \frac{\sin(|\theta|)}{|\theta|}S + \frac{1-\cos(|\theta|)}{|\theta|^2}S^2) \tag{3.10}$$

$$S = \Delta t \Omega_{ib}^{i} \tag{3.11}$$

$$|\theta| = \Delta t \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{3.12}$$

The mechanization will provide an estimate of the velocities and the attitude (represented as a rotation matrix, Euler angles and so on) of the device in the inertial frame. The mechanization should be repeated for each particle. In order to include process noise, two approaches are utilized in this thesis. The first approach is based on using the calibration of accelerometers and gyroscopes, which can provide the stochastic characteristics of the sensors. In this approach, samples can be taken from a Gaussian distribution with a known noise variance for each sensor. This value can then be added to the raw reading, and finally, the process of mechanization (as explained above) can be repeated for each particle. The second approach is to use error models such as constant velocity, where the noise can be added after the mechanization of the velocities. Both of these approaches are tested in this thesis.

In the loosely coupled scheme, the predicted pose from the particle proposal should be compared to the observed pose estimated using the current image. In order to achieve this, the two poses should be transformed into a single frame. The observed camera's pose is in the object frame, and as mentioned earlier, this pose is estimated directly using the segmented object in the image. Therefore, it is independent of the particle's pose (the pose estimation is explained in Sections 4.4 to 4.6). This observed camera's pose in the object frame is denoted by a homogeneous matrix H_c^o. The overall structure of a homogeneous matrix is shown in Equation 3.13, which includes both a rotation matrix (R) and a translation vector (r). The following equations aim to show how to transform the estimated pose of the camera from the mechanization (which is in the inertial frame) to the object's frame.

$$H = \begin{bmatrix} R & r \\ \mathbf{0_{1x3}} & 1 \end{bmatrix} \tag{3.13}$$

With The help of mechanization, we know the transformation from IMU's frame to the inertial frame (denoted as H_b^i). It is assumed that IMU's frame coincides with the body frame, and the subscript b is used to denote IMU's frame. Furthermore, the extrinsic calibration parameters of the camera and IMU are known, and it is denoted as H_c^b. Therefore, it is possible to derive the transformation from the camera to inertial frame using the proposal distribution. This is denoted as $\widetilde{H}_c^i$ as shown in Equation 3.14 (the symbol $\widetilde{(.)}$ denotes predicted values). If the object is initialized in the map, the transformation from the object's frame to the inertial frame is also known (H_o^i); therefore, Equation 3.15 can be used to derive the predicted pose of the camera in the object's frame.

$$\widetilde{H}_c^i = H_b^i H_c^b \tag{3.14}$$

$$\widetilde{H}_c^o = H_i^o \widetilde{H}_c^i \tag{3.15}$$

In order to weigh the particle, Equation 3.3 can now be used. The weights, as mentioned, are proportionate to the observation likelihood. In the loosely coupled scheme, this observation likelihood is not calculated directly, but it is approximated using Equation 3.16.1. In Equation 3.16.1, the updated weight ($w_{k+1}^{[n]}$) is calculated using the current weight of the particle ($w_k^{[n]}$) multiplied by an exponential proportional to the distance vector (Δe). The distance vector is calculated between the estimated position of the camera using the mechanization ($\tilde{r}_{k+1}^{o,[n]}$, estimated with $\widetilde{H}_c^o$) and the estimated position of the camera using object pose estimation (r_{k+1}^o, estimated with H_c^o) (see Equation 3.16.2). This equation assumes that the errors in the observation likelihood of a single particle follow a Gaussian distribution; however, it is important to note that from this assumption, it does not follow that the distribution of particles is Gaussian. Similar weighting methods were used in the past for point-based RBPF-SLAM (Montemerlo & Thrun,

2003). In Equation 3.16.1, all the constants are included in the term η, and since the weight of the particles should sum to one, it is not required to calculate the constants explicitly for each particle. The covariance matrix ($\Sigma_{\Delta e}$) can be set experimentally. The details about how to set values of covariance appropriately are provided in the following.

$$w_{k+1}^{[n]} = \eta\, w_k^{[n]} \left[exp(-\frac{1}{2} \Delta \acute{e}_{k+1}^{[n]} \Sigma_{\Delta e}^{-1} \Delta e_{k+1}^{[n]}) \right] \tag{3.16.1}$$

$$\Delta e_{k+1}^{[n]} = \tilde{r}_{k+1}^{o,[n]} - r_{k+1}^{o} \tag{3.16.2}$$

The abovementioned method can be used when the pose estimation using the camera is also known up to the correct scale. Unfortunately, without a rangefinder such as an ultrasonic sensor, it is not possible to estimate the object's pose with the correct scale (unless the size of the object is assumed to be known). Therefore r_{k+1}^{o} will only be estimated up to an unknown scale while $\tilde{r}_{k+1}^{o,[n]}$ has a scale. In this case, an alternative approach based on the unit vectors is shown in Equation 3.17. The unit vectors can be derived by normalizing the vectors. For the normalizations, each element of the vector is divided by the magnitude of the vector (denoted with symbol $||.||$).

$$\Delta \hat{e}_{k+1}^{[n]} = \tilde{r}_{k+1}^{o,[n]} / ||\tilde{r}_{k+1}^{o,[n]}|| - r_{k+1}^{o} / ||r_{k+1}^{o}|| \tag{3.17}$$

Setting the values for the covariance matrix ($\Sigma_{\Delta e}^{-1}$) is challenging using the normalized vector since the distance $\Delta \tilde{e}_{k+1}^{[n]}$ is difficult to interpret. One solution is to measure the angle between the two vectors, as shown in Equation 3.18. The angle between the two vectors can be understood as shown in Figure 3.5. We expect this angle not to be larger than 5° to 10°; however, setting this value depends on the IMU errors and the errors in the pose estimation generally. Overall, one of the issues with the loosely coupled method is the difficulty of determining the covariance matrix associated with the uncertainties in $\Delta \hat{e}_{k+1}^{[n]}$.

$$\Delta \hat{e}_{k+1}^{[n]} = atan2\ (\tilde{r}_{k+1}^{o,[n]} / ||\tilde{r}_{k+1}^{o,[n]}||, r_{k+1}^{o} / ||r_{k+1}^{o}||\ ) \tag{3.18}$$

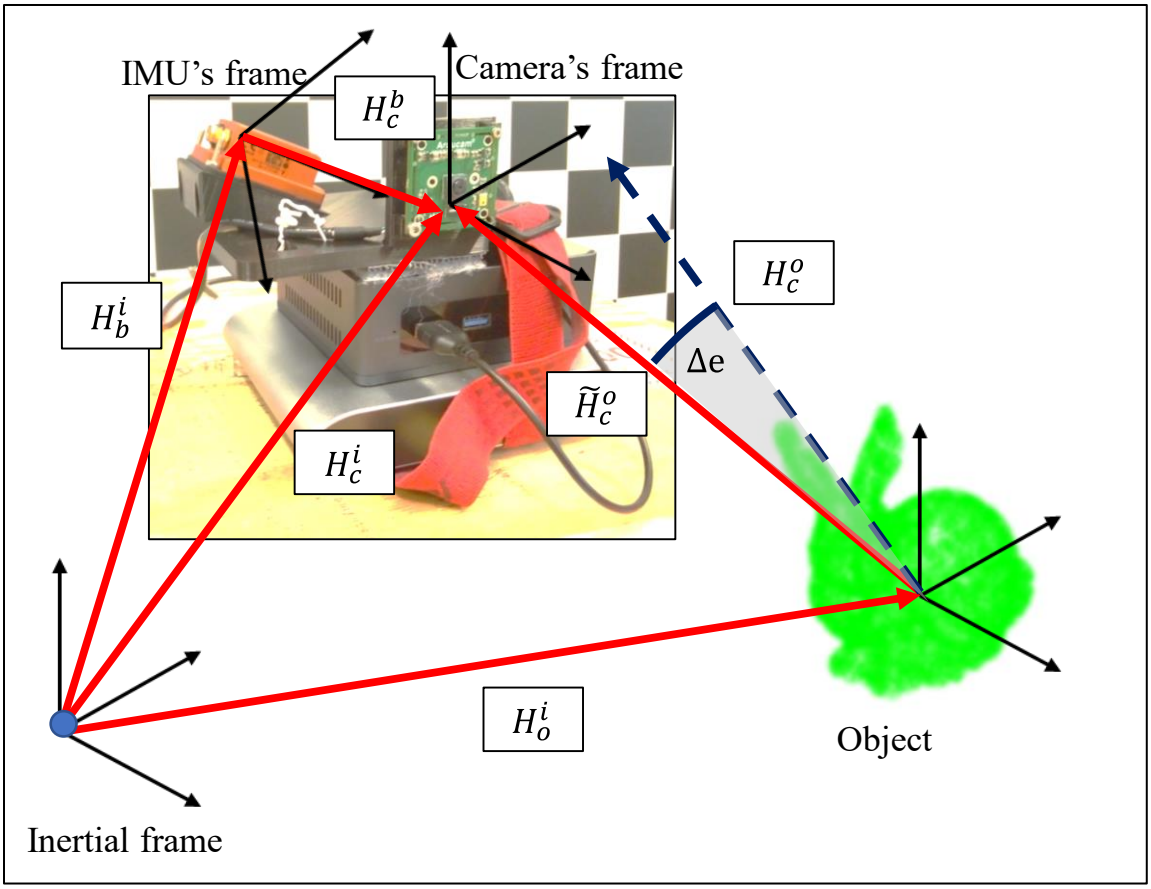


Figure 3.5: Illustration of the estimated and observed object pose error in loosely coupled object-level SLAM.

3.4 Tightly Coupled IMU/Monocular Camera Object-level RBPF SLAM

The tightly coupled approach has many common procedures with the loosely coupled method. Here, the repetition is avoided, and only the differences are highlighted. In the developed tightly coupled IMU/camera object-level SLAM, the particles are weighed using the observation likelihood. The likelihood is proportionate to the distance between the predicted ($\tilde{z}$) and the observed contour (z). The distance between points on the contour cannot be measured directly. This is because the correspondence between the points on the predicted and the observed contours is unknown. Therefore, instead of a point-to-point distance, the Intersection over Union (IoU) area of the two contours is used as the **similarity measure.** This measure is inversely proportional to the distance. Similar approaches have been used to

define the distance between two contours in the past (Tsai et al. 2003). The IoU can be 1 at maximum when two contours exactly coincide, while it is 0 when one contour is completely outside of another.

In order to obtain the predicted contour ($\tilde{z}_{k+1}^{[n]}$), the 3D points on the object's surface are projected onto the image plane (via the predicted pose of the object in the camera's frame). The predicted contour is estimated as the boundary around these points, similar to some methods in the past (Rosenhahn et al., 2007). In order to obtain the observed contour (z_{k+1}), object segmentation is used (explained in Section 4.2). Figure 3.6 shows a schematic of the explained process. In the tightly coupled method, the weights of the particle are updated using Equation 3.19. The distance ($d(\tilde{z}_{k+1}^{[n]}, z_{k+1})$) is defined as the inverse of IoU shown in Equation 3.20. The flowchart of the tightly coupled method is shown in Figure 3.7.

$$w_{k+1}^{[n]} = \eta\, w_k^{[n]} \left[exp\left(-\frac{1}{2\sigma_d} \left(d\left(\tilde{z}_{k+1}^{[n]}, z_{k+1} \right) - 1 \right)^2 \right) \right] \tag{3.19}$$

$$d\left(\tilde{z}_{k+1}^{[n]}, z_{k+1} \right) = (IoU(\tilde{z}_{k+1}^{[n]}, z_{k+1}))^{-1} \tag{3.20}$$

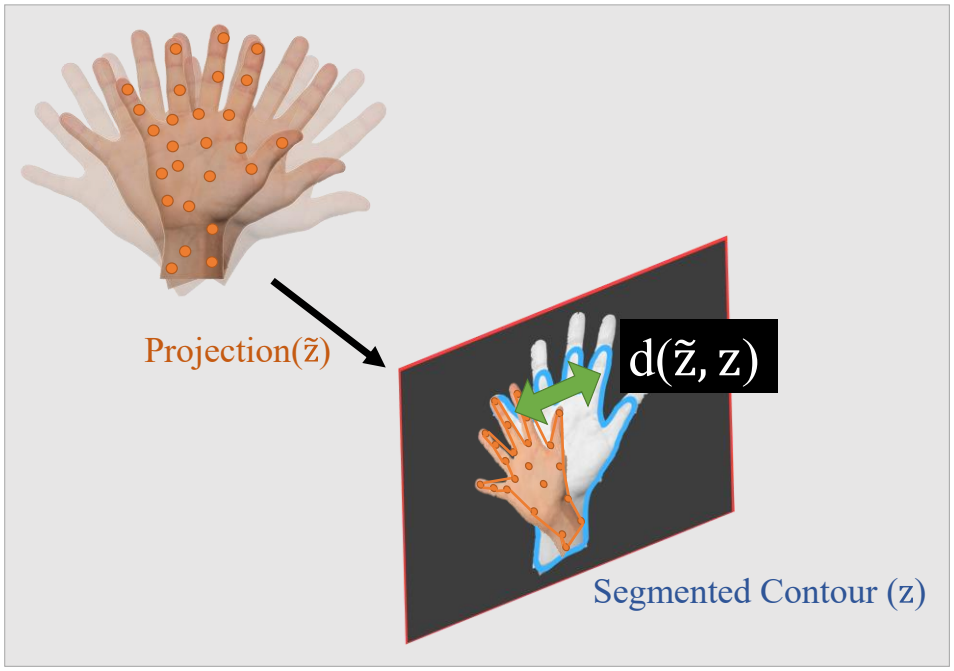


Figure 3.6: Particles are weighted using the distance between the predicted and observed contour of the object.

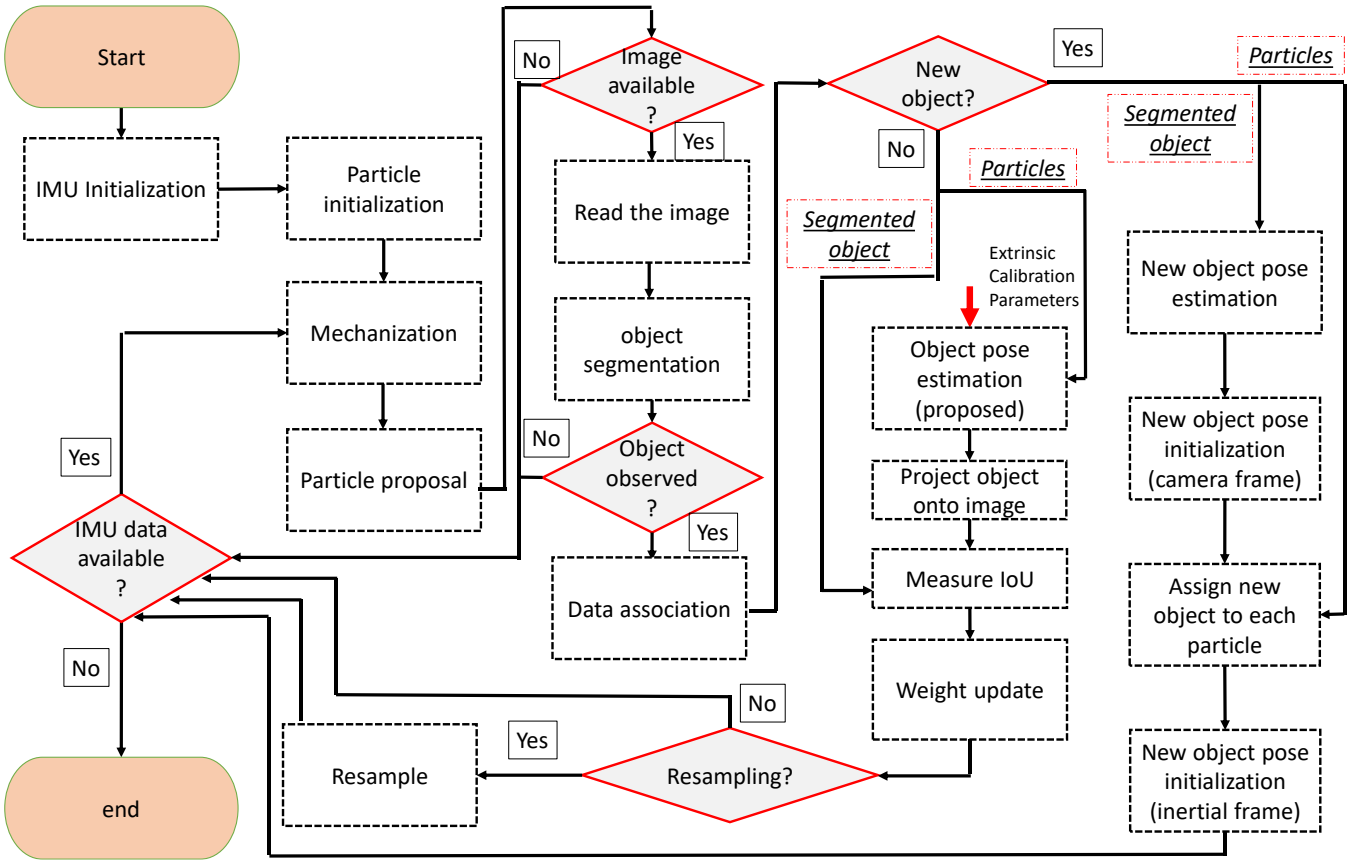


Figure 3.7: Flowchart of the proposed IMU/monocular tightly-coupled object-level SLAM

The tightly coupled method, as it will be shown in Chapter 7, can produce a more accurate solution than the loosely coupled. However, particle weighting in the tightly coupled can be computationally expensive. The following should be taken into consideration for real-time implementation.

1. Initially, only the centroid of the 3D points on the object's surface can be projected onto the image. If the centroid does not fall close to the segmented object, a small weight can be immediately assigned to the corresponding particle without further computations.
2. For all the remaining particles, all the 3D points are projected onto the image, and the centroid is estimated again. Similar to the previous step, if the distance between the estimated and segmented

object's centroids is above a certain threshold, the corresponding particle is assigned a small weight and further computations are avoided.

3. It is important to note that not all the points in the 3D model are required to be projected onto the image for a reasonably accurate estimation of the predicted contour. Therefore, it is possible to down-sample the 3D points of the object to further decrease the computational burden.

Figure 3.8 shows the developed fast method to measure the IoU for each particle. In order to measure IoU, the following steps are taken. First, the center of 3D points on the object's surface is projected onto the image using the estimated particle's pose (assuming the camera's calibration matrix is known). It is also assumed that the segmented object in the image is available. Then, the distance between the projected and segmented object's center is measured to test whether it is smaller than a threshold. If the distance is smaller, then the down-sampled object's 3D points are projected onto the image and the boundary of the projected points is found. There are many approaches to finding such boundaries. In this thesis, a MATLAB function designed for this task is utilized. In the next step, the projected boundary points are rasterized (they are inserted into an image where pixels corresponding to the boundary are assigned as 1 and the remaining pixels are assigned 0). In order to ensure that the rasterized boundary is connected, morphological dilation is used. Finally, the inside of this connected boundary can efficiently be assigned with 1 using the known morphological operation of hole filling (Woods & Gonzalez, 2008). The mentioned steps of detecting boundary, rasterizing and filling the image can also be repeated for the segmented object. Since the output of DL is already a binary image, it is unnecessary to go through the same steps, and this process can be skipped. Finally, an additional step of size reduction is also performed. Size reduction can greatly reduce the computational cost of measuring IoU, and in our experiments, it did not affect the performance of the tightly coupled method. Once the two masks are available logical operations on binary images can be used to calculate IoU.

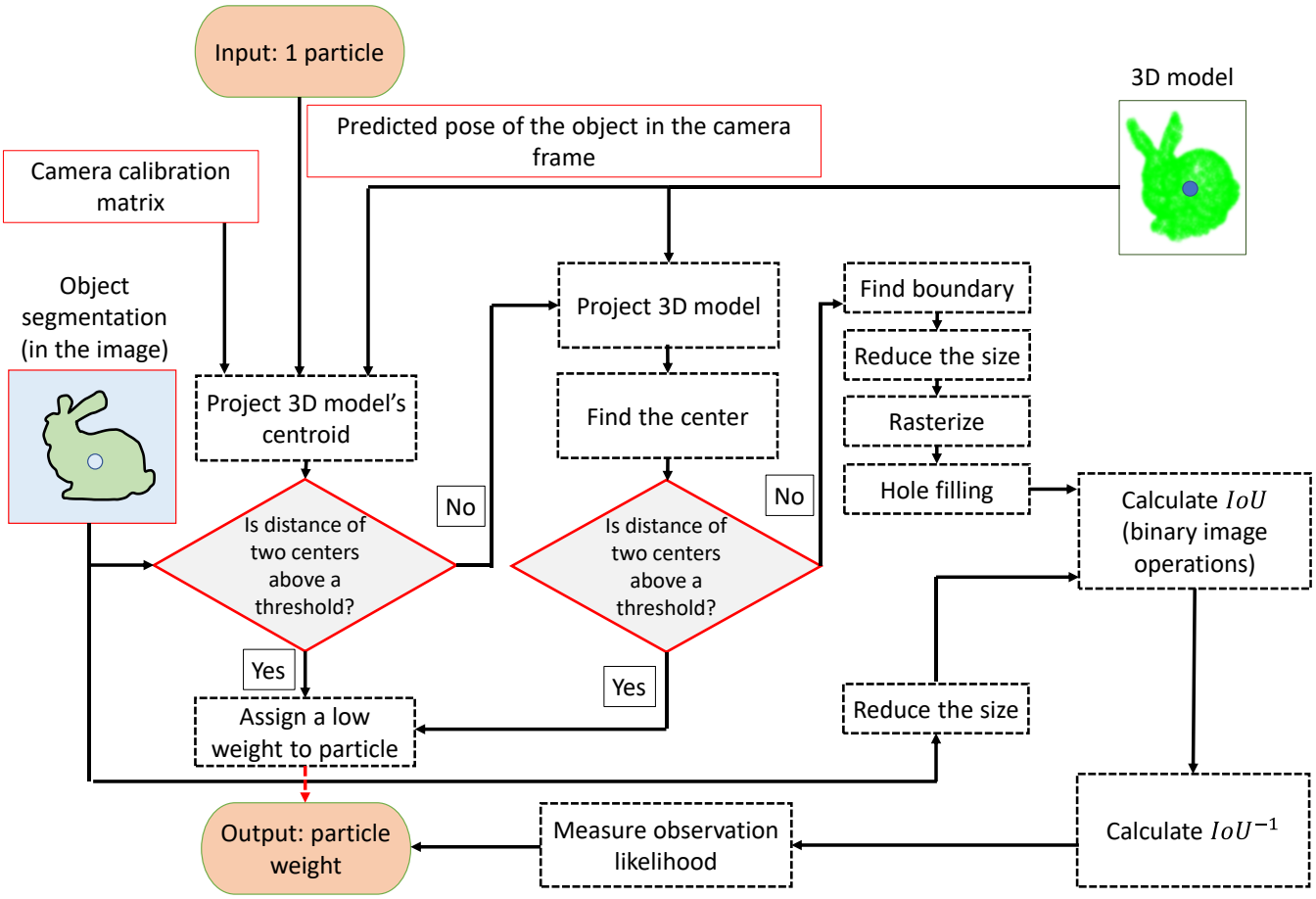


Figure 3.8: Flowchart of the process particle weighting in tightly-coupled object-level SLAM

3.5 Wheel Odometry/Monocular Camera Object-level SLAM

For the wheeled robot, loosely and tightly coupled methods are implemented similarly to the previous section. Here, only the differences are highlighted. For the wheeled robot, the mechanization can be replaced with the kinematic model. The kinematic model is a set of equations explaining the motion of a robot using velocities. Since the utilized robot in this thesis is a differential-drive robot (see Chapter 6), the kinematic modelling for this type of robot is explained in this section.

In a differential-drive robot, often two fixed wheels exist (free wheels in the front or back are considered to not have significance on the kinematic modelling). In order to derive the kinematic equations, the steps in (Siegwart et al., 2011) are followed (more details are included here). In this

approach, the velocities in the body frame(b) are projected onto the direction of the motion of each wheel. For a given **fixed wheel**, Figure 3.9 depicts how the kinematic model can be derived. This figure includes the main directions and angles to illustrate the projection (additional line segments are included for convenience). The movement of the fixed wheel instantaneously is constrained to the rolling direction, and perpendicular to this is the direction of slippage assumed here to have 0 velocity component. The vector from the center of the robot (P) to the center of the wheel is denoted as l (its length is denoted as L). The symbol α corresponds to the angle between l and X^b, and symbol β corresponds to the angle between l and the axis of rotation of the wheel. In Figure 3.9, the green line is parallel to the rolling direction. The dashed blue circle corresponds to the circular motion centred at the origin of the body frame. The red line shows the tangent of the circular motion.

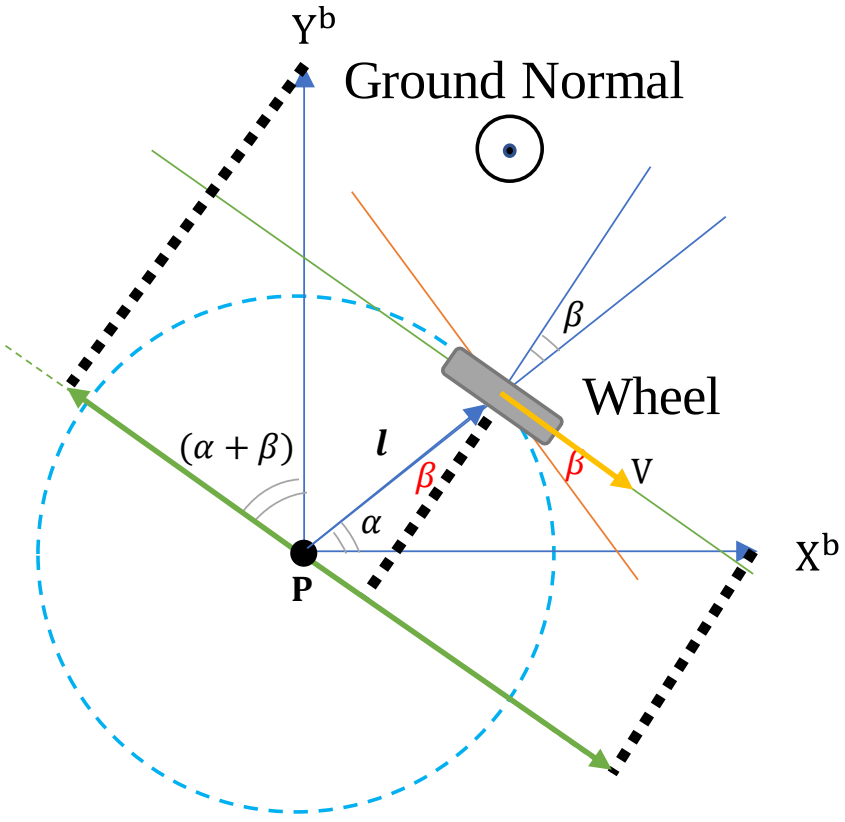


Figure 3.9: Illustration for calculating velocities in the direction of motion of a fixed wheel

For a differential-drive robot, the motion is constrained to a plane and can be defined using three states (position and heading). To find the motion along the rolling direction, the first step is to calculate the projection of the three velocities (two linear and one rotational) in the body frame onto the rolling direction of the wheel (yellow arrow in Figure 3.9). If the state in the body frame is denoted as a vector ξ^b and the corresponding velocities are denoted as $\dot{\xi}^b([\dot{x}^b, \dot{y}^b, \dot{\theta}]')$, projecting $\dot{x}^b$ and $\dot{y}^b$ onto the rolling direction, Equations 3.21.1 and 3.21.2 are obtained. These equations can be verified using Figure 3.9 (see how $\dot{x}^b$ and $\dot{y}^b$ project onto the green axis). The last component is the velocity due to the rotational motion of the robot. The direction of this motion is tangential to the circle in Figure 3.9 and has two components. One component is projected along the green axis, and the other component is projected along the axis of the rotation of the wheel (the wheel cannot move in this direction). The component along the green axis is shown in Equation 3.21.3.

$$\dot{x}^b cos(\tfrac{\pi}{2} - (\alpha + \beta)) = \dot{x}^b sin(\alpha + \beta) = \dot{\xi}_1^b sin(\alpha + \beta) \quad (3.21.1)$$

$$-\dot{y}^b cos(\alpha + \beta) = -\dot{\xi}_2^b cos(\alpha + \beta) \quad (3.21.2)$$

$$\dot{\theta}(-Lcos(\beta)) = \dot{\xi}_3^b(-Lcos(\beta)) \quad (3.21.3)$$

Assuming that there is no slippage motion (only rolling motion), the rotational motion generated by the motors ($D\dot{\varphi}$) should be equal to the summation of Equations 3.21.1, 3.21.2 and 3.21.3. This is expressed in Equation 3.22, where D is the radius of the wheel, and φ is the angular velocity of the wheel (known from the wheel encoder). The goal is to estimate the velocities of the robot in the inertial frame ($\dot{\xi}^i$). The transformation of the velocity state vector from the body to the inertial frame is shown in Equation 3.23. The rotation matrix (R_i^b) is defined in Equation 3.24, where θ^i is the robot's heading in the inertial frame.

$$[sin(\alpha + \beta) \ -cos(\alpha + \beta) \ -Lcos(\beta)]'\dot{\xi}^b - D\dot{\varphi} = 0 \quad (3.22)$$

$$\dot{\xi}^b = R_i^b \dot{\xi}^i \tag{3.23}$$

$$R_i^b = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

In order to obtain the complete kinematic model, Equation 3.22 should be considered for each fixed wheel. Since the differential-drive robot has two fixed wheels, two equations are required. The values α and β can be replaced appropriately for the robot that is used in the experiments (a schematic is provided in Figure 3.10). Equation 3.25 shows the stacked equations where the unknowns are moved to the left-hand side. The inverse of the matrix shown in the bracket on the right-hand side of Equation 3.25 does not exist, and the Moore-Penrose pseudoinverse can be used instead (denoted using the symbol $(.)^+$). The results are shown in Equation 3.26. It is possible to rewrite Equation 3.26 in terms of the forward and angular velocity of the robot in the body frame, as shown in Equation 3.27.

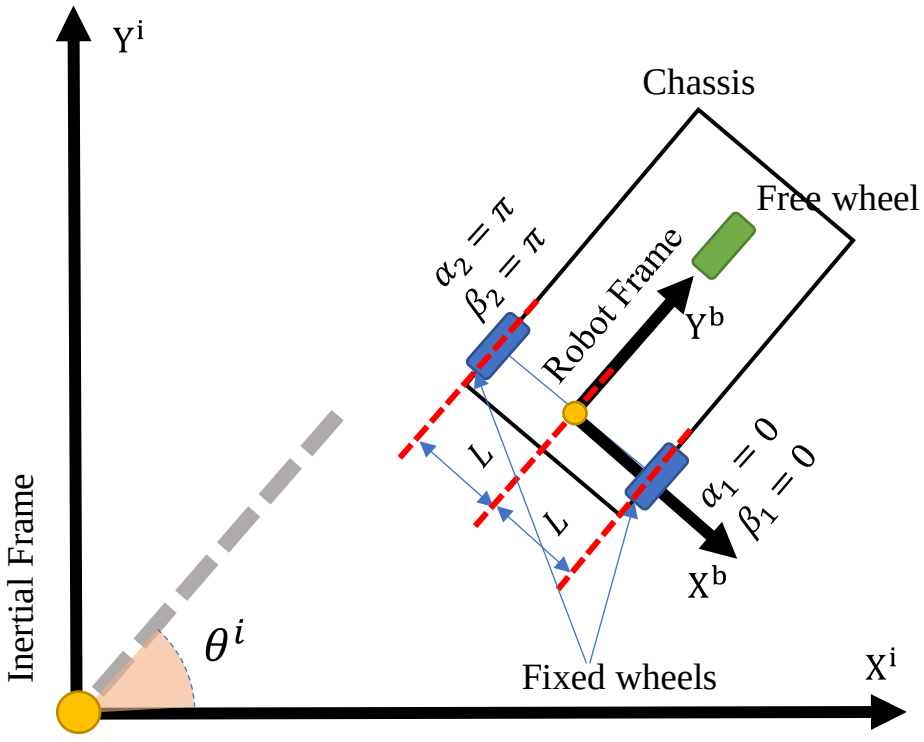


Figure 3.10: Illustration of the parameters of the fixed wheels of a different wheel robot.

$$\dot{\xi}^i = \left[\begin{bmatrix} 0 & -1 & -L \\ 0 & -1 & L \end{bmatrix}\begin{bmatrix} cos(\theta^i) & sin(\theta^i) & 0 \\ -sin(\theta^i) & cos(\theta^i) & 0 \\ 0 & 0 & 1 \end{bmatrix}\right]^+ \begin{bmatrix} D\dot{\varphi}_1 \\ D\dot{\varphi}_2 \end{bmatrix} \tag{3.25}$$

$$\dot{\xi}^i = \begin{bmatrix} sin(\theta^i)/2 & sin(\theta^i)/2 \\ -cos(\theta^i)/2 & -cos(\theta^i)/2 \\ -1/(2L) & 1/(2L) \end{bmatrix}\begin{bmatrix} D\dot{\varphi}_1 \\ D\dot{\varphi}_2 \end{bmatrix} \tag{3.26}$$

$$\dot{\xi}^i = \begin{bmatrix} cos(\theta^i) & -sin(\theta^i) & 0 \\ sin(\theta^i) & cos(\theta^i) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -(D\dot{\varphi}_1 + D\dot{\varphi}_2)/2 \\ -(D\dot{\varphi}_1 - D\dot{\varphi}_2)/2L \end{bmatrix} = \begin{bmatrix} sin(\theta^i) & 0 \\ -cos(\theta^i) & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.27}$$

The kinematic model in Equation 3.27 can be converted to the standard form in Equation 3.28.1. To discretize the state space, Equation 3.29 can be used. The complete derivation for the discretized space is given in Appendix A. Since A_c is all zeros, the matrix $e^{A_c \Delta t}$ is the identity matrix. It is also assumed for simplicity that B_c is constant during one epoch. However, since B_c in fact varies with θ^i, this assumption is more accurate for shorter time intervals (Δt). Equation 3.30.1 shows the derived discretized kinematic model and can replace mechanization in Sections 3.3 and 3.4.

$$\dot{\xi}^i = A_c \xi^i + B_c u \tag{3.28.1}$$

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad A_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_c = \begin{bmatrix} sin(\theta^i) & 0 \\ -cos(\theta^i) & 0 \\ 0 & -1 \end{bmatrix} \tag{3.28.2), (3.28.3), (3.28.4}$$

$$\xi_{k+1} = e^{A_c \Delta t}\xi_k + \int_k^{(k+1)\Delta t} e^{-A_c(t-(k+1)\Delta t)} B_c u \, dt \tag{3.29}$$

$$\xi_{k+1} = A_d \xi_k + B_d u_k \tag{3.30.1}$$

$$u_k = u \qquad A_d = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B_d = B_c(\Delta t) \tag{3.30.2), (3.30.3), (3.30.4}$$

3.6 Object-level RBPF-SLAM with Ultrasonic Rangefinder

RBPF-SLAM was originally implemented using rangefinders (Montemerlo et al., 2003), where it was assumed that the distance from the sensor to the observed points is approximately known. Unfortunately, a monocular camera does not observe the distance to the points, and the uncertainty of the point's distance from the camera using a single image is large. There are two approaches to initializing objects. In delayed initialization, the object is only inserted into the map once it is observed in many images, while in undelayed initializations, the object is inserted into the map once it is observed in a single image. In Section 2.6, it is argued that undelayed initialization is more advantageous than delayed initialization.

In this thesis, two undelayed initialization methods are used. The first approach relies only on the fusion of the camera with an IMU (or wheel odometer). With this approach, the uncertainty of the object's pose is high with one observation. However, utilizing two or three observations can reduce the uncertainty in the object's pose. The reduction in the uncertainty is due to the IMU's capacity to estimate the distance between camera poses, which will eventually resolve the scale ambiguity. This process might take a long time, and many particles should be used to represent such uncertainty in the initialization.

The second undelayed initialization is based on the fusion of the monocular camera with a rangefinder. In the past, the data from accurate laser rangefinders were often fused with the monocular camera. One difference between object-level and classical solutions to the SLAM is that the objects correspond to volumetric bodies and occupy a larger number of pixels in the images. Thus, it is likely that less accurate rangefinders (such as ultrasonic sensors) could return distance readings from some parts of the object. The provided distance by the ultrasonic rangefinder can reduce uncertainty in the initialization significantly. In addition to undelayed initialization, the observed distance from a rangefinder can significantly reduce the search space for particles with accurate poses. For instance, particles that fall far

from the measured distance to the object can be assigned with a low weight immediately, without further evaluation of observation likelihood (this is very advantageous for the tightly coupled method where it suffers from computational cost due to estimation of likelihood). It will be discussed in the following how the data from the ultrasonic rangefinder and camera can be fused.

The ultrasonic rangefinder's beam angle depends on the physical characteristics of the sensor and ultrasound wave propagation in the air (Strickland & King, 1993). Nowadays, a common sensor used in many low-end robots is HC-SR04. This sensor approximately has a beam angle of 15° (much higher than the beam angle of a LiDAR). This large beam angle can cause the ultrasonic waves to hit multiple points in the vicinity. Some or most of these points might be a part of the background clutter. In these circumstances, a false data association can occur. Therefore, the beam angle should be taken into consideration in determining whether the wave has reflected back from the object or from the background.

In order to address this issue, a new method is developed in this section that projects the point p (see Figure 3.11), as well as a defined uncertainty region around this point onto the monocular camera's image. Assuming that the extrinsic calibration parameters of the ultrasonic sensor and camera are known (Section 5.4); this uncertainty region can be estimated using rangefinder distance reading (z_d) and the beam angle (ψ) (both of these quantities are known). The beam angle causes a larger uncertainty as the distance between the ultrasonic rangefinder and the point c increases. The uncertainty region can be defined as a spherical volume centred at the point c at a distance z_d from an ultrasonic sensor. The radius of this region (s) can be approximately estimated using Equation 3.31. Figure 3.11 provides more details about this process. The objective is to examine if the projected uncertainty region intersects the segmented object in the image. In order to achieve this, the uncertainty sphere should be projected onto the image.

$$s = z_d / \tan(\psi/2) \tag{3.31}$$

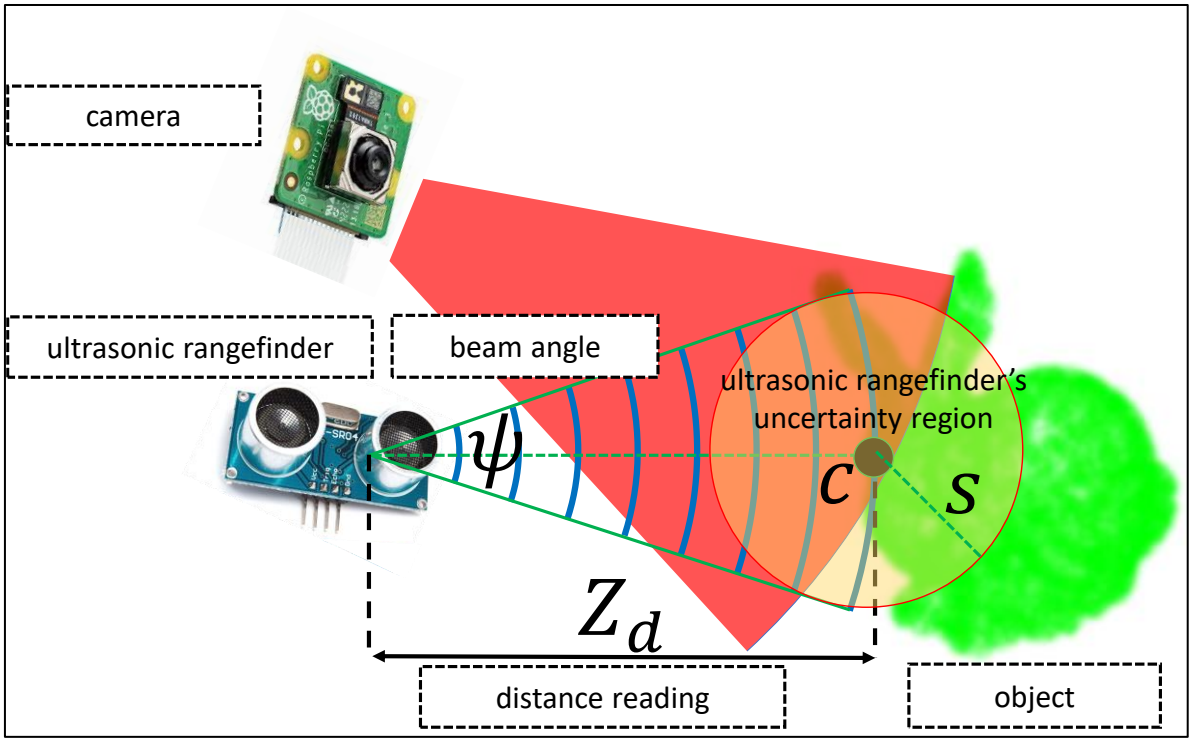


Figure 3.11: Illustration of uncertainty region of the ultrasonic rangefinder reading

In order to project the sphere onto the image, many methods can be used. One possible approach is to project a subset of points on the sphere onto the image. Once the points are projected, an approximate ellipse can be fitted. This ellipse can be used to measure the intersection area with the segmented object in the image. One of three scenarios can occur:

1. The ellipse is completely inside the segmented object.
2. The ellipse is partially inside the segmented object.
3. The ellipse is completely outside of the segmented object.

If cases 1 or 3 occur, accepting or rejecting the rangefinder's reading is easy. Further, it is possible to know which object the distance reading corresponds to. If case 2 occurs, it is still possible that the ultrasonic sensor has measured the distance to the object. In this case, the intersection area over the area inside the ellipse can be used to determine if the distance reading should be accepted or rejected. The flowchart of the algorithm is shown below in Figure 3.12.

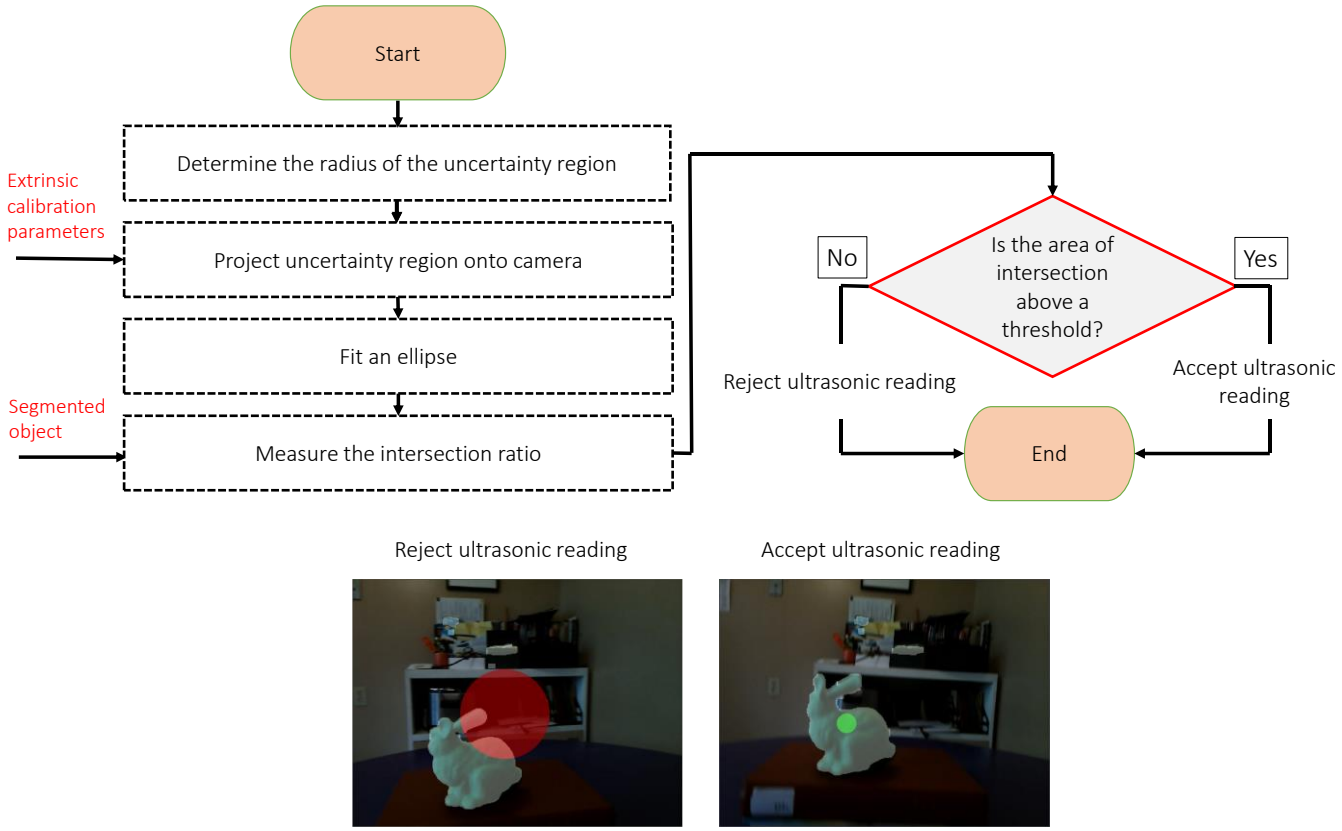


Figure 3.12: The flowchart of accepting/rejecting a distance measurement received from the ultrasonic rangefinder

3.7 Challenges Related to the Observation Likelihood

In this section, some of the challenges regarding RBPF-SLAM due to the observation likelihood will be discussed. Overall, four common Challenges (I-IV) will be discussed in this section. These challenges are selected as the most frequently occurring in developed object-level solutions. As it was mentioned in the prior works (Grisetti et al,. 2005), one challenge (Challenge I) with RBPF-SLAM is that often; observation likelihood has low uncertainty (high precision) while proposal distribution has larger uncertainty (low precision). In order to understand this phenomenon, an example of possible observation likelihood and proposal distribution is shown in Figure 3.13. Such discrepancy between the two distributions can lead to a lack of the particle drawn closer to the peak of the observation likelihood. Most particles will have a lower weight. Therefore the solution can diverge.

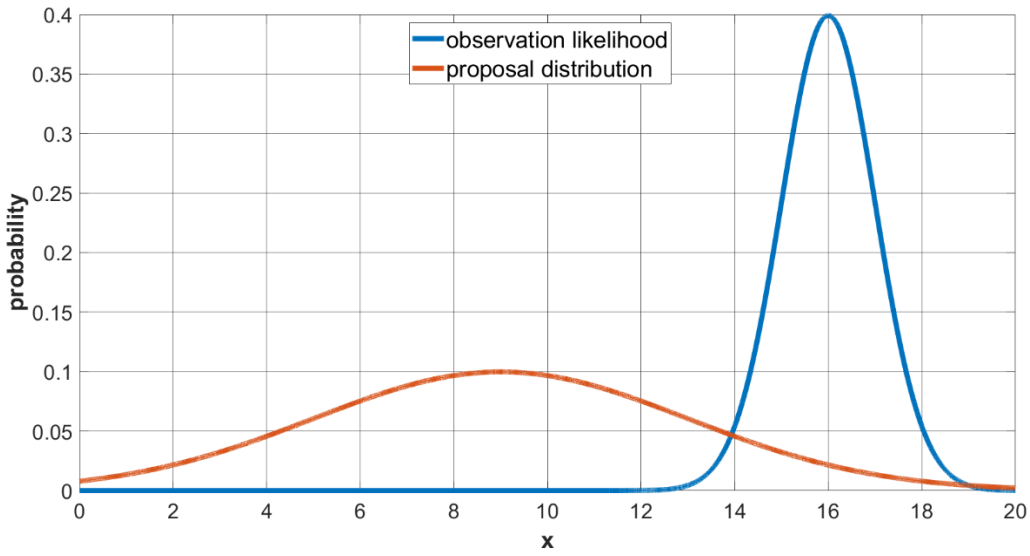


Figure 3.13: A comparison of possible likelihood and the proposal distribution

In order to demonstrate this issue in the context of object-level SLAM, the problem is illustrated schematically in the image domain. In Figure 3.14, given the poses of the object and camera associated with a particle, the object's 3D model is projected onto the image (predicted contour) and compared to the segmented object (observed contour) in the image. Figure 3.14 illustrates that the two particles (p_1 and p_2) have zero weights (since they have zero IoU with the segmented object); however, p_1 is much closer to the segmented object. This figure shows the likelihood is peaking in a region narrowly around the object, which can be difficult to sample as most regions are not covered by this observation likelihood.

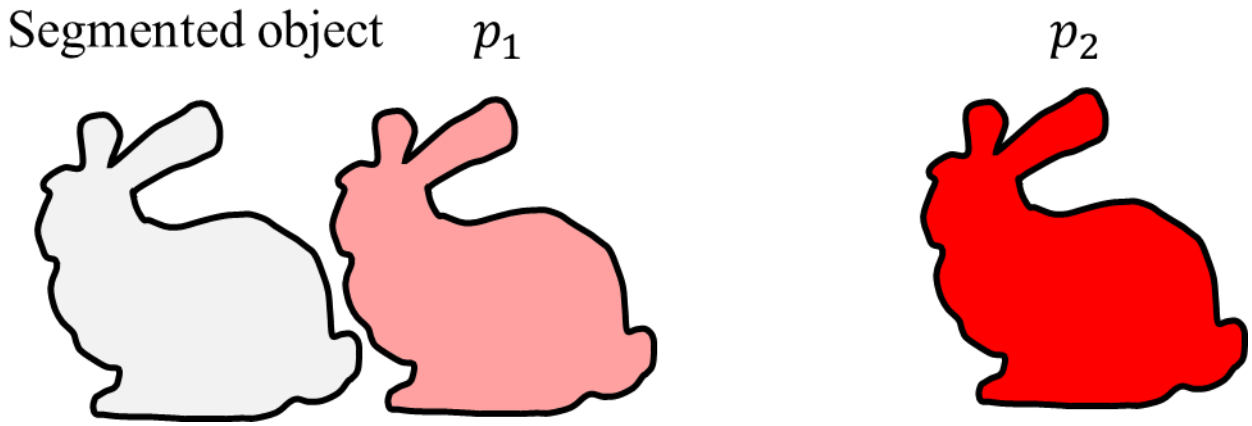


Figure 3.14. Illustration of Challenge I with the observation likelihood

The second challenge (Challenge II) with the observation likelihood can be due to partial occlusion of the object or failure in segmenting a large number of pixels belonging to the object. Figure 3.15 (top row) shows an example where an object is partially occluded (the black rectangle represents an occlusion). Here, the likelihoods of p_1 and p_2 are very similar since the estimated IoU is similar (and low). However p_1 is much closer to the real solution. A similar challenge (Challenge III) occurs when some of the background pixels are segmented as a part of the object, as shown in Figure 3.15 (bottom row). This challenge rarely happens, and it can be due to errors in the object segmentation process. In Figure 3.15 both p_1 and p_2 receive a high weight (as opposed to Challenge II, where both particles receive a low weight), while $p_1$is much closer to the solution.

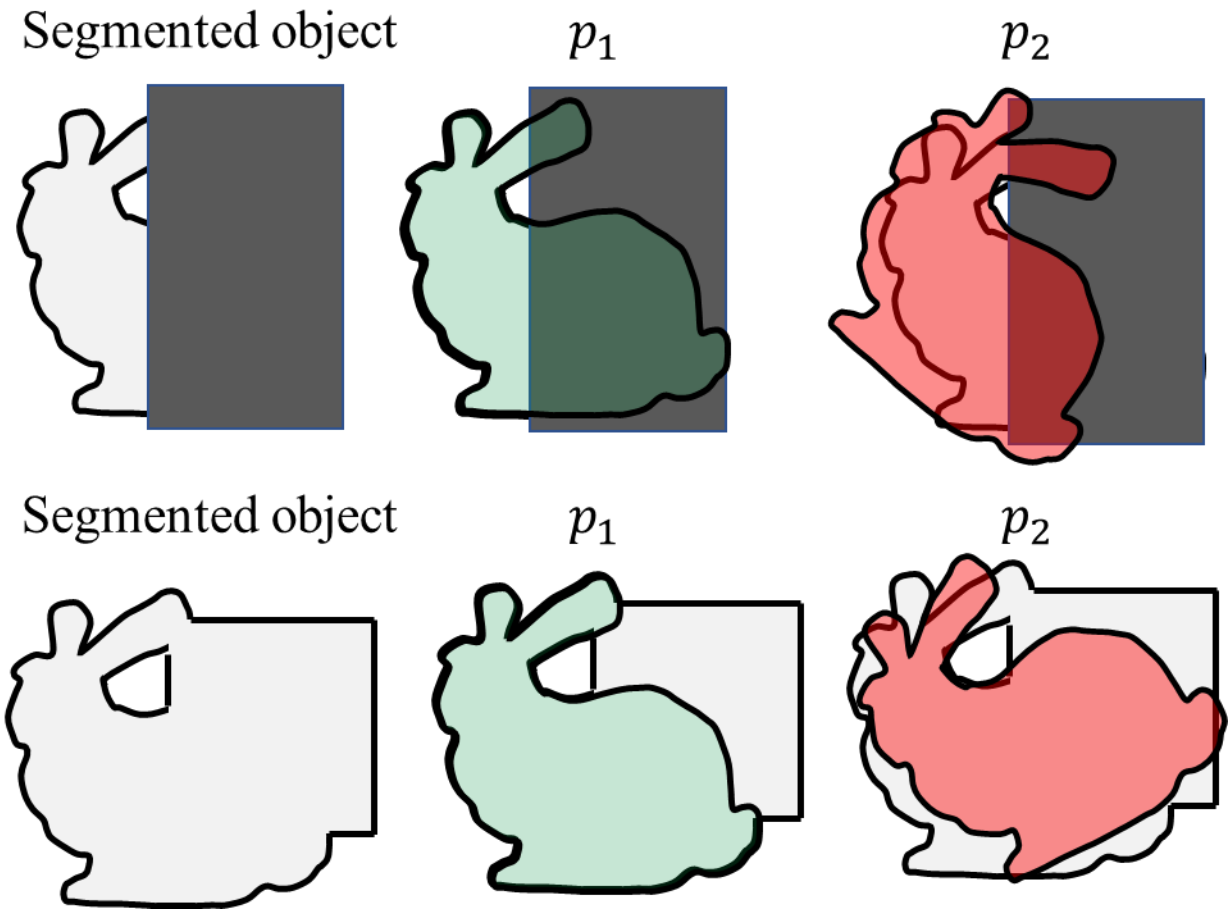


Figure 3.15: Illustration of Challenge II and Challenge III with the observation likelihood

The abovementioned possible challenges can be resolved in most situations, as will be demonstrated in Chapter 7. Challenge I does not pose a significant issue for the developed method in the experiments since there are enough particles found intersecting the object of interest. Challenge II frequently happens in experiments as objects can get occluded. It is important to note that even though better and worse particles can be assigned to low IoU, the weights from the previous steps (where the objects were not occluded) still have an effect. This can be seen in the particle weighting equation, as shown before in Equation 3.19. Further, even though some particles with less accurate poses can obtain similar weights in a single epoch, as the camera moves, they will quickly move away from the solution (and receive a low weight again). Challenge III requires falsely identifying parts of the background as a part of the objects, and it rarely happened in our experiments. However, similar to Challenge II, the particle's weight from the previous step and the accumulation of more observations can often resolve this challenge.

Challenge IV relates to anisotropic changes in the likelihood along different camera axes. Figure 3.16 illustrates this problem. In Figure 3.16 (left), the particle is moved to the right slightly. The IoU (and the observation likelihood) immediately drops significantly. In contrast, in Figure 3.16 (right), the camera moves backwards (in the direction pointing from the camera's center to the object). It can be seen that even though the camera's distance from the object is increasing (the object looks smaller), the IoU and (therefore, the likelihood) stay relatively high.

Unlike Challenges I-III, Challenge IV has the most impact on the accuracy of the developed object-level solution. This challenge will result in lower precision in the direction pointing from the camera toward the object relative to the two other directions. Challenge IV is closely related to the scale ambiguity of the monocular camera, and all the experiments conducted in Chapter 7 are impacted by this challenge. More information will be provided in Chapter 7.

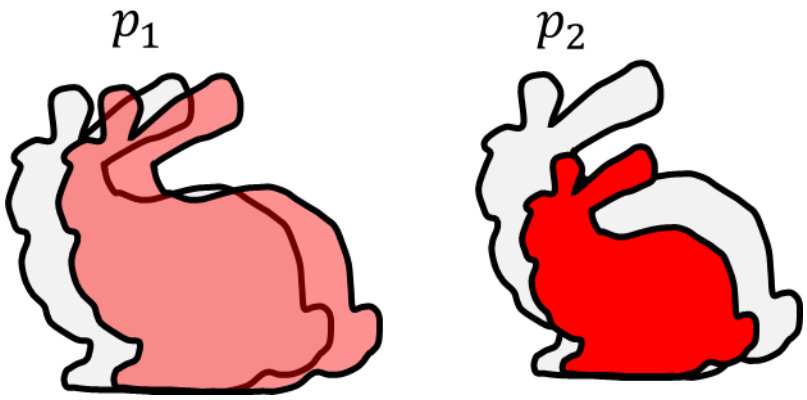


Figure 3.16: Illustration of Challenge IV with the observation likelihood

3.8 Chapter Summary

In this chapter, the main methodology of the developed tightly and loosely coupled object-level RBPF solutions is explained. These methods utilize an IMU mechanization or (kinematic modelling) to predict the device's trajectory. This prediction is corrected using the monocular camera's observations of the objects.

The developed tightly and loosely coupled methods require other modules, such as object segmentation and object pose estimation. Object segmentation is a necessary component for direct evaluation of the observation likelihood, as introduced in Section 3.4 for the tightly coupled solution. The pose estimation is important in the initialization of the object. One important characteristic of many objects in the indoor environment is the lack of identifiable salient feature points on their surface. In the next section, object segmentation and the developed shape-based coarse-to-fine pose estimation methods are explained.

Chapter 4: Object Segmentation, Representation, and Pose Estimation

4.1 Overview

In this chapter, the important topics of object segmentation, representation and pose estimation toward the implementation of object-level RBPF-SLAM are explained in detail. Object segmentation is one of the most important components of the developed RBPF-SLAM. RGB cameras capture a large amount of radiometric information. The appearance of an object is subjected to variations due to the illumination conditions, and the shape of the object can change due to viewpoint variation and camera calibration parameters. Therefore, object segmentation should exhibit robustness to these sources of distortions. In order to address these, a novel segmentation approach is introduced in Section 4.2. The developed method is a DL-based object segmentation. As pointed out in Section 2.2, one of the biggest efforts in the field of DL is to synthesize images to address the problem of massive training data requirements. In Section 4.2, a novel hybrid approach for synthesizing training images is developed. This approach only simulates the pose and the illumination effects on the foreground (object of interest), while the background is captured using real images from the environment of experiments.

The next important component is object representation. The objects in this thesis are represented as organized 2D contour sets denoted as the shape-prior set. This method of object representation provides robustness to the scale, in-plane rotation, in-plane translation, and 3D viewpoint changes. Building this shape-prior set is explained in Section 4.3.1. As mentioned in the literature review in Section 2.3, it is more advantageous to represent object contours as explicit parameterization rather than images. The parameterization of contours using the B-splines is explained in Section 4.3.2.

The final two sections of this chapter are concerned with coarse-to-fine pose estimation. Since, in this thesis, the objects are represented using their contours, classical PnP methods cannot be utilized for the pose estimation directly. The developed contour-based pose estimation using a monocular camera

includes two steps. In the first step, an initial pose is estimated using the shape of the segmented object (coarse estimation). In order to obtain a coarse pose estimation, the segmented contour of the object is matched to the organized shape-prior set using a fast-matching technique. This matching provides an estimation of the pose, assuming that the camera is directly looking into the object. The coarse estimation is explained in Section 4.4. As it was explained in Section 2.4, most state-of-the-art pose estimation methods cannot be applied when the object is symmetrical. It will be shown in Section 4.5 that the developed method can address the challenges with symmetrical objects. In Section 4.6, pose refinement (the second step) is explained. This pose refinement method is based on finding matching features between the 2D contour of the projected 3D model of the object and the segmented contour of the object in the image. In summary, Figure 4.1 shows an overview of this chapter and how it relates to the developed object-level SLAM explained in the previous section. The tightly coupled solution only utilizes pose estimation and refinement in the initialization stage, while the loosely coupled solution depends on pose estimation to update the weights and resample particles. Both of these approaches rely on object segmentation.

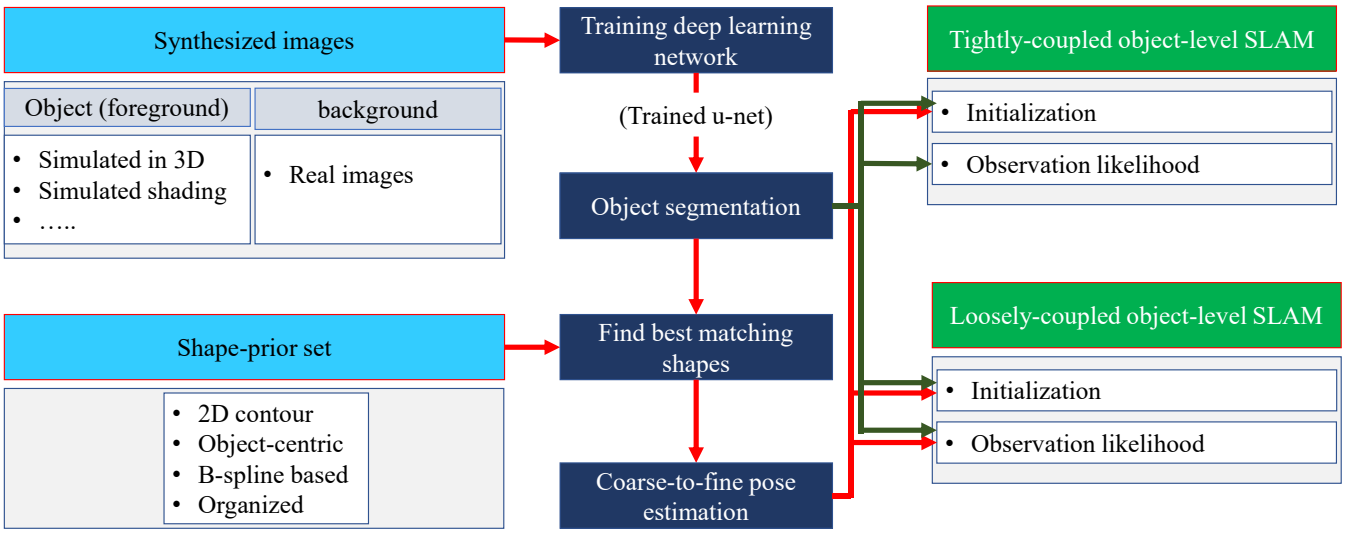


Figure 4.1: Overview of the chapter. The topics discussed in this chapter are shown in light and dark blue.

4.2 Object Segmentation Using Synthesized Images

One of the challenges in DL-based methods is the requirement for a large data set. Unfortunately, there are no available online training image databases for many of the objects found in indoor environments. Therefore, DL-based segmentation cannot be used by many researchers. In this thesis, in order to train DNN, a novel hybrid technique to synthesize images is developed. This approach uses domain randomization (Tremblay et al., 2018) to improve the training. Domain randomization is a method of synthetic data generation that introduces variations in the object and scene characteristics. For example, the illumination conditions, camera viewpoints, intrinsic camera parameters, colour of the object, the pose of the object, and image noise are all among such characteristics. The randomization in these parameters can help train a DNN, which is able to segment the object in real images under different photometric and geometrical distortions.

In this thesis, in order to achieve domain randomization, a hybrid method is used. In hybrid methods (Georgakis et al., 2017), the background is obtained using real images with the object (foreground) excluded. The object of interest is then superimposed onto these images. However, using this simple approach, the changes caused to the object's appearance and shape (e.g., due to the variations in the illumination conditions) cannot be captured effectively. In order to account for these changes, a 3D CAD model of the object of interest can be utilized. Such 3D models can be used to obtain the silhouette of the object as the result of perspective projection. Further, the surface normal (included in most CAD models) can be used to estimate the shadows, shadings, and highlights on the object caused by a light source. Finally, in projecting the object onto the camera, the intrinsic calibration matrix can also be varied. Figure 4.2 illustrates the overall procedure of the developed synthetic image generation. For the shading and highlights, classical Phong's light model (Edward & Shreiner, 2013) is used. Table 4.1 summarizes different domain randomization parameters utilized in our experiments.

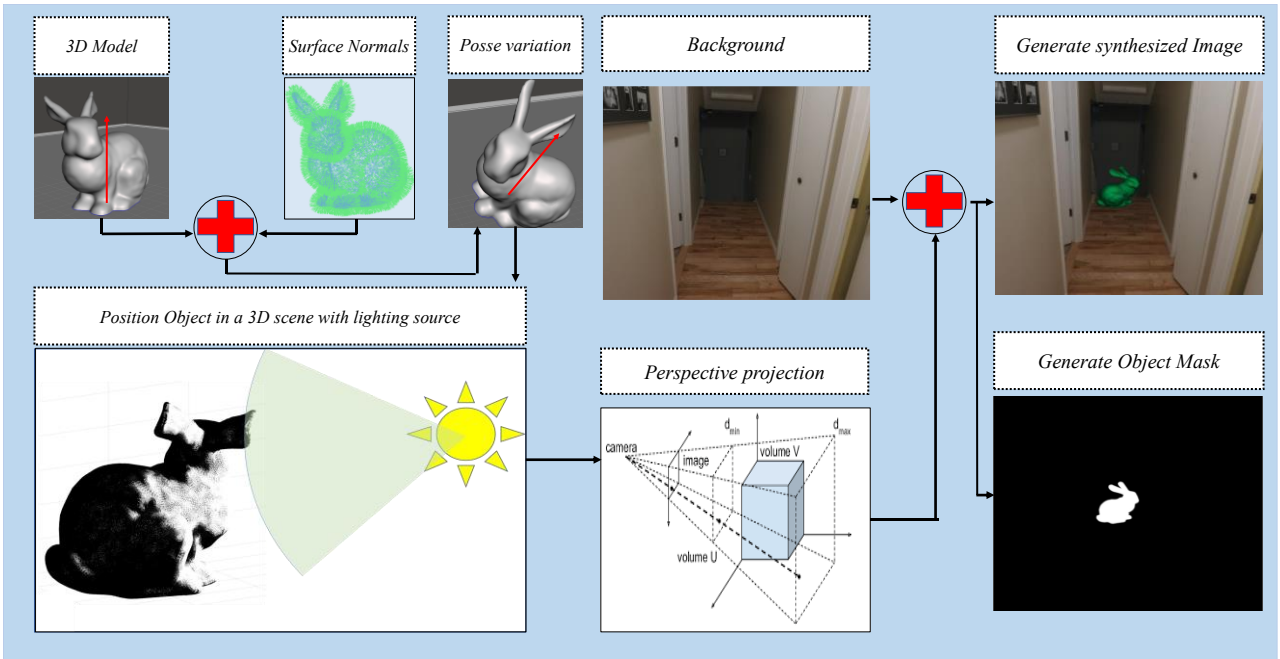


Figure 4.2: Overview of the procedure to build synthetic images.

Table 4.1: Domain randomization parameters

Domain	Randomized parameters	Purpose
Light source	Intensity and position	Increase robustness to illumination condition variations
Object pose	Scale, orientation and translation	Increase robustness to the camera's viewpoints variation
Object motion	Gaussian filter variance	Increase robustness to motion blur
Object colour	R/G/B colours	Increase robustness to object colour variation
Camera calibration	Camera calibration parameters	Increase robustness of the camera's intrinsic parameters
Background	Images from different indoor environments	Increase robustness to background clutter

The generated images are used to train a segmentation DNN known as U-Net. The architecture of U-Net is shown in Figure 4.3. U-Net receives an RGB image input and outputs an image of the same size. In the output image, the pixels belonging to the foreground (or the object) are represented with values closer to one, while the pixels belonging to the background object are represented with values closer to

zero. In Figure 4.3, the size of a layer is shown with three numbers, where the first number denotes the feature layers, and the next two numbers denote a layer's height and width. For example, using this notation, 3×128×128 denotes an input image with three feature layers (red, green, and blue channels) and a resolution of 128 × 128 pixels. Different operations are applied to these layers. The details about such operations can be found in the fundamentals DL textbook and are left out here for brevity. Figure 4.4 shows an example of the segmented object (water bottle cap) acquired with the help of a trained U-Net (the output of the U-Net is superimposed on the input as a mask). Details about hyperparameters used for this DNN are explained in Section 6.8.

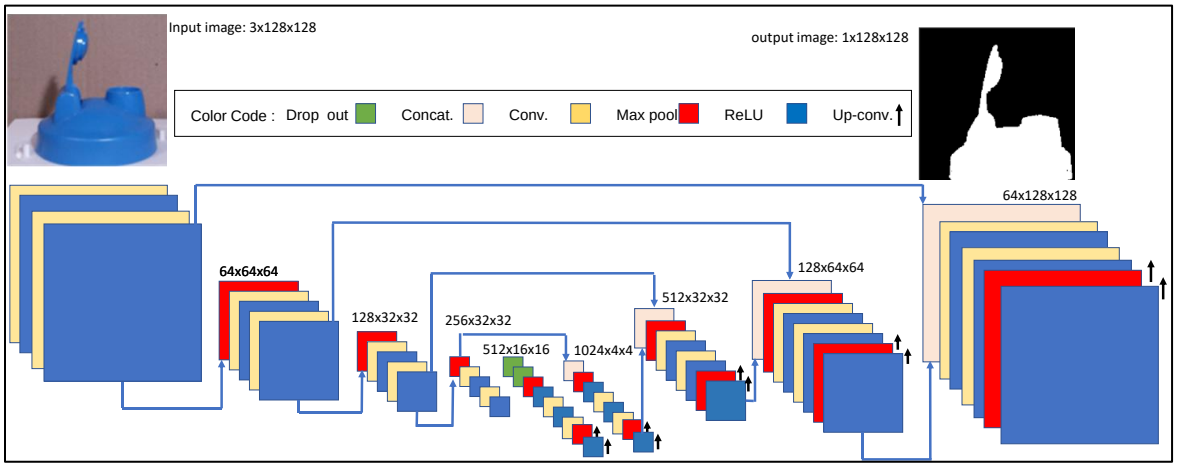


Figure 4.3 : U-Net architecture.

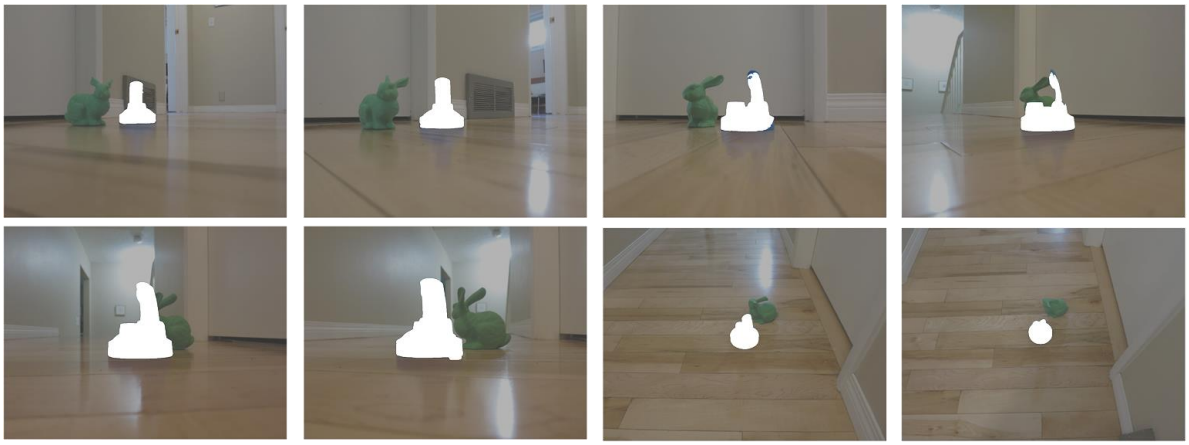


Figure 4.4: Examples of segmented object masks generated with the help of the trained DNN

4.3 Object Representation Using Shape-Prior Set

Object representation is prior information stored about the object's appearance and/or shape. The segmented objects in the previous step are matched to this representation, and the output of such matching is used in the pose estimation. The objects in this thesis are represented using a set of organized 2D contours. These contours correspond to the perspective projection of the 3D CAD model of the object from different viewpoints. Representing objects as a set of 2D contours has the advantage that no information about the texture of the object is required. Further, the contours are more robust to variations in the illumination conditions in comparison to textures. Finally, matching such 2D contours to the segmented object in the image is easier than matching the 3D model directly.

In the past, 2D contours were often stored as images. However, the scale, in-plane rotation and the translation of such contours cannot be modified easily. Such variations are required in matching representation to the segmented contours. In this thesis, 2D contours are represented as B-splines where arbitrary scale, in-plane rotation, and translation can readily be applied. In order to make the object representation invariant to the viewpoint changes, a set of contours is built in an offline phase by projecting the CAD model of the object onto virtual images (Section 4.3.1). These projected contours are then parameterized using B-splines. Finally, the B-splines are grouped together to assemble an organized shape-prior set. In order to avoid confusion, the contours are represented as B-splines, referred to as the **boundary**. In summary, B-splines have three advantages that are helpful for shape matching and pose estimation:

- In-plane translation, rotation and scale changes can be applied to B-splines easily.
- B-splines are parametric curves. Therefore, the order of the points on the contour is known.
- B-splines represent contours with the same number of coefficients, regardless of the number of pixels detected in the image. This will be explained in Section 4.3.2.

4.3.1 Developed Shape-prior Set

In order to build the shape-prior set, virtual camera viewpoints are sampled in an object-centric coordinate frame. In the next step, the CAD model of the object is projected into these viewpoints. Figure 4.5 shows the schematic of this process. In this figure, the viewpoints are illustrated with their coordinate frames (red, green and blue arrows). In these frames, the blue and red arrows correspond to the image axes, and the green arrow corresponds to the axis perpendicular to the image plane. For the projections, a calibration matrix with realistic parameters is utilized. In Chapter 7, it will be illustrated that this shape-prior set can help find the best matching shapes to the segmented object in the image with different cameras. As it is shown in Figure 4.5, the shape-prior set is organized in a two-dimensional array-like data structure based on their φ and λ. These angles are the spherical coordinates of the camera's position in an object-centric viewpoint. In Section 4.4, it will be shown that organizing shape-prior sets in this manner will help develop a fast-searching algorithm, therefore, greatly reducing the matching time.

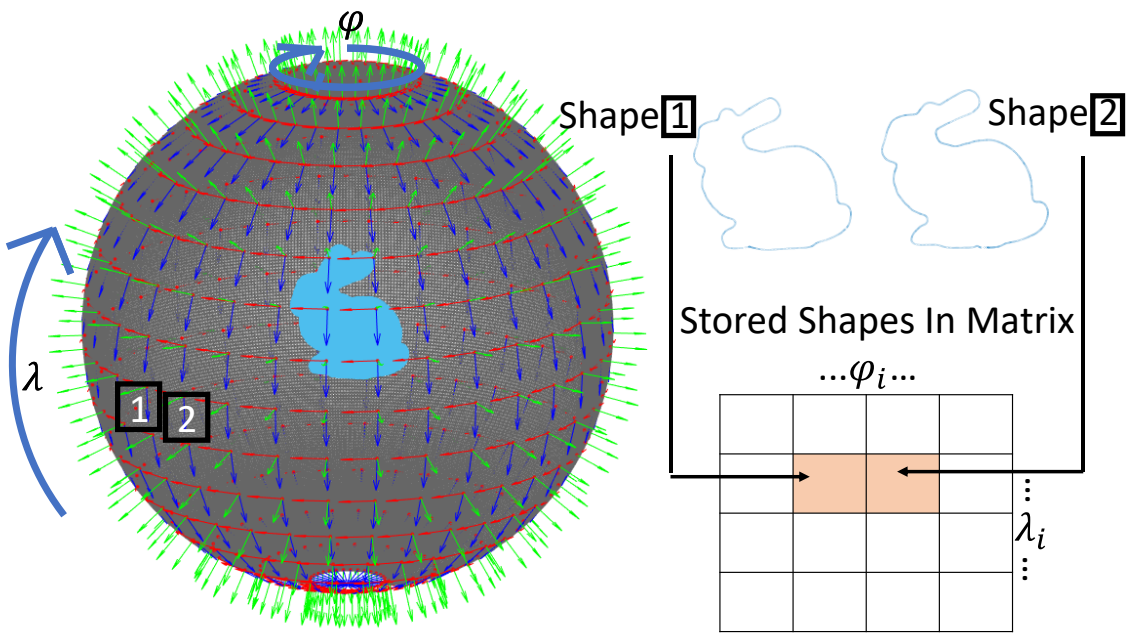


Figure 4.5: Shape-prior sets are generated using object-centric viewpoints.

4.3.2 B-splines

B-splines are piece-wise polynomial curves that are defined as Equation 4.1. In Equation 4.1, a curve ($c(s)$, where s is the parameter) can be defined by selecting a set appropriate B-spline basis (B_n) and control points (c_n) (where N_B is the number of basis). Each basis vector is non-zero in a certain span and 0 otherwise. The vector Q denotes B-spline coefficients while $B(s)$ is the matrix built by concatenating the basis (the transpose is denoted as $(.)'$).

$$c(s) = \sum_{n=0}^{N_B-1} c_n B_n(s) = B'(s)Q \tag{4.1}$$

The process of estimating B-spline coefficients using the boundary of the segmented object (the boundary is the outline of the segmented object) can be achieved using the least square technique as shown in Equation 4.2, where $c(s)$ is the boundary of the segmented object.

$$Q = \big(B(s)B'(s)\big)^{-1} B(s)c(s) \tag{4.2}$$

In order to define a set of B-spline basis, one needs to determine the number of polynomial pieces (the number of basis vectors), the degree of each polynomial, and knot multiplicity. The knots are points where two adjacent polynomial pieces intersect, and knot multiplicity is the quantity that defines the continuity of the polynomials at the point of intersection. As an example, for two polynomials of degree h, a knot multiplicity, same as h, indicates discontinuity of the two polynomials and their derivatives at this knot. Similarly, a knot multiplicity of 0 indicates the continuity of the polynomials and all their derivatives up to order $h - 1$. Any knot multiplicity between 0 and h indicates discontinuity of derivatives from a certain degree and higher. Please refer to (Piegl & Tiller, 1996) for more details on B-spline curves. Examples of the B-spline basis for two different sets of knot multiplicity are shown in Figure 4.6 (each colour indicates one basis vector). B-splines can represent almost any 2D shape, even curves that are discontinued, which can be achieved by modifying the knot multiplicities.

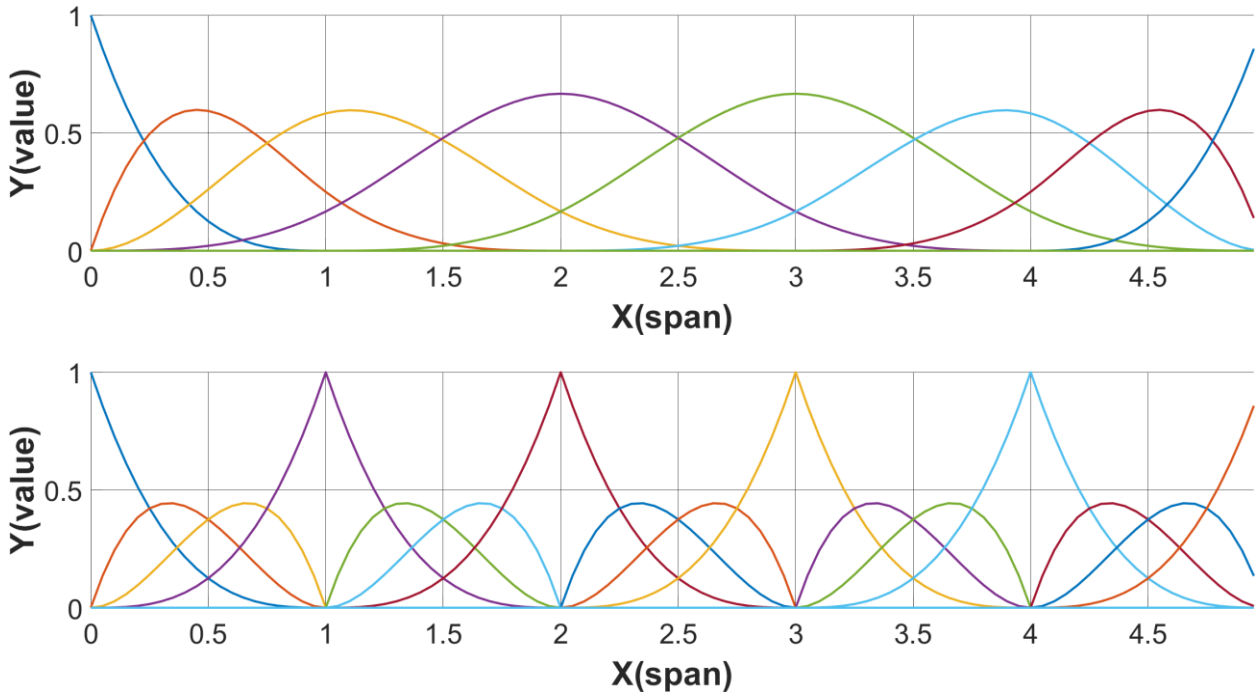
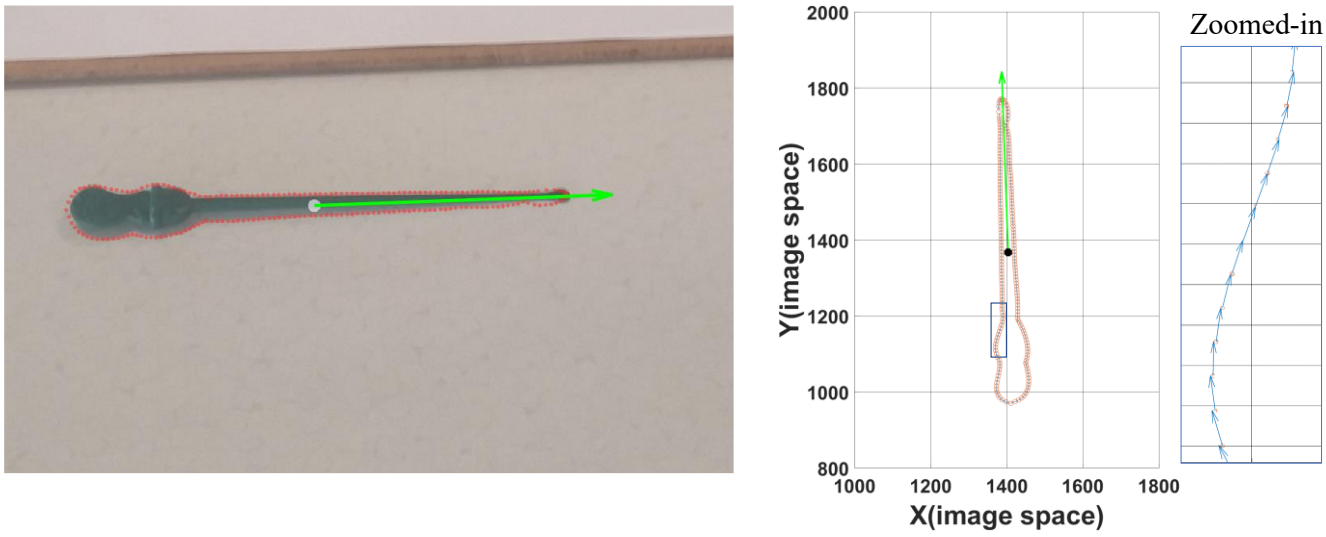


Figure 4.6: B-spline basis of different order and knot multiplicity. The top row shows a B-spline basis of order 4 and a knot multiplicity of 2. The bottom row shows the B-spline basis of order 4 and the knot multiplicity of 3.

Parameterizing shapes using B-splines facilitates the calculation of the tangents and the normal vectors at any point given on the shape. Further, properties such as centroid, area, and second moments can be calculated using B-splines efficiently. It is explained in Section 4.4 that the centroid and area are important properties in matching the boundary of the segmented object and the shapes in the shape-prior set. In Figure 4.7 (a), the fitted B-spline to the contour of the segmented object is shown as a red point, the first column of the second-order moment matrix is shown as a green arrow, and the centroid of the B-spline is shown as a white circle. Figure 4.7 (b) shows a closer look at this B-spline. The zoomed-in figure shows the smoothness of the representation as well as the defined tangents at each point on this curve.

B-spline parameterization imposes ordering in boundary points. This property helps to find the nearest points without using algorithms such as kd-tree (Bentley, 1979). The nearest points are required in the matching step (Section 4.4); therefore, B-splines reduce the computations of the developed method.



(a) (b)

Figure 4.7: Illustration of the fitted B-spline. Figure (a) shows the B-spline fit to the object's boundary in the image. Figure (b) shows a closer look at the fitted boundary.

One of the challenges in using Equation 4.2 to fit B-splines to the pixels in the image is that such fitting is not constrained to any subspace. Therefore, inaccuracies in the segmentation will be directly passed on to the matching stage. This issue can cause wrong matches and, finally, wrong pose estimation. It will be shown in the following that Equation 4.2 can be modified slightly using a projection matrix W.

The projection matrix is built using a subset of shapes in the shape-prior set. By using a projection matrix, the fitted B-spline is forced to reside in a vector space defined by the shapes in the shape-prior set. This approach increases robustness to the noise in the images or small occlusions of the object boundary. Formally, the projection matrix is defined by concatenating the B-spline coefficients ($Q_{.,x}$ and $Q_{.,y}$) of the shapes in the shape set as shown in Equation 4.3. The first two columns in this equation introduce invariance to in-plane translation (bold represents vector). The third and fourth columns introduce invariance to in-plane rotation and scale changes (Q_m is created by averaging all the shapes in the shape set). The remaining coefficients correspond to the shape prior set. Increasing the number of shapes results in a fitting that follows the pixels more accurately (but is more affected by the noise as well).

$$W = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathrm{Q}_{m,x} & -\mathrm{Q}_{m,y} & \mathrm{Q}_{1,x} & -\mathrm{Q}_{1,y} \cdots \mathrm{Q}_{n,x} & -\mathrm{Q}_{n,y} \\ \mathbf{0} & \mathbf{1} & \mathrm{Q}_{m,y} & \mathrm{Q}_{m,x} & \mathrm{Q}_{1,y} & \mathrm{Q}_{1,x} \cdots \mathrm{Q}_{n,y} & \mathrm{Q}_{n,x} \end{bmatrix} \quad (4.3)$$

In order to explain the fitting method mathematically, one can commence with Equation 4.4. Initially, the coefficients $\hat{Q}$ are estimated using the detected object boundary in the image (c_{img}). In the next step, the corresponding coefficients in the projection space (O, a vector) are found using Equation 4.5. The main difference between Equations 4.4 and 4.5 and Equations 4.1 and 4.2 is the inclusion of the W matrix in estimating the coefficients.

$$\hat{Q} = (BB')^{-1}Bc_{img} \quad (4.4)$$

$$O = ((W'BB'W)^{-1}W'BB')\hat{Q} \quad (4.5)$$

The coefficients O are in the projected space (W). Equation 4.6 can be used to find the updated B-spline coefficients. Finally, the corresponding curve points can also be reestimated using Equation 4.7.

$$Q = WO \quad (4.6)$$

$$c_{fit} = B'Q \quad (4.7)$$

Unfortunately, the fitted coefficient does not correspond directly to any of the shapes in the shape-space and often corresponds to a weighted combination of many more shapes (some of these shapes are not close to each other in object-centric coordinates). This behaviour is due to the high dimensions in the column space of the projection matrix (in our experiments, the shape set includes more than 30,000 shapes). However, the explained method can still be used to fit a smooth boundary to the detected pixels around the segmented object. In the next section, a fast-searching algorithm is introduced to find the best matching shapes in the shape-prior set with the help of estimated coefficients Q and c_{fit}.

4.4 Finding Best-Matching Shapes (coarse pose estimation)

In order to obtain a coarse pose estimation, the segmented boundary of the object should be matched to the shapes in the shape-prior set. The output of such a process is a list of the best matching shapes in the shape-prior set. From this point on, the boundary of the segmented object in the image is denoted as the query shape for brevity. The process of matching is shown in Figure 4.8.

In order to match, the unknown in-plane rotation, translation, and scale between the shape-prior and the query shape must be found. In order to achieve this, relative translation and the scale between the query shape and a prior shape are found in the first step. Since both shapes are parameterized using B-splines, the area and the centroid can be calculated very fast. Figure 4.8. shows this process (see query shape (blue) is scaled and translated (red shape) in Figure 4.8(b)).

In order to estimate the relative orientation, one possible solution is to find a corresponding feature point on the query shape and the shape prior. For fast feature detection, the feature of interest is defined as the point that is furthest away from the center of the boundary (see Figure 4.8(b)). Unfortunately, due to the distortions (e.g., occlusions), false feature correspondence might be established between the two shapes, and therefore, inaccurate estimation of the in-plane rotation will be acquired. In order to address this issue, more than one correspondence hypothesis can be evaluated, and the rotation which achieves the highest IoU area can be selected. In order to consider multiple hypotheses, a list of furthest points is considered. Since B-splines are parameterized curves, this list can be organized into groups immediately without the requirement to perform the Nearest Neighbour Search (NNS). In each group, the point furthest from the center is designated as a possible feature. The rotation is estimated for the corresponding features, and the IoU between the transformed query shape and the shape prior is estimated. Finally, the in-plane rotation that results in the largest IoU is selected (see Figure 4.8 (c)). In practice, often testing about 2 to 3 hypotheses is sufficient to find the best match.

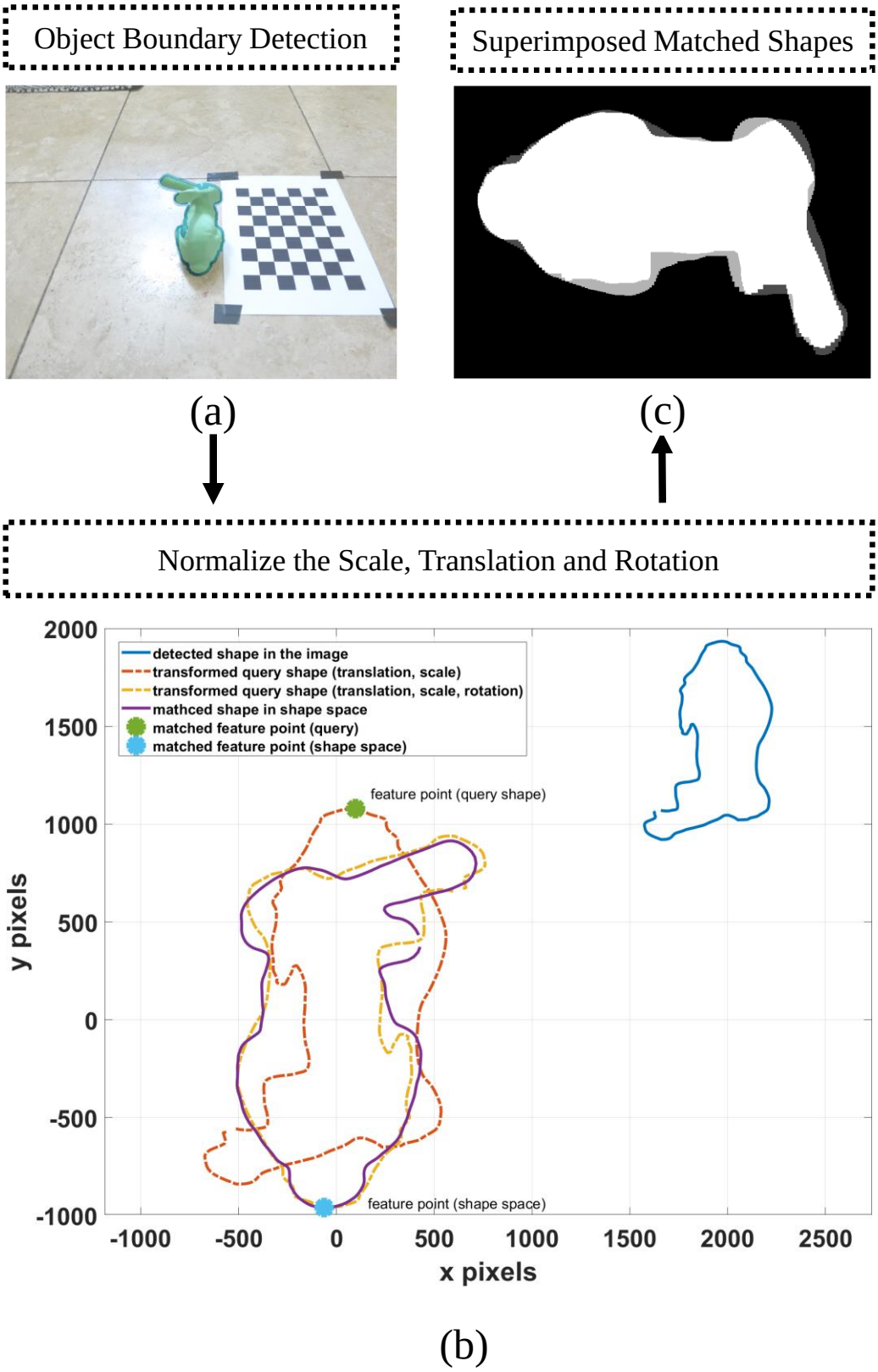


Figure 4.8: Illustration of matching a query shape to a shape-prior.

Unfortunately, using an exhaustive search, the before-mentioned process should be repeated for all the shapes in the shape-prior set to find the best matches. However, an exhaustive search does not utilize the fact that the developed shape-prior set is organized in an array-like two-dimensional structure using spherical coordinates (φ and λ). An alternative faster matching method is to take advantage of the fact that the object-centric shape-prior set (S) (as shown in Figure 4.5) is organized.

In this fast-matching method, initially, several shapes in the shape-prior set are selected randomly at the four-quadrant (q). Each quadrant is built by dividing the range in $\varphi_i \; and \; \lambda_i$ into two halves. Next, the sum of all IoUs (T_{IoU}) for this selected set of shapes in each quadrant is measured. Each IoU is measured by the process that is just explained above. One (or more) quadrants q^* with the least total distance (highest T_{IoU}) are selected as the likely space that can include the closest shapes to the query shape. This quadrant is then divided into four smaller quadrants with new ranges in φ_{i+1} and λ_{i+1}, and the previous process is repeated. Furthermore, at every iteration, the matched shapes are ranked based on the IoU area and stored in a list (S^r), while the indices of tested shapes are recorded in a list as well (S^c) to avoid checking the same shape multiple times. The process stops when the number of remaining shapes in the current quadrant ($n(\lambda_{i+1}) \times n(\varphi_{i+1})$) is smaller than the number of random shape selections (a). At this point, all the remaining shapes are tested in this quadrant, and the algorithm terminates. A pseudocode of the aforementioned algorithm is shown in Figure 4.9. The output of the algorithm is a ranked list of all the matched shapes based on IoU. In order to keep the best matches and decrease the computational cost in the further steps of pose estimation, a fixed number of shapes (in our experiments, 50) can be kept, and the remaining shapes can be discarded.

To compare the developed fast-matching method with an exhaustive search, the number of computed IoU for the two searching algorithms is recorded. It is found the fast-matching method can return most of the best-matching shapes by only testing 24.10% of the total shapes in the shape-prior set.

Pseudocode: Finding best matching shapes in the shape space ($\boldsymbol{I}$, $\boldsymbol{S}$, $\boldsymbol{N}$,$\boldsymbol{a}$)

Inputs	Outputs
$\boldsymbol{I}$: image, $\boldsymbol{S}$: shape-space, $\boldsymbol{N}$: deep learning network, $\boldsymbol{a}$: initial number of random shapes	$\boldsymbol{S^r}$: Best shapes

iteration: $\boldsymbol{i} = 1$, checked shapes: $\boldsymbol{S_1^c}=\emptyset$, ranked shapes: $\boldsymbol{S_1^r} = \emptyset$

Step1: Object contour detection.

$\boldsymbol{O}$ =segment object ($\boldsymbol{I}$, $\boldsymbol{N}$)

$\boldsymbol{S^I}$ =detect object's contour ($\boldsymbol{O}$)

Step 2: Find the best matching shapes.

while $\boldsymbol{n}(\boldsymbol{\varphi}_{i+1}^{q^*}) \times \boldsymbol{n}(\boldsymbol{\lambda}_{i+1}^{q^*}) > \boldsymbol{a}$ // number of: $\boldsymbol{n}(.)$

- $\{(\boldsymbol{\varphi}_{i+1}^{q}, \boldsymbol{\lambda}_{i+1}^{q}), q \in [1,4]\}$ = divide range to quadrants $(\boldsymbol{\varphi}_{i}^{q^*}, \boldsymbol{\lambda}_{i}^{q^*})$
- $\{\boldsymbol{S}_{i+1}^{q}, q \in [1,4]\}$=select random shapes $(\boldsymbol{S}, \boldsymbol{S}_i^c, (\boldsymbol{\varphi}_{i+1}^{q}, \boldsymbol{\lambda}_{i+1}^{q}))$
- $\boldsymbol{S}_{i+1}^{c}$=update checked shapes $(\boldsymbol{S}_i^c, \boldsymbol{S}_{i+1}^{q})$
- $\boldsymbol{d}_{IoU}^{q}$=Measure IoU $(\boldsymbol{S}^I, \boldsymbol{S}_{i+1}^{q})$ // remove in-plane translation, rotation and scale
- $\boldsymbol{S}_{i+1}^{r}$=Update best matches list $(\boldsymbol{S_i^r}, \boldsymbol{d}_{IoU}^{q})$
- for each q
  - $\boldsymbol{T}_{IoU}^{q} = \sum \boldsymbol{d}_{IoU}^{q}$
- end
- q^* =Select quadrant with largest total IoU $(\boldsymbol{T}_{IoU}^{q})$

end

Figure 4.9: Pseudocode of the developed fast-matching algorithm.

Estimating the object's pose can be accomplished as the shapes in the shape-prior include an object-centric coordinate (φ and λ) of the camera. Further, the in-plane rotation between the image and the monocular camera is also known (as explained in Section 4.4). Therefore, this information can be used to estimate a coarse estimation of the object's pose in the camera's frame. This pose is accurate only when the camera is directly looking into the object (the camera's coordinate center, the image's principal point, and the centroid of the 3D object are on a line). However, as it will be shown in Chapter 7, the initial pose estimation stays highly accurate when the assumption is not valid. Further improvement to the pose estimation can be obtained using refinement (Section 4.6).

4.5 Pose Estimation of Symmetrical Objects

With the help of the ranked list of best-matching shapes (S_r), it is possible to obtain a coarse pose and use it in the object-level RBPF-SLAM solution. However, for the objects that are symmetrical (they appear to have symmetrical silhouettes from multiple viewpoints), more than one shape in the ranked shape list should be selected. Figure 4.10 demonstrates that with the help of object segmentation (Figure 4.10 (a)) and matching, plausible viewpoints for this object can be identified. In Figure 4.10 (b), red points identify the best-matching shapes in the shape-prior set (lighter colours correspond to higher IoU). Figure 4.10 (c) shows the corresponding poses of the best-matched shapes in the shape-prior set.

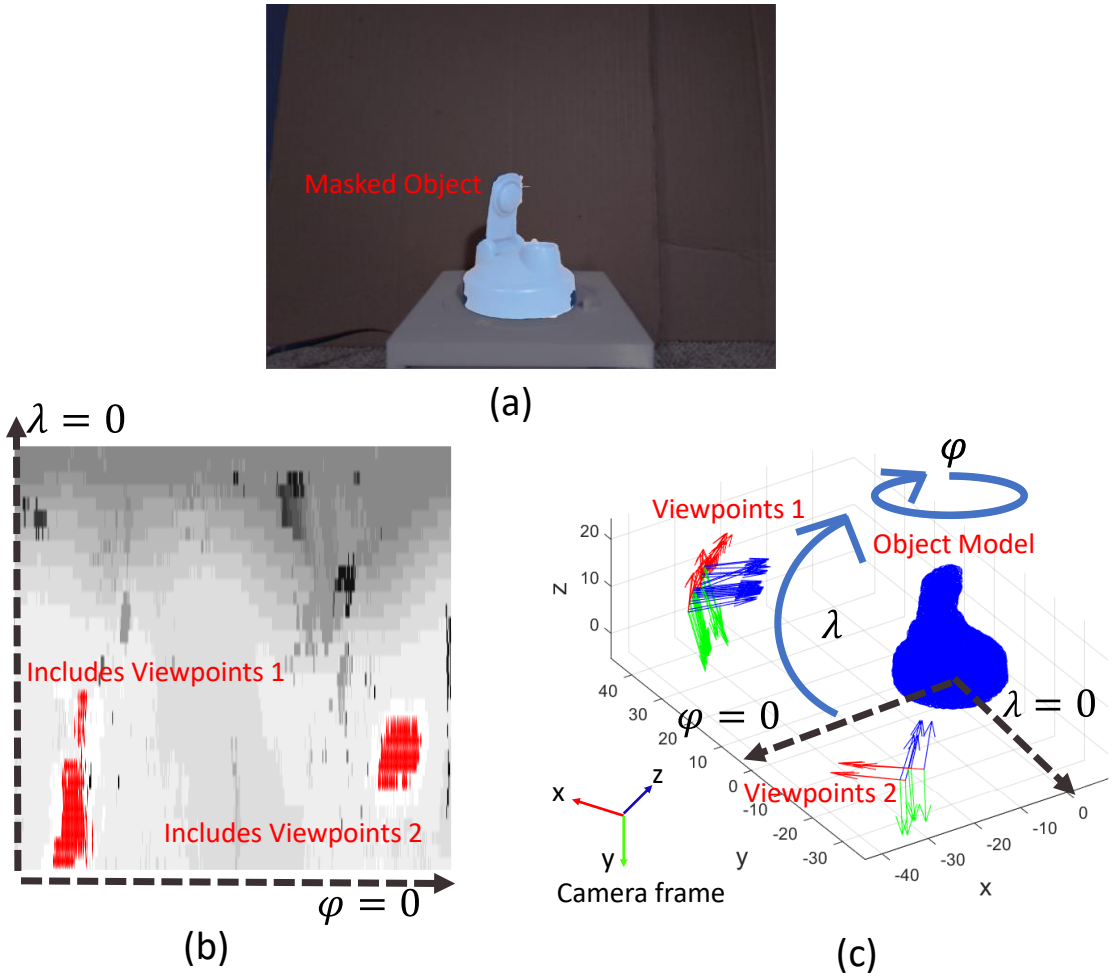


Figure 4.10: Illustration of pose estimation of a symmetrical object.

4.6 Pose Refinement

Pose estimation using only one image is very important in the developed loosely coupled object-level SLAM (Section 3.3); therefore, the accuracy of this estimation should be improved. This improvement can be obtained through the process of pose refinement. The pose refinement in this section relies on the coarse pose estimation and the boundary of the segmented object. The coarse pose is used to project the 3D model of the object onto the image. Once the points are projected, their boundary in the image is found. This projected boundary will reside at the center area of the image, and through the process of refinement, it should align with the boundary of the segmented object. At this point, some feature correspondences (2D-to-2D) between the projected and segmented boundaries should be found. With the help of such correspondence, it is also possible to find correspondence (2D-to-3D) of the points on the segmented boundary and the 3D points in the model. Finally, using these correspondences, a PnP algorithm can be used to find the pose (see Figure 4.11)

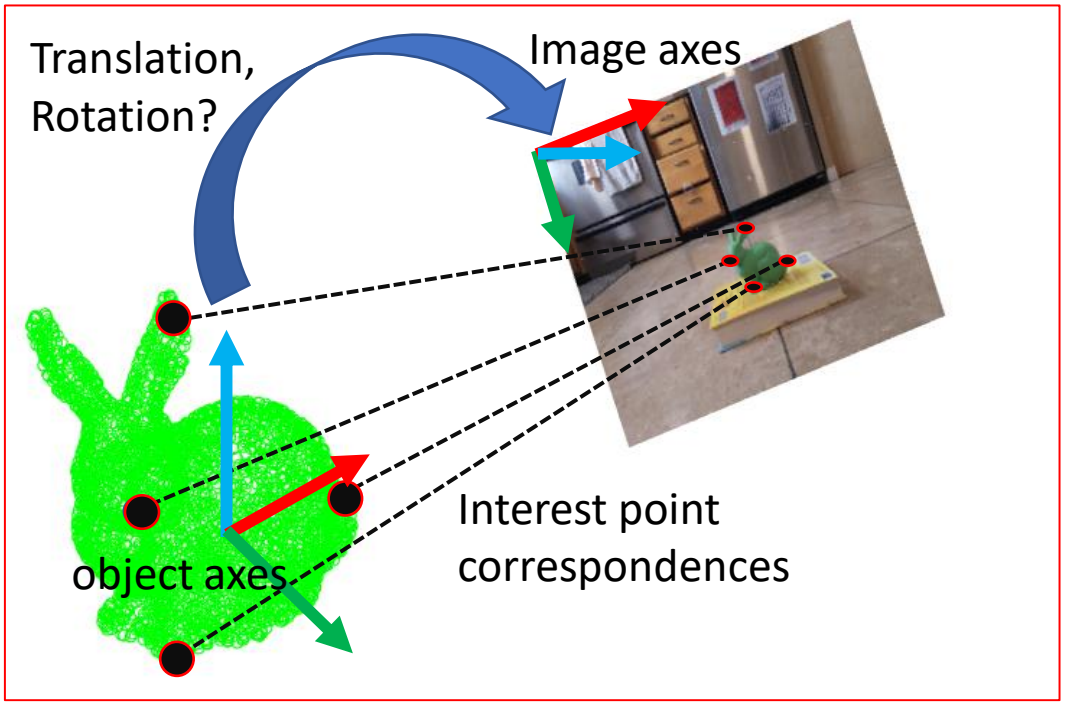


Figure 4.11: A set of 2D-to-3D correspondences are required to solve a PnP problem.

In this thesis, for 2D-to-2D correspondences, the points with large curvature on the two boundaries (the projected and the segmented) are detected. The curvature of a point can be defined as shown in Equation 4.8, where $c(s) = [x(s), y(s)]'$ is the curve (Equation 4.1). The variables x', y', x'' and y'' denote the first and second-order derivates of B-splines with respect to s. The estimated value κ for each point can be used to select a group of points with the highest curvature.

$$\kappa = \frac{|x'(s)y''(s) - x''(s)y'(s)|}{\left((x'(s))^2 + (y'(s))^2\right)^{3/2}} \tag{4.8}$$

In order to establish 2D-to-2D correspondences, the projected boundary is translated to the segmented boundary first. The scale differences between these two boundaries are resolved using the ratio of the areas. The correspondence can then be established by finding the closest features. This is achieved using a range search algorithm. A range search is similar to NNS, where the nearest points within a maximum radius from a query point are found. If multiple neighbours are detected in a given range, only the closest one is used for correspondence. If no points are found with the maximum radius, then this feature is discarded. The radius for the range search is set to 15 pixels in the experiments. The 2D-to-2D correspondences can be seen in Figure 4.12. In this figure, the boundary of the projected object (red) is centred in the image, as expected from the coarse estimation discussions. These boundary points are scaled and translated to the segmented object (yellow), and finally, 2D-to-2D correspondence is established (four examples of the established correspondences are shown with numbers). In the next step, 2D-to-3D correspondence can be established since the correspondence between projected points and 3D points on the object is known. The 2D-to-3D correspondence can be used to solve a PnP problem. In the thesis, P3P (Gao et al., 2003) with RANSAC (Torr & Zisserman, 2000) is used (this algorithm is available in MATLAB). The explained procedure can be shown in Figure 4.13.

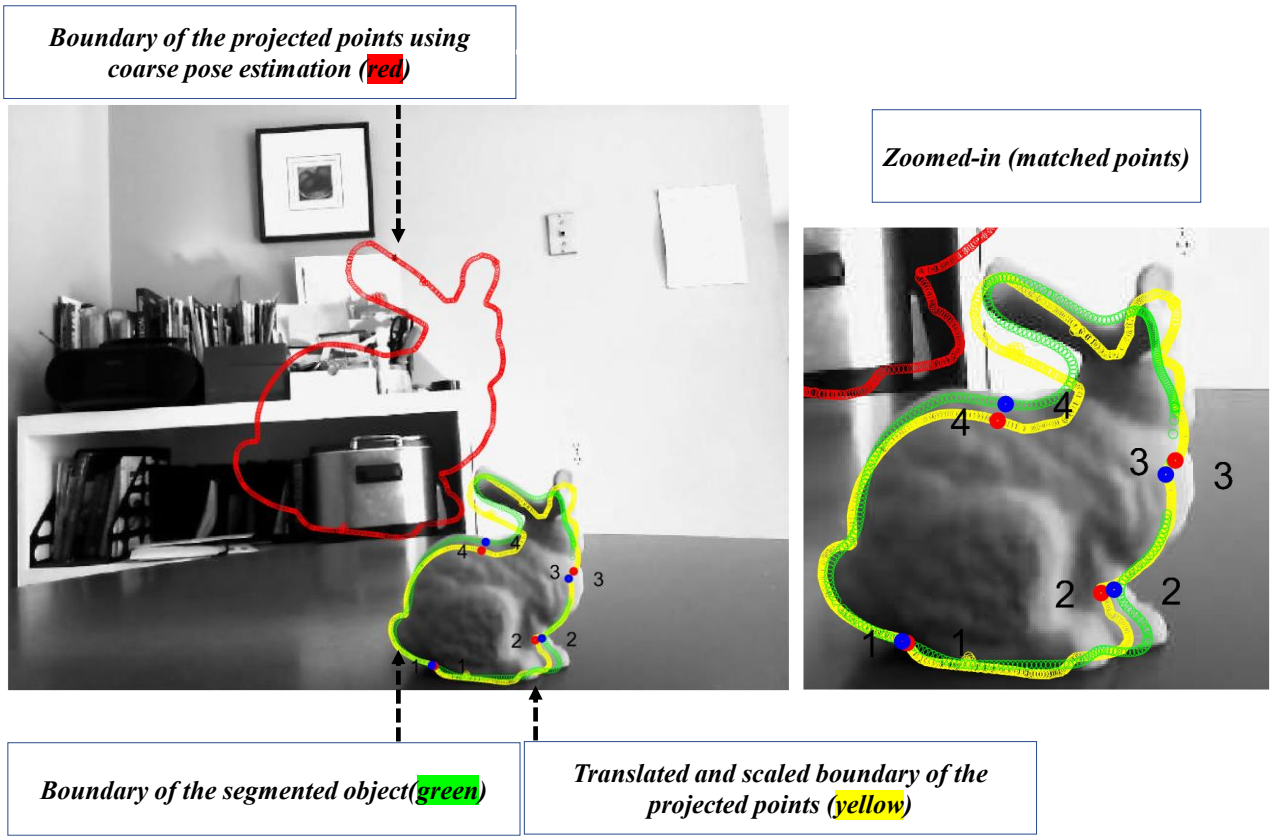


Figure 4.12: Establishing 2D-to-2D correspondence using coarse pose estimation

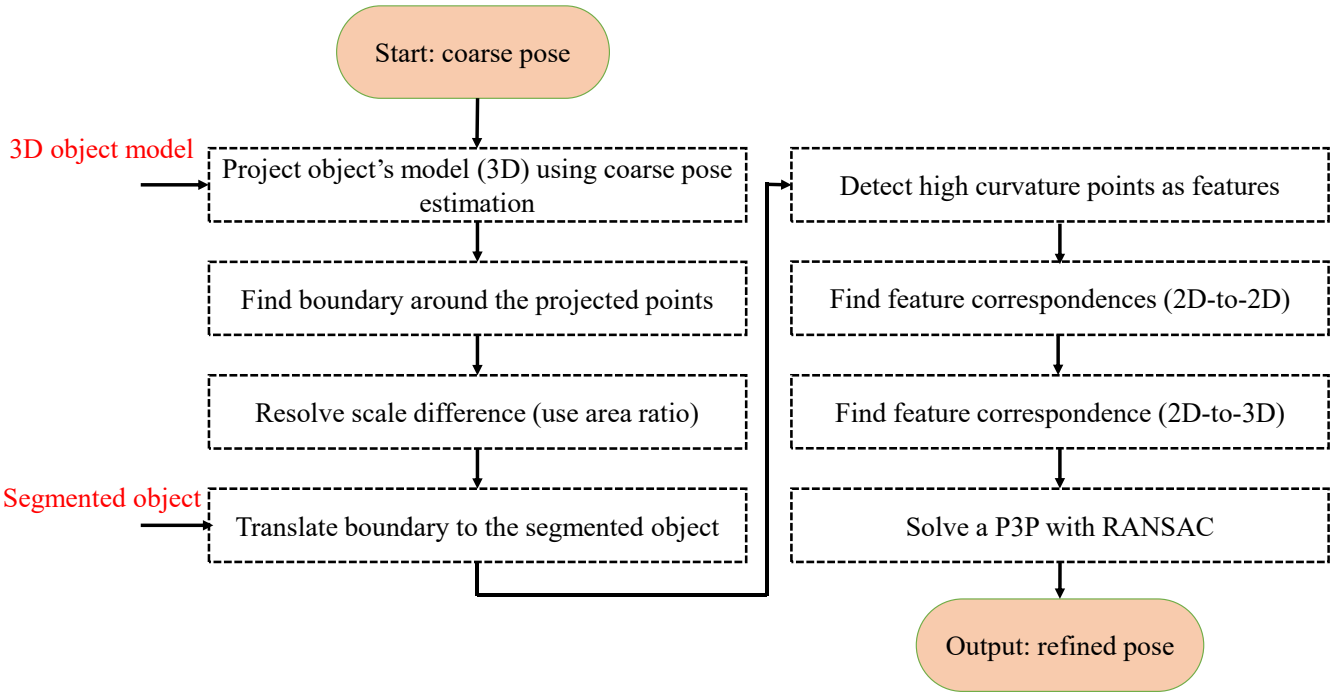


Figure 4.13: The flowchart of the developed pose refinement

4.7 Chapter Summary

In this chapter, the developed object segmentation using synthetic data generation and the contour-based coarse-to-fine pose estimation algorithm were explained. These important components are required in the initialization and the particle weighting processes, as introduced in Chapter 3.

A second important component of the particle weighting process is to know the relative orientation and the translation between the sensors. The boresight calibration of IMU and monocular cameras is a challenging task. The main reason for this is the lack of a common observation between the two sensors. Estimating these calibration parameters in the past often relied on an independent trajectory estimation using each sensor. Due to the error accumulation in the IMU mechanization, such processes require GNSS/IMU integration. However, this integration is not possible in the indoor environment. In the next section, it will be shown that it is possible to utilize the structure of the indoor manmade environment to obtain the boresight calibration parameters.

In the past, the boresight and lever arm calibration of a monocular camera and a rangefinder was estimated using planar targets. However, detecting a sufficient number of points on these targets is challenging using an ultrasonic sensor due to the large beam angle. In the next section, a method is proposed that utilizes the structure of the indoor manmade environment directly to estimate these parameters.

Chapter 5: Extrinsic (Boresight and Lever Arm) Calibration of Sensors

5.1 Overview

Sensor fusion is important in many state-of-the-art solutions to the SLAM. The developed tightly/loosely coupled object-level solution also relies on sensor fusion. The sensor fusion of a monocular camera with an IMU (as explained in Chapter 3) allows motion prediction. Further, such sensor fusion is necessary to estimate a map with a real scale. Similarly, the sensor fusion of a monocular camera with a rangefinder, such as an ultrasonic sensor, provides range measurements. Such range measurements are helpful in the undelayed initialization and particle weighting processes.

One necessary component for these fusions is to find the extrinsic calibration parameters of the sensors. Extrinsic calibration refers to the process of estimating translation (lever arm) and orientation (boresight) between the coordinate frames of two or more sensors. As discussed in Section 2.6, state-of-the-art extrinsic calibration methods of IMU and monocular cameras often rely on equipment such as a turntable that is not accessible on-site. Some other calibration methods rely on an accurate and independent estimation of the trajectory with each sensor. But, standalone trajectory estimation, especially for low-cost IMU, is not possible due to the accumulation of errors in dead-reckoning.

In this chapter, a novel method of boresight calibration of an IMU and a monocular camera is developed. This approach does not depend on expensive equipment or independent trajectory estimation. This developed method leverages DL to estimate the normal vector to the floor in indoor environments. This vector is antiparallel to the direction of the local gravity vector. Since the local gravity vector can be estimated in the IMU's frame, a correspondence between observations in the IMU and monocular camera frame can be established. Finally, such correspondence can be used to solve for the boresight parameters. The developed method is designed to be used for handheld devices, small drones, and indoor robots.

Section 5.2 will explain the floor-segmentation method using DL, and Section 5.3 will explain the boresight calibration of IMU and monocular cameras.

In Section 5.4, the developed boresight and lever arm calibration of the ultrasonic rangefinder and the monocular camera are explained. Based on the reviewed literature (Section 2.6), the calibration of 2D rangefinders and monocular cameras requires the detection of at least 15-20 points on special calibration targets (e.g., checkerboard) (Unnikrishnan & Hebert, 2005). However, detecting such a number of points is very challenging with an ultrasonic sensor that has a low angular resolution (very large targets might be required). In this thesis, indoor structures are used for calibration instead of relying on special targets. The developed extrinsic calibration utilizes the 2D line-segment maps built by the ultrasonic sensor in indoor manmade environments. The 2D maps are built using state-of-the-art methods for wheeled robots equipped with ultrasonic rangefinders (Abadi & El-Sheimy, 2022).

5.2 Floor-Segmentation Using Deep Learning

Floor segmentation in indoor manmade environments can provide useful information. Floors in most of these environments are often planar surfaces where the normal to this plane accurately aligns with the direction of the local gravity vector. This alignment can be used to solve for the boresight calibration parameters of an IMU and a monocular camera. A requirement for such a process is to segment the floor in the images. Floor segmentation in the past is mainly used for navigation (Berenguel-Baeta et al., 2020; Posada et al., 2010; Zhou & Li, 2006). These approaches are not robust to distortions caused by the reflections, shadows, shadings, and textures on the floor. Recently, DL provided solutions for accurate semantic image segmentation. Such capacity can also be used for robust floor segmentation in the presence of the abovementioned distortions. To the best of the author's knowledge, no general-purpose floor segmentation in the indoor environment has been proposed in the past.

The developed DL-based segmentation utilizes U-Net architecture. However, the training data for floor segmentation is not synthesized. Unlike objects, the floors do not have a general shape and can only be defined in terms of their spatial relationships to other surfaces and objects in indoor manmade environments. For example, a floor is often encompassed by vertical structures (e.g., walls). Therefore, in order to generate synthetic images, a complete simulation of an indoor environment might be required, and the hybrid method discussed in Section 4.2 cannot achieve high segmentation precision (and recall). In order to address the large data requirement, the image set captured from the indoor environments is augmented using in-plane rotation (image plane), brightness modifications, and more. The binary output masks for training are generated with the help of Grab-cut (Rother et al., 2004), which requires user interaction.

Some of the results of the estimated floor segmentation with the help of the trained DNN are shown in Figure 5.1. The first image (Figure 5.1(a)) is captured inside the same indoor environment as the training set; however, from a different viewpoint and using a different camera. It can be seen that the floor segmentation is successful (the green area is the segmented floor). As pointed out previously, distortions such as the ones caused by the reflections on the floor can challenge classical segmentation methods. The second image (Figure 5.1 (b)) is acquired using the Google search engine. The indoor environment is very different from the training set, and the floor has many textures. Such textures will typically cause severe issues for classical segmentation methods. It can be seen that the floor segmentation is also successful to a great extent. The third image (Figure 5.1(c)) shows another image acquired using the Google search engine, and most pixels belonging to the floor are identified. The fourth image (Figure 5.1(d)) shows an example of utilizing the trained DNN in outdoor scenes where the road and grass are segmented instead of the floor segment. The relative success of the developed floor segmentation in such a variety of scenes can be understood by considering common features in these scenes. One common feature, especially in

indoor manmade environments, is the box-like structure. The floor in such environments is often a planar surface that lies below the horizon and is confined by vertical structures (e.g., walls). Some of these common features are also present in the images shown in Figure 5.1. It is possible that the trained DNN is capturing such aspects.

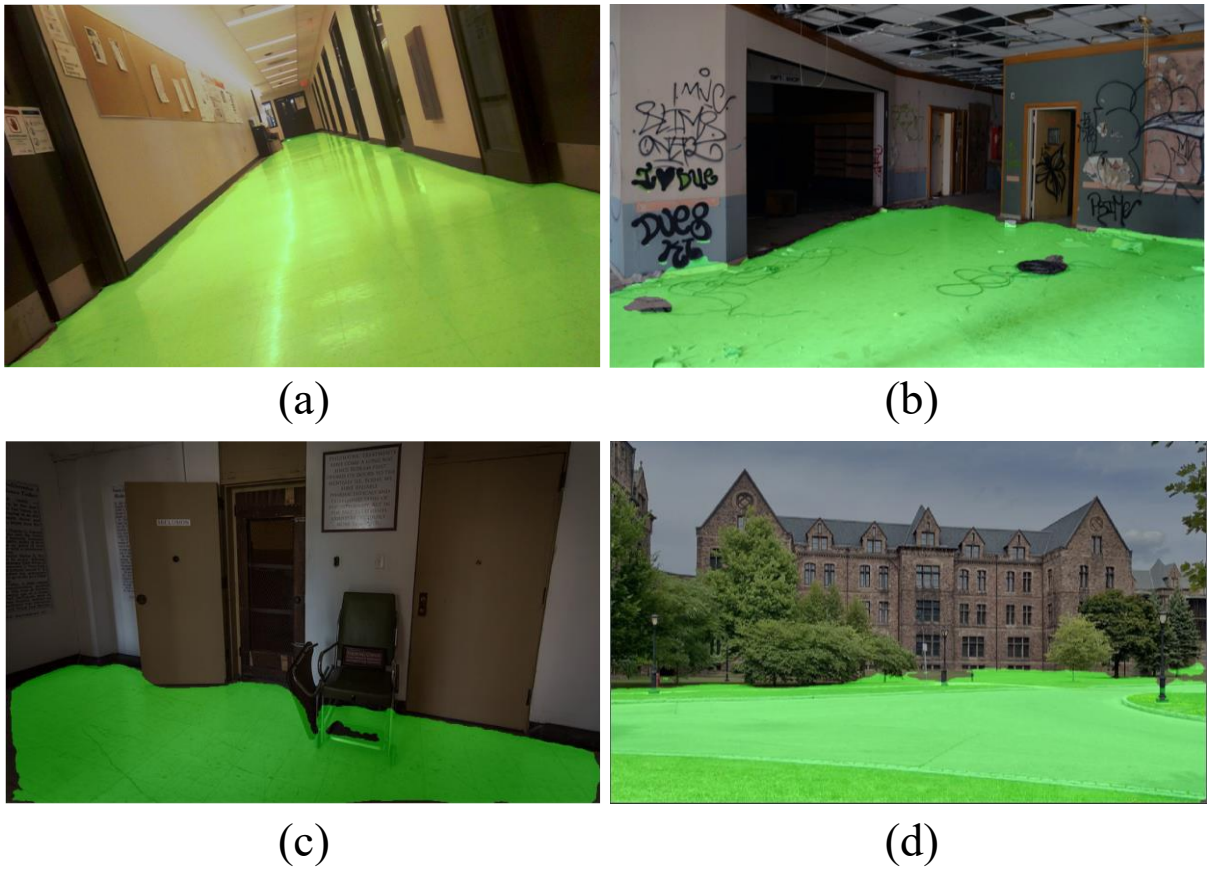


Figure 5.1: The floor(or road) segmentation in different environments with the help of the developed approach Image references (Creative Common license): (b) Courtesy of L. Brett." Hospital Lobby," Accessed: July 2023. (c) Courtesy of L. Warren," Middle East Wing, Old Buffalo State Hospital," Accessed: July 2023, (d) Courtesy of L. Viv," Trans-Allegheny Lunatic Asylum," Accessed: July 2023.

The performance of the developed floor segmentation is tested using precision and recall. Precision (Pr) and recall (Re) are defined in Equations 5.1 and 5.2 where TP, FP, and FN correspond to true positives, false positives, and false negatives, respectively. All three values are calculated at the pixel level. The training image masks are obtained manually using the Grab-cut method.

$$Re = TP/(TP + FN) \tag{5.1}$$

$$Pr = TP/(TP + FP) \tag{5.2}$$

The different precision and recall values are obtained by varying the binarization threshold for the output image. This hyperparameter threshold can be set to a value between 0 to 1. The precision versus recall plot is shown in Figure 5.2 for three image sets by varying the binarization threshold between 0.01 to 0.99. The image sets (easy) and (challenging) are taken from the same environment as the training set. The challenging set includes images taken from viewpoints very different from the training set, while the image set (easy) includes images obtained from similar viewpoints. The image set (internet) was obtained using the Google search engine to find images of indoor scenes such as office spaces, hospitals, and airports. Despite the difference in the training images and the test images, for the image set (internet), precision and a recall over 0.9 can be obtained. It can be seen that higher precision and recall values are obtained as the environment and the camera viewpoints become similar to the training images.

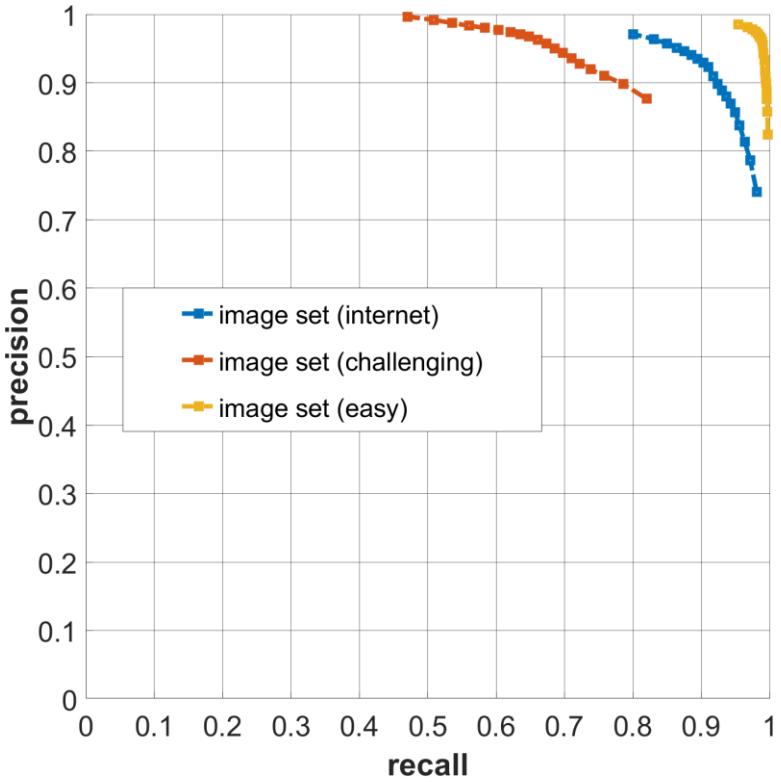


Figure 5.2: Precision versus recall plot of the trained floor segmentation for different data sets

5.3 IMU/Monocular Camera Boresight Calibration

IMU and monocular cameras sense different phenomena. In this thesis, in order to find the boresight calibration, a novel method is developed that does not depend on the estimation of trajectory or utilizing special targets. The developed method is based on detecting normal vectors to the floor plane in an indoor manmade environment using Vanishing Points (VP). These environments often have planar surfaces parallel or perpendicular to each other. Such structural constraints are known as Manhattan World Constraint (MWC) (Peasley et al., 2012) and have been used in the past to detect VP (Elloumi, Treuillet, and Leconge 2014; Gakne and O'Keefe 2017; Huang et al. 2018). Such VPs can estimate the orientation of the camera with respect to the indoor structure. The detection of VPs relies on finding line segments in the environment using Line Segment Detector (LSD) (Von Gioi et al., 2012). Unfortunately, these detected line segments can correspond to highlights, shadows, and shading edges (van de Weijer et al., 2007) which will confuse VP detection algorithm. In order to improve the performance of VP detection, it is often assumed one of the axes of the camera's coordinate frame is parallel to the vertical structure (such as the edges of walls) (Camposeco & Pollefeys, 2015). Therefore, detecting a single VP is sufficient to estimate the orientation of the camera with respect to an indoor structure. Such assumptions entail restricting the camera's range of motion significantly (for example, cameras should be attached to a vehicle that is assumed to move locally on a plane). However, these restrictions are not suitable for estimating the boresight calibration parameters.

In this thesis, a method is developed to detect vertical VP corresponding to the normal of the floor segment in the image frame without restricting the motion of the camera. The normal to the floor plane can be assumed to be antiparallel to the gravity direction in most indoor manmade environments. Since the IMU can easily detect the gravity vector, the unknown orientation parameters of the IMU and the camera can be estimated readily. Figure 5.3 shows the schematic of the developed method.

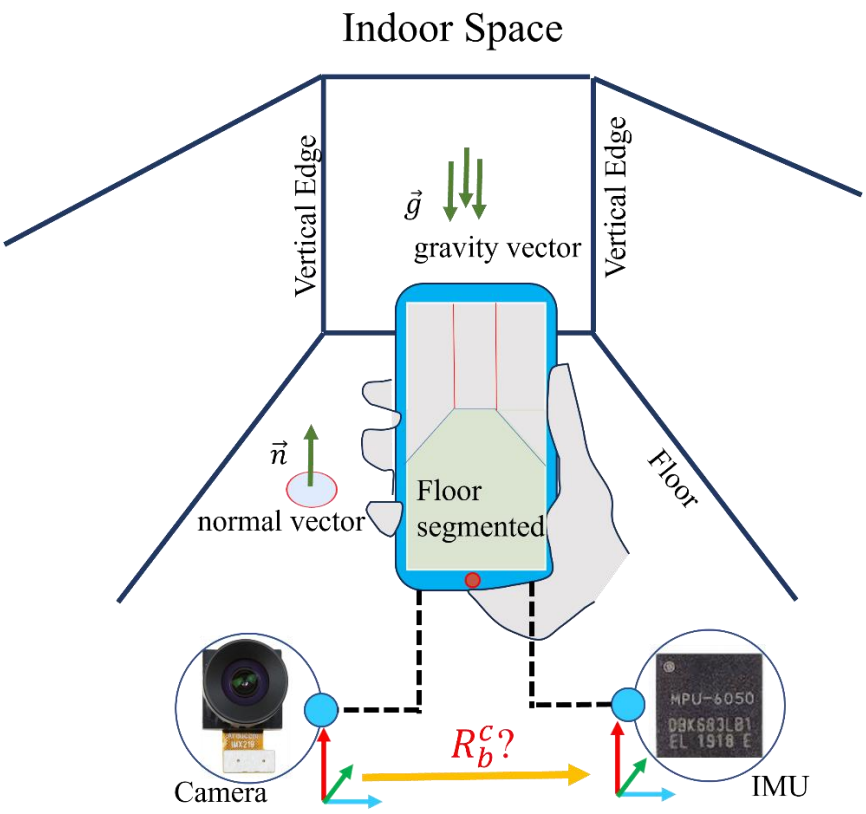


Figure 5.3: Gravity vectors in the indoor environment can be used to find the boresight calibration parameters of an IMU and a camera.

An overview of the developed method is explained in the following (see Figure 5.4), and mathematical details are provided later in this section. The algorithm has three steps. In the first step, floor segmentation is used to detect one of three VPs corresponding to the indoor structure. This VP is found using the intersection of the lines on the boundary of the floor segment (this will be illustrated later in this section). Since the boundary of the floor segment lies on the floor plane, the corresponding VP is denoted as Horizontal Vanishing Point (HVP). In the second step, a line-segment detection algorithm (such as LSD) is applied to the image. The detected line segments that converge to HVP are removed, as these line segments cannot correspond to any vertical structure. Further, the line segments detected inside the floor segment in the image are also removed. In the third step, the remaining line segments are intersected. The corresponding vanishing directions are placed inside histogram bins.

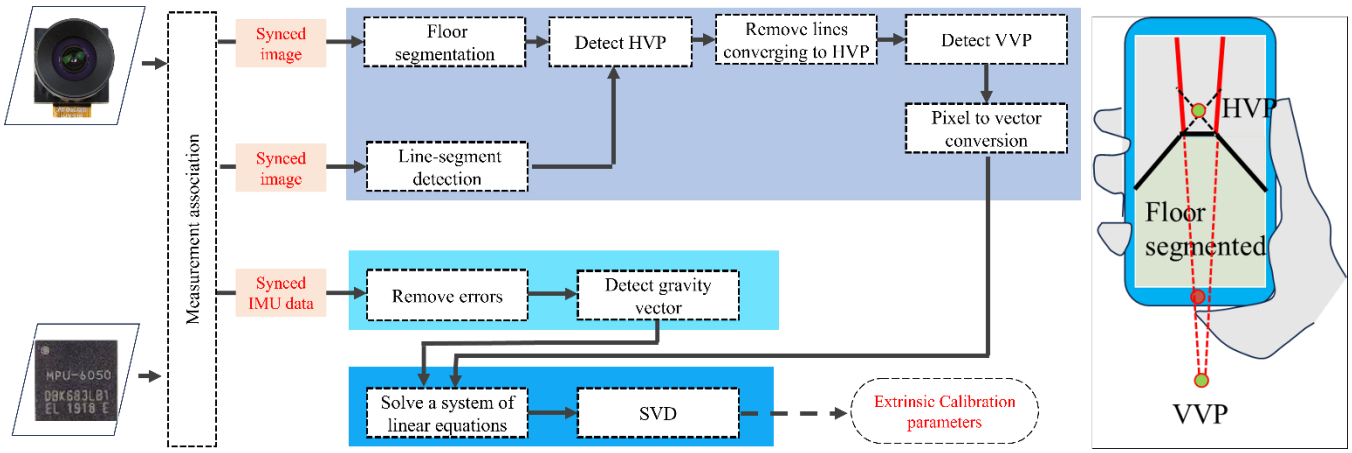
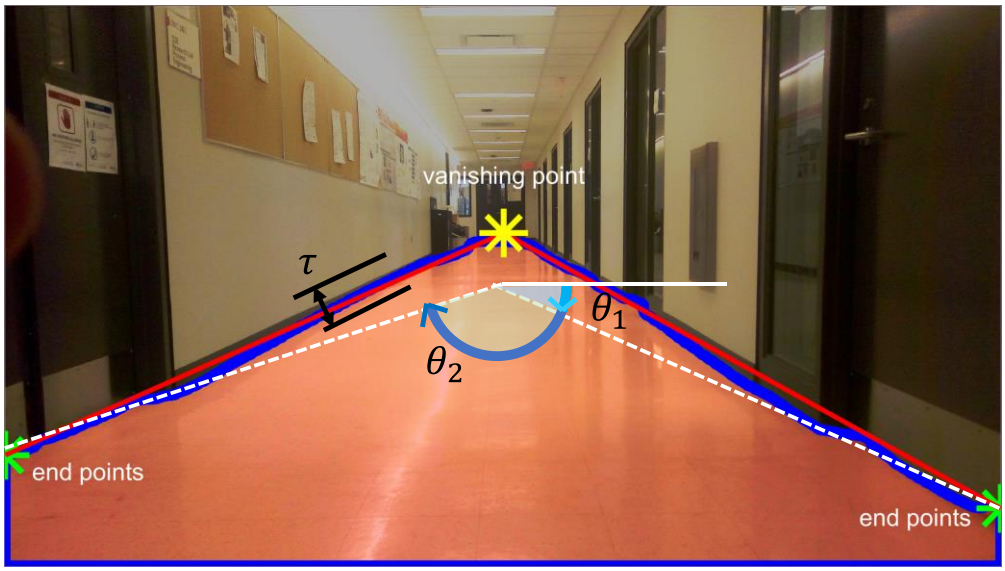


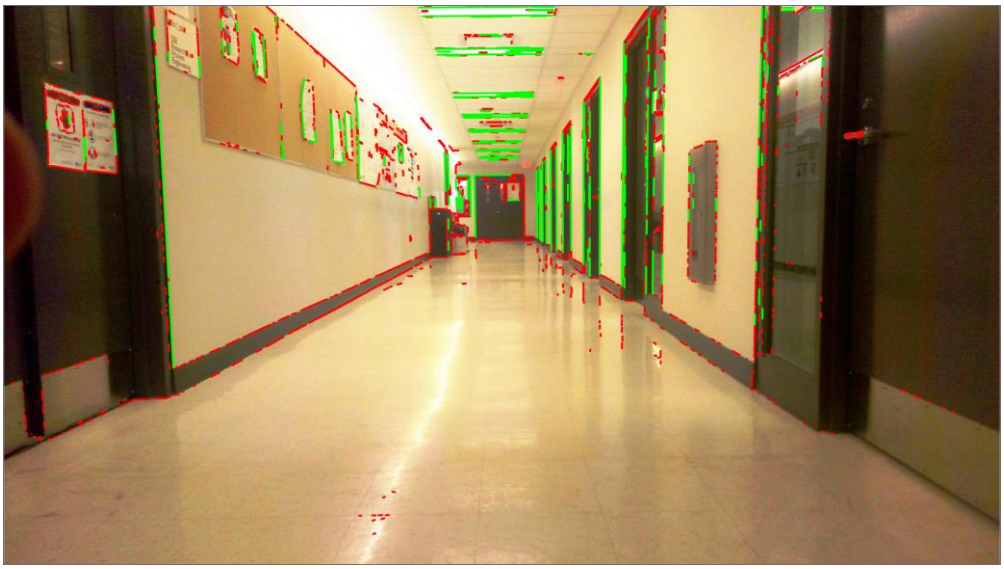
Figure 5.4: The flowchart of the proposed boresight calibration method

Finally, histogram voting is used to select the direction with the largest number of votes. Since the remaining line segments mostly belong to the vertical structure, the bin with the highest vote most likely corresponds to the Vertical Vanishing Point (VVP). The flowchart of this method is shown in Figure 5.4.

In order to find HVP, the boundary of the floor segment is extracted first (the floor segment and its boundary are shown as red and blue in Figure 5.5 (a), respectively). Two points on the floor boundary are designated as endpoints. These endpoints should satisfy two conditions. The first condition is that they should be very close to the edges of the image, and the second condition is that the angle (in the polar coordinate representation of point coordinates) between the two endpoints should be maximum. These two angles are shown as θ_1 and θ_2 in Figure 5.5 (a). Line segments starting from each endpoint are searched in a range from 0° to 180°. Finally, two lines (one for each endpoint) that have the largest number of boundary points falling within a threshold (τ) at its vicinity are selected. These lines are shown in red in Figure 5.5 (a). The intersection of these two lines identifies the HVP (shown in yellow). The line segments corresponding to HVP and the line segment on the floor are removed in order to find the VVP. Figure 5.5 (b) shows the eliminated and survived line segments in red and green, respectively.



(a)



(b)

Figure 5.5: Figure (a) illustrates the detection of HVP. Figure (b) shows line segments (shown in green) that can correspond to vertical structures.

The developed method is explained mathematically in the following. Initially, the detected VVPs (p_j) (one in each image) is converted to direction vectors in the camera frame (v_j^c) using Equation 5.3 (j is the enumerating index). The camera calibration matrix (K) should be estimated using a self-calibration

process. A detailed explanation of the intrinsic camera calibration process is provided in Section 6.7, and the camera calibration matrix (K) here is assumed to be known. Equation 5.4 shows the unknown rotation transformation between the camera and the IMU's coordinate frames. In this equation, the elements of the rotation matrix are shown as r , and the gravity vector in the IMU frame (body frame) is shown as (v_j^b). If the unknown and known variables are reorganized in standard least square form, Equation 5.5 can be derived.

$$v_j^c = K^{-1}p_j / ||K^{-1}p_j|| \tag{5.3}$$

$$v_j^b = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} v_j^c \tag{5.4}$$

$$\begin{bmatrix} v_j^b(1) \\ v_j^b(2) \\ v_j^b(3) \end{bmatrix} = \begin{bmatrix} v_j^c(1) & v_j^c(2) & v_j^c(3) & 0_{1x3} & 0_{1x3} \\ 0_{1x3} & v_j^c(1) & v_j^c(2) & v_j^c(3) & 0_{1x3} \\ 0_{1x3} & 0_{1x3} & v_j^c(1) & v_j^c(2) & v_j^c(3) \end{bmatrix} \boldsymbol{r_{9x1}} \tag{5.5}$$

Each image provides three equations (shown in Equation 5.5). These equations can be stacked in Equation 5.6 (the number of images is shown with the index J). The solution to this simple least square problem is shown in Equation 5.7, which also includes a weight matrix W (will be explained in the following). The estimation parameters $\boldsymbol{r}$ can be reorganized into a rotation matrix ($\tilde{R}$). Finally, Singular Value Decomposition (SVD) can be used to make sure that the rotation matrix is orthonormal. The SVD decomposition is shown in Equation 5.8, where the columns of U and V are orthonormal, and D is a diagonal matrix with positive real elements. The correction to the rotation parameters is shown in Equation 5.9.

$$v_{1:J}^b = A\boldsymbol{r} \tag{5.6}$$

$$\boldsymbol{r} = (A'WA)^{-1}A'Wv_{1:J}^b \tag{5.7}$$

$$\tilde{R} = UDV' \tag{5.8}$$

$$R = UV' \tag{5.9}$$

The precision of the estimated parameters can be calculated using error propagation. In order to achieve this, the covariance of the gravity vector (C^b) sensed in the IMU's frame (v_j^b) should be known. This covariance can be approximated by analyzing the noise characteristics in the accelerometer reading. The variance of this noise is estimated at the beginning of each experiment. In order to achieve this, systematic errors such as biases, scales and non-orthogonalities of the axes are compensated in the raw readings. The remaining errors are assumed to be due to white noise. The process is repeated for each axis of the IMU frame. The estimated covariance matrix is used to create C^b matrix. With the approximated covariance matrix of the observations, the covariance of the parameters $\boldsymbol{r}$ (C^r) can finally be estimated. The relationship between the covariance matrix and error parameters is shown in Equation 5.10. In order to obtain a minimum variance estimation (Petovello, 2003), the weight matrix is substituted with the inverse covariance of the observations as shown (Equation 5.11), and Equation 5.12 can be derived.

$$C^r = (A'WA)^{-1}(A'WC^bWA)(A'WA)^{-1} \tag{5.10}$$

$$W = (C^b)^{-1} \tag{5.11}$$

$$C^r = (A'(C^b)^{-1}\,A)^{-1} \tag{5.12}$$

5.4 Monocular Camera andUltrasonic Sensor Boresight and Lever Arm Calibration

In this section, the extrinsic calibration of an ultrasonic rangefinder and a monocular camera using the floor-segmentation algorithm is explained. The developed method of calibration is based on feature correspondence. The feature correspondence that is established between the two sensors is between the line segments detected in a 2D map (built by the ultrasonic sensor) and the boundary of the floor segment

(detected in the image by the monocular camera).In order to build 2D maps, a robot equipped with an ultrasonic sensor mounted on a rotating platform is used in this thesis (Chapter 6 will provide more information about this platform). Such a rotating platform can produce panoramic range observations of the environment, but unlike LiDARs (which produce dense point clouds), ultrasonic sensors provide Regions of Constant Depth (RCD) (Leonard et al., 1992). In the past, RCDs were used to build maps of the environment using the occupancy grid technique (Elfes, 1989). Unfortunately, such maps do not provide features such as corners and line segments and cannot be used directly to find the extrinsic calibration parameters of an ultrasonic sensor and a camera.

More recently, a mapping method has been proposed (Abadi & El-Sheimy 2022) designed to detect such features using RCDs. Figure 5.6 (a) provides an example to explain this method. In Figure 5.6 (a), the environment that is used for mapping is shown. The features in the environments, such as the corners (shown in the blue box) and line segments (shown as red line segments), can be detected using an ultrasonic sensor. The floor segment (shown in green) can be detected in the image, as mentioned earlier. In this experiment, the robot moves from the starting to the ending points and builds the map. The acquired sparse point cloud is shown in Figure 5.6 (b). Figure 5.6 (c) shows the map built using the line segments (red shows the final results) and the corner features (shown in blue). The occupancy grid map is shown in Figure 5.6 (d) for comparison. Such a map built by line segments can be used in the next step to find the extrinsic calibration parameters. The developed calibration algorithm assumes that the camera's distance from the floor segment is known (h). The known height makes it possible to intersect rays starting from the camera's coordinate origin and passing through the boundary pixels with the floor plane using back projection. Finally, the error function, defined as the distance between the back-projected pixels and the 2D line segments in the map (built by the ultrasonic sensor), can be minimized to obtain the optimum extrinsic calibration parameters. In the following, each step of the extrinsic calibration is explained.

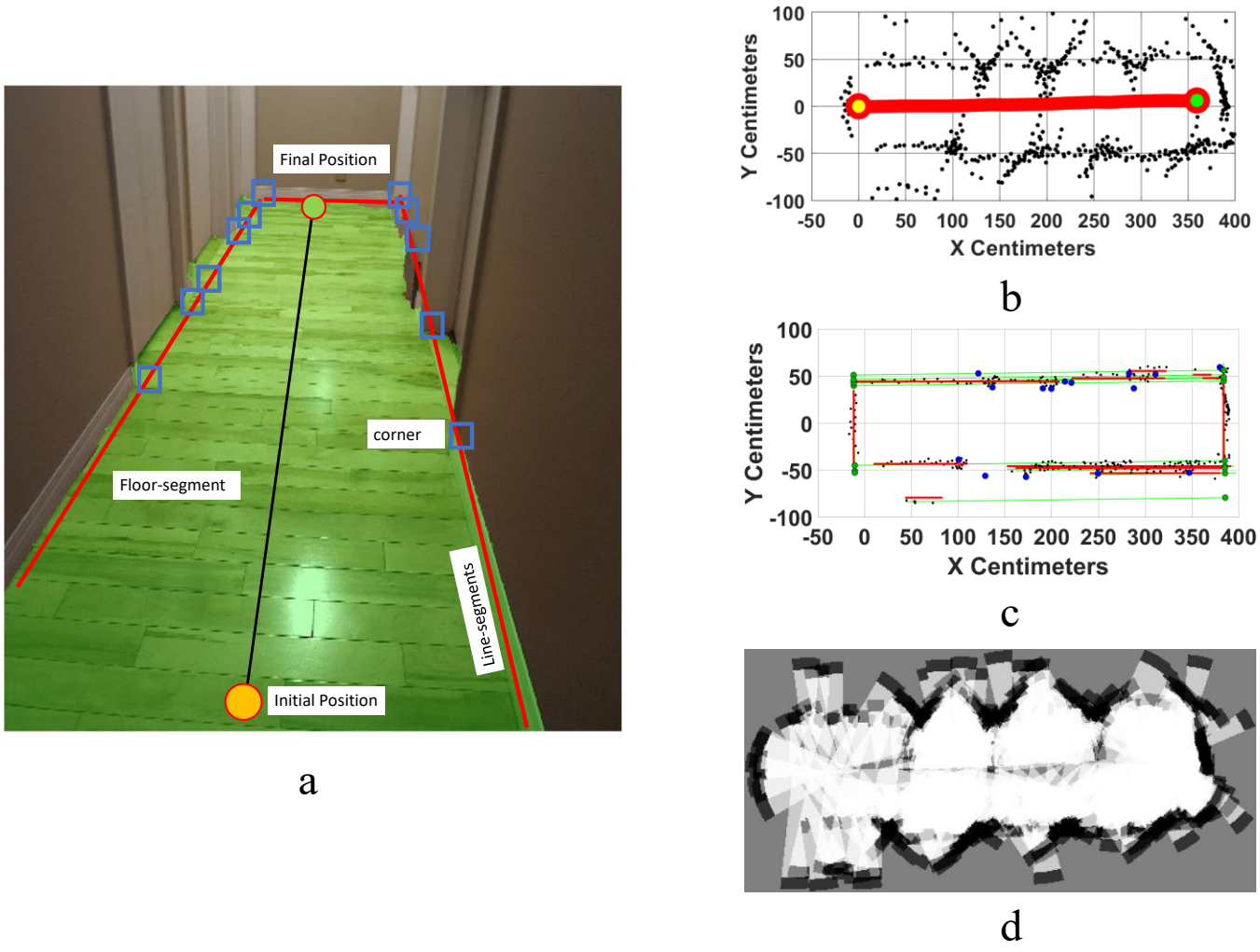


Figure 5.6: Figure (a) shows an image of the environment the robot builds the map. Figure (b) shows the point cloud obtained using an ultrasonic rangefinder. Figure (c) shows the line-segment map. Figure (d) shows the occupancy grid map.

In order to explain the beforementioned steps, the back projection of the pixels of the floor boundary is explained first. In Figure 5.7, the map coordinate and camera frames are denoted as m and c superscripts, respectively. The vector v^m is the direction from the camera center towards the point on the floor. The γ is the angle between the vector -Z^m and v^m. The goal is to derive the coordinates of the point on the floor segment($\tilde{P}$) in the map frame (the **predicted** coordinates of the boundary point in the map frame). The map frame is defined using the initial pose of the robot.

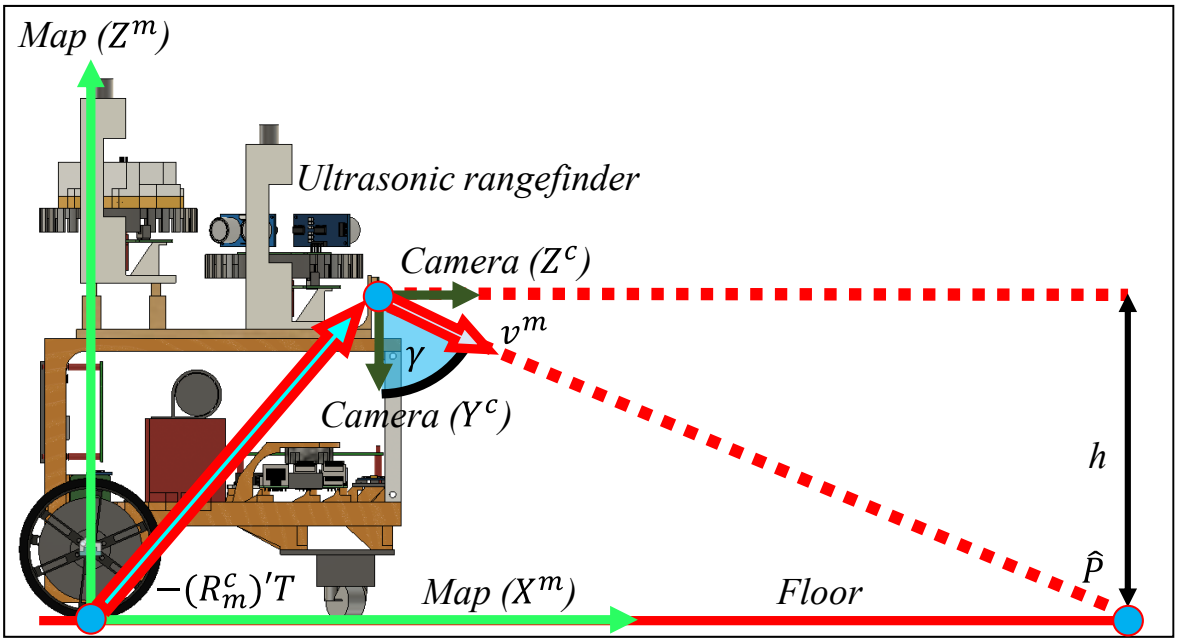


Figure 5.7: Illustration of ray intersection with the floor segment

In order to explain the back projection mathematically, one can commence with the camera's perspective projection shown in Equation 5.13. In this equation, the homogenous pixel coordinates and the intrinsic calibration matrix of the camera are denoted as p, and K, respectively. The p is a pixel that corresponds to the boundary of the floor segment, and it is known using floor segmentation. The rotation and the translation between the camera and the map coordinate are denoted as R_m^c and T (both unknown at this point). The symbol | denotes the concatenation. The direction pointing from the camera's center to a point on the boundary frame (r^c) can be calculated using Equation 5.3 and will not be repeated here. This direction vector is in the camera's frame and should be transformed into the map frame using rotation. R_c^m as shown in Equation 5.15 (where R_c^m is the transpose of R_m^c). Note both r^c and r^m are unit vectors.

$$p = K[R_m^c | T]\hat{P} \tag{5.13}$$

$$r^m = R_c^m r^c \tag{5.14}$$

The γ angle between r^m and Z^m is derived in Equation 5.15 and can be used to calculate the unknown distance between the camera's center and the point $\tilde{P}$ using basic trigonometry as shown in Equation 5.16 (the inner and cross products of vectors are denoted with $<.,.>$ and $\times$). Finally, the scale (α) can be applied to find the position $\tilde{P}$ in Equation 5.17. In Equation 5.17, the vector addition between the translation from the camera's center to the world coordinate ($-R_c^m T$) and translation from the point $\tilde{P}$ to the camera's center (αr^m) is shown. The rotation matrix can be decomposed into three rotations, as shown in Equation 5.18. The objective is to estimate the pose of the camera with respect to the map's frame (($\kappa^*, \varphi^*, \theta^*$) and $T = [x^*, y^*, h]$). This pose is closely related to the extrinsic calibration parameters.

$$\gamma = atan2\left(\left|\left|r^m \times -Z^m\right|\right|, < r^m, -Z^m >\right) \tag{5.15}$$

$$\alpha = h/\cos(\gamma) \tag{5.16}$$

$$\tilde{P} = -R_c^m T + \alpha r^m \tag{5.17}$$

$$R_c^m = R_z(\kappa^*) R_y(\theta^*) R_x(\varphi^*) \tag{5.18}$$

In order to estimate the optimum pose, an error function should be defined. As mentioned earlier, it is assumed that ultrasonic rangefinder has produced line segments and corners as the map of the environment. These line segments correspond to a horizontal intersection of vertical structure. The error function is defined as the distance between these line segments and the back-projected points of the boundary of the floor segment. This error function is shown in Equation 5.19. In this equation $\tilde{P}_j$, P_j, B denote estimated back-projected points, the corresponding point on the line segment, and the set of all the boundary points. The symbol n denotes the total number of points in set B. In order to solve Equation 5.19, the corresponding points P_j should be found first.

$$e(\tilde{P}_j, P_j) = (1/n) \sum_{j \epsilon B} ||\tilde{P}_j - P_j|| \tag{5.19}$$

In this thesis, two methods for building correspondence are used. The two approaches are compared to each other in Figure 5.8. In Figure 5.8 (a), the boundary pixels on the floor segment are shown in red. These pixels are back-projected onto the map in Figure 5.8 (b-e) (shown in red as well). The line segments detected in the map are shown in green (Figure 5.8 (b-e)). The blue points denote the corresponding points. The black line segments (with a darker colour) correspond to the segments used for distance ($||\hat{P}_j - P_j||$) calculation. In the first method for error calculation (e_1), the correspondence is made by intersecting the line segments with the rays passing through the camera's center. These intersection points are shown in Figure 5.8(b, c). In the second approach (e_2), the correspondence is made by finding the closest point in the line segment orthogonally. Figure 5.8(d, e) shows the correspondences for e_2. In Figure 5.8 (b, d), the φ is set at -15°, while in Figure 5.8 (c, e), the φ is set to -10.3°. The larger discrepancy between the back-projected points and the line segments for $\varphi = -15^{\circ}$ is apparent in Figure 5.8 (b, d) (the ground-truth φ is -9.8°). The distances corresponding to e_1 varies more significantly from -15° to -10.3°. The accuracy of the estimated extrinsic calibration parameters using e_1 and e_2 will be compared in Chapter 7.

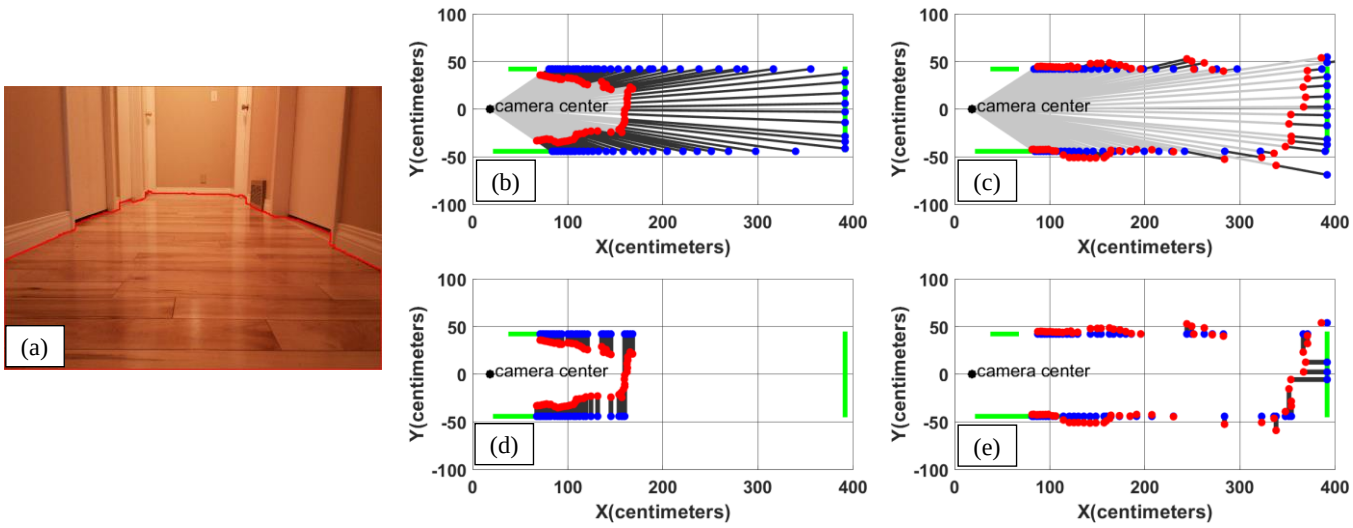


Figure 5.8: This figure shows the correspondence between the line segment detected using an ultrasonic sensor and the back-projected pixels on the image.

Unfortunately, errors e_1 and e_2 are not convex with respect to the parameter of extrinsic calibration. This is illustrated by evaluating both errors in Figure 5.9 (the results for φ and κ are shown). In this figure, the error function e_1 and e_2 are shown in red and blue colours, respectively. It can be seen that convexity is attained only locally. To solve this non-convex problem, an exhaustive search approach can be used. In order to achieve this, the error can be evaluated for sampled angles 0.1° interval. However, due to the high computational complexity of such an approach, the applicability in the real-world application would be very limited. Further, for the translation vector, similar sampling is infeasible. Therefore, to decrease the processing time, angles are optimized in a sequential manner, which reduces the number of tests from a^3 to $3a$ (where a is the total number of angles to be tested in one dimension). Further, the θ^* corresponding to rotation around the axis of the normal floor plane can be optimized using Cox's point-to-line registration approach (Cox, 1991). Cox's algorithm can also provide the in-plane translation (x^* and y^*). The flowchart of the developed algorithm is shown in Figure 5.10. The developed extrinsic calibration is evaluated in Chapter 7.

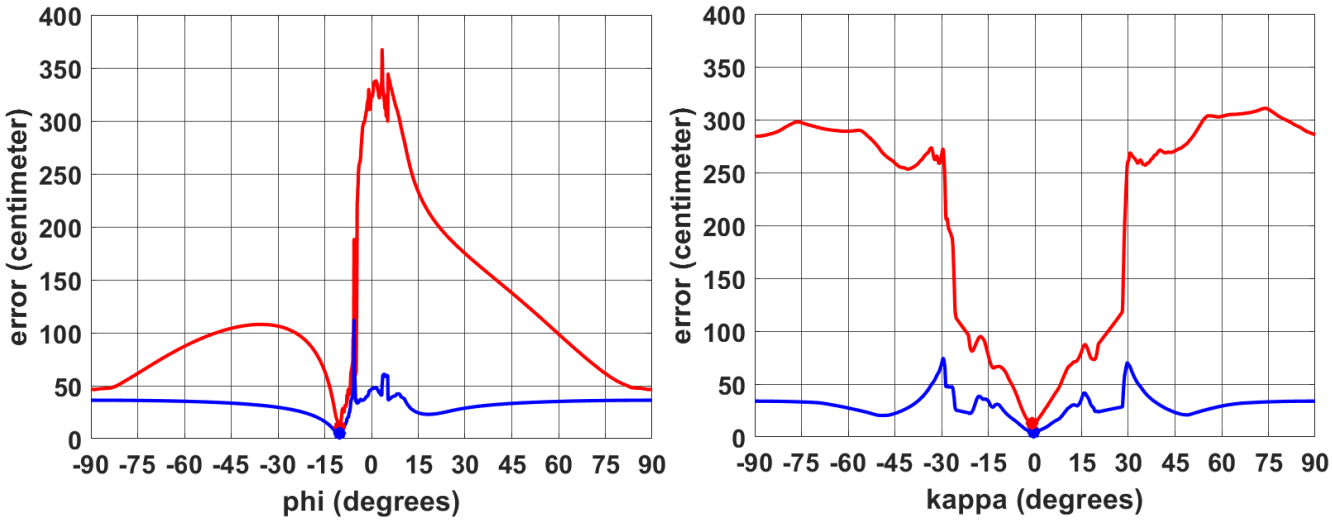


Figure 5.9: Error functions with respect to the φ and κ parameters are non-convex, with only a local convexity

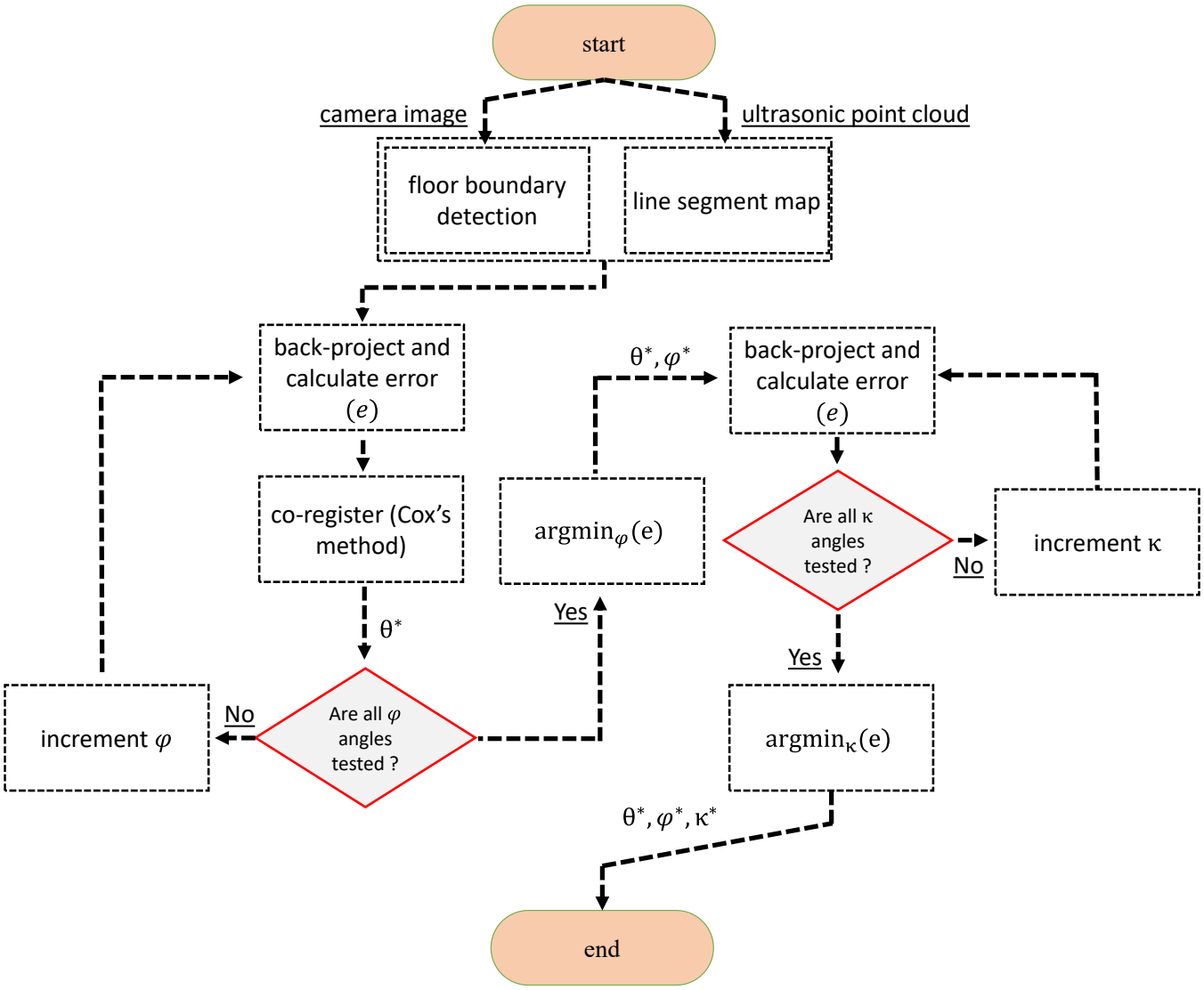


Figure 5.10: The flowchart of the developed method.

5.5 Chapter Summary

The proposed tightly and loosely coupled object-level solution to the SLAM is based on IMU and monocular camera fusion. The fusion of these sensors for the developed algorithm requires the estimation of boresight and the lever arm calibration parameters. These important parameters are often difficult to obtain. While in the outdoor environment, IMU/GNSS integration is often required to find these parameters, in the indoor environment, such solutions are not possible to utilize. Thus, this chapter

develops a new method to take advantage of the MWS in indoor environments to address such challenges. This method utilizes floor segmentation and vanishing point detection to identify vertical direction. The vertical direction is further assumed to correspond to the direction of the gravity vector. This correspondence is utilized to obtain the boresight calibration parameters of an IMU and a monocular camera.

In order to obtain the boresight and lever arm calibration parameters of an ultrasonic sensor and a monocular camera, a similar approach is utilized. In this method of calibration, the boundary of the detected floor segment in the image is matched to the 2D line-based maps (built using an ultrasonic sensor). This approach is useful for wheeled mobile robots, where such small 2D maps can reliably be built, often using MWC.

In Chapters 3 to 5, the main methodology and the required components of the developed object-level solution are explained. In order to test the developed method, two platforms are utilized. The first platform is a handheld device, and the second platform is a mobile-wheeled robot. The details of these platforms and the methods to estimate the reference solutions are explained in the next section.

Chapter 6: Experimental Setup

6.1 Overview

The developed RBPF-SLAM can be implemented on different platforms. Wheeled Mobile Robots in the indoor environment are one of the common platforms that can be used with the developed approach. Section 6.2 provides the specification of the developed differential-drive mobile robot for small indoor environments (e.g., most domestic environments). The designed platform includes ultrasonic rangefinders, a monocular camera, and wheel encoders. This robot is used to evaluate the wheel odometer/ monocular camera tightly/loosely coupled object-level SLAM.

With the advent of smartphones, handheld devices have gained larger popularity. Unlike wheeled robots, these devices can move fast and in directions that are less predictable. In order to show that the developed object-level RBPF-SLAM solution can be utilized under such circumstances, a handheld device is designed. This device includes a Micro Electronic Mechanical System (MEMS) IMU, a monocular camera and an ultrasonic rangefinder. The platform is used to evaluate the IMU/monocular camera tightly/loosely coupled object-level SLAM. The specifications of this device are provided in Section 6.3.

In order to assess the performance of the developed methods, the estimated results should be compared to a reference solution. Reference solutions can help evaluate precision, recall, position accuracy, orientation accuracy, and so on. References solutions are denoted as ground truth if obtained through measurements. In order to evaluate the performance of the developed RBPF-SLAM methods for each of the abovementioned platforms, different ground-truth estimation methods are utilized. For the differential wheel robot, the first type of ground truth is obtained by measuring the distance vector between the robot's initial and final poses. This measurement is obtained using distance measuring tools (such as measuring tape); therefore, no further explanations are provided. The second type of ground truth is obtained using an accurate beacon-based positioning system. This system can estimate the pose of the

robot at certain points along the trajectory during the navigation. This beacon-based positioning system is explained in Section 6.4. For the handheld device, unfortunately, the abovementioned beacons cannot provide a 6DoF pose estimation during the motion. In this case, the ground truth is estimated by measuring the distance of the device's initial and final poses. Due to the simplicity of this approach, it is not explained further. It is also important to note that measuring position and orientation error is not the only means to assess the performance of the developed object-level solution. Other assessment metrics based on reprojection error, IoU (between the estimated and observed contour of the object), can also provide insights into the performance. These are explained in Chapter 7 in detail.

The boresight calibration of IMU and the monocular camera is evaluated on the developed handheld. For this device, an accurate CAD model exists; therefore, no further considerations are given to obtain reference solutions. The monocular camera and IMU boresight calibration are also implemented on smartphone devices. The method of estimating ground truth solutions for smartphones is explained in Section 6.5. Section 6.6 explains how the reference solutions are obtained for the object pose estimation. Section 6.7 provides details about how the intrinsic calibration parameters of the cameras are estimated. Finally, Section 6.8 provides details of the trained DNNs.

6.2 The Designed Differential-drive Robot

In indoor manmade environments such as houses, a robot's overall size should often be small. The smaller size and smaller power often pose restrictions for the type of sensory equipment that can be installed in these robots. However, it is often expected for a robot to be equipped with lightweight 2D rangefinders (e.g., ultrasonic sensors), wheel encoders and monocular cameras. A platform is designed to be able to store the data received from each of these sensors in a stop-and-go fashion while at the same time being able to communicate (send data and receive commands) with the user. The 3D model of this

robot and the included sensors are shown in Figure 6.1. The robot includes one microprocessor (Raspberry Pi) and two microcontrollers (Arduino and ESP8266).

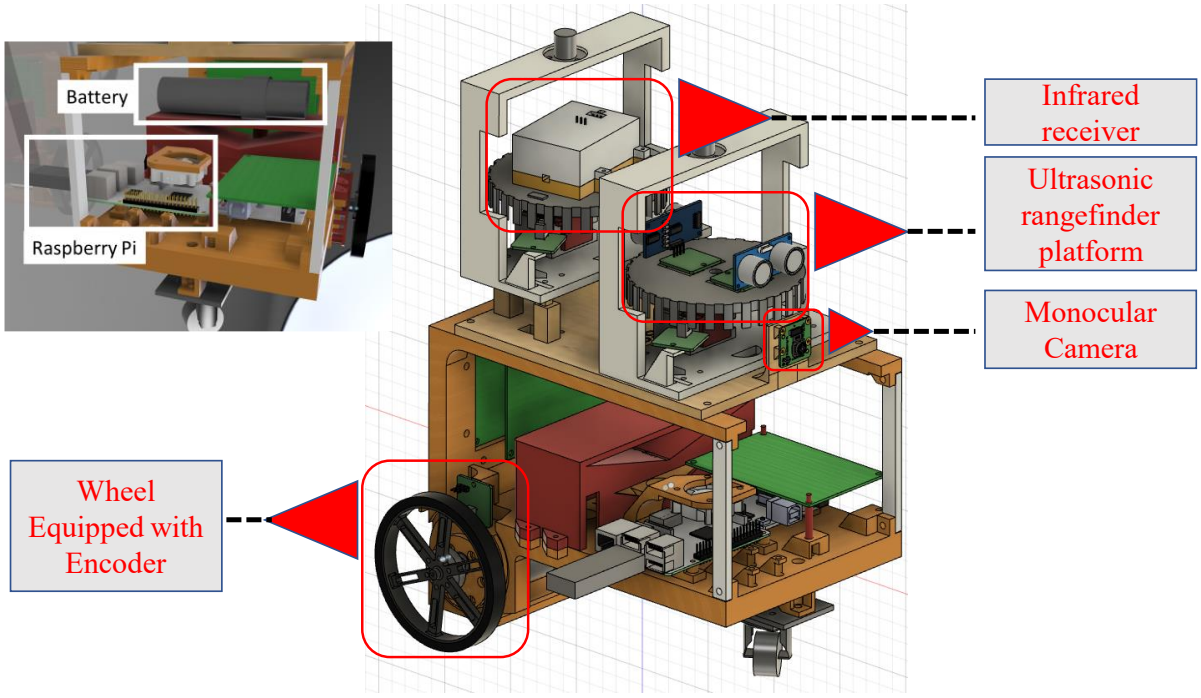


Figure 6.1: The CAD model of the designed indoor robot. The designed platform includes many sensors, such as an ultrasonic rangefinder (seen in the close view on the right), a monocular camera and an infrared receiver.

The details about the equipped sensors are provided below:

1. The monocular camera used is a Raspberry Pi Camera Module V2 (RPV2) with 8-megapixel. RPV2 is developed for smaller projects with often lower available processing power. The maximum resolution of this camera is 3280×2464, and it has a fixed focal length. The intrinsic parameters for this camera are acquired using MATLAB's Camera Calibrator toolbox.
2. In order to build point clouds, two HC-SR04 ultrasonic rangefinders are mounted on a platform. The platform rotates while the ultrasonic rangefinders measure the distance. Once the platform completes one rotation, it returns to its initial pose. Figure 6.2 shows the rotating platform and an example of a collected sparse point cloud of the environment. The platform itself (as it can be seen

in Figure 6.2) is an encoder disk. Therefore, the rotation can be counted with high accuracy. Such an encoder has a simple mechanism, and its output is either 0 or 1 based on whether the photo-interrupter's signal pathway is blocked. Due to the simplicity of such a design, often, no calibration is required.

3. Similar encoders are used to measure the angular velocity of the wheels. There is no specific calibration required for these wheel encoders as well.
4. Finally, an infrared (IR) receiver is included with this robot. This receiver is crucial to provide the ground truth estimation, and it is explained in more detail in Section 6.4 (and Appendix B). This receiver is also mounted on a rotating platform with a defined initial pose.

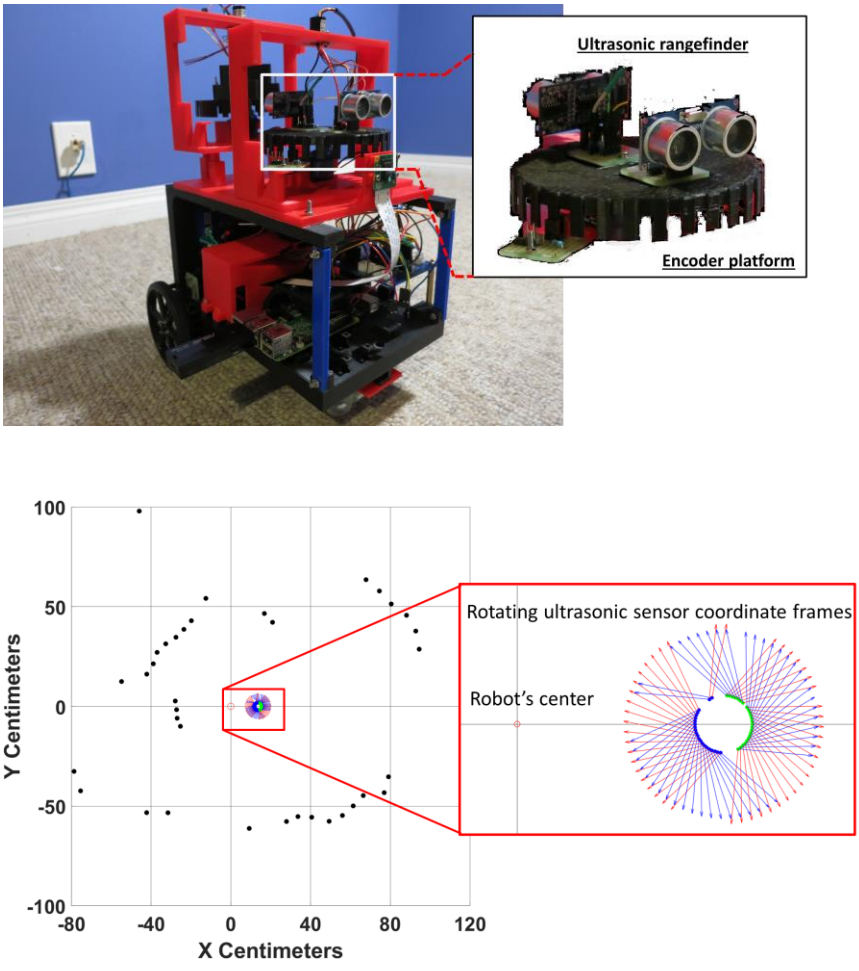


Figure 6.2: The designed ultrasonic rangefinder and an example of a point cloud generated by this platform.

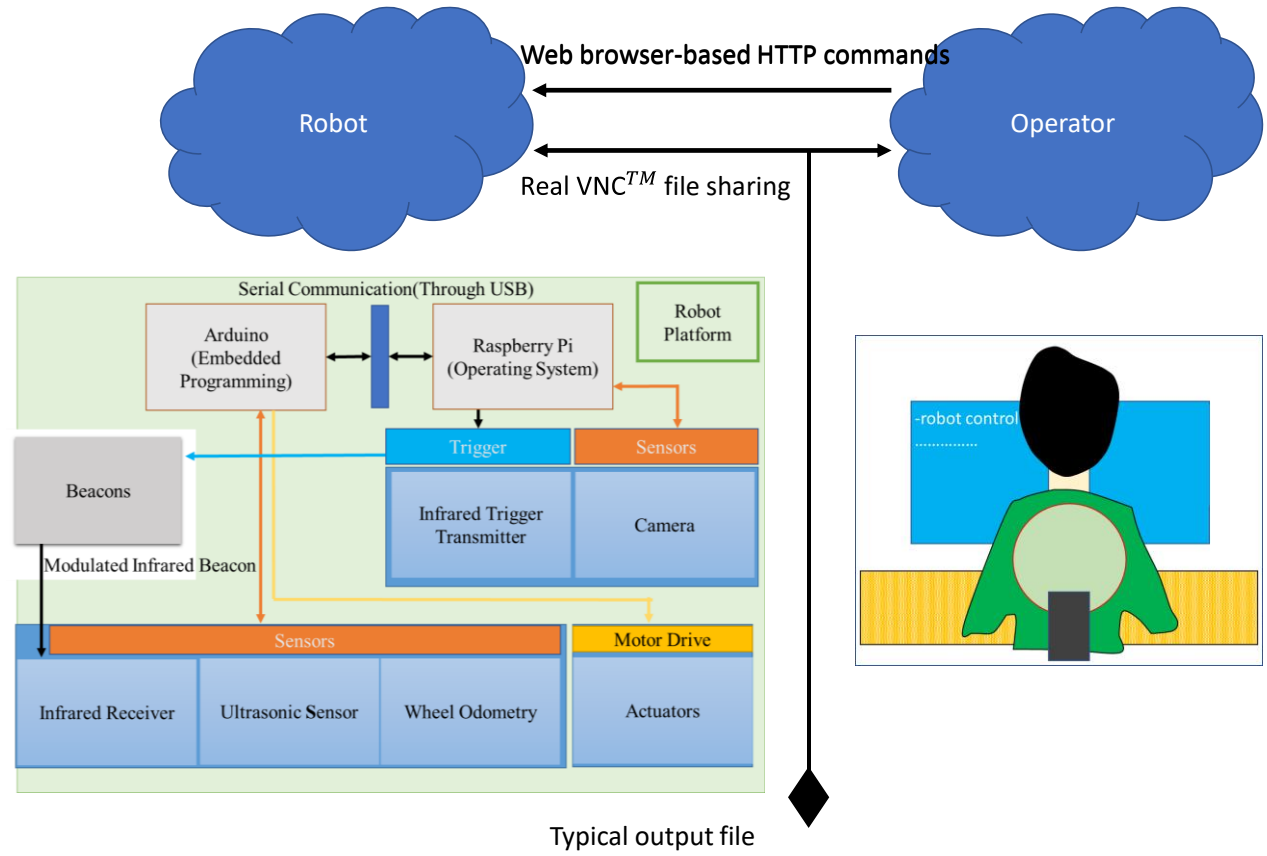
The origin of the body frame is defined at the center of the line connecting the two wheels. The motion model for this robot was explained in detail in Section 3.5; therefore, repetition is avoided here. An overview of the sensors on the board is shown in Table 6.1. The extrinsic calibration parameters between the sensor and the coordinate frame of the body are obtained from the 3D CAD model.

Table 6.1: The specifications of the sensors and calibrations used in the differential wheel robot.

Sensor	***Intrinsic calibration***	***Extrinsic calibration***	***Additional information***
Monocular camera	MATLAB's toolbox	Provided by 3D CAD model	Raspberry Pi Camera Module V2
2D rangefinder	No calibration is required	Provided by 3D CAD model	HC-SR04 mounted on a rotation platform
Wheel encoders	No calibration is required	Provided by 3D CAD model	Custom-built encoders with large disk
Infrared receiver	No calibration is required	Provided by 3D CAD model	Used to estimate the ground truth position

The schematic of the designed software of the system is shown in Figure 6.3. The microprocessor (Raspberry Pi) and the two microcontrollers (Arduino and ESP8266) are each responsible for communicating with certain sensors/actuators. The tasks shared between the microcontroller and the Raspberry Pi ensure that there is shorter latency created compared to delegating all the tasks to a single processor. The main software runs on Raspberry Pi. Once this software starts, client-server communication through the HTTPS interface becomes available to the user. The user can upload a predetermined sequence of commands as a text file or can send individual commands separately. The obtained sets of commands (such as actuating the motors or requesting data acquisition from one of the sensors) are sent from Raspberry Pi to appropriate microcontrollers and sensors. Once the command is sent, the main software waits until it receives a response and then saves the results of running that command. The output files include text files (as seen in Figure 6.3) and images. This text file includes the readings from each wheel encoder, the point cloud from the ultrasonic rangefinder platform, and observations by IR receivers. The images obtained using Raspberry Pi PiCamera are referenced in this

text file as well. Each sensor reading is timestamped using Coordinated Universal Time (UTC). In this example shown in Figure 6.3, the robot activates the two ultrasonic rangefinders and reads the distance measured (the distances readings are 8.0 and 43.3 centimetres). In the second step, the robot captures an image. In the third step, the main program sends a command to Arduino to active motors and then reads the angular velocities measured by the wheel encoders (in this example, 0.64 and 0.61 turns per second with a duration of motion of 0.89 seconds). Finally, it records the data received from the beacons.




```
['1652806909.2338755', 'ultrasonic', '8.0', '43.3']
['1652806920.5067935', 'PiCamera', 'source', 'img_0000.jpg']
['1648610748.8857887', 'motionandsense', 0.64, 0.61, 1.0, 1.0, 0.89]
['1652806924.9619453', 'infrared-complete', '0 0 0 0 0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1']
```

Figure 6.3: In this figure, the communication between the processors and microcontrollers, as well as the communication between the operator and robot, is shown. The robot also receives signals from the beacons to measure the ground-truth position and orientation.

6.3 The Designed Handheld Device

Handheld devices have gained a lot of attraction in recent years. Unlike wheeled robots that often have a constrained motion to a plane, handheld devices can move in a wider range of directions. Further, these devices can be subjected to vibration of the hands of the user. Therefore, it is important to investigate the applicability of the developed solution for handheld devices. In this thesis, a handheld device that is equipped with a monocular camera, IMU and an ultrasonic rangefinder is designed. The device is operated using a Mini Desktop PC (Beelink Mini S), which is relatively inexpensive compared to smartphones nowadays. The designed platform is shown in the images in Figure 6.4. The monocular camera board is designed by Arducam and equipped with an 8 Mega Pixel CMOS IMX219 sensor, which is a low-cost and low-resolution camera. The IMU is an Xsens mti-g-710 which is a MEMS sensor with lower cost compared to tactical grade IMUs. The noise density of the gyroscope and accelerometer for this sensor is 0.01° s/$\sqrt{HZ}$, and 60 $\mu g/\sqrt{HZ}$ on each axis. The ultrasonic sensor is HC-SR04 with 15° angular resolution (and approximately ± 2 cm error in the range reading). The same model was used in the previous platform. The data is collected and timestamped using the Mini Desktop PC.

The intrinsic calibration of the monocular camera is obtained using MATLAB's toolbox. The IMU is calibrated using the 6-position static test, which provides biases, scales, and non-orthogonality error parameters. In practice, bias is the most important error parameter recommended to be estimated for the developed RBPF-SLAM. The developed RBPF-SLAM solution can mitigate other types of errors caused by scale and non-orthogonality. As is the case with most other proposed methods in the past, no specific calibration for the ultrasonic sensor is performed. However, in Chapter 7, it will be shown the best results will be obtained by setting the angular resolution to 14°-16° which is very close to the nominal value. Table 6.2 provides some information about the sensors used in this platform. The extrinsic calibration parameters are all estimated with the help of a CAD model.

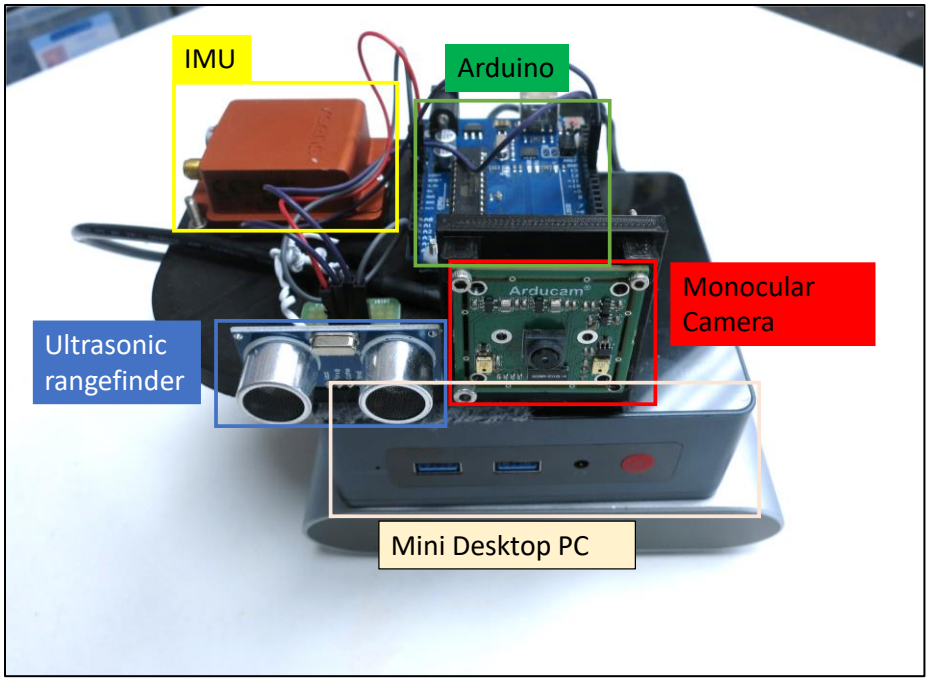


Figure 6.4: The designed handheld device includes an IMU, a monocular camera and an ultrasonic rangefinder.

Table 6.2: The specifications of the sensors and calibrations used in the handheld device.

Sensor	***Intrinsic calibration***	***Extrinsic calibration***	***Additional information***
Monocular camera	MATLAB's calibration toolbox	Provided by 3D CAD model	Arducam with 8MP IMX219
2D rangefinder	No calibration is required	Provided by 3D CAD model	Single HC-SR04
IMU	6-position static test	Provided by 3D CAD model	Xsens mti-g-710

The software design of the platform is shown in Figure 6.5. The main software runs on the Mini Desktop PC. This software is responsible for receiving data from the sensors, and it is coded in Python programming language. The images are captured using OpenCV libraries. The data from IMU is captured using the Software Developing KIT (SDK) provided by the manufacturing company of Xsens. The ultrasonic sensor data is captured on an Arduino and subsequently sent to the Mini Desktop PC using Universal Asynchronous Receiver Transmitter (UART) communication which is one of the most common methods of sending and receiving data between an Arduino and a PC. Figure 6.5 shows an example of possible output files. The reading of each sensor is timestamped with UTC.

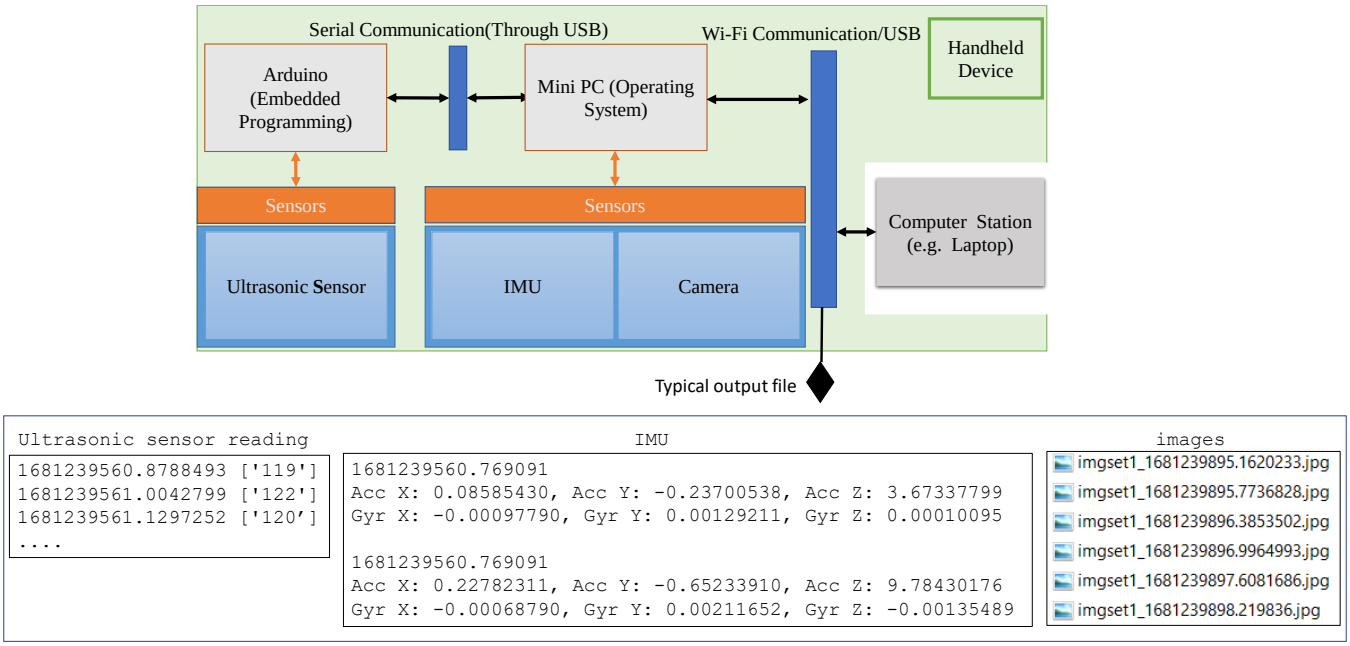


Figure 6.5: Overview of the architecture of the designed handheld device.

6.4 Infrared-based Bearing-only Beacons

In this section, the developed indoor beacons are introduced. This system is designed to provide ground truth heading and the position of the robot and depends on establishing a Line of Sight (LoS) from the beacon to the receiver (mounted on the robot). This can be a limitation in many scenarios; however, the accuracy of the developed indoor beacon system is much higher (approximately 3 to 11 cm) compared to state-of-the-art Bluetooth-based beacons (approximate 1-meter error) (Ivanov, 2021; Jin et al., 2023).

The developed system estimates the position based on triangulation equations. In the past, the solution to these non-linear equations is based on linearization (e.g., EKF). The uncertainty of such a solution can be evaluated locally using the covariance matrix. The linearization of the equations of triangulation is avoided in this thesis. This can be achieved by utilizing the fact the environment of the experiments is relatively small (limited 10 by 10 meters), which provides an opportunity to evaluate the likelihood analytically in a grid defined in the map. Further, since the overall size of the environment is small, the cell size can be chosen to be with dimensions as small as one centimetre (see Appendix B)

The developed system can be divided into transmitter and receiver modules. Three transmitter modules are shown in Figure 6.6. Each transmitter consists of a programmable microcontroller, a circuit with a 555 timer Integrated Circuit (IC), and an array of IR transmitters that send modulated signals at 38kHz. The receiver module is mounted on the rotating platform of the robot. This receiver is concealed inside a box which blocks light, including IR, from penetrating in all directions except at a narrow-angle. By limiting LoS, it is ensured that higher angular precision can be achieved. The receiver is triggered every time it senses a modulated IR signal. Once the IR receiver completes a rotation, the output is a binary value (indicating triggered or not) at each step. Each step is defined by a fixed angle of rotation (approximately 6°). Once the binary sequence is acquired, the average angle within the **runs** is detected (a run is defined as a sequence of uninterrupted 1s, as shown in Figure 6.7). The final output of this process is a series of angles, denoting the observed orientation from the robot to each beacon.

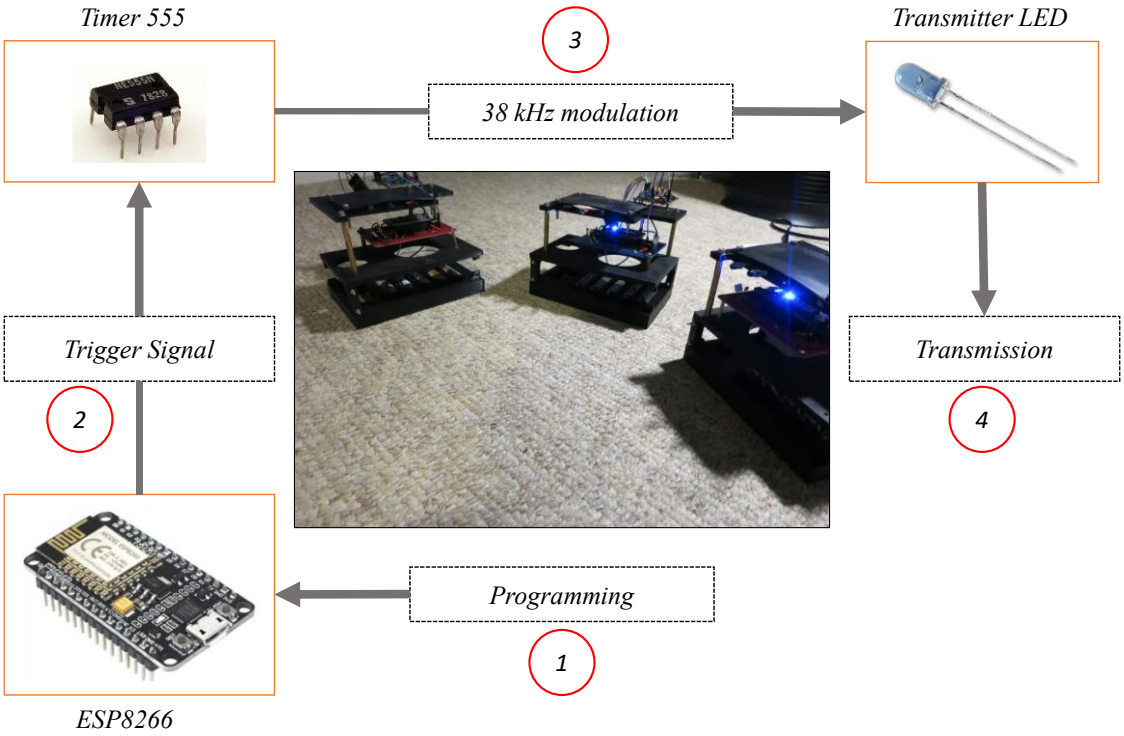


Figure 6.6: A depiction of the transmitter modules of the beacon-based positioning systems.

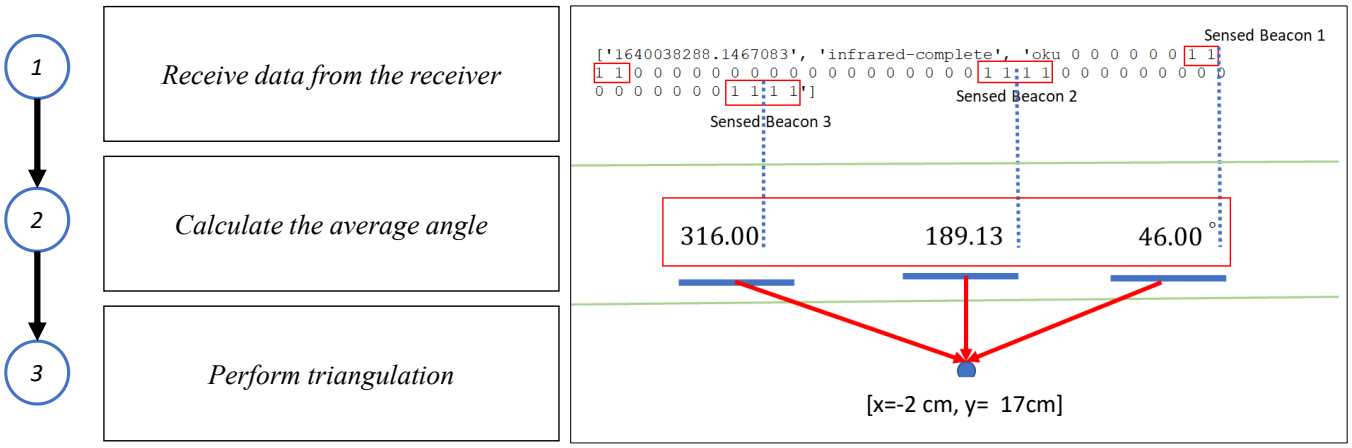


Figure 6.7: The procedure of estimating position using observed angles from the robot to each beacon

The obtained angles can be used to solve equations of triangulation. As mentioned, the developed approach achieves this by evaluating the observation likelihood in the map. A number of positions estimated using the beacons and the true position are shown in Figure 6.8. In these examples, three beacons are used (labelled in yellow). The likelihood in the map is encoded with the colour intensity (the darker the colour, the higher the likelihood). The true position of the robot is shown as green, and one thousand cells with the highest estimated observation likelihoods are shown in blue. As is known in typical beacon-based (or satellite-based) positioning systems, the estimated position can be affected by the geometry of the beacons and the robot. The estimated observation likelihood clearly shows the effect of the geometry on the accuracy of the estimated position as well. For example, when the robot is close to the line going from one beacon to another, the uncertainty in the direction of this line is larger. At the same time, the uncertainty decreases substantially perpendicular to this line (see also Figure 6.9). In contrast, as the robot moves to the centroid of three beacons, the uncertainty becomes more isotropic (more uniformly distributed errors in x and y directions). Quantitively the accuracy of the method is measured to be in the range of 3 to 11 cm, based on the geometry throughout an extensive study around the indoor environment.

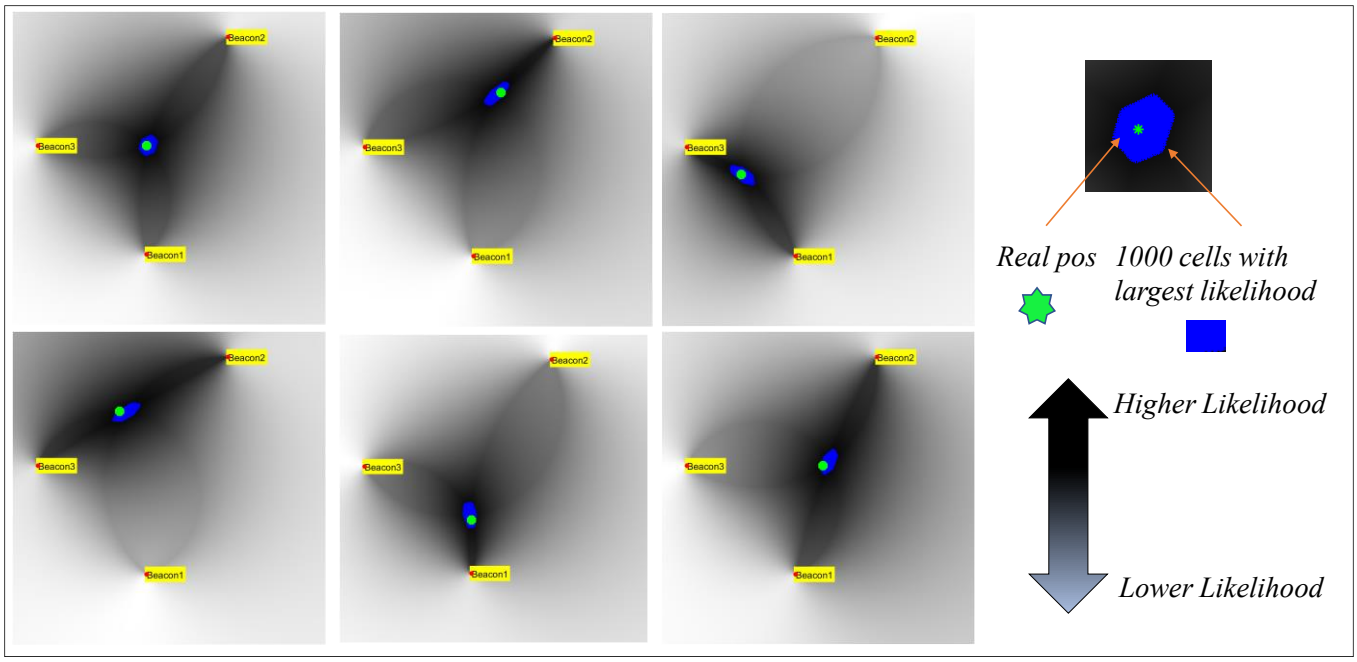


Figure 6.8: The estimated location and the observation likelihood of the robot using three beacons. The geometry of the location of the beacons and the robot affects the uncertainty.

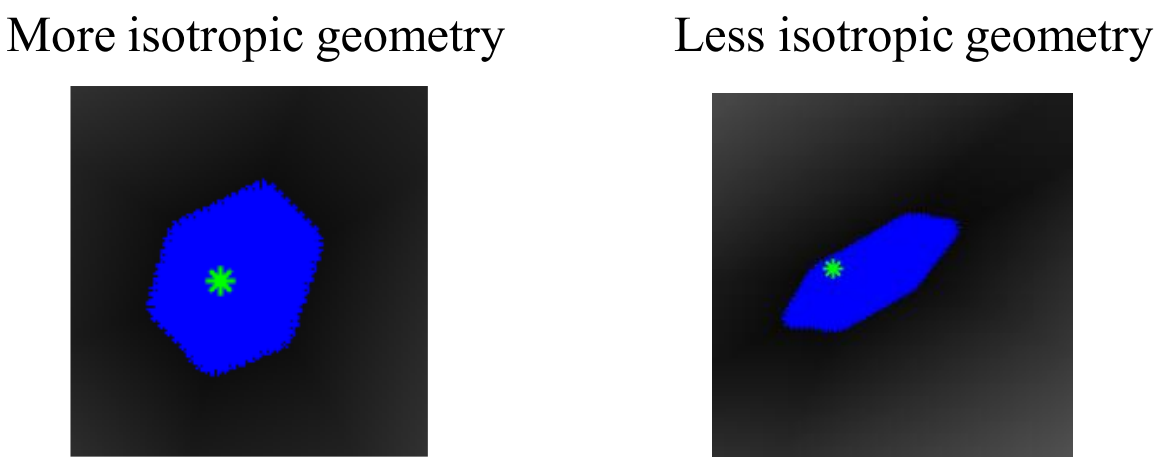


Figure 6.9: The uncertainty of the estimated positions is due to the geometry.

Table 6.3: The lowest error is achieved when the geometry is more isotropic in all directions (approximately 3 cm). However, the error increases to 11 cm for all the tests.

Set of test positions	***Accuracy (Absolute Mean Error)***
More isotropic geometry subset	~ 3cm
All the tests	~ 11cm

6.5 Ground Truth for Boresight Calibration

In order to assess the developed boresight calibration of the IMU and the monocular camera, experiments are performed using two devices. One of these is the handheld device introduced earlier. The reference boresight calibration parameters are obtained from the 3D CAD; therefore, no further attempt is made to measure this device's ground truth calibration parameter.

The second device tested for boresight calibration is a smartphone. Here it is assumed that the orientation parameters between a smartphone's camera and IMU are not provided by the manufacturer, and therefore, it should be estimated. The platform is shown in Figure 6.10. The 3D printed box is used to estimate the ground truth boresight calibration parameters; however, the boresight calibration method can be used alone with the smartphone. The ground truth estimation is very similar to the developed boresight calibration method, and it utilizes the gravity vector. The designed platform (see the black box in Figure 6.10) is aligned with gravity in at least one image. In order to ensure that the direction is aligned with gravity, a leveling instrument is utilized. The image captured at this leveled pose is saved as the **first image**.

Figure 6.10: The devices were used to test the IMU/monocular camera boresight calibration.

The relative poses of all other images with the first image are estimated with the help of the fundamental matrix (F), as shown in Equation 6.1. In this equation, homogenous coordinates of image pixels are denoted as p_k^i and p_k^j (i and j are the image indices, and k is the corresponding pixel index). The fundamental matrix is a 3×3 matrix, and therefore it has 9 unknown parameters. However, this matrix only has 7 degrees of freedom (at least 7 independent equations are required to solve for F). The well-known eight-point approach can be used to estimate F (Hartley, 1997). In the experiments, in order to find the fundamental matrix accurately, Apriltags (Olson, 2011) are used as the features. Aprtiltags are fiducial markers, and each maker has an associated identification, which makes it possible to establish a correspondence between two tags captured from different images (see Figure 6.11). Once the correspondences are found, the fundamental matrix in 6.1 can be estimated.

$$\left(p_k^j\right)' F p_k^i = 0 \tag{6.1}$$

The obtained fundamental matrix can be decomposed into a translation (T) (up to an unknown scale (s)) and a relative rotation matrix R_i^j using Equation 6.2 ($[T]_\times$ is the skew-symmetric T). Assuming one of the frames i or j is the first frame (f) (here $i = f$ without loss of generality), the relative rotation matrix can be used to estimate the gravity direction in the other image (Equation 6.3). This estimation is possible since the gravity vector is known in the first frame. Since the gravity vector is known in the IMU's frame (b), the unknown rotation matrix between the IMU and the camera can be shown in Equation 6.4. This matrix can be estimated using the method explained in Section 5.3 and will not be repeated here. Figure 6.11 shows a schematic of the explained process, where the known transformations are shown in green arrows and the unknown boresight calibration is shown in red (which can be estimated with the help of known transformations).

$$F = (K')^{-1}\left([T]_\times R_f^j\right)K^{-1} \tag{6.2}$$

$$v_j^c = R_f^j v_f^c \tag{6.3}$$

$$v_j^b = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} v_j^c \tag{6.4}$$

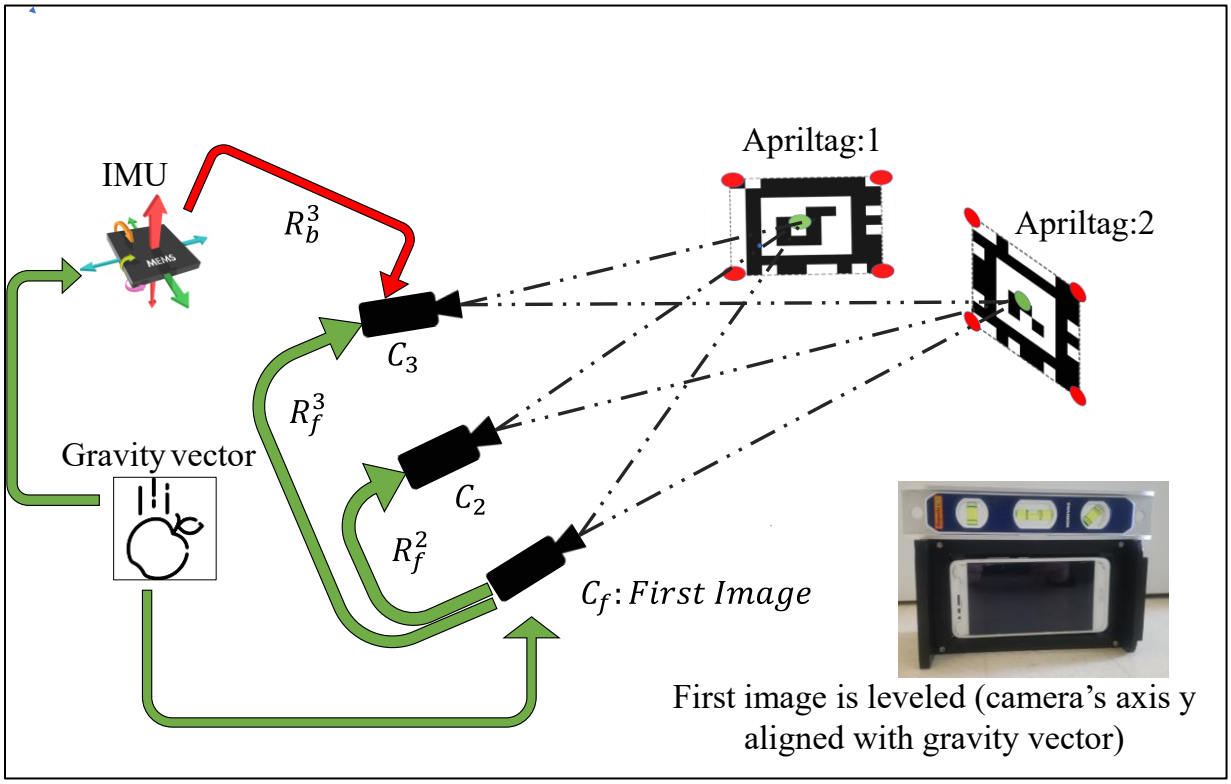


Figure 6.11: The process of estimating ground truth boresight calibration parameters.

6.6 Reference Solution for Pose Estimation

Object pose estimation is one of the important aspects of the developed solution. In addition to measuring the accuracy of the RBPF-SLAM in terms of the final position and some positions during the navigation, the accuracy of the pose estimation is also measured independently in this thesis. These experiments to measure the accuracy of the pose estimation is towards assessing the coarse-to-fine method explained in Chapter 4.

Obtaining a reference solution for the pose of an object is a challenging task. In order to obtain such a solution for the pose of the camera with respect to an object, the following steps are taken. Initially,

the object is placed on a turntable. Then the homography (H) between the planar surface of the turntable's plate and the camera is found. In order to estimate the homography, a correspondence of at least four points in the image and the planar structure should be established. In the experiments, this homography can be estimated by selecting four corners of the surface of the turntable's plate. In Figure 6.12, the projected points from the world frame to the camera's frame are shown with the help of estimated homography. The center of the turntable's plate is also defined to be the center of the world coordinate frame. The homography can be decomposed into a rotation and a translation (up to an unknown scale s) if the calibration matrix (K) is known (Zhang, 2000). This can be achieved by using Equations provided from 6.5 to 6.8 ($r_{1:3}$ denotes the column vector $r = [r_1, r_2, r_3]'$). Here it is assumed the length and width of the turntable plate are known accurately (since the plate is 3D printed). Therefore, this provides an accurate estimation of the scale. The definitions of the homography, rotation matrix and translation vectors are provided in Equation 6.9.

$$r_{1:3} = sK^{-1}h_{1:3} \tag{6.5}$$

$$r_{4:6} = sK^{-1}h_{4:6} \tag{6.6}$$

$$r_{7:9} = r_{1:3} \times r_{4:6} \tag{6.7}$$

$$T = sK^{-1}h_{7:9} \tag{6.8}$$

$$H = \begin{bmatrix} h_1 & h_4 & h_7 \\ h_2 & h_5 & h_8 \\ h_3 & h_6 & h_9 \end{bmatrix} \qquad R = \begin{bmatrix} r_1 & r_4 & r_7 \\ r_2 & r_5 & r_8 \\ r_3 & r_6 & r_9 \end{bmatrix} \qquad T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{6.9.1,6.9.2,6.9.3}$$

Once the initial pose of the camera is known, the subsequent poses can also be estimated using the turntable's accurate rotation. These poses are shown in Figure 6.13. In this figure, it is assumed the world coordinate frame is attached to the turntable's rotating plate.

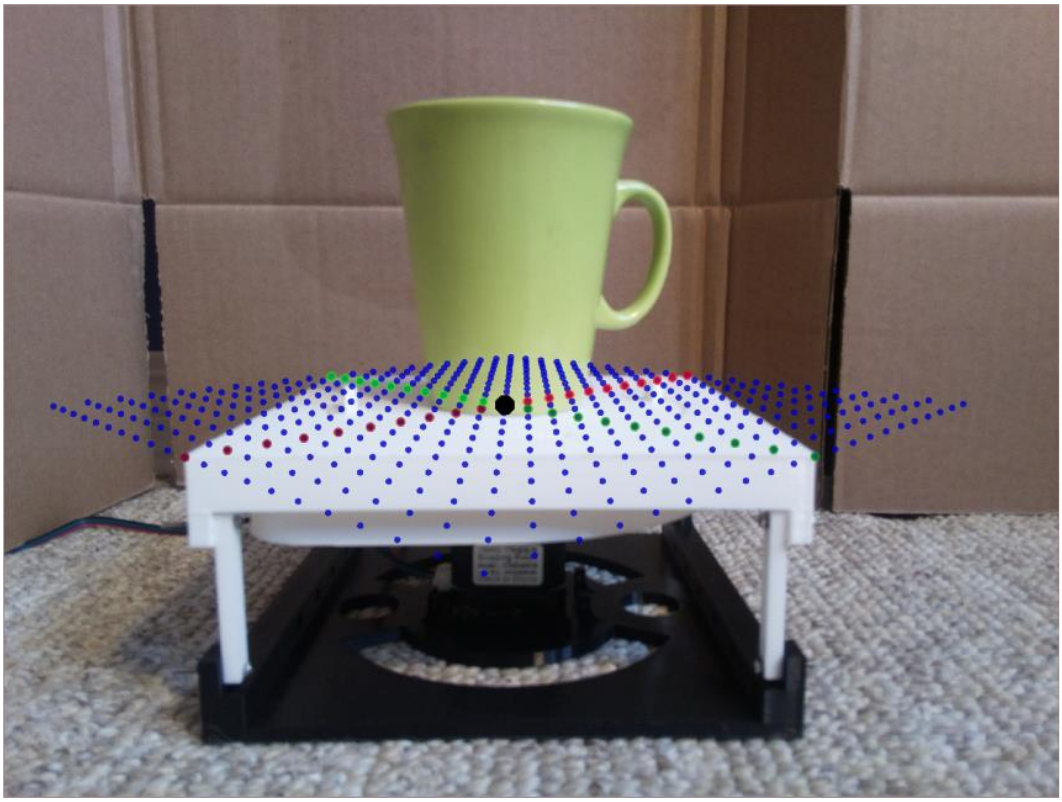


Figure 6.12: The projection of the points in the world frame to the camera's frame

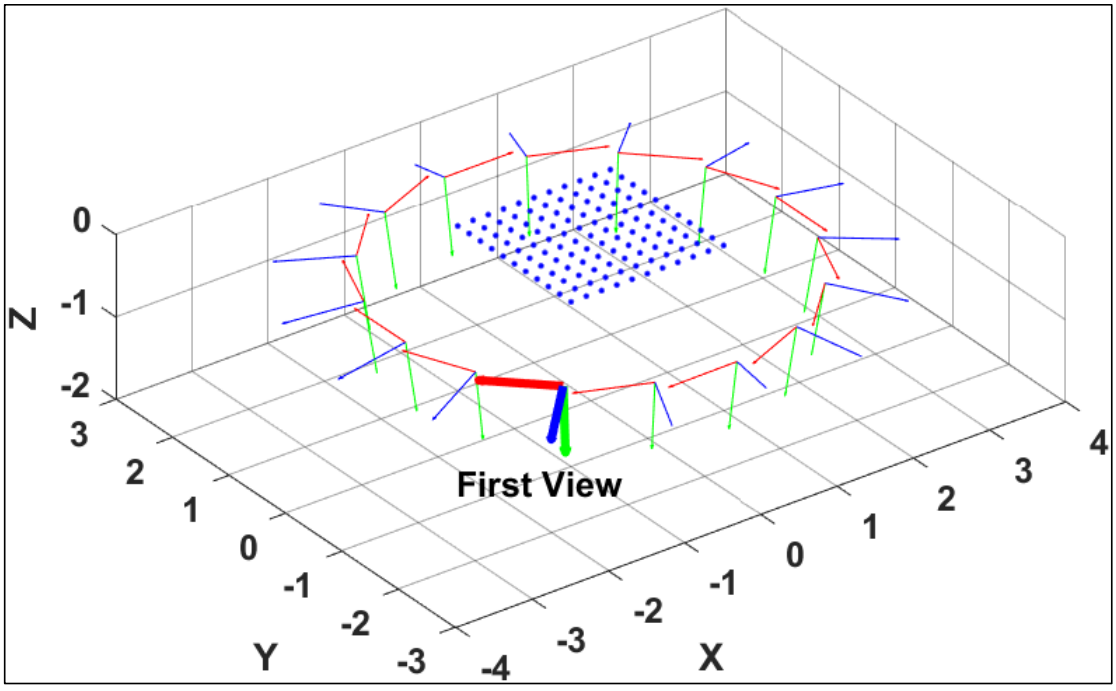


Figure 6.13: The estimated poses of the camera

The explained technique above helps to find a reference pose of the camera in the coordinate frame defined using the plate of the turntable. However, this frame is not the same as the object's coordinate frame (the estimated pose is in the frame of the object). This transformation can be estimated using two approaches. The rigorous approach is to digitally reconstruct the object using the turntable. Three reconstructions are shown in Figure 6.14, where the method in (Asl Sabbaghian Hokmabadi & El-Sheimy, 2022) is utilized. This 3D reconstruction method is based on the silhouette of an object. The silhouette-based reconstruction only requires segmenting the object of interest (foreground) from the background. The segmented area identifies a mask for the reconstruction algorithm. The pixels located inside the mask are back-projected and intersected with a virtual box around the turntable. This virtual box is divided into cells. As the turntable rotates, each cell's occupancy probability is updated. Once all the images are processed, a threshold is utilized to identify cells with higher occupancy values. These cells correspond to the object.

The reconstruction object can be registered to the 3D CAD model of the object. This CAD model was used to build the shape-prior set, and therefore, the estimated object pose is in the corresponding coordinate frame. The registration can follow these steps:

1. Convert 3D CAD model to a point cloud.
2. Find the relative translation and the scale between the CAD model and the reconstruction.
3. Manually initialize the relative orientation.
4. Use a point-to-point Iterative Closest Point (ICP) method to optimize the initial estimates.

Finally, the ground truth pose of the object in the camera frame can be obtained. The second approach is to pay attention to the fact that the object of interest has only a rotation around the z-axis with the turntable. This rotation can be determined by measuring this angle once the object is placed on the turntable.

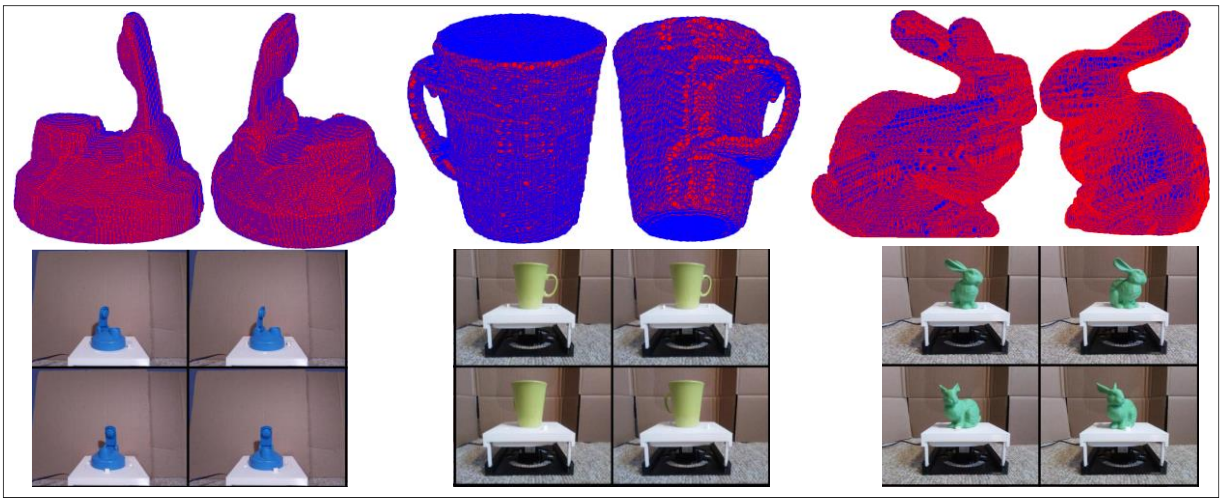


Figure 6.14: The 3D reconstruction of the object of the interest

6.7 Camera Calibration

The estimation of the intrinsic calibration parameters of a camera is an important requirement for the proposed solution. In the tightly coupled fusion, the particle weight update is achieved after the projection of the object onto the image. Such projections require the known intrinsic camera calibration parameters. Further, the developed extrinsic calibration of an IMU and a camera assumes that the intrinsic camera calibration parameters are known.

Some of the calibration parameters of a camera can be represented in the form of a matrix known as the intrinsic calibration matrix (K) (as shown in Equation 6.10). This matrix has five parameters. Two of these parameters (o_x, o_y) are the coordinates of the principal point of the camera (located on the image plane). The principal point is the orthogonal projection of the camera's center (an ideal point where light rays are assumed to intersect) onto the image's plane. Further, the camera calibration matrix includes a focal length (f). This length is typically reported in millimetres (mm). The focal length is the distance of the camera's center from the image plane. The parameters a_x and a_y correspond to the number of pixels per unit (e.g., mm). With these definitions, the parameters of the calibration matrix do not have physical

units, and this definition of the calibration matrix is consistent with the classical literature (Hartley & Zisserman, 2003)

$$K = \begin{bmatrix} a_x f & 0 & o_x \\ 0 & a_y f & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

A calibration matrix defines an ideal pinhole camera. Unfortunately, due to sources of distortion, real cameras are affected by other types of errors. One type of distortion that is common in many cameras is the radial distortion. The radial distortion is shown mathematically in Equations 6.11 and 6.12, where $\Delta r_{dist}^{'}$ is defined in Equation 6.13 ($\Delta x_{dist}^{'}$ and $\Delta y_{dist}^{'}$ are the camera's distance from the principal point in the x and y directions, respectively). The radius (r) is defined in Equation 6.14, where x' and y' are normalized pixel coordinates ($x' = (x - o_x)/a_x f$ and $y' = (y - o_y)/a_y f$). The calibration process estimates the coefficients k_1 and k_2 (higher order terms often are assumed to be negligible). The camera calibration is performed using MATLAB's Camera Calibration toolbox. This software toolbox requires a checkerboard (a planar target) and can estimate the five parameters of the camera calibration matrix and the radial coefficients (among other error parameters). The errors caused by the radial distortions can be mitigated in the images with the help of functions provided in MATLAB's Computer Vision toolbox. Once radial distortion is mitigated in each image, the camera calibration matrix is the only set of intrinsic calibration parameters required in the process of tightly and loosely coupled solutions. The estimated intrinsic calibration parameters for each of the cameras used in the experiments (for the SLAM) are provided in Table 6.4.

$$\Delta y_{dist}^{'} = y' * (\Delta r_{dist}^{'}/r) \tag{6.11}$$

$$\Delta x_{dist}^{'} = x' * (\Delta r_{dist}^{'}/r) \tag{6.12}$$

$$\Delta r_{dist}^{'} = k_1 r^3 + k_2 r^5 \tag{6.13}$$

$$r = \sqrt{((x')^2 + (y')^2)} \tag{6.14}$$

Table 6.4 Summary of the estimated intrinsic camera calibration parameters

Camera	$a_x f, a_y f$ *(pixels)*	o_x, o_y *(pixels)*	*Radial distort. (k1,k2)*	*Image size (pixels)*	*Reproj. error (pixels)*
Raspberry Pi Camera V2	2530,2525	1623,1239	0.1745, 0.3764	3280,2464	0.68
Arducam with 8MP	514,515	332,257	0.1575,-0.1413	640,480	0.18

6.8 Deep Neural Network Training

In this section, some of the hyperparameters for the trained DNN are discussed. This thesis uses two DNNs for the object-segmentation and the floor-segmentation. Such DNNs are trained using U-Net architecture (as explained in Sections 4.2 and 5.2). Other important methods and hyperparameters should be known for the training process. Some of these hyperparameters are mentioned below.

- The larger **image sizes** are often preferred as they can lead to higher precision in object segmentation. However, an increase in the image size will increase the memory requirement for the training. Therefore, the size of the images should often be kept within the limits of the computational memory.
- One of the most important characteristics of a DNN is the larger number of layers than the classical shallow networks. The number of **weight parameters** (also known as trainable parameters) of a DNN depends on the number of layers. However, increasing the number of layers (and thus the number of parameters) will often require a larger number of input images and training time.
- As the number of input data for a DNN is very large (hundreds of thousands up to millions), processing all the data at the same time is often not possible. Thus, the data is provided to a training algorithm in batches. The **batch size** or batch dimension refers to the number of images provided to a training algorithm at each iteration.

- The initial weights assigned to a DNN should be adjusted based on the input data (these weights are initialized either randomly or based on a pre-trained DNN). The algorithm for adjusting these weights is known as the **training method**. In the training phase, the weights are adjusted to minimize the error (the difference between the predicted and the actual output data). In practice, this process is iterative. There are numerous methods introduced in the literature in the past decades for training. As the cost function defined in training is often differentiable, the gradient of this function can be estimated. In order to minimize the cost function, the direction of the gradient that can lead to the largest decrease in the cost function can be selected. This method of minimization is known as the gradient descent. If the batches are selected at random for the training, the resultant method is known as Stochastic Gradient Descent (SGD) (Chollet, 2021).
- The allowed maximum amount of change for the weights in each iteration can be tuned using a **learning rate** hyperparameter. The higher learning rates can increase the speed of convergence, but it is possible that the global minimum is missed. Table 6.5 summarizes some of the hyperparameters for the trained DNN.

Table 6.5. Summary of the hyperparameters used to train DNN

Hyperparameters/Methods	***Details***
Number of trainable parameters	31,000,000
Image input size (type)/ Output format (type)	256, 256 (RGB images)/ 256, 256, (Binary)
Training algorithm	Stochastic Gradient Descent
Batch size	32
Initial Learning Rate	0.01

Chapter 7: Results

7.1 Overview

In this chapter, the main results of the developed object-level SLAM are provided. These include the assessment of the accuracy, runtime, and other aspects of the tightly and loosely coupled approaches. The reference solutions introduced in Section 6 are among the techniques used for the performance evaluations. In addition to evaluating the developed solution to the object-level SLAM, some components, such as object segmentation, pose estimation and refinement, are evaluated separately. These evaluations can provide insight into the performance of individual components of object-level SLAM independent of the others. As an overview, the results include the following:

1. *Precision and recall analysis of the object segmentation using different domain randomization techniques* (Section7.2)
2. *Qualitative and quantitative analysis of coarse pose estimation without refinement* (Section 7.3). The pose estimation method is compared to the reference solution (obtained with the help of a turntable) and to Zhang's method.
3. *Quantitative and qualitative analysis of refined pose estimation* (Section 7.4).
4. *Assessment of the performance of the object-level SLAM using the differential-drive robot (for a single object)* (Section 7.5)*:* The experiment in this section investigates the performance (e.g., the accuracy of the estimated trajectory) of the developed method under different conditions and possible sources of error. Such conditions include the camera's distance from the object (closer or further), illuminations (dimmer or brighter scenes), and occlusions.
5. *Assessment of the object-level SLAM performance using the handheld device (for a single object)* (Section 7.6)*:* In this section, the designed handheld device is used to assess the developed

IMU/monocular camera object-level SLAM. Similar to experiments in the previous section, the method is assessed under different conditions and possible sources of error.

6. *Assessment of the tightly coupled object-level SLAM performance with ultrasonic rangefinder fusion (for a single object)* (Section 7.7). In this section, the ultrasonic rangefinder's observations are used to provide the distance to the objects in the scenes. The results are compared in terms of accuracy and runtime, among other criteria, to the standalone IMU/monocular-based solution.
7. *Assessment of the tightly coupled object level SLAM performance with ultrasonic rangefinder (for multiple objects)* (Section 7.8). Multiple objects can improve the overall accuracy; however, it also can introduce new challenges. Such challenges include the requirement for initialization of the new objects during the navigation. In this section, the performance of the developed method using more than a single object is assessed.

In the experiments of Sections 7.2 to 7.8, it is assumed that the extrinsic calibration parameters of the sensors, such as the IMU and the monocular camera, are known. However, as was explained in Chapter 5, in practical circumstances, such assumptions cannot be made. The performance of the developed boresight calibration technique is assessed in Section 7.9. Finally, the performance of extrinsic (boresight and lever arm) calibration of the ultrasonic rangefinder and camera is assessed in Section 7.10.

7.2 Precision and Recall of the Object Segmentation

In this section, the performance of the object segmentation method is evaluated. Object segmentation is a crucial component in the developed solution to object-level-based SLAM. Object segmentation is used in estimating the object's pose in the initialization phase and the particle weighting process of the tightly coupled SLAM. The developed object segmentation is trained using a synthesized training image set (as explained in Section 4.2). In order to assess performance, precision and recall are utilized. These assessment criteria are measured at the pixel level by comparing the reference solution and

the output of the trained DNN. The reference segmentation of the object of interest is obtained manually using the Grab-cut algorithm. The experiments are performed in many indoor environments (some of the images from these indoor environments are shown in Figure 7.1). For these experiments, a Canon PowerShot s110 is used.

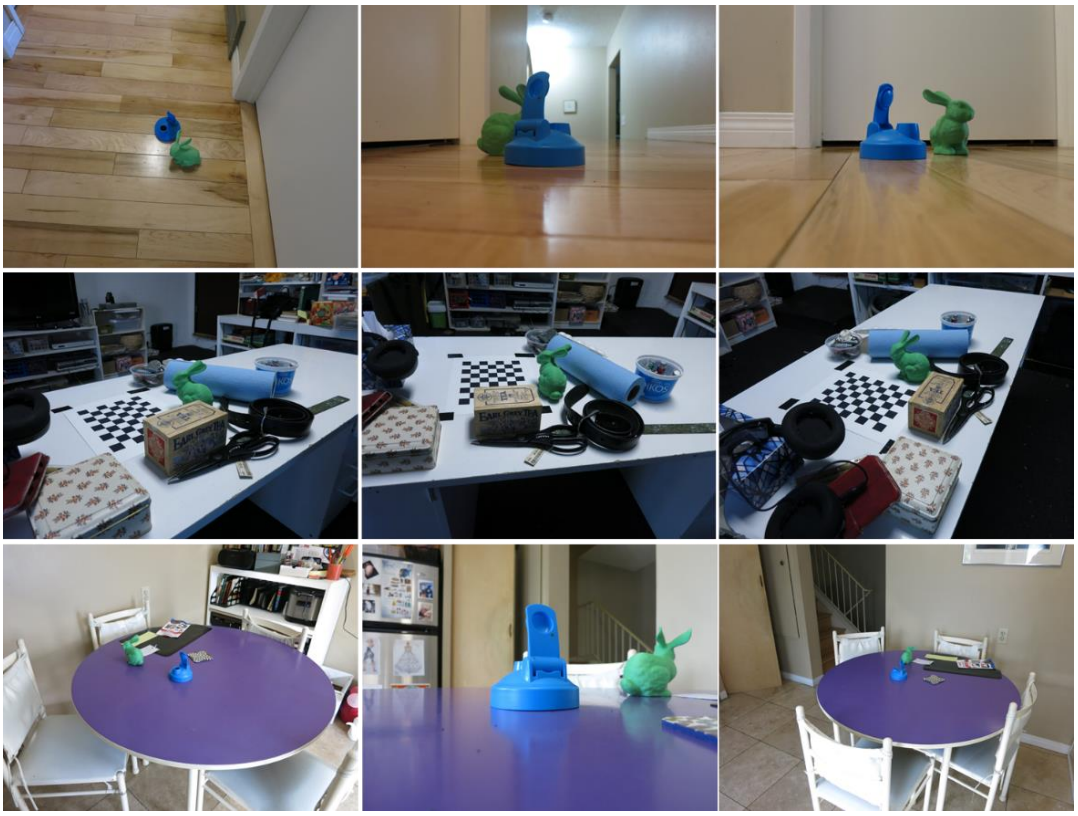


Figure 7.1: Illustration of some of the environments for the experiments.

The developed DNNs are trained using synthesized images. Synthesizing images is achieved based on the concept of domain randomization (explained in Section 4.2). In order to investigate the effectiveness of different domain randomization factors (e.g., illumination condition, camera calibration parameters), ten DNNs are trained. In training DNNs, progressively, more training images are added. At each new set, one or more aspects are considered in domain randomization. The results are summarized in Table 7.1. Since the output images using U-Net include more than two values, a process of binarization

is required to separate the two segments (background and the foreground(object)). This threshold is the only hyperparameter that can be required for tuning in the experiments. High recall and high precision for the developed pose estimation, in general, are required. The results in Table 7.1 report the precision for a recall of approximately 85%. The networks DNN 1 and DNN 2 are trained with image resolutions of 256×256 and 128×128 pixels, respectively, and approximately 27k images (k here denotes one thousand). The precision values for these two cases indicate that better results are achieved for the lower resolution. The lower precision of the set trained with higher image resolution (DNN 1) might be due to the higher ratio of training parameters to the number of input images, which leads to a lower capacity to generalize from training to test sets (overfitting occurs).

In the second set of experiments (DNN 3 and DNN 4), the image database size was increased to 49k by adding extreme viewpoints (cameras that are too close or too far from the object). It can be seen that adding such extreme viewpoints, overall, does not increase performance. Conversely, in certain scenes, precision is reduced (realistic viewpoints are added in DNN 9 and DNN 10). In the third set of experiments (DNN 5 and DNN 6), the image database size has increased to 59k by including new backgrounds. These backgrounds, for example, have a single hue or are from an outdoor environment and are different from the test environments. As with the previous test, overall, no significant gain in precision is obtained. In the fourth set of experiments (DNN 7 and DNN 8), the size of the image database has increased to 120k by including different illumination conditions (including extreme conditions). The precision of this experiment has improved overall. Finally, realistic viewpoints and illumination conditions are included in training DNN 9 and DNN 10. The obtained results indicate that considering domain randomization factors corresponding closer to the real-world scenarios plays a significant role in increasing the precision. For the four best DNNs, a complete precision versus recall plot is provided in Figure 7.1 (the binarization threshold is varied in a range from 0.01 to 0.99).

Table 7.1: The precision of DNNs using different numbers of training images, image resolution and more.

Trained DNN	Experiment Information									
	Input	***# Image***	***Test 1 (p)***	***Test 2 (p)***	***Test 3 (p)***	***Test4 (p)***	***Test5 (p)***	***Test6 (p)***	***Test 7 (p)***	***description***
DNN 1	256x256	27k	0.19	0.07	0.18	0.13	0.04	0.20	0.33	original set
DNN 2	128x128	27k	**0.65**	0.02	0.75	**0.77**	0.03	**0.81**	0.92	original set
DNN 3	256x256	49k	0.60	0.48	0.69	0.24	**0.09**	0.32	0.51	+extreme viewpoints variations
DNN 4	128x128	49k	0.39	0.17	0.58	0.19	0.04	0.24	0.93	+extreme viewpoints variations
DNN 5	256x256	59k	0.13	0.22	0.14	0.12	0.05	0.06	0.28	+unrealistic backgrounds
DNN 6	128x128	59k	0.30	**0.63**	0.18	0.16	**0.10**	0.18	0.54	+unrealistic backgrounds
DNN 7	256x256	120k	0.44	0.36	0.74	0.03	0.02	0.04	**0.95**	+extreme illumination conditions
DNN 8	128x128	120k	**0.65**	0.08	**0.88**	0.54	0.02	0.77	0.91	+extreme illumination conditions
DNN 9	256x256	180k	**0.95**	**0.81**	**0.94**	**0.93**	**0.39**	**0.99**	**0.99**	+realistic viewpoints +realistic illumination
DNN 10	128x128	180k	**0.90**	**0.60**	**0.89**	**0.64**	0.06	**0.94**	**0.94**	+realistic viewpoints +realistic illumination

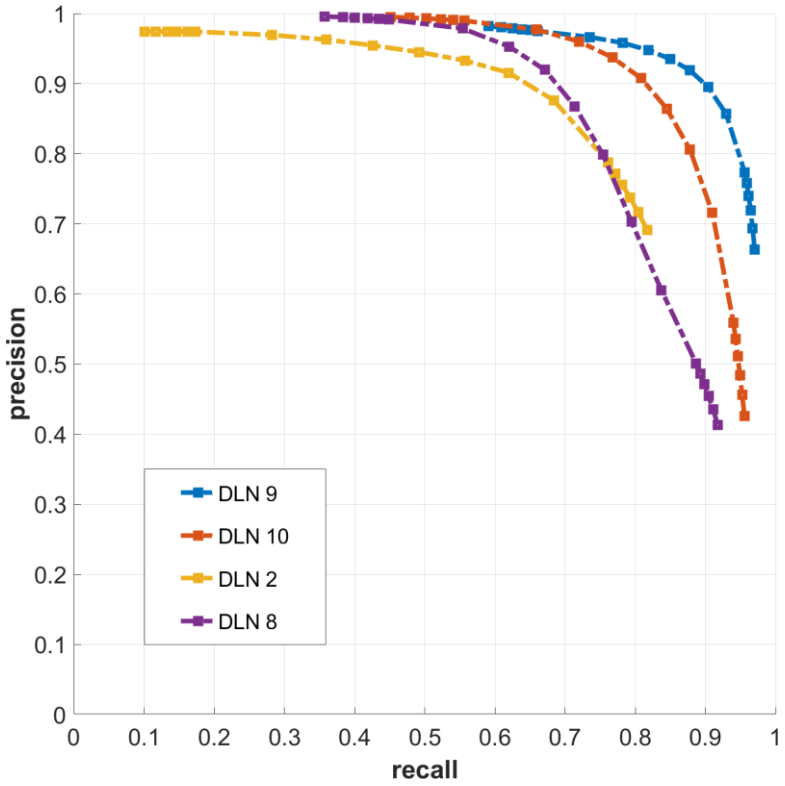


Figure 7.2: The precision versus recall plot of four DNNs with the best performance.

7.3 Qualitative and Quantitative Analysis of Coarse Pose Estimation

In this section, the accuracy of the coarse pose estimation is evaluated. The developed pose estimation (explained in Chapter 4) is object-centric. Therefore, the estimation is only accurate if the camera is looking at the object of interest (the center of the camera, the principal point and the center of the 3D model of the object are aligned). As the results in this section will indicate, the pose estimation stays relatively accurate, even when the camera is not looking at the object.

For the purpose of quantitative analysis, the reference solution (for the relative pose of the object) is obtained using a turntable (Chapter 6). The results are summarized and shown in Table 7.2. The accuracy of orientation parameters is estimated by decomposing the rotation matrix into three Euler angles. These decomposed angles are used to measure the Euclidean distance from the reference to the estimated angles. Since the center of the CAD model (where the object's pose is estimated with respect to) and the coordinate center of the turntable (where reference rotation and translation are measured) are often not known, the error in relative translation is reported. This relative translation is measured as the vector from a designated camera pose to other camera poses. The designated viewpoint can be the first image (or any other image in the set). Relative translation cancels out the unknown offset between the two centers (of the turntable and the object). The last assessment criterion is the ratio of the relative translation error to the magnitude of the relative translation vector, as reported in Table 7.2.

The experiments are performed with two camera-to-object distances (short-range (~50 cm) and medium-range (~100 cm)). As can be seen from the results, the orientation error of approximately 9.96° to 10.31° has been achieved for these tests. One possible reason for the better performance in the experiment with a shorter range is the higher precision and recall of the object segmentation. The boundary of the segmented object of interest is used in finding the best-matched shapes in the shape-prior set (as explained in Chapter 4).

Table 7.2: The orientation and translation errors of the developed pose estimation.

Exp.	Experiment Details			
	Orientation error (degrees)	***Translation error (cm)***	***Translation error ratio (%)***	***Description***
Test 1	9.96	3.77	0.09	short-range
Test 2	10.31	2.13	0.12	medium-range

Figure 7.3 shows the orientation error for each image. The symbol # indicates the index of the image (a total of 33 images used for the test). As can be seen, the overall mean error is affected by one or two images with large errors. The median rotation error is 7.56° .

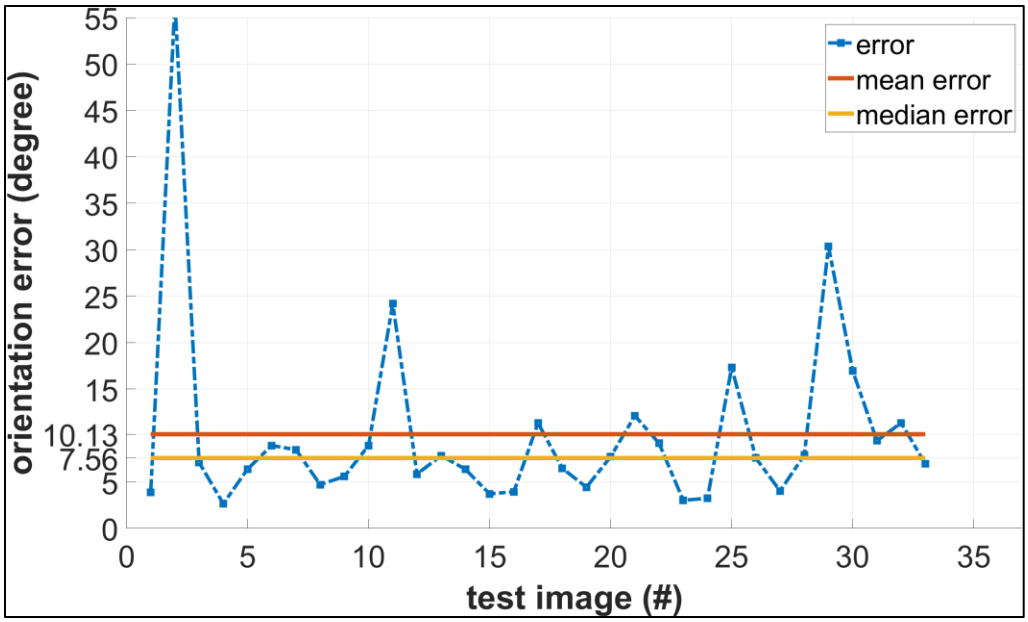


Figure 7.3: Orientation error for each image

The second quantitative analysis is conducted to compare the developed pose estimation to Zhang's method (Zhang, 2000). Zhang's method is a planar-based camera calibration method that also estimates the relative pose of the camera with respect to the checkboard. The advantage of Zhang's method is that it can be used in different indoor scenes as long as the target (a checkerboard) is present in the image. The disadvantage of this method is that it does not produce poses as accurately as using the

turntable. It will be shown the developed pose estimation outperforms Zhang's method. Due to an unknown translation vector between the coordinate origin of the checkerboard and the CAD model, only the orientation error is reported. In order to measure this error, the object is aligned with the checkboard's axes in the scene. The accuracy is measured in four scenes, and multiple images are included for each scene. Based on the results in Table 7.3, the orientation error becomes larger as the distance of the camera from the object increases. This can be due to the fact that Zhang's method depends on corner detection, and as the camera becomes closer to the checkerboard, the accuracy of such detection increases. In order to compare the results qualitatively, the object's 3D model and its coordinate frame are projected onto the image with the help of known pose parameters in Figure 7.4. This figure illustrates that the developed method can estimate a more accurate pose of the object compared to Zhang's method.

Further qualitative analysis of the developed method is provided in Figure 7.5. It can be seen that the estimated pose is very close to the expected pose in different indoor scenes . The images show variations in the camera's viewpoint, illumination conditions, as well as different levels of background clutter. In some images, the object is occluded to demonstrate the partial robustness of the pose estimation to this source of error. It can be seen that the developed method can still estimate a relatively accurate pose under mild to moderate occlusions.

Table 7.3: Orientation error between the developed and Zhang's pose estimation.

Exp.	**Experiment Details**		
	Orientation error (degrees)	*Range*	*Background clutter*
Test 3	12.79	Very Short	Low
Test 4	13.27	Short	Low
Test 5	15.38	Short	High
Test 6	19.43	Medium	High

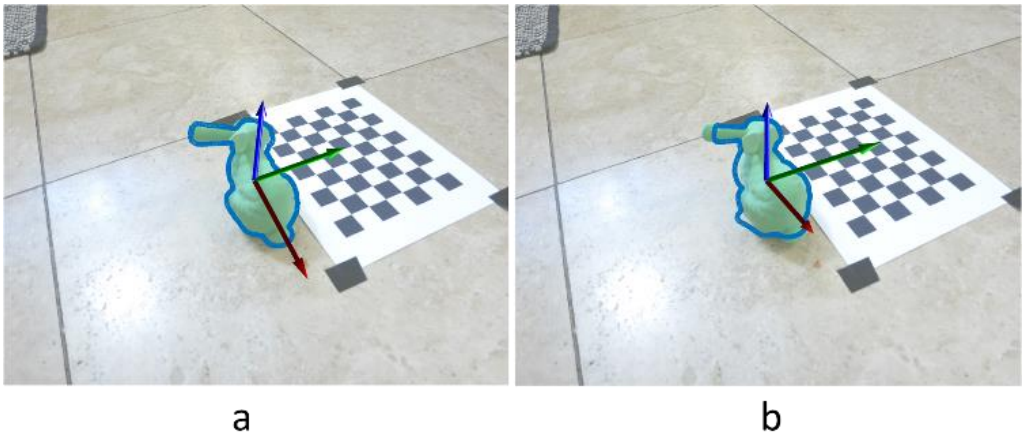


Figure 7.4: Qualitative comparison of the developed and Zhang's pose estimations. The developed method is shown in (a), and Zhang's method is shown in (b).

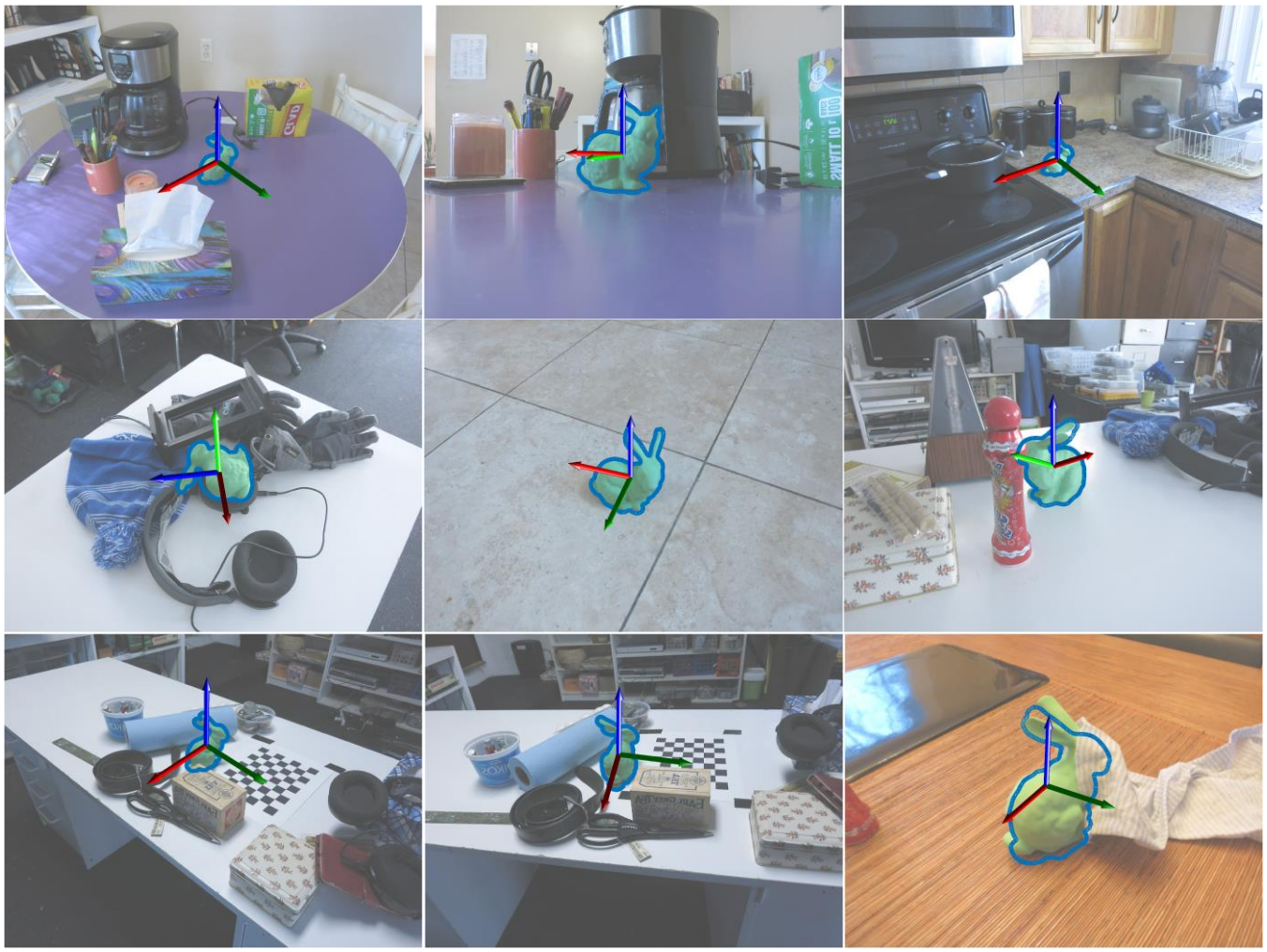


Figure 7.5: Qualitative assessment of the developed pose estimation technique.

7.4 Quantitative and Qualitative Analysis of Refined Pose Estimation

The procedure explained in the previous section can only provide an accurate pose estimation if the camera is directly looking at the object. In the cases that this cannot be held true, the coarse estimation of the pose should be refined. The developed loosely coupled object-level SLAM solution is directly affected by the independent pose estimation using the camera; therefore, pose refinement can significantly impact the accuracy of this approach.

In order to evaluate the pose refinement, two assessment metrics are used. The first assessment metric is the reprojection error, and it is defined as the distance between the projected points of the object's 3D model onto the image and the observed points on the boundary of the segmented object. In order to measure this error, one-to-one correspondence between points on the projected and the observed boundaries should be established. As explained in Section 4.6, such feature correspondence is already established in the refinement process. Therefore, the reprojection error can be measured readily. The second assessment metric is the IoU of the projected and observed boundaries. Unlike the reprojection error, the IoU considers every point on the boundary to assess the accuracy of the pose refinement. Further, IoU is independent of point-to-point feature correspondence. The two methods are illustrated in Figure 7.6, which shows that even though the predicted (yellow) and observed boundaries (green) are very similar, the reprojection error is high (6.98 pixels on average). Similarly, the IoU is relatively low (0.579 with a maximum possible value of 1 and minimum 0). Figure 7.7 shows the initial boundary, the observed boundary, and the boundary obtained after refining the pose in red, yellow, and purple, respectively. As mentioned, the initial pose estimation assumes that the camera is directly looking at the object. It is important to note that the refined pose is not only an in-plane translation of the coarse (initial) pose estimation. This can be seen in the image in the third row and the second column in Figure 7.7, where the object's 3D orientation is also improved significantly.

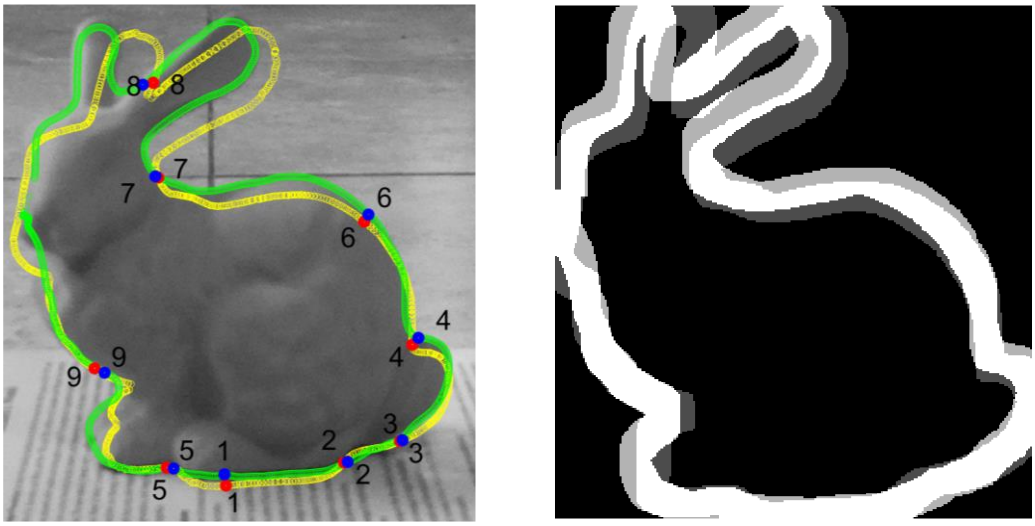


Figure 7.6: Comparison of reprojection and IoU values.

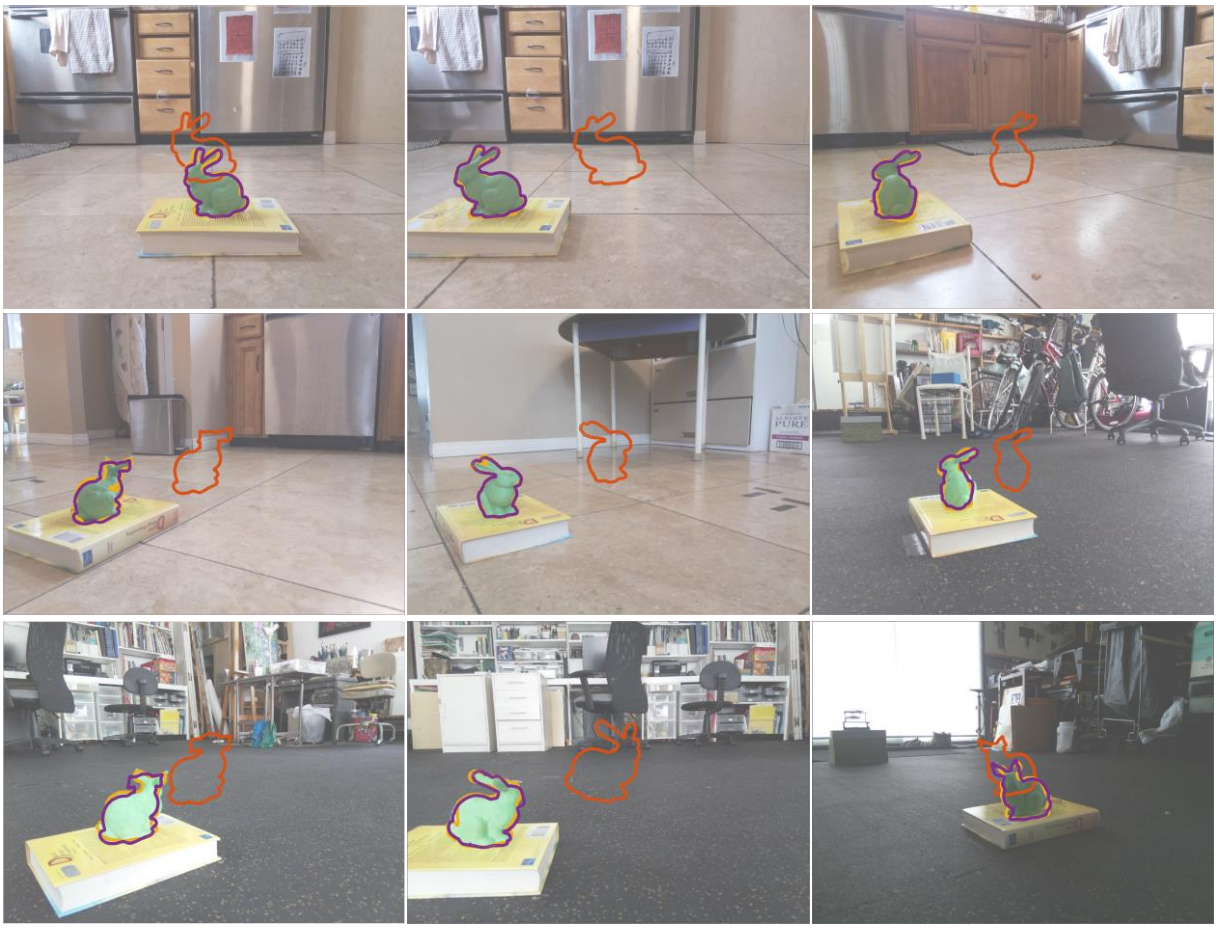


Figure 7.7: Illustration of coarse and refined poses

For quantitative analysis, the experiments are performed in four indoor environments. In these experiments, the distance to the object and illumination conditions are varied. Table 7.4 summarizes the reprojection error. The object's distance to the camera did not affect the values in the reprojection error significantly. Figure 7.8 shows the reprojection error for each individual image in each set. It is important to note these reprojection values are estimated with the developed method of feature correspondence. There are at least two reasons for large reprojection errors:

- The feature correspondence can be inaccurate. This is because the correspondences (as explained in Section 4.5) are established by finding the nearest features, which depend on the coarse pose estimation.
- The segmentation does not accurately correspond to the boundary of the object. This can be seen in Figure 7.6 (the green boundary in the left image). The resolution of images matters significantly in the results of segmentation. The results from the previous section were obtained using a Canon PowerShot S110, and the result in this section is obtained using RPV2, which produces lower-quality images.

Table 7.4: Reprojection error of the developed pose refinement.

Exp.	**Experiments details**			
	Mean Error (pixels)	*Median Error (pixels)*	*Range*	*More information*
Test 7	7.99	7.96	short	low background clutter, bright light
Test 8	8.01	7.69	long	low background clutter, dim light
Test 9	8.61	8.37	long	low background clutter, bright lights
Test 10	8.80	8.22	long	high background clutter, dim light

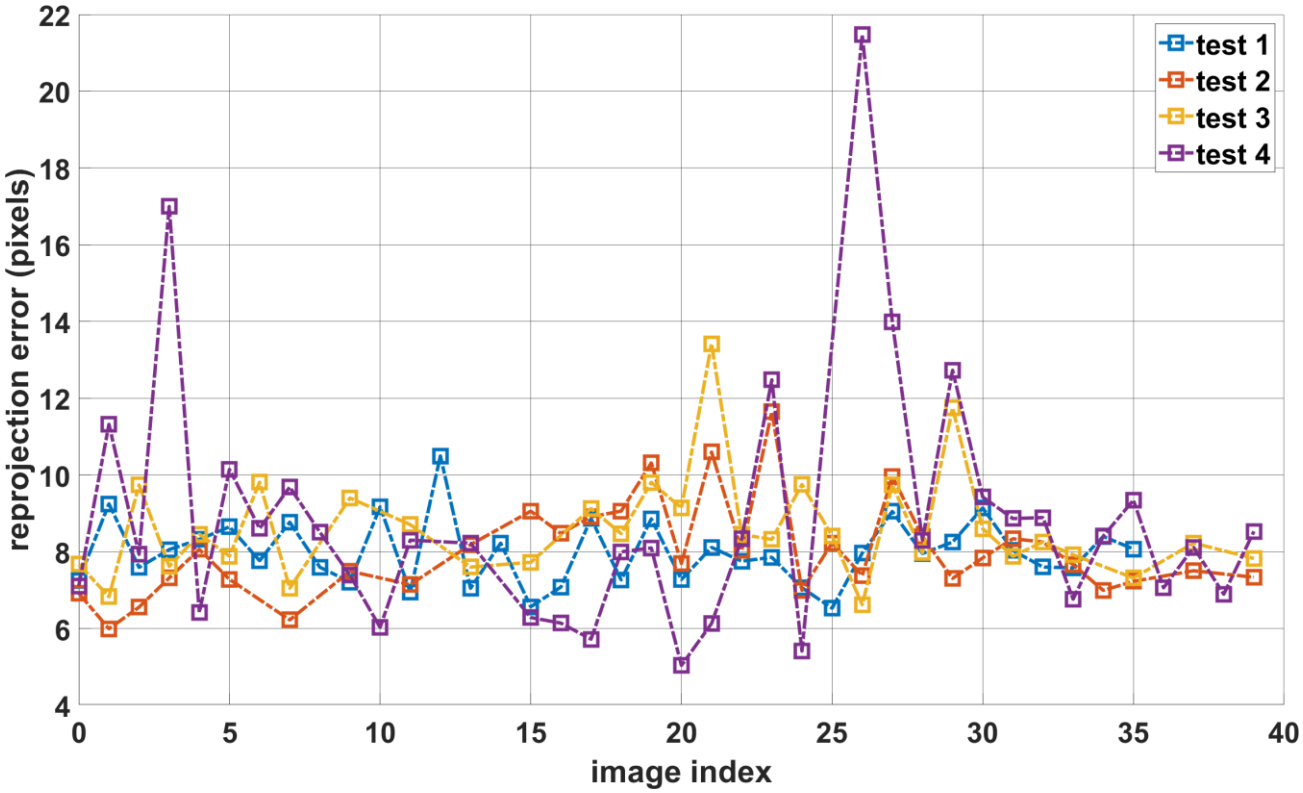


Figure 7.8: Illustration of reprojection error for each image in four tests.

The IoU values are provided in Table 7.5. In these experiments and all the subsequent experiments in this chapter, the reported IoU is the average of all the tested images in that experiment. Based on this table, it seems that the background clutter has a significant impact on the accuracy. The lower background clutter in Tests 7, 8, and 9 results in a better performance than Test 10, which has higher background clutter. In Table 7.6, the corrections to the orientation after the pose refinement are shown. The correction in the orientation is measured as before, where the rotation matrix is decomposed into three Euler angles, and the distance between the angles before and after applying pose refinement is measured. Since the pose estimation is obtained using a single image, no scale is available. Therefore, translation correction is not reported. From Table 7.6, it can be seen that a significant correction in the orientation is obtained after the refinement. The result in this section indicates that the pose estimation has errors. In the following sections, it will be shown that through the process of RBPF, better camera poses can be achieved.

Table 7.5: IoU after pose refinement.

Exp.	Experiments details			
	Mean Error	*Median Error*	*Range*	*More information*
Test 7	0.56	0.57	short	low-background clutter, bright light
Test 8	0.68	0.67	long	low background clutter, dim light
Test 9	0.65	0.67	long	low background clutter, bright lights
Test 10	0.43	0.44	long	high background clutter, dim light

Table 7.6: Orientation correction after pose refinement.

Exp.	Experiments details		
	Orientation correction (degrees)	*Range*	*More information*
Test 7	28.55	short	low background clutter, bright light
Test 8	41.09	long	low background clutter, dim light
Test 9	42.98	long	low background clutter, bright lights
Test 10	27.39	long	high background clutter, dim light

7.5 Assessment of the Object-level Solution Using the Differential-drive Robot (single object)

In this section, the accuracy of the estimated trajectory using a differential-drive robot is investigated. Similar to the previous sections, the effects of illumination conditions, camera distance from the object, and background clutter on the accuracy of the solution are studied. The error in the position is estimated using two different approaches. In the first approach, the reported error is the distance between the ground truth and the estimated final positions. In the second approach, beacons with an accuracy of approximately 3 to 11 cm are utilized to measure the error in the estimated position at different points along the trajectory of the robot. Both these methods are explained in Chapter 6. The images for these experiments are captured in a stop-and-go fashion. The image segmentation is applied to all the images.

One of the key components of any RBPF-SLAM is the weighting and resampling processes. The weighting step in an RBPF-SLAM can fail in the tightly and loosely coupled methods due to challenging scenarios. These scenarios are investigated in Section 3.7. Here, a summary and further details are provided:

a) The object of interest is present in the image but only partially segmented. Possible reasons for this challenge to occur are occlusions of the object of interest by other objects, the long distance of the object from the camera, and more. (**Challenge II**, Section 3.7).

b) The object of interest is present in the image, but object segmentation has low precision. This challenge can occur if other objects with similar appearance or shape in the background are segmented as well (**Challenge III**, Section 3.7)

In the loosely coupled solution, the particle weighting is based on an independent pose estimation using the current image and the motion model (i.e., using the robot's wheel odometry). However, in the tightly coupled method, particle weighting is based on the projected and observed object boundary (pose estimation from one image is only used in object initialization). Therefore, camera poses are refined indirectly in the resampling process and pose estimation from one image is not required (as explained in Section 3.4). This makes the tightly coupled approach not reliant on coarse-to-fine pose estimation. As shown in Section 7.2, the coarse estimation of the pose has only low to medium robustness to errors in segmentation; therefore, it is expected for the loosely coupled approach to be more affected by Challenges II and III. However, the tightly coupled method is also affected by both challenges. For example, when Challenge II occurs (e.g., due to occlusions), the maximum possible particle weight (when prediction and observation coincide perfectly) is limited to the visible area of the object. Such scenarios can result in the particles with more and less accurate poses receiving similar (and low) weights (as explained in Section 3.7).

To address possible degradation in the accuracy due to these challenges, a mechanism for failure detection should be utilized. In order to identify if Challenge II has occurred, the newly estimated weights can be investigated. If these weights are all low, then this can indicate Challenge II has occurred. In such cases, the observation from the current epoch can be disregarded to avoid degradation of the accuracy.

Challenge III can occur when some background pixels are labelled as objects. However, in the experiments, this issue rarely occurs. If the background pixels are far from the object of interest, the accuracy of neither loosely nor tightly coupled methods will be reduced with the help of one additional step after segmentation. This step is the identification of connected pixels in the segmented binary image (which can be achieved using classical computer vision techniques). Once these connected groups are identified, the one group that has the largest number of pixels can be selected. These pixels most likely correspond to the object of interest. The remaining pixels can be labelled as background. In addition to the fault detection mechanisms, both tightly and loosely coupled methods propagate the weight of the particle to the next epochs, and if Challenge II or Challenge III occurs in only several images, the solution can recover from these problems.

Three experiments are conducted to evaluate the tightly and loosely coupled methods. Table 7.7 summarizes the error in centimetres, and Table 7.8 shows the error relative to the total trajectory length. In order to estimate the best position using all the particles, two approaches are used. The first approach reports the weighted average of the particle poses (denoted as Expected Value (EV)), and the second approach reports the pose of the particle with maximum weight (denoted as Maximum a posteriori (MAP)). The number of particles utilized for these experiments for tightly coupled is 10,000, and for loosely coupled, it is 50,000 particles. Further, each experiment is performed five times, and the average of 5 experiments is reported as the final accuracy. However, no significant difference in the accuracy among these repeated experiments was observed. The short-range corresponds to approximately 35 cm,

and the longer range corresponds to 75 cm. In these tests, the loosely coupled solution produces a more accurate final position overall. One reason for the better performance of this method is the estimation of the pose of the object with respect to the camera in each new image, independent of the current particle proposal. This can provide a certain degree of robustness to the accumulated errors during the navigation and errors in the initializations. However, it is important to note that tightly coupled have performed better along the trajectory (this will be demonstrated clearly in the following).

Table 7.7: The error in the position of the tightly and loosely coupled solutions

Exp.	**Experiments details**					
	Loosely coupled		*Tightly coupled*		*Range*	*More information*
	EV (cm)	*MAP (cm)*	*EV (cm)*	*MAP (cm)*		
Test 11	23.2	19.6	12.7	27.7	short	low background clutter, bright light
Test 12	10.1	89.5	25.0	46.9	long	low background clutter, dim light
Test 13	20.5	92.4	46.2	29.4	long	low background clutter, bright lights

Table 7.8: The error ratio in the position to the path length of the tightly and loosely coupled solutions.

Exp.	**Experiments details**					
	Loosely coupled		*Tightly coupled*		*Range*	*More information*
	EV/DT	*MAP/DT*	*EV/DT*	*MAP/DT*		
Test 11	0.06	0.05	0.03	0.07	short	low background clutter, bright light
Test 12	0.01	0.12	0.03	0.06	long	low background clutter, dim light
Test 13	0.03	0.13	0.06	0.04	long	low background clutter, bright lights

The results for the IoU are provided in Table 7.9. This table reflects the average IoU measured for the best particle across all the epochs, but failure cases (which can result in very small IoU) are not considered. The failure percentage is reported in Table 7.10, and it is very small for the tightly coupled method in comparison to the loosely coupled method. This is since, as mentioned earlier, the tightly coupled method is less influenced by Challenge II and III. Tables 7.9 and 7.10 show that the IoU is reduced (and the failure rate increased) for the loosely coupled method for longer ranges. These experiments show that even with some images failing to provide an update (particle weighing and resampling), both methods can produce an overall error in the trajectory between 0.01 to 0.06 of the total path (see the EV/DT in Table 7.8).

Table 7.9: IoU of the back-projected 3D CAD model and the segmented object.

Exp.	**Experiments details**			
	Loosely coupled	*Tightly coupled*	*Range*	*More information*
Test 11	0.73	0.86	short	low background clutter, bright light
Test 12	0.69	0.89	long	low background clutter, dim light
Test 13	0.61	0.84	long	low background clutter, bright lights

Table 7.10: failure-rate (the smaller, the better)

Exp.	**Experiments details**			
	Loosely coupled	*Tightly coupled*	*Range*	*More information*
Test 11	0.20	0.12	short	low background clutter, bright light
Test 12	0.34	0.00	long	low background clutter, dim light
Test 13	0.41	0.19	long	low background clutter, bright lights

More experiments using different numbers of particles are provided in Table 7.11. In this table, it can be seen that the failure rate for the loosely coupled is overall higher. This rate for the loosely coupled decreases overall from 0.48 to 0.17 as the number of particles increases from 1000 to 20000. However, it seems that a very small improvement has been gained beyond 12500 particles. Conversely, once the number of particles is above 5000, no failure has been reported for the tightly coupled.

The accuracy and the IoU of tightly coupled remain similar after 5000 particles, exhibiting that increasing the number of particles beyond this point will not impact the accuracy. However, the runtime significantly increased for the tightly coupled as the number of particles increased. This increase in runtime is expected as tightly coupled requires the projection of the particles onto the image and finding the boundary of the projected points. Finding the boundary is the computational bottleneck of the process. In practice, several steps can be taken to reduce the runtime. These steps are explained previously in Section 4 but will be briefly repeated here. One approach to reducing the computational cost is initially to project only the center of the 3D CAD model. If this center falls away (more than a certain pixel threshold) from the observed contour, the particles will be assigned to a low weight. In our experiments, a large percentage of particles are assigned with a low weight using simple steps. But despite using such additional steps, the runtime can still remain high.

For the loosely coupled, the overall reported IoUs are less than the tightly coupled method. However, the runtime is shorter since updating the weights of the particles in the loosely coupled does not require projection of the 3D model (and finding its boundary). Instead, the weights are updated using the predicted pose (obtained from the wheel odometry) and the observed pose (obtained from the monocular camera) (as explained in Chapter 3). The estimated error in these experiments does not assess the performance along the trajectory. In order to compare the trajectories of the two methods, more details are provided in the following.

Table 7.11: This table illustrates the mean, IoU elapsed time for the experiments for different numbers of particles.

Particle number	**Experiments details**							
	Loosely coupled				*Tightly coupled*			
	EV (cm)	*IoU no units*	*Time (s)*	*Fail rate no units*	*EV (cm)*	*IoU no units*	*Time (s)*	*Fail rate no units*
1000	20.0	0.697	34.5	0.48	-	-	-	1.00
2500	9.7	0.760	36.2	0.23	-	0.527	76.8	0.92
5000	22.6	0.730	42.8	0.26	12.1	0.879	413.7	0.00
7500	32.3	0.684	49.5	0.26	12.3	0.895	623.4	0.00
10000	29.3	0.674	61.4	0.31	11.9	0.892	827.5	0.00
12500	14.0	0.715	70.1	0.20	12.1	0.899	1019.0	0.00
15000	11.25	0.712	77.9	0.14	12.0	0.906	1246.5	0.00
17250	7.1	0.819	88.5	0.17	12.3	0.904	1441.6	0.00
20000	15.2	0.767	100.7	0.17	12.3	0.903	1595.5	0.00

The error ellipsoids, the particles, and the trajectory estimated for the tightly and loosely coupled method (Test 12) are shown in Figures 7.9 and 7.10, respectively. The ground truth final position (GT) and the estimated final position (End) is also shown in these figures. The estimated trajectory is in 6DoF; however, only the top view is shown for clarity. In order to illustrate the errors, ellipsoids are fitted to the particles that contain 50% of the overall weight (the corresponding particles are shown in green points). The actual trajectory of the robot is approximately a circle around the object. These figures show that the tightly coupled method produces a more accurate trajectory. Also, from Figure 7.9 (tightly coupled) failure rate is 0, while Figure 7.10 (loosely coupled) exhibits a failure to update in one epoch (where the uncertainty grows large, the algorithm failed to update).

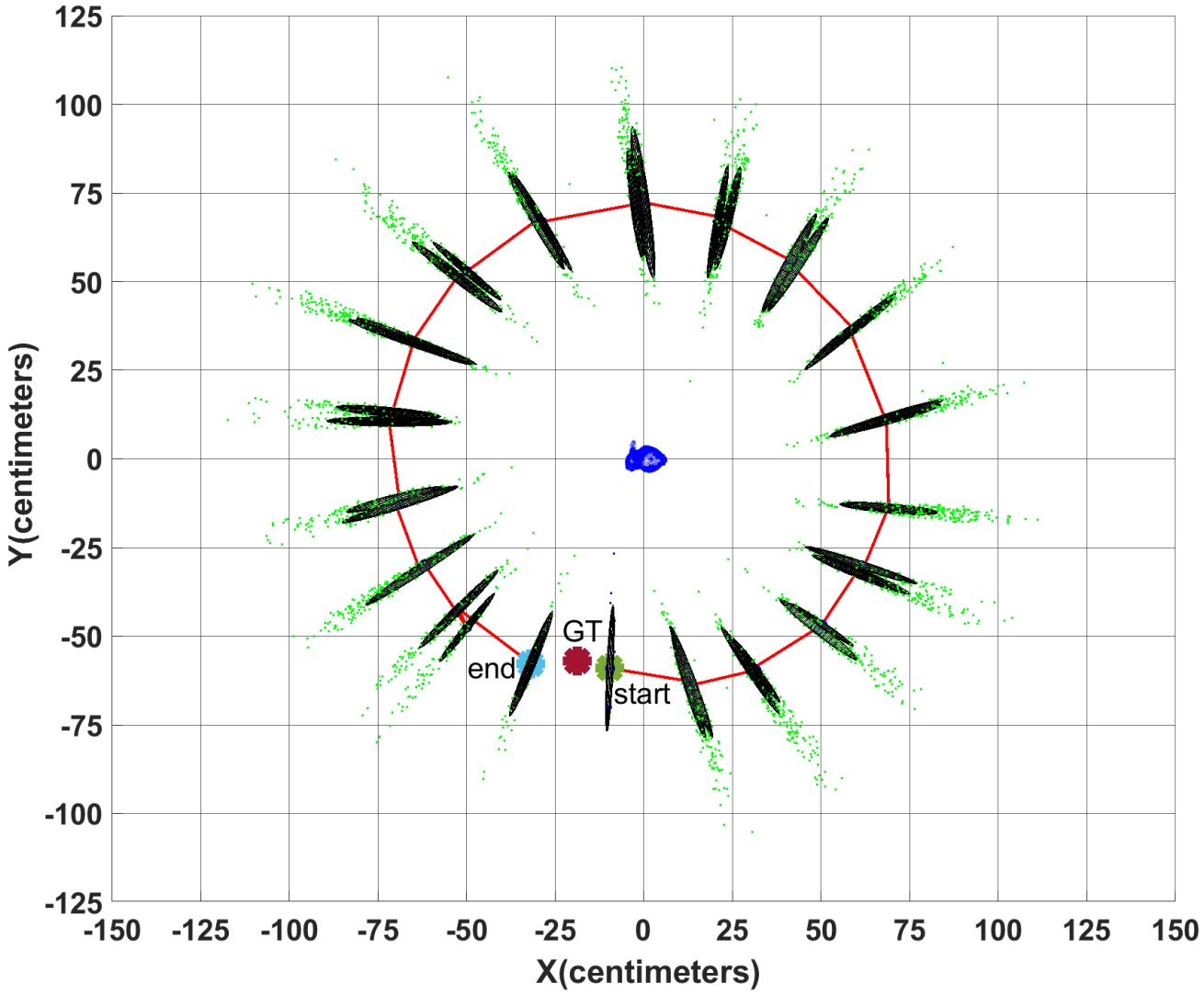


Figure 7.9: Particles and uncertainty ellipsoids of the tightly coupled solution.

Figures 7.9 and 7.10 show that the uncertainty in the direction from the object's center to the camera's coordinate center is larger. This larger uncertainty is expected due to the scale ambiguity with monocular cameras (see Section 3.7, **Challenge IV**). Further, It is important to note that the estimated distribution of the particles is not ellipsoidal as well. The exact shape of the error is different based on the pose of the camera with respect to the object. For illustration, a schematic of possible uncertainty is shown in Figure 7.10. The non-ellipsoidal shape of the error makes the PF-based approach more suitable than methods such as EKF for this problem.

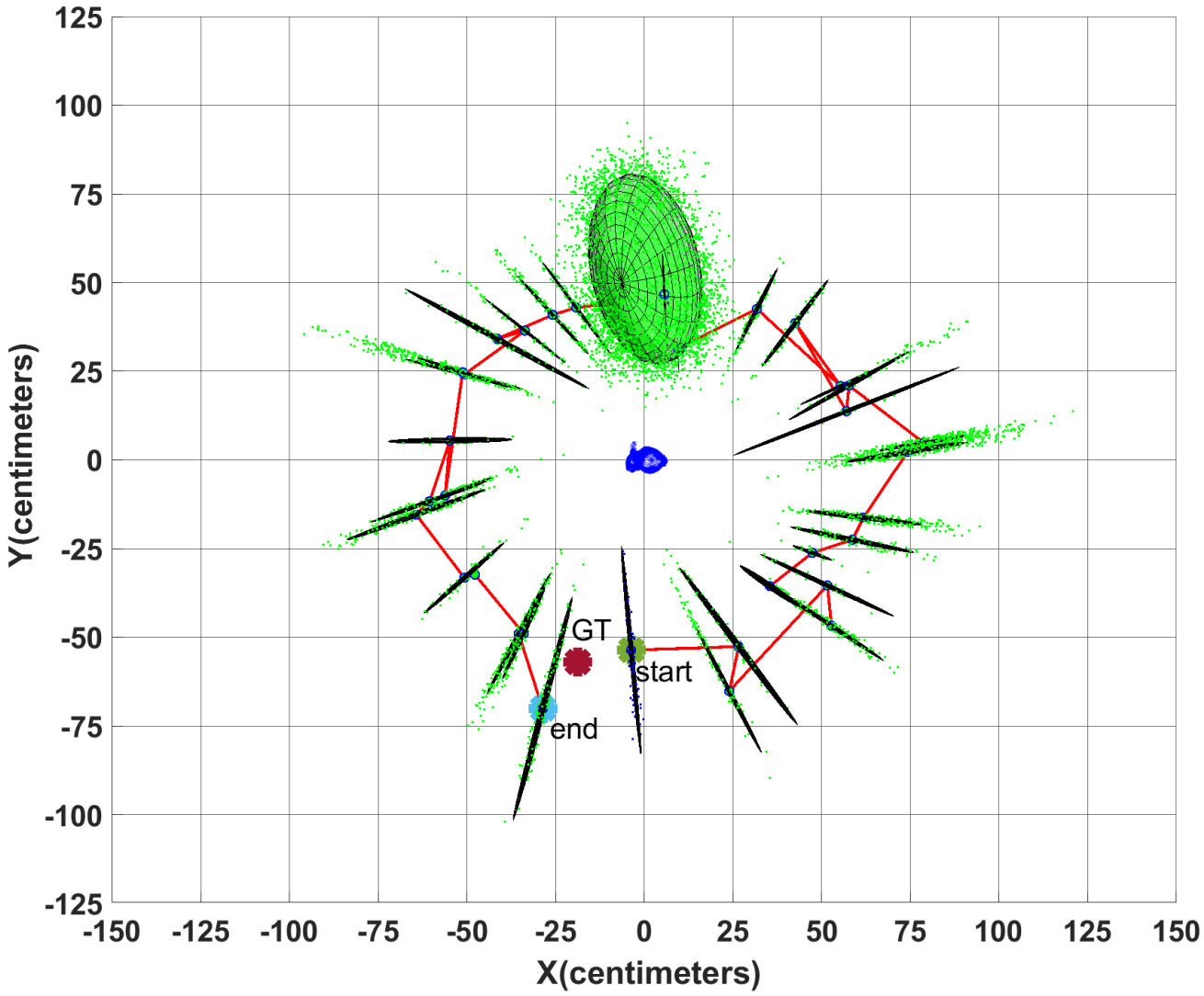


Figure 7.10: Particles and uncertainty ellipsoids of the loosely coupled solution.

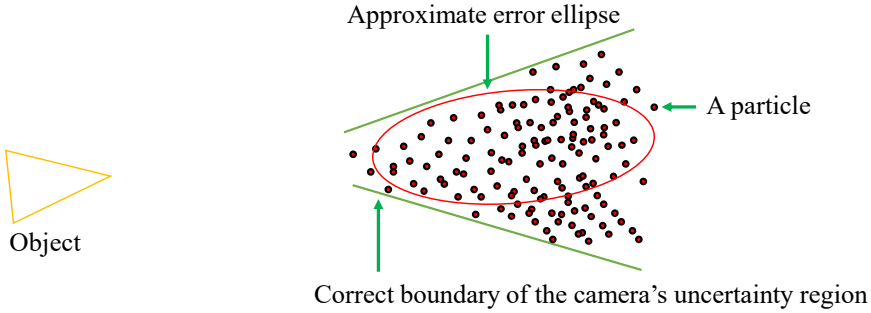


Figure 7.11: Comparison of a hypothetical set of particles and the fitted ellipses schematically

In the second set of experiments, beacons are utilized to measure the accuracy of the estimated pose during navigation. Three beacons are positioned in the indoor environment, and the robot moves around the object of interest. The ground truth position is measured with respect to the beacons with the help of the developed method in Appendix B. In Figure 7.12. a set of 1000 most likely positions are shown in blue, and the position corresponding to the maximum likelihood is shown in a white asterisk. As is seen in this figure, the uncertainty is very isotropic due to the position of the beacons and the robot.

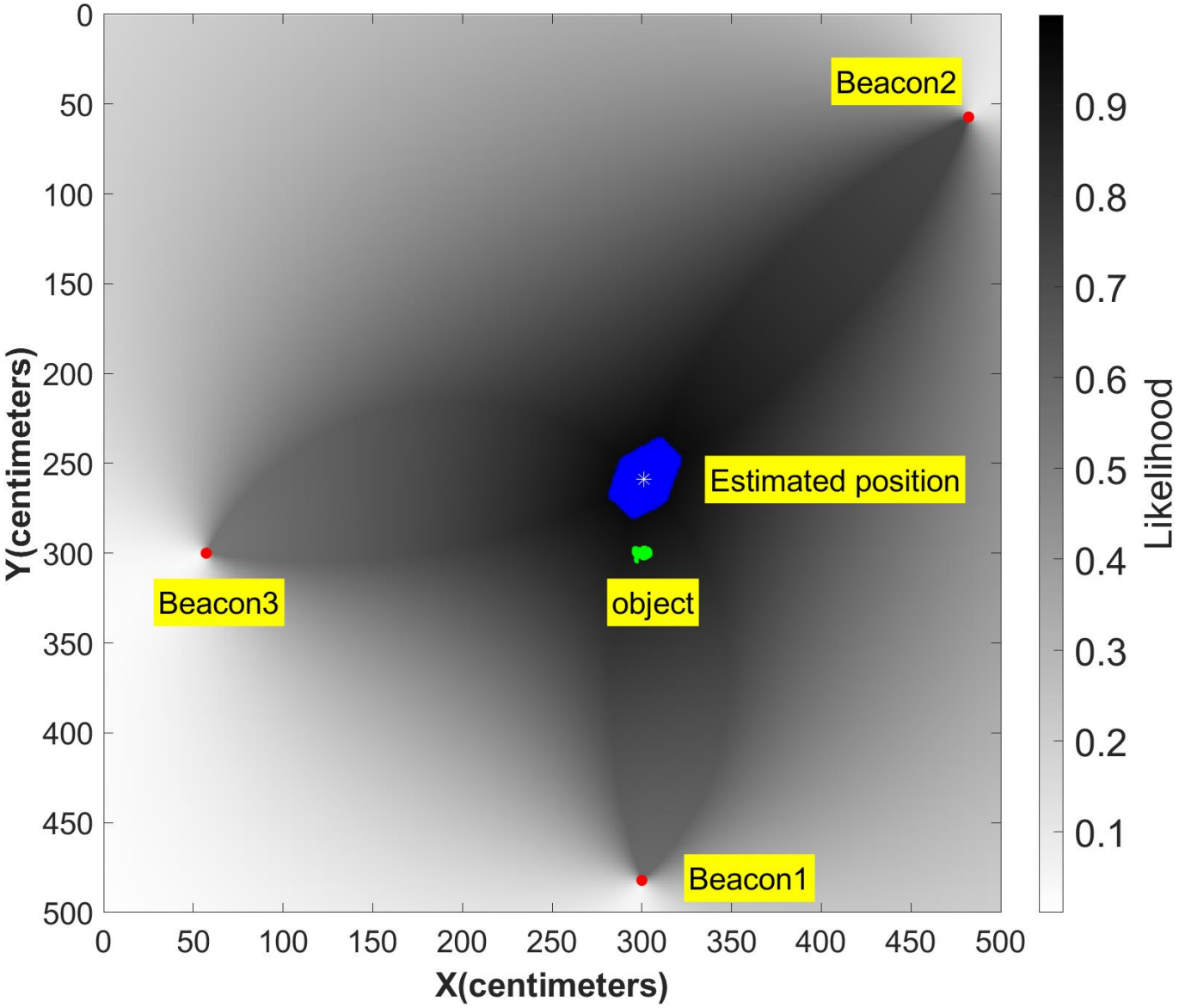


Figure 7.12: The best and the uncertainty estimates of the robot's position using the beacons

Table 7.12 shows the results obtained for the errors of tightly/loosely coupled methods in four tests. Test 15 includes severe occlusion to demonstrate the robustness of methods in such scenarios. The error of the estimated pose is measured in many locations along the trajectory, and the reported number is the average of these errors. Table 7.12 shows that the tightly coupled solution provides an overall lower error (with an error rate between 0.03% to 0.06 % of the total trajectory). The main contributor to the lower accuracy of loosely coupled in this experiment is the higher background clutter and occlusions, leading to higher failure rates. The IoU and the failure rate for these experiments are provided in Table 7.13 and Table 7.14. As was expected, the tightly coupled approach produces a much higher IoU and a much lower failure rate.

The estimated position, the ground truth, and the uncertainty of the ground truth in x and y axes are shown in separate plots in Figure 7.13 for Test 14. The estimated position falls within the confidence interval of the ground truth in most cases (especially in the x direction). The ground truth is measured at eight positions. The confidence interval is obtained by finding the range of cells in the x and y axis where their overall likelihood reaches 96.00% (more information about beacons is in Appendix B).

Table 7.12: Error in the estimated position of the tightly and loosely coupled methods

Exp.	**Experiments details**					
	Loosely coupled		*Tightly coupled*		*Occlusion Exists?*	*More information*
	EV (cm)	*EV/DT no units*	*EV (cm)*	*EV/DT no units*		
Test 14	40.0	0.053	24.4	0.032	no	bright light, longer trajectory
Test 15	37.7	0.089	19.2	0.045	yes	dim light, shorter trajectory
Test 16	39.2	0.091	24.3	0.057	no	dim light, shorter trajectory
Test 17	41.9	0.112	14.1	0.038	no	bright light, shorter trajectory

Table 7.13: IoU (no units) of the tightly/loosely coupled methods

Exp.	Experiments details			
	Loosely coupled	*Tightly coupled*	*Occlusion Exist?*	*More information*
Test 14	0.440	0.763	no	bright light, longer trajectory
Test 15	0.529	0.816	yes	dim light, shorter trajectory
Test 16	0.531	0.830	no	dim light, shorter trajectory
Test 17	0.511	0.845	no	bright light, shorter trajectory

Table 7.14: The failure rate (no units)of the tightly/loosely coupled methods

Exp.	Experiments details			
	Loosely coupled	*Tightly coupled*	*Occlusion Exist?*	*More information*
Test 14	0.61	0.06	no	bright light, longer trajectory
Test 15	0.37	0.00	yes	dim light, shorter trajectory
Test 16	0.36	0.04	no	dim light, shorter trajectory
Test 17	0.48	0.00	no	bright light, shorter trajectory

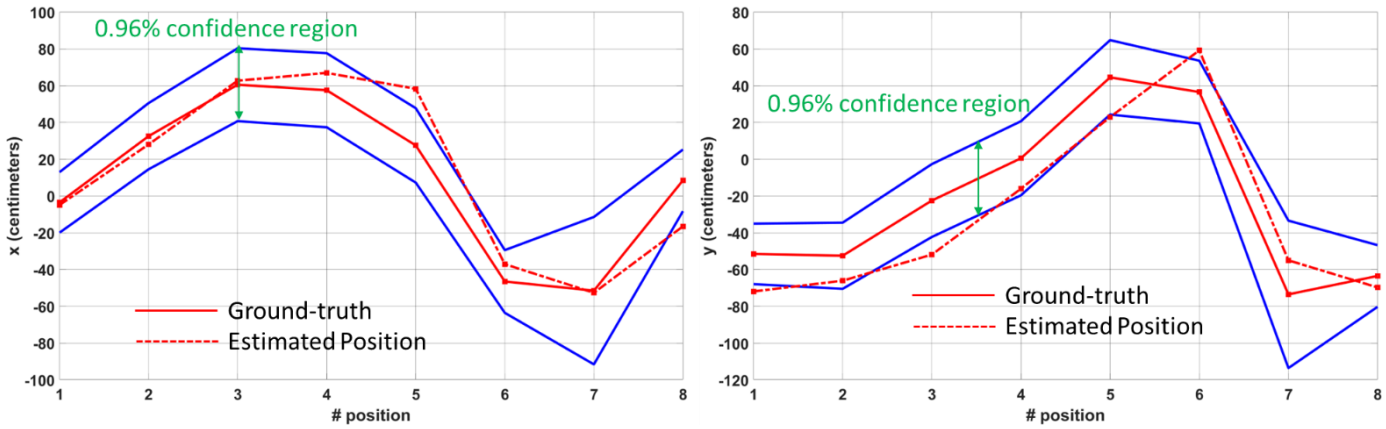


Figure 7.13: Comparison of the estimated positions in the x and y directions.

The estimated trajectory of the robot, particles, and the error ellipses for the tightly coupled method (Test 16) is shown in Figure 7.14. This experiment includes some failure rates (see where the uncertainty increases). The two failure cases in these experiments are mainly due to background clutter, but overall, failure cases happen infrequently. Further, as can be seen in Figure 7.14, the solution can recover one or two epochs after the failure. On the other hand, the loosely coupled method, unfortunately, fails many times along the trajectory (as can be seen in Figure 7.15).

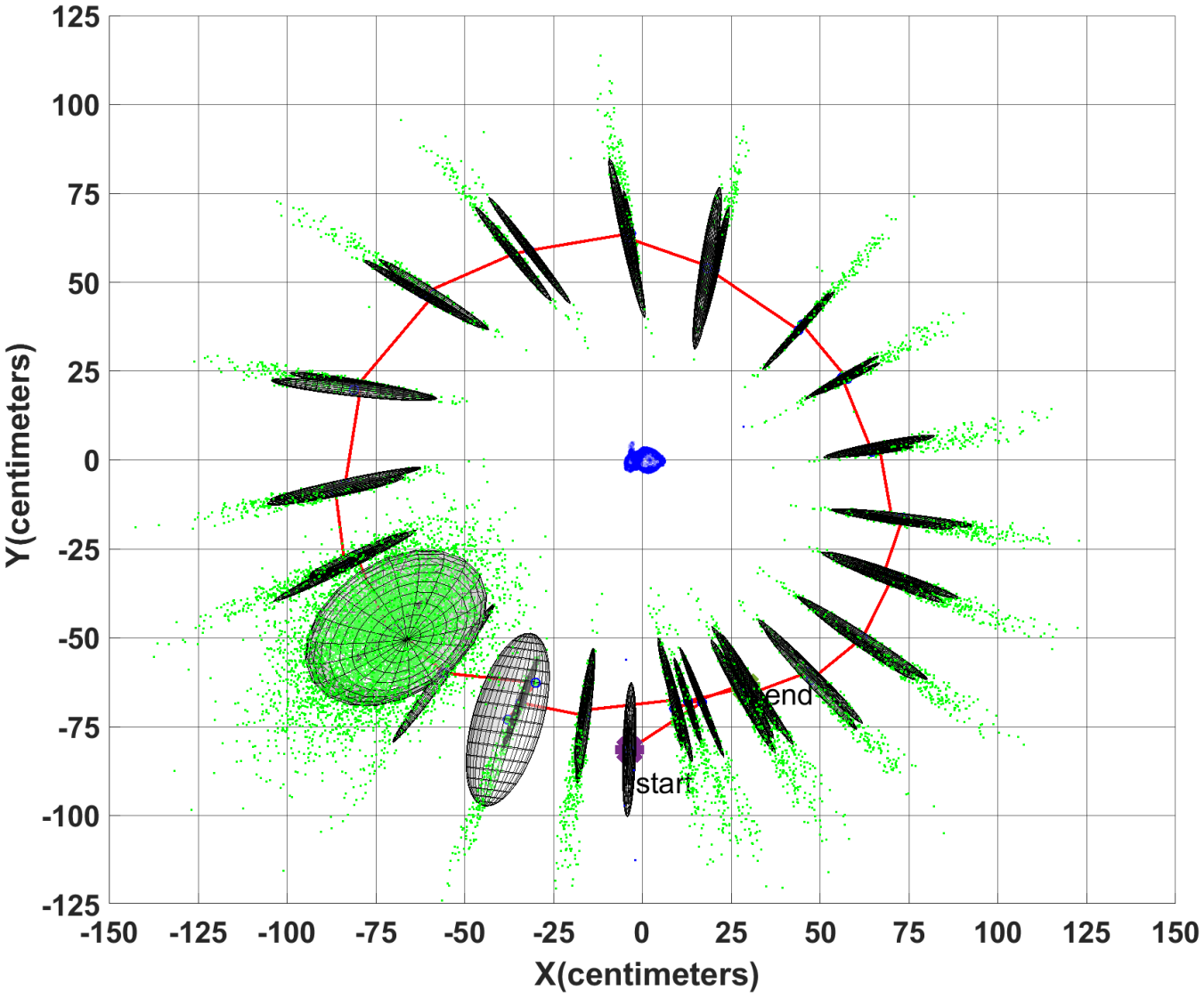


Figure 7.14: Particles and uncertainty ellipsoid of the trajectory for tightly coupled method.

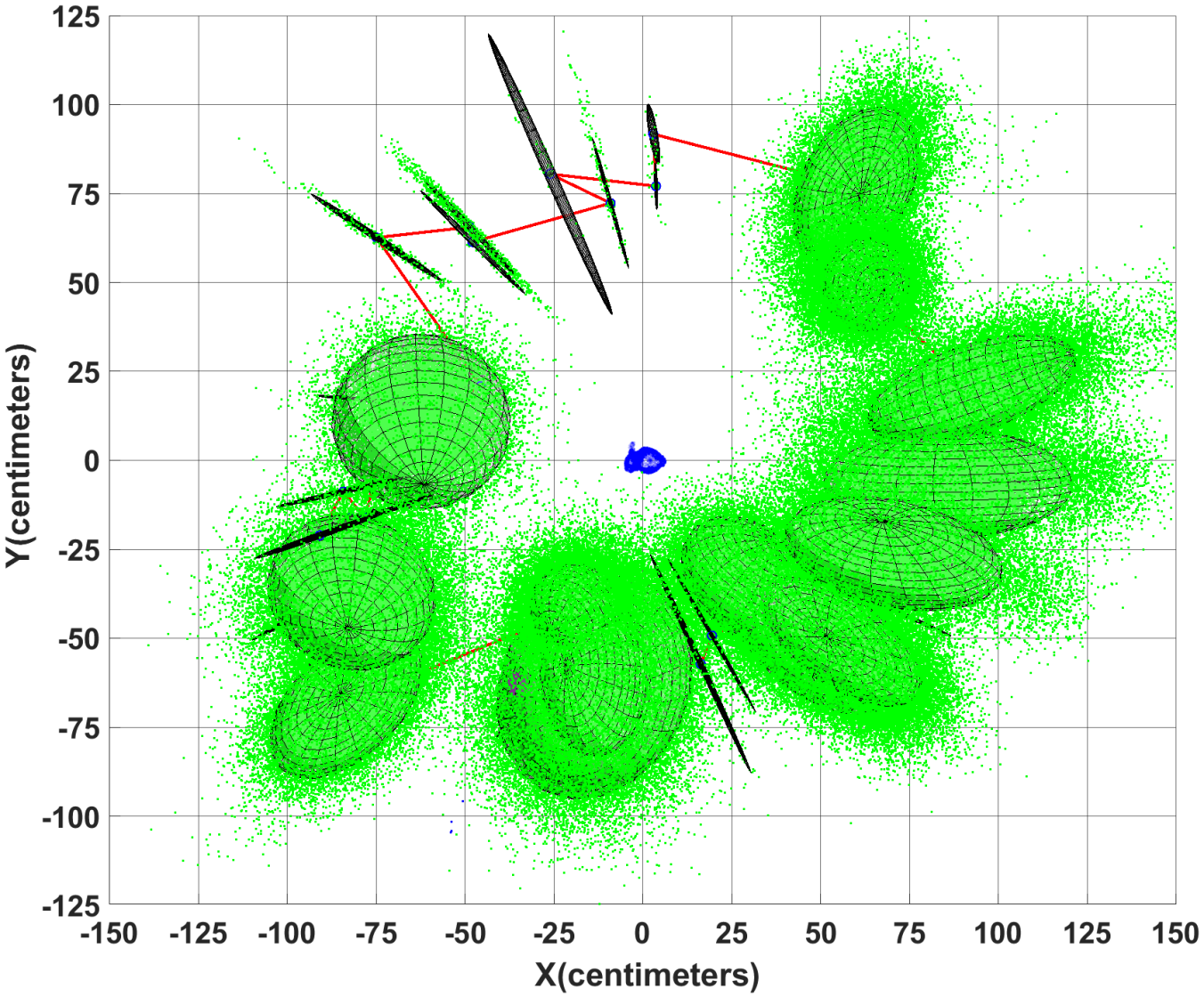


Figure 7.15: Particles and uncertainty ellipsoid of the trajectory for loosely coupled method.

The performance of the loosely and tightly coupled solutions for the different particle numbers is provided in Table 7.15. Based on these results, to achieve a comprise between the accuracy and the computational time for the tightly coupled approach, approximately 15000 particles can be used. Similar to the previous results (Table 7.11), increasing particles increases the IoU values of the tightly coupled method. Unfortunately, increasing the particles for the loosely coupled achieves improvements up to 5000 particles. Increasing the particles beyond these numbers does not seem to result in improvements.

Table 7.15: The results for different numbers of particles

Particle number	Experiments details							
	Loosely coupled				*Tightly coupled*			
	EV (cm)	*IoU no units*	*Time (s)*	*Fail rate.*	*EV (cm)*	*IoU No units*	*Time (s)*	*Fail rate.*
3500	124.6	0.349	55.8	0.78	94.8	0.592	260.9	0.32
5000	62.6	0.456	65.8	0.63	23.8	0.764	399.4	0.00
7500	63.9	0.461	80.2	0.58	14.2	0.806	529.7	0.00
10000	42.2	0.506	104.7	0.51	21.6	0.794	717.7	0.00
12500	59.6	0.409	126.5	0.56	20.2	0.811	927.9	0.02
15000	40.7	0.538	146.7	0.49	15.3	0.831	1061.6	0.00
17250	56.7	0.482	175.6	0.56	13.1	0.837	1172.7	0.00
20000	66.3	0.414	199.9	0.54	14.9	0.830	1402.2	0.00
40000	58.6	0.442	524.2	0.36	15.7	0.849	2933.9	0.02

In order to investigate how the errors in the motion model can influence the accuracy of the pose estimations, distortions are added to the odometry measurements. The odometer's reading can provide linear and angular velocities (as explained in Section 3.5). The distortions (denoted as ΔV_d) are added to linear velocity (the unit is in cm/s) every epoch that an odometry reading becomes available. This type of distortion can be interpreted as a bias error. Figure 7.16 summarizes these results. The magnitude of the distortion ranges from 0 to 30 cm/s. The IoU values decrease as the distortions increase. However, IoU stays relatively high up to 15 cm/s error; therefore, it can be concluded that the developed method has a certain level of robustness to systematic errors in odometry readings.

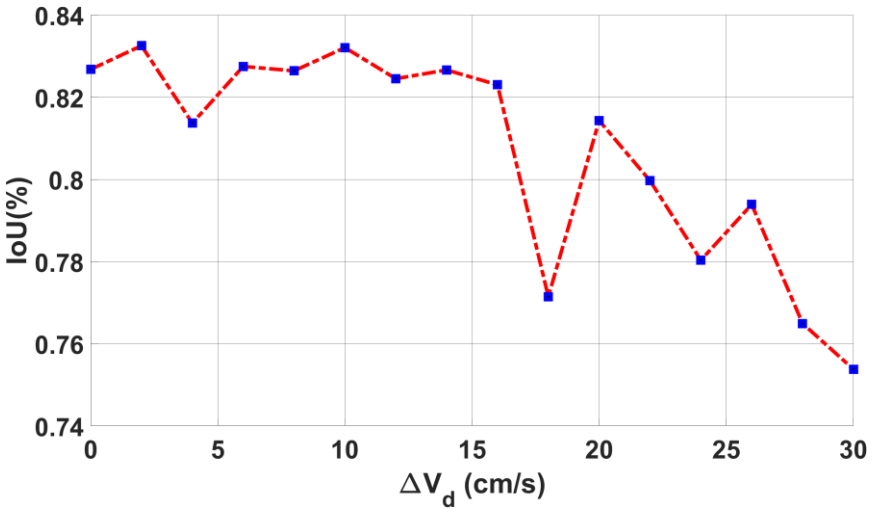


Figure 7.16: IoU for a range of distortions added to the wheel odometry.

7.6 Assessment of the Object-level Solution Using the Handheld Device (for a single object)

In this section, the result of tightly and loosely coupled object-level with IMU/monocular camera fusion is provided. The ground truth is measured as the distance between the initial and the final position of the device (see Chapter 6). The experiments are performed in two different indoor environments where the object of interest is placed on a table, and the device is moved around the object, capturing it from different viewpoints. The performance of the methods has been studied under different illumination conditions, levels of background clutter, and distances of the camera to the object of interest. The IMU measurements are captured at 50 Hz, and the images are captured at 2-3 Hz for these experiments. The image segmentation is performed in each image (for the loosely coupled method, the pose estimation is performed in each image, while for the tightly coupled method, the pose is estimated only in one image). There are key differences between the handheld device and the wheeled robot. Unlike wheeled robots, a handheld device can suffer from the distortions caused by human hand vibrations. Further, the motion estimated from MEMS-based IMU is often less accurate than the wheel odometry as the duration of the trajectory increases. Finally, unlike wheeled robots, which have a motion constrained to a plane (the robot is moving on a floor), the handheld device's motion is not restricted to a plane. Due to these sources of

distortions and higher degrees of freedom, it is possible that more particles will be required. This can lead to an increase in the computations. The effect of particle number is studied in this section as well.

In the following experiments, two approaches are used to generate the proposal distribution using IMU mechanization. One approach is to add uncertainty to the raw readings from the accelerometers and the gyroscope. In this approach, the additive noise is sampled from a Gaussian distribution with a known variance. This variance is obtained for each accelerometer/gyroscope axes in an offline calibration process. The second approach is to add the noise to the estimated attitude and the velocity of the camera's trajectory after the pose is estimated using mechanization. Unlike the first method, it is not clear how the variance for the noise can be set for this approach, and often tuning is required. In the experiments, both approaches are used to include noise.

In order to compare the tightly and loosely coupled methods, three experiments are performed. The number of particles for both cases is set to 25000. The errors are shown in Table 7.16 (in cm) and Table 7.17 (ratio to total trajectory length). The shorter range approximately corresponds to 50 cm while the longer corresponds to 100 cm. Overall, the tightly coupled outperforms the loosely coupled. Based on these experiments, the tightly coupled solution has an error of approximately 5.4 cm to 13.0 cm using EV for the position estimation (error ratio of 0.013 to 0.028). Further, an error of approximately 2.8 to 11.2 cm is obtained using MAP for the position estimation (error ratio of 0.006 to 0.039). The experiments indicate that the camera's overall distance from the object of interest did not have a deterministic effect on the errors. Further trajectory duration does not increase the error, indicating the solution is not drifting over time. The method in (Chermak et al., 2019) utilized a similar solution to the SLAM problem using IMU/ camera fusion and achieved an error of approximately 0.01 to 0.037 (ratio to the total length of trajectory). These errors are very similar to the results obtained in this thesis. Further, their approach relies on a stereo camera.

Table 7.16: The error (cm) in the estimated position of the tightly and loosely coupled using the handheld device

Exp.	Experiments details					
	Loosely coupled		*Tightly coupled*		*Range*	*More information*
	EV (cm)	*MAP (cm)*	*EV (cm)*	*MAP (cm)*		
Test 18	54.6	17.2	7.6	11.2	short	low clutter, shorter trajectory
Test 19	170.9	6.1	5.4	9.7	long	low clutter, longer trajectory
Test 20	11.5	9.7	13.0	2.8	long	low clutter, longer trajectory

Table 7.17: The error (ratio) in the estimated position of the tightly and loosely coupled using the handheld device

Exp.	Experiments details					
	Loosely coupled		*Tightly coupled*		*Range*	*More information*
	EV/DT (no units)	*MAP/DT (no units)*	*EV/DT (no units)*	*MAP/DT (no units)*		
Test 18	0.191	0.060	0.026	0.039	short	low clutter, shorter trajectory
Test 19	0.426	0.015	0.013	0.024	long	low clutter, longer trajectory
Test 20	0.024	0.021	0.028	0.006	long	low clutter, longer trajectory

Four more experiments were conducted using the tightly coupled method. The results of these experiments are summarized in Table 7.18 (error in cm) and 7.19 (error over the total distance travelled). The obtained error ranges from 4.1 to 13.1 cm. Further, the error is about 0.005 to 0.021 of the total distance travelled (see EV/DT). Similar to the previous experiments. in these experiments, the solution did not experience drifts. And it can be expected to achieve similar results for much longer motions. The distance of the camera from the object of interest also seems not to affect the accuracy. However, a lower

cluttered environment results in smaller errors, as expected. The failure rate is 0.00 (see Table 7.20), and thus, it was possible to update the weights for every image. The IoU is reported to be in a range from 0.805 to 0.821. Overall, no degradation of the performance for the handheld device in comparison to the wheeled robot is observed.

Table 7.18: The error (cm) in the estimated position of tightly coupled using the handheld device

Exp.	**Experiments details**			
	Tightly coupled		*Range*	*More information*
	EV (cm)	*MAP (cm)*		
Test 21	7.8	22.4	short	lower clutter, shorter trajectory
Test 22	4.1	18.9	long	lower clutter, shorter trajectory
Test 23	12.3	35.7	medium	higher clutter, longer trajectory
Test 24	13.1	32.7	medium	higher clutter, longer trajectory

Table 7.19: The error (ratio) in the estimated position of tightly coupled using the handheld device

Exp.	**Experiments details**			
	Tightly coupled		*Range*	*More information*
	EV/DT (no units)	*MAP/DT (no units)*		
Test 21	0.015	0.043	short	lower clutter, shorter trajectory
Test 22	0.005	0.023	long	lower clutter, shorter trajectory
Test 23	0.018	0.052	medium	higher clutter, longer trajectory
Test 24	0.021	0.052	medium	higher clutter, longer trajectory

Table 7.20: The IoU and the failure rate of tightly and loosely coupled using the handheld device

Exp.	Experiments details			
	Tightly coupled		*Range*	*More information*
	IoU *(no units)*	*Fail rate* *(no units)*		
Test 21	0.805	0.00	short	lower clutter, shorter trajectory
Test 22	0.805	0.00	long	lower clutter, shorter trajectory
Test 23	0.819	0.00	medium	higher clutter, longer trajectory
Test 24	0.821	0.00	medium	higher clutter, longer trajectory

The trajectory of the device, as estimated during the experiments, is shown in Figure 7.17 (Test 21), Figure 7.18 (Test 24), and Figure 7.19. Figure 7.17 (a) shows the EV (expected value of the particles) and the trajectory in green and red, respectively. In this experiment, the handheld device is moved around the object (shown in blue) while capturing it from different poses. The device is moved up and down while the range of the camera to the object is kept approximately the same. The errors are estimated by measuring the distance between the start and the end positions (and compared to the ground truth value). From the particles and the corresponding error ellipsoids (see Figure 7.17 (b)), it can be seen the uncertainty is the highest in the direction from the camera to the object. This is similar to the results obtained using the wheeled robot in the previous section and it is due to the scale ambiguity of the camera.

The errors follow a similar pattern as shown in Figure 7.18 for Test 24. The key difference is that Test 24 has a longer trajectory than Test 21. Despite this longer trajectory, no significant drifting of the solution is observed (see error ellipsoids and the particles in Figure 7.18). This is an expected result since the position of the device is updated within fixed intervals using the images, therefore, mitigating drift. The solution is expected to not drift for longer trajectories as long as the update using images is available.

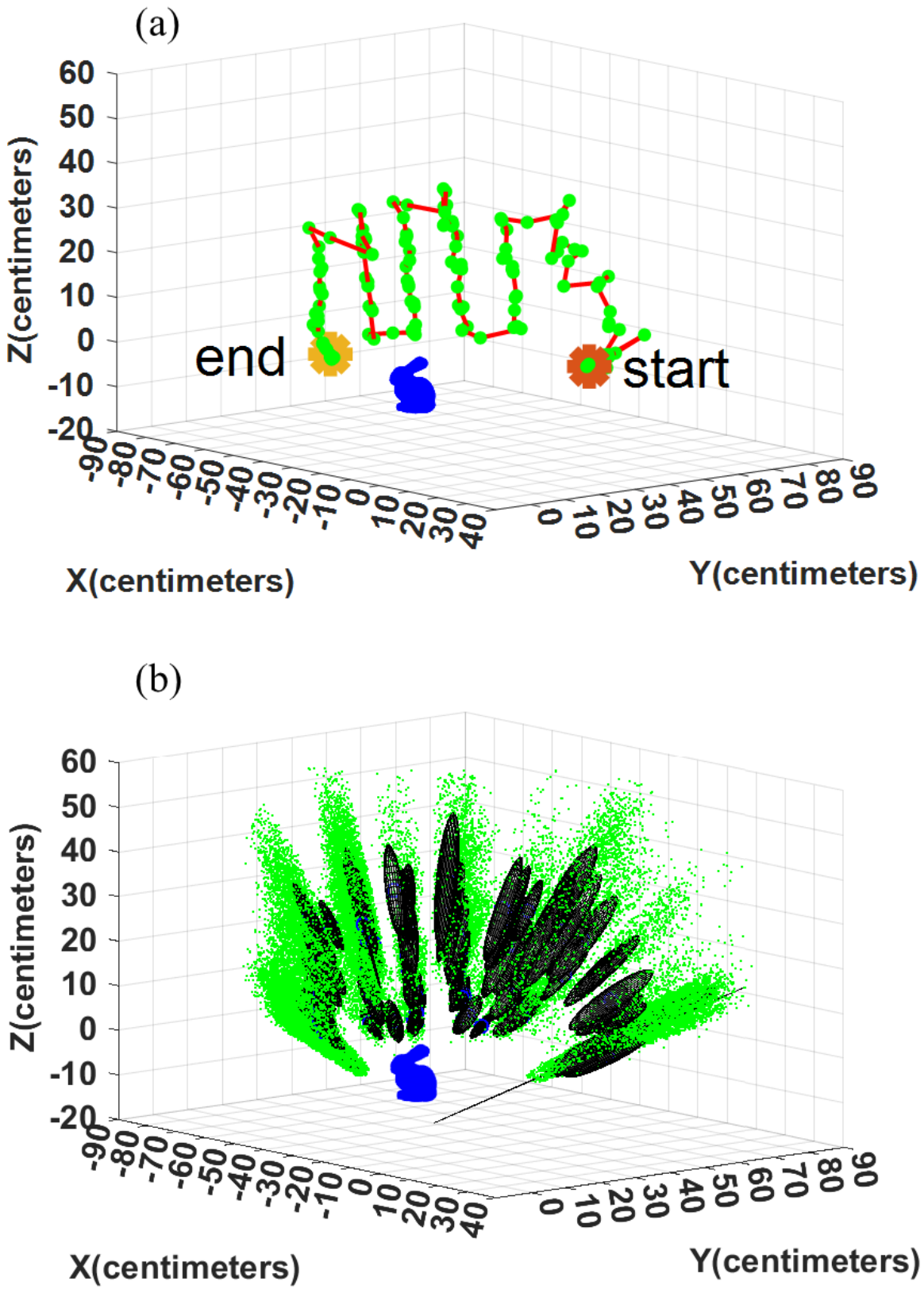


Figure 7.17: The estimated trajectory and the error ellipsoids of Test 21.

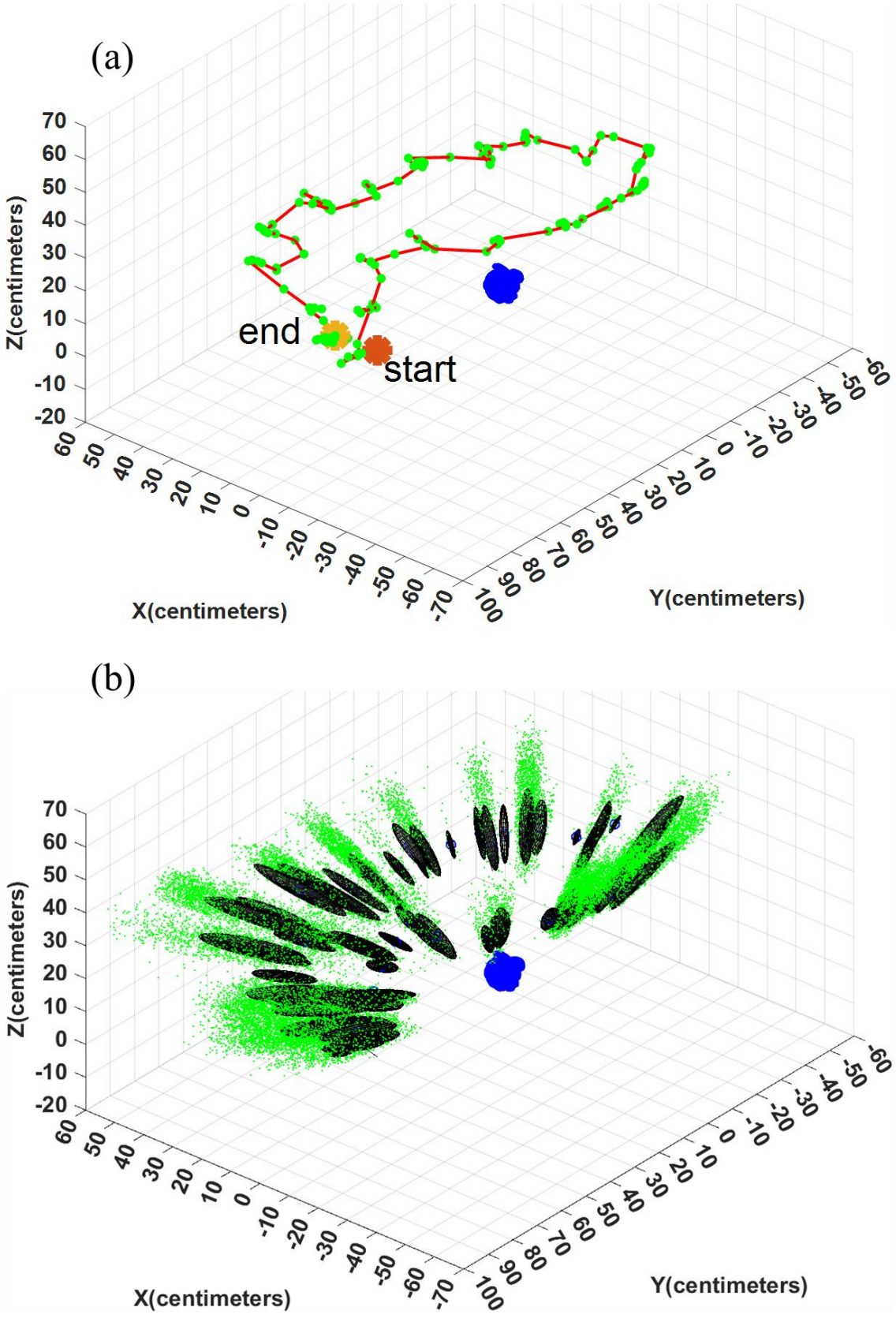


Figure 7.18: The estimated trajectory and the error ellipsoids for Test 24.

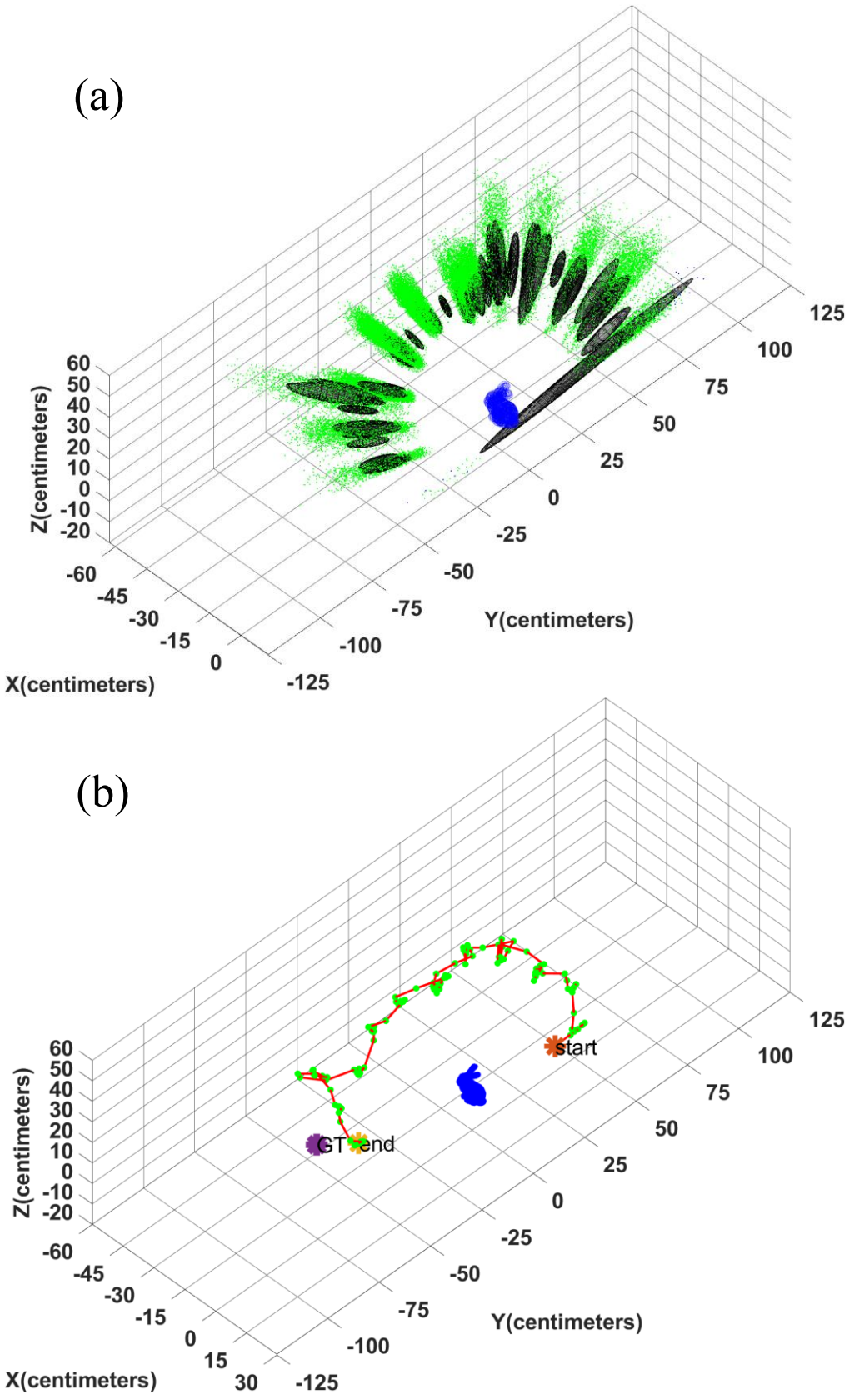


Figure 7.19: The estimated trajectory and the error ellipsoids.

The performance of the tightly coupled method is tested using different numbers of particles. The results are summarized in Table 7.21. Based on this table, IoU values have increased consistently with the increasing number of particles. However, the improvement is marginal (IoU is increased from 0.813 for 5000 particles to 0.837 for 19000 particles). Based on the results shown in Table 7.21, a similar error is achieved using particle numbers in the range from 5000 to 19000. The error in the range is between 10 to 13.4 cm, and increasing the number of particles from 5000 to 19000 does not show any consistent pattern in reducing error. The error might be due to other sources that are not addressed by increasing the number of particles. Such systematic errors can be due to Challenges II and III, as mentioned previously. Based on the results in this section, it is recommended to limit the number of the particles for the handheld device to 5000 since the runtime of the algorithm increases as more particles are used (e.g., increasing the number of particles from 5000 to 19000, approximately increase runtime twice). In the next section, it will be shown how the runtime can decrease further by using an ultrasonic rangefinder.

Table 7.21: Summary of the performance of the tightly coupled method using different numbers of particles.

Particle number	***Experiments details (Tightly coupled)***				
	EV (cm)	*EV/DT (no units)*	*IoU (no units)*	*Time (s)*	*Fail rate. (no units)*
5000	12.4	0.018	0.813	6149.2	0.00
7000	11.4	0.020	0.821	8577.4	0.00
9000	10.9	0.019	0.824	9291.8	0.00
13000	10.8	0.018	0.830	11015.2	0.00
17000	13.4	0.022	0.835	12488.7	0.00
19000	11.0	0.018	0.837	13390.3	0.00

7.7 Assessment of the Tightly Coupled Object-Level SLAM with Ultrasonic Rangefinder

In this section, the results using the fusion of a monocular camera and an ultrasonic sensor are provided. With the help of a sensor such as an ultrasonic rangefinder, the initialization can be achieved without delays. Further ultrasonic sensors provide additional depth information that can help eliminate many particles that do not fall within a valid uncertainty region around the distance from the observed object. The elimination of the particles (though assigning low weights) will greatly decrease the computational cost of the developed algorithm. This was discussed in Chapter 3 extensively.

Ultrasonic rangefinder's transmitted waves have a chance of missing the target objects. This issue happens when the camera observes the object of interest, but the ultrasonic sensor is not facing the object. Therefore, the wave can hit another object. In order to detect this, a simple test is designed in this thesis (see Section 3.6). The test measures the overlapping area between the projected uncertainty region of the ultrasonic observation (onto the image) with the segmented object. If this overlapping area passes a certain threshold (μ), then it is more likely that the ultrasonic wave is reflecting from the object of interest. If the threshold value is set high, it might lead to false negatives; if set low, it can lead to false positives.

In order to test the developed method, the threshold has varied between 0.1 to 0.9. Further, the beam angle of the ultrasonic rangefinder is assumed to be between 10° to 20° (the nominal beam angle of the sensor used in the experiments is 15°). The results are provided in Figure 7.20. Figure 7.20 (a) shows that for a higher beam angle ($16^{\circ}$to 20°) and a threshold larger than 0.4, a precision of 1 can be achieved. Figure 7.20 (b) shows that for lower beam angle values (10° to 16°) and a wide range of thresholds (0.2 to 0.8), the recall value remains in the range from 0.7 to 0.8. Since it is important to avoid false positives, attaining higher precision is preferable. Based on this discussion, it seems that a beam angle of about 14 to 16° (very close to the nominal beam angle) and a threshold of 0.4 to 0.8 can be utilized for the task of

object-level SLAM. Such a large range of possible values for the threshold shows the ease of tuning the developed method.

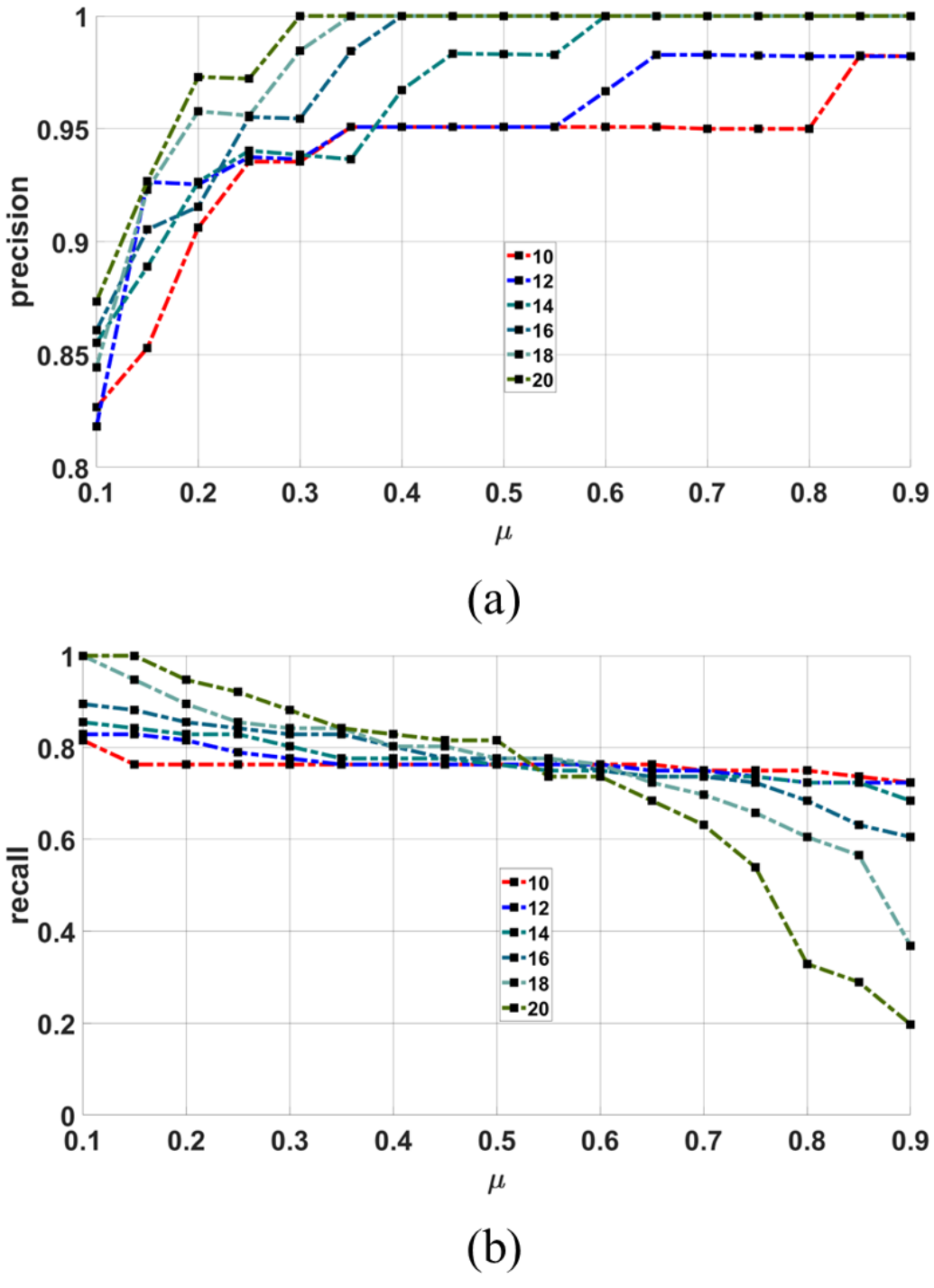


Figure 7.20: Precision(a) and recall (b) using different thresholds and beam angle values.

As mentioned, fusion with an ultrasonic rangefinder can improve computation efficiency. For the comparison, the runtime and other performance metrics of the IMU/camera tightly coupled method with and without fusion with an ultrasonic sensor are provided in Table 7.22. Based on this table, no significant improvement in the performance has been obtained in terms of accuracy using fusion with the ultrasonic sensor. However, a significant reduction in the runtime of the algorithm is obtained.

Table 7.22: Comparison of the tightly coupled method with (w) and without (w/o) ultrasonic rangefinder.

Exp.	**Experiments details**							
	Tightly coupled (W/O Ultrasonic)				*Tightly coupled (W Ultrasonic)*			
	EV (cm)	*IoU (no units)*	*Time (s)*	*Fail rate (no units)*	*EV (cm)*	*IoU (no units)*	*Time (s)*	*Fail rate (no units)*
Test 25	6.5	0.815	9753.3	0.02	9.4	0.747	4107.3	0.03
Test 26	16.2	0.810	8036.9	0.04	13.4	0.722	3922.3	0.04
Test 27	10.0	0.846	5879.7	0.03	8.0	0.768	2786.4	0.03

7.8 Assessment of the Tightly Coupled Object-level SLAM (for multiple objects)

In the previous sections, all the experiments relied on a single object. In real-life scenarios, relying on a single object for the localization of the camera is disadvantageous. The object of interest can be occluded frequently, and therefore, the particle filter will fail to perform weighting and resampling. The lack of possibility to update the particle weights using observations will force the algorithm to rely on the motion model to estimate the trajectory of the robot/device. However, this motion model is based on dead reckoning and under such circumstances, it is possible that the accuracy of the solution will degrade due to the accumulation of the error.

A possible approach to address this issue is to rely on more than a single object during the navigation. In multiple object-based solutions, it is more likely to find one or more objects that are not occluded. In this section, the developed tightly coupled method is used with multiple objects. The main algorithmic difference between using multiple objects instead of a single object is the requirement to perform an initialization anytime a new object is observed.

Throughout the navigation, the developed algorithm tests if any of the objects are observed in the image. Based on the number of objects identified in the image, two possible scenarios can occur. In the first scenario, only one object is observed in the image. In this case, a data association is performed to determine which object the observation belongs to. The weighting and resampling steps for this scenario are the same as it was for a single object. In the second scenario, more than one object is found in the image. In this case, the IoU for each object can be estimated separately (for a particle), and the maximum value can be used to estimate the weight. It is important to note that this is not the only possible approach to weigh the particles. However, since the developed method is affected by Challenge II most frequently (which occurs during the occlusions), using a single object with higher IoU can be more advantageous (since the occluded objects will result in lower IoU and it is more advantageous not to rely on these objects in the weight update process.)

The results for the test with two objects are provided in Table 7.23. Based on this table, the error is reported to be 9.1 cm (0.012 of the total path traversed by the handheld device) using EV as the metric of assessment and 17.3 cm (0.023 of the total path traversed by the handheld device) using MAP as the metric of assessment. These results are very similar to the previous studies using a single object. Therefore, no significant improvement or degradation of the accuracy has been observed using the two objects. Further, the IoU is reported to be 0.815, which is also very similar to the results obtained using a single object, as provided in the previous sections.

Table 7.23: The results are obtained using two objects. Based on this table, no significant improvement or degradation of the accuracy is observed using two objects.

Exp	Experiments details					
	EV (cm)	*MAP (cm)*	*EV/DT (no units)*	*MAP/DT (no units)*	*IoU (no units)*	*Fail rate (no units)*
Test 28	9.1	17.3	0.012	0.023	0.815	0.03

7.9 Boresight Calibration of Monocular Camera and IMU

The fusion of an IMU and a monocular has been used throughout this chapter to provide a solution to the object-level SLAM. A requirement for such fusions is to find the boresight calibration parameters between the two sensors. This section will follow the methodology explained in Chapter 5 to find the boresight calibration parameters. The developed method relies on estimating the normal vector to the floor in the camera's frame. This normal can be assumed to be antiparallel with gravity vector to great accuracy (and can be checked if it is so with a simple levelling device). Finally, the correspondence between the two vectors (normal and the gravity vector) can be used to find the boresight calibration parameters.

The experiments are performed using two devices. One of these is the handheld device introduced earlier. This device includes an Xsens IMU and an Arducam camera. The reference solution for the boresight calibration parameters is based on CAD models. The second device is a smartphone. The ground truth boresight calibration for the smartphone is obtained with the help of the method that is developed in Section 6.5. Experiments are conducted in four indoor environments. The environments of the experiments are shown in Figure 7.21. These images were captured at the University of Calgary. The synchronization of the IMU measurements and the images is achieved using UTC timestamps.

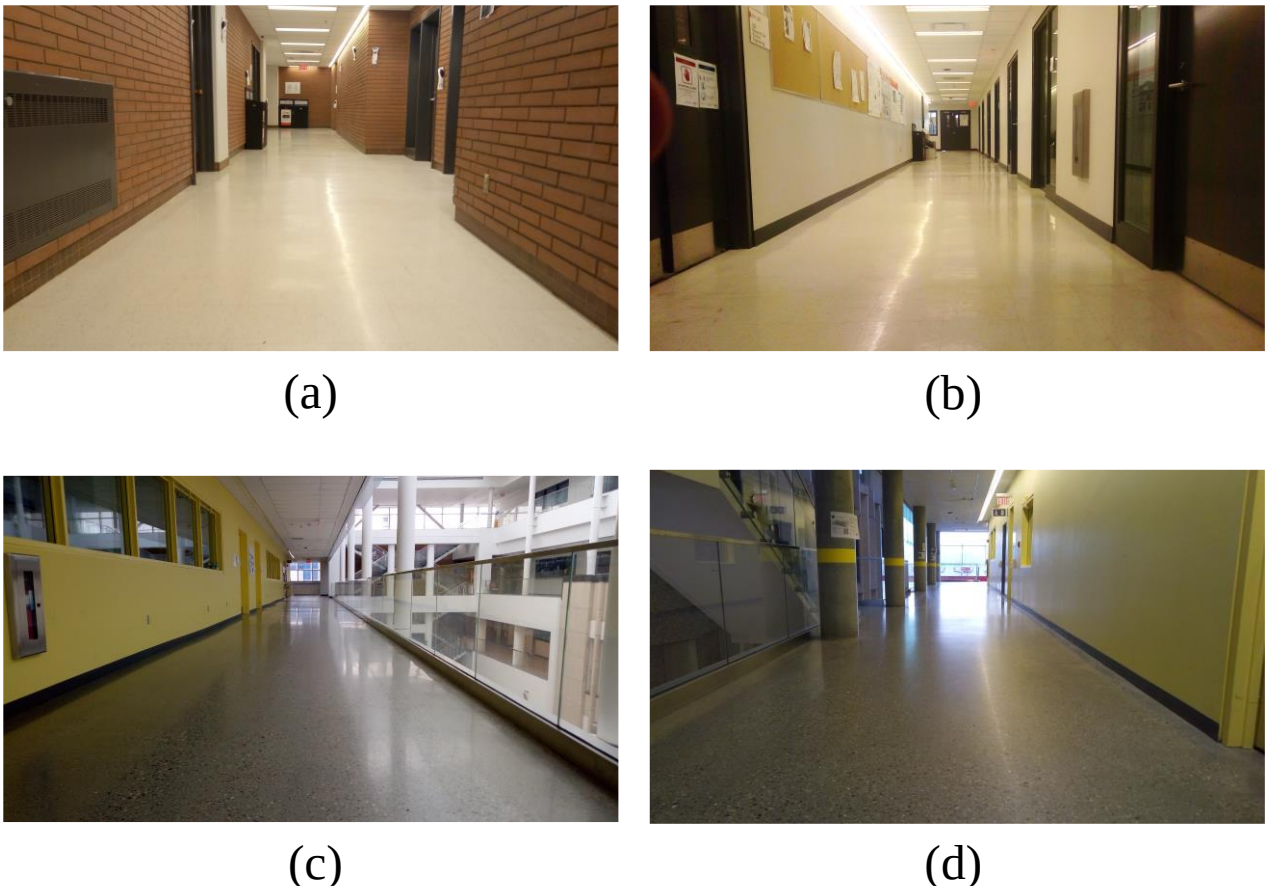


Figure 7.21: The four scenes where the boresight calibration is performed.

The accuracy of the developed method depends on correctly identifying the VPs. These VPs are divided into vertical (VVP) and horizontal (HVP) ones (see Section 5.3). In Table 7.24, the following notations are used for brevity. Stage 1 refers to HVP detection, and Stage 2 refers to VVP detection. Stage 1 (successful) denotes a successful detection, and Stage 1 (failed) denotes failed detection. Both stages can fail due to numerous reasons. For instance, a failure in VVP detection can be due to falsely identifying a VP corresponding to the horizontal structure (such as the boundary of the floor segment). A correct or incorrect identification is determined based on human judgment. Further, the error is measured using the inner product of the corresponding vectors to the estimated and the reference VVP. The reference VVP is found by identifying two edges corresponding to vertical structures (such as the edges of a wall) and intersecting lines corresponding to these edges.

The results (Table 7.24) show that the success rate of both stages simultaneously is 0.62 (Stage1 (successful) and Stage2 (successful)). Further, from Table 7.24, it can be demonstrated that whenever Stage 1 is successful, Stage 2 rarely fails (Stage 1 (successful) and Stage 2 (failed) only occur with a rate of 0.01). This indicates that removing the majority of the line segments that correspond to the HVP increases the success rate of detecting VVP. Finally, in the case that Stage 1 fails, Stage 2 is only successful at a rate of 0.22. This indicates that directly attempting to detect VVP is a more difficult task to achieve. The case Stage1 (failed) and Stage2 (failed) is not included in Table 7.24 for brevity. Based on Table 7.24, the accuracy of the detected VVP is overall very high (the inner product is about 0.98 with a maximum of 1).

Table 7.24: The success rate of stages 1 and 2 and the accuracy of VVP detection

Exp.	**Experiment's details**	
	occurrence rate *(no units)*	*inner product* *(of estimated and reference VVP)*
stage1 (successful) and stage2 (successful)	0.62	0.98
stage1 (successful) and stage2 (failed)	0.01	0.02
stage1 (failed) and stage2 (successful)	0.22	0.99

In order to measure the accuracy of the boresight calibration, the rotation matrix (corresponding to the orientation parameters of the boresight calibration) is decomposed into three Euler angles, and the Euclidean distance between the ground truth and the estimated angles is used to report the error. The results are provided in Table 7.25 for the handheld device. This table shows that the orientation error depends on the scene. The main reason for this dependence is the difference in precision and recall values achieved for the floor segmentation in each scene (the results are not provided here). Further, Table 7.25 shows that increasing the number of images improves the accuracy, and the best results are obtained by

combining the images from scenes b and c (3.49° error). The obtained results also outperform Kalibr (Furgale et al., 2013), a well-known method of the boresight calibration of IMU and monocular cameras that require checkerboards. It is also important to note that Kalibr requires initialization of the parameters. The accuracy of the developed calibration does not strictly increase with the addition of more images. To investigate this issue, the accuracy is measured for a subset of images in each scene. The subsets are created by randomly removing one image at a time and estimating the calibration parameters. A boxplot summarizing the results is provided in Figure 7.22. Further, Table 7.26 provides a summary of the median, 25th and 75th percentile for the error in the estimated orientation. Table 7.26 shows that the median error is 1.19° for images in scenes b & c, which is an improvement over using all the images (as shown in Table 7.25). The results indicate that the developed method lacks robustness to outliers. One possible solution is to use techniques robust to outliers (e.g., RANSAC) which is discussed in Chapter 8.

Table 7.25: The error in the orientation angles for the boresight calibration of IMU and camera (handheld device)

Exp.	**Experiment details**		
	Euler angles(degrees)	*Error(degrees)*	*Number of images*
scene a	[160.12, 21.36, -80.13]	19.72	28
scene b	[179.54, 22.81, -87.94]	3.62	66
scene c	[175.46, 22.75, -88.09]	4.93	74
scene d	[151.44 26.08, -63.08]	39.37	21
scene b & c	[179.84,22.80, -86.51]	3.49	140
Kalibr	[178.57, 22.5, -86.50]	3.80	199
reference solution	[180.00, 22.94, -90.00]	N/A	N/A

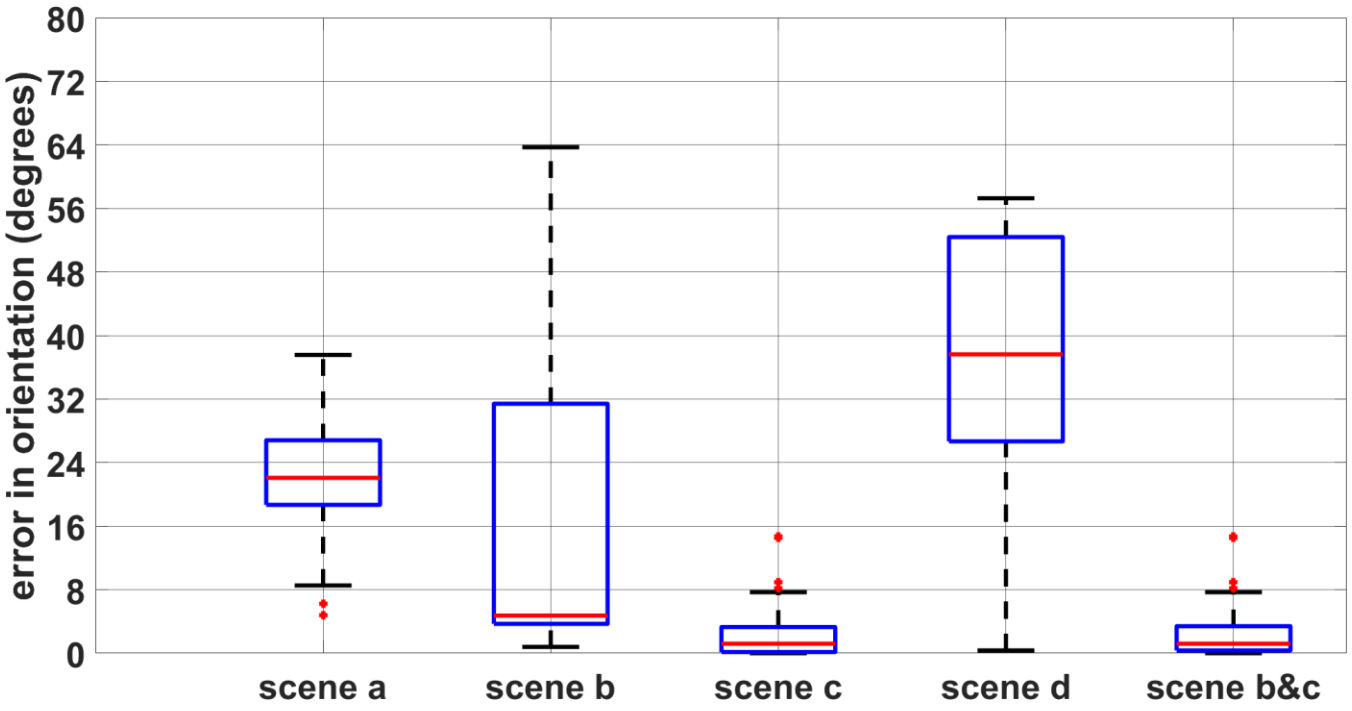


Figure 7.22: Boxplot of the measured error in each scene

Table 7.26: The obtained errors in the estimated orientation parameters using only a subset of the images.

Exp.	**Scenes**				
	Scene a	*Scene b*	*Scene c*	*Scene d*	*Scene b&c*
Error (75^{th} quartile) (degrees)	26.82	31.41	3.30	52.40	3.40
Error (median) (degrees)	22.07	4.74	1.19	37.63	1.19
Error (25^{th} quartile) (degrees)	18.68	3.69	0.14	26.67	0.35

Figure 7.23 shows the error of the developed method using different numbers of images for individual tests in more detail for scenes b & c. The accuracy is eventually improving by adding more images. However, the minimum (reported as 0.0031°) is not achieved using all the images, while using all the images results in 3.4926° error. The median error, as mentioned, is 0.1952°.

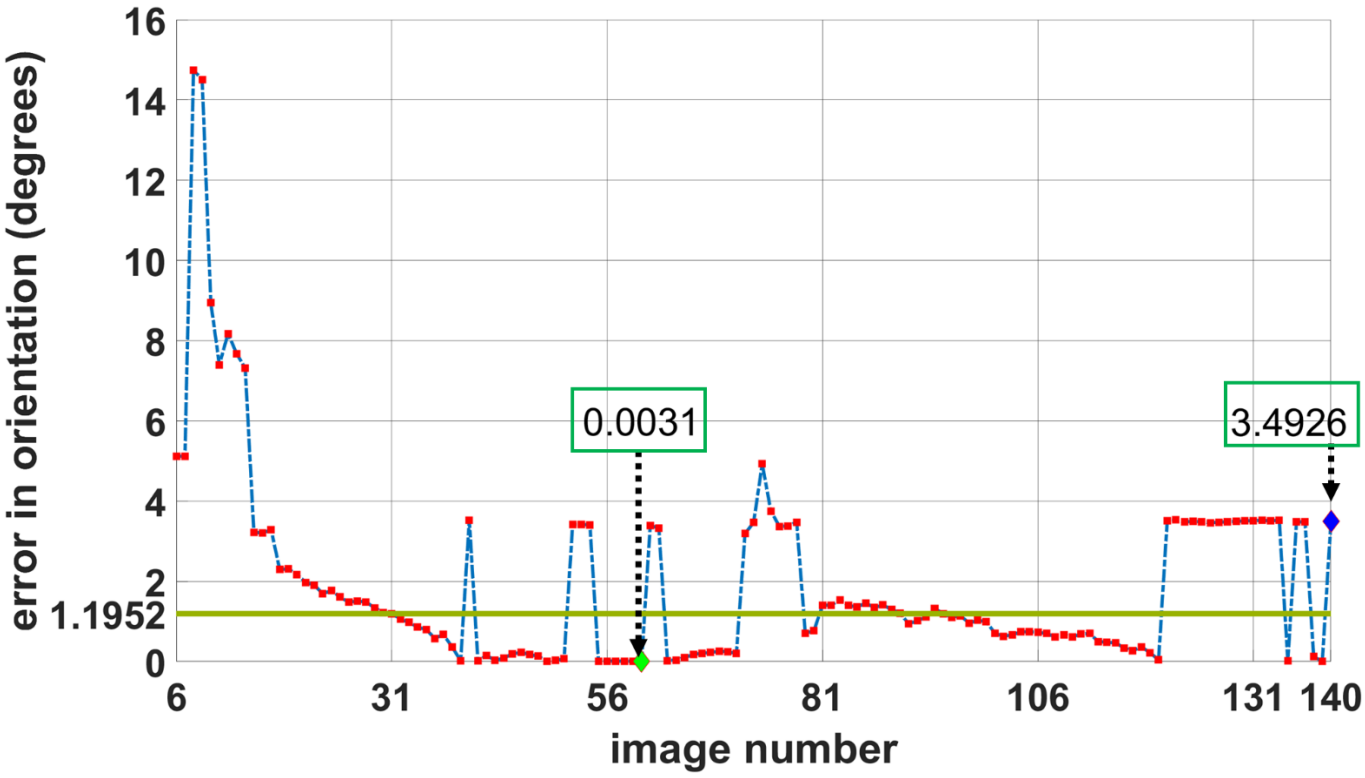


Figure 7.23: The error in the orientation parameters using different numbers of the images.

An important aspect of an estimation is to evaluate the uncertainty of the parameters. Such evaluation can be readily achieved using the known equations of error propagation (see Section 5.3 for more information). The developed boresight calibration estimates the nine elements of the rotation matrix (r_1:r_9) corresponding to the orientation between the two sensors (IMU and monocular camera). The estimated parameters, the uncertainty and the reference parameters are provided in Figure 7.24. The ground truth parameter is shown in a dashed red line, and the estimated parameter for a given number of images is shown in the solid red lines. The range of three standard deviations (-3σ,3σ) is shown in blue. It can be seen that all the elements are within the estimated uncertainty expect for r_5. These results indicate that increasing the number of images indeed helps some of the estimated parameters converge to the reference. This can be observed best by inspecting r_1, r_2, r_3 and r_6 .However, systematic errors also exist (see r_5 and r_9).

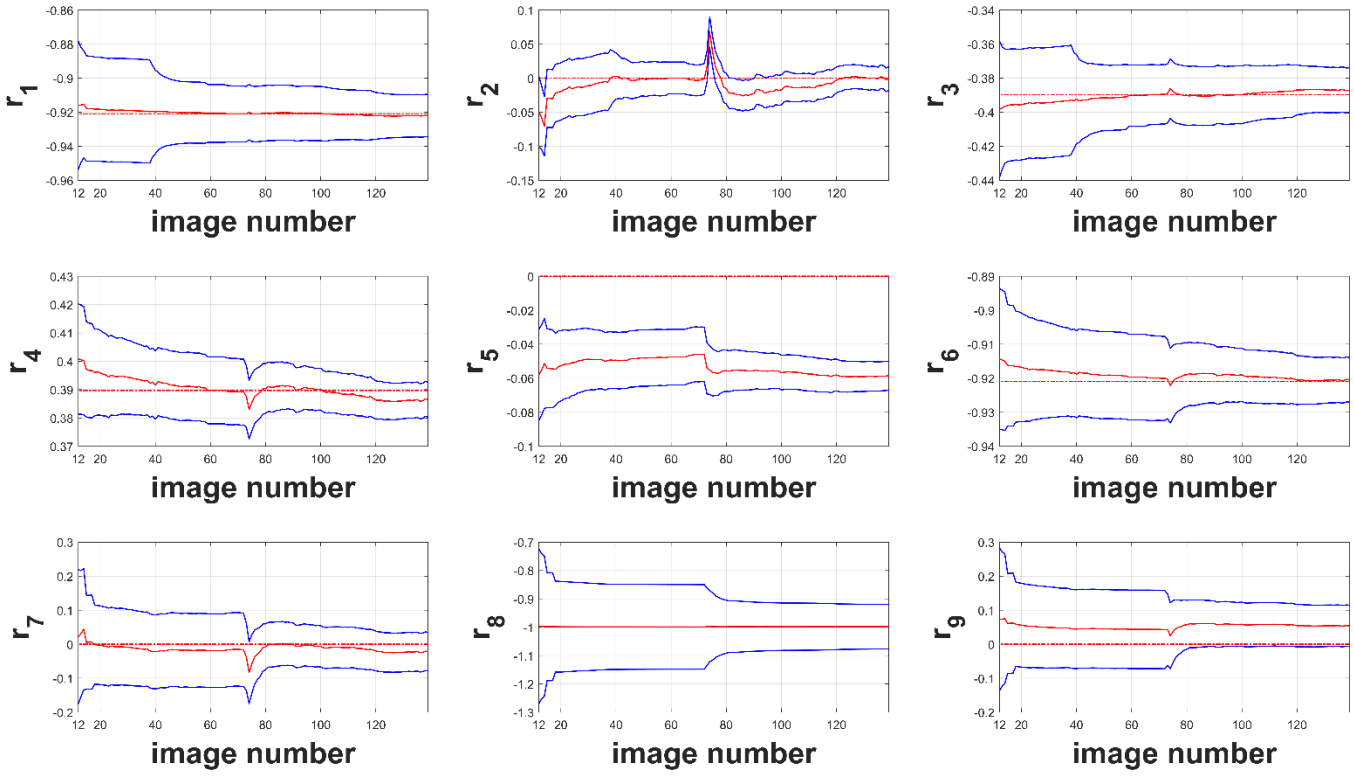


Figure 7.24: The estimated parameters and their uncertainty for different numbers of images.

Finally, the boresight calibration parameters obtained using a smartphone are provided in Table 7.27. The obtained result using the smartphone overall has a larger error due to the fewer images utilized.

Table 7.27: The error of the orientation angles for the boresight calibration of IMU and camera (smartphone)

Exp.	**Experiments**		
	Euler angles (degrees)	*Error(degrees)*	*Number of images*
scene a	[90.9,16.3,167.5]	14.4	10
scene b	[98.6, -15.92,165.5]	28.2	10
scene c	[91.3,13.2,168.3]	12.4	13
scene d	fail	fail	N/A
ground-truth	[97.89,9.37,178.07]	N/A	N/A

7.10 Extrinsic Calibration of a Monocular Camera and an Ultrasonic Rangefinder

Extrinsic (boresight and lever arm) calibration of a monocular camera and a rangefinder, such as an ultrasonic sensor with low angular resolution, is a difficult task. Classical methods rely on 20-30 points to be detected on a planar target (such as a checkerboard) with the rangefinder. Failure to detect such a number of points can result in a decrease in the accuracy of the extrinsic calibration. In Section 5.4, a novel method for the calibration of an ultrasonic rangefinder and a monocular camera is introduced. This approach relies on a 2D line-based map of an indoor manmade environment. Such maps can be built using state-of-the-art techniques designed for wheeled robots equipped with ultrasonic rangefinders. These line segments can be matched to the back-projected floor segment's boundary, detected in the image.

In this section, the results of the estimation of the orientation parameters are provided. The developed method is exhaustive and can be implemented in two different approaches. The first approach is a step-by-step optimization (sequential) of the parameters, and the second is a simultaneous estimation of the parameters. Due to the exhaustive nature of the developed algorithm, simultaneous optimization of all the angles is computationally very costly. The errors of the estimated extrinsic calibration parameters are provided in Table 7.28. It can be seen that the estimated orientation parameters are less than one degree for both sequential and simultaneous methods. Further, no significant improvement is achieved by estimating all the parameters at once.

The estimated parameters for extrinsic calibration can vary using different cost functions for the optimization. One of the main contributors to this difference in the results is how the features from the back-projected floor segment's boundary (identified in the image) are matched to the line segments in the map. In this section, two error functions, e_1 and e_2, with different matching strategies are tested (see Section 5.4). Based on Table 7.28, for the simultaneous estimation of all the angles, no difference can be seen using these functions. However, for the sequential approach, e_2 offers a lower error.

Table 7.28: The error in the orientation angles of the boresight calibration of the ultrasonic rangefinder and camera

Method	Experiment information	
	Euler angles (degrees)	*Error (degrees)*
Sequential (e_1)	[-10.1, -0.8, -0.9]	0.94
Sequential (e_2)	[-10.1, -0.8, -0.4]	0.86
Simultaneous (e_1)	[-10.1, -0.33, -1.3]	0.92
Simultaneous (e_2)	[-10.1, -0.33, -1.3]	0.92
Ground truth	[-9.8, 0.0, -0.5]	N/A

In order to further assess the developed extrinsic calibration, structures in the indoor environment observed by the camera are utilized for the qualitative analysis. In Figure 7.25, white-dotted line segments in the 2D map frame (built by the ultrasonic rangefinder) are projected onto the camera. The line segments are expected to be parallel to the edges of the parquet. In the image, one example of such an edge is highlighted with the red line segment. Further, the line segment corresponding to the wall at the end of the hallway has a known distance in the 2D map (and white-dotted line segments are selected up to this wall). If the extrinsic calibration parameters are accurate, the projection of these points onto the image would as well extend to the wall (the corresponding edge is shown as a green line in Figure 7.25 (a and b)). The projection after the extrinsic calibration can be seen in Figure 7.25 (a). Figure 7.25 (b) shows the projection without extrinsic calibration, which exhibits a large error. Even though the edges and white-dotted lines are approximately parallel, due to the larger error in the initial extrinsic calibration parameters, white points stopped before the green line segment.

In Figure 7.25 (c), the points with known coordinates in the map frame are projected onto the image. This figure shows the projected points using developed and ground truth calibration parameters in red and green, respectively.

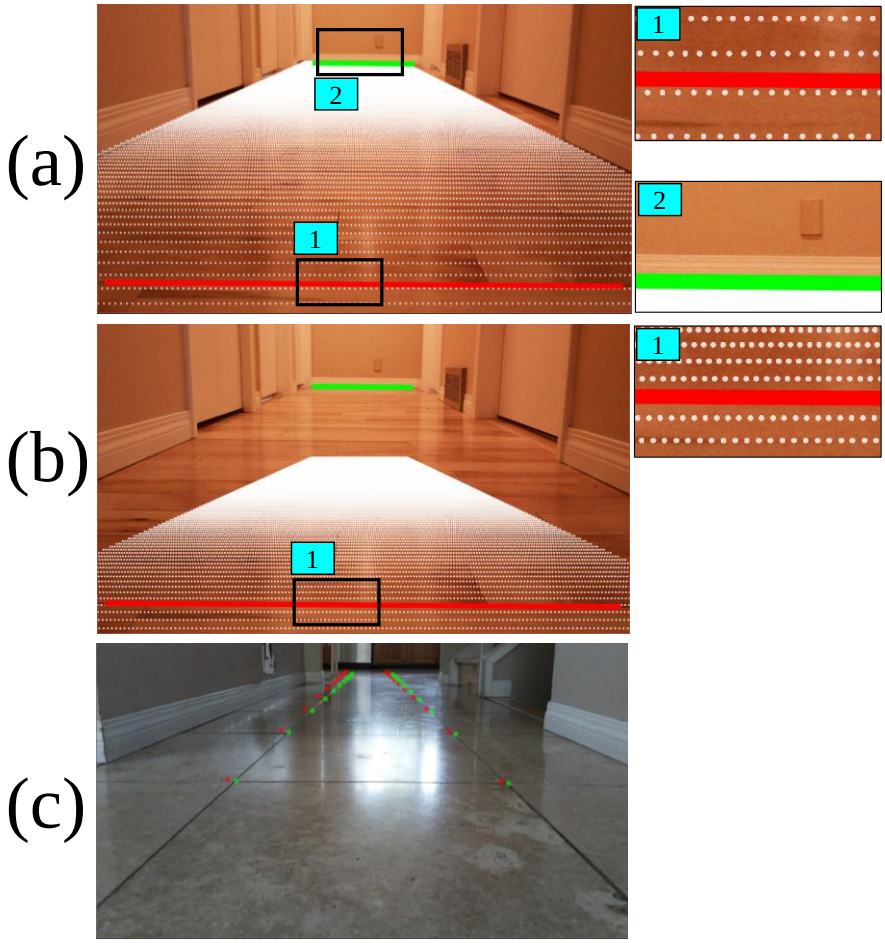


Figure 7.25: Qualitative analysis of the errors in the extrinsic calibration of the ultrasonic rangefinder and the monocular camera.

Chapter 8: Summary and Future Work

8.1 Overview of the Developed Object-level SLAM

The growing body of research in recent years has paved the path to move away from classical geometrical primitive-based solutions to more modern object-based solutions to the SLAM problem. The advent of such solutions is due to the successful replacement of classical object tracking and object detection with more accurate and robust DL-based alternatives. Using objects to solve the SLAM problem has several advantages over the classical methods. Object-level solutions are more suitable for intelligent interactions of the robot with its environment. Such intelligent interactions can depend on tasks such as finding, grabbing, and moving objects. Some modern-day systems depend on such tasks. These include robots in warehouses, robots in assembly lines, and more. Besides these robots, object-level approaches offer an alternative solution to the SLAM problem used in many handheld devices such as smartphones.

In recent decades, the RGB camera has been the most important sensor for many solutions to SLAM. This sensor can offer an abundance of radiometric information about a robot's surroundings (or a device), and many lower-cost devices and robots include this sensor nowadays. Unfortunately, relying on a single sensor to perform a complicated task such as object-level SLAM is very difficult. Therefore, in many state-of-the-art solutions, data fusion has become an important topic. Data fusion can improve accuracy by taking advantage of observations received from different types of sensors, such as IMUs. Similar to monocular cameras, IMUs are integrated into most navigation systems.

In this thesis, a solution to the SLAM problem is developed using the fusion of a monocular camera, an IMU and a 2D rangefinder. This solution is one of the first to implement RBPF to address object-level SLAM. RBPF is a type of PF; therefore, there is no requirement for the linearization of the observation and motion models. Furthermore, the Gaussian error assumption is not necessary for this type of DBN as well. These two advantages of RBPF are very important in the context of object-level SLAM.

For example, the pose uncertainties of the symmetrical objects and uncertainties that arise due to object occlusions can best be represented using non-gaussian distributions under most circumstances. The developed method also relies on the silhouette of the object rather than features on the surface of the object (the silhouette is closely related to the shape). Thus, objects that lack texture on their surface can also be incorporated into the solution. Object-level SLAM is based on many induvial components. These components are integrated into a mapping and localization framework. In the following, a summary of the gaps and the obtained results using the developed object-level framework as well as individual components, are provided.

8.2 Summary of the Main Results (Object-level Framework)

The current solutions to object-level SLAM can be categorized into two groups. The first group uses semantic knowledge derived from objects to add constraints to the SLAM problem. Such constraints can help the solution by, for example, identifying false data associations. However, this category is an extension of the classical solution to SLAM and cannot be used to help the robot to perform object-level tasks. The second category of solutions is based on directly building the map and localizing sensors with respect to the objects. The map includes objects (with their 6DoF) and thus can be used to perform object-level tasks. Unfortunately, the state-of-the-art techniques in the second category rely on an initial estimation of the map and the trajectory using classical methods (such as ORB-SLAM) and suffer from the disadvantages associated with such methods.

In this thesis, a novel object-level RBPF-based solution to the SLAM problem using the fusion of an IMU and monocular camera is developed. RBPF-SLAM includes three main steps: particle proposal, particle weighting, and particle resampling. State-of-the-art object-level solutions often rely on heuristic motion estimation (such as constant velocity) for particle proposal; however, the developed approach utilizes IMU mechanization, which allows for more accurate motion estimation in a short time. Further,

the landmarks (here objects) are initialized in the map in an undelayed fashion in the developed solution. The undelayed initialization will provide the possibility of immediate weight update and resampling. Without such initialization, the trajectory should rely on a standalone IMU solution (for the initialization period), which will often result in large errors.

In this thesis, two novel particle weighting approaches are developed. In the tightly coupled approach, particles are weighted based on observation likelihood defined using IoU (of the predicted and observed object boundary). This approach does not require point-to-point matching and thus does not suffer from false correspondence typical of such methods. Further, the tightly coupled method does not require direct pose estimation of the objects. The loosely coupled approach provides a less accurate but computationally more efficient alternative to the tightly coupled approach. The particles are weighted using a technique that does not depend on distance measurement from the object. This technique is important since obtaining such distances using a monocular camera is not possible. After extensive studies of different aspects of the developed methods, the following results are obtained using the differential-drive robot:

1. The tightly coupled method achieved an error of approximately 12.7 to 46.2 cm using EV for the position estimation (the ratio of the error to the total path length is from 0.03 to 0.06). The error of approximately 27.7 to 46.9 cm is obtained using MAP for the position estimation (the ratio of the error to the total path length is from 0.04 to 0.07). The tightly coupled method achieves a high IoU between 0.76 to 0.89. Failure in weight update has rarely been encountered using tightly coupled methods (the failure rate is in most experiments 0). Based on the experiment, the length of the trajectory and, occlusions, and illumination conditions did not significantly impact the performance.

2. The loosely coupled method achieves an error of approximately 10.1 to 41.9 cm using EV for the position estimation (the ratio of the error to the total path length is from 0.01 to 0.11). The error of approximately 19.6 to 92.4 cm is obtained using MAP for the position estimation (the ratio of the error to the total path length is from 0.05 to 0.13). The loosely coupled method achieves IoU between 0.44 to 0.73. The failure in weight update for the loosely coupled method is between 0.20 to 0.61. Overall, based on IoU, and failure rate, the loosely coupled method performs worse than the tightly coupled method. Also, the detailed analysis of the robot's trajectory confirms this. However, the runtime of loosely coupled is much less than that of tightly coupled. Based on the experiment, the trajectory length, occlusions, and illumination conditions did not significantly impact the performance.
3. The number of particles has an impact on the accuracy of the solution for both tightly coupled and loosely coupled methods. For the tightly coupled approach, based on the results, it seems increasing particles above 12500 does not introduce any significant reduction in the errors in the position. The improvements in the IoU are also small. The loosely coupled approach seems to have the best solution for approximately the same number of particles.

The following results are obtained using the handheld device:

1. The tightly coupled method achieves an error of approximately 4.1 to 13.1 cm using EV for the position estimation. The ratio of the error to the total path length is from 0.005 to 0.028. The error of approximately 2.8 to 35.7cm is obtained using MAP for the position estimation. The ratio of the error to the total path length is from 0.006 to 0.052. The tightly coupled method achieves a high IoU between 0.805 to 0.821. In our experiment, no failure rate has been observed for the tightly coupled method using a handheld device.

2. The loosely coupled method achieves an error of approximately 11.5 to 170.9 cm using EV for the position estimation. The ratio of the error to the total path length is from 0.024 to 0.426. The error of approximately 6.1 to 17.2cm is obtained using MAP for the position estimation. The ratio of the error to the total path length is from 0.015 to 0.060. Overall, based on IoU, the failure rate of the loosely coupled method is worse than the tightly coupled method. However, the runtime of the loosely coupled is much less than the tightly coupled method.
3. Based on the experiments, increasing the number of particles from 5000 to 19000 did not significantly improve the performance of the tightly coupled. Therefore, for the handheld device, it is recommended to use 5000-6000 particles.

One of the challenges of the object-level SLAM problem is object initialization. As mentioned, initialization without a rangefinder (which was the case for the previously summarized experiments) can only be achieved by using a larger uncertainty. In this thesis, a novel fusion of an ultrasonic rangefinder with a monocular camera has been developed. One challenge with the fusion of a rangefinder with a monocular camera is the fact that only a sparse set of pixels in the image can be assigned with a depth observation. Thus, such data fusion is not very advantageous for classical point-based methods. However, in the context of the developed object-level solution, the image of an object occupies many pixels, and it is highly possible that a rangefinder (such as ultrasonic) will return depth estimation corresponding to a few of the pixels belonging to the object segment. In addition, using this fusion, a new technique is developed that identifies particles with highly inaccurate poses in the early stage without a requirement to calculate the observation likelihood. This can reduce the computational cost of the algorithm substantially for the tightly coupled approach. For the fusion of the ultrasonic sensor with the monocular camera, the following results have been obtained:

1. The developed ultrasonic-based fusion approximately reduced the runtime by half. No significant improvement or degradation of accuracy was observed.
2. The developed approach of data association (which identifies if the ultrasonic sensor reading is received from the back background clutter or the object of interest) achieves high precision and recall in a wide range of values for threshold μ (between 0.4 to 0.8). This wide range shows the ease of tuning such hyperparameters in real-world applications.
3. The best result in terms of both precision and recall has been achieved when the angular resolution is assumed to be close to the nominal value (this range is between 14° to 16°, while 15° is the nominal value).

All the previously mentioned algorithms are tested in many experiments. These experiments include different illumination conditions (such as dim or bright room lights), different camera distances from the object, and different trajectory lengths. Based on these results, it seems that the developed tightly coupled method can perform well in different conditions. The abovementioned results have summarized the key findings of the developed object-level framework. These solutions depend on the success of some of its components. In the following, the results for each component are provided.

8.3 Summary of the Results of Object-segmentation, Representation and Pose Estimation.

One of the most important components in an object-level solution is the object detection/segmentation. Classical computer vision object detection/segmentation has recently been replaced with DL methods. A disadvantage of most DL-based approaches is the requirement for massive training data. In order to address this, data synthesizing is considered in the past. The hybrid method is one of many approaches for synthesizing images, which requires the least amount of human labour. In the hybrid method, only some of the aspects of the environment or the object are synthesized. In this thesis,

thousands of training images are generated using this hybrid method for object segmentation. The summary of the results is provided in the following.

1. The developed object segmentation has achieved high precision and recall in many different indoor scenes (scenes different in the level of background clutter). The highest precision and recall have been achieved using a training set of 180k images and an image resolution of 256 by 256 pixels. In five out of seven tests, a precision above 94% and a recall above 85 % have been achieved.
2. It is shown that it is very important to include realistic viewpoints, illumination conditions and image backgrounds in the synthesized training set.

Another important component of an object-level SLAM is object representation. In the past, most methods represented objects using feature points, which required texture on the objects. Other methods using an object's shape could be divided into 3D and 2D shape representations. The 3D representation cannot directly be matched to the silhouette of the object in the image (since the image provides a 2D contour of the object). In 2D methods, object shape is represented with the help of shape-prior sets. Such a set captures the shape of an object from different viewpoints as images. Unfortunately, image-based shape priors are not robust to in-plane translation, rotation, and scale variations. These variations will be required in the matching process. In order to address this issue in this thesis, the 2D contour of the object is represented using explicit parameterization. The advantage of this shape-prior set is in the following.

1. This shape-prior can be directly matched to the observed contour of the object in the image. Further, the shape-prior set is organized. Therefore providing a means for fast matching.
2. This shape-prior captures the object from a different viewpoint, thus, possessing invariance to 3D shape variations. Further in-plane rotation, in-plane translation, and scale variations can be applied easily in the matching process.
3. This shape-prior set can provide a coarse pose estimation.

The last component of an object-level SLAM is object pose estimation. The current one-step (end-to-end) DL methods require large training sets and/or cannot produce one-to-many solutions in circumstances where objects are symmetrical. For the two-step approaches, the features are detected in the first step, and in the second step, the pose is estimated with respect to these features. The pose estimation in the past is achieved with the help of a PnP algorithm. One drawback of this approach is the assumption that the salient features on the object's surface can be detected. In this thesis, in order to address this problem, a contour-based coarse-to-fine pose estimation is developed. In the coarse step, a set of closest shapes in the shape-prior set to the observed contour of the object in the image is found. In the pose refinement, the projected 3D model and the observed contour of the object are used to establish a point-to-point correspondence. Finally, the pose of the camera is solved by a PnP method. The following highlights the pose estimation results.

1. The error of the estimated pose is about 9.96° to 10.31°
2. The refined pose estimation achieves 7.96 to 8.22 pixels errors (with blunder cases included) and IoU of 0.44 to 0.67.
3. The pose estimation exhibits a certain level of robustness to the occlusions.

8.4 Summary of the Results of Extrinsic Calibration

Tightly and loosely coupled methods depend on known extrinsic calibration parameters. In the past, the boresight calibration of an IMU and a monocular camera were estimated using special devices such as turntables. Unfortunately, These types of equipment are unavailable on-site in most circumstances. In this thesis, instead of relying on special equipment, the structure of the manmade indoor environment is used to derive extrinsic parameters. The developed approach is based on finding the normal to-floor plane in the camera's frame. If this vector is assumed to be antiparallel to the gravity vector's direction, a correspondence between this vector and the gravity vector sensed by the accelerometers in an IMU unit

can be established. Finally, such correspondence can be used to find the boresight calibration parameters. The developed boresight calibration method has been tested in four indoor scenes. Further, it is compared the known methods of boresight calibration. The following results are obtained:

1. The accuracy of the parameter estimation depends on the scene and the image number. In most tests, the data sets with a larger number of images result in higher accuracy. However, as will be explained in the following, the accuracy does not always increase by adding more images. The error using 140 images is reported as 3.49°. This error is smaller than the known Kalibr method (with an accuracy of 3.80°).
2. The detection of HVP was very important in achieving such accuracy. In almost every case that the HVP is detected, VVP is also detected with very high accuracy.

The extrinsic calibration is also performed using an ultrasonic rangefinder and camera. This method is designed for the wheeled indoor robot capable of building 2D maps of the environment. Similar to the IMU and camera calibration, the floor segmentation in the image is used for the calibration. The developed method matches the back-projected pixels on the floor-segment boundary to the line segment built by the ultrasonic sensor. Based on the results, the developed sequential and simultaneous methods can both achieve a similar accuracy of less than 1°. However, the sequential approach is much faster.

8.5 Future Work and Limitations

The following limitations of this research should be addressed in the future:

- The tightly coupled RBPF-SLAM produced an accurate estimation of the trajectory. However, the computational cost of this algorithm is high. The high computational cost is mainly due to the requirement to estimate a boundary around the projected points of the object (onto the image). In the future, a possible approach that can replace this step with a faster boundary detection algorithm can significantly reduce the overall computational cost.

- The loosely coupled RBPF-SLAM has produced lower accuracy results in most scenarios compared to the tightly coupled method. One challenge with loosely coupled is the high failure rate which is directly related to the pose estimation from a single image. Therefore, improving pose estimation will be very beneficial for this method.
- The developed object representation is based on the shape of the object, and it can be used in circumstances where the objects do not have texture on their surface. Further representation is organized, which facilitates matching. Despite this, the matching process is still slow. This is because the different hypotheses should be tested in order to find the most likely in-plane rotation between the observed shape and the shapes in the shape-prior set. Each of these hypotheses requires measuring one IoU. In the future, IoU can be replaced with other methods of measuring such distance to improve the speed of the algorithm.
- The developed object pose estimation is a coarse-to-fine technique that yields accurate results. However, this estimation is not robust to severe occlusions. The main reason for this is due to the errors in the segmentation and in the coarse pose estimation (which includes matching the observed contour of the object to the shape-prior set). Thus, improving the performance of those two steps will also improve the overall accuracy of pose estimation.
- The uncertainty region for the ultrasonic sensor's readings is assumed to be spherical. However, a better approximation of this region can be considered. For example, the uncertainty in the depth (in the direction from the ultrasonic sensor to the observed point) is governed by the precision in the range, while the uncertainty in the direction perpendicular to this is governed by the angular precision (beam angle).

- The developed IMU-camera boresight calibration is designed for indoor manmade environments. One of the shortcomings of this method is the susceptibility to blunders. In the future, it is recommended to apply RANSAC-based robust estimators to address this issue.
- The developed extrinsic calibration of an ultrasonic rangefinder and a monocular is tested on a robot equipped with wheel odometry. The developed method of calibration depends on 2D line-based maps of the indoor environment built using ultrasonic rangefinders. However, this method cannot be used easily for other platforms. In the future, a 3D plane-based mapping algorithm can be developed using IMU and ultrasonic rangefinder. Such maps can be used with a similar technique to estimate the extrinsic calibration parameters.
- In the experiments, the influence of the distance of the camera from the object on the accuracy of the trajectory estimation is investigated. However, due to the difficulty of an accurate estimation of such distance from the object (while the device is moving), a qualitative assessment is provided. In the future, this distance should be measured quantitatively.

References

Abadi, I., & El-Sheimy, N. (2022). Manhattan World Constraint for Indoor Line-based Mapping Using Ultrasonic Scans. *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–8.

Ahn, S., Choi, M., Choi, J., & Chung, W. K. (2006). Data association using visual object recognition for EKF-SLAM in home environment. *IEEE International Conference on Intelligent Robots and Systems*, 2588–2594. https://doi.org/10.1109/IROS.2006.281936

Asl Sabbaghian Hokmabadi, I. (2018). *Localization on Smartphones Using Visual Fingerprinting*. Master's thesis, University of Calgary, Calgary, AB, Canada.

Asl Sabbaghian Hokmabadi, I., & El-Sheimy, N. (2022). Probabilistic Silhouette-Based Close-Range Photogrammetry Using a Novel 3d Occupancy-Based Reconstruction. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *43B2*, 343–350. https://doi.org/10.5194/isprs-archives-XLIII-B2-2022-343-2022

Avidan, S. (2004). Support Vector Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(8), 1064–1072. https://doi.org/10.1109/TPAMI.2004.53

Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixao, T. M., Mutz, F., & others. (2020). Self-driving cars: A survey. *Expert Systems with Applications*, 113816.

Bailey, T. (2003). Constrained initialisation for bearing-only SLAM. *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, *2*, 1966–1971.

Bentley, J., & Friedman, J. (1979). Data Structures for Range Searching. *ACM Computing Surveys*, *11*(4), 397–409. https://doi.org/10.1145/356789.356797

Bentley, J. L. (1979). Decomposable searching problems. *Information Processing Letters*, *8*(5), 244–251. https://doi.org/10.1016/0020-0190(79)90117-0

Berenguel-Baeta, B., Guerrero-Viu, M., Nova, A., Bermudez-Cameo, J., Perez-Yus, A., & Guerrero, J. J. (2020). Floor Extraction and Door Detection for Visually Impaired Guidance. *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 1222–1229. https://doi.org/10.1109/ICARCV50220.2020.9305464

Bernreiter, L., Gawel, A., Sommer, H., Nieto, J., Siegwart, R., & Lerma, C. C. (2019). Multiple hypothesis semantic mapping for robust data association. *IEEE Robotics and Automation Letters*, *4*(4), 3255–3262.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). Fully-convolutional siamese networks for object tracking. *European Conference on Computer Vision*, 850–865.

Bhat, G., Johnander, J., Danelljan, M., Shahbaz Khan, F., & Felsberg, M. (2018). Unveiling the power of deep tracking. *Proceedings of the European Conference on Computer Vision (ECCV)*, 483–498.

Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2544–2550. https://doi.org/10.1109/CVPR.2010.5539960

Bolme, D. S., Draper, B. A., & Beveridge, J. R. (2009). Average of synthetic exact filters. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009, 2009 IEEE*, 2105–2112. https://doi.org/10.1109/CVPRW.2009.5206701

Bowman, S. L., Atanasov, N., Daniilidis, K., & Pappas, G. J. (2017). Probabilistic data association for semantic slam. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1722–1729.

Caccamo, S., Ataer-Cansizoglu, E., & Taguchi, Y. (2017). Joint 3D reconstruction of a static scene and moving objects. *2017 International Conference on 3D Vision (3DV)*, 677–685.

Calisi, D., Farinelli, A., Iocchi, L., & Nardi, D. (2007). Autonomous exploration for search and rescue robots. *WIT Transactions on the Built Environment*, *94*. WIT Press, Southampton, UK.

Camposeco, F., & Pollefeys, M. (2015). Using vanishing points to improve visual-inertial odometry. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 5219–5225.

Caselles, V., Kimmel, R., & Sapiro, G. (1997). Geodesic Active Contours. *International Journal of Computer Vision*, *22*(1), 61–79. https://doi.org/10.1023/A:1007979827043

Castle, R. O., Klein, G., & Murray, D. W. (2010). Combining monoSLAM with object recognition for scene augmentation using a wearable camera. *Image and Vision Computing*, *28*(11), 1548–1556. https://doi.org/10.1016/j.imavis.2010.03.009

Chan, T. F., & Vese, L. A. (2001). Active contours without edges. *IEEE Transactions on Image Processing*, *10*(2), 266–277. https://doi.org/10.1109/83.902291

Chen, H., Yang, Z., Zhao, X., Weng, G., Wan, H., Luo, J., Ye, X., Zhao, Z., He, Z., Shen, Y., & Schwertfeger, S. (2020). Advanced mapping robot and high-resolution dataset. *Robotics and Autonomous Systems*, *131*, 103559. https://doi.org/10.1016/J.ROBOT.2020.103559

Chermak, L., Aouf, N., Richardson, M., & Visentin, G. (2019). Real-time smart and standalone vision/IMU navigation sensor. *Journal of Real-Time Image Processing*, *16*(4), 1189–1205. https://doi.org/10.1007/s11554-016-0613-z

Chollet, F. (2021). *Deep learning with Python*. Manning. Shelter Island, New York,

Choudhary, S., Trevor, A.J., Christensen, H.I. and Dellaert, F., (2014), September. SLAM with object discovery, modeling and mapping. *In* 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1018-1025). IEEE.

Civera, J., Gálvez-López, D., Riazuelo, L., Tardós, J. D., & Montiel, J. M. M. (2011). Towards semantic SLAM using a monocular camera. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1277–1284.

Cohen, L. D. (1991). On active contour models and balloons. *CVGIP: Image Understanding*, *53*(2), 211–218. https://doi.org/10.1016/1049-9660(91)90028-N

Cox, I. J. (1991). Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, *7*(2), 193–204.

Cremers, D. (2002). *Statistical shape knowledge in variational image segmentation.* PhD thesis. University of Mannheim.

Daum, F., & Huang, J. (2003). Curse of dimensionality and particle filters. *2003 IEEE Aerospace Conference Proceedings (Cat. No. 03TH8652)*, *4*, 4_1979–4_1993.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1052–1067.

Dellaert, F., & Kaess, M. (2006). Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, *25*(12), 1181–1203.

Deng, X., Mousavian, A., Xiang, Y., Xia, F., Bretl, T., & Fox, D. (2021). Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking. *IEEE Transactions on Robotics*, *37*(5), 1328–1342.

Dharmasiri, T., Vincent, L., & Drummond, T. (2016). *MO-SLAM: Multi Object SLAM with Run-Time Object Discovery through Duplicates*. https://doi.org/10.13140/RG.2.2.33667.50727

Doherty, K., Fourie, D., & Leonard, J. (2019). Multimodal semantic slam with probabilistic data association. *2019 International Conference on Robotics and Automation (ICRA)*, 2419–2425.

Dong, W., & Isler, V. (2018). A novel method for the extrinsic calibration of a 2D laser rangefinder and a camera. *IEEE Sensors Journal*, *18*(10), 4200–4211.

Dwibedi, D., Misra, I., & Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. *Proceedings of the IEEE International Conference on Computer Vision*, 1301–1310.

Eade, E., & Drummond, T. (2006). Scalable monocular SLAM. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *1*, 469–476. https://doi.org/10.1109/CVPR.2006.263

Eade, E., & Drummond, T. (2009). Edge landmarks in monocular SLAM. *Image and Vision Computing*, *27*(5), 588–596.

Edward A., and Shreiner D. (2012). *Interactive Computer Graphics*. 6th ed. USA: Addison-Wesley, Pearson.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, *22*(6), 46–57. https://doi.org/10.1109/2.30720

Elloumi, W., Treuillet, S., & Leconge, R. (2014). Real-time camera orientation estimation based on vanishing point tracking under Manhattan World assumption. *Journal of Real-Time Image Processing*, *13*, 669–684.

Emami, P., Pardalos, P. M., Elefteriadou, L., & Ranka, S. (2020). Machine Learning Methods for Data Association in Multi-Object Tracking. *ACM Computing Surveys*, *53*(4), 1–33. https://doi.org/10.1145/3394659

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

Foulonneau, A., Charbonnier, P., & Heitz, F. (2009). Multi-reference shape priors for active contours. *International Journal of Computer Vision*, *81*(1), 68.

Furgale, P., Rehder, J., & Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1280–1286.

Gakne, P. V., & O'Keefe, K. (2017). Monocular-based pose estimation using vanishing points for indoor image correction. *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–7.

Gallagher, A. C., & C., A. (2005). Using Vanishing Points To Correct Camera Rotation In Images. *The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, 460–467. https://doi.org/10.1109/CRV.2005.84

Gálvez-López, D., Salas, M., Tardós, J. D., & Montiel, J. M. M. (2016). Real-time monocular object SLAM. *Robotics and Autonomous Systems*, *75*, 435–449. https://doi.org/10.1016/j.robot.2015.08.009

Gao, X.-S., Hou, X.-R., Tang, J., & Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(8), 930–943.

Georgakis, G., Mousavian, A., Berg, A. C., & Kosecka, J. (2017). Synthesizing training data for object detection in indoor scenes. *ArXiv Preprint ArXiv:1702.07836*.

Gomez-Ojeda, R., Briales, J., Fernandez-Moral, E., & Gonzalez-Jimenez, J. (2015). Extrinsic calibration of a 2D laser-rangefinder and a camera based on scene corners. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 3611–3616.

Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113.

Gouiaa, R., & Meunier, J. (2014, May). 3D Reconstruction by Fusioning Shadow and Silhouette Information. *Proceedings - Conference on Computer and Robot Vision, CRV 2014*. https://doi.org/10.1109/CRV.2014.58

Grisetti, G., Stachniss, C., & Burgard, W. (2005). Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2432–2437.

Harris, C., & Stephens, M. (1988). A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, 23.1-23.6. https://doi.org/10.5244/C.2.23

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

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press. doi:10.1017/CBO9780511811685

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 2961–2969.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., & Navab, N. (2013). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, 548–562.

Hosseinzadeh, M., Latif, Y., Pham, T., Suenderhauf, N., & Reid, I. (2018). Structure aware SLAM using quadrics and planes. *Asian Conference on Computer Vision*, 410–426.

Hu, Y., Fua, P., Wang, W., & Salzmann, M. (2020). Single-stage 6d object pose estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2930–2939.

Huang, W., & Liu, H. (2018). Online initialization and automatic camera-IMU extrinsic calibration for monocular visual-inertial SLAM. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 5182–5189.

Huang, Z., Gu, N., Lin, C., Shen, J., & Chang, J. (2018). Real time vanishing points detection on smartphones under Manhattan world assumption. *Pattern Recognition Letters*, *115*, 117–127.

Ivanov, R. (2021). Accuracy analysis of BLE beacon-based localization in smart buildings. *Journal of Ambient Intelligence and Smart Environments*, *13*(4), 325–344.

Jensfelt, P., Ekvall, S., Kragic, D., & Aarno, D. (2006). Augmenting slam with object detection in a service robot framework. *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 741–746.

Jin, Z., Li, Y., Yang, Z., Zhang, Y., & Cheng, Z. (2023). Real-Time Indoor Positioning Based on BLE Beacons and Pedestrian Dead Reckoning for Smartphones. *Applied Sciences*, *13*(7), 4415.

Joshi, N., Sharma, Y., Parkhiya, P., Khawad, R., Krishna, K. M., & Bhowmick, B. (2018). Integrating objects into monocular slam: Line based category specific models. *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, 1–9.

Kaess, M. (2015). Simultaneous localization and mapping with infinite planes. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 4605–4611.

Kassir, A., & Peynot, T. (2010). Reliable automatic camera-laser calibration. *Australasian Conference on Robotics and Automation*, *2010*.

Kelly, J., & Sukhatme, G. S. (2009). Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA)*, 360–368.

Khalid, S. (2012). Incremental indexing and retrieval mechanism for scalable and robust shape matching. *Multimedia Systems*, *18*(4), 319–336.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 225–234. https://doi.org/10.1109/ISMAR.2007.4538852

Kwok, N. M., & Dissanayake, G. (2003). Bearing-only SLAM in indoor environments using a modified particle filter. *In Proceedings of the Australian Conference on Robotics and Automation*. http://scholar.google.com/scholar?cluster=15299272901741552295&hl=ar&as_sdt=0,5

Kwok, N. M., & Dissanayake, G. (2004). An efficient multiple hypothesis filter for bearing-only SLAM. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, *1*, 736–741.

Kyriacou, T., Bugmann, G., & Lauria, S. (2005). Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems*, *51*(1), 69–80. https://doi.org/10.1016/j.robot.2004.08.011

Lazaros, N., Sirakoulis, G. C., & Gasteratos, A. (2008). Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics*, *2*(4), 435–462.

Leonard, J. J., Durrant-Whyte, H. F., & Cox, I. J. (1992). Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, *11*(4), 286–298. https://doi.org/10.1177/027836499201100402

Lewis, J. P. (2010). Fast normalized cross-correlation, 1995. *Vision Interface*, *2010*, 120–123.

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High Performance Visual Tracking with Siamese Region Proposal Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8971–8980. https://doi.org/10.1109/CVPR.2018.00935

Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, *11*(3), 283–318. https://doi.org/10.1007/BF01469346

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37.

Lobo, J., & Dias, J. (2007). Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, *26*(6), 561–575.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440.

Lowe, D. G. (2004). Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–11020042. https://doi.org/http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

Ma, L., & Sibley, G. (2014). Unsupervised dense object discovery, detection, tracking and reconstruction. *European Conference on Computer Vision*, 80–95.

MacKay, C. T., & Moh, T.-S. (2021). Learning for Free: Object Detectors Trained on Synthetic Data. *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 1–8. https://doi.org/10.1109/IMCOM51814.2021.9377353

Matthews, I., Ishikawa, T., & Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 810–815. https://doi.org/10.1109/TPAMI.2004.16

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., Huertas, A., Stein, A., & Angelova, A. (2007). Computer Vision on Mars. *International Journal of Computer Vision*, *75*(1), 67–92. https://doi.org/10.1007/s11263-007-0046-z

Merrill, N., Guo, Y., Zuo, X., Huang, X., Leutenegger, S., Peng, X., Ren, L., & Huang, G. (2022). Symmetry and uncertainty-aware object slam for 6dof object pose estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14901–14910.

Montemerlo, M., & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, *2*(July), 1985–1991. https://doi.org/10.1109/robot.2003.1241885

Montemerlo, M., Thrun, S., Roller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *IJCAI International Joint Conference on Artificial Intelligence*, 1151–1156.

Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 3565–3572.

Mungúia, R., Castillo-Toledo, B., & Grau, A. (2013). A robust approach for a filter-based monocular simultaneous localization and mappng (SLAM) system. *Sensors (Switzerland)*, *13*(7), 8501–8522. https://doi.org/10.3390/s130708501

Mungu'ia, R., & Grau, A. (2012). Monocular SLAM for visual odometry: A full approach to the delayed inverse-depth feature initialization method. *Mathematical Problems in Engineering*, *2012*.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, *31*(5), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, *33*(5), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

Murphy, K., & Russell, S. (2001). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. *Sequential Monte Carlo Methods in Practice*, 499–515. https://doi.org/10.1007/978-1-4757-3437-9_24

Nettleton, E. W., Durrant-Whyte, H. F., Gibbens, P. W., & Göktogan, A. H. (2000). Multiple-platform localization and map building. In *Sensor fusion and decentralized control in robotic systems III*, international society for optics and photonics.Vol 4196, pp. 337– 347.

Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*, 2320–2327. https://doi.org/10.1109/ICCV.2011.6126513

Nguyen, X.-D., You, B.-J., & Oh, S.-R. (2008). A simple framework for indoor monocular SLAM. *International Journal of Control, Automation, and Systems*, *6*(1), 62–75.

Nicholson, L., Milford, M., & Sunderhauf, N. (2019). QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robotics and Automation Letters*, *4*(1), 1–8. https://doi.org/10.1109/LRA.2018.2866205

Nistér, D., Naroditsky, O., & Bergen, J. (2004). Visual odometry. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *1*(C). https://doi.org/10.1109/cvpr.2004.1315094

Noureldin, A., Karamat, T. B., & Georgy, J. (2012). *Fundamentals of inertial navigation, satellite-based positioning and their integration*. Springer Science & Business Media. Berlin/Heidelberg, Germany.

Ok, K., Liu, K., Frey, K., How, J. P., & Roy, N. (2019). Robust object-based SLAM for high-speed autonomous navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, *2019-May*, 669–675. https://doi.org/10.1109/ICRA.2019.8794344

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*, 3400–3407. https://doi.org/10.1109/ICRA.2011.5979561

Oron, S., Bar-Hille, A., & Avidan, S. (2014). Extended Lucas-Kanade tracking. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8693 LNCS*(PART 5), 142–156. https://doi.org/10.1007/978-3-319-10602-1_10

Parkhiya, P., Khawad, R., Murthy, J. K., Bhowmick, B., & Krishna, K. M. (2018). Constructing category-specific models for monocular object-slam. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 4517–4524.

Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., & Daniilidis, K. (2017). 6-dof object pose from semantic keypoints. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2011–2018.

Peasley, B., Birchfield, S., Cunningham, A., & Dellaert, F. (2012). Accurate on-line 3D occupancy grids using Manhattan world constraints. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5283–5290.

Petovello, M. G. (2003). *Real-time integration of a tactical-grade IMU and GPS for high-accuracy positioning and navigation*. Ph.D. thesis, University of Calgary, Calgary, Canada.

Pillai, S., & Leonard, J. J. (2015). Monocular SLAM supported object recognition. *Robotics: Science and Systems*, *11*. https://doi.org/10.15607/RSS.2015.XI.034

Pinto, L., Forlani, G., & others. (2002). A single step calibration procedure for IMU/GPS in aerial photogrammetry. *International Archives of Photogrammetry and Remote Sensing*, *34*(B3), 210–213.

Posada, L. F., Narayanan, K. K., Hoffmann, F., & Bertram, T. (2010). Floor segmentation of omnidirectional images for mobile robot visual navigation. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 804–809. https://doi.org/10.1109/IROS.2010.5652869

Prisacariu, V. A., Kähler, O., Murray, D. W., & Reid, I. D. (2013). Simultaneous 3D tracking and reconstruction on a mobile phone. *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 89–98.

Qian, Z., Kartik P., Jie F., and Jing X. (2021). "Semantic slam with autonomous object-level data association." In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11203-11209. IEEE.

Rad, M., & Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *Proceedings of the IEEE International Conference on Computer Vision*, 3828–3836.

Rasmussen, I., Kvalsvik, S., Andersen, P.-A., Aune, T. N., & Hagen, D. (2022). Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine. *Applied Sciences*, *12*(17). https://doi.org/10.3390/app12178534

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 234–241.

Rosenhahn, B., Brox, T., & Weickert, J. (2007). Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, *73*(3), 243–262.

Rother, C., Kolmogorov, V., & Blake, A. (2004). " GrabCut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, *23*(3), 309–314.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., & Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1352–1359.

Saleh, F. S., Aliakbarian, M. S., Salzmann, M., Petersson, L., & Alvarez, J. M. (2018). Effective use of synthetic data for urban scene semantic segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 84–100.

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Silpa-Anan, C., & Hartley, R. (2008). Optimised KD-trees for fast image descriptor matching. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. https://doi.org/10.1109/CVPR.2008.4587638

Smith, P., Reid, I. D., & Davison, A. J. (2006). Real-time monocular SLAM with straight lines. *Proc 17th British Machine Vision Conference*, Edinburgh, Sept 2006

Smith, R. C., & Cheeseman, P. (1986). On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, *5*(4), 56–68. https://doi.org/10.1177/027836498600500404

Solà, J., Monin, A., Devy, M., & Lemaire, T. (2005). Undelayed initialization in bearing only SLAM. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2751–2756. https://doi.org/10.1109/IROS.2005.1545392

Song, Y., Zhang, Z., Wu, J., Wang, Y., Zhao, L., & Huang, S. (2021). A right invariant extended kalman filter for object based slam. *IEEE Robotics and Automation Letters*, *7*(2), 1316–1323.

Strickland, W. H., & King, R. H. (1993). *Characteristics of the ultrasonic ranging sensors in an underground environment*. Pittsburgh, PA (United States); Bureau of Mines. https://www.osti.gov/biblio/6128933

Sundermeyer, M., Marton, Z.-C., Durner, M., Brucker, M., & Triebel, R. (2018). Implicit 3d orientation learning for 6d object detection from rgb images. *Proceedings of the European Conference on Computer Vision (ECCV)*, 699–715.

Terzopoulos, D., & Szeliski, R. (1993). Tracking with Kalman Snakes. In *Active Vision* (pp. 3–20). MIT Press.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, *128*(1–2), 99–141. https://doi.org/10.1016/S0004-3702(01)00069-8

Torr, P. H. S., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, *78*(1), 138–156.

Tran, T.-T., Pham, V.-T., & Shyu, K.-K. (2013). Moment-based alignment for shape prior with variational B-spline level set. *Machine Vision and Applications*, *24*, 1075–1091.

Treiber, M. A. (2010). *An introduction to object recognition: selected algorithms for a wide variety of applications*. Springer Science & Business Media.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 969–977.

Tsai, A., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, W. E., & Willsky, A. (2003). A shape-based approach to the segmentation of medical imagery using level sets. *IEEE Transactions on Medical Imaging*, *22*(2), 137–154.

Unnikrishnan, R., & Hebert, M. (2005). Fast extrinsic calibration of a laser rangefinder to a camera. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09*.

van de Weijer, J., Gevers, T., & Gijsenij, A. (2007). Edge-Based Color Constancy. *IEEE Transactions on Image Processing*, *16*(9), 2207–2214. https://doi.org/10.1109/TIP.2007.901808

Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., & Randall, G. (2012). LSD: a line segment detector. *Image Processing On Line*, *2*, 35–55.

Wang, J., Rünz, M., & Agapito, L. (2021). DSP-SLAM: Object oriented SLAM with deep shape priors. *2021 International Conference on 3D Vision (3DV)*, 1362–1371.

Woods, E. R., & Gonzalez, C. R. (2008). *Digital image processing*. Pearson Education Ltd. Upper Saddle River, NJ, USA

Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *ArXiv Preprint ArXiv:1711.00199*.

Yang, S., & Scherer, S. (2019). CubeSLAM: Monocular 3-D Object SLAM. *IEEE Transactions on Robotics*, *35*(4), 925–938. https://doi.org/10.1109/TRO.2019.2909168

Zakharov, S., Shugurov, I., & Ilic, S. (2019). Dpod: 6d pose object detector and refiner. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1941–1950.

Zhang, H., Hu, S., & Zhang, X. (2014). SIFT flow for large-displacement object tracking. *Appl. Opt.*, *53*(27), 6194–6205. https://doi.org/10.1364/AO.53.006194

Zhang, Q., & Pless, R. (2004). Extrinsic calibration of a camera and laser range finder (improves camera calibration). *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, *3*, 2301–2306.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334.

Zhou, J., & Li, B. (2006). Robust Ground Plane Detection with Normalized Homography in Monocular Sequences from a Robot Platform. *2006 International Conference on Image Processing*, 3017–3020. https://doi.org/10.1109/ICIP.2006.312972

Appendices

Appendix A: Discretization of State-space Equations

Equation A.1 holds for the derivative of an exponential. Further, if it is assumed that the solution to the differential equation in Equation 3.28.1 has the form e^{At}, by multiplying both sides in Equation 3.28.1 to e^{-At}, Equation A.2 is obtainable.

$$\frac{de^{At}}{dt} = Ae^{At} = e^{At}A \tag{A.1}$$

$$e^{-A_c t}\dot{\xi} = e^{-A_c t}A_c\xi + e^{-A_c t}B_c u \;\; \rightarrow \; e^{-A_c t}\dot{\xi} - e^{-A_c t}A_c\xi = e^{-A_c t}B_c u \; \rightarrow \frac{d(e^{-A_c t}\xi)}{dt} = e^{-A_c t}B_c u \tag{A.2}$$

In Equation A.2, the right-hand side equation is derived using Equation A.1 and linearity of the derivation operator. The integral of Equation A.2 (right-most equation) is calculated in Equation A.3.

$$\int_0^T \frac{d(e^{-A_c\, t}\,\xi)}{dt}\,dt = \int_0^T e^{-A_c t}B_c u\,dt \qquad \rightarrow \qquad e^{-A_c T}\xi(T) - X(0) = \int_0^T e^{-A_c t}B_c u\,dt \tag{A.3}$$

Equation A.3 (rightmost equation) can be reorganized as:

$$\xi(T) = e^{A_c T}(\,\xi(0) + \int_0^T e^{-A_c t}B_c u\,dt) \;\rightarrow \xi(T) = e^{A_c T}\xi(0) + \int_0^T e^{-A_c(t-T)}B_c u\,dt \tag{A.4}$$

If $T = k\Delta T$ where ΔT is the step length and k is an index, then Equation A.4 is expanded to Equation A.5.

$$\xi_k = \; e^{A_c\, k\Delta T}\xi(0) + \int_0^{k\Delta T} e^{-A_c(t-k\Delta T)}B_c u\,dt \tag{A.5}$$

The future step X_{k+1} follows Equation A.5 and is shown in Equation A.6 and reorganized in Equation A.7.

$$\xi_{k+1} = e^{A_c(k+1)\Delta T}\xi(0) + \int_0^{(k+1)\Delta T} e^{-A_c(t-(k+1)\Delta T)}B_c u\,dt \tag{A.6}$$

$$\xi_{k+1} = e^{A_c\,\Delta T}[e^{A_c k\Delta T}\xi(0) + \int_0^{k\Delta T} e^{-A_c(t-k\Delta T)}B_c u\,dt] + \int_k^{(k+1)\Delta T} e^{-A_c(t-(k+1)\Delta T)}B_c u\,dt \qquad \text{(A.7)}$$

The expression in the bracket in the A.7 is Equation A.5, and Equation A.7 simplifies to Equation A.8.

$$\xi_{k+1} = e^{A_c\,\Delta T}\xi_k + \int_k^{(k+1)\Delta T} e^{-A_c(t-(k+1)\Delta T)}B_c u\,dt \qquad \text{(A.8)}$$

The assumptions of the constant B_c and u (for an interval) can be invoked for Equation A.8. Since these values are constant can be moved out of the integral operator. Furthermore, since A_c is zero, Equation A.8 can be simplified to 3.30.1

Appendix B: Infrared Beacons

B.1 Overview of the Beacon System

This appendix explains the positioning system developed to estimate the ground truth for some of the experiments in Chapter 7. The system of beacons named Indoor Bearing Only Beacons (IBOB) includes a receiver and several transmitters. This positioning system is based on the Angle of Arrival (AoA) method. Many classical AoA-based positioning systems measure the direction from the transmitter to the receiver. This approach can only provide the position and not the receiver's heading. IBOB measures the angle from the receiver to the transmitters (the transmitter sends omnidirectional signals). The approach will allow measuring the heading of the receiver as well. Section B.2 provides the mathematical derivation leading to the position estimation with the help of the beacons.

The developed beacons can provide approximately a position estimation with an error of 3 to 11 cm, depending on how the transmitters and the receivers are located with respect to each other in the environment. This accuracy is relatively high in comparison to the state-of-the-art Bluetooth-based beacons. However, this system can only provide a position estimation if the receiver is in the transmitters' Line of Sight (LoS). Further, this system is designed for the indoor environment where the interference from the sunlight is low.

IBOB transmitters itself include three sub-components: a microcontroller, a circuit that includes a 555 Integrated Circuit (IC) timer, and an array of IR Light Emitting Diodes (LED). A microcontroller is a device that can run simple programs, often in a loop. Here ESP8266 is used as a low-cost microcontroller which has the capability to send and receive data from the user through Wi-Fi. The microcontroller is programmed to trigger the timer circuit. Once the timer circuit is triggered, it creates a modulated IR signal at about 38kHz. The IR LED is a TSAL6400 (Vishay©), compatible with the TSOP32538 (Vishay©) sensor installed on the receiver. The distance range of the signal is about 45 meters. The beacons can

transmit at a large angle, depending on the number of IR LEDs in the array (photos of the components of the transmitter are provided in Chapter 6).

The second component of IBOB is the receiver. The receiver includes an IR sensor which is concealed inside a box. This box has a narrow opening that blocks the light in all directions except the opening (including modulated 38kHz IR). This box has been mounted on an encoder disk that can rotate with the help of a servo motor (SG90). The encoder disk is built using 3D printing technology. While the servo motor causes the platform to rotate, the encoder wheel can block the photo-interrupter. If such a blocking happens, the photo-interrupter produces a high voltage (and produces a low voltage otherwise). By counting the number of high and low voltages and knowing exactly the step size of the encoder's pitch (about 6°), the angle of rotation can be easily calculated. The receiver module is shown in Figure B.1 on the robot (designated inside the circle). A closer look at the sub-components inside the receiver is shown in Figure B.2.

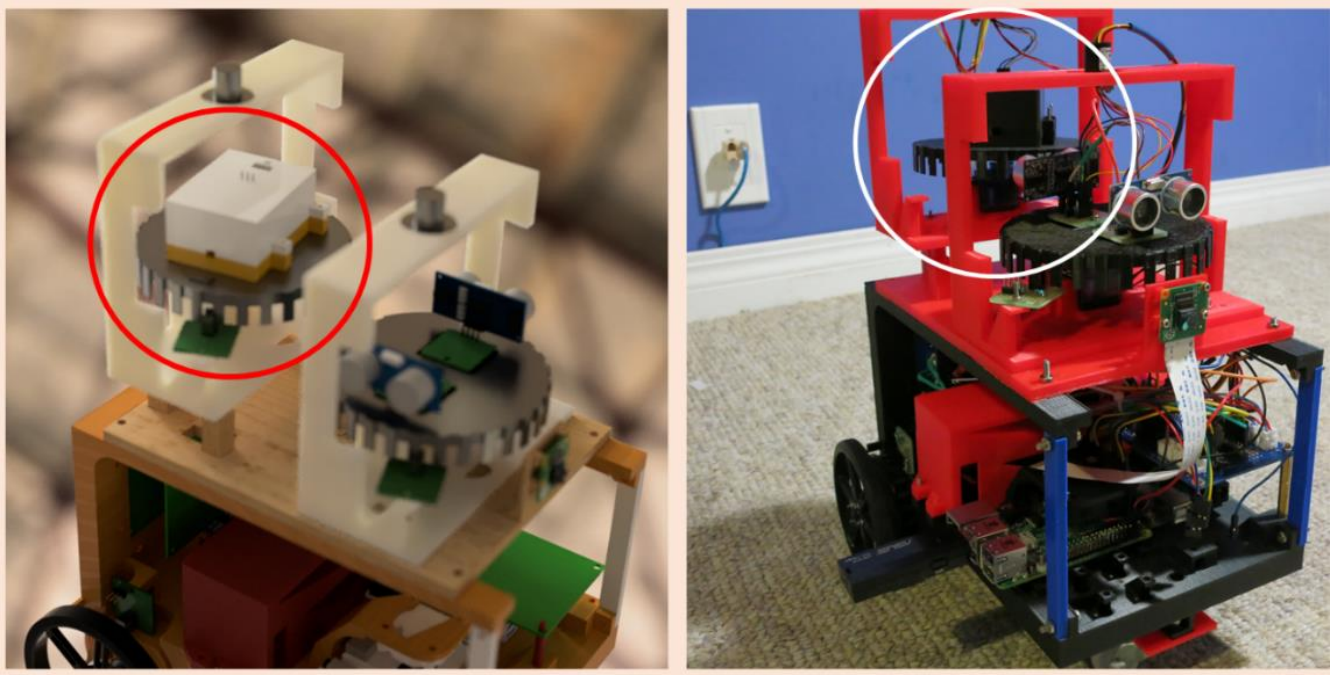


Figure B.9.1: The 3D CAD model (left) and a picture of the robot(right). The receiver component is shown inside the circle.

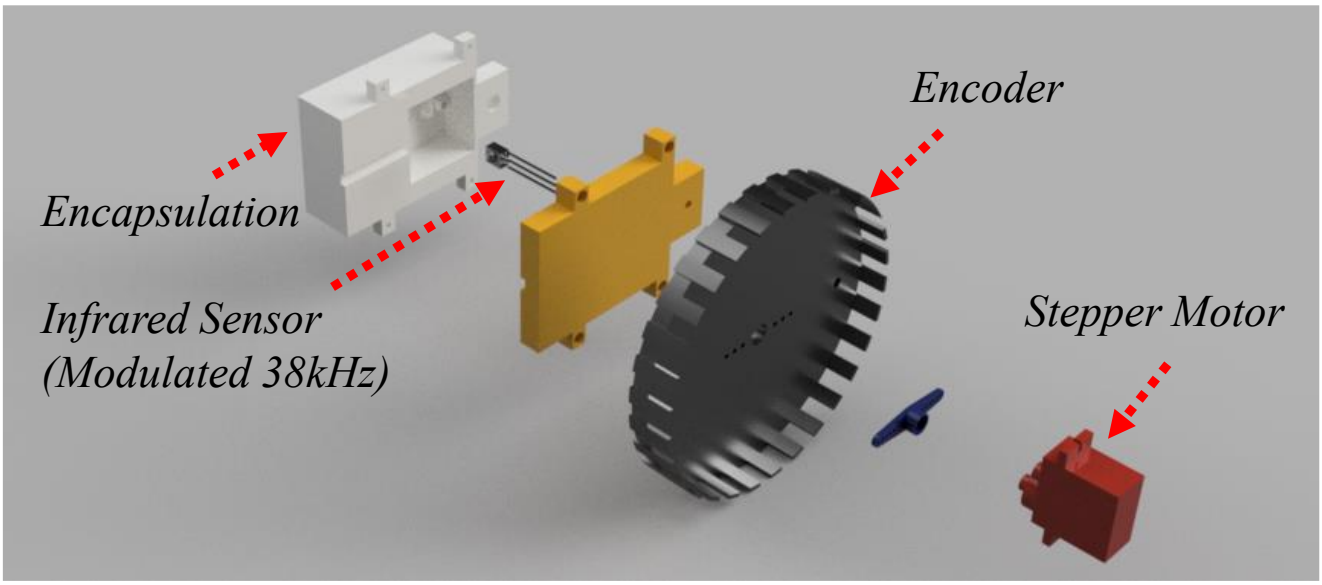


Figure B.9.2: The building components of the IBOB receiver.

After every step, the servo motor stops very shortly to let the sensor detect any incoming modulated IR signals. If an IR signal is received, this will be recorded as a binary value of 1 in an array. Otherwise, a 0 will be recorded.

This encoder platform has a default pose denoted as the zero position. The zero position corresponds to the largest pitch on the encoder platform. The pitch can be identified by the microcontroller (connected to the photo-interrupter), which measures the time the photo-interrupter was blocked. The zero position can be identified since the corresponding pitch is larger than the others. Thus, the photo-interrupter will be blocked for a longer time. Once the encoder completes one rotation, it will provide a binary array as the output. The following known information is used to estimate the angles from the robot to each beacon using the binary array (the flowchart of the explained process is provided in B.3):

- Each binary number in the array has a known angle from the zero position.
- The relative translation and the rotation of the receiver in the zero position from the robot coordinate frame are known.

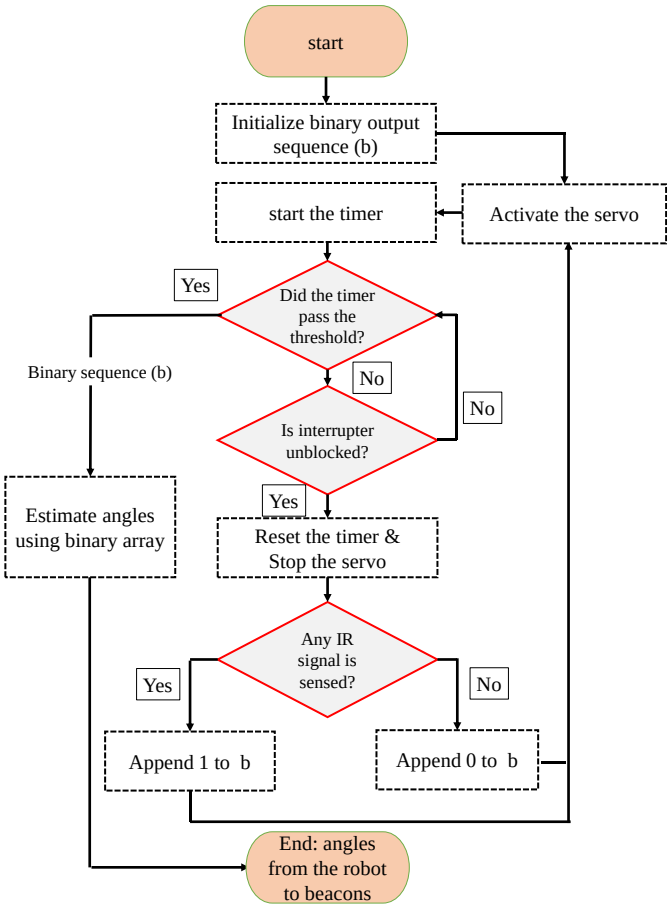


Figure B.9.3: This flowchart shows how the receiver records a binary array corresponding to orientation observations.

B.2 Methodology

In this section, it will be explained how the position can be estimated using the observed angles. The beacon-based system provides a set of angles from each beacon to the robot, as shown in the simple schematic in Figure B.4. The observed angle can be mathematically written as Equation B.1. In this Equation, the observed angle, the position of the beacon (in an inertial frame), the position of the robot (in an inertial frame), the heading of the robot (in an inertial frame) are denoted as z_b, $\left[x_b^i, y_b^i\right]'$, $\left[x^i, y^i\right]'$, and θ^i, respectively. The goal is to obtain the unknowns ($\left[x^i, y^i, \theta^i\right]'$) using the known position of the beacons $\left[x_b^i, y_b^i\right]'$ and observed angle (z_b).

$$z_b = atan2\left(y^i - y_b^i, x^i - x_b^i\right) - \theta^i \tag{B.1}$$

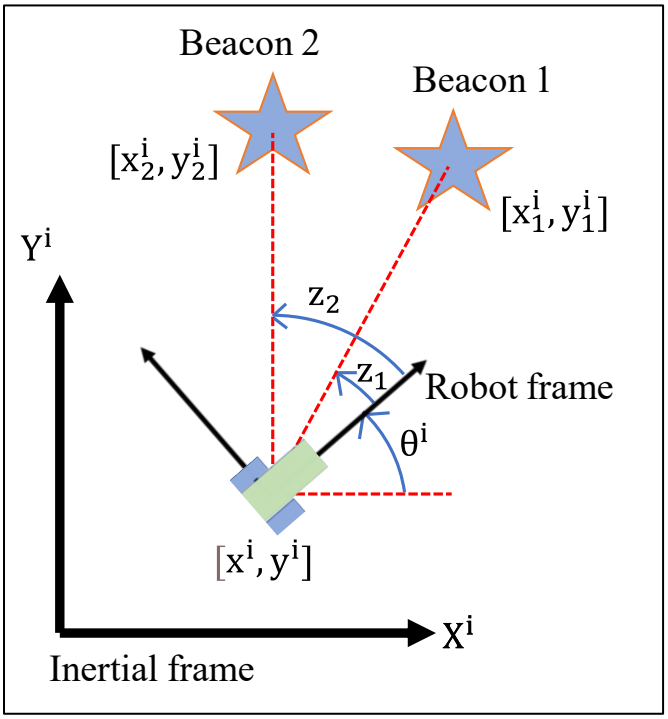


Figure B.9.4: Schematic of the observed angles and how it is related to the robot's pose.

Since the number of beacons is often more than the unknowns (here, there are three unknowns), the exact solution to this problem, in general, does not exist. In these circumstances, a classical approach is to rely on least-square estimation, which requires linearization of Equation B.1. Linearized equations can also help provide the covariance of the estimated position through the known error propagation techniques. One issue with the least square method is that it requires initialization; more importantly, this approach cannot provide a global uncertainty estimation of the robot's pose.

Due to the two problems mentioned above, a novel method for position estimation is developed in this appendix. This approach evaluates a function that approximates the observation likelihood globally. Further, this function is robust to errors in the form of bias in the observations.

If the map of the environment is represented as a two-dimensional grid, the robot can assume a finite set of possible positions. If the position of the robot and the observation are known, it is possible to estimate the robot's heading. The estimated heading is shown in Equation B.2, which is just a simple rearrangement of Equation B.1. The error term (ε) represents a bias in the observations. Such bias can happen due to issues with the zero-positioning of the platform. Also, a subscript b is used to denote the heading(θ_b) estimated using z_b and $\left[x_b^i, y_b^i\right]'$.

$$\theta_b^i = atan2\left(y^i - y_b^i, x^i - x_b^i\right) - z_b + \varepsilon \tag{B.2}$$

The orientation shown in Equation B.2 is in an inertial frame, and if there is more than one beacon available, it is possible to measure the difference (d_{jk}) as shown in Equation B.3 (where $j \neq k$). Here it is assumed that the θ is in the range from - π to π radians. If the total number of available beacons is denoted as I, the following error function can be defined in Equation B.4. The value of this function can either be 0 or a positive value. The goal is to minimize Equation B.4. Ideally, if the observations did not have noise and the robot was situated in one of the corners of the defined grid, all the orientations would agree and thus, the zero of the function $e\left(x^i, y^i\right)$ can be attained. However, in practice, the noise and distortions can contaminate the observation. One characteristic of Equation B.4 is that it is not affected by the bias in the observations (ε), as it will be cancelled out in calculating d_{jk}.

$$\mathrm{d}_{jk} = \begin{cases} \left|\theta_j^i - \theta_k^i\right|, & \left|\theta_j^i - \theta_k^i\right| \leq \pi \\ 2\pi - \left|\theta_j^i - \theta_k^i\right|, & \left|\theta_j^i - \theta_k^i\right| > \pi \end{cases} \tag{B.3}$$

$$e(x^i, y^i) = \sum_{\substack{k \neq j \\ j,k \in I}} \mathrm{d}_{jk} \tag{B.4}$$

A second characteristic of Equation B.4 is that it can be evaluated for a given cell in the map and observations without prior information about the heading. Since it was assumed that the map has a finite

number of cells, this function can be evaluated for every cell. Figure B.5 illustrates an example where the values of θ_b^i (here $b \in \{1,2\}$) are evaluated for each cell given z_1(red vector field) and z_2 (blue vector field). The function e is also calculated in each cell in the map. The lower values are assigned to darker colours and the higher values are assigned to lighter colours. It can be seen from this figure that when the estimated heading (θ_b^i) is closer for both observations (the headings are more aligned), the value of e is smaller. In contrast, when the estimated headings are less aligned e becomes larger.

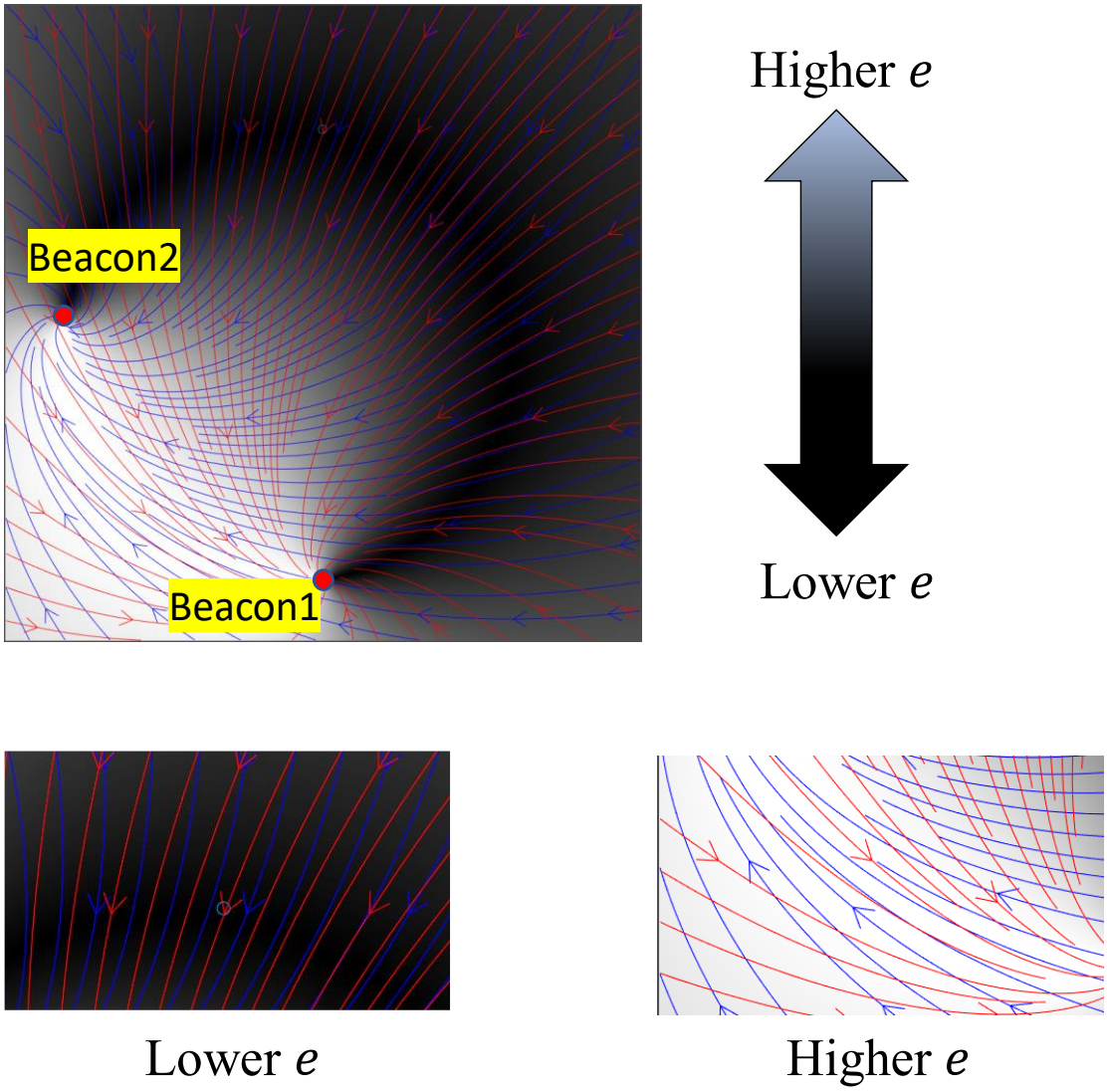


Figure B.9.5: This figure demonstrates the estimated e for each cell in a map using observations from two beacons.

The function e is related to the observation likelihood. This can be shown by investigating $p(Z|x^i, y^i, \theta^i)$ (where $Z = [z_1, \dots, z_I]'$), which is the observation likelihood.In order to evaluate $p\left(Z\middle|x^i, y^i, \theta^i\right)$, one can commence with the residual (v_b) shown in Equation B.5. The residual is computed by measuring the difference between the actual observation (z_b) and the estimate of this value (the term shown inside the bracket in Equation B.5). Since the values are angular, the maximum distance is expected to be 180°. In order to ensure this, Equation B.6 can be used to define residuals instead. If it can be assumed that the observations are independent (and they do not have a bias), it can be assumed that the errors **locally** follow a Gaussian distribution. If this assumption can be held true; then for a given cell, Equation B.7 can be used to evaluate the likelihood (where Σ_v is a diagonal matrix denoting the covariance of the observations, $|\Sigma_v|$ is its determinant, and $V = [\hat{v}_1 \quad \dots \quad \hat{v}_I]'$). Since e (Equation B.4) is not dependent on θ^i, the dependency of the likelihood on θ^i can also be removed by using the Maximum Likelihood Estimation (MLE) as shown in Equation B.8 for each cell. The $\hat{\theta}^i$ can be inserted into $p\left(Z\middle|x^i, y^i, \hat{\theta}^i\right)$.

$$v_b = [atan2\left(y^i - y_b^i, x^i - x_b^i\right) - \theta^i] - z_b \tag{B.5}$$

$$\hat{v}_b = \begin{cases} |v_b|, & |v_b| \leq \pi \\ 2\pi - |v_b| \ , & |v_b| > \pi \end{cases} \tag{B.6}$$

$$p\left(Z\middle|x^i, y^i, \theta^i\right) = \left(\frac{1}{(2\pi)^{I/2}\sqrt{|\Sigma_v|}}\right) exp\left(-\frac{1}{2}V'\Sigma_v^{-1}V\right) \tag{B.7}$$

$$\hat{\theta}^i = argmax_{\theta^i}\, p\left(Z\middle|x^i, y^i, \theta^i\right) \tag{B.8}$$

The e computed in Equation B.3 is related closely to the observation likelihood $p\left(Z\middle|x^i, y^i, \hat{\theta}^i\right)$. This can be shown by first defining function $f(e)$ in Equation B.9. In Figure B.6; it is shown empirically that Equations likelihood and B.9 will result in similar probabilities (Figure B.6 (a) shows the evaluation

of Equation B.9, and Figure B.6 (b) shows the evaluation of the likelihood). The red points show 1000 points with the highest values. Based on this figure, f is a smoother alternative to the likelihood.

$$f(e) = \left(\frac{1}{\sigma_e\sqrt{2\pi}}\right) exp(-\frac{1}{2\sigma_e^2}e^2) \tag{B.9}$$

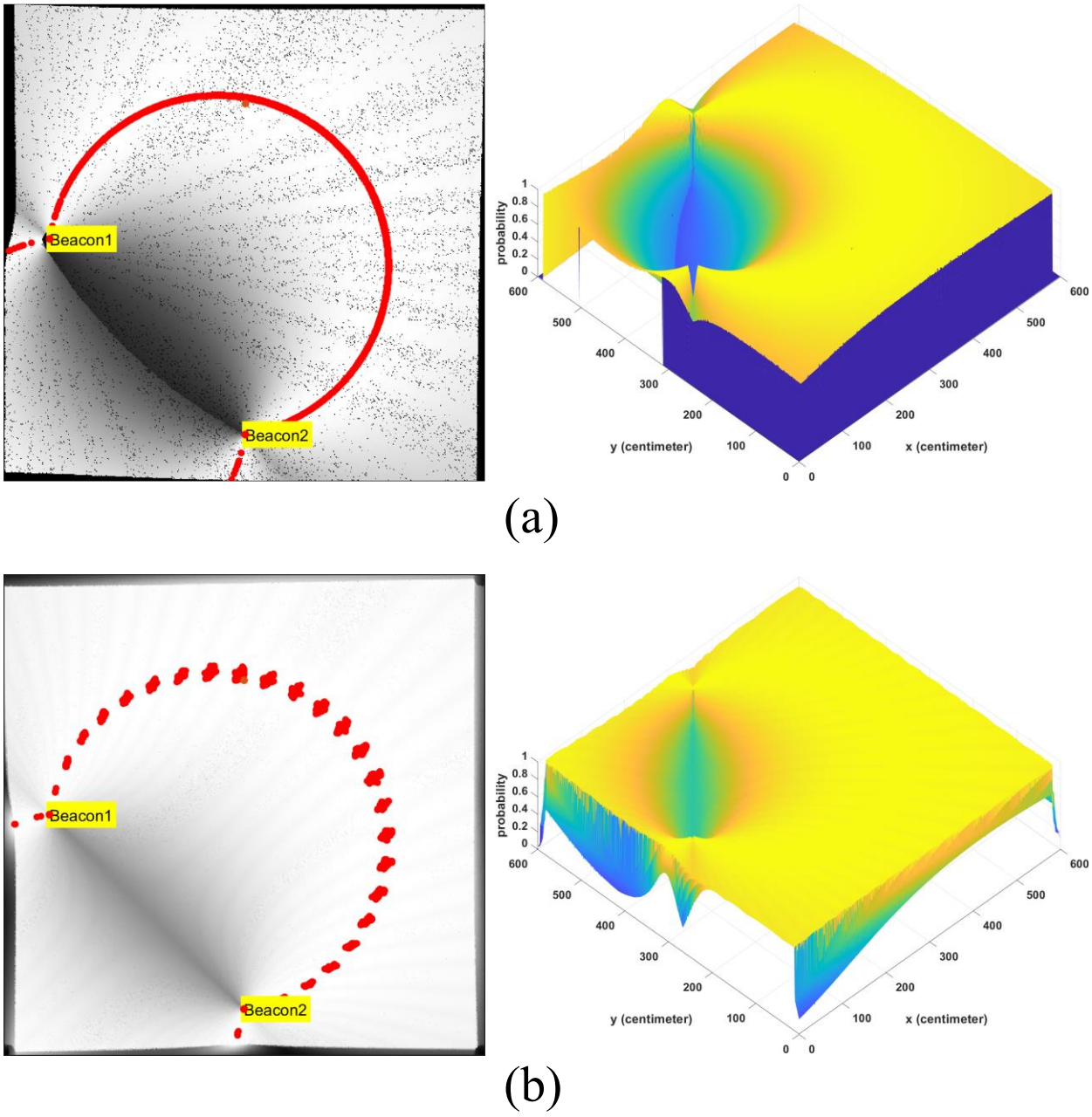


Figure B.9.6: A comparison of the approximate observation likelihood (a) and the likelihood (b)

Based on the illustrations $f(e)$ can be used to replace the likelihood function. It is important to note that evaluating $p\left(Z\middle|x^i, y^i, \hat{\theta}^i\right)$ requires finding the MLE for each cell over all possible robot headings. In our computations, this is achieved by discretizing the heading between -180° to 180° using 5° step. This results in approximately 72 times more computations than evaluating $f(e)$.

Additional to these advantages, the developed method estimates the heading with the highest likelihood for a given cell. Thus, the dimensionality of the unknown state of the robot has been reduced from three to two. With the help of the developed approach, particle filter-based positioning systems are only required to take samples in two dimensions which are more computationally efficient.

In the following, an overview of the characteristics of positioning using the developed beacon system is explained. As it is known, the position estimation depends on the geometry of the transmitters and the receivers (geometry here refers to the relative positioning of the transmitter and the receiver in the map). In order to demonstrate this, Figure B.7 is provided. In this figure, the robot (shown in a light blue circle) is moving downwards (this is the true position of the robot). Two beacons are used to estimate the position of the robot (the beacons are shown in yellow labels). The darker blue points show the 1000 cells with the highest likelihood values. It can be seen that the true position remains close to the points with the highest likelihood values. However, there is an ambiguity along an arc passing through the beacons and the robot.

The reason for this can be explained by a simpler schematic shown in Figure B.8, where the robot is exactly situated on the line from one beacon to the other. Any location along this line will result in the same observations $[z_1 = 0^\circ, z_2 = 180^\circ]$. Thus, two beacons can only locate the robot up to an ambiguity along this line, and it is important to place the third beacon not close to this line.

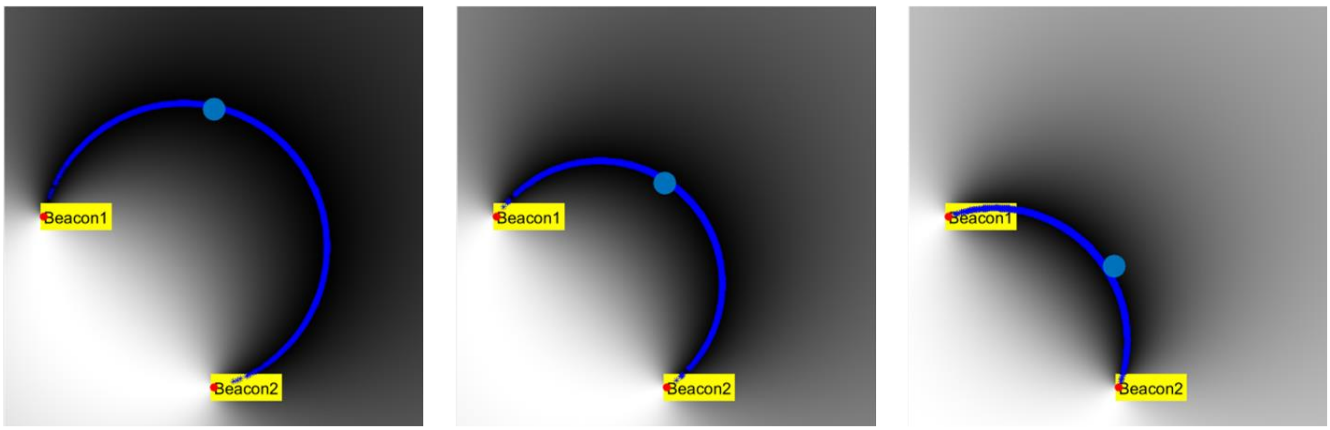


Figure B.9.7: This figure illustrates the estimated position of the robot using the developed likelihood function.

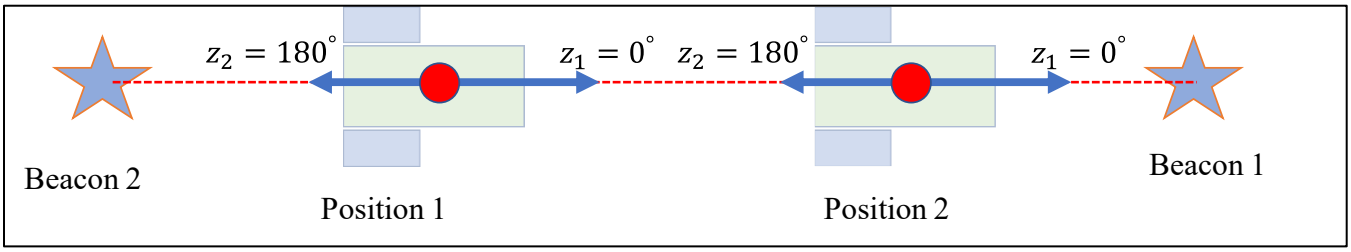


Figure B.9.8: A schematic illustrating the ambiguity of the robot's position along the line connecting the two beacons.

Finally, an example of the estimated position using three beacons is provided in Figure B.9. In this figure, the beacons are shown in red, and the estimated position (the best estimate corresponds to MLE) is shown as a brown point. The cells with the highest likelihood are shown in a darker gray, and the cells with the lower likelihood are shown in a lighter gray. Three circles are shown in the dashed line. Each circle passes through two beacons and the position of the robot. Intuitively the position of the robot should be in the intersected point of all three circles, and it can be seen indeed the intersection point of all three beacons corresponds to the estimated MLE.

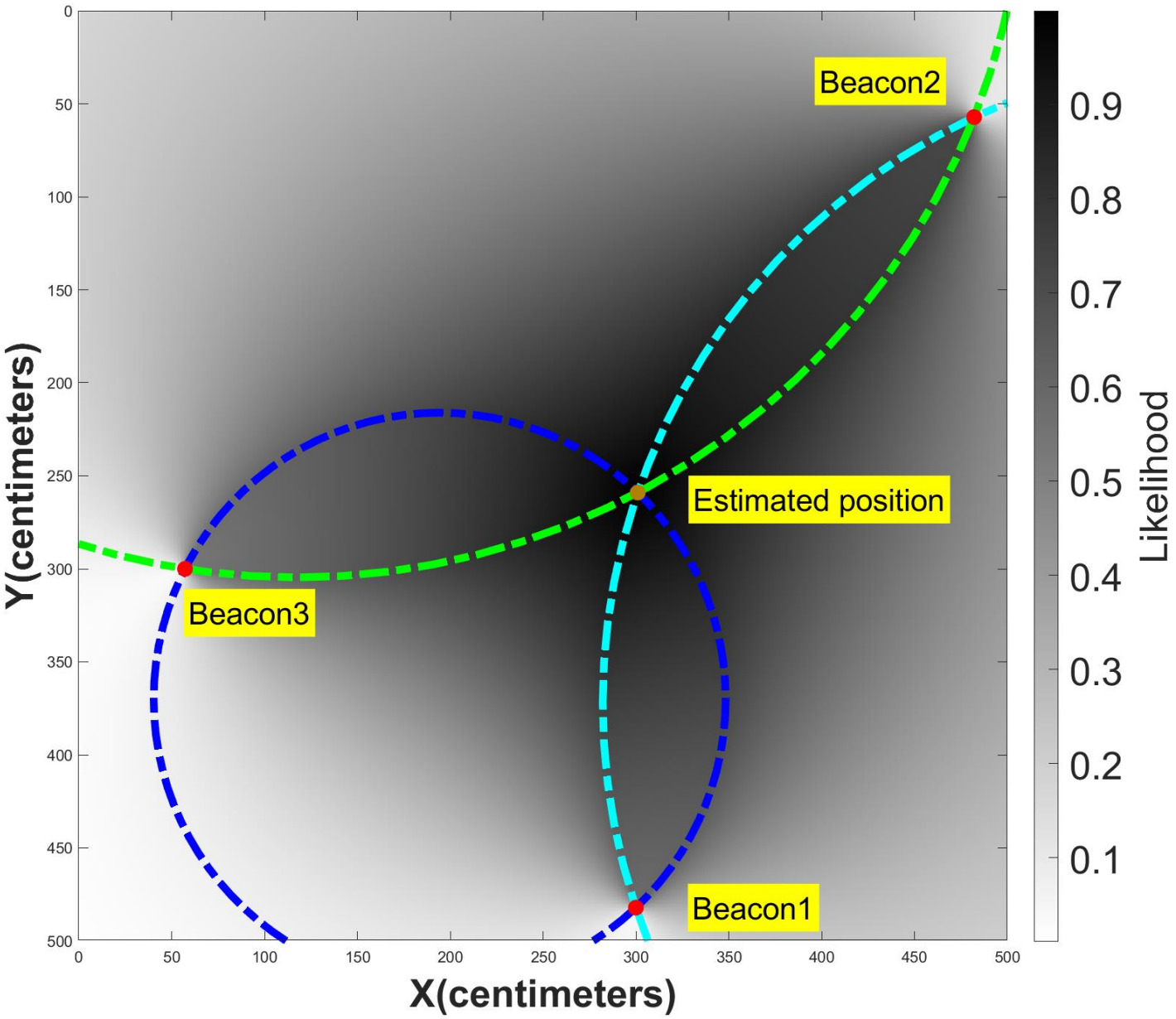


Figure B.9.9: This figure illustrates the position estimation using MLE with the help of a developed function using three beacons.