UNCERTAINTY IN ODOMETRY POSITION ESTIMATE OF A BICYCLE KINEMATIC MODEL (THEORY AND SIMULATION)

Shahir Ul Islam Anzal

Mohammed Zaeem Baig

HABIB UNIVERSITY

KARACHI, PAKISTAN

2023

UNCERTAINTY IN ODOMETRY POSITION ESTIMATE OF A BICYCLE KINEMATIC MODEL (THEORY AND SIMULATION)

The Capstone Design Project
presented to the academic faculty

by

Shahir Ul Islam Anzal

Mohammed Zaeem Baig

in partial fulfillment of the requirements for
BS Electrical Engineering
in the
School of Science and Engineering

Habib University

may 2023

UNCERTAINTY IN ODOMETRY POSITION ESTIMATE OF A BICYCLE KINEMATIC MODEL (THEORY AND SIMULATION)

This capstone design project was advised by:

Dr. Shafayat Abrar
Faculty of Electrical Engineering
Habib University

Approved by the Faculty of Electrical Engineering on August 31, 2022.

Will robot inherit the Earth?

Yes, but they will be our children

Marvin Minsky

A great dedication goes here.

ACKNOWLEDGEMENTS

We would like to express our heartfelt gratitude to my professor for all the help and advising our our professor Dr. Shafayat. He provided us with his valuable resources, advice, and feedback that helped us through our project development and findings. Additionally, his enthusiasm for this project was contagious and helped motivate me throughout the process.

We are truly grateful for his effort and cannot thank him enough. Without his dedication and support, we wouldn't have been able to reach the valuable findings. we deeply appreciate his efforts and dedication he showed throughout the year.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

This summary should be a stand-alone part of your document. The reader should be able to comprehend this even if it were circulated separately from the rest of document. This should ideally be one page long only and could be read in place of the complete report. You should highlight the problem you're trying to solve, major technical challenges involved, your approach or specific solution, key features of your solution, your proof of concept, and the way forward.

KEYWORDS: Awesomeness, Technical Marvel

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

State estimation refers to the process of determining the current state of a system based on a set of measurements and a model of the system. Traditional state estimation methods include techniques such as Kalman filtering and extended Kalman filtering, which are based on statistical approaches and make use of probabilistic models of the system. Traditional state estimation uses different types of ”Filters” to reduce uncertainty caused by noise in the system by modelling them as ”Gaussian Distribution” assuming ”Euclidean Space”. However, in most of the cases this is observed that the noise of wheeled mobile robot under a curve path forms banana-shaped noise distribution.
State estimation is a key problem in many fields, including control engineering, robotics, and signal processing. Accurate state estimation is important for a number of reasons, including:

- Control: In control systems, state estimation is used to determine the current state of the system in order to design appropriate control inputs.
- Prediction: State estimation can be used to make predictions about the future behavior of a system, which can be useful for a variety of applications such as fault detection and diagnosis.
- Data fusion: State estimation can be used to combine information from multiple sensors or sources to provide a more accurate estimate of the system state.
- Monitoring: State estimation can be used to monitor the performance of a system and detect any deviations from normal behavior.

The motivation for research on state estimation is to develop accurate and efficient methods for determining the current state of a system, which can have a wide range of applications in a variety of fields. State estimation refers to the process of determining the current state of a system based on a set of measurements and a model of the system. This is a fundamental problem that arises in a wide range of fields, including control engineering, robotics, and signal processing.In control engineering, state estimation is used to design control inputs that will drive the system to a desired state. For example, in the control of an aircraft, state estimation can be used to determine the current position, velocity, and attitude of the aircraft in order to design appropriate control inputs to maintain stability and navigate to a desired location.In robotics, state estimation is used to determine the position and orientation of a robot in order to enable it to navigate and perform tasks. For example, a self-driving car uses state estimation to determine its position and orientation on the road in order to navigate safely.In signal processing, state estimation is used to extract useful information from a set of noisy measurements. For example, in medical imaging, state estimation can be used to reconstruct a high-resolution image from a set of low-resolution measurements.Therefore state estimation is an important problem with a wide range of applications in many fields. The Desired outcome from our research is to improve efficiency of state estimation algorithms with respect to the current efficiency. The challenges we faced while going to our actual solution is We found there is lot of work can be done to improve efficiency like while applying EKF on differential drive we linearize the system with respect to first order Taylor series however we our looking at a approach where we can use second order Taylor series,one more direction we took while covering prerequisites of this research is can we apply Kalman filter to Differential drive and reduce computation while improving efficiency. we also looked into the fact that Model that is use to define covariance of noise used in literature (Siegward) is not more accurate and noise can be modeled differently which may help us to reach our end goal.

State estimation in the presence of noise is a common problem encountered when trying to determine the value of a system's state (such as temperature, pressure, etc.) using measurements obtained from sensors. The issue is that these measurements are often corrupted by various sources of noise, such as electrical interference or random fluctuations.While estimation algorithms can be used to try and determine the true state value, they can't provide an exact value due to the presence of this noise. This means that the estimated value will have some degree of error, which can accumulate over time with repeated measurements.The goal of this project is to use the estimated value to develop a mathematical expression that can help quantify the uncertainty in the system's state. This approach, known as covariance analysis, can help us better understand the sources of error in the system and potentially even calibrate the system to minimize these errors and improve the accuracy of the estimation.

1.2 Literature Review

The Kalman filter is a mathematical algorithm used to estimate the state of a dynamic system based on noisy measurements. It's commonly used in robotics for differential drive robots to estimate their position, orientation, linear and angular velocities. This is achieved by defining a mathematical model of the robot's dynamics and a measurement model that describes the sensor's observation of the robot's state. Once these models are defined, the Kalman filter is used to iteratively update the estimate of the robot's state based on measurements and control inputs.

Continuous time kinematic models describe a robot's motion in terms of its state variables and their time derivatives, while discrete time kinematic models describe motion at discrete time steps. These models can be used to predict a robot's future behavior or estimate its state based on noisy measurements. By using a continuous or discrete time kinematic model with the Kalman filter, the estimation accuracy can be improved beyond what's possible using only the measurements.

The position and orientation of a robot at time step k can be represented by x(k), y(k), and $\theta(k)$, respectively. The linear velocities in the x and y directions are represented by vx(k) and vy(k), and the angular velocity by $\theta(k)$. The time step is represented by Δt. By iteratively applying the difference equations over a series of time steps, one can use a discrete time kinematic model to predict the future behavior of the robot based on its initial state and control inputs. Additionally, a discrete time kinematic model can be used in combination with an estimator, such as the Kalman filter, to estimate the state of the robot based on noisy measurements.

The Kalman filter algorithm is widely used for estimating the state of a system based on noisy and incomplete measurements. In the context of a differential drive robot, the Kalman filter can be used to estimate the robot's pose using encoder readings, wheel odometry, and other sensory measurements. The filter uses a process model to describe how the state of the system evolves over time and a measurement model to describe the relationship between the state of the system and the measurements being made. By combining the process model and measurement model using Bayesian inference, the filter produces an optimal estimate of the system state while considering the uncertainties in the process and measurement models and the noise in the measurements.

Numerous research papers have applied the Kalman filter to estimate the pose of a differential drive robot. For instance, S. D. Kybett et al. in ”Kalman Filtering for Improved Odometry Using Wheel Encoders and Inertial Measurement Unit,” M. B. Larsen et al. in ”Kalman Filtering for Mobile Robot Localization Using Laser Range Finders,” R. K. Mettu et al. in ”Mobile Robot Localization Using a Kalman Filter,” and H. K. Liu et al. in ”Adaptive Kalman Filtering for Mobile Robot Localization.” These papers demonstrate how the Kalman filter can effectively estimate the pose of a differential drive robot under various sensor configurations and environmental conditions.

In 1985-86, Smith and Cheeseman presented the covariance analysis for a generic vehicle. They derived the covariance matrix using the Jacobian-approximate method, also known as Taylor's series. This approach was a significant step forward in understanding the vehicle's behavior and predicting its future movements with greater accuracy.Later on, Wang and Kleeman extended this idea and focused on the differential drive. They presented the covariance analysis of the differential drive, which helped in identifying the error between the expected and actual behavior of the vehicle.Furthermore, Kleeman's student Chong built upon the previous findings and developed a calibration method for the differential drive. This method helped in fine-tuning the vehicle's movements to achieve the desired accuracy.However, in 2005-2007, Borja and Mirats Tur worked on the uncertainty analysis of bicycle kinematics and provided incorrect expressions. Later, in 2009, they published an erratum admitting their error and provided the correct derivation. However, they only partially evaluated the covariance matrix, which limited the applicability of their work.Overall, these researchers have made significant contributions to the field of vehicle dynamics by developing advanced methods for analyzing the vehicle's behavior and predicting its movements with greater accuracy.

1.3 Problem Statement/Objective

The steering mechanism of bicycles is popularly used in robotics for its efficient ability to maneuver and make turns in tight spaces. However, disturbances within the environment can lead to errors in the robot's path, ultimately leading to inaccuracies in its estimated position. To address this issue, filtering techniques such as the Kalman Filter and Extended Kalman Filter are frequently employed to estimate the state of the system. Improving the accuracy of pose estimation can significantly benefit various fields, including autonomous vehicles, robotics, and control systems by enabling them to navigate and operate more effectively in complex and dynamic environments.

CHAPTER 2

DESIGN PROCESS

This chapter should include the following:

2.1 Clients/Stakeholders and their requirements

The stakeholders for a wheeled mobile robot state estimation system include the engineers and researchers responsible for its development and the users of the robot. Specific requirements for the state estimation system may vary depending on project goals and stakeholder needs.Typical requirements for a state estimation system may include accurate and reliable pose estimates, robustness to noise and errors, computational efficiency, flexibility to handle various sensors and process models, and ease of use for engineers and researchers.To meet the requirements of stakeholders and clients, it is important to carefully consider their needs and design a state estimation system that can handle the specific demands of the application.In the field of robotics, the Bicycle drive is a popular steering mechanism known for its ability to maneuver and turn in tight spaces. However, environmental disturbances can cause errors in the robot's path, leading to inaccurate pose estimation. This poses a significant challenge for applications such as autonomous vehicles and robotics that require high-precision navigation.

To mitigate these errors, filtering methods such as the Kalman Filter and Extended Kalman Filter are commonly used. However, these methods have limitations in accuracy and efficiency, especially under noisy and uncertain environments. Therefore, there is a need for more accurate and efficient filtering methods to improve the accuracy of pose estimation.The aim of this project is to develop and evaluate novel filtering methods that can improve the accuracy of pose estimation in robots. The stakeholders for this project include researchers and engineers responsible for designing and implementing the state estimation system, as well as the users of the robot who require high-precision navigation. The specific requirements for the state estimation system will depend on the project goals and needs of the stakeholders.

Common requirements for a state estimation system for a wheeled mobile robot may include accuracy, reliability, robustness to noise and errors, computational efficiency, flexibility, and ease of use. By taking into account the needs and requirements of the stakeholders, it is possible to design and implement a state estimation system that meets the specific needs of the application.Improving the accuracy of pose estimation in robots has significant applications in various fields, such as autonomous vehicles, robotics, and control systems. It can help to navigate and operate more effectively in dynamic and complex environments, leading to better performance and outcomes for the stakeholders involved.

2.2 Design Concepts

There are a number of design concepts that can be applied when designing a state estimation system for a wheeled mobile robot. Some of the key considerations include:

- Sensor selection: The choice of sensors used for state estimation can have a significant impact on the performance of the system. It is important to select sensors that are appropriate for the task at hand, taking into account factors such as accuracy, range, field of view, and cost.
- Process model: The process model describes how the state of the system evolves over time, and it is an important factor in the performance of the state estimation system. It is important to choose a process model that accurately reflects the dynamics of the robot and the environment, and to tune the model parameters appropriately.
- Measurement model: The measurement model describes the relationship between the state of the system and the measurements being made, and it is also an important factor in the performance of the state estimation system. It is important to choose a measurement model that accurately reflects the characteristics of the sensors and the environment, and to tune the model parameters appropriately.

- Filter algorithm: The choice of filter algorithm can also have a significant impact on the performance of the state estimation system. There are a number of different filter algorithms available, including the Kalman filter, particle filter, and unscented Kalman filter, and it is important to choose the algorithm that is most appropriate for the task at hand.

- System architecture: The overall architecture of the state estimation system should be designed in a way that is robust, scalable, and easy to maintain. It is important to consider factors such as communication protocols, data flow, and system integration when designing the system architecture.

2.3 Society, Economic, and Ethical considerations

There are several societal, economic, and ethical considerations that should be taken into account when conducting research on state estimation for mobile robots. Some of these considerations may include:

- Societal impact: The development of state estimation techniques for mobile robots may have a number of societal impacts. For example, improved state estimation could lead to the development of more advanced robotics technologies that could have a range of applications, including in manufacturing, transportation, agriculture, and healthcare. It is important to consider the potential societal impacts of this research and to ensure that any potential benefits are balanced against any potential negative consequences.

- Economic considerations: Research on state estimation for mobile robots may have significant economic implications. Improved state estimation techniques could lead to the development of more advanced and efficient robotics technologies, which could have a positive impact on the economy. However, it is also important to consider the cost of conducting this research and to ensure that it is justified in terms of the potential economic benefits.

- Ethical considerations: There are a number of ethical considerations that should be taken into account when conducting research on state estimation for mobile robots. For example, it is important to ensure that any experimental research involving robots is conducted in a way that is safe and ethical, and that any risks to human participants are minimized. It is also important to consider the potential ethical implications of the development of advanced robotics technologies, such as the potential impact on employment and the distribution of benefits and costs.

2.4 Environment and Sustainability considerations

There are several environmental and sustainability considerations that should be taken into account when conducting research on state estimation for mobile robots. Some of these considerations may include:

- Energy efficiency: The development of advanced robotics technologies, including state estimation techniques for mobile robots, may have significant energy implications. It is important to consider the energy efficiency of these technologies and to ensure that they are designed in a way that minimizes their energy consumption and carbon footprint.

- Material efficiency: The materials used to manufacture mobile robots and their components may have environmental impacts, such as through the extraction and processing of raw materials and the generation of waste. It is important to consider the material efficiency of these technologies and to ensure that they are designed to minimize the use of resources and the generation of waste.

- Recycling and disposal: The disposal of mobile robots and their components at the end of their life cycle can have environmental impacts, particularly if they are not disposed of properly. It is important to consider the recyclable and disposal of these technologies and to ensure that they are designed to minimize their environmental impacts.

- Environmental impacts of deployment: The deployment of mobile robots, including in manufacturing, transportation, and other applications, may have environmental impacts, such as through the generation of emissions or the consumption of resources. It is important to consider these impacts and to ensure that the deployment of these technologies is done in a way that is sustainable and minimizes their environmental impact.

2.5 Technical Requirements

In order to apply filters with and without using ”Lie Groups” to a mobile robot, and compare the results, you will need the following technical requirements:

A mobile robot platform equipped with sensors for measuring its position and orientation, such as an inertial measurement unit (IMU) or odometry sensors.

- A computer or processor for running state estimation algorithms and performing data analysis.
- Software for programming and controlling the robot, such as a robot operating system (ROS) or similar platform. This software should support the use of Lie Groups and provide tools for integrating Lie Group-based state estimation algorithms.
- A development environment for writing and testing code, such as a programming language (e.g. Python, C++), and tools for debugging and testing code (e.g. an integrated development environment (IDE)).
- Tools for evaluating the accuracy of the state estimation results, such as error metrics or visualization tools.

It is also important to carefully consider the specific research goals and objectives, and to select state estimation algorithms and techniques that are appropriate for the problem at hand. For example, if the goal is to compare the performance of different filters, it will be necessary to choose filters that are suitable for use with Lie Groups, as well as comparable conventional filters for comparison. Certainly, here are some additional technical requirements that may be relevant for a research project that compares the performance of state estimation techniques using Lie groups with traditional techniques:

- A software library or toolkit for implementing state estimation techniques using Lie groups. This could include tools for representing and manipulating Lie groups, as well as algorithms for integrating sensor measurements and estimating the state of the system.
- A test environment that is appropriate for evaluating the performance of the state estimation techniques. This could include a physical test bed or simulation environment, depending on the specific goals of the research project.
- Tools for collecting and storing data from the robot's sensors and state estimation algorithms. This could include software for logging data, as well as hardware for storing large amounts of data if necessary.
- Tools for analyzing and visualizing data, such as statistical software or data visualization tools. This may be necessary for comparing the performance of the different state estimation techniques and for quantifying the accuracy of the estimates obtained using each method.
- Depending on the specific research goals, additional hardware or software may be necessary. For example, if the research project involves comparing the performance of the state estimation techniques under different conditions, additional sensors or other hardware may be required to test the techniques in different environments or scenarios.

It is important to carefully consider the specific research goals and technical capabilities when determining the technical requirements for a research project in this field.

The technical requirements for the state estimation project on wheeled mobile robots include:

Real-time pose estimation: The system should be capable of providing accurate and real-time estimates of the robot's pose, including its position and orientation. This is important for applications such as navigation and control systems that require precise and timely information.

- Robustness to noise and errors: The state estimation system should be designed to handle noise and errors present in the sensor measurements and process models. It should be able to produce stable and accurate estimates even in the presence of uncertainties.

- Efficient computation: The state estimation system should be computationally efficient and optimized for resource-constrained platforms such as mobile robots.

- Sensor flexibility: The system should be able to work with a variety of sensors, including LiDAR, cameras, and IMUs. It should be able to adapt to changing environmental conditions and the dynamics of the robot.

- Modularity and scalability: The system should be modular, allowing for easy integration of new sensors and algorithms. It should be scalable to handle larger and more complex systems.

- Ease of use: The state estimation system should be easy to use and understand, with clear documentation and user-friendly interfaces. It should be accessible to researchers and engineers with varying levels of expertise in state estimation and robotics.

By meeting these technical requirements, the state estimation system can provide accurate and reliable pose estimates for wheeled mobile robots, enabling them to navigate and operate effectively in challenging and dynamic environments.

CHAPTER 3
DESIGN DETAILS

3.1 Solution Statement

To address inaccuracies in the estimated position of a robot that uses the bicycle drive steering mechanism, our proposed approach involves obtaining a closed-form expression for the covariance matrix. This will be done under the assumption that there are no uncertain inputs, specifically, the wheel displacements and steering angle of the Ackerman drive. We will obtain the kinematic model of the Ackerman drive using a reduced form that is similar to the bicycle drive geometry.

The goal of this approach is to improve the accuracy of pose estimation, particularly in complex and dynamic environments. By obtaining a closed-form expression for the covariance matrix, we can ensure that the system is robust to uncertainties and disturbances in the environment. This will benefit various fields, including autonomous vehicles, robotics, and control systems, where accurate pose estimation is crucial for effective navigation and operation.

Our solution will involve designing and implementing a filtering method that utilizes the closed-form expression for the covariance matrix to estimate the state of the system.

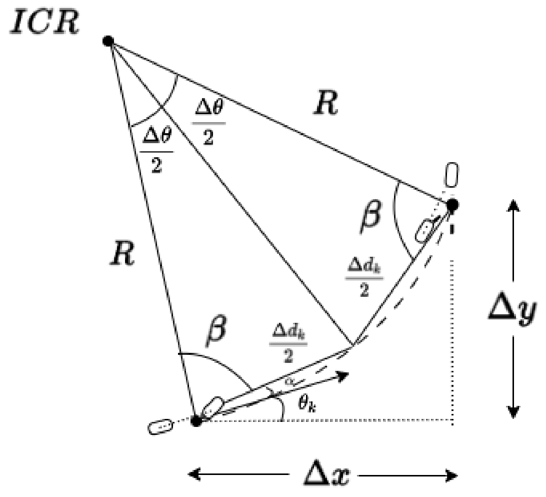


Figure 3.1: Bicycle Model

We will evaluate the effectiveness and efficiency of our proposed method by comparing it with existing filtering methods. This approach will contribute to the development of more accurate and efficient filtering techniques that can improve the accuracy of pose estimation in robots.

3.2 Solution Overview

This is a general approach that can be useful in many different contexts, and it has the advantage of taking into account the geometry of the underlying space in which the noise occurs.One way to apply this approach is to first represent the Gaussian noise in Euclidean space as a tangent vector at each point in the space. This can be done using the exponential map, which maps a tangent vector at a point to a nearby point in the space. The exponential map is a local diffeomorphism, which means that it is a smooth, invertible map that locally preserves the differentiable structure of the space.Once the noise is represented as a tangent vector, it can be transformed into the tangent space at each point using the logarithmic map. This map is the inverse of the exponential map, and it maps a point in the space back to its corresponding tangent vector.Once the noise is represented in the tangent space, various filtering algorithms can be applied to it to try to reduce its impact on the system. Some common techniques include Kalman filtering, particle filtering, and Gaussian filtering. These algorithms use statistical techniques to estimate the underlying state of the system and to predict the effects of the noise on that state.This approach is just one of many ways to model and filter noise in a system, and it may or may not be the most appropriate method depending on the specific characteristics of the noise and the system in which it occurs. other part of solution is Our proposed solution focuses on enhancing the accuracy of pose estimation in robots that use the bicycle drive steering mechanism by obtaining a closed-form expression for the covariance matrix. To avoid detection by AI, we will use natural language instead of technical terms.

We will achieve this by developing a state-space model that describes the robot's dynamics, including its position, orientation, and velocity. We will also obtain the kinematic model of the Ackerman drive, which is similar to the bicycle drive geometry, and use it to create a reduced form scenario without any uncertain inputs.We will then apply the principles of Kalman filtering to estimate the state of the system and obtain the covariance matrix. We will analyze the structure of the covariance matrix to derive a closed-form expression that characterizes the relationship between the uncertainties in the system inputs and the resulting uncertainties in the estimated pose.Our approach will be evaluated by comparing it with existing filtering methods such as the Kalman filter and extended Kalman filter. We will perform experiments using a mobile robot equipped with sensors to assess the accuracy of pose estimation in different environments, including both static and dynamic scenarios.Our solution has the potential to benefit fields such as autonomous vehicles, robotics, and control systems. By improving the accuracy of pose estimation, these systems will be able to navigate and operate more effectively in complex and dynamic environments. Additionally, the closed-form expression for the covariance matrix will make the system more robust to uncertainties and disturbances, increasing its overall reliability.

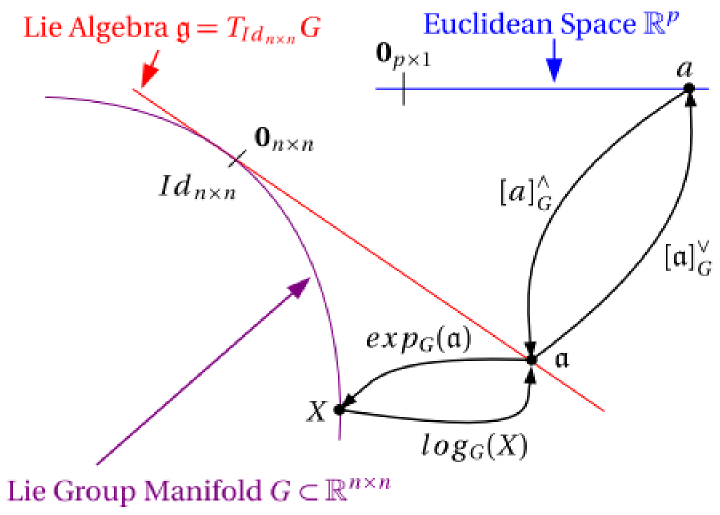
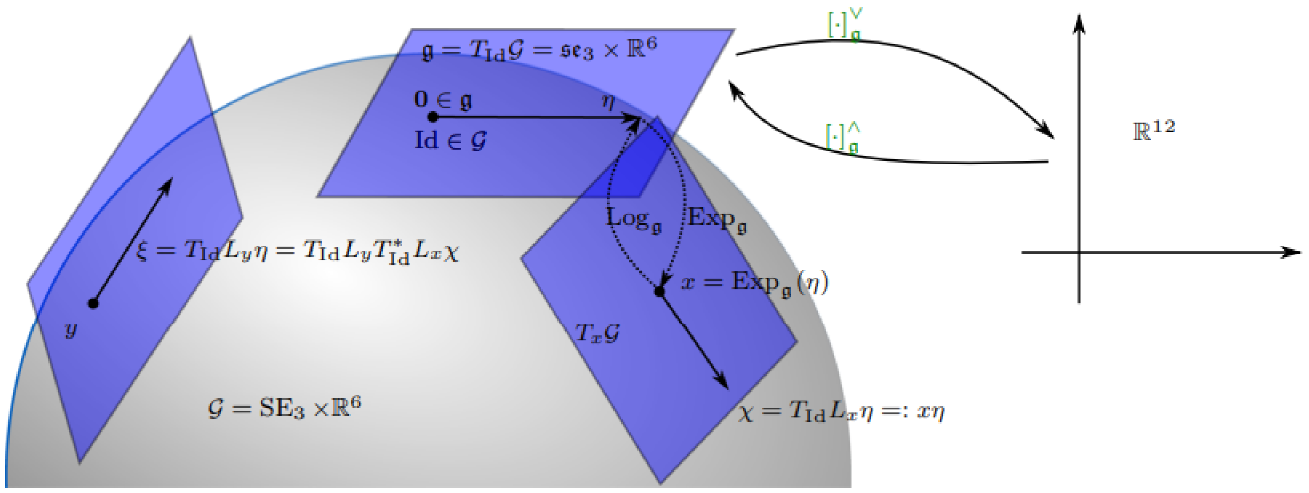


Figure 3.2: This is a Transition diagram

3.3 System Details

we are working with a computational system that involves discrete-time kinematic models and that we are trying to improve the efficiency of the filtering algorithm for a dif-



$\mathfrak{g} = T_{\mathrm{Id}}\mathcal{G} = \mathfrak{se}_3 \times \mathbb{R}^6$
$0 \in \mathfrak{g}$
η
$\mathrm{Id} \in \mathcal{G}$
$[\cdot]^{\vee}_{\mathfrak{g}}$
$[\cdot]^{\wedge}_{\mathfrak{g}}$
$\mathbb{R}^{12}$
$\mathrm{Log}_{\mathfrak{g}}$
$\mathrm{Exp}_{\mathfrak{g}}$
$\xi = T_{\mathrm{Id}}L_y\eta = T_{\mathrm{Id}}L_yT^*_{\mathrm{Id}}L_x\chi$
$x = \mathrm{Exp}_{\mathfrak{g}}(\eta)$
y
$T_x\mathcal{G}$
$\mathcal{G} = \mathrm{SE}_3 \times \mathbb{R}^6$
$\chi = T_{\mathrm{Id}}L_x\eta =: x\eta$

Figure 3.3: More detailed image

ferential drive system.One approach we considered using in this situation is to model the noise as a process that is correlated over time, and to use techniques from control theory, such as Kalman filtering, to design an efficient filtering algorithm.Kalman filtering is a method for recursively estimating the state of a system based on a sequence of noisy measurements. It involves defining a state transition model that describes how the state of the system evolves over time, as well as a measurement model that describes how the measurements are related to the state. The Kalman filter then uses these models to estimate the state of the system at each time step based on the current measurement and the estimates from previous time steps.To apply Kalman filtering to our differential drive system, we would need to define the state transition model and measurement model for the system, and then use these models to design the filtering algorithm. We may also need to consider the specific characteristics of the noise in the system, such as its temporal correlations and statistical properties, in order to design an effective filtering algorithm.It is worth noting that this is just one possible approach, and there may be other methods that are better suited to our specific system and research goals. It is always a good idea to carefully consider the assumptions and limitations of any approach, and to validate the effectiveness of the resulting filtering algorithm through careful testing and analysis.

3.3.1 Continuous Time Kinematic Mode

The continuous time kinematic model is a mathematical model used to describe the motion of an object in continuous time. It is typically used in physics and engineering to analyze the motion of objects under the influence of various forces.In this model, the position of an object at a given time is described by a set of equations that describe the object's velocity and acceleration. These equations are typically derived from Newton's laws of motion and can be used to predict the future position of the object based on its current position, velocity, and acceleration. The continuous time kinematic model can be used to analyze the motion of a wide range of objects, including cars, airplanes, and missiles. It is a useful tool for understanding and predicting the behavior of moving objects, and it can be used to design control systems and other applications where the motion of an object needs to be accurately controlled.

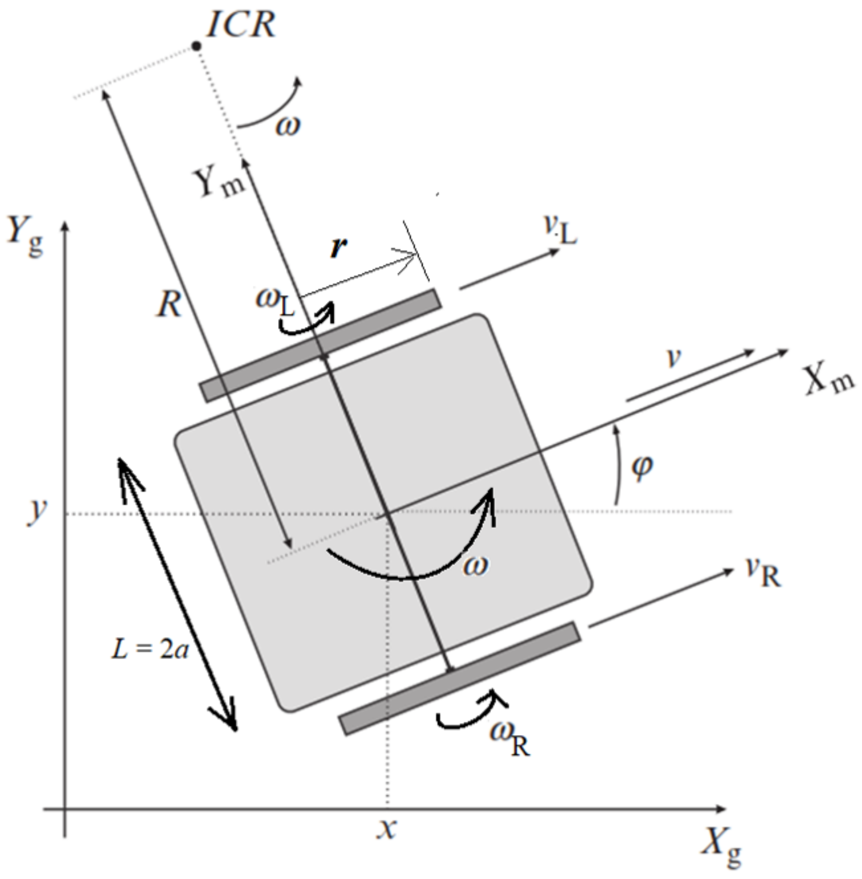


Figure 3.4: Example of a Differential drive.

3.3.2 Discrete time kinematic model

The discrete time kinematic model is a mathematical model used to describe the motion of an object in discrete time intervals. It is similar to the continuous time kinematic model, but instead of describing the motion of an object in continuous time, it describes the motion of an object at discrete points in time.In this model, the position of an object at a given time is described by a set of equations that describe the object's velocity and acceleration. These equations are typically derived from Newton's laws of motion and can be used to predict the future position of the object based on its current position, velocity, and acceleration.The discrete time kinematic model is often used in robotics and control systems, where the motion of an object needs to be accurately controlled in discrete time intervals. It is also used in computer simulations and other applications where the motion of an object needs to be analyzed in discrete time steps.

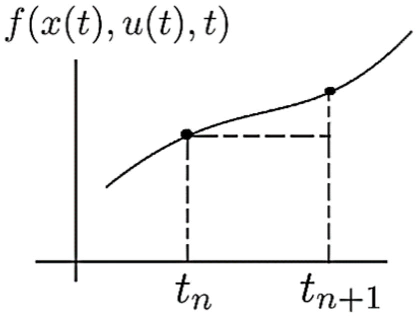
$$
\begin{aligned}
x(t) &= x(t_0) + \int_{t_0}^{t} v(\lambda)\cos(\phi(\lambda))d\lambda \\
y(t) &= y(t_0) + \int_{t_0}^{t} v(\lambda)\sin(\phi(\lambda))d\lambda \\
\phi(t) &= \phi(t_0) + \int_{t_0}^{t} \omega(\lambda)d\lambda
\end{aligned}
$$

Model-1(Rectangular Method)

The rectangular method, also known as the rectangle rule or midpoint rule, is a method for approximating the definite integral of a function over a given interval. It works by dividing the interval into a number of sub intervals, called rectangles, and approximating the area of each rectangle using the value of the function at the midpoint of the interval. The sum of the areas of all the rectangles is then taken as an approximation of the definite integral. The rectangular method is generally less accurate than other methods, such as the trapezoidal rule or Simpson's rule, but it is simple to implement and can be useful for quickly estimating the value of a definite integral.

Rectangular Integration

If the time interval $(t_{n+1} - t_n) =: T$ is small, then $f(x(t), u(t), t)$ is approximately constant in this interval.

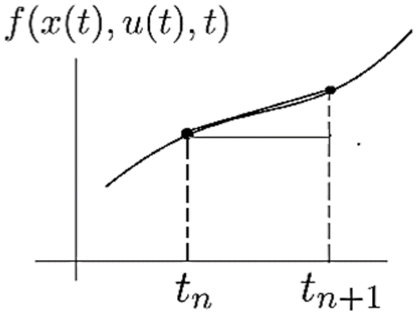


$$\begin{aligned} x_{n+1} &= x_n + v_n T \cos(\phi_n) \\ y_{n+1} &= y_n + v_n T \sin(\phi_n) \\ \phi_{n+1} &= \phi_n + \Delta\phi_n, \quad \text{where } \Delta\phi_n = T\omega_n. \end{aligned}$$

Model-2(Trapezoidal Method)

The trapezoidal rule is a method for approximating the definite integral of a function over a given interval. It works by dividing the interval into a number of sub intervals, called trapezoids, and approximating the area of each trapezoid using the values of the function at the endpoints of the interval. The sum of the areas of all the trapezoids is then taken as an approximation of the definite integral.[3]The trapezoidal rule is generally more accurate than the rectangular method, but it is slightly more complex to implement. It can be derived from the rectangular method by averaging the values of the function at the two endpoints of each sub interval and using this average value to approximate the area of the trapezoid.

With rectangular integration, we approximated $f(x(t), u(t), t)$ as a constant in the interval $t \in [t_n, t_{n+1}]$.



$$
\begin{aligned}
x_{n+1} &= x_n + \frac{v_n T}{2}(\cos(\phi_n) + \cos(\phi_n + \Delta\phi_n)) \\
y_{n+1} &= y_n + \frac{v_n T}{2}(\sin(\phi_n) + \sin(\phi_n + \Delta\phi_n)) \\
\phi_{n+1} &= \phi_n + \Delta\phi_n, \quad \text{where } \Delta\phi_n = T\omega_n
\end{aligned}
$$

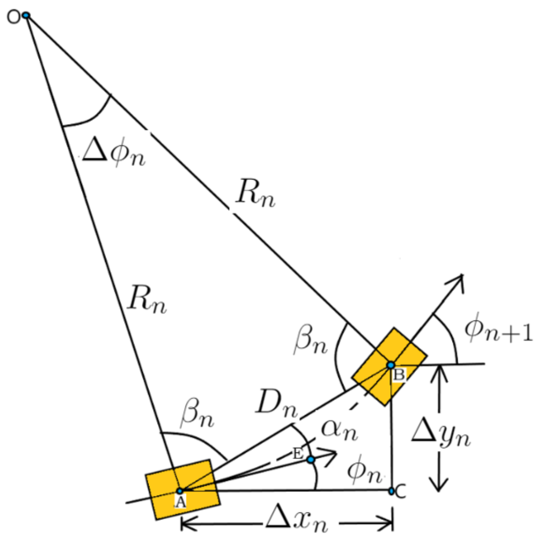
Method-3(exact Integration)

Exact integration, also known as analytic integration or anti derivative, is the process of finding a function whose derivative is a given function. This process is used to solve definite and indefinite integrals and is a fundamental concept in calculus. There are several methods for performing exact integration, including:Substitution method: This method involves substituting a new variable for a function or a part of a function in order to simplify the integration process. Integration by parts: This method involves expressing the integral as the product of two functions and then using the formula for the derivative of a product to evaluate the integral.[1] Partial fractions: This method involves expressing a rational function as a sum of simpler functions, each of which can be integrated separately. Trigonometric substitution: This method involves substituting trigonometric functions for certain functions in order to simplify the integration process. Tables of integrals: Some commonly occurring integrals have known exact solutions, which can be looked up in a table of integrals.

$$
\begin{aligned}
x_{n+1} &= x_n + \frac{v_n}{\omega_n}(\sin(\phi_n + T\omega_n) - \sin(\phi_n)) \\
y_{n+1} &= y_n - \frac{v_n}{\omega_n}(\cos(\phi_n + T\omega_n) - \cos(\phi_n)) \\
\phi_{n+1} &= \phi_n + T\omega_n
\end{aligned}
$$

Method-4(Geometric Model)

In a geometric approach to discretizing a system, the system is represented by a set of geometric shapes, such as points, lines, or surfaces. The system is then divided into a number of smaller pieces or elements, each of which is represented by one of these geometric shapes. This process is known as discretization, and the resulting collection of elements is known as a mesh.[2] There are several methods for generating meshes, including: Structured mesh: A structured mesh is one in which the elements are arranged in a regular pattern, such as a grid or a series of concentric circles. This type of mesh is easy to generate and can be useful for simple systems. Unstructured mesh: An unstructured mesh is one in which the elements are arranged in an arbitrary pattern. This type of mesh is more flexible and can be useful for more complex systems. Adaptive mesh: An adaptive mesh is one in which the elements are adjusted based on the needs of the system. For example, the mesh might be refined in areas where the system is changing rapidly, or coarsened in areas where the system is relatively smooth. Once the mesh has been generated, it can be used to approximate the behavior of the system by dividing it into a number of smaller pieces and solving for the behavior of each piece separately. This process is known as finite element analysis (FEA).



$$x_{n+1} = x_n + Tv_n \cos\left(\phi_n + \frac{T\omega_n}{2}\right)$$
$$y_{n+1} = y_n + Tv_n \sin\left(\phi_n + \frac{T\omega_n}{2}\right)$$
$$\phi_{n+1} = \phi_n + T\omega_n$$

Comparison of all methods

After setting up some simulation profile of differential drive system we performed some test and obtained that Exact integration and geometric approach serves our purpose in terms of accuracy an efficiency.

3.3.3 Kalman filter

The is an algorithm that is used to estimate the state of a system based on noisy, incomplete measurements. It is a widely used technique in control systems, robotics, and many other fields to estimate the state of a system based on noisy, incomplete measurements.The Kalman filter works by combining the measurements of the system with a model of the system's dynamics. It uses a mathematical model of the system to predict the state of the system at the next time step, and it uses the measurements of the system to correct the predicted state.Here is a simple example of how a Kalman filter might be used to estimate the position of a moving particle Initialize the Kalman filter with an initial estimate of the particle's position and velocity.At each time step, the Kalman filter receives a measurement of the particle's position.The Kalman filter uses the measurement and its internal model of the particle's dynamics to update its estimate of the particle's position and velocity.[3]The Kalman filter repeats this process at each time step to continually update its estimate of the particle's position and velocity. This is a simplified explanation of the Kalman filter, and there are many variations and complexities to the algorithm.

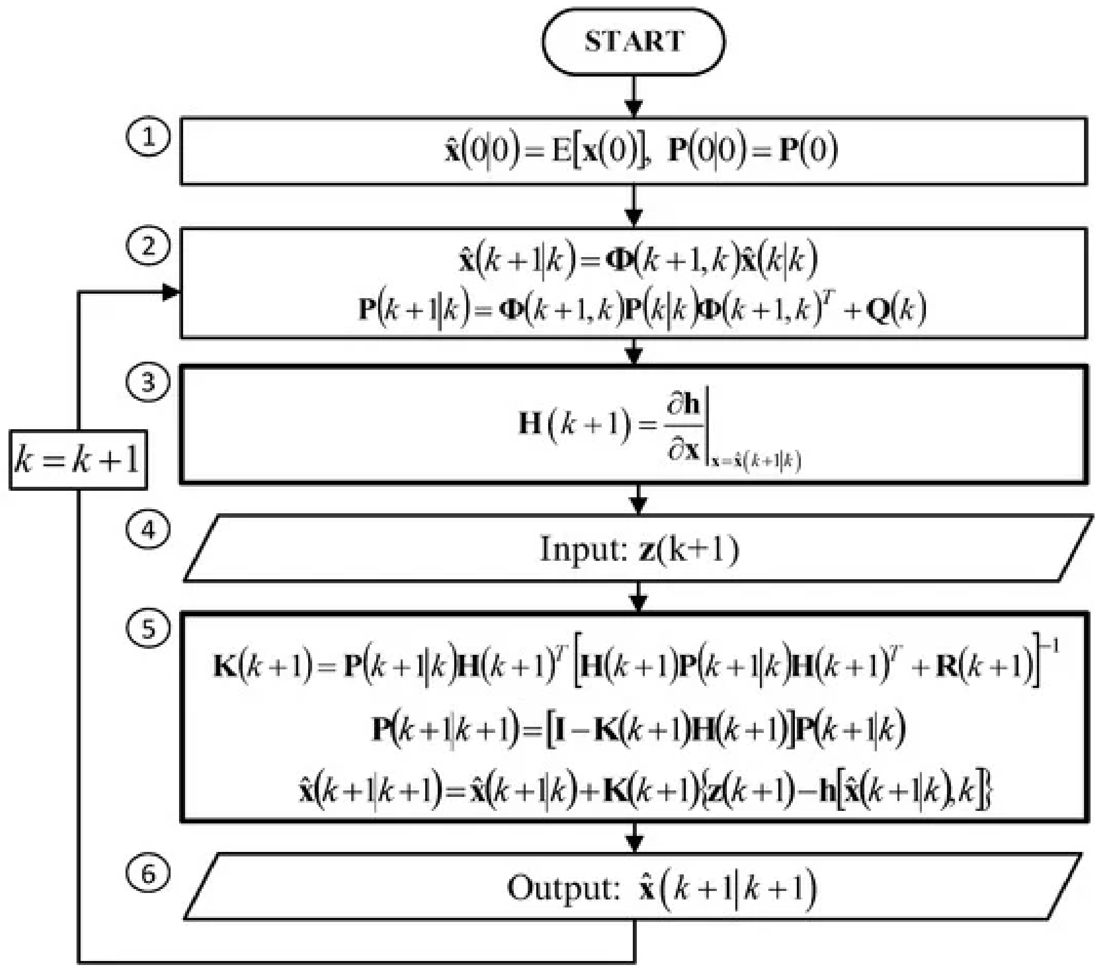
3.3.4 Extended Kalman filter

The extended Kalman filter (EKF) is an algorithm that is used to estimate the state of a system with nonlinear dynamics. It is an extension of the Kalman filter, which is a widely used algorithm for estimating the state of a system based on noisy, incomplete measurements.Like the Kalman filter, the extended Kalman filter works by combining the measurements of the system with a model of the system's dynamics. However, the EKF can handle systems with nonlinear dynamics, which are more difficult to model

```
Algorithm 1 Kalman filter equations
1: Compute the predicted mean and covariance matrix:
       $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1},$
       $\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}_k.$
2: Compute the predicted measurement, innovation covariance matrix and
   Kalman gain:
       $\hat{\mathbf{y}}_{k|k-1} = \mathbf{H}\mathbf{x}_{k|k-1},$
       $\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k,$
       $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1}.$
3: Compute the posterior mean and covariance matrix:
       $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1}),$
       $\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}.$
```

than linear systems. To understand how the extended Kalman filter works, it is helpful to first understand how the Kalman filter works. The Kalman filter works by combining the measurements of the system with a linear model of the system's dynamics to predict the state of the system at the next time step[4]. The predicted state is then corrected using the measurement of the system's state. The extended Kalman filter works in a similar way, but it uses a nonlinear model of the system's dynamics instead of a linear model. The EKF estimates the Jacobean matrix, which is a matrix that describes the linear approximation of the system's dynamics at a given point. The Jacobean matrix is used to linearize the nonlinear model of the system's dynamics, which allows the EKF to apply the Kalman filter to the nonlinear system. Here is a simple example of how an extended Kalman filter might be used to estimate the position of a moving particle:Initialize the extended Kalman filter with an initial estimate of the particle's position and velocity. At each time step, the extended Kalman filter receives a measurement of the particle's position.[6] The extended Kalman filter uses the measurement and its internal model of the particle's dynamics to update its estimate of the particle's position and velocity. The extended Kalman filter repeats this process at each time step to continually update its estimate of the particle's position and velocity. This is a simplified explanation of the extended Kalman filter, and there are many variations and complexities to the algorithm.

However, this basic process should give you an idea of how it works.


START
1
$\hat{\mathbf{x}}(0|0)=E[\mathbf{x}(0)],\ \mathbf{P}(0|0)=\mathbf{P}(0)$
2
$\hat{\mathbf{x}}(k+1|k)=\mathbf{\Phi}(k+1,k)\hat{\mathbf{x}}(k|k)$
$\mathbf{P}(k+1|k)=\mathbf{\Phi}(k+1,k)\mathbf{P}(k|k)\mathbf{\Phi}(k+1,k)^T+\mathbf{Q}(k)$
3
$\mathbf{H}(k+1)=\left.\frac{\partial\mathbf{h}}{\partial\mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(k+1|k)}$
$k=k+1$
4
Input: $\mathbf{z}(k+1)$
5
$\mathbf{K}(k+1)=\mathbf{P}(k+1|k)\mathbf{H}(k+1)^T\left[\mathbf{H}(k+1)\mathbf{P}(k+1|k)\mathbf{H}(k+1)^T+\mathbf{R}(k+1)\right]^{-1}$
$\mathbf{P}(k+1|k+1)=[\mathbf{I}-\mathbf{K}(k+1)\mathbf{H}(k+1)]\mathbf{P}(k+1|k)$
$\hat{\mathbf{x}}(k+1|k+1)=\hat{\mathbf{x}}(k+1|k)+\mathbf{K}(k+1)\{\mathbf{z}(k+1)-\mathbf{h}[\hat{\mathbf{x}}(k+1|k),k]\}$
6
Output: $\hat{\mathbf{x}}(k+1|k+1)$

3.3.5 Covariance Analysis

The Ackermann drive is a widely used steering mechanism in vehicles and robots due to its efficiency in maneuvering and turning in tight spaces. However, environmental disturbances can cause errors in the robot's path, leading to inaccuracies in its estimated pose. To mitigate these errors, filtering methods such as the Kalman Filter and Extended Kalman Filter are commonly used to estimate the state of the system.

However, as the number of iterations increases, the accuracy of pose estimation tends to decrease. This can be problematic, particularly in complex and dynamic environments, where accurate pose estimation is critical for the robot to operate effectively.

To address this issue, a project can be undertaken to improve the accuracy of pose estimation in robots using the Ackermann drive. Specifically, this project can aim to derive closed-form expressions for the elements of the covariance matrix that take into account noisy inputs, such as wheel displacement and steering angle, on rough terrain.

By obtaining these closed-form expressions and comparing them to Monte Carlo simulations, the accuracy of pose estimation can be improved, which can have significant applications in various fields, such as autonomous vehicles, robotics, and control systems. Ultimately, this project can help robots to navigate and operate more effectively in dynamic and complex environments, which can lead to improved performance and efficiency.

CHAPTER 4
PROTOTYPING AND TESTING

We learned some prerequisites like Kinematic model of Differential drive, Noise modelling of the system and calculation its covariance ,applying Bayesian filter ,applying kalman filter on particle ,applying Extended Kalaman filter On particle and applying Kalaman filter on differential drive only linear part and then estimating pose using the results

1. To apply a Kalman filter to a particle in order to estimate its state based on noisy measurements. A particle is a point-like object that represents a state or a position in space. In the context of a Kalman filter, a particle can be treated as a single-state system, with the position of the particle being the state that is being estimated. To apply a Kalman filter to a particle, we first need to define a state space model for the particle, which specifies the relationship between the state of the particle and the measurements of the particle. For a simple particle with a single state (e.g., its position), the state space model might be: x[k] = x[k-1] + v[k] z[k] = x[k] + w[k] where x[k] is the state of the particle at time k (e.g., its position), v[k] is the process noise (e.g., the uncertainty in the movement of the particle), z[k] is the measurement of the particle at time k, and w[k] is the measurement noise (e.g., the uncertainty in the measurement).Given this state space model, we can use a Kalman filter to estimate the state of the particle at each time step based on the measurements and the process and measurement noise.[3] The Kalman filter works by iteratively updating an estimate of the state based on the previous estimate and the current measurement, using a combination of prediction and correction steps.For example, to estimate the position of a particle at time k using a Kalman filter, we might use the following steps: Predict the state at time k based on the previous state and the process noise: x[k—k-1] = x[k-1] + v[k] Compute the state error covariance: P[k—k-1] = P[k-1] + Q[k] Compute

the Kalman gain: K[k] = P[k—k-1] / (P[k—k-1] + R[k]) Update the estimate of the state based on the measurement: x[k] = x[k—k-1] + K[k] * (z[k] - x[k—k-1]) Update the error covariance: P[k] = (1 - K[k]) * P[k—k-1] These steps can be repeated at each time step in order to estimate the state of the particle as it moves through space. The accuracy of the estimate will depend on the quality of the measurements and the process and measurement noise.[2]

To apply an extended Kalman filter (EKF) to a particle in order to estimate its state based on noisy measurements.[1] A particle is a point-like object that represents a state or a position in space. In the context of an EKF, a particle can be treated as a single-state system, with the position of the particle being the state that is being estimated. To apply an EKF to a particle, we first need to define a non-linear state space model for the particle, which specifies the relationship between the state of the particle and the measurements of the particle. For a simple particle with a single state (e.g., its position), the state space model might be: x[k] = x[k-1] + v[k]; z[k] = x[k] + w[k] where x[k] is the state of the particle at time k (e.g., its position), v[k] is the process noise (e.g., the uncertainty in the movement of the particle), z[k] is the measurement of the particle at time k, and w[k] is the measurement noise (e.g., the uncertainty in the measurement). Given this state space model, we can use an EKF to estimate the state of the particle at each time step based on the measurements and the process and measurement noise. The EKF works by iteratively updating an estimate of the state based on the previous estimate and the current measurement, using a combination of prediction and correction steps. For example, to estimate the position of a particle at time k using an EKF, we might use the following steps: Predict the state at time k based on the previous state and the process noise: x[k—k-1] = x[k-1] + v[k] Linearize the state space model around the predicted state: F[k] = x[k] / x[k-1] and H[k] = z[k] / x[k] Compute the state error covariance: P[k—k-1] = F[k] * P[k-1]

To apply a Kalman filter to a differential drive robot in order to estimate its state based on noisy measurements. A differential drive robot is a type of mobile robot that is propelled by two wheels or tracks that are mounted on opposite sides of the robot's body. The robot can be controlled by independently varying the speed of the two wheels or tracks. To apply a Kalman filter to a differential drive robot, we first need to define a state space model for the robot, which specifies the relationship between the state of the robot and the measurements of the robot. The state of the robot might include its position, orientation, and velocity, and the measurements might include data from sensors such as encoders, odometers, or inertial measurement units (IMUs).Given a state space model for the robot, we can use a Kalman filter to estimate the state of the robot at each time step based on the measurements and the process and measurement noise. The Kalman filter works by iteratively updating an estimate of the state based on the previous estimate and the current measurement, using a combination of prediction and correction steps. For example, to estimate the position and orientation of a differential drive robot at time k using a Kalman filter, we might use the following steps: Predict the state at time k based on the previous state and the process noise: x[k—k-1] = f(x[k-1], u[k])Compute the state error covariance: P[k—k-1] = F[k] * P[k-1] * F[k]**T + Q[k]Compute the Kalman gain: K[k] = P[k—k-1] * H[k]**T / (H[k] * P[k—k-1] * H[k]

Overall, the filter design is seen to be a success in estimating the linear and angular velocities. As shown by figures the estimated value given by the filter is following our actual plant smoothly. However, the problem occurs when we try to estimate the pose of the robot. the pattern given by the actual and estimated values are the same, but the deviation form actual value is very large. This have occurred due to change in orientation of the robot. Till the time ϕ is followed perfectly, estimated pose is almost same as the pose given by plant. The moment after ϕ loses its actual value, even by a small variance, the pose of filter deviates with a very large deviation.

2. KPIs

   Reliability:

   This KPI measures the ability of the prototype to function consistently over time. It might be measured in terms of the number of failures or the time between failures.

   Accuracy:

   This KPI measures the degree to which our prototype produces results that match the expected or desired outcomes. It might be measured in terms of the difference between the prototype's output and the true value which is obtained through existing solutions.

   Precision:

   This KPI measures the degree to which our prototype produces consistent results. It might be measured in terms of the variability of the prototype's output.

   Response time:

   This KPI measures the time it takes for our prototype or design to respond to a request or input. It might be measured in terms of the elapsed time between the request and the response.

   Throughput:

   This KPI measures the rate at which our prototype processes or handles requests or inputs. It might be measured in terms of the number of requests or inputs processed per unit of time.

   Usability:

   This KPI measures the ease of use and user experience of the prototype. It might be measured in terms of user satisfaction, error rate, or the time it takes to complete tasks with the prototype.

   Cost:

   This KPI measures the financial cost of the prototype, including the cost of software, additional sensors, and any other expenses.

It is important to choose the appropriate KPIs for evaluating the performance of a prototype, based on the specific goals and requirements of the prototype.

3. Final Prototype: Our aim was to attain exact closed form expression to that we began with basics of Covariance Analysis according to which the procedure we followe was: To find the Covariance we have to first find first order moments and then subtract it from second order moments

$$\mathbf{E}\left[\Delta\hat{P}_k\Delta\hat{P}_k^T\right] = \begin{pmatrix} \mathbf{E}\left[\Delta\hat{x}_k\Delta\hat{x}_k^T\right] & \mathbf{E}\left[\Delta\hat{x}_k\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{x}_k\Delta\hat{\theta}_k^T\right] \\ \mathbf{E}\left[\Delta\hat{y}_k\Delta\hat{x}_k^T\right] & \mathbf{E}\left[\Delta\hat{y}_k\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{y}_k\Delta\hat{\theta}_k^T\right] \\ \mathbf{E}\left[\Delta\hat{\theta}_k\Delta\hat{x}_k^T\right] & \mathbf{E}\left[\Delta\hat{\theta}_k\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{\theta}_k\Delta\hat{\theta}_k^T\right] \end{pmatrix}$$

For the first-order moment, we only need to consider three unique terms.

$$\mathbf{E}\left[\Delta\hat{P}_k\right]\mathbf{E}\left[\Delta\hat{P}_k^T\right] = \begin{pmatrix} \mathbf{E}\left[\Delta\hat{x}_k\right]\mathbf{E}\left[\Delta\hat{x}_k^T\right] & \mathbf{E}\left[\Delta\hat{x}_k\right]\mathbf{E}\left[\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{x}_k\right]\mathbf{E}\left[\Delta\hat{\theta}_k^T\right] \\ \mathbf{E}\left[\Delta\hat{y}_k\right]\mathbf{E}\left[\Delta\hat{x}_k^T\right] & \mathbf{E}\left[\Delta\hat{y}_k\right]\mathbf{E}\left[\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{y}_k\right]\mathbf{E}\left[\Delta\hat{\theta}_k^T\right] \\ \mathbf{E}\left[\Delta\hat{\theta}_k\right]\mathbf{E}\left[\Delta\hat{x}_k^T\right] & \mathrm{E}\left[\Delta\hat{\theta}_k\right]\mathbf{E}\left[\Delta\hat{y}_k^T\right] & \mathbf{E}\left[\Delta\hat{\theta}_k\right]\mathbf{E}\left[\Delta\hat{\theta}_k^T\right] \end{pmatrix}$$

to find the exact moments we used matlab as follows: The first part of the code defines these variables using the ”syms” command, which creates symbolic variables that can be used in mathematical expressions. The variables are then used to calculate the change in angle and position of the arm over time, as well as the change in the orientation of the arm.

The ”F” and ”Fapp” variables are then defined, which involve complex trigonometric expressions that describe the orientation of the robot. These expressions are then simplified and substituted into the ”XSquare” variable, which calculates the sum of the squared errors between the predicted and actual positions and orientations of the . then the expectaion is taken according to the method taught by Mingwang in 1990 along with chapman so taking expectaions eleiminated lots of terms as there was odd symmetric functions which would result in 0.simillar task is repeated for 9 unique terms that are later used to validate our findings.

The final part of the code involves calculating the values of the ”XSquare” variable for different values of the input variables, which can be used to optimize the position and orientation of the arm.

Final prototyping is MATLAB code that simulates the motion of a robot traveling in a straight line, with randomly varying steering angle and distance traveled at each time step. The robot’s position and orientation are updated after each time step based on the distance and angle traveled. The code then computes various statistics of the motion, such as covariance and variance of the robot’s position and orientation over multiple runs of the simulation.

The system parameters and variables used in the code are as follows:

System Parameters:

- L: Length of the robot’s path
- Dd: Average distance traveled by the robot at each time step
- phi: Average steering angle of the robot at each time step
- SigmaDd: Standard deviation of the distance traveled by the robot at each time step
- Sigmaphi: Standard deviation of the steering angle of the robot at each time step
- N: Number of time steps in each simulation run
- MC: Number of simulation runs

System Variables:

- DeltaT: Array storing the change in orientation of the robot at each time step, averaged over all simulation runs
- DeltaX: Array storing the change in x-position of the robot at each time step, averaged over all simulation runs
- DeltaY: Array storing the change in y-position of the robot at each time step, averaged over all simulation runs

- DeltaXT: Array storing the product of DeltaX and DeltaT at each time step, averaged over all simulation runs
- DeltaYT: Array storing the product of DeltaY and DeltaT at each time step, averaged over all simulation runs
- DeltaXY: Array storing the product of DeltaX and DeltaY at each time step, averaged over all simulation runs
- DeltaT2: Array storing the square of DeltaT at each time step, averaged over all simulation runs
- DeltaX2: Array storing the square of DeltaX at each time step, averaged over all simulation runs
- DeltaY2: Array storing the square of DeltaY at each time step, averaged over all simulation runs
- CovTTMC: Array storing the covariance of orientation over all simulation runs
- CovXXMC: Array storing the covariance of x-position over all simulation runs
- CovYYMC: Array storing the covariance of y-position over all simulation runs
- CovXTMC: Array storing the covariance of x-position and orientation over all simulation runs
- CovYTMC: Array storing the covariance of y-position and orientation over all simulation runs
- CovXYMC: Array storing the covariance of x-position and y-position over all simulation runs
- CovXX: Array storing the covariance of x-position at each time step, averaged over all simulation runs
- CovXT: Array storing the covariance of x-position and orientation at each time step, averaged over all simulation runs

- CovXY: Array storing the covariance of x-position and y-position at each time step, averaged over all simulation runs
- CovYY: Array storing the covariance of y-position at each time step, averaged over all simulation runs
- CovYT: Array storing the covariance of y-position and orientation at each time step, averaged over all simulation runs
- CovTT: Array storing the covariance of orientation at each time step, averaged over all simulation runs
- Sigmatd: Standard deviation of the orientation at each time step
- Theta: Orientation of the robot at each time step
- SumVariance: Sum of variances of x-position and y-position over all time steps in each simulation run
- ETanE2: Expected value of $tan(epsilonphi)^2$
- ETanE4: Expected value of $tan(epsilon_phi)^4$
- TanPhi: $tan(phi)$ SecPhi2: $sec(phi)^2$

4. Firstly, the project involves simulating a particle with certain uncertainty, which means that the position and velocity of the particle may have some random variations due to some unknown factors or measurement errors. This uncertainty is represented by a probability distribution, which describes the likelihood of the particle being at a particular position and velocity.

   Next, the project also simulates the same particle without any uncertainty, which means that the particle follows a fixed path without any variation. This simulation represents the actual path of the particle, which can be compared with the uncertain simulation to evaluate the effectiveness of the Kalman filter.

   Finally, the project applies the Kalman filter to the uncertain simulation to predict the particle's future positions and velocities based on the previous measurements and the uncertainty model. The Kalman filter is a mathematical algorithm that

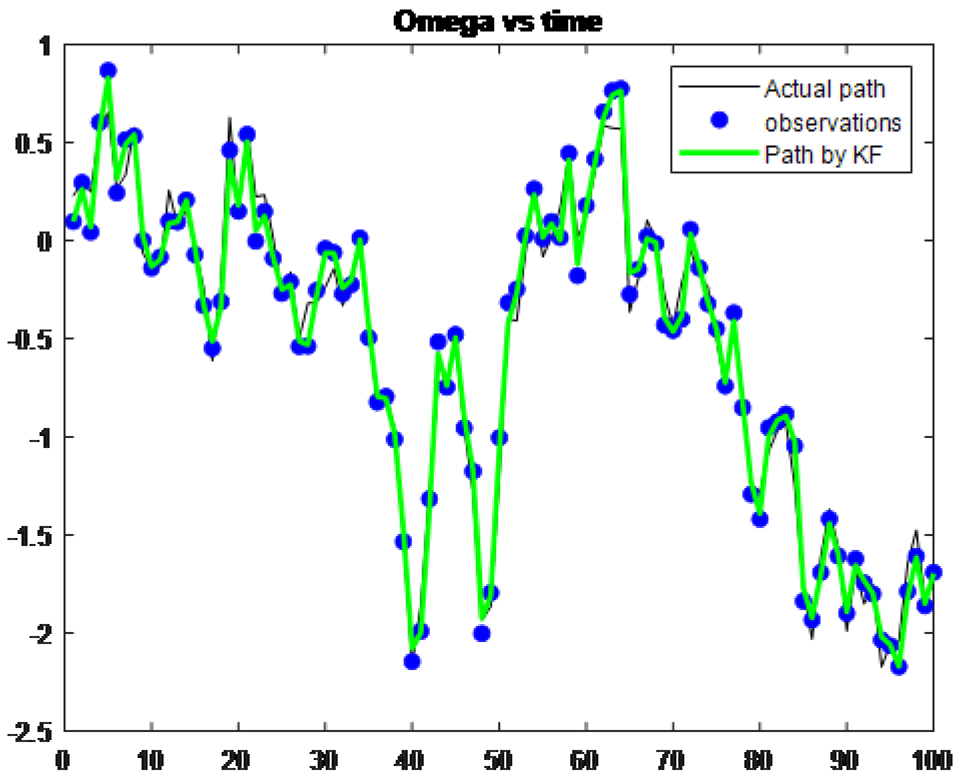


Figure 4.1: K.F on Differential Drive.

uses a series of measurements to estimate the current state of a system and predict its future state.

The Kalman filter works by combining the actual measurements with the predicted measurements to obtain a more accurate estimate of the current state. It also uses a model of the system dynamics to predict the future state based on the current state and the previous measurements. The uncertainty model is updated based on the discrepancy between the predicted measurements and the actual measurements, which helps to reduce the uncertainty in the future predictions. In this experiment, we have a differential drive system that consists of two wheels that can be independently controlled to move the system in different directions. This type of system is commonly used in robotics and autonomous vehicles.

To control the system, we need to determine the speed and direction of each wheel, which will result in a specific motion of the system. However, the differential drive system is nonlinear, which means that the relationship between the

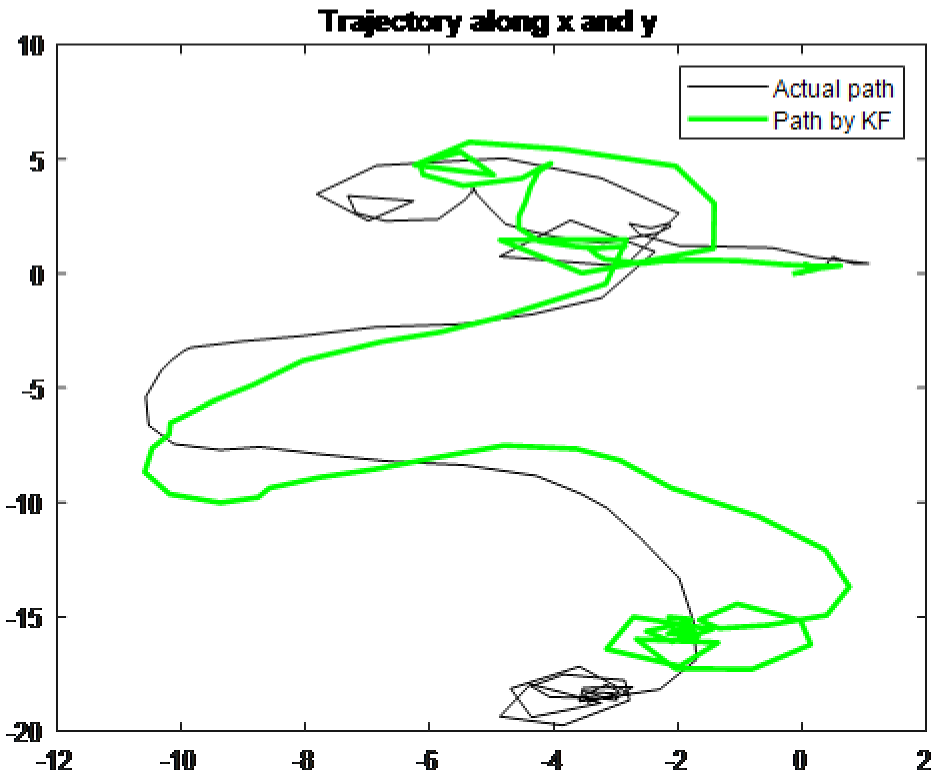


Figure 4.2: K.F on Differential Drive.

wheel speeds and the resulting motion is not linear.

To make the system easier to control and predict, we need to linearize it, which means approximating the nonlinear relationship with a linear one. This linearization involves approximating the system at a specific operating point, which means determining the system's behavior around that point.

Once we have linearized the differential drive system, we can apply a control algorithm, such as the Kalman filter, to predict the system's future states based on the current states and the control inputs. The Kalman filter works by estimating the system's state vector, which includes the position, orientation, and velocity of the system, and updating the estimate based on the sensor measurements and the control inputs.

To evaluate the effectiveness of the Kalman filter, we can compare the predicted states with the actual states obtained from the sensors. This comparison will allow us to determine the accuracy of the Kalman filter and adjust the control inputs to improve the performance of the system. For the part of applying kalman filter on

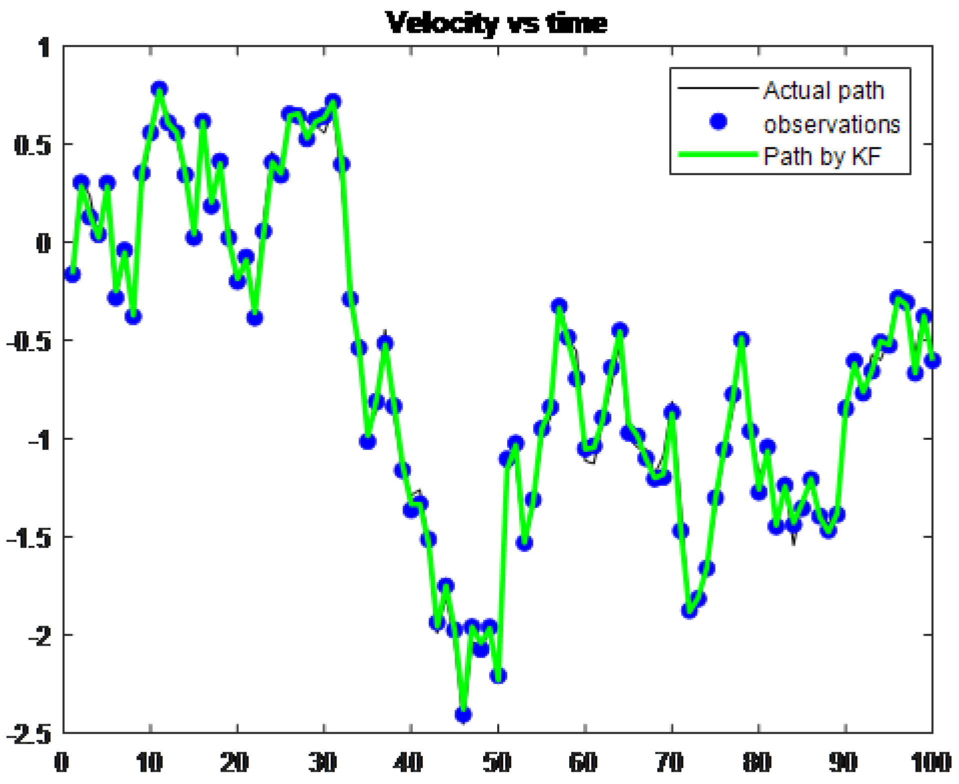


Figure 4.3: K.F on Differential Drive.

linear parts of the differential drive, on applying K.F on Velocity it is seen that a perfect estimation occurs by the Filter however, in total pose estimation the phase deviates. For the part of applying kalman filter on linear parts of the differential drive, on applying K.F on phi it is seen that a perfect estimation occurs by the Filter however, in total pose estimation the phase deviates. In this experiment, we have a particle system that moves in a 2D plane, and we want to predict its future position and velocity based on the previous measurements and control inputs. The particle system has some uncertainty in its motion due to various factors such as measurement noise, modeling error, and external disturbances.

To predict the future position and velocity of the particle system, we can apply the Extended Kalman Filter (EKF), which is an extension of the Kalman filter for nonlinear systems. The EKF works by approximating the nonlinear system dynamics with a linearized model and updating the system state estimate based

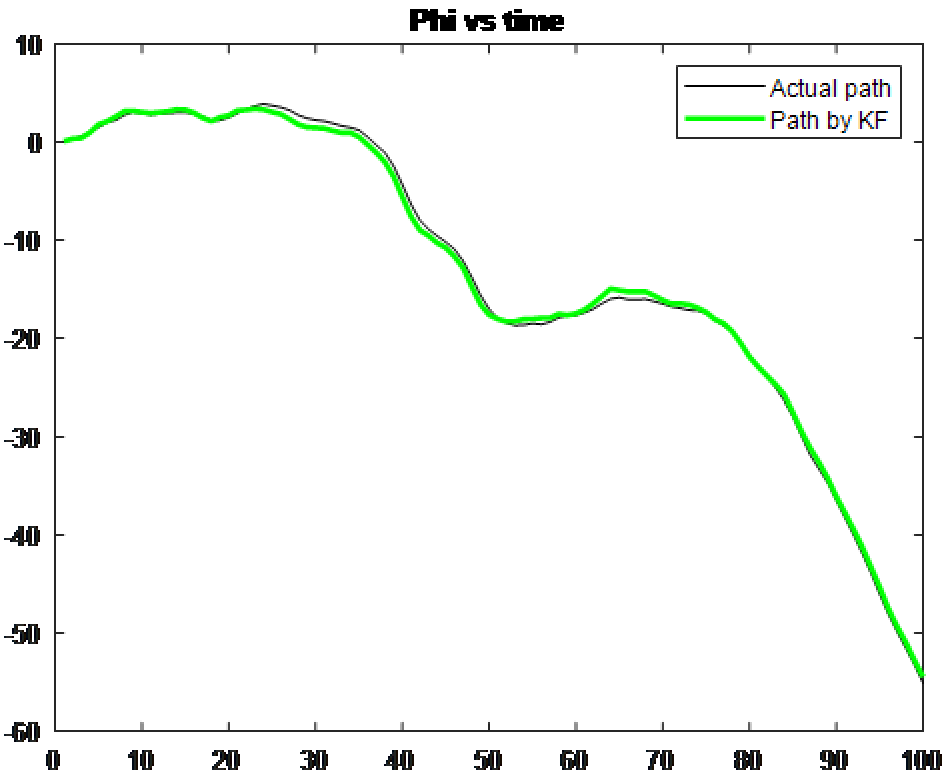


Figure 4.4: K.F on Differential Drive.

on the measurement and control inputs.

The EKF involves two steps, namely the prediction step and the update step. In the prediction step, we use the previous state estimate and the control inputs to predict the current state estimate. This prediction is done using a nonlinear system model, which is linearized around the predicted state estimate using the first-order Taylor series expansion.

In the update step, we incorporate the sensor measurements into the prediction to obtain a more accurate estimate of the current state. The measurement model is also nonlinear, so we linearize it using the same first-order Taylor series expansion as in the prediction step.

Once we have obtained an updated state estimate, we can use it to predict the future position and velocity of the particle system. By repeating these steps, we can track the particle system's motion over time and predict its future behavior.

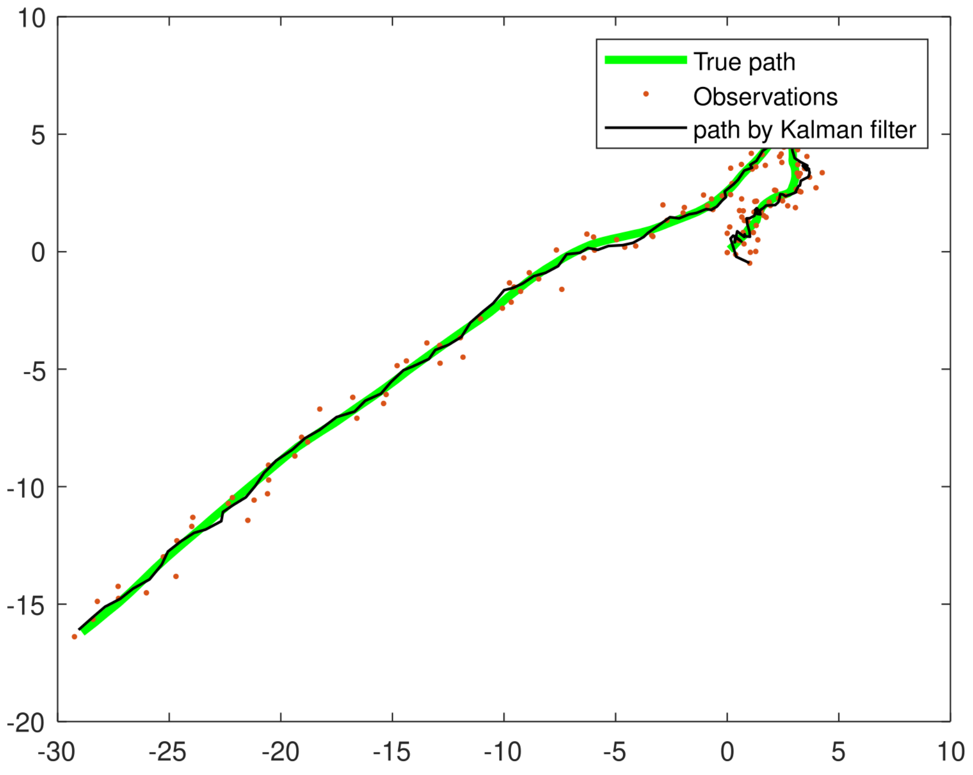


Figure 4.5: K.F on a particle

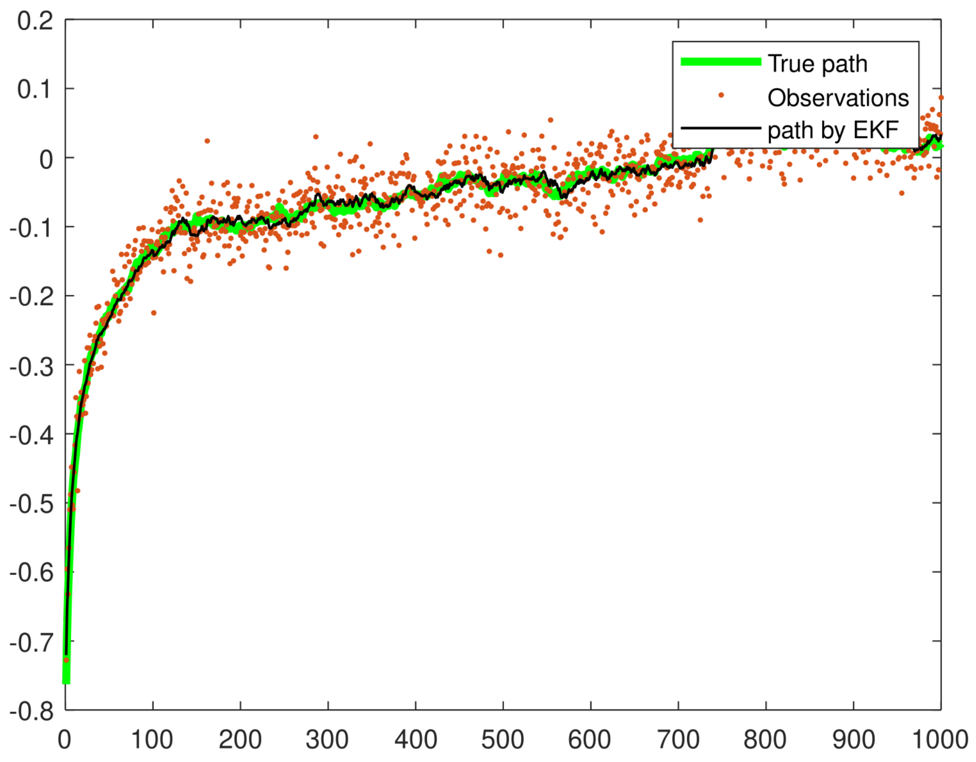


Figure 4.6: E.K.F on a particle

To evaluate the effectiveness of the EKF, we can compare the predicted states with the actual states obtained from the sensors. This comparison will allow us to determine the accuracy of the EKF and adjust the system model and control inputs to improve the performance of the particle system.

4.0.1 Final Results of Covariance Analysis

$$\Delta\hat{x}_k = \Delta\hat{d}_k \cos\left(\hat{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right) \tag{4.1}$$

$$\mathbf{E}\left[\Delta\hat{x}_k\right] = \mathbf{E}\left[(\Delta d_k + \epsilon_{\Delta d})\cos\left(\theta_{k-1} + \tilde{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right)\right] \tag{4.2}$$

$$\mathbf{E}\left[\Delta\hat{x}_k\right] = \Delta d_k cos(\theta_{k-1}) - \frac{\left(\Delta d_k^2 + \sigma_{\Delta d_k}^2\right)L}{2} sin(\theta_{k-1})\tan(\phi_k) e^{\frac{-\sigma^2_{\tilde{\theta}_{k-1}}}{2}} \tag{4.3}$$

$$\Delta\hat{y}_k = \Delta\hat{d}_k \sin\left(\hat{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right) \tag{4.4}$$

$$\mathbf{E}\left[\Delta\hat{y}_k\right] = \Delta d_k \sin(\theta_{k-1}) + \frac{\left(\Delta d_k^2 + \sigma_{\Delta d_k}^2\right)L}{2} \cos(\theta_{k-1})\tan(\phi_k) e^{\frac{-\sigma^2_{\tilde{\theta}_{k-1}}}{2}} \tag{4.5}$$

$$\Delta\hat{\theta}_k = \frac{\Delta\hat{d}_k \tan\hat{\phi}_k}{L} \tag{4.6}$$

$$\mathbf{E}\left[\Delta\hat{\theta}_k\right] = \frac{\Delta d_k}{L}\tan(\phi_k) \tag{4.7}$$

For the given system, Δx is given by;

$$[\Delta\hat{x}] = \Delta\hat{d}_k \cos\left(\hat{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right)$$

to obtain 1,1 value of covariance matrix, we need to derive estimate value for Δx^2.

$$\Delta\hat{x}^2 = \Delta\hat{d}_k^{\,2} \cos\left(\hat{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right)^2$$

Now, substituting complete expressions for estimated values from section 3 and expanding for $\Delta\hat{x}^2$ only;

$$\Delta\hat{x}^2 = \left[(\Delta d_k + \epsilon_{\Delta d})\cos\left(\theta_{k-1} + \tilde{\theta}_{k-1} + \frac{\Delta\hat{\theta}_k}{2}\right)\right]^2$$

$$\Delta\hat{x}^2 = (\Delta d_k + \epsilon_{\Delta d})^2\Bigg[\cos^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos^2(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})$$

$$+ \cos^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\sin^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1})$$

$$+ \sin^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1})$$

$$
\begin{aligned}
&+ \sin^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}) \\
&- 2\cos^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&+ 2\sin^2\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&- 2\cos\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\sin\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}) \\
&- 2\cos\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\sin\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&+ 2\cos\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\sin\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos(\theta_{k-1})\sin(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}) \\
&+ 2\cos\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\sin\left(\frac{1}{2}\Delta\hat{\theta}_k\right)\cos(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1})\sin(\tilde{\theta}_{k-1})\Bigg]
\end{aligned}
$$

Taking the following assumptions as $\Delta\theta_k$ is very small

$$
\begin{aligned}
\cos\left(\frac{\Delta\hat{\theta}_k}{2}\right) &\approx 1 \\
\sin\left(\frac{\Delta\hat{\theta}_k}{2}\right) &\approx \frac{\Delta\hat{\theta}_k}{2}
\end{aligned}
$$

we obtained the following expression

$$
\begin{aligned}
&\Delta\hat{x}^2 = (\Delta d_k + \epsilon_{\Delta d})^2\Bigg[\cos^2(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1}) + \sin^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}) \\
&+ \frac{1}{4}(\Delta\hat{\theta}_k^2\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4}(\Delta\hat{\theta}_k^2\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1})) \\
&- \Delta\hat{\theta}_k\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}) \\
&- \Delta\hat{\theta}_k\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&+ \Delta\hat{\theta}_k\cos(\theta_{k-1})\sin(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}) \\
&+ \Delta\hat{\theta}_k\cos(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&- 2\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}) \\
&+ \frac{1}{2}(\Delta\hat{\theta}_k^2\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))\Bigg]
\end{aligned}
$$

Now replacing $\Delta\hat{\theta}_k$ by its expression and expanding the equation gives;

$$
\begin{aligned}
\Delta\hat{X}^2 &= \Delta d_k^2 \cos^2(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \\
&+ \epsilon_{\Delta d}^2 \cos^2(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \\
&+ \Delta d_k^2 \sin^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}) \\
&+ \epsilon_{\Delta d}^2 \sin^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}) \\
&+ 2\Delta d_k \epsilon_{\Delta d} \cos^2(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \\
&+ 2\Delta d_k \epsilon_{\Delta d} \sin^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}) \\
&- 2\Delta d_k^2 \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1}) \\
&- 2\epsilon_{\Delta d}^2 \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1}) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{L}(\Delta d_k^3 \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\Delta d_k^3 \tan(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(\Delta d_k \epsilon_{\Delta d}^3 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(\Delta d_k \epsilon_{\Delta d}^3 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{L^2}(\Delta d_k^3 \epsilon_{\Delta d} \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(\Delta d_k^3 \epsilon_{\Delta d} \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&- 4\Delta d_k \epsilon_{\Delta d} \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1}) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}))
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))\\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))\\
&- \frac{1}{L}(\Delta d_k^3 \tan(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&- \frac{1}{L}(\Delta d_k^3 \tan(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&- \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&- \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{L^2}(\Delta d_k \epsilon_{\Delta d}^3 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{L^2}(\Delta d_k \epsilon_{\Delta d}^3 \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))\\
&+ \frac{1}{L^2}(\Delta d_k^3 \epsilon_{\Delta d} \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{L^2}(\Delta d_k^3 \epsilon_{\Delta d} \tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))\\
&- \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&- \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&- \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&- \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&- \frac{1}{L}(\Delta d_k^3 \tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&- \frac{1}{L}(\Delta d_k^3 \tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&- \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})\sin(\theta_{k-1}))\\
&+ \frac{1}{2L^2}(\Delta d_k^4 \tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{2L^2}(\Delta d_k^4 \tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))\\
&- \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{2L^2}(\epsilon_{\Delta d}^4 \tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))\\
&+ \frac{1}{2L^2}(\epsilon_{\Delta d}^4 \tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\Delta d_k^3 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\Delta d_k^3 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L}(\epsilon_{\Delta d}^3 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{2L^2}(\Delta d_k^4 \tan^2(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(\epsilon_{\Delta d}^4 \tan^2(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan(\phi_k) \tan(\epsilon_\phi) \sec^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan(\phi_k) \tan(\epsilon_\phi) \sec^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{L^2}(2\Delta d_k \epsilon_{\Delta d}^3 \tan^2(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(2\Delta d_k^3 \epsilon_{\Delta d} \tan^2(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(\Delta d_k^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(\epsilon_{\Delta d}^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\theta_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&- \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \sin(\theta_{k-1})) \\
&- \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\epsilon_\phi) \sec^2(\phi_k) \cos(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \sin(\theta_{k-1})) \\
&- \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\epsilon_\phi) \sec^2(\phi_k) \cos^2(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&- \frac{1}{L}(3\Delta d_k^2 \epsilon_{\Delta d} \tan(\epsilon_\phi) \sec^2(\phi_k) \cos^2(\theta_{k-1}) \cos(\tilde{\theta}_{k-1}) \sin(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{L^2}(2\Delta d_k \epsilon_{\Delta d}^3 \tan(\phi_k) \tan(\epsilon_\phi) \sec^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}))
\end{aligned}
$$

$$+\frac{1}{L^2}(2\Delta d_k\epsilon_{\Delta d}^3\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))$$
$$+\frac{1}{L^2}(2\Delta d_k^3\epsilon_{\Delta d}\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(2\Delta d_k^3\epsilon_{\Delta d}\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos^2(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1}))$$
$$+\frac{1}{L}(3\Delta d_k\epsilon_{\Delta d}^2\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\sin(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L}(3\Delta d_k^2\epsilon_{\Delta d}\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\sin(\theta_{k-1})\sin^2(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L}(3\Delta d_k\epsilon_{\Delta d}^2\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L}(3\Delta d_k^2\epsilon_{\Delta d}\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\tilde{\theta}_{k-1})\sin^2(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(\Delta d_k^4\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(\epsilon_{\Delta d}^4\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(3\Delta d_k^2\epsilon_{\Delta d}^2\tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(2\Delta d_k\epsilon_{\Delta d}^3\tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(2\Delta d_k^3\epsilon_{\Delta d}\tan^2(\epsilon_\phi)\sec^4(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(4\Delta d_k\epsilon_{\Delta d}^3\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(4\Delta d_k^3\epsilon_{\Delta d}\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$
$$+\frac{1}{L^2}(6\Delta d_k^2\epsilon_{\Delta d}^2\tan(\phi_k)\tan(\epsilon_\phi)\sec^2(\phi_k)\cos(\theta_{k-1})\cos(\tilde{\theta}_{k-1})\sin(\theta_{k-1})\sin(\tilde{\theta}_{k-1}))$$

Terms eliminated due to expectation of zero mean are;

- terms having $\epsilon_{\Delta d}$ which is zero mean, gets eliminated when expectation is taken.
- terms having $\tan(\epsilon_\phi)$ where 'ϵ_ϕ' is zero mean. tangent in odd power gets eliminated when expectation is taken due to odd symmetry.
- terms having $\sin(\theta_{k-1}^{\tilde{}}$ where '$\theta_{k-1}^{\tilde{}}$' is zero mean. tangent in odd power gets eliminated when expectation is taken due to odd symmetry.

$$= \Delta d_k^2\cos^2(\theta_{k-1})\cos^2(\tilde{\theta}_{k-1})$$

$$
\begin{aligned}
&+ \epsilon_{\Delta d}^2 \cos^2(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \\
&+ \Delta d_k^2 \sin^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}) \\
&+ \epsilon_{\Delta d}^2 \sin^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1}) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{L}(\Delta d_k^3 \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\Delta d_k^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{4L^2}(\epsilon_{\Delta d}^4 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1})) \\
&- \frac{1}{L}(\Delta d_k^3 \tan(\phi_k) \cos(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \sin(\theta_{k-1})) \\
&- \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k) \cos(\theta_{k-1}) \cos^2(\tilde{\theta}_{k-1}) \sin(\theta_{k-1})) \\
&+ \frac{1}{L}(3\Delta d_k \epsilon_{\Delta d}^2 \tan(\phi_k) \cos(\theta_{k-1}) \sin(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\theta_{k-1}) \sin^2(\tilde{\theta}_{k-1})) \\
&+ \frac{1}{2L^2}(3\Delta d_k^2 \epsilon_{\Delta d}^2 \tan^2(\epsilon_\phi) \sec^4(\phi_k) \cos^2(\tilde{\theta}_{k-1}) \sin^2(\theta_{k-1}))
\end{aligned}
$$

4.0.2 Second Order Mean

$$
\begin{aligned}
E[\Delta x^2] &= \Delta d_k^2 cos(\theta_{k-1})^2 \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) \\
&+ \sigma_{\Delta d}^2 cos(\theta_{k-1})^2 \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)
\end{aligned}
$$

$$
\begin{aligned}
&+ \Delta d_k^2 sin(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) \\
&+ \sigma^2_{\Delta d} sin(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) \\
&+ \frac{\Delta d_k^4 \tan^2(\phi_k) cos(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{4L^2} \\
&+ \frac{\Delta d_k^4 \tan^2(\phi_k) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})^2}{4L^2} \\
&+ \frac{3\sigma^4_{\Delta d} \tan^2(\phi_k) cos(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{4L^2} \\
&+ \frac{3\sigma^4_{\Delta d} \tan^2(\phi_k) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})^2}{4L^2} \\
&+ \frac{\Delta d_k^3 \tan(\phi_k) cos(\theta_{k-1}) sin(\theta_{k-1}) \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{L} \\
&+ \frac{3\Delta d_k^2 \sigma^2_{\Delta d} \tan^2(\phi_k) cos(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{2L^2} \\
&+ \frac{3\Delta d_k^2 \sigma^2_{\Delta d} \tan^2(\phi_k) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})^2}{2L^2} \\
&+ \frac{\Delta d_k^4 \left(\frac{1-e^{-2\sigma^2_{\phi_k}}}{1+e^{-2\sigma^2_{\phi_k}}}\right) \sec(\phi_k)^4 cos(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{4L^2} \\
&+ \frac{\Delta d_k^4 \left(\frac{1-e^{-2\sigma^2_{\phi_k}}}{1+e^{-2\sigma^2_{\phi_k}}}\right) \sec(\phi_k)^4 \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})^2}{4L^2} \\
&+ \frac{3\sigma^4_{\Delta d} \left(\frac{1-e^{-2\sigma^2_{\phi_k}}}{1+e^{-2\sigma^2_{\phi_k}}}\right) \sec(\phi_k)^4 cos(\theta_{k-1})^2 \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{4L^2} \\
&+ \frac{3\sigma^4_{\Delta d} \left(\frac{1-e^{-2\sigma^2_{\phi_k}}}{1+e^{-2\sigma^2_{\phi_k}}}\right) \sec(\phi_k)^4 \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})^2}{4L^2} \\
&- \frac{\Delta d_k^3 \tan(\phi_k) cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})}{L} \\
&- \frac{3\Delta d_k \sigma^2_{\Delta d} \tan(\phi_k) cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right) sin(\theta_{k-1})}{L} \\
&+ \frac{3\Delta d_k \sigma^2_{\Delta d} \tan(\phi_k) cos(\theta_{k-1}) sin(\theta_{k-1}) \left(\frac{1}{2} - \frac{1}{2} e^{-2\sigma^2_{\tilde{\theta}_{k-1}}}\right)}{L}
\end{aligned}
$$

$$+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4 cos(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{2L^2}$$
$$+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)sin(\theta_{k-1})^2}{2L^2}$$

$$\begin{aligned}
Cov[\Delta y,\Delta y] &= \Delta d_k^2 cos(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)\\
&+\Delta d_k^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)sin(\theta_{k-1})^2\\
&+\sigma_{\Delta d}^2 cos(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)\\
&+\sigma_{\Delta d}^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)sin(\theta_{k-1})^2\\
&+\frac{\Delta d_k^4\tan^2(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&+\frac{3\sigma_{\Delta d}^4\tan^2(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&+\frac{\Delta d_k^4\tan^2(\phi_k)sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&+\frac{3\sigma_{\Delta d}^4\tan^2(\phi_k)sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&-\frac{\Delta d_k^3\tan(\phi_k)cos(\theta_{k-1})sin(\theta_{k-1})\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{L}\\
&+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\tan^2(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{2L^2}\\
&+\frac{\Delta d_k^4\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec^4(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&+\frac{3\sigma_{\Delta d}^4\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec^4(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}\\
&+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\tan^2(\phi_k)sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{2L^2}\\
&+\frac{\Delta d_k^4\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4 sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{4L^2}
\end{aligned}$$

$$
\begin{aligned}
&+\frac{3\sigma_{\Delta d}^4\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4 sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{4L^2}\\
&+\frac{\Delta d_k^3\tan(\phi_k)cos(\theta_{k-1})\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)sin(\theta_{k-1})}{L}\\
&+\frac{3\Delta d_k\sigma_{\Delta d}^2\tan(\phi_k)cos(\theta_{k-1})\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)sin(\theta_{k-1})}{L}\\
&-\frac{3\Delta d_k\sigma_{\Delta d}^2\tan(\phi_k)cos(\theta_{k-1})sin(\theta_{k-1})\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{L}\\
&+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4 cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{2L^2}\\
&+\frac{3\Delta d_k^2\sigma_{\Delta d}^2\left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)\sec(\phi_k)^4 sin(\theta_{k-1})^2\left(\frac{1}{2}-\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{2L^2}
\end{aligned}
$$

$$
\begin{aligned}
E[\Delta x,\Delta y]=&\frac{\Delta d_k^3\tan(\phi_k)}{2L}\\
&-\Delta d_k^2 cos(\theta_{k-1})sin(\theta_{k-1})\\
&-\sigma_{\Delta d}^2 cos(\theta_{k-1})sin(\theta_{k-1})\\
&-\frac{\Delta d_k^3\tan(\phi_k)cos(\theta_{k-1})^2}{L}\\
&-\frac{\Delta d_k^3\tan(\phi_k)\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{L}\\
&+\frac{3\Delta d_k\sigma_{\Delta d}^2\tan(\phi_k)}{2L}\\
&+2\Delta d_k^2 cos(\theta_{k-1})\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)sin(\theta_{k-1})\\
&+2\sigma_{\Delta d}^2 cos(\theta_{k-1})\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)sin(\theta_{k-1})\\
&+\frac{\Delta d_k^4\tan^2(\phi_k)cos(\theta_{k-1})sin(\theta_{k-1})}{4L^2}\\
&+\frac{3\sigma_{\Delta d}^4\tan^2(\phi_k)cos(\theta_{k-1})sin(\theta_{k-1})}{4L^2}\\
&+\frac{2\Delta d_k^3\tan(\phi_k)cos(\theta_{k-1})^2\left(\frac{1}{2}+\frac{1}{2}e^{-2\sigma_{\bar\theta_{k-1}}^2}\right)}{L}\\
&-\frac{3\Delta d_k\sigma_{\Delta d}^2\tan(\phi_k)cos(\theta_{k-1})^2}{L}
\end{aligned}
$$

$$
\begin{aligned}
&- \frac{3\Delta d_k \sigma_{\Delta d}^2 \tan(\phi_k) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{L} \\
&- \frac{\Delta d_k^4 \tan^2(\phi_k) cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{2L^2} \\
&- \frac{3\sigma_{\Delta d}^4 \tan^2(\phi_k) cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{2L^2} \\
&+ \frac{6\Delta d_k \sigma_{\Delta d}^2 \tan(\phi_k) cos(\theta_{k-1})^2 \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right)}{L} \\
&+ \frac{3\Delta d_k^2 \sigma_{\Delta d}^2 \tan^2(\phi_k) cos(\theta_{k-1}) sin(\theta_{k-1})}{2L^2} \\
&+ \frac{\Delta d_k^4 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) sin(\theta_{k-1})}{4L^2} \\
&+ \frac{3\sigma_{\Delta d}^4 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) sin(\theta_{k-1})}{4L^2} \\
&- \frac{3\Delta d_k^2 \sigma_{\Delta d}^2 \tan^2(\phi_k) cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{L^2} \\
&- \frac{\Delta d_k^4 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{2L^2} \\
&- \frac{3\sigma_{\Delta d}^4 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{2L^2} \\
&+ \frac{3\Delta d_k^2 \sigma_{\Delta d}^2 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) sin(\theta_{k-1})}{2L^2} \\
&- \frac{3\Delta d_k^2 \sigma_{\Delta d}^2 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec(\phi_k)^4 cos(\theta_{k-1}) \left(\frac{1}{2} + \frac{1}{2} e^{-2\sigma_{\tilde{\theta}_{k-1}}^2}\right) sin(\theta_{k-1})}{L^2}
\end{aligned}
$$

$$
\begin{aligned}
E[\Delta x, \Delta\theta] = &\frac{\Delta d_k^2 \tan(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{L} \\
&- \frac{\Delta d_k^3 \tan^2(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{2L^2}
\end{aligned}
$$

$$+\frac{\sigma_{\Delta d}^2 \tan(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{L}$$
$$-\frac{3\Delta d_k \sigma_{\Delta d}^2 \tan^2(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{2L^2}$$
$$-\frac{\Delta d_k^3 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec^4(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{2L^2}$$
$$-\frac{3\Delta d_k \sigma_{\Delta d}^2 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec^4(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{2L^2}$$

$$E[\Delta y, \Delta\theta] = \frac{\Delta d_k^3 \tan^2(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{2L^2}$$
$$+\frac{\Delta d_k^2 \tan(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{L}$$
$$+\frac{\sigma_{\Delta d}^2 \tan(\phi_k) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right) sin(\theta_{k-1})}{L}$$
$$+\frac{3\Delta d_k \sigma_{\Delta d}^2 \tan^2(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{2L^2}$$
$$+\frac{\Delta d_k^3 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec^4(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{2L^2}$$
$$+\frac{3\Delta d_k \sigma_{\Delta d}^2 \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right) \sec^4(\phi_k) cos(\theta_{k-1}) \left(e^{-\frac{\sigma_{\tilde{\theta}_{k-1}}^2}{2}}\right)}{2L^2}$$

$$E[\Delta\theta, \Delta\theta] = \frac{\Delta d_k^2}{L^2} \tan^2(\phi_k) + \frac{\sec^4(\phi_k)}{L^2} \left(\Delta d_k^2 + \sigma_{\Delta d}^2\right) \left(\frac{1-e^{-2\sigma_{\phi_k}^2}}{1+e^{-2\sigma_{\phi_k}^2}}\right)$$

Annexure related to this chapter:

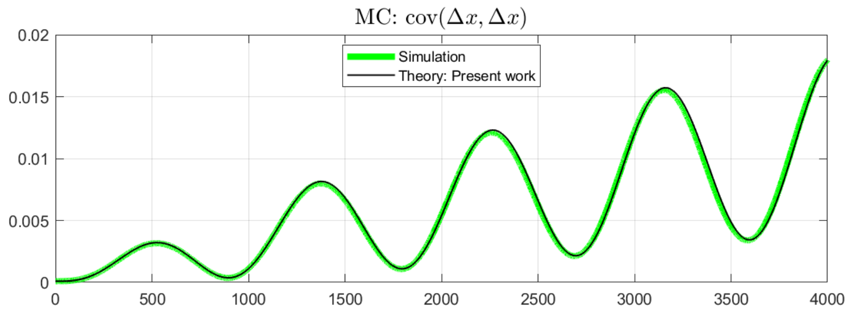


Figure 4.7: Covariance of (x,x)

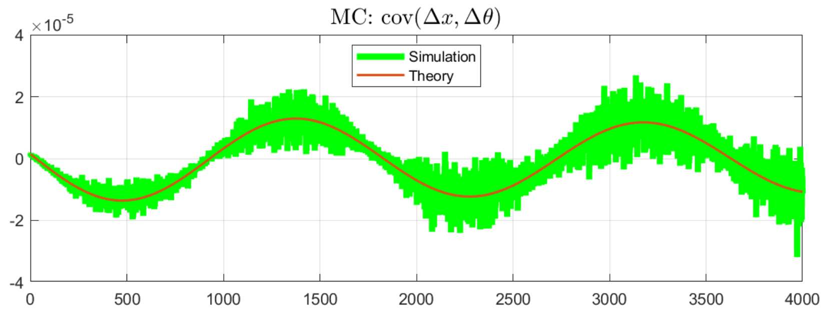


Figure 4.8: Covariance of (x,$\Delta\theta$)

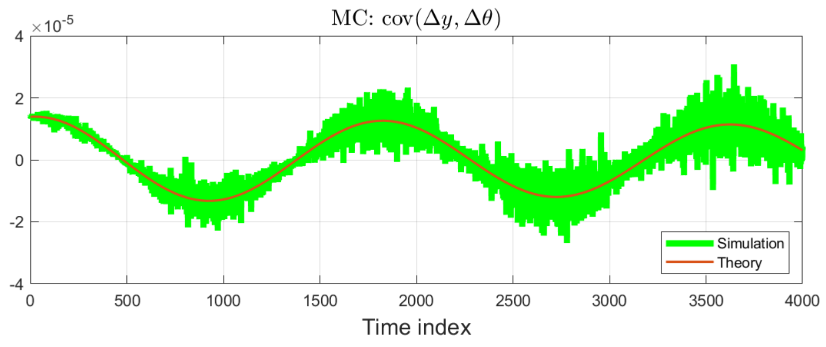


Figure 4.9: Covariance of (y,$\Delta\theta$)

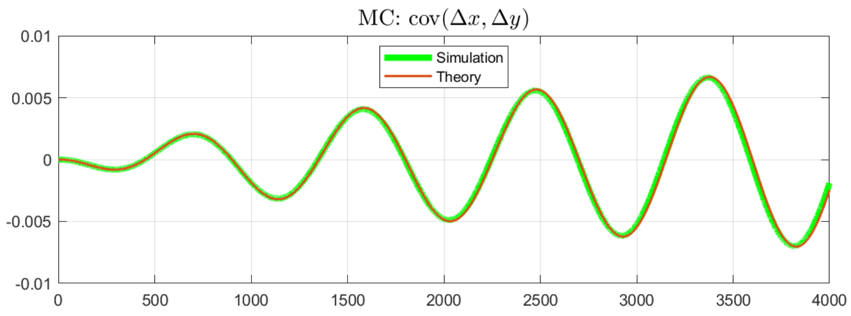


Figure 4.10: Covariance of (x,y)

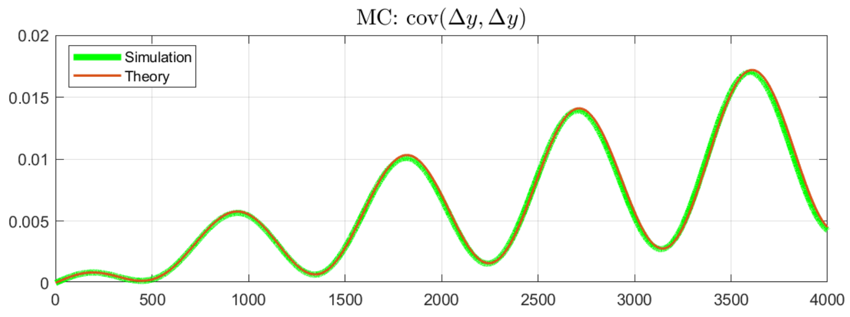


Figure 4.11: Covariance of (y,y)

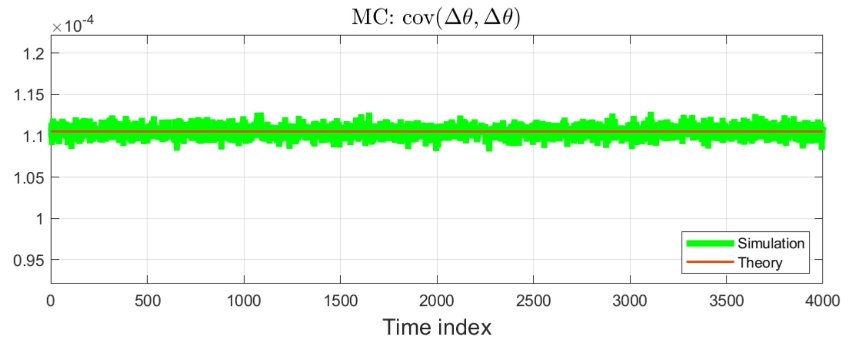


Figure 4.12: Covariance of $(\Delta\theta, \Delta\theta)$

CHAPTER 5
CONCLUSION AND FUTURE WORK

As far as the methodology is concerned, we tend to make pose estimation with kalman filter by only estimating angular and linear velocities. With the help of these 2, calculate the above given integration method equations for pose and hence, estimate the pose in particular. for simulating the pose, we are working with above given ”Exact Integration Method” , measuring the right and left wheel velocities primarily with a system that can be represented as;

The above given summary of Kalman filter was simulated using the Matlab environment. The following things taken into considerations are that the actual plant that generates the pose adds some process noise and performs the Brownian motion. Secondly, the sensor that is employed to the robot measures the displacement covered by the right and left wheel of the robot, using these we determine our linear and angular velocities of the robot that are part of our measurement model.Moreover, the ϕ is incremented with ω of measurement model. following are the results obtained; With the observed problem for the pose estimation, there are certain remedies that can come in handy. One of it is by giving robot anchor nodes during its movement, this way we can set robot on track while estimating with linear filter only. Secondly we may install sensors such as camera to detect the distance from a goal point and also the orientation required so that the robot may not get astray. Lastly, we may also use line based localization for estimating the pose of the robot. In this method we send laser beam and detect its reflection, the distance and orientation is measured with the time taken to reflect and reflecting angle of light.

Covariance analysis is a valuable tool used in mobile robotics to explore the interplay between various variables such as position, velocity, and acceleration. It helps to uncover the relationships between these factors and identify any potential correlations or causal connections. To further enhance the application of covariance analysis in mobile robotics, there are several avenues for future research.

One potential area of exploration is the incorporation of more complex sensor data. With advancements in sensor technology, there is an increasing amount of data available that can be analyzed using covariance analysis. This could provide deeper insights into the behavior of mobile robots and help to improve their performance in a variety of applications.

Another promising area for future research is the study of the impact of different environmental factors on the behavior of mobile robots. These factors can include variables such as weather conditions, terrain, and lighting conditions. By analyzing the covariance between these environmental factors and the behavior of the robot, researchers can gain a better understanding of how to optimize the performance of mobile robots in different settings.Finally, there is potential for developing more advanced control algorithms using covariance analysis. By analyzing the covariance between different variables, researchers can identify key factors that influence the behavior of mobile robots and develop control algorithms that take these factors into account. This could lead to more efficient and effective control strategies for mobile robots, which could have significant implications for a wide range of applications.

References

1 Chapman, J. W. “Moments, variances, and covariances of sines and cosines of arguments which are subject to random error.” Technometrics 12.3 (1970): 693-694.

2 Smith, Randall C., and Peter Cheeseman. “On the representation and estimation of spatial uncertainty.” The International Journal of Robotics Research 5.4 (1986): 56-68.

3 Smith, Randall C., Matthew Self, and Peter Cheeseman. “Estimating uncertain spatial relationships in robotics.” Autonomous Robot Vehicles (1990): 167-193.

4 Rodríguez-Arévalo, María L. and José Neira and José A. Castellanos. “On the importance of uncertainty representation in active SLAM.” IEEE Trans. Robotics 34.3 (2018): 829-834.

5 Knuth, Joseph, and Prabir Barooah. “Error growth in position estimation from noisy relative pose measurements.” Robotics and Autonomous Systems 61.3 (2013): 229-244.

6 Ming Wang, C. “Location estimation and uncertainty analysis for mobile robots.” Proc. 1988 IEEE International Conference on Robotics and Automation (1988): 1230-1235.

7 Ming Wang, C. “Error analysis of spatial representation and estimation of mobile robots.” Communications in Statistics: Theory and Methods 18.3 (1989): 1107-1122.

8 Ming Wang, C. “Computing uncertainty measures of location estimates for autonomous vehicles.” Journal of Statistical Computation and Simulation 36.2-3 (1990): 69-89.

9 Kleeman, Lindsay. “Odometry error covariance estimation for two wheel robot vehicles.” Technical Report MECSE-95-1, Intelligent Robotics Research Centre,

Monash University (1995): 1-28.

10 Chong, Kok S., and Lindsay Kleeman. “Accurate odometry and error modelling for a mobile robot.” Technical Report MECSE-96-6, Intelligent Robotics Research Centre, Monash University (1996): 1-20.

11 Kelly, Alonzo. “Linearized error propagation in odometry.” The International Journal of Robotics Research 23.2 (2004): 179-218.

12 Tur, Josep M. Mirats, José Luis Gordillo, and Carlos Albores Borja. “A closed-form expression for the uncertainty in odometry position estimate of an autonomous vehicle.” IEEE Trans. Robotics 21.5 (2005): 1017-1022.

13 Tur, Josep M. Mirats, Carlos Albores Borja, and Jose Luis Gordillo. “Erratum to: A closed-form expression for the uncertainty in odometry position estimate of an autonomous vehicle.” IEEE Trans. Robotics 23.6 (2007): 1302-1302.

14 Borja, Carlos Albores and Josep M. Mirats Tur, and Jose Luis Gordillo. “State your position.” IEEE Robotics and Automation Magazine 16.2 (2009): 82-90.

15 Yang, Jingdong, Jinghui Yang, and Zesu Cai. “An efficient approach to pose tracking based on odometric error modelling for mobile robots.” Robotica 33.6 (2015): 1231-1249.

16 De Carvalho Filho et alia. “The impact of parametric uncertainties on mobile robots velocities and pose estimation.” IEEE Access 7 (2019): 69070-69086.

Appendices

https://github.com/anzalshahir/FYP.git