Stochastic Modeling for Mobile Manipulators

Xie Jiayu
Marquette University

STOCHASTIC MODELING FOR MOBILE MANIPULATORS

by

Xie Jiayu

A Dissertation submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

December 2021

ABSTRACT
STOCHASTIC MODELING
FOR MOBILE MANIPULATORS

Xie Jiayu

Marquette University, 2021

Mobile manipulators are valuable and highly desired in many fields, especially in industrial environments. However, determining the end effector position has been challenging for scenarios where the base moves at the same time that the arm follows commands to perform specific tasks. Earlier works have attempted to dynamically evaluate the problem of positioning error for mobile manipulators, but there is still room for further improvement. In this thesis, we devise a dynamical model that leverages stochastic search strategies for mobile manipulators. More specifically, we develop a dynamic model that estimates the position of the robot using an Unscented Kalman filter. Simulations using the Robot Operating System (ROS) and Gazebo were carried out to evaluate our model. Our results for the stochastic method show that it outperforms a deterministic approach (spiral search) under specific Kalman filter covariances of the process and observation noises. Compared to the state of the art, our proposed approach is more robust and efficient, proving to work under different arrangement scenarios with significant better performance.

ACKNOWLEDGMENTS

Xie Jiayu

I would like to start by thanking my advisor, Dr. Henry Medeiros, for the opportunity to work under his supervision. Without any doubt, I had developed so much in all aspects of my life, from technical to personal perspectives.

I would also like to thank the committee members, Dr. Bostelman and Dr. Yaz, for accepting to serve in the committee and contribute with their vast experience.

Many thanks to all the other professors from the department who helped me become the professional I am today. Thanks also to Ms. Katie Tarara for having my back and facilitating all the bureaucracy that I had to go through during this period.

I am also very grateful for the friends I met at the COVISS lab: Jamir, German, Siddique, Scott, Byron, Reza and Paris. Thanks for those coffee/beer breaks. Thanks German for being a great peer, who taught me a lot about Kalman filters.

I am eternally grateful to Philipe, who spent so much time iterating over my thesis as I was writing. I would not have gone this far without his support and care. More than that, thanks for inspiring me to be a better person.

Thanks to Sydney and Alex for the memorable and special moments we spent together. I am more than grateful for participating in so many special occasions of their lives. I hope to visit them soon.

Thank you for the roommates I had during this period: Katie, Natasha, Steve and Chris. They made my stay here easier, specially during the pandemic times.

Last, but not least, many thanks to my family and friends back in Brazil. Your support was essential, even though our connection was through long-distance calls.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Mobile robots, such as the one illustrated in Figure 1.1, can facilitate a variety of human tasks, especially in industrial environments. They can reduce human efforts and increase the safety of tasks within industrial facilities [3, 4, 5]. In addition, mobile manipulators have been used to carry materials between workstations [6] and worked alongside humans on moving assembly lines [7]. Their applications are not limited to industrial environments. Within healthcare field, robotics can make a big difference to tackle the current challenges faced by both patients and providers: elderly individuals require daily assistance and many healthcare facilities do not have enough personnel and when there is, the associated costs are high. Robots can assist the healthcare professionals to fulfill these needs. For instance, in [8] a system that integrates a robotic bed and a mobile manipulator was designed for bedside assistance.

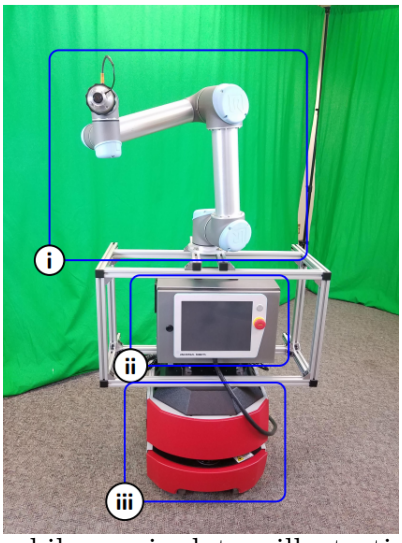
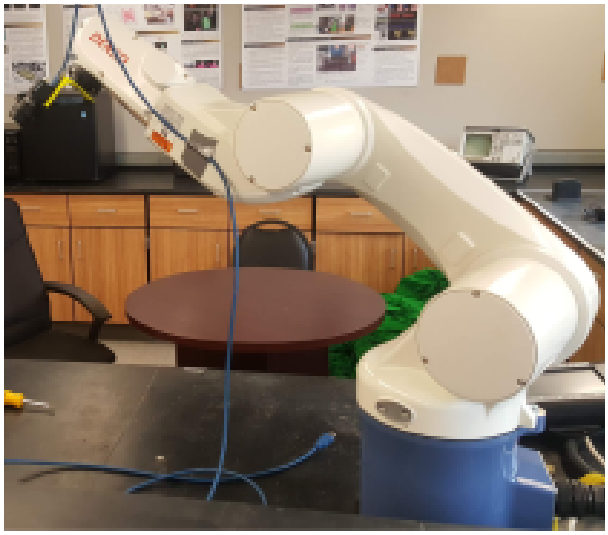
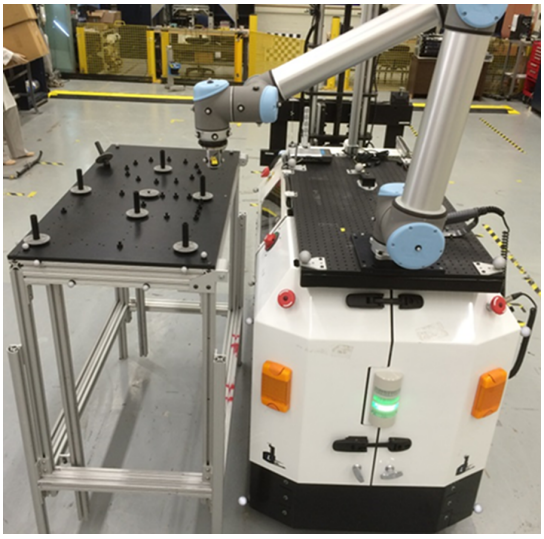


Figure 1.1: Elements of a mobile manipulator, illustrating the manipulator arm (i), its control box (ii) and the mobile base (iii). Figure from [1].

Although mobile manipulators have been in use for a while, they still face some limitations. Traditional stationary manipulators have a limited workspace, since the fixed position of the robot base dictates a static area within which the goal can be accomplished. Mobile manipulators, on the other hand, have the potential to fill this gap, since they require minimal reconfiguration to enable reassignments [9]. Furthermore, the use of mobile manipulators over fixed robotics in manufacturing environments can result in more agile and money-saving scenarios, as a single mobile manipulator can replace several stationary manipulators. Figure 1.2 shows one example of a stationary and a mobile manipulators.



(a) A stationary manipulator.



(b) A mobile manipulator.

Figure 1.2: An example showing the two types of manipulators. In (a), the base is fixed and in (b) the base can move.

Regardless of their numerous advantages, mobile manipulators still lack accuracy in end effector positioning in the absence of external localization and feedback control. This is due to the dynamic interaction in scenarios where both the manipulator and the mobile platform move simultaneously [10]. Some previous works have

been proposed to address this drawback. In [11] for example, the authors used deterministic approaches such as a spiral search around the expected location, with the platform stationary.

1.1 Motivation

Research interest in mobile manipulators has been steadily increasing over the last two decades [12, 13, 14]. Studies have been conducted in robot navigation, path planning, and performance evaluation [15, 16, 17]. Mobile manipulators have been the subject of study in diverse applications related to problems in anthropomorphic locomotion mechanisms [18, 19, 20] or industrial applications where force amplification is needed, especially to optimize ergonomy. For instance, studies that aim to build ergonomic models where reduced muscle activation is needed to perform a drilling task [21].

Despite the wide range of applications and environments in which mobile manipulators can be useful, dynamic manipulation is still a challenging task [7, 22, 23]. The conventional way of dealing with manipulators focuses on control and stability of both the base and arm [24, 7]. Since mobile manipulators are complex systems, different research areas are concerned with specific elements of such systems. For instance, in [25], the focus was on the design of a Cartesian controller that combines the movement of the base and the arm to achieve specific 3D poses of the end effector, but the system lacks robustness to calibration issues when performing experiments with a real robot. According to the authors, the end effector reached the goal pose in the simulations, whereas in the real world experiments the end effector was slightly shifted with respect to the marker, due to arm calibration bias.

Mobile manipulators require significant calibration because they tend to suffer from reduced precision in comparison with stationary manipulators, where there is no motion of the base [26]. Existing methods to address this problem have used determin-

istic perspectives [11], with the platform stationary, to localize a set of retroreflective markers utilizing a photoeletric laser sensor attached to its end effector.

We decided to build our approach upon the method proposed by Yaw et al. [26]. That particular work considers the problem of dynamic performance measurement for mobile manipulators, which is also one of the main goals of this thesis. In addition to this common purpose, the authors built two dynamic search methods to locate fiducials from a mobile platform: a spiral-based deterministic mechanism and a stochastic procedure based on a Kalman filter. Both methods allow the robot to explore a calibration artifact and search for fiducials with the base in motion.

Despite the good results presented in [26], improvements could be made in the theoretical model of the stochastic system. For instance, a dynamic adjustment of the process covariance can improve the estimation step of the Kalman filter. This can be achieved by increasing the covariance of the observation noise when a marker is not found after the search process. As a consequence, we expect this procedure to result in the generation of sample points that are closer to the fiducial position and thus increase the probability of locating it. Furthermore, the need to perform a search if there is no fiducial interception could be reduced if we take into account the time interval between observations in the prediction process. Consequently, the performance would improve due to a lower overall search time.

1.2 Objectives

In this thesis we aim to devise a stochastic model to predict the position of a mobile manipulator as it moves along a stationary path, and to quickly locate objects of interest in the presence of motion uncertainty. Building upon the approach proposed in [26], we provide a more precise theoretical model of the platform and a more comprehensive assessment of the performance of the system as a function of several parameters that impact the configuration of the environment where the robot

operates.

1.2.1 Specific objectives

To achieve our overarching goal, we propose the following four specific objectives.

Objective 1: Design a mathematical model of the system

Prior to designing a stochastic search method, we need to define the problem mathematically. In order to more accurately reflect the characteristics of the actual system, we model the problem using a state-space representation. By iteratively refining and evaluating the mathematical model, it is possible to determine if any modelling assumption or approximation is invalid.

Objective 2: Design Kalman filters to estimate the state of the mobile manipulator

The main objective of the thesis is to design a Kalman filter that best fits our problem definition. The Kalman filter is a recurrent algorithm that estimates the state of a system given measurements observed over time [27]. In our problem, we make use of the Kalman filter to estimate the location of the base at the position where it should find a fiducial. A laser sensor attached to the end effector is used to localize the fiducials. When a detection happens, measurements from the laser are collected and transformed through the state of the mobile manipulator and used as inputs for the next prediction. These inputs are used to refine the estimated base position where the mobile manipulator should locate the next fiducial in the sequence.

Objective 3: Implement the system using the Robot Operating System (ROS)

Robot Operating System (ROS) is a framework of libraries and tools to perform most common robotic tasks. Another advantage of ROS is the fact that it facilitates collaborative robotics software development across universities and institutions [28]. We make use of this framework to avoid the need to re-implement commonly used features in most of robotic applications, such as inter-process communication, navigation, drivers, motion planning and perception, among others.

Objective 4: Evaluate the system using the Gazebo simulator

In order to validate our model, we need to perform extensive simulated experiments. This objective also serves to assess the robustness of the devised method to different base velocities and distinct path arrangements.

1.3 Contributions

We developed a Bayesian algorithm using an Unscented Kalman filter to better estimate the pose of the mobile manipulator in a scenario where the arm and the base move simultaneously. More specifically, we consider a scenario in which both the Autonomous-Unmanned Ground Vehicle (A-UGV) and the robotic arm move simultaneously while the mobile arm localizes small objects in a known reconfigurable mobile manipulator artifact (RMMA) such as the one shown in Figure 1.3. The small objects consist of multiple retroreflective markers embedded in the RMMA. While the A-UGV moves at a pre-defined speed in a certain direction (which can be either in the positive or negative direction of the x-axis), a 6 degrees-of-freedom (DOF) manipulator mounted on the A-UGV has to align its end effector in the same position and orientation with respect to each marker in sequence until the entire artifact has been observed. In order to intercept the markers, a laser sensor is mounted to the

end effector as shown in Figure 1.4.

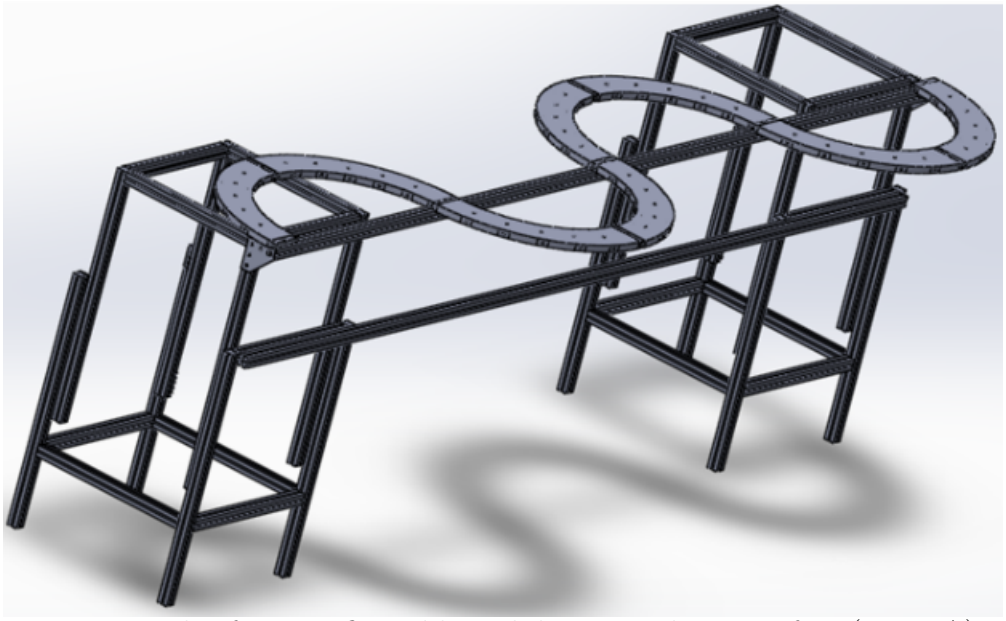


Figure 1.3: Example of a reconfigurable mobile manipulator artifact (RMMA) similar to the one we used in our experiments.

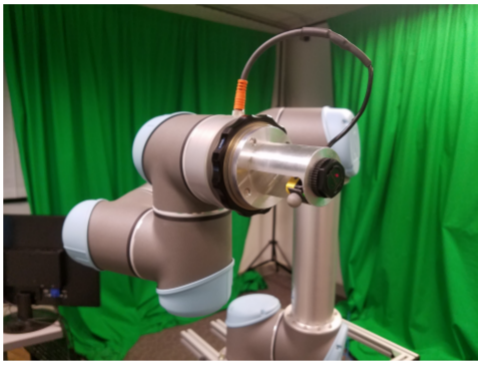


Figure 1.4: Laser sensor mounted on the end effector of the UR5 manipulator used in this thesis.

Hence, the main contribution of this work is a dynamical model system that may be used to estimate the state of a mobile manipulator over time, correcting the position of the end effector under uncertainties and platform noises.

1.4 Organization

This thesis is organized as follows. Chapter 2 presents the background on the kinematics of mobile manipulators, and provides two types of search methods explored in this work. It also discusses the software infrastructure under which the proposed model was implemented and tested followed by an overview of the state-space representation and the different categories of Kalman filters. Chapter 3 describes our proposed approach using a linear Kalman filter, followed by some of the limitations of this method and how we mitigated them by approaching our problem using a non-linear perspective. Chapter 4 explains how the system model was implemented and evaluated using ROS and how the data was collected, describing the configuration of the simulated platform as well as the experimental setup. Chapter 5 presents the results obtained in this thesis and an analysis of the results. Chapter 6 discusses the final conclusions from this work and lists some possible future works.

CHAPTER 2
BACKGROUND AND RELATED WORK

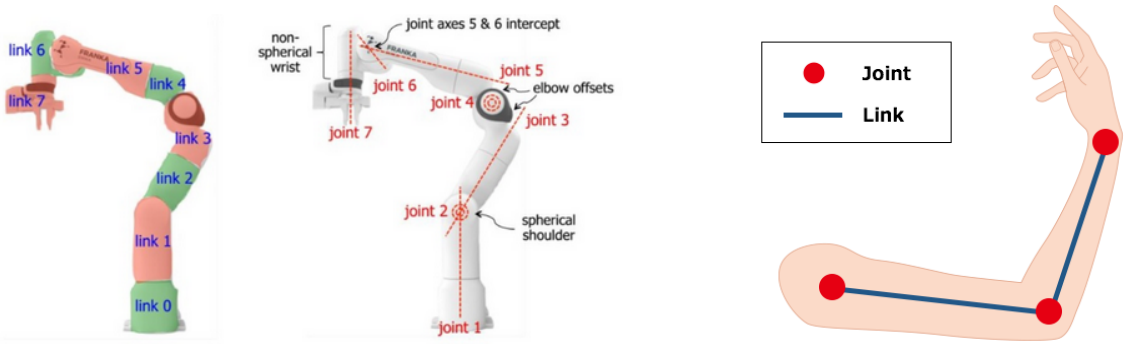
In this chapter, we first provide the necessary background on the kinematics of mobile manipulators, which includes both the task of forward kinematics as well as the task of inverse kinematics. Next, we transition to a discussion of search methods for robotic manipulators and provide an overview of how this task has been approached in the literature, from a deterministic point of view to a stochastic perspective. In the sequence, we provide an overview of the software infrastructure later used to evaluate our dynamic model. Finally, we cover the concepts of state space representation, followed by Kalman filters, their applications, and comparisons between linear and non-linear estimation models.

2.1 Robot Kinematics

A robotic mechanism can be expressed as a set of rigid structures named *links* that are connected through a set of *joints*. Figure 2.1 illustrates the structure of a robotic arm, contrasting the links and joints in a robotic arm to the representation of a human arm. Kinematics is the study of the relationships between the robot's joints and its corresponding spatial layout, so that the position of the robot and its motion can be accurately estimated. A robot's layout can be represented with either a list of joint coordinates (angle and translation with respect to a reference frame) or with a spatial characterization of its links in the corresponding 2D or 3D workspace.

2.1.1 Forward Kinematics

Forward Kinematics (FK) is the procedure of obtaining the coordinate frames of a robot's links, given a configuration and the robot's kinematic structure as input [30]. For instance, the forward kinematics problem of a stationary robotic arm is



(a) Structure of a robotic arm.

(b) Links and joint in human arm.

Figure 2.1: An example showing (a) the structure of links and joints in a robotic arm and (b) how similar it is to a human arm. Reproduced with permission from [29].

equivalent to calculating the Cartesian position and orientation of the end effector, given the joint angles as input.

2.1.2 Inverse Kinematics

Inverse Kinematics (IK) is the inverse process. That is, given the relative positions and orientations of the end effector relative to the base, the task is to find the values of all of the joint positions [31]. For a robotic arm, the problem can be translated to the following question: what are the joint angles of the robot so the end effector achieves a specific position and orientation in a Cartesian coordinate system? Whereas for a mobile robot the task would be the calculation of the wheel velocities given the goal pose of the robot [29]. Figure 2.2 shows a simple example [1] of the difference between a FK problem (left) and an IK problem (right).

2.2 Search methods for mobile manipulators

In the context of this thesis, the purpose of a search process is to increase the probability of locating specific fiducials in the reconfigurable artifact. The need to

[1] Source: (https://www.cs.cmu.edu/ motionplanning/lecture/Chap3-Config-Space_howie.pdf)

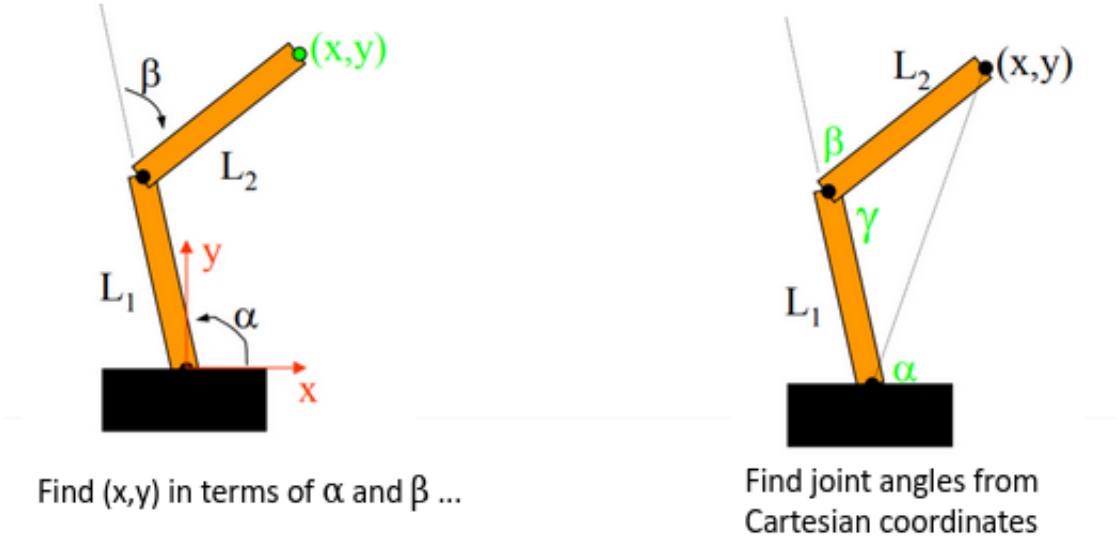


Figure 2.2: Comparison between Forward Kinematics (left) and Inverse Kinematics (right), where L_1 and L_2 are the links, α. β, and γ are the joint angles and (x, y) is the end effector position in x and y coordinates.

perform such an operation is to increase the chances of locating each marker. In this thesis, we explore two methods: a stochastic search method using a Kalman filter and a deterministic approach based on a spiral search strategy.

The spiral search method was initially proposed in [26], which adapted the method described in [11] for use in dynamic manipulation scenarios. The main focus of this thesis are stochastic search strategies. The principal hypothesis guiding our preference for a stochastic approach over a deterministic one is that the estimates generated by a Kalman filter should yield more precise sample positions in which the object of interest is likely to be located. In contrast, the deterministic approach consists of a brute force search, significantly increasing the effort needed to locate the markers.

Another advantage of the Kalman filter is the fact that it is an optimal estimator from a least squares perspective as long as the model satisfies the filter's assumptions, that is, when both the observation and process noises are normally dis-

tributed and the system dynamic and observation models can be represented using linear relationships [32].

Spiral search

The spiral search algorithm consists of creating evenly spaced waypoints on a spiral placed at the expected position of the fiducial. Figure 2.3 illustrates the waypoints generated along the spiral path. In terms of parameterization, both the gap between waypoints along the path of the spiral and the distance between adjacent paths can be adjusted and optimized according to the search area to be covered. More implementation details of this method can be found in [26] and [11].

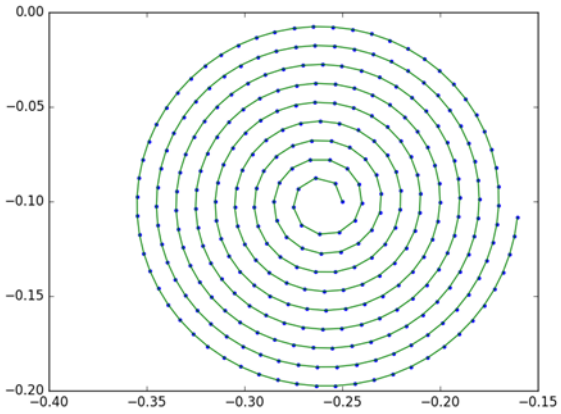


Figure 2.3: Sample points from the spiral search method.

Stochastic search

In contrast with the deterministic approach, the stochastic method builds its search path from random samples generated according to an estimated probability distribution of the location of the fiducial. As detailed in Chapter 3, in this thesis

a recursive Bayesian estimator (more specifically, variations of the Kalman filter) is used to estimate this distribution. First, it estimates the base position at which the end effector attached to the arm should locate the fiducial. From the predicted base position, through the kinematics of the robot, we can obtain the predicted end effector position and then perform the sampling centered in this position.

The search path starts with samples collected from corresponding regions of high probability according to the estimated distribution, moving to lower probability samples. In this way, it is expected a balance between exploration and exploitation that increases the probability of interception, and this reduces the search time [1].

In more details, the proposed stochastic sampling utilizes a Gaussian distribution, centered at the estimated position of the fiducial with a covariance equal to the prediction covariance also provided by the Bayesian estimator. The sample points are generated according to Equation 2.1

$$s_k \sim \mathcal{N}(\hat{x}_{k|k-1}, \mathcal{P}_{k|k-1}), \tag{2.1}$$

which represents a normal distribution with mean $\hat{x}_{k|k-1}$ (predicted position of the end effector) and covariance $\mathcal{P}_{k|k-1}$.

Following [26], we discretize the search path to reduce the amount of overlap in the cumulative receptive field. The motivation for the discretization is that the normal distribution used to generate samples in the stochastic search method is continuous, such that without this procedure a large number of samples are very closely spaced, as Figure 2.4 illustrates.

2.2.1 Motion compensation

As mentioned in [26], as the base of the robot is in motion during the search process, the search points generated as described in Equation 2.1 must be compen-

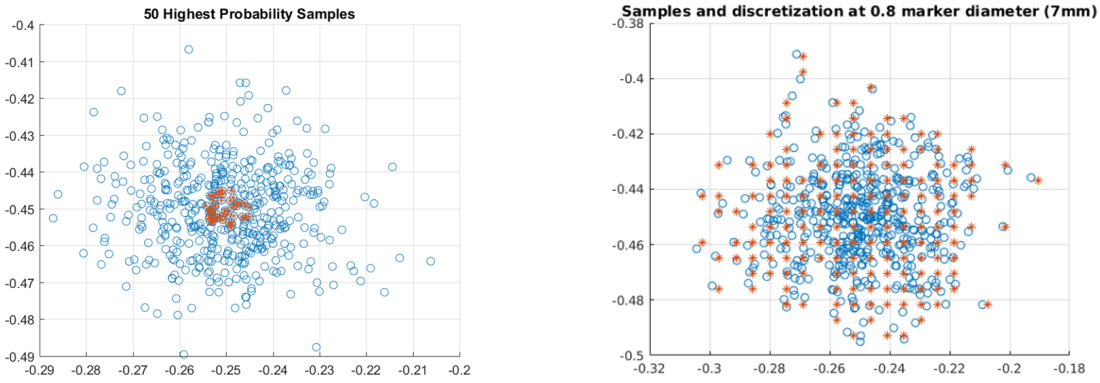


Figure 2.4: Discretization of the stochastic search sample points. a) The blue circles represent 200 samples generated by the stochastic search method and the orange stars represent the 50 highest probability points. b) After discretization, the points are distributed over the entire search area, rather than being localized near the center of the distribution.

sated for the base displacement. That is,

$$\hat{s}_k = s_k + v_b \delta_k, \tag{2.2}$$

where $\hat{s}_k$ is k-th velocity-compensated sample, v_b is the base velocity and δ_k is the time elapsed as the manipulator moves between point $k-1$ and k. Figure 2.5 illustrates the samples generated by the Gaussian distribution and the velocity-compensated samples.

2.3 ROS and Gazebo

In this section we present an overview of the software infrastructure used in this work.

2.3.1 Robot Operating System (ROS)

ROS is a set of software tools and libraries created with the purpose to facilitate the control and handling of different types of robotic modules [28]. The need for an open collaboration framework across the robotic research community was

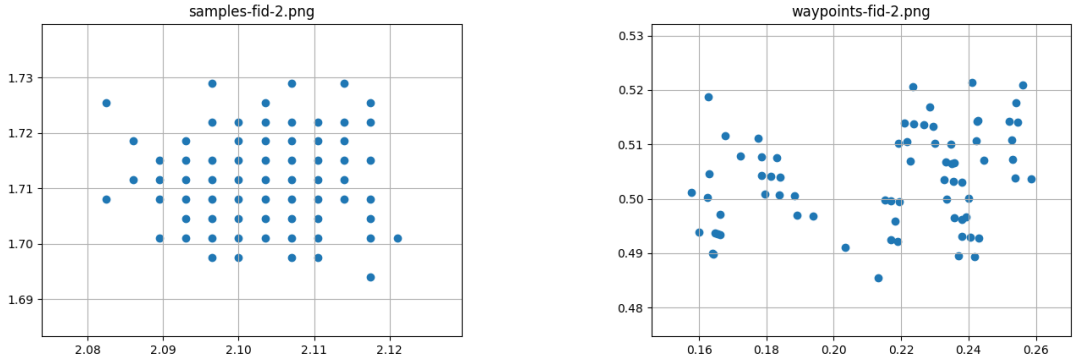


Figure 2.5: Example of one set of sample points, (a) before velocity-compensation and (b) after velocity-compensation.

the motivation to develop such a system, built to encourage collaborative software development in robotics. Since its introduction in 2009 [33], many tasks have been developed using ROS. Some common robotic tasks within the ROS community include: object retrieval, map building, and navigation in indoor and outdoor environments. In the following paragraphs, we summarize some features of ROS based on information provided in [34] and [28].

ROS structure

The communication among the elements of a ROS-based system is based on a peer-to-peer network, and the main components of the connection graph are described below:

Nodes: Nodes are the processes that perform computation. For example, we can have one node to control a laser range-finder, a node to control the wheel motors, and another to perform localization.

Master: The master node allows other nodes to find each other, exchange messages, or invoke services. They are routed via a transport system with publish/subscribe semantics. All the nodes must subscribe to the Master in order to publish or collect

data. An example is shown in Figure 2.6.

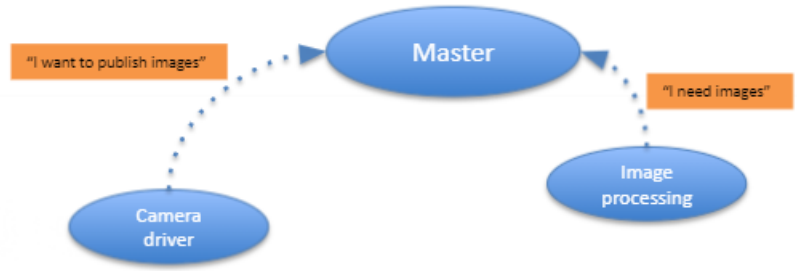


Figure 2.6: Illustration of the relationship between the ROS master and the nodes (blue ellipses). All nodes must subscribe to the ROS master.

Topics: Nodes communicate with each other by publishing messages to topics, as Figure 2.7 illustrates. A talker node publishes a message to a topic, which is then accessible to corresponding listener nodes.

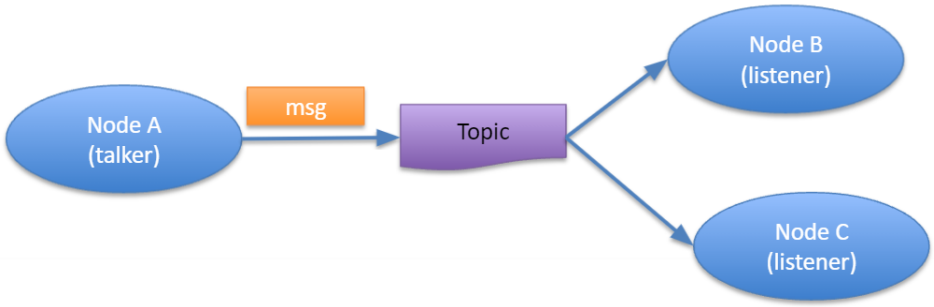


Figure 2.7: Illustration of the ROS communication process based on topics.

ROS Services: Services allow one node to call a function that executes in another node. It employs a Service/Client model with a one-to-one request-response approach.

ROS Packages: The software in ROS is organized in packages. A package is a combination of code, data, and documentation to address a specific robotic need,

such as the *tf* package described below.

TF Package

One of the most common tasks in robotics is coordinate frame transformation. For every task we want the robot to perform, there is always a target system of coordinates that is not necessarily the same as the source. Both source and target coordinates are defined with reference to some global coordinate system. The *tf* package provides a convenient way of handling different coordinate systems by enabling the tracking of each coordinate frame over time and allowing the user to request any transformation between two frames at any time. Figure 2.8 shows the relations among some of the coordinate frames in our ROS-based robot model.

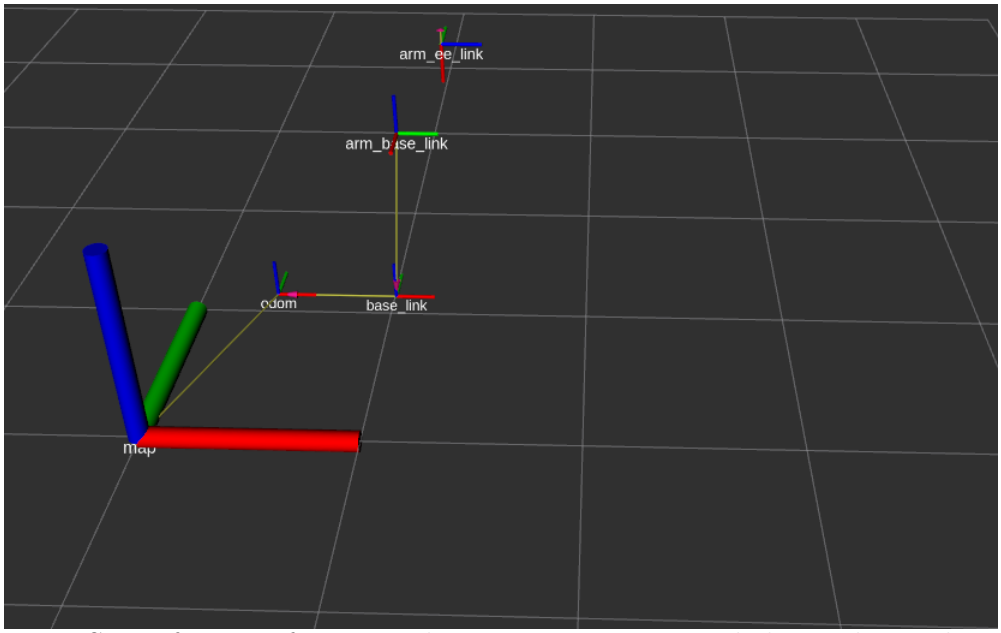


Figure 2.8: Some frames of our simulation environment and their relationships. The *map* frame represents the origin of the global coordinate system, the *odom* frame corresponds to the position of the base with respect to the map as measured by its odometry, the *base_link* is the body frame of the robot, *arm_base_link* represents the reference frame of the arm and *arm_ee_link* is for the frame of the end effector.

2.3.2 Gazebo

Gazebo is a 3D dynamic simulator that reproduces robots in complex indoor and outdoor environments [35]. It allows the user to define robots using the Simulation Description Format (SDF) or the Unified Robot Description Format (URDF). Such models consist of a structured description of the characteristics of the robot, such as its shape, joints, and wheels. Gazebo also provides tools to create the environment surrounding the robot, such as walls and objects. We used the Gazebo simulator to rapidly and safely test our model. The choice of this specific simulator is due to its interface that can be connected with ROS through the *gazebo_ros* package, allowing the user to quickly interchange real processes with simulated ones. Figure 2.9 illustrates the environment we created in Gazebo for the work described in this thesis.

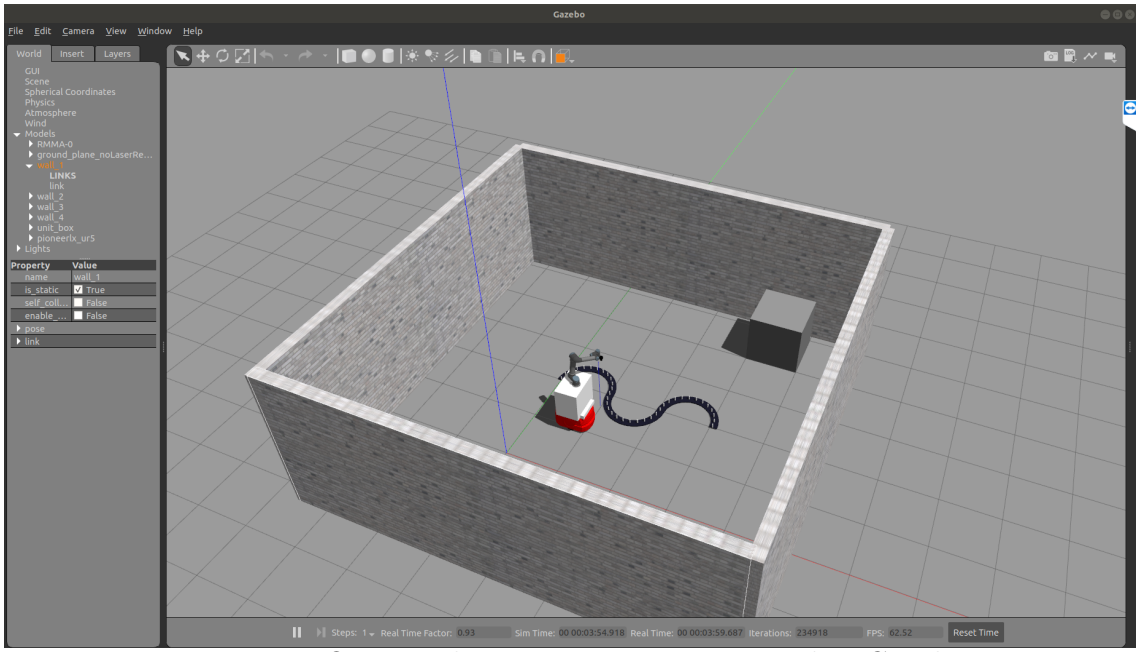


Figure 2.9: Our simulation environment created in Gazebo.

2.4 State-space representation

Representing systems with differential equations or transfer functions becomes unmanageable as these systems become more complex, especially when they have multiple inputs and outputs. The state-space representation is a powerful strategy to alleviate this problem.

Following the notation in [36], the state-space representation of a discrete system of order n can be mathematically defined by a minimum set of variables $x_i[k]$ with $i = 1, ..., n$. Together with the knowledge of those variables at an initial time step k_0 and the system inputs for time steps $k \geq k_0$, these are sufficient to predict the future state and outputs of the system.

For discrete-time Linear Time-Invariant (LTI) systems, state-space representations generally comprise two equations: (a) the "state equation" which describes how the input influences the state; and (b) the "output equation" which describes how the state together with the input impact the output. Figure 2.10 illustrates the dynamic behavior of the system inputs and outputs.

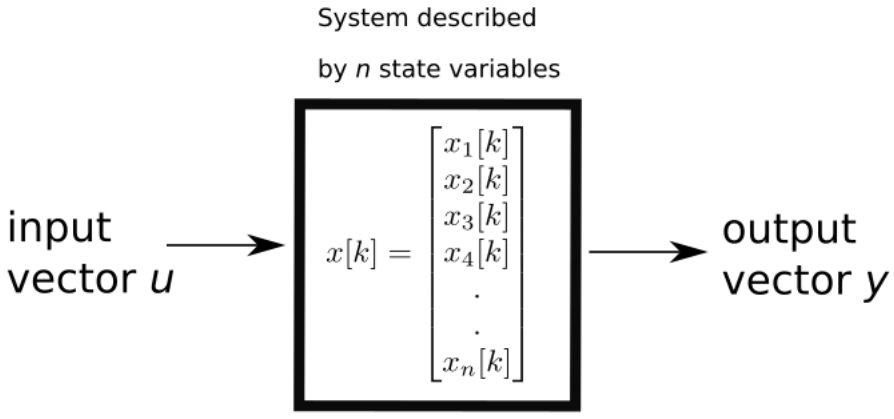


Figure 2.10: Diagram depicting the relationship between the inputs and the outputs of a system represented using the state-space representation.

2.5 Kalman filters

The Kalman filter is a recurrent algorithm that estimates the state of a system given measurements observed over time [27]. Highly accurate under uncertainty and noise, Kalman filters are largely used in many applications, such as target tracking, feature tracking in computer vision, filtering credit market data in economics [37], and terrain-referenced navigation [38].

The formulation of a Kalman filter is based on the state-space representation and is composed of two main steps: the *prediction step* and the *update step*. Figure 2.11 shows the steps inside each of these two main blocks, which can be described as follows. First, the *prediction* step captures the evolution of the state from time $k-1$ to time k. That is, the filter predicts the state of the system by making use of the system's model together with the knowledge of its state at timestep $k-1$. The *update* step then captures the relationship between the state and the measurement, with a comparison between the prediction made by the filter at time $k-1$ and the actual measurements collected at time k. Based on how accurate this last prediction was, the filter then assigns a new weighed estimate for the state at timestep k.

2.5.1 Linear Kalman Filter

A linear Kalman filter can be described as follows. The notation we use in this section and later in Chapter 3 to describe our proposed model follows the one adopted in [39].

Prediction

$$\begin{aligned} \hat{x}[k|k-1] &= F_k \hat{x}[k|k-1] + B_k u[k] \\ P[k|k-1] &= F_k P[k|k-1] {F_k}^T + Q_k \end{aligned} \tag{2.3}$$

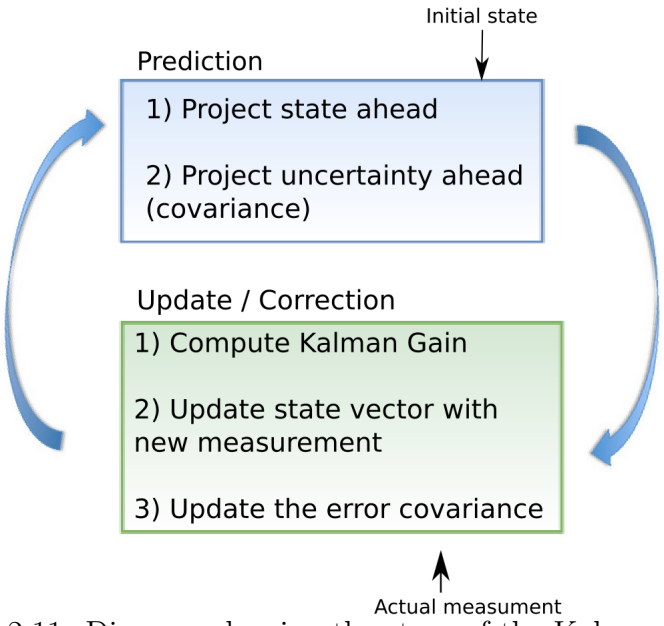


Figure 2.11: Diagram showing the steps of the Kalman filter.

Innovation

$$\begin{aligned} \tilde{y}[k] &= z[k] - H_k \hat{x}[k|k-1] \\ S_k &= H_k P[k|k-1] {H_k}^T + R_k \\ K_k &= P[k|k-1] {H_k}^T + {S_k}^{-1} \end{aligned} \tag{2.4}$$

Update

$$\begin{aligned} \hat{x}[k|k] &= \hat{x}[k|k-1] + K_k \tilde{y}[k] \\ P[k|k] &= (I - K_k H_k) P[k|k-1] \\ \tilde{y}[k|k] &= z[k] - H_k \hat{x}[k|k], \end{aligned} \tag{2.5}$$

where $\hat{x}[k|k-1]$ is the predicted filter state associated with a covariance matrix $P[k|k-1]$ at time step k given the state at $k-1$. F_k is the state transition matrix, while Q_k represents the process noise covariance. $\tilde{y}[k]$ is the innovation, which is the difference between the measurement $z[k]$ and the predicted state projected into the measurement space using the observation matrix H_k. R_k is the observation noise uncertainty associated with the system uncertainty S_k projected onto the observation

space. The Kalman gain at time step k is given by K_k. At the update step, $\hat{x}[k|k]$ is the corrected state estimate associated with a covariance matrix $P[k|k]$ at time step k. I is the identity matrix and $\tilde{y}[k|k]$ is the updated innovation.

The filter is initialized with a previously known state and its associated covariance. In the prediction step, the process model is responsible for predicting the state for the next time step and then adjusting the confidence to account for the corresponding uncertainty. When a measurement is collected with a corresponding confidence about its accuracy, the residual between the predicted observation and the actual measurement is computed. The Kalman gain can be understood as a scaling factor that weights the influence of the measurement and the prediction according to their uncertainty. This scaling factor is then used to set a state between the prediction and the measurement. Finally, the filter updates the confidence in the state according to the reliability of the measurement.

The propagation of a Gaussian random variable (GRV) through the system dynamics plays an essential role in the Kalman filter functionalities. In the next section, we explain in details how this propagation occurs within non-linear systems.

2.5.2 Extended (EKF) and Unscented Kalman Filter (UKF)

According to [2], the Extended Kalman Filter (EKF) is the most used technique for non-linear estimation. However, in that same paper, the authors point out limitations of this approach that can be mitigated by the Unscented Kalman Filter (UKF). Thus, the UKF is typically more accurate than the EKF for non-linear systems. In this section, we present a theoretical background of how these two algorithms work, justifying our choice for the UKF over the EKF in this thesis. Finally, a mathematical formulation of the UKF is provided.

Regarding the propagation of a Gaussian random variable (GRV) through the system dynamics, the EKF approximates the state distribution as a GRV and then

propagates it by means of a first-order linearization of the system. If the system is highly non-linear, however, such a transformation can introduce large errors into the posterior mean and covariance, occasionally causing a divergence in the filter. The UKF, on the other hand, addresses this issue from another perspective, making use of a deterministic sampling approach. This consists of approximating the state distribution by a GRV, but with a minimal set of selected sample points named *sigma points* that encompass the true mean and uncertainty of the GRV. In the UKF, a weighted sample mean and covariance are then computed after the sigma points are transformed by a nonlinear function. The posterior mean and covariance can then be estimated from the transformed sample points to capture the posterior parameters precisely to the third order of Taylor series expansion of the non-linear dynamic and observation models. Figure 2.12 shows the different mechanisms through which the GRV is propagated in both methods.

In terms of the design process, specifying a UKF does not differ much from a linear Kalman filter, except for one main aspect: instead of defining a matrix F and a matrix H for state transition and measurement, respectively, we replace them by non-linear functions $f(.)$ and $h(.)$. Equations 2.6, 2.7 and 2.8 show that the estimation procedure employed by the UKF is very similar to how a linear Kalman filter works.

Prediction

$$\begin{aligned} \mathcal{Y} &= f(\chi) \\ \hat{x}[k|k-1] &= \sum w^m \mathcal{Y} \\ P[k|k-1] &= \sum w^c (\mathcal{Y} - \hat{x}[k|k-1])(\mathcal{Y} - \hat{x}[k|k-1])^T + Q_k \\ \mu_z &= \sum w^m h(\mathcal{Y}) \end{aligned} \tag{2.6}$$

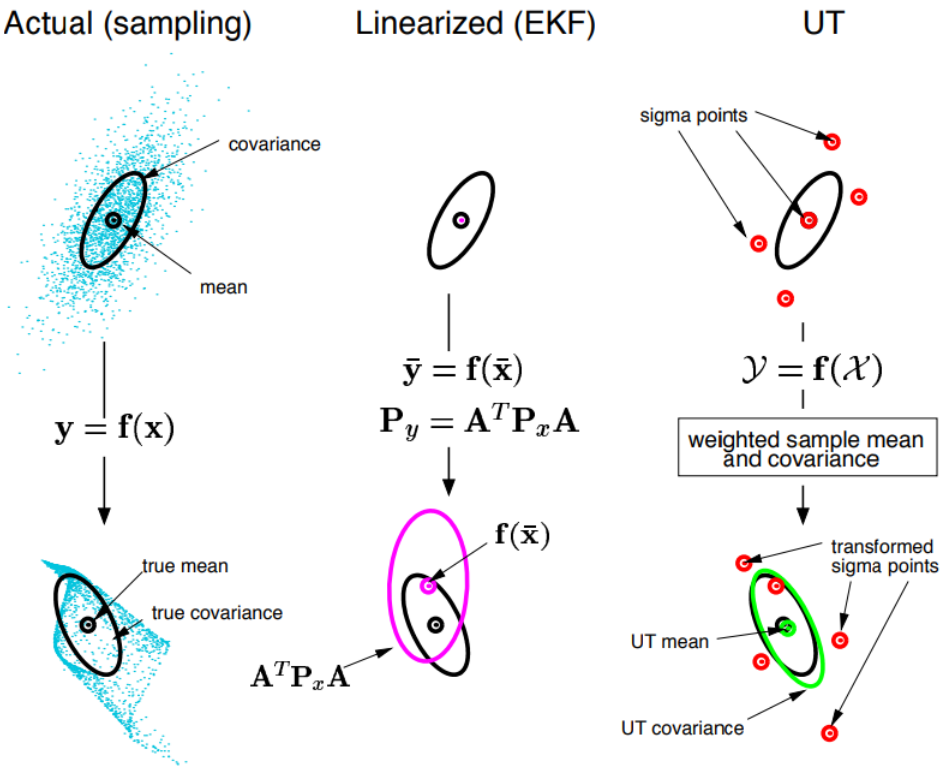


Figure 2.12: Illustration of the propagation of an actual GRV through a first-order linearization and using the unscented transform. Figure reproduced with permission from [2].

Innovation

$$\begin{aligned} \tilde{y}[k] &= z[k] - \mu_z \\ P_z &= \sum w^c(h(\mathcal{Y}) - \mu_z)(h(\mathcal{Y}) - \mu_z)^T + R_k \\ K_k &= \left[\sum w^c(\mathcal{Y} - \hat{x}[k|k-1])(h(\mathcal{Y}) - \mu_z)^T\right] {P_z}^{-1} \end{aligned} \tag{2.7}$$

Update

$$\begin{aligned} \hat{x}[k|k] &= \hat{x}[k|k-1] + K_k\tilde{y}[k] \\ P[k|k] &= P[k|k-1] - KP_zK^T, \end{aligned} \tag{2.8}$$

where χ are the sigma points that are related to the corresponding weights w^m and w^c. $\mathcal{Y}$ is the projection of the sigma points according to the process model defined by a non-linear function $f(.)$, $\hat{x}[k|k-1]$ is the predicted state associated with the covariance

$\mathcal{P}[k|k-1]$ at time step k given $k-1$. μ_z and P_z are the mean and covariance of the measurement sigma points after the unscented transform. The prior sigma points are used to update the filter in the measurement space. Hence, they are converted through a measurement function $h(.)$. The residual is given by $\tilde{y}[k]$ and K_k is the Kalman gain at time step k.

Generating sigma points

As explained in [40], the UKF takes $n_\sigma = 2n + 1$ samples to represent the distribution of the current state, where n is the dimension of the state vector as mentioned in Section 2.4. Let us consider the posterior state $x[k|k]$ and covariance $P[k|k]$, which represent the distribution of the current state. The first sigma point is the mean of the state, that is $\chi_0 = x_{k|k}$. The remaining $2n$ points are generated following Equation 2.9.

$$\boldsymbol{\chi}_i = \begin{cases} x[k|k] + \left[\sqrt{(n+\lambda)P[k|k]}\right]_i & \text{for i=1, \ldots, n} \\ x[k|k] - \left[\sqrt{(n+\lambda)P[k|k]}\right]_{i-n} & \text{for i=(n+1), \ldots, 2n} \end{cases} \tag{2.9}$$

where $\lambda = \alpha^2(n+\kappa) - n$ controls how the sigma points are distributed and weighted, which depends on parameters α and κ. The greater the parameter α the further the sigma points spread from the mean. Following [40], a good choice for Gaussian problems is to set $\kappa = 3 - n$ and $0 \leq \alpha \leq 1$.

CHAPTER 3
PROPOSED APPROACH

In this chapter, we discuss in detail the model we developed for evaluating the performance of the mobile manipulator to carry out a search process. We start by recalling the motivation for this thesis and the expected improvements over the current state-of-the-art stochastic search method [26]. Then, we analyse the proposed system model using a linear Kalman filter, discuss the limitations of this approach for our problem, and finally, present a non-linear model using the UKF.

As mentioned in Chapter 1, our starting point was to study and explore state-of-the-art search methods. Our main reference was [26], where the authors describe a Bayesian framework for prediction of the end effector position and subsequent Kalman filter-based stochastic search of retroreflective markers. Although their method outperformed a deterministic search approach, there is still room for improvement.

First, we approach the problem linearly using a linear Kalman filter as our Bayesian estimator. We propose to dynamically adjust the covariances based on the source of the measurements. In this way, we indirectly integrate knowledge of the trajectory at every prediction. We expect this approach to increase the probability of locating each marker, since the sampling points would be more reliable and closer to the marker position. Furthermore, we also propose to incorporate the search time into the state of the system. By doing so, we expect the prediction to be more accurate and thus reduce the frequency of searches.

We consider first a linear model that follows the equations described in Section 2.5.1. However, experiments reported in Section 3.1.3 revealed that the assumptions made for the design of such linear model did not sufficiently hold. In particular, we observed during experimentation variations in base velocity, which was originally assumed to be relatively constant. Our hypothesis is that some acceleration that was

unaccounted for was causing drifts in both x and y coordinates of the robot's base. In order to mitigate this issue, we adapted our model to a non-linear system that exploits instead the concept of the Unscented Kalman Filter. The results provided by each model as well as comparisons against the spiral search and the state-of-the-art models are presented in Chapter 5.

3.1 System model

As previously mentioned, this thesis aims to design a mathematical model to dynamically predict the position of an A-UGV that moves along a known artifact while carrying a 6DOF robotic manipulator. The objective of the robot is to align its end effector to fiducials fixed along the corresponding artifact, intercepting them one by one.

In our model, the state $x[k]$ of the robot base at a time instant k is given by its position $(x_b[k], y_b[k])$ and its velocity $(\dot{x_b}[k], \dot{y_b}[k])$, whereas the observed measurement $z[k]$ corresponds to the observed base position when the robot intercepts a fiducial. The position of the end effector is given by $(x_e[k], y_e[k])$, with the anticipated arm displacement between the end effector and the base position denoted by the 2-D vector $u[k]$. The values of $u[k]$ are based on the arrangement of fiducials and the extent of the arm. That is,

$$x[k] = \begin{bmatrix} x_b[k] \\ y_b[k] \\ \dot{x_b}[k] \\ \dot{y_b}[k] \end{bmatrix}, \quad z[k] = \begin{bmatrix} z_x[k] \\ z_y[k] \end{bmatrix}, \quad u[k] = \begin{bmatrix} u_x[k] \\ u_y[k] \end{bmatrix}.$$

The fiducial positions are given by the array x_f:

$$x_f = \begin{bmatrix} x_{f,x}[1] & x_{f,x}[2] & x_{f,x}[3] & ... & x_{f,x}[N] \\ x_{f,y}[1] & x_{f,y}[2] & x_{f,y}[3] & ... & x_{f,y}[N] \end{bmatrix} \tag{3.1}$$

where N is the number of fiducials. In the remainder of this document, we use the index $j = 1, ..., N$ to indicate each fiducial.

For simplicity, we assume that the base of the robot moves only along the x-axis, resulting in $\dot{y_b}[k] = 0$. In the presence of noise, the base velocity over the x-axis is given by $\dot{x_b}[k] = v_b + w_x[k]$, where v_b is the user-defined velocity and $w_x \sim \mathcal{N}(0, \mathcal{Q}_x^2)$ is the corresponding noise modeled as zero-mean, normally distributed with standard deviation $\mathcal{Q}_x$. Analogously, the base velocity over the y-axis is given by $\dot{y_b}[k] = 0 + w_y[k] = w_y[k]$, where similarly $w_y \sim \mathcal{N}(0, \mathcal{Q}_y^2)$ is the velocity noise along the y-axis. Our initial state vector can then be written as

$$x[1] = \begin{bmatrix} x_i \\ y_i \\ v_b \\ 0 \end{bmatrix}, \tag{3.2}$$

where x_i and y_i are the coordinates of the initial base position, which corresponds to the location of the base where it intercepts the first fiducial. This is when the Kalman filter is initialized.

Figure 3.1 provides a representation of the scenario captured by the proposed dynamic model. The blue boxes represent the base of the robot while the black arrows represent the displacement of the arm after each prediction. The respective projections u_x, u_y of the arm on the $x-$ and the $y-$ axes are illustrated in pink, while the red circles correspond to the fixed fiducials.

We recall that the base does not stop during the search process. Thus, in some scenarios it might be necessary to skip fiducials that have not been yet intercepted but are relatively far from the base when the search is concluded. To handle such cases, the next fiducial i_k to be located is defined according to

$$i_k = \underset{i > i_{k-1}}{\operatorname{argmin}} \left(|x_{f,x}[i] - x_{f,x}[1]| - |\tilde{t}_f[i_{k-1}] v_b| \right), \tag{3.3}$$

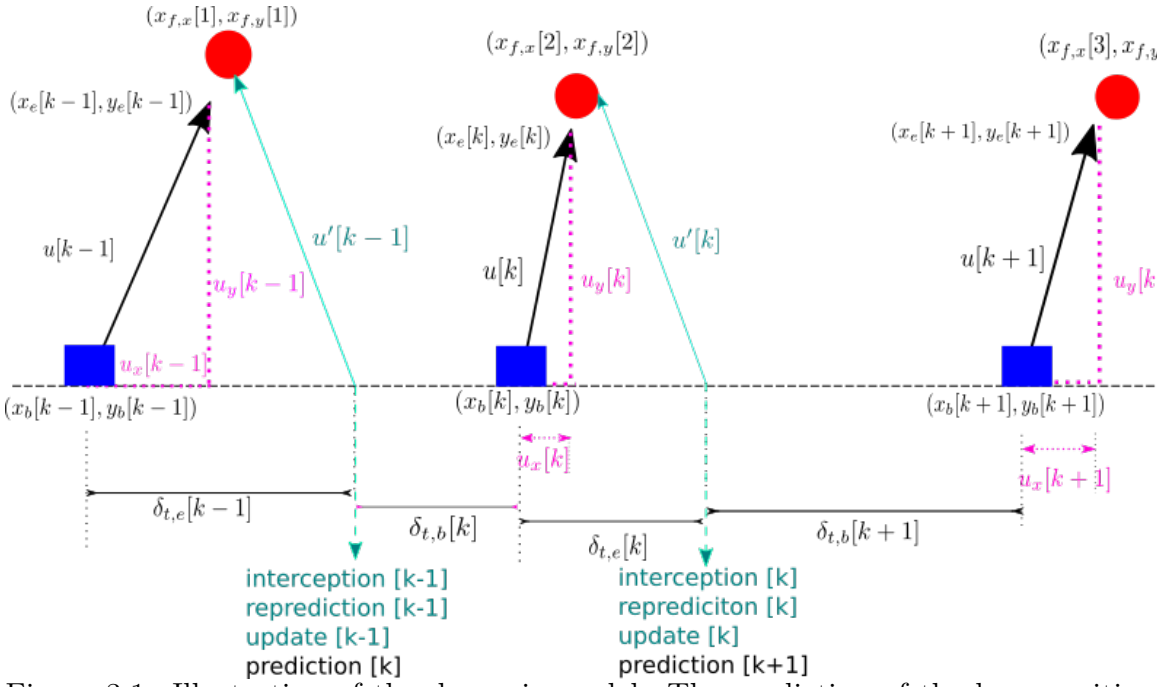


Figure 3.1: Illustration of the dynamic model. The prediction of the base position at step k takes place at the previous interception. When the mobile manipulator reaches the predicted position $(x_b[k], y_b[k])$, it starts searching for the fiducial as the base continues to move. The elapsed time to perform the search is given by $\delta_{t,e}[k]$ and the displacement of the arm after the search process is $u'[k]$.

where $\tilde{t}_f[i_{k-1}]$ is the time between the interception (or end of search) of the first fiducial and fiducial i_{k-1}, and $x_{f,x}[i]$ represents the x coordinate of the i-th fiducial. That is, i_k is the fiducial that is closest to the base after it concluded its search for the previous fiducial i_{k-1}. Similarly, the variable $\tilde{t}_f[j]$ corresponds to the absolute time to get to a fiducial j. Mathematically, this corresponds to:

$$\tilde{t}_f[j] = \frac{|x_{f,x}[j] - x_{f,x}[1]|}{|v_b|}, \qquad j = 1, \ldots, N. \tag{3.4}$$

Figure 3.2 illustrates the timeline of the events. The search process begins when the base has reached the predicted base position and starts looking for the fiducial i_k, whose identity is giving by Equation 3.3.

In our model, the observations are given by the relative position between the end effector and a fiducial marker whose location is known. The base position is then

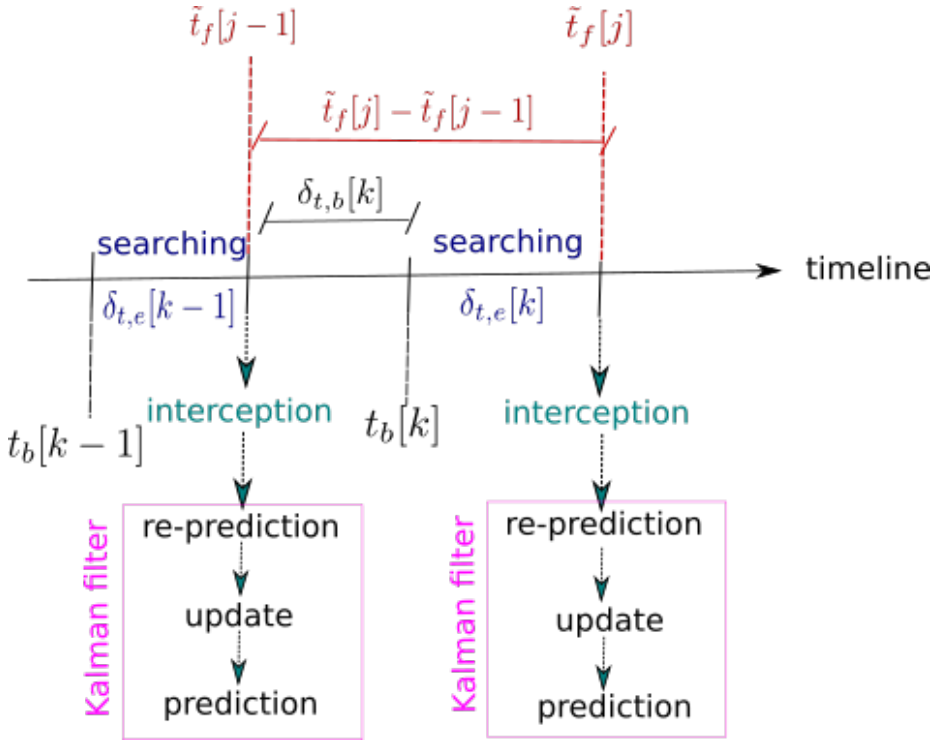


Figure 3.2: Timeline of the events of the dynamic model. The reprediction step is to account for the search time spent on the last step, $k-1$. Then the filter updates the state vector with the measurement collected at step $k-1$. Next, it predicts the next base position for step k.

estimated from the end effector position through the vector $u[k]$, which represents the displacement between the manipulator and the base. This vector can be estimated using the robot's kinematics, as shown in Equation 3.5.

$$z[k] = \begin{bmatrix} x_b[k] + u_x[k] \\ y_b[k] + u_y[k] \end{bmatrix} \tag{3.5}$$

Based on the state-space representation just described and summarized in Section 2.4 and Equation 3.1, the dynamic and observation models are then given by

$$x[k] = F_k x[k-1] + w_k, \tag{3.6}$$

$$z[k] = H_k x[k] + v_k, \tag{3.7}$$

where F_k is the state-transition matrix, H_k is the observation (measurement) matrix, and w_k and v_k are the zero-mean, normally distributed process and observation noises. Analogous to the aforementioned w_k formulation, mathematically, the observation noise is represented as $v_k \sim \mathcal{N}(0, \mathcal{R}_k^2)$, where $\mathcal{R}_k$ denotes the corresponding standard deviation at time k. Since we do not have any control acting on the system (i.e., no external force or input that affects the state), there is no control input in Equation 3.6.

3.1.1 Proposed dynamic model using linear Kalman filter

Our initial assumptions were that the base only moves in the x direction, whereas the arm can move in both x and y directions in order to locate the fiducials along the path. Under these assumptions and not having any external force and acceleration, the system can thus be modelled using a linear Kalman filter.

Since the base moves at constant velocity along x-axis, the dynamics of the predicted target state are then given by:

$$\begin{cases} x_b[k] &= x_b[k-1] + \delta_{t,b}[k]\dot{x_b}[k-1] \\ y_b[k] &= y_b[k-1] + \delta_{t,b}[k]\dot{y_b}[k-1] \end{cases} \tag{3.8}$$

Two prediction events must be taken into consideration. First, we refer simply as *prediction* to the estimation of $x[k]$ that takes place at the time $\tilde{t}_f[i_{k-1}]$ when the fiducial i_{k-1} is intercepted or the corresponding search for it has ended (i.e., no interception but all search samples have already been considered). Once the subsequent search ends at time $\tilde{t}_f[i_k]$, a second prediction named *reprediction* takes place to re-estimate the base position with the most up-to-date information available. That is, both the update step as well as the reprediction take place regardless of whether an interception occurred or not. The difference is how we collect the observation to update the state of the filter. In the case of interception, the observation is measured

based on the fiducial position and the arm displacement at the time of interception:

$$\begin{cases} z_x[k] &= x_{f,x}[i_k] - {u_x}'[k] \\ z_y[k] &= x_{f,y}[i_k] - {u_y}'[k] \end{cases} \tag{3.9}$$

In the case of failure, we simply obtain the base position from the odometry and use it as our observation. However, since the odometry is not as reliable as the manipulator's kinematics, it introduces additional noise into the system. In such cases, before updating the filter, we increase the covariance $\mathcal{R}$ of the observation noise v_k by multiplying it by a fixed scaling factor γ.

As we mentioned before, we initially assume that the robot only moves in the direction of the $x-$axis, with a constant velocity v_b. Then, as Figure 3.1 indicates, the expected time $\delta_{t,b}[k]$ between the interception of the k-th fiducial and the fiducial intercepted at step $k-1$ can be computed according to

$$\delta_{t,b}[k] = \frac{|(x_{f,x}[i_k] - u_x[k]) - (x_{f,x}[i_{k-1}] - u'_x[k-1])|}{|v_b|}. \tag{3.10}$$

Once a fiducial is intercepted, a *prediction* step is performed based on the time $\delta_{t,b}[k]$. That is, the state transition matrix F_k for the *prediction* step is given by

$$F_k = \begin{bmatrix} 1 & 0 & \delta_{t,b}[k] & 0 \\ 0 & 1 & 0 & \delta_{t,b}[k] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.11}$$

After the search process for the fiducial i_k ends, the *reprediction* step is performed using the interval $\delta_{t,e}[k]$ to correct for the motion of the base during that period. That is, the state transition matrix takes the form

$$F_k = \begin{bmatrix} 1 & 0 & \delta_{t,e}[k] & 0 \\ 0 & 1 & 0 & \delta_{t,e}[k] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.12}$$

where the search-time $\delta_{t,e}[k]$ now replaces the $\delta_{t,b}[k]$ term used in the previous *prediction* step.

In Figure 3.1, $u[k] = (u_x[k], u_y[k])$ and $u'[k] = (u'_x[k], u'_y[k])$ correspond to the position of the arm before and after the search process, respectively. Whereas $u'_x[k-1]$ can be obtained from the kinematics of the manipulator, $u_x[k]$ must be determined according to the position of the next fiducial. Once the fiducial i_k has been determined according to Equation 3.3, the displacement $u_x[k]$ of the manipulator with respect to the base can be determined based on the position of i_k and the position of the base once it concluded the search for fiducial i_{k-1}. That is,

$$u_x[k] = \text{sign}(v_b) \cdot min(u_{x-max}, |x_{f,x}[i_k] - (x_i + \tilde{t}_f[i_{k-1}]v_b)|), \tag{3.13}$$

where x_i represents the x coordinate of the initial position of the base, u_{x-max} is the maximum allowable displacement of the manipulator, and the term $\text{sign}(v_b)$ accounts for the fact that the direction of motion of the manipulator depends on the direction of motion of the base. Since we assume the velocity along the y direction is zero, it follows that

$$u_y[k] = x_{f,y}[i_k] - y_i, \tag{3.14}$$

where y_i is the y coordinate of the initial position of the base.

3.1.2 Proposed observation model using linear Kalman filter

Once a fiducial is intercepted, we can measure the approximate position of the base according to the known position of the fiducial and the corresponding displacement of the manipulator, i.e.,

$$z[k] = x_f[i_k] - u'[k]. \tag{3.15}$$

Since the interception takes place $\delta_{t,e}[k]$ seconds after the predicted base position, this observation is equivalent to

$$z[k] = x_b[k] + \delta_{t,e}[k]\dot{x}_b[k]. \tag{3.16}$$

The *reprediction* step of our algorithm already accounts for the search time $\delta_{t,e}[k]$. Thus, it follows that our observation matrix is given simply by

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

3.1.3 Limitations of the linear model

During the experimental evaluation, we noticed that there is a drift in the motion of the base which leads to a drift in the end effector as well, as the arm is mounted on top of the base and thus is subject to any drift of the base. As seen in Figures 3.3 and 3.4, there is a substantial amount of motion along the y-axis, i.e., the drift is significant and causes the assumptions underlying the linear model to be violated. This led us conclude that our model is not sufficient to address the actual characteristics of the problem and we mitigated this issue by designing a non-linear model of the system.

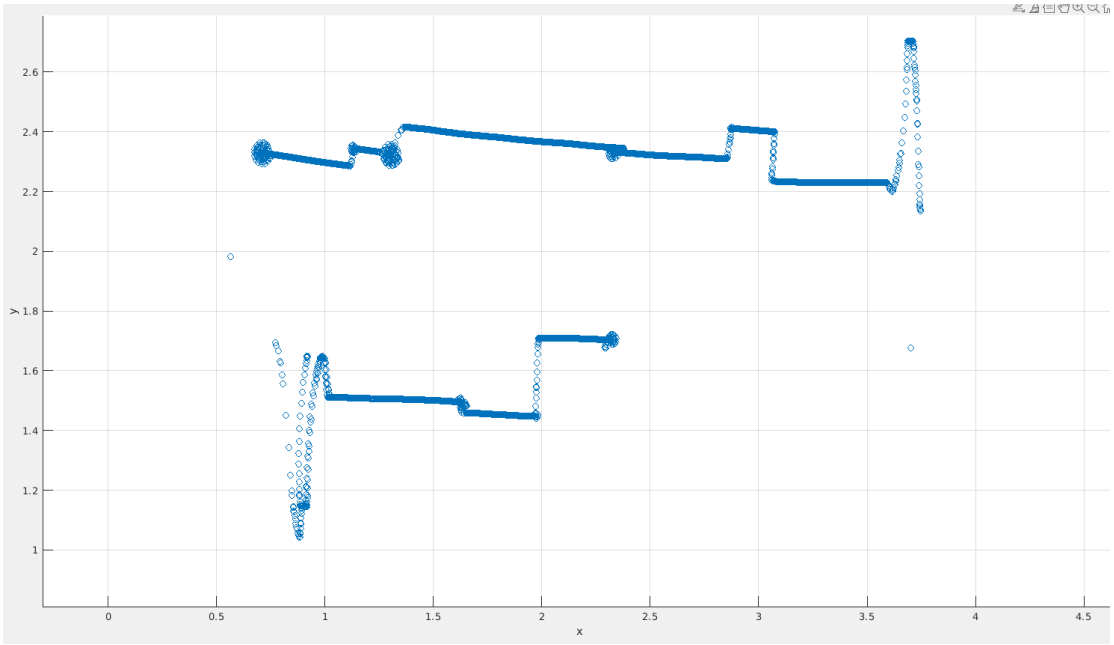


Figure 3.3: Trajectory of end effector positions during an experiment using the spiral search method described in Section 2.2.

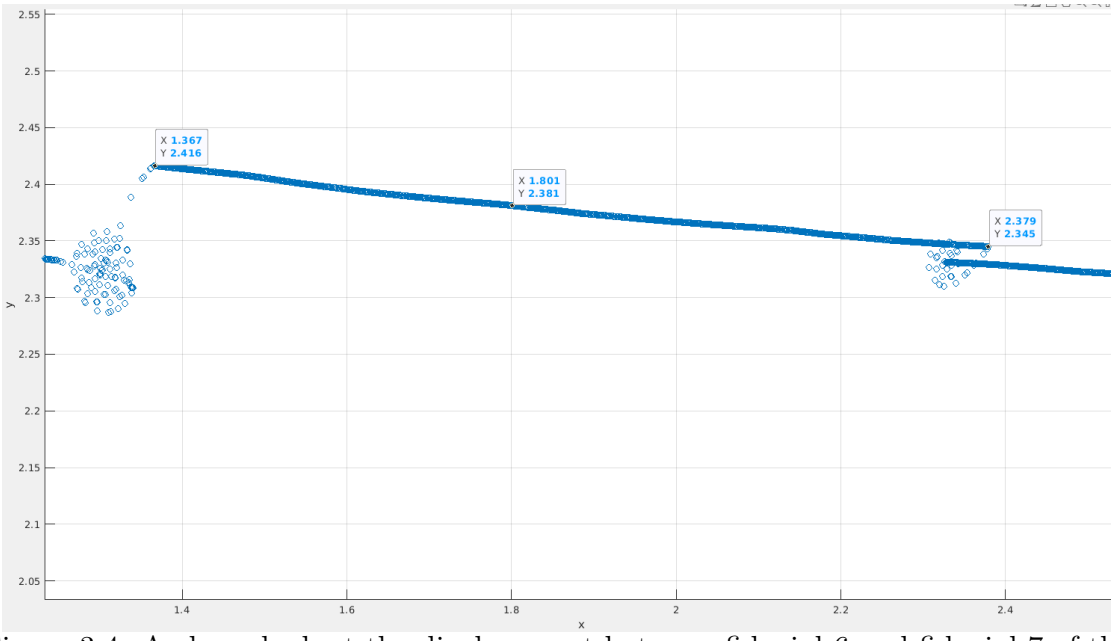


Figure 3.4: A closer look at the displacement between fiducial 6 and fiducial 7 of the arrangement 4.9 during spiral search.

3.1.4 Proposed dynamic model using Unscented Kalman filter

Due to the significant drifts illustrated in Figures 3.3 and 3.4, we noticed that the base does not move in parallel with the x-axis as we had initially assumed. As a consequence, we see a remarkable variation of $\dot{x}_b[k]$ over time, which leads to inaccurate predictions of $\delta_{t,b}$. Hence, we considered addressing this problem using a UKF.

The usage of the estimated base velocity $\dot{x}_b[k]$ in the prediction step reflects better the problem peculiarities. For instance, in the linear model we could not make use of the base velocity $\dot{x}_b[k]$ when predicting $\delta_{t,b}$ in order to maintain the linearity of the system. Hence, the predicted $\delta_{t,b}$ was based on a constant base velocity whereas the actual velocity changes over time.

Furthermore, in Equation 3.10, we were using $x_{f,x}[i_{k-1}] - u'_x[k-1]$ as the current base position, which propagates any kinematics or odometry error impacting $u'[k]$. To better estimate the base position after the search process we can leverage the state values after reprediction, since this step compensates for the search time during which the base continues to move. In contrast to the linear formulation, the UKF-based model makes it possible to use the current state vector to compute $\delta_{t,b}$. Under these new conditions, we thus reformulate Equation 3.10 as

$$\delta_{t,b}[k] = \frac{|(x_{f,x}[i_k] - u_x[k]) - x_b[k]|}{|\dot{x}_b[k-1]|}. \tag{3.17}$$

Another significant change when transitioning from the linear model to the non-linear one is the definition of $u[k]$. Since we can leverage the state vector in the prediction step, Equations 3.13 and 3.14 can be replaced by

$$\begin{cases} u_x[k] &= \text{sign}(v_b) \cdot min(x_{max}, |x_{f,x}[i_k] - x_b[k-1]|) \\ u_y[k] &= \text{sign}(v_b) \cdot min(y_{max}, |x_{f,y}[i_k] - y_b[k-1]|) \end{cases}. \tag{3.18}$$

All the other equations remain the same as introduced for the linear formulation. Thus, the proposed dynamic problem can be written as

$$\begin{bmatrix} x_b[k] \\ y_b[k] \\ \dot{x_b}[k] \\ \dot{y_b}[k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta_{t,b}[k] & 0 \\ 0 & 1 & 0 & \delta_{t,b}[k] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_b[k-1] \\ y_b[k-1] \\ \dot{x_b}[k-1] \\ \dot{y_b}[k-1] \end{bmatrix} + w_k \tag{3.19}$$

$$\begin{bmatrix} x_b[k] \\ y_b[k] \\ \dot{x_b}[k] \\ \dot{y_b}[k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta_{t,e}[k] & 0 \\ 0 & 1 & 0 & \delta_{t,e}[k] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_b[k] \\ y_b[k] \\ \dot{x_b}[k] \\ \dot{y_b}[k] \end{bmatrix} + w_k. \tag{3.20}$$

The corresponded observation model is then given by

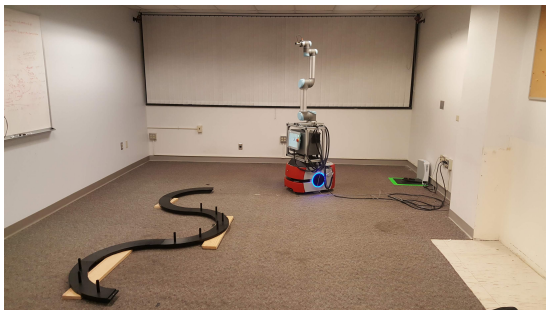
$$\begin{bmatrix} z_x[k] \\ z_y[k] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_b[k] \\ y_b[k] \\ \dot{x_b}[k] \\ \dot{y_b}[k] \end{bmatrix} + v_k, \tag{3.21}$$

where $\delta_{t,b}$ is described in Equation 3.17 and $\delta_{t,e}$ is measured and collected during simulation.

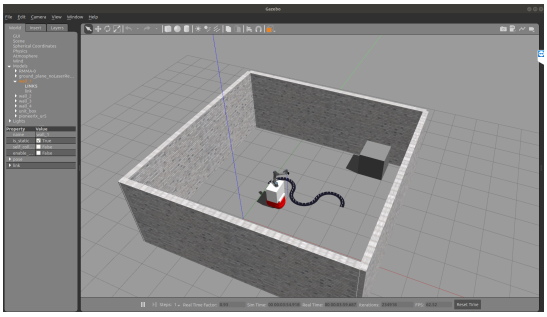
CHAPTER 4
DATA COLLECTION SYSTEM

In this chapter, we present the robot model used for our data collection system, followed by an explanation of how we implemented our model in ROS, as well as a description of the experimental setup, which includes the different fiducial arrangements used to evaluate the robustness of our model. Finally, we provide a discussion about the parameters used for fine tuning the system model.

The mobile manipulator used for data collection is based on a Universal Robotics UR5 robotic manipulator attached to an Adept Pioneer LX mobile base (Figure 4.1a), which we also refer to as the automated guided vehicle (AGV). For a fully functional simulation of this system, we have designed and implemented a simulation model in the Gazebo environment (Figure 4.1b).



(a) Pioneer LX + Universal Robotics UR5.



(b) The simulated version of Pioneer LX in Gazebo.

Figure 4.1: Our robot used to collect data, Pioneer LX (a) and the correspondent robot in Gazebo.

4.1 ROS module implementation

Figure 4.2 summarizes the implementation of the proposed system. The UR5 manipulator is controlled by the MoveIt! planning libraries, available in the ROS-Industrial package [41]. The base is controlled using the interfaces provided by the native navigation stack available in ROS. The navigation stack allows the robot to perform self-localization using its two-dimensional laser scanner, making it possible to programatically send the robot to predefined goal positions while maintaining our ability to control its linear and angular velocities. Finally, these modules are then controlled by a Python script, which incorporates a motion model based on the UKF also implemented in Python.

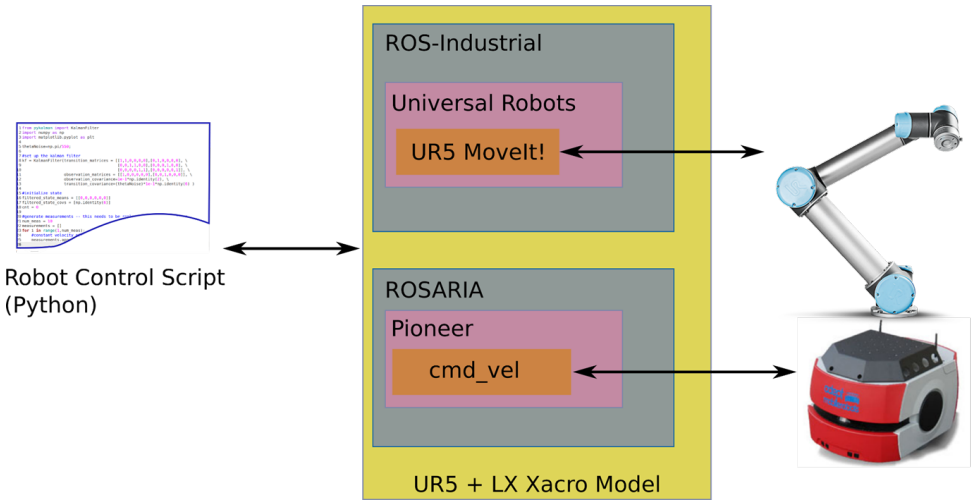


Figure 4.2: Architecture of the the software stack for the mobile manipulator in the simulation environment. Source: [1].

Figure 4.3 shows the overall ROS node structure of the system. Motion planning is performed by the *move base* ROS package. An occupancy cost map is generated based on the stationary map as well as on the laser scan observations collected by the robot. This cost map is used by a grid-based global trajectory planning method

that uses Dijkstra's algorithm to compute the optimal robot trajectory to a destination node as well as trajectories produced by a local planner based on the Dynamic Window Approach [42]. The remaining nodes in the graph are responsible for the control of the manipulator (*move group* and *robot state publisher*) and the hardware interface (*RosAria* and *ur driver*).

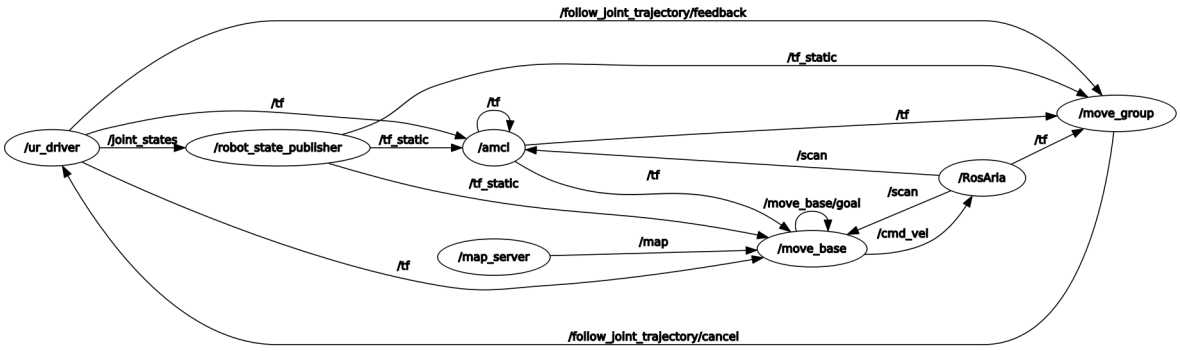


Figure 4.3: ROS node graph showing the software structure of the robotic platform. Base navigation is performed by the *map server, amcl*, and *move base* nodes. The *move group* and *robot state publisher* nodes are responsible for the control of the manipulator. The interface with the hardware is performed by the *RosAria* (base) and *ur driver* (manipulator) nodes.

Map Building

Figure 4.4 shows the simulation environment considered in this work and the corresponding map obtained by the system using the laser scanner attached to the robot. To incorporate the mapping functionalities into our simulation environment, we extended our model to include a two-dimensional laser scanner based on the ROS packages for SICK laser scanners[1]. Our model is compatible with the ROS navigation stack and hence can be utilized in conjunction with any ROS-based mapping library.

[1]http://wiki.ros.org/sick scan

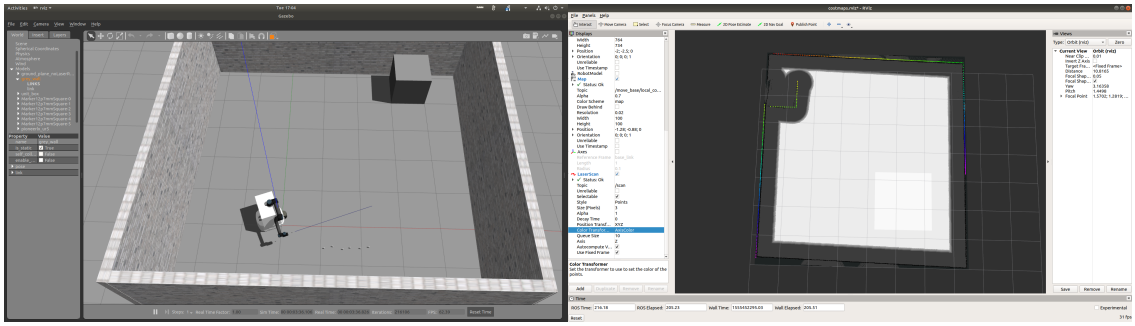


Figure 4.4: Simulation environment incorporating mapping functionalities. *Left:* the simulation environment exploited in this thesis, which includes walls and a disambiguation object (box on the top right corner) to allow robot localization. *Right:* the map generated for this corresponding environment using the corresponding two-dimensional laser scan (green structure aligned with the boundaries of the map in the picture). The gray region corresponds to areas of high collision probability, as estimated by the navigation algorithms.

The static navigation map is constructed using the ROS gmapping package[2], which provides a laser-based simultaneous localization and mapping (SLAM) algorithm. The map is generated by collecting the laser scans obtained by the robot as it moves around the environment.

Localization and Motion Planning

The command requests to the navigation stack are performed through the simple ROS action client shown in Figure 4.5. The robot pose is received using a ROS tf listener[3].

Once a stationary map is created, the robot can localize itself using the *amcl* (Adaptive Monte Carlo Localization) ROS package. More specifically, using the corresponding stationary map provided by a map server ROS node, this package exploits the adaptive sequential Monte Carlo approach proposed in [43] to estimate the robot pose. Figure 4.6 illustrates the samples of the robot pose produced by *amcl*. As

[2]http://wiki.ros.org/gmapping
[3]http://wiki.ros.org/tf

```
def gotoPose(pose):
    goal_pose = MoveBaseGoal()

    goal_pose.target_pose.header.frame_id = 'map'
    goal_pose.target_pose.pose.position.x = pose[0][0]
    goal_pose.target_pose.pose.position.y = pose[0][1]
    goal_pose.target_pose.pose.position.z = pose[0][2]

    (goal_pose.target_pose.pose.orientation.x, \
     goal_pose.target_pose.pose.orientation.y, \
     goal_pose.target_pose.pose.orientation.z, \
     goal_pose.target_pose.pose.orientation.w) = \
        quaternion_from_euler(pose[1][0], pose[1][1], pose[1][2])

    client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
    client.wait_for_server()
    client.send_goal(goal_pose)
    client.wait_for_result()
```

Figure 4.5: ROS action client for communication with the navigation stack.

the figure indicates, as the robot moves and collects additional observations using its laser scanner, the more accurate pose estimates are obtained.

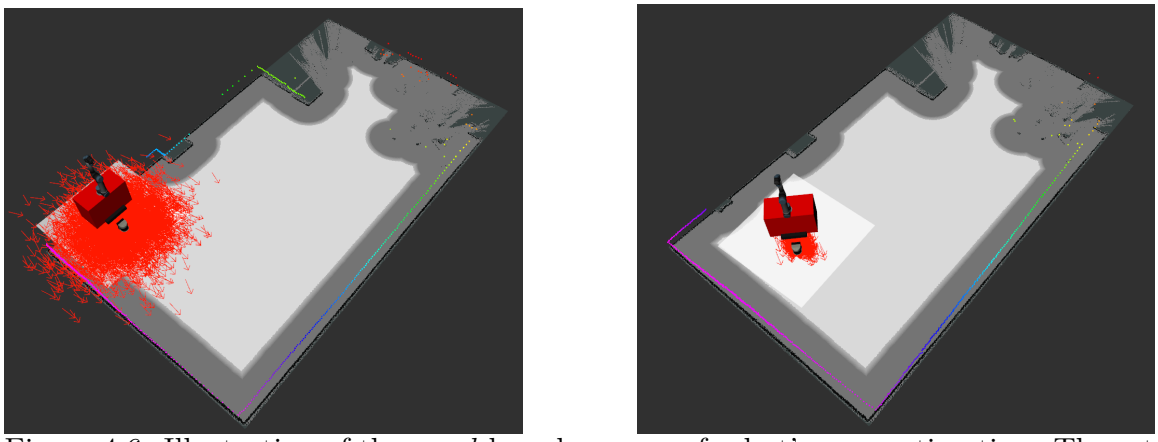


Figure 4.6: Illustration of the *amcl*-based process of robot's pose estimation. The set of vectors in red represent the robot's pose samples generated by the *amcl* node. As the robot moves from the position shown on the left figure to the position on the right figure, the localization algorithm refines its estimates and thus leads to the observed reduction in the variance of the samples around the robot position.

4.2 Robot calibration procedure

Once the robot reaches its initial position, using the ROS action client described in Section 4.1, we command the base to move along a known path with a constant velocity along the direction of the x axis and heading angle θ_b. We recall the fact that the base does not stop moving unless it is on registration mode, which is used to stop the robot to ensure the first fiducial is located. As explained in Section 3.1, this first observation is necessary for initialization of the Kalman filter.

In terms of notation, we recall that the arrangement of fiducials is framed as a set with $j = 1, ..., N$ fiducials distributed at the positions with coordinates $(x_{f,x}[j], x_{f,y}[j])$. While the base moves along the x-axis, the arm moves at each step according to the predictions generated by the dynamic model in an attempt to intercept the fiducials. Once the base reaches the predicted fiducial location, a search process is triggered unless the fiducial is immediately located. The search method depends on the pre-defined procedure: stochastic or determinic.

After the search, if there is an interception, then an observation (base position) is computed as the difference between the position of the fiducial and the arm displacement in which the end effector intercepted the marker. Otherwise, the observation is collected from the base's odometry. Together with the observation measurements, the search time and base position error are also recorded. The flowchart in Figure 4.7 summarizes the approach in which we integrated the Python script for robot control with the mathematical model using UKF. The first part of the flowchart corresponds to registration, in which we perform 5 attempts to intercept the first fiducial with the base stopped. We have to make sure that the first fiducial is found because we need to initialize the filter with this observation.

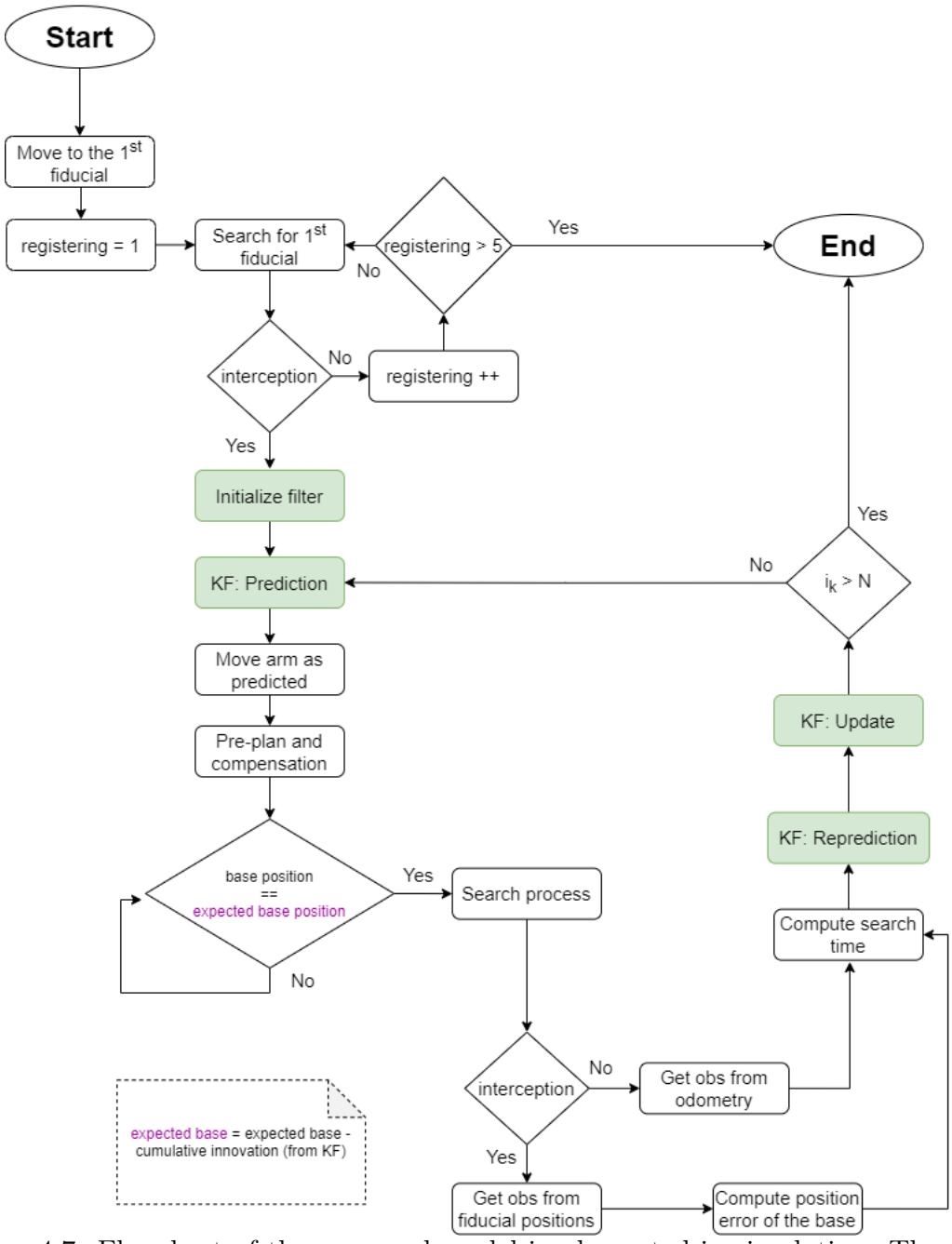


Figure 4.7: Flowchart of the proposed model implemented in simulation. The green boxes correspond to the motion module, implemented separately from the rest of the code.

4.3 Configuration of the simulated platform

Our simulation experiments consisted of evaluating the proposed dynamic model system using the Robot Operating System (ROS) and the Gazebo simulator. Since ROS is the current de-facto standard development tool for robotics, we opt for this framework with the intent of increasing the access to our work by researchers and practitioners in this field. In addition to the mobile manipulator, we integrated a model of the NIST Reconfigurable Mobile Manipulator Artifact (RMMA) into our simulation environment. The test points or fiducials consist of retroreflective markers embedded in the RMMA. While the base moves along a pre-defined test path, the arm must align a laser pointer to a flat reflector within each fiducial in sequence until the entire artifact has been observed. In order to detect the fiducials, a laser sensor is mounted to the end effector. Figure 4.8 illustrates our complete simulation enviroment.

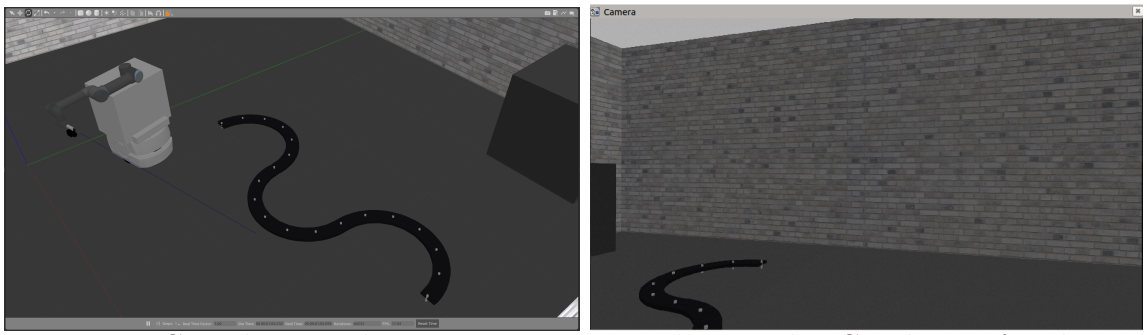


Figure 4.8: Complete simulation environment implemented in Gazebo. *Left:* The simulation environment containing the corresponding mobile manipulator, the RMMA and the disambiguation object *Right:* An illustration of the corresponding view provided by the camera mounted to the end effector of the mobile manipulator.

4.4 Experimental setup

Our experimental setup for Gazebo simulations follows the procedures described in [1]. We first initialize the robot at a fixed position in which the arm can reach the first fiducial in the test sequence. For our main set of experiments, the fiducials are arranged as in Figure 4.9, with corresponding positions with respect to the global coordinate frame shown in Table 4.1. For a fair comparison of results with the work described in [1], we perform additional experiments using the fiducial arrangement described in [1]. This arrangement corresponds to Arrangement #2 (Figure 4.10) in this thesis. Furthermore, we also use the arrangement in which the fiducials are placed along the RMMA, as illustrated in Figure 4.11.

Figure 4.9: Fiducial Arrangement #1, with corresponding positions listed in Table 4.1.

Table 4.1: Arrangement #1: Positions of the fiducials as arranged in Figure 4.9.

$x_f[i]$	1	2	3	4	5	6	7	8	9
$x_{f,x}[i]$ (m)	1.62	1.97	2.31	3.09	2.88	2.34	1.35	1.11	0.72
$x_{f,y}[i]$ (m)	1.51	1.46	1.71	2.23	2.4	2.31	2.25	2.33	2.29

Each simulation run is performed for 5 different base velocities, ranging from 1.0 cm/s to 3.0 cm/s with a step of 0.5 cm/s. We repeated each of these runs 10 times. The computer used to run all the simulations was an HP Z240 with an Intel(R) Core(TM) i7-6700 CPU (3.40GHz) and 16 Gb of RAM. The ROS distribution used was the *melodic*.

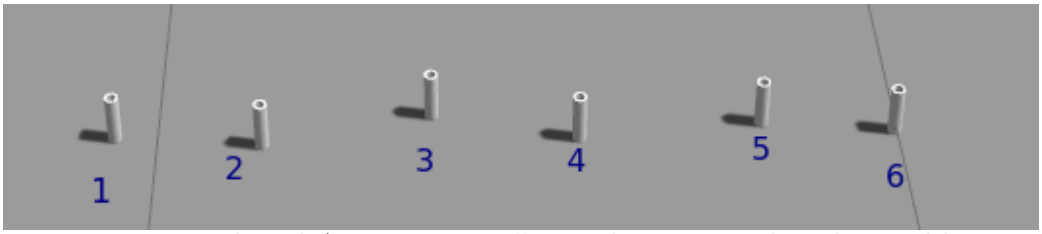


Figure 4.10: Fiducial Arrangement #2, with positions listed in Table 4.2.

Table 4.2: Arrangement #2: Positions of the fiducials as arranged in Figure 4.10.

$x_f[i]$	1	2	3	4	5	6
$x_{f,x}[i]$ (m)	0.94	1.14	1.37	1.57	1.82	2.00
$x_{f,y}[i]$ (m)	0.34	0.32	0.37	0.32	0.34	0.32

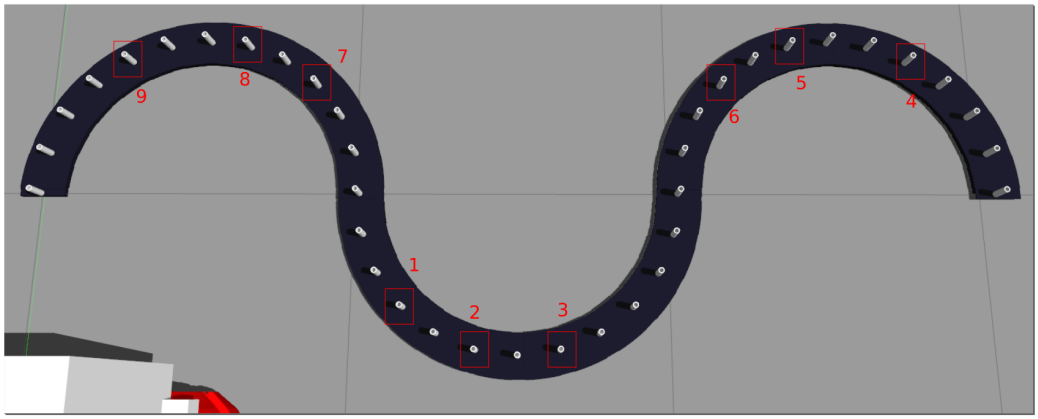


Figure 4.11: Fiducial Arrangement #3, with positions listed in Table 4.3.

Table 4.3: Arrangement #3: Positions of the fiducials as arranged in Figure 4.11.

$x_f[i]$	1	2	3	4	5	6	7	8	9
$x_{f,x}[i]$ (m)	1.16	1.39	1.66	2.79	2.41	2.18	0.87	0.50	0.25
$x_{f,y}[i]$ (m)	1.64	1.51	1.51	2.44	2.49	2.36	2.36	2.51	2.44

4.5 Fine tuning of the system model

In order to optimize the performance of our model, we fine-tuned the Kalman filter parameters as well as additional parameters that impact the search process. Such parameters of interest are described below. The performance of the system for different values of the parameters is presented in Table 5.3 of Section 5.5. The fine-tuning experiments were repeated 5 times for each configuration.

Sample gap: This parameter represents the quantization step for the end-effector pose samples generated by the search procedure. Hence, it affects the density of the search points. The need to fine tune this parameter can be observed in

Figure 2.4.

Covariance $\mathcal{Q}$ of the process noise: As the nomenclature suggests, this parameter refers to how much noise the filter assumes is present in the process of predicting the next state of the robot. The greater this value, the higher the covariance P associated to the estimation at each prediction.

Covariance $\mathcal{R}$ of the observation noise: This variable impacts the weight given by the filter to the current measurement during the update of the robot state vector. The lower this value, the more reliable is the source from which the filter collects the observation data. We recall that in the case of fiducial interception, the observation used during update corresponds to the base position taken from the detected fiducial through the robot's kinematics. If the search ends without an interception, the observation is simply the base position obtained from its odometry after the search ends. Since the measurement using the fiducial positions is more reliable than the one corresponding to the odometry, we increase the covariance $\mathcal{R}$ for those failure cases. In Table 5.3 of Section 5.5, these parameters are distinguished from each other by the subscript "failure" and "interception".

Initial covariance $\mathcal{P}$ of the estimate noise: This initial covariance is a measure of how accurate the first observation is. It is used to initialize the filter and thus affects the functionality of the filter, since a poor initial covariance will be cascaded through the whole chain of the prediction-update loop . In a scenario where the filter is well modeled, the covariance $\mathcal{P}$ associated with the estimates from the filter should decrease over time. Otherwise, the filter's performance may decrease as its predictions are accompanied by higher uncertainties.

CHAPTER 5
EXPERIMENTAL RESULTS AND ANALYSIS

To assess the overall performance and the robustness of the proposed dynamic model, we performed experiments using different filter parameters, fiducial arrangements, and base velocities. In addition to the interception rate, the search time and the position error were also computed as evaluation metrics. For each of the considered metrics, results obtained with our models are compared to traditional deterministic search methods.

5.1 Evaluation metrics

The metrics used to determine the precision and efficacy of the proposed model are listed as follows:

Interception rate: The total number of intercepted fiducials across repeated runs of the experimental procedure, divided by the total number of experiments performed (e.g., *number of different base velocities* $\times$ *number of runs*).

Average search time: The sum of all the search times divided by the total number of searches performed over multiple experimental runs (e.g., *number of fiducials* $\times$ *number of base velocities* $\times$ *number of runs*).

Position error of the base: The difference between the observed base position and the corresponding prediction.

5.2 Experimental results

The results in this section are divided into three groups. First, we provide a comparison between the performance of the linear Kalman filter and the UKF. We then contrast the search performance under different fiducial arrangements. Finally,

the last set of experiments focuses on the effects of fine-tuning the UKF parameters. In Section 5.7, we provide a discussion of the results presented in this section.

5.3 UKF vs. linear Kalman filter

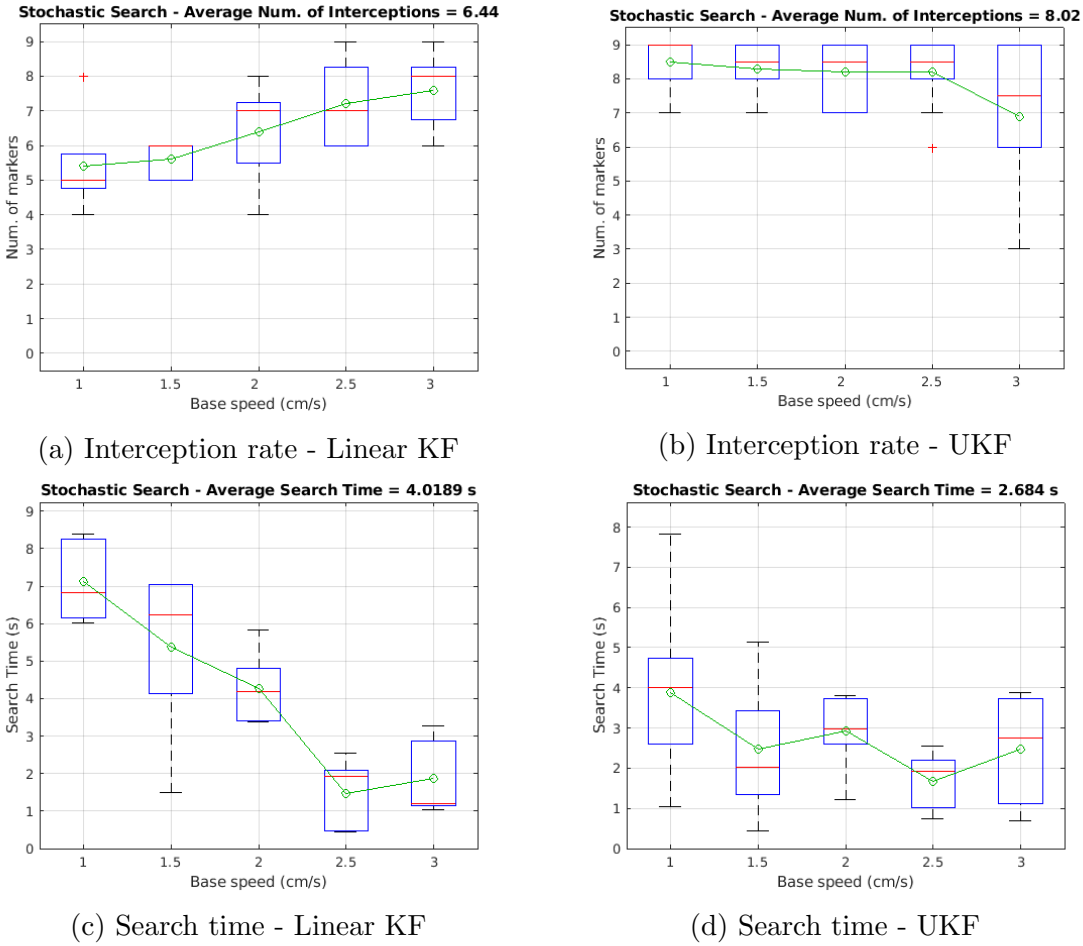
Experiments comparing the linear and non-linear (i.e., UKF) versions of the Kalman filter were carried out for fiducial Arrangement #1, which consists of 9 fiducials (3 on the first side + 6 on the second side) distributed as shown in Figure 4.9. For both methods, the corresponding KF was initialized independently for each side, such that error from the first side does not accumulate and impact the performance on the second side. The common Kalman filter parameters were set to $\mathcal{Q} = 6 \times 10^{-8}$, $R_{\text{interception}} = 4 \times 10^{-8}$, and $R_{\text{failure}} = 4 \times 10^{-4}$, with the *samplegap* search parameter set to 0.5. Interception rates and search times as a function of base speed for both models are presented in Figure 5.1, and the results are summarized in Table 5.1.

Method	Interception [%]	Search time [s]
Linear KF	71.55	4.02
UKF	89.11	2.68

Table 5.1: Average interception rate and average search time achieved by the linear Kalman filter and the fine-tuned UKF under Arrangement #1.

5.4 UKF vs. spiral search under different fiducial arrangements

The experiments described in this section were carried out with the same Kalman filter parameters listed in Section 5.3, but with the three different fiducial arrangements illustrated in Figures 4.9, 4.10, and 4.11. The primary goal of these experiments is to evaluate the robustness of the proposed model to different fiducial



(a) Interception rate - Linear KF

(b) Interception rate - UKF

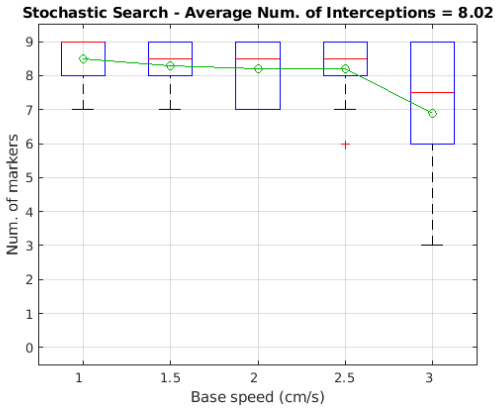
(c) Search time - Linear KF

(d) Search time - UKF

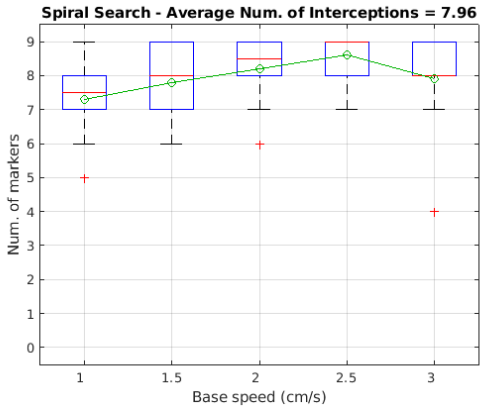
Figure 5.1: Comparison between the linear Kalman Filter and the fine-tuned UKF in terms of interception rate and search time for Arrangement #1.

arrangements. Moreover, we compare our model with the spiral search method proposed in [26]. Figures 5.2, 5.3 and 5.4 compare the performance of our fine-tuned model against the spiral method for each fiducial arrangement. Table 5.2 contrasts the overall performance of both methods for each arrangement.

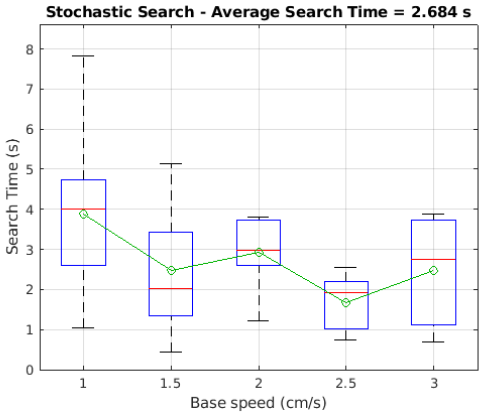
We noticed that there is some variability in the experimental runs. Specifically, when re-initializing ROS/Gazebo between runs, we observed a significant fluctuation in the interception rates obtained in our experiments. Figure 5.5 shows the interception rates of our model and the spiral search strategy for different runs of the experiment.



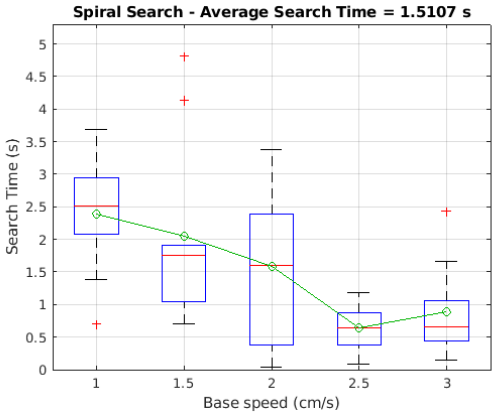
(a) Interception rate - UKF



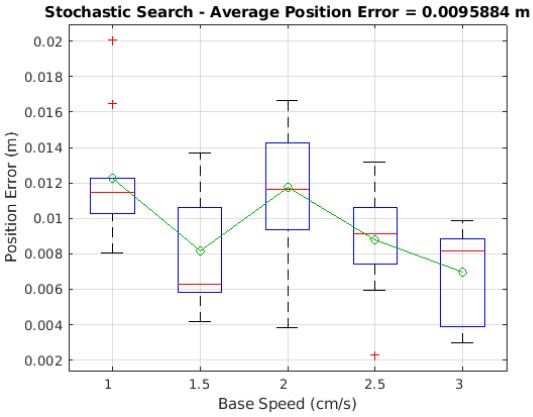
(b) Interception rate - Spiral



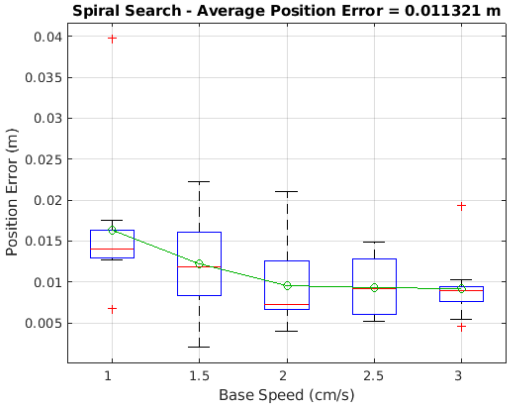
(c) Search time - UKF



(d) Search time - Spiral

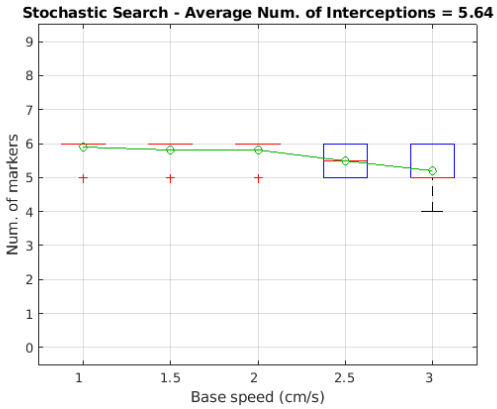


(e) Position error - UKF

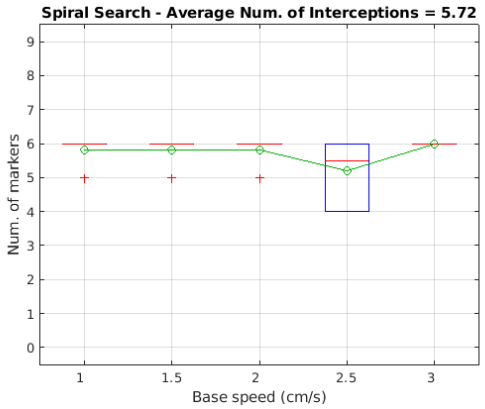


(f) Position error - Spiral

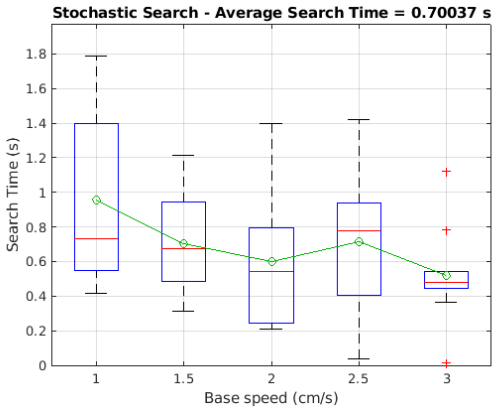
Figure 5.2: Performance comparison between the optimal fine-tuned UKF against the spiral search method for Arrangement #1.



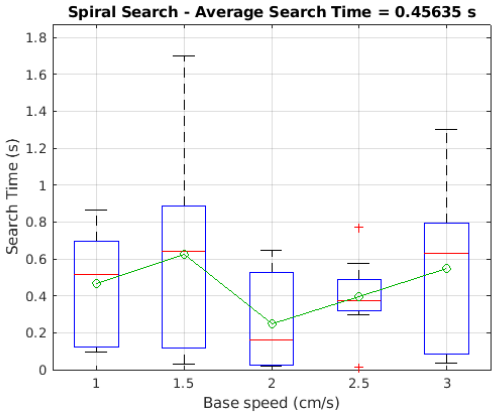
(a) Interception rate - UKF



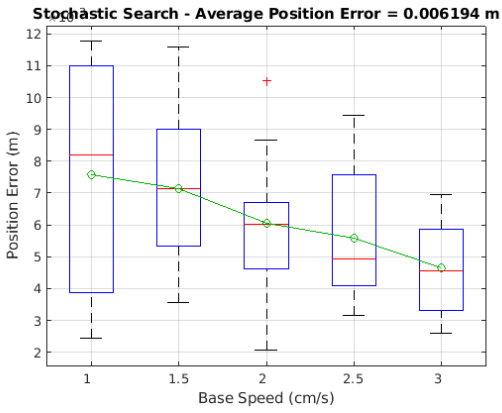
(b) Interception rate - Spiral



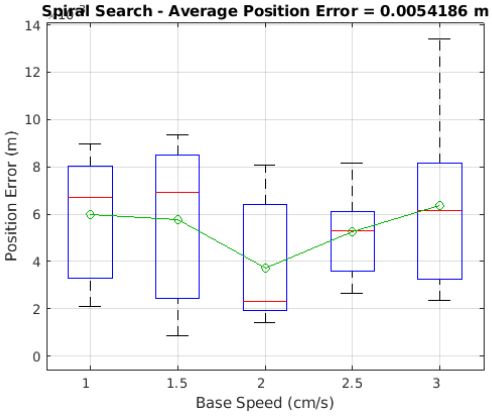
(c) Search time - UKF



(d) Search time - Spiral

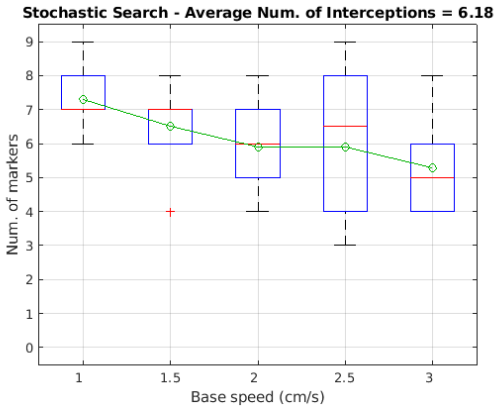


(e) Position error - UKF

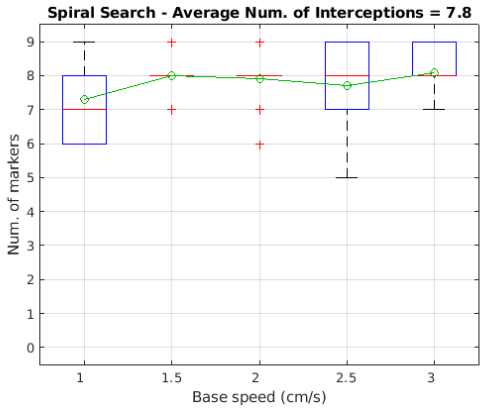


(f) Position error - Spiral

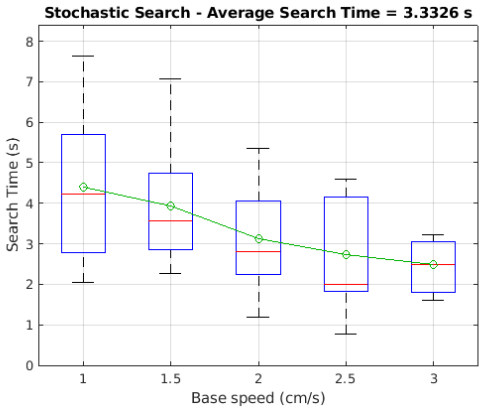
Figure 5.3: Performance comparison between the optimal fine-tuned UKF against the spiral search method for Arrangement #2.



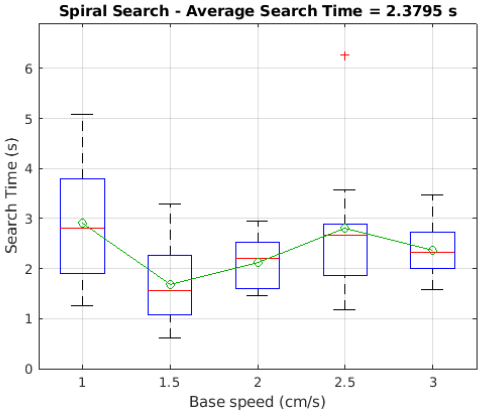
(a) Interception rate - UKF



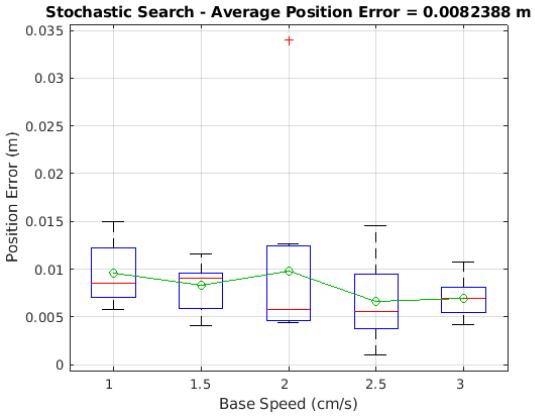
(b) Interception rate - Spiral



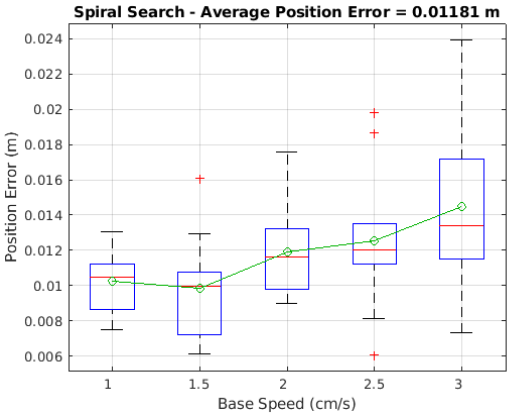
(c) Search time - UKF



(d) Search time - Spiral



(e) Position error - UKF



(f) Position error - Spiral

Figure 5.4: Performance comparison between the optimal fine-tuned UKF against the spiral search method for Arrangement #3.

Arrangement	Method	Interception [%]	Search time [s]	Position error [mm]
#1	Spiral search	88.44	1.51	11.32
	UKF	89.11	2.68	9.59
#2	Spiral search	95.33	0.46	5.42
	UKF	94.00	0.70	6.19
#3	Spiral search	86.67	2.38	11.81
	UKF	68.67	3.33	8.24

Table 5.2: Performance under different fiducial arrangements and search methods.

5.5 Fine-tuning of the model

The experiments described in this section were carried out using fiducial Arrangement #1 at first. The same configurations used to fine tune the system model under this arrangement are then used to fine tune the model under Arrangements #2 and #3. The experimental procedure was the same as the one described in Section 5.3 but with the difference that we now vary the *samplegap*, $\mathcal{Q}$, $R_{interception}$, $R_{failure}$ and $\mathcal{P}$ parameters of the Kalman Filter described in Section 4.5. As Table 5.3 indicates, for each set of experiments, we change one parameter at a time. In order to accelerate the data collection process, the fine-tuning experiments were carried out using 5 runs for each configuration, with the exception of our original configuration #1 (10 runs). The best model achieved by each arrangement was repeated 5 additional times for a fair comparison with the spiral method and with our original configuration. That is, for Arrangement #1, Configuration #6 showed the best performance, thus we deemed this configuration #6_A, and #6_B corresponds to five additional runs of the same configuration.

Figure 5.2 shows the performance of our optimal fine-tuned UKF model against the spiral search method, when deployed in Arrangement #1. Figure 5.6 illustrates in more details the interception per fiducial for the best configuration we achieved

	Configuration							
Parameters	#1	#2	#3	#4	#5	#6_A	#6_B	#7
samplegap	0.5	1.0	1.5	0.5	0.5	0.5	0.5	1.0
$\mathcal{Q}$	6e-8	6e-8	6e-8	4e-8	1e-7	6e-8	6e-8	6e-8
$R_{interception}$	4e-8	4e-8	4e-8	4e-8	4e-7	4e-8	4e-8	4e-8
$R_{failure}$	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
$\mathcal{P}$	6e-8	6e-8	6e-8	4e-8	6e-8	1e-7	1e-7	1e-7
Interception	78.89%	85.78%	81.78%	78.22%	68.00%	90.67%	89.11%	81.78%
Search time [s]	3.41	2.15	2.04	3.86	4.32	2.52	2.68	2.52
Base error [mm]	8.69	9.09	10.89	8.42	6.97	9.96	9.59	10.35

Table 5.3: Fine tuning of UKF and search parameters for Arrangement #1.

after fine-tuning the parameters.

After fine-tuning, the UKF outperformed the spiral search method for Arrangement #1 in terms of interception rate and position error, which motivated us to use the same configuration of parameters for Arrangements #2 and #3. Figure 5.7 shows the performance for these two arrangements. As the figure indicates, the optimal set of parameters for one fiducial arrangement does not necessarily perform well for different arrangements. That is, model fine-tuning is needed for each arrangement.

We also explored the same configurations (from #1 to #6) as in Table 5.3 under Arrangements #2 and #3. The results are shown in Tables 5.4 and 5.5, respectively. And the comparison between the fine-tuned model of UKF with spiral search are shown in Figures 5.3 and 5.4.

5.6 Performance as a function of distance between fiducials

As Tables 5.6 and 5.8 indicate, in Arrangements #1 and #3 (Figures 4.9 and 4.11), the distance between fiducials 6 and 7 is much higher than the average inter-fiducial distance. This implies a larger gap for the robot to accumulate uncertainties. That is, by the time the robot reaches fiducial 7, its actual state differs significantly

	Configuration						
Parameters	#1	#2_A	#2_B	#3	#4	#5	#6
samplegap	0.5	1.0	1.0	1.5	0.5	0.5	0.5
$\mathcal{Q}$	6e-8	6e-8	6e-8	6e-8	4e-8	1e-7	6e-8
$R_{interception}$	4e-8	4e-8	4e-8	4e-8	4e-8	4e-7	4e-8
$R_{failure}$	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
$\mathcal{P}$	6e-8	6e-8	6e-8	6e-8	4e-8	6e-8	1e-7
Interception	91.33%	95.00%	94.00%	85.00%	86.67%	92.67%	86.67%
Search time [s]	1.12	0.66	0.70	0.76	1.12	1.35	1.35
Base error [mm]	4.75	6.78	6.19	4.87	4.43	5.10	4.85

Table 5.4: Fine tuning of UKF and search parameters for Arrangement #2.

	Configuration						
Parameters	#1	#2_A	#2_B	#3	#4	#5	#6
samplegap	0.5	1.0	1.0	1.5	0.5	0.5	0.5
$\mathcal{Q}$	6e-8	6e-8	6e-8	6e-8	4e-8	1e-7	6e-8
$R_{interception}$	4e-8	4e-8	4e-8	4e-8	4e-8	4e-7	4e-8
$R_{failure}$	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
$\mathcal{P}$	6e-8	6e-8	6e-8	6e-8	4e-8	6e-8	1e-7
Interception	60.44%	71.56%	68.67%	59.56%	57.33%	56.00%	61.33%
Search time [s]	5.27	3.14	3.33	2.66	5.06	5.79	5.36
Base error [mm]	8.29	8.61	8.24	7.72	7.87	9.62	6.11

Table 5.5: Fine tuning of UKF and search parameters for Arrangement #3.

from the prediction computed after the search for fiducial 6. If the robot then fails to intercept fiducial 7, it is unable to correct this error and the uncertainty is propagated to the subsequent fiducials. This reduces the interception rate for the last three fiducials and consequently the overall performance of the system. Figure 5.8 illustrates this behavior.

	Arrangement #1						
Fiducials IDs considered	1 to 2	2 to 3	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9
Distance between fiducials [m]	0.35	0.34	0.21	0.54	0.99	0.22	0.42
Interception rate (UKF)	0.88	0.88	0.94	0.94	0.74	0.78	0.86
Interception rate (Spiral)	0.84	0.78	0.96	0.94	0.80	0.88	0.76

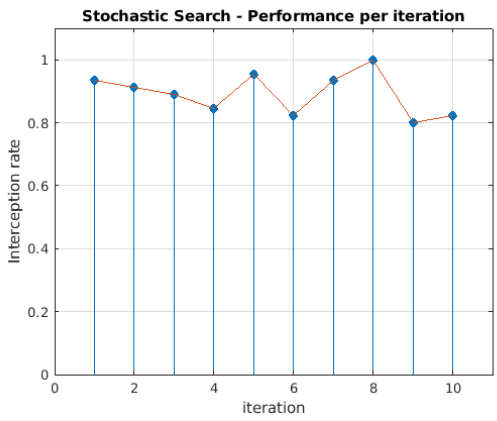
Table 5.6: Interception rate for both stochastic and deterministic methods as a function of distance between fiducials in Arrangement #1, Figure 4.9.

	Arrangement #2				
Fiducials IDs considered	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6
Distance between fiducials [m]	0.2	0.23	0.2	0.25	0.18
Interception rate (UKF)	0.88	0.92	1.00	0.92	0.92
Interception rate (Spiral)	1.00	0.96	0.86	0.94	0.96

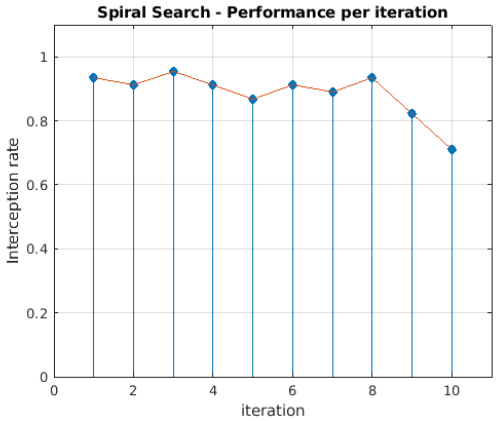
Table 5.7: Interception rate for both stochastic and deterministic methods as a function of distance between fiducials in Arrangement #2, Figure 4.10

	Arrangement #3						
Fiducials IDs considered	1 to 2	2 to 3	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9
Distance between fiducials [m]	0.23	0.27	0.38	0.23	1.31	0.37	0.25
Interception rate (UKF)	0.62	0.42	0.88	0.86	0.32	0.56	0.52
Interception rate (Spiral)	0.94	0.76	0.86	0.90	0.42	0.96	0.96

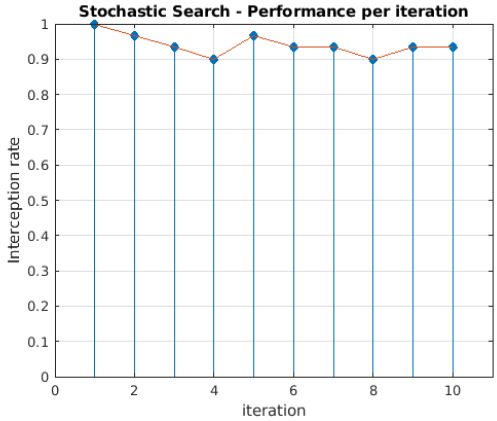
Table 5.8: Interception rate for both stochastic and deterministic methods as a function of distance between fiducials in Arrangement #3, Figure 4.11



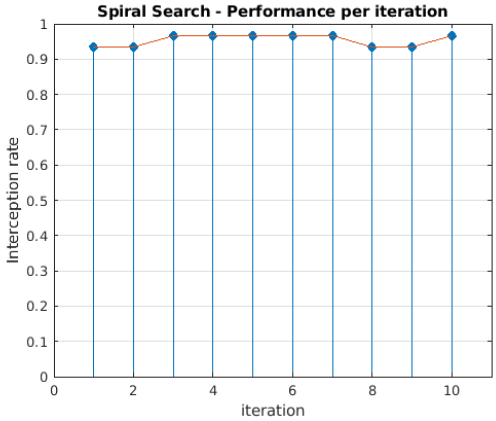
(a) Arrangement #1 - UKF



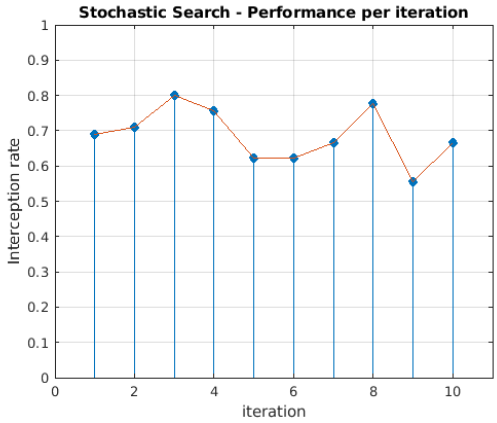
(b) Arrangement #1 - Spiral



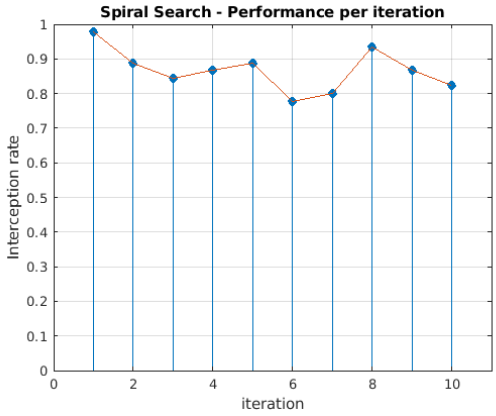
(c) Arrangement #2 - UKF



(d) Arrangement #2 - Spiral



(e) Arrangement #3 - UKF



(f) Arrangement #3 - Spiral

Figure 5.5: Performance as a function of iterations. Different rows correspond to different fiducial arrangements. *Left:* results with the UKF. *Right:* results with the spiral search.

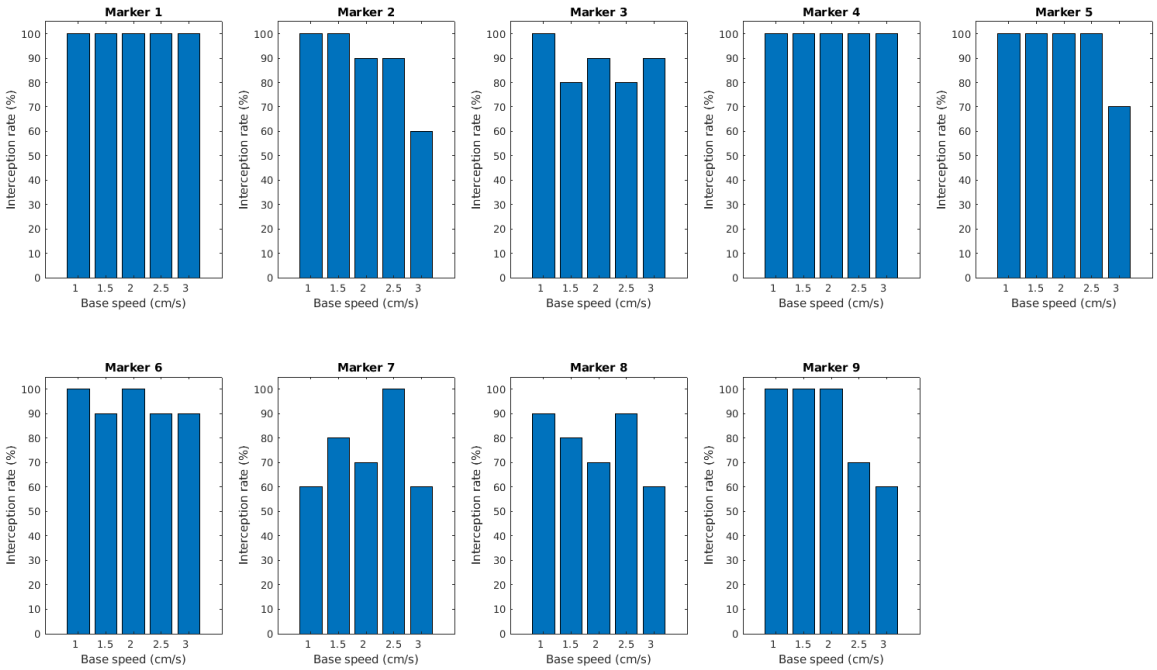
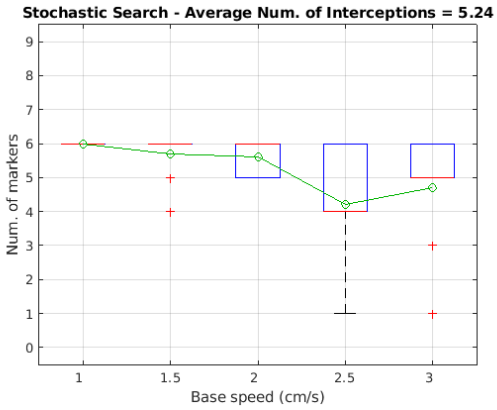
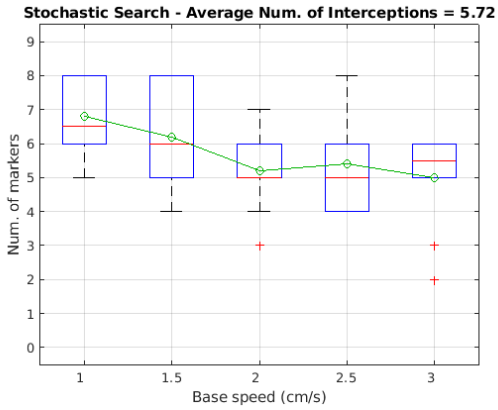


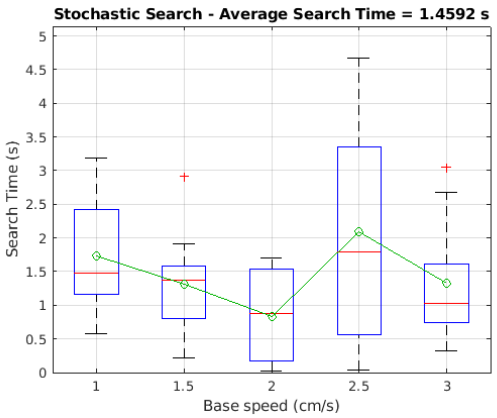
Figure 5.6: Interception per fiducial over 10 runs per base velocity, with Arrangement #1 (Figure 4.9) and UKF model configuration #6_B of Table 5.3.



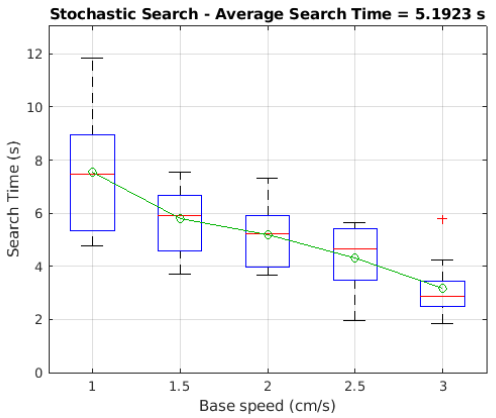
(a) Interception rate for Arrangement #2.



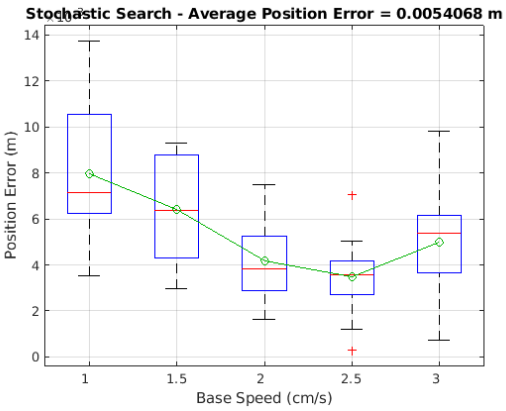
(b) Interception rate for Arrangement #3.



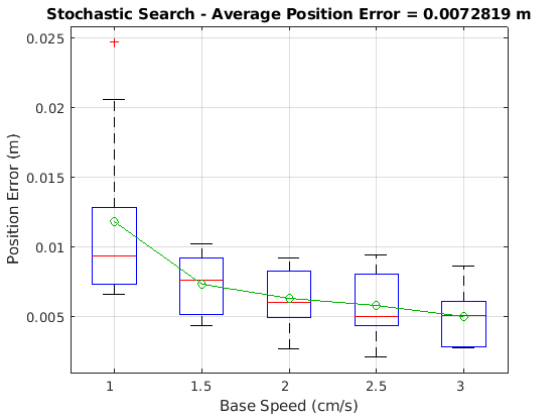
(c) Search times of Arrangement #2.



(d) Search times of Arrangement #3.

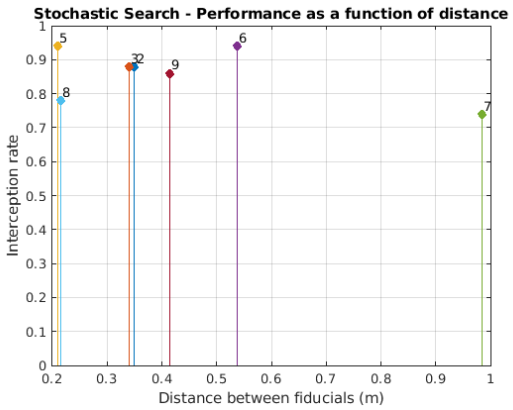


(e) Position error of Arrangement #2.

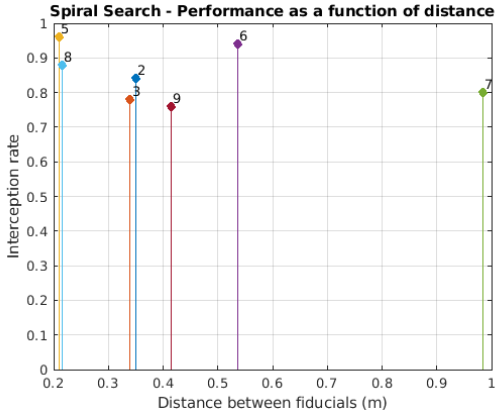


(f) Position error of Arrangement #3.

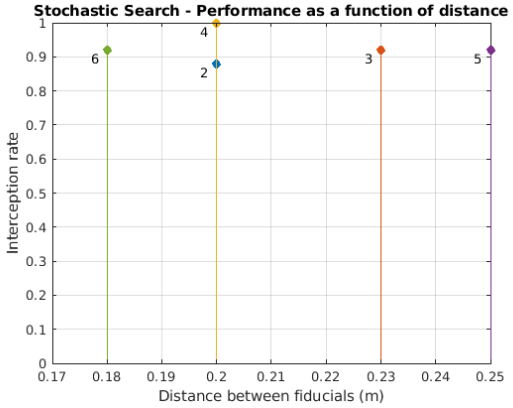
Figure 5.7: Performance of the fine-tuned model for Arrangements #2 and #3.



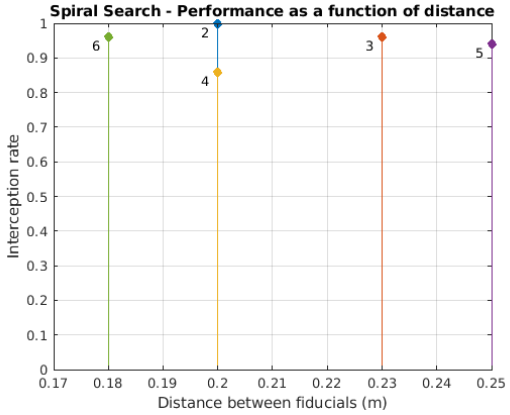
(a) Arrangement #1 - UKF



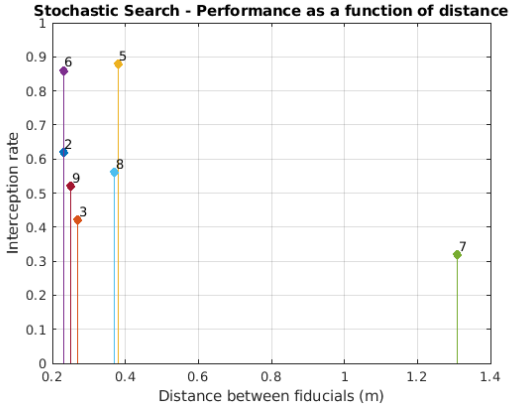
(b) Arrangement #1 - Spiral



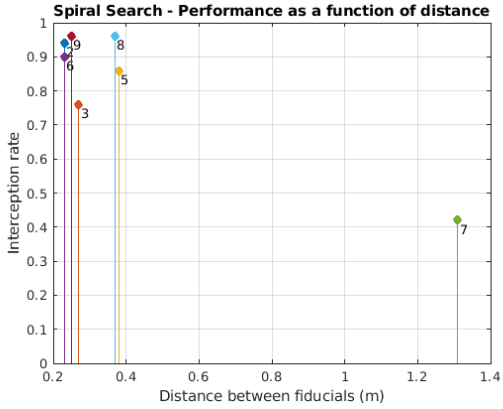
(c) Arrangement #2 - UKF



(d) Arrangement #2 - Spiral



(e) Arrangement #3 - UKF



(f) Arrangement #3 - Spiral

Figure 5.8: Performance as a function of distances between fiducials for each arrangement after fine tune. Different rows correspond to different fiducial arrangements. *Left:* results with the UKF. *Right:* results with spiral search.

5.7 Discussion of the results

Figure 5.1 shows that by changing the model from linear to non-linear, more fiducials are intercepted. One likely justification is that the UKF model better represents the actual characteristics of the system, which leads to a performance improvement of the filter. By incorporating the non-linearity of the system into its prediction step, the state vector better reflects the actual position of the base, and thus lower search times are needed to locate the fiducials.

Another interesting issue to point out is the fact that the linear Kalman filter shows an upward trend in interception rate as the velocity increases. A likely justification for this behaviour is that by moving the base faster there is less room for the motion drift shown in Figure 3.4. Hence, the linear Kalman filter performs well in those scenarios. In contrast, the UKF incorporates the drift into its estimates, such that for base velocity $v_b = 1$ cm/s the robot intercepts almost all the 9 fiducials in most experimental runs of Arrangement #1. However, this explanation needs further investigation, as discussed in Chapter 6. As Table 5.1 indicates, with the same parameters, in comparison with the linear model, the UKF increases the interception rate by 17.56% while reducing the average search time by 33.33%.

In Section 5.4, we evaluated how the proposed dynamic model using UKF performs under different fiducial arrangements. Figures 5.2, 5.3 and 5.4 show that regardless of the arrangement, there is a downward trend in interception rate as the velocity increases (for the UKF). This is because at higher base velocities, the compensated samples are forced to spread over longer distances that might be out of the robot's workspace. Furthermore, due to higher velocities of the base, for longer search intervals, frequently it is not possible to reposition the arm in time to intercept the next fiducial. This scenario occurs quite often, especially when moving from fiducial 7 to fiducial 8 in Arrangement #1 (Figure 4.9), where the fiducials are close

to each other. Figure 5.6 illustrates this pattern: when marker 7 is not intercepted at higher velocities, then marker 8 is less likely to be found as well. In contrast, for $v_b = 1$ cm/s, we see a higher interception of marker 8 even though the interception of marker 7 is lower at that velocity. This is because lower velocities allow the arm to be repositioned before approaching the next fiducial.

For Arrangement #2, our proposed approach outperformed the KF-based approach proposed in [26], which was our main baseline. Tested under the same fiducial configuration used in [26], our model achieved an average performance of 91.33% over 10 runs. In contrast, as reported in [26], the model presented by Yaw et al. achieved 71.17% in their simulated experiments for the same number of iterations.

It is worth mentioning that the average position error in all the three arrangement scenarios is lower for the UKF than for the spiral search approach. This is because the predictions from the Kalman filter help the mobile manipulator get closer to the fiducial and is thus more accurate than the deterministic approach. The mean position errors for each method and each arrangement can be found in Table 5.2.

Another interesting observation regarding the results in Section 5.4 is that the spiral search method intercepted more fiducials with a lower search time, before fine tuning the UKF model. Two main issues justify this pattern. First, we noticed that the performance of the Kalman filter decreases when the distance between two fiducials is significantly high (for instance, between fiducial 6 and 7 in Arrangements #1 and #3). With longer distances, we observe higher motion drift in the direction of the y axis. Hence, by the time the mobile manipulator reaches the predicted state, its actual state may not reflect the state that was predicted at the previous time step. This is evident in Figure 5.8, where interception rate is very high for short distances between fiducials but it decreases significantly when the mobile manipulator has to travel further (e.g., when it approaches fiducial 7 in Arrangements #1 and #3). It is

noticeable that the spiral search also performs poorly for this fiducial in particular, due to the drift that increases along the distance travelled. However, since we designed our Kalman filter to update with an observation even when a fiducial interception does not occur, for fiducial 7 it receives an inaccurate measurement from the base's odometry after the search process ends. This inaccurate measurement influences the following predictions, leading to poor performance for the remaining fiducials. Figure 5.6 shows that the last three fiducials have the poorest interceptions rates.

The second reason why the spiral search method outperformed the UKF in our experiments before fine tuning is due to the ROS kinematics solvers on which we rely to position the arm. We noticed that, besides the drift of the base, the arm is not positioned to our predicted position, introducing additional drift in the end effector when compared to the desired position. We mitigated the arm positioning issue by correcting the pose of the manipulator to place its end effector in a specific location such that the distance from the base to the end effector corresponds to the predicted arm displacement. However, when the drift of the base is significant, this strategy may place the arm further from the fiducial location, instead of correcting it. In these scenarios, the stochastic sampling points are not centered at the expected marker position. Instead, they are centered at the location of the predicted displacement of the arm plus the drift of the base. Hence, the fiducial would be less likely to be intercepted. In contrast, the deterministic approach does not sample according to a probabilistic distribution, such that, even in the presence of drift, the sampling points still cover the fiducial position, which will most likely be intercepted after a longer search period.

In Table 5.3, we presented the results of experiments in which we vary the process and observation noises in an attempt to improve the performance of the UKF. We exploit different combinations of these parameters and inspect the influence of each variable in the overall performance. The results indicate that increasing the

initial covariance $\mathcal{P}$ results in more accurate estimates, leading to higher interception rates.

Figure 5.2 then compares our fine-tuned model against the spiral search for Arrangement #1. As presented in Table 5.3, with a more accurate estimate of the initial covariance of the process noise, our proposed model intercepted 89.11% of the fiducials, against the 88.44% interception rate obtained by the spiral search strategy. In terms of search time, the stochastic approach took an average search time of 2.68s, whereas the mean search time for the spiral search strategy was 1.51s. In terms of position error, it is clear that the UKF performs better, with a mean position error of only 9.59mm, against a mean position error of 11.32mm for the spiral method.

However, the fine-tuned model (Configuration #6_B) obtained with Arrangement #1 does not show significant improvement for the other two arrangements, as illustrated in Figure 5.7. For Arrangement #2, the interception rate is 87.33% for this fine-tuned model, a drop of almost 4% when compared to the model before the attempt to fine-tune the model for the same arrangement. For Arrangement #3, the fine-tuned model of Arrangement #1 reached an interception rate of 63.56%, an increase of 3.12% compared with the version prior to fine-tuning. Despite this improvement, the performance of the UKF for Arrangement #3 is still lower than that of the spiral method.

In an attempt to obtain a UKF model that performs better than spiral search for Arrangements #2 and #3, we explored the same parameter configurations listed in Table 5.3 for these two arrangements. The results are shown in Tables 5.4 and 5.5. For Arrangement #2, we observe that a higher sample gap improves the interception rate from 91.33% to 95.00%, which is comparable to the performance of spiral search for this same arrangement, which has an interception rate of 95.33%, as shown in Table 5.2. Another interesting issue to point out is the fact that increasing the process covariance $\mathcal{Q}$ results in higher interception rates. For Arrangement #3, the

interception rate improved from 60.44% to 68.67%. More extensive fine-tuning of the model parameters would likely lead to additional performance improvements.

Finally, as seen in Figure 5.5, the performance per iteration varies substantially, especially in cases when it is necessary to re-initialize ROS/Gazebo. This variability in the experimental runs demonstrates the complexity of the overall system, and justifies the need for additional parameter refinement.

CHAPTER 6
CONCLUSION

In this thesis, we have dealt with the problem of dynamic modeling of mobile manipulators. We present potential applications of manipulators within different environments, such as healthcare, manufacturing, navigation, and especially industrial automation. Regardless of their numerous applications, mobile manipulators still lack accuracy in end effector positioning in the absence of external localization and feedback control. The need to calibrate mobile manipulators stems from the fact that they tend to suffer from reduced precision in comparison with stationary manipulators, where there is no motion of the base. As a consequence, the need for a dynamic performance measurement arises. Two search methods have already been proposed in earlier works: a stochastic approach and a deterministic one. The methods proposed in this thesis build upon these approaches.

Our first observation was that progress could be made in the theoretical model of the stochastic search procedure. For instance, a dynamic adjustment of the process covariance can improve the estimation step of the Kalman filter. This can be accomplished by increasing the covariance of the observation noise when a marker is not found after the search process. As a consequence, this procedure results in the generation of sample points that are closer to the fiducial position and thus increase the probability of locating it. Consequently, the search time could be reduced if we take into account the interval between observations in the prediction process, such that the sampling will be more precise and thus, less amount of time will be needed to find the fiducial.

Based on these observations, we developed a dynamical model that better reflects the characteristics of mobile manipulators. More specifically, we developed a Bayesian algorithm using an Unscented Kalman filter to better estimate the pose

of the mobile manipulator in a scenario where the arm and the base move simultaneously. As demonstrated in Chapter 5, although our stochastic search mechanism, performs significantly better than our baseline, it needs to be fine tuned to outperform the deterministic method. Our additional fine tuning experiments revealed that, under specific process and observation covariances, it is possible to obtain an improved stochastic model that performs comparably to and sometimes outperforms the spiral search method. The remaining set of search parameters, such as the sample quantization step also play an important role in the performance of the stochastic search algorithms. However, this optimal set of parameters depend on the arrangement of the fiducials, since the fine tuned system model for Arrangement #1 does not outperform the spiral search method when tested under Arrangements #2 and #3, which require a different set of parameter values. This leads us to conclude that the complexity of the system and the drift of the base affect the performance of the stochastic search, and fine tuning is thus needed. We determined that designing an effective calibration procedure using a Kalman filter requires knowledge of the peculiarities of the environment as well as the characteristics of the noise corrupting its dynamic state and the measurements.

6.1 Future work

Our proposed model using a UKF only performed better than the spiral search strategy after we fine-tuned the model parameters. However, manually fine-tuning those parameters requires significant human effort. One line of future work would be a more comprehensive study of how the covariances impact the performance and whether it is possible to estimate these uncertainties using automated techniques.

Another immediate future work direction that we have already started pursuing is the implementation and evaluation of our model in a real mobile manipulator.

This will validate the conclusions obtained from the simulated experiments and ensure that our proposed approach is applicable to real-world scenarios.

Moreover, our mobile manipulator platform and its corresponding model can be further extended by incorporating a camera mounted to the end effector of the manipulator. This would make it possible to explore vision-based pose estimation algorithms to improve the localization of the platform and reduce fiducial interception errors.

BIBLIOGRAPHY

[1] Samuel Amoako-Frimpong. *Search Methods for Mobile Manipulator Performance Measurement*. PhD thesis, Marquette University, 2018.

[2] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[3] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME, Trans., Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[4] Jo Bo Rosen. The gradient projection method for nonlinear programming. Part I. linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, 1960.

[5] Xavier Lamy, Frédéric Collédani, Franck Geffard, Yvan Measson, and Guillaume Morel. Human force amplification with industrial robot: Study of dynamic limitations. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2487–2494. IEEE, 2010.

[6] Roger Bostelman, Ya-Shian Li-Baboud, Soocheol Yoon, Mili Shah, and Omar Aboul-Enein. Towards measurement of advanced mobile manipulator performance for assembly applications. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2020 [Online]. https://doi.org/10.6028/NIST.TN.2108 (Accessed November 10, 2021).

[7] Bradley Hamner, Seth Koterba, Jane Shi, Reid Simmons, and Sanjiv Singh. Mobile robotic dynamic tracking for assembly tasks. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2489–2495. IEEE, 2009.

[8] Ariel S Kapusta, Phillip M Grice, Henry M Clever, Yash Chitalia, Daehyung Park, and Charles C Kemp. A system for bedside assistance that integrates a robotic bed and a mobile manipulator. *Plos one*, 14(10):e0221854, 2019.

[9] Henrik Christensen, Nancy Amato, Holly Yanco, Maja Mataric, Howie Choset, Ann Drobnis, Ken Goldberg, Jessy Grizzle, Gregory Hager, John Hollerbach, et al. A roadmap for us robotics–from internet to robotics 2020 edition. *Foundations and Trends® in Robotics*, 8(4):307–424, 2021.

[10] ZVIS Roth, B Mooring, and Bahram Ravani. An overview of robot calibration. *IEEE Journal on Robotics and Automation*, 3(5):377–385, 1987.

[11] Roger Bostelman, Tsai Hong, and Jeremy Marvel. Performance measurement of mobile manipulators. In *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2015*, volume 9498, page 94980E. International Society for Optics and Photonics, 2015.

[12] Yuandong Yang and Oliver Brock. Elastic roadmaps-motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010.

[13] Dov Katz, Emily Horrell, Yuandong Yang, Brendan Burns, Thomas Buckley, Anna Grishkan, Volodymyr Zhylkovskyy, Oliver Brock, and Erik Learned-Miller. The umass mobile manipulator uman: An experimental platform for autonomous mobile manipulation. In *Workshop on manipulation in human environments at robotics: science and systems*. Citeseer, 2006.

[14] Qiang Huang, Kazuo Tanie, and Shigeki Sugano. Stability compensation of a mobile manipulator by manipulatorpaper motion: feasibility and planning. *Advanced Robotics*, 13(1):25–40, 1998.

[15] Michael Schneier, Michael Schneier, and Roger Bostelman. Literature review of mobile robots for manufacturing, nist interagency/internal report (nistir). Technical report, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2015 [Online]. https://doi.org/10.6028/NIST.IR.8022 (Accessed November 10, 2021).

[16] ZE Chebab, JC Fauroux, N Bouton, Y Mezouar, and L Sabourin. Autonomous collaborative mobile manipulators: State of the art. In *Symposium on Theory of Machines and Mechanisms/UMTS2015/TrISToMM*, 2015.

[17] Margaret E Jefferies and Wai-Kiang Yeap. *Robotics and cognitive approaches to spatial mapping*. Springer, 2008.

[18] Miomir Vukobratovic and Branislav Arso Borovac. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004.

[19] Daniel E Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. on man-machine systems*, 10(2):47–53, 1969.

[20] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[21] Antonio Gonzales Marin, Mohammad S Shourijeh, Pavel E Galibarov, Michael Damsgaard, Lars Fritzsch, and Freek Stulp. Optimizing contextual ergonomics models in human-robot interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[22] Guilherme N DeSouza and Avinash C Kak. A subsumptive, hierarchical, and distributed vision-based architecture for smart robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):1988–2002, 2004.

[23] Youngrock Yoon, Akio Kosaka, and Avinash C Kak. A new kalman-filter-based framework for fast and accurate visual tracking of rigid objects. *IEEE Transactions on Robotics*, 24(5):1238–1251, 2008.

[24] Mustafa Mashali, Rdwan Alqasemi, and Rajiv Dubey. Mobile manipulator dual-trajectory tracking using control variables introduced to end-effector task vector. In *2016 World Automation Congress (WAC)*, pages 1–6. IEEE, 2016.

[25] Emilia Brzozowska, Oscar Lima, and Rodrigo Ventura. A generic optimization based cartesian controller for robotic mobile manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2054–2060. IEEE, 2019.

[26] Amoako-Frimpong Samuel Yaw, Matthew Messina, Henry Medeiros, Jeremy Marvel, and Roger Bostelman. Stochastic search methods for mobile manipulators. *Procedia Manufacturing*, 17:976–984, 2018.

[27] Youngjoo Kim and Hyochoong Bang. Introduction to kalman filter and its applications. *Introduction and Implementations of the Kalman Filter, F. Govaers, Ed. IntechOpen*, 2019.

[28] Open Source Robotics Foundation, howpublished = `https://www.ros.org/about-ros/`, note = Accessed: 2021-09-24.

[29] Robot Kinematics in a Nutshell, author = Lentin Joseph, howpublished = `https://robocademy.com/2020/04/21/robot-kinematics-in-a-nutshell/`, note = Accessed: 2021-10-28.

[30] Kris Hauser. Robotic Systems. `http://motion.cs.illinois.edu/RoboticSystems/Kinematics.html`, 2020. [Online; Accessed 25-August-2021].

[31] Handbook of Robotics, author=Waldron, Ken and Schmiedeler, Jim, howpublished = `https://www.cc.gatech.edu/~hic/8803-mobile-08/slides/1_draft.pdf`, note = Accessed: 2021-10-17.

[32] Optimal Estimator - Kalman Filter, howpublished = https://homes.esat.kuleuven.be/~maapc/static/files/cacsd/slides/chapter6.pdf, note = Accessed: 2021-10-17.

[33] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[34] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[35] Gazebo tutorial - Beginner: Overview, howpublished = http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1, note = Accessed: 2021-10-15.

[36] Derek Rowell. State-Space Representation of LTI Systems. https://web.mit.edu/2.14/www/Handouts/StateSpace.pdf, 2002. [Online; Accessed 27-August-2021].

[37] Olivier Guéant and Jiang Pu. Mid-price estimation for european corporate bonds: a particle filtering approach. *Market microstructure and liquidity*, 4(01n02):1950005, 2018.

[38] Burak Turan and Ali Turker Kutay. Particle filter studies on terrain referenced navigation. In *Proceedings of IEEE/ION PLANS 2016*, pages 949–954, 2016.

[39] Kalman and Bayesian Filters in Python, author = Labbe, Roger, howpublished = https://filterpy.readthedocs.io/en/latest/kalman/kalmanfilter.html#id1, note = Accessed: 2021-10-15.

[40] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. Oregon Health & Science University, 2004.

[41] Moveit - ROS Package Summary, howpublished = http://wiki.ros.org/moveit, note = Accessed: 2021-10-18.

[42] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[43] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2002.

[44] Roger Bostelman. *Performance Measurement Of Mobile Manipulators*. PhD thesis, University Of Burgundy, France, March 2018.

[45] Roger Bostelman, Tsai Hong, and Elena Messina. Intelligence level performance standards research for autonomous vehicles. In *Proceedings FinE-R Workshop - IROS*, pages 48–54, 2015.

[46] Somrita Chattopadhyay, Shayan A. Akbar, Noha M. Elfiky, Henry Medeiros, and Avinash Kak. Measuring and modeling apple trees using time-of-flight data for automation of dormant pruning applications. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[47] Brad Hamner, Seth Koterba, Jane Shi, Reid Simmons, and Sanjiv Singh. An autonomous mobile manipulator for assembly tasks. *Autonomous Robots*, 28(1):131–149, 2010.

[48] Baoshuang Ge, Hai Zhang, Liuyang Jiang, Zheng Li, and Maaz Mohammed Butt. Adaptive unscented kalman filter for target tracking with unknown time-varying noise covariance. *Sensors*, 19(6):1371, 2019.

[49] World-beam qs18llp series datasheet - banner engineering, howpublished = `https://info.bannerengineering.com/cs/groups/public/documents/literature/118900.pdf`, note = Accessed: 2021-10-06.

[50] Roger Bostelman, Tsai Hong, Jeremy Marvel, et al. Survey of research for performance measurement of mobile manipulators. *Journal of Research of the National Institute of Standards and Technology*, 121(3):342–366, 2016.

[51] Roger Bostelman, Joe Falco, and Tsai Hong. Performance measurements of motion capture systems used for agv and robot arm evaluation. In Roger Bostelman and Elena Messina, editors, *Autonomous Industrial Vehicles: From the Laboratory to the Factory Floor*, chapter 7. STP1594, 2016.

[52] Robot Kinematics, howpublished = `http://motion.cs.illinois.edu/roboticsystems/kinematics.html`, note = Accessed: 2021-10-11.