Technische Universiteit **Eindhoven** University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Controller Design and Position Estimation of a Unicycle Type Robot

Internship report

Aniket Sharma
DC 2017.015

Supervisors:
Patrick Smulders (Segula, Eindhoven)
Prof. dr. Henk Nijmeijer (TU/e)

Eindhoven, January 2017

Abstract

This work discusses the design of a path following controller for a unicycle type mobile robot. A kinematic controller for the robot is designed using the technique of input-output linearization. The controller is tested in simulations and applied to the robot and results are plotted and tabulated showing that the designed kinematic controller is sufficient for the purpose of path following for the given unicycle type vacuum cleaning robot.
Autonomous robots and vehicles require accurate positioning and localization for their working, basically guidance, navigation, and control. In most cases two or more different sensors are used to obtain reliable data that is useful for control systems. This work also presents an algorithm for data fusion for the localization of a unicycle type mobile vacuum cleaning robot. Odometery and Ultra-wideband radio signals are used for the localization of the robot. The algorithm developed is based on the Extended Kalman Filter (EKF). The aim is to get a more accurate and enhanced measurement of the position and orientation of the robot in the office space.

Preface

This document is the work I have carried out over the three months of my internship at Segula Technologies, Eindhoven. But the work will not be complete unless I thank my mentor and supervisors for their support.
I would first like to thank Mr. Patrick Smulders, my supervisor at Segula Technologies, Eindhoven, who had entrusted me with carrying out this project and for always being supportive. I would also like to thank him for taking time to review my work and give constructive feedback.
I would also like to thank Roy A.J. Berkeveld, at Segula Technologies, Eindhoven. He has helped throughout the project with the programming and electronics of the project which were new to me. Also for reviewing my work and giving positive feedback.
Finally, my thanks to Prof. Dr. Henk Nijmeijer, my supervisor at the university for trusting me with this project and for the encouragement and confidence to explore the field of robotics during the course of my internship.

Contents

List of Figures

List of symbols

$\mathbf{V}_l$ Voltage suppl;ied to the motor on the left wheel [V]

$\mathbf{V}_r$ Voltage supplied to the motor on the right wheel [V]

x Position of the robot along the x axis [m]

y Position of the robot along the y axis [m]

θ Orientation of the robot in space [deg]

C Center of the axis joining the two wheels [-]

L Distance between the two wheels [m]

R Radius of the wheels [m]rad/s]

$\mathbf{v}_l$ Angular velocity of the right wheel [rad/s]

ω Angular velocity of the robot (heading rate) [rad/s]

v Forward velocity of the robot [m/s]

$\mathbf{x}_1$ Position of the robot along the x axis [m]

$\mathbf{x}_2$ Position of the robot along the y axis [m]

$\mathbf{x}_3$ Orientation of the robot in space [deg]

Y Output function for the controller = $h(x)$ [-]

λ Path parameter for the parameterized curve [-]

Γ Sub-manifold for the desired trajectory [-]

u Control input to the system []

n Relative degree of the system [-]

$\alpha(x)$ Desired reference trajectory [-]

$\mathbf{z}_1$ Mapping of x_1 in the new coordinates [-]

$\mathbf{z}_2$ Mapping of x_2 in the new coordinates [-]

$\mathbf{z}_3$ Mapping of x_3 in the new coordinates [-]

$\mathbf{u}_{new}$ Auxiliary control input [-]

e Error in the desired output [-]

$\mathbf{a}_1$ **and** a_0 Constants [-]

$\mathbf{v}_{left}$ Forward velocity of the left wheel [m/s]

$\mathbf{p}_{l1}$ Current reading of the left encoder [-]

$\mathbf{p}_{l0}$ Previous reading of the left encoder [-]

$\mathbf{p}_{r1}$ Current reading of the right encoder [-]

$\mathbf{p}_{r0}$ Previous reading of the right encoder [-]

$\mathbf{K}_1$ and K_2 Gains to the controller [-]

$\mathbf{x}_0$ Initial position along the x axis [m]

$\mathbf{y}_0$ Initial position along the y axis [m]

θ_0 Initial orientation on space [deg]

$\mathbf{T}$ Integration time (time for one time step) [s]

$\mathbf{v}_0^x$ Initial velocity in the x direction $[m/s]$

$\mathbf{v}_0^y$ Initial velocity in the y direction $[m/s]$

$\mathbf{v}^x$ Current linear velocity in the x direction $[m/s]$

$\mathbf{v}^y$ Current linear velocity in the y direction $[m/s]$

ω_0 Initial angular velocity $[rad/s]$

$\mathbf{a}^x$ Linear acceleration in the x direction $[m/s^2]$

$\mathbf{a}^y$ Linear acceleration in the y direction $[m/s^2]$

$\mathbf{a}^\omega$ Angular acceleration $[rad/s^2]$

$\mathbf{a}^t$ Linear acceleration in the tangential direction $[m/s^2]$

$\mathbf{a}^n$ Linear acceleration in the normal direction $[m/s^2]$

$\mathbf{v}^t$ Linear velocity in the tangential direction $[m/s]$

$\mathbf{v}^n$ Linear velocity in the normal direction $[m/s]$

$\mathbf{d}$ Time delay in the UWB system [s]

$\mathbf{x}_k$ Predicted position in discrete time along the x axis [m]

$\mathbf{y}_k$ Predicted position in discrete time along the y axis [m]

θ_k Predicted orientation in discrete time in space [deg]

$\mathbf{x}_{k-1}$ Predicted position in discrete time along the x axis at time $k-1$ [m]

$\mathbf{y}_{k-1}$ Predicted position in discrete time along the y axis at time $k-1$ [m]

θ_{k-1} Predicted orientation in discrete time in space at time $k-1$ [deg]

$\hat{x}$ State estimate of the position of the robot along the x axis [m]

$\hat{y}$ State estimate of the position of the robot along the y axis [m]

$\widehat{\theta}$ State estimate of the orientation of the robot in space [rad]

$\hat{a}^\omega$ State estimate of the angular acceleration $[rad/s^2]$

$\hat{a}^t$ State estimate of the linear acceleration in the tangential direction $[m/s^2]$

$\hat{a}^n$ State estimate of the linear acceleration in the normal direction $[m/s^2]$

$\hat{v}^t$ State estimate of the linear velocity in the tangential direction $[m/s]$

$\hat{v}^n$ State estimate of the linear velocity in the normal direction $[m/s]$

$\widehat{\omega}$ State estimate of the angular velocity of the robot (heading rate) $[rad/s]$

$\mathbf{F(t)}$ Dynamic coefficient matrix [-]

$\mathbf{u(t)}$ Input vector matrix [-]

$\mathbf{w(t)}$ Process noise [-]

$\mathbf{Q}$ Gaussian random variable with a zero mean and covariance [-]

$\mathbf{z(k)}$ Column vector of the state measurements from the sensors [-]

$\mathbf{H}$ Measurement sensitivity matrix [-]

$\mathbf{v}_k$ Measurement noise of the system [-]

$\mathbf{J}$ Gaussian random variable with a zero mean and covariance [-]

Chapter 1

Introduction

Autonomous guided vehicles are one of the main topics of discussion in many different sectors of the industry today. This work is a continuation of a master thesis which describes a low cost robotics platform which is designed for high positioning accuracy that enables it to accurately and effectively cover the given area, and also evaluate algorithms for coordinated working with multiple such robots. In simpler words the final goal of the previous work is to cover a designated area using up to 5 autonomous robots which would communicate with each other and complete the task in an effective and efficient way without having a large amount of overlap in the area covered by two different robots.
This project has two goals. The first goal being the designing of a path following controller for the unicycle type robot, which would enable the robot to follow any kind of path or trajectory that is given to the robot. The second goal is to develop a sensor fusion algorithm using an extended Kalman filter to attain the position and orientation of the robot with minimal uncertainties. The robot chosen for the project is an autonomous vacuum cleaning robot of the Neato Xv series, and is as shown in figure 1.1

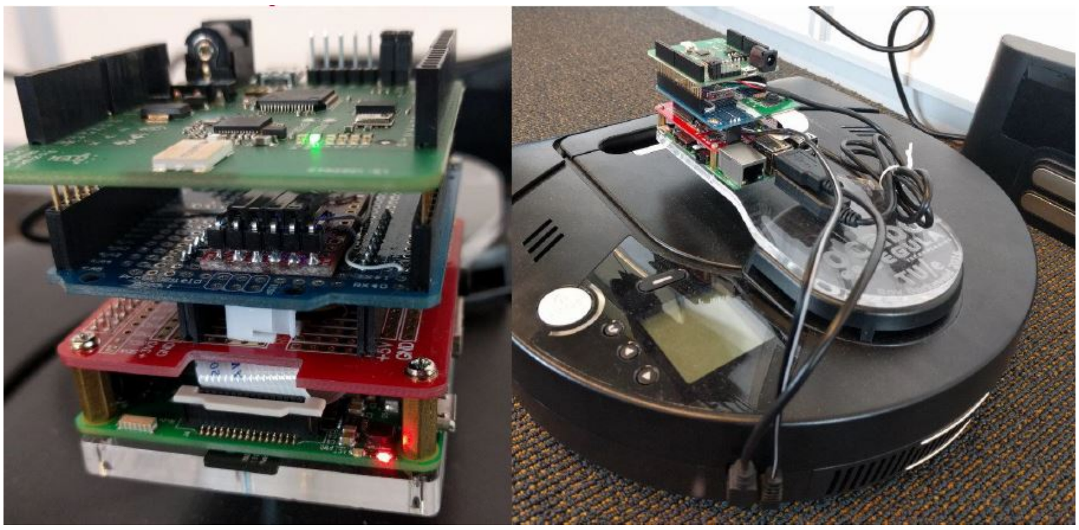

Figure 1.1: Autonomous Vacuum cleaning robot - Neato Xv series

In the market today there are many robots that can operate in an autonomous manner, but all of them work on a randomly generated path algorithm and always have large amount of overlap in the area covered, which requires more time than is necessary for the task. Thus, to reduce the time required the simplest way would be to reduce the overlap in the area covered, that is to develop an algorithm which would ensure that the robot moves in a planned and systematic manner. To achieve that basically the robot needs a control system which can tell the robot its exact location in the designated area with minimal error in the location. A control system which helps the robot to get from one point (initial position) to another point(namely the desired end position, which basically means a control system which helps the robot to follow a desired path or trajectory, i.e. a path following control algorithm. Finally a part which would decide the path that the robot needs to take in order to cover the desired area with close to zero overlap.

1.1 Scenario Description

The autonomous vacuum cleaning robot should be able to clean the entire office space in a systematic and efficient manner. The office space is set up with Ultra-wideband (UWB) beacons which are used for the purpose of localization of the robot in the given office space. Using the information from the UWB beacons the control system should be able to guide the robot to the set points, a set of coordinates that are like the immediate destination of the robot, that are provided to it by the planner (part of the system responsible for giving the desired path to the robot which uses the inbuilt Lidar on the robot to get a map of the immediate surroundings).

1.2 Report Structure

The report is structured as follows, the second chapter gives an insight into what other work has been done similar to this project, that is work related to path following controllers and sensor fusion techniques to minimize the error in the localization. The third Chapter deals with the system architecture that is used for this project and the mathematical model of the robot. The fourth chapter describe the designed path following controller and the implementation of the controller onto the robot. The fifth chapter explains the sensor fusion algorithm. The sixth and final chapter consists of the conclusion and the work that is still left for the final completion of the project.

Chapter 2

Literature Background

This chapter describes the various similar works that have been done and can be used as references to successfully achieve the goal of developing a path following controller for the unicycle type mobile robot and to perform sensor fusion for the localization. The chapter is divided into two parts the first describing about the controller and the second describing the sensor fusion algorithms.

2.1 Controller design

There are many different types of controllers that can be used to achieve the task of path following or trajectory tracking. Some of the methods are discussed in this chapter and taken as references to design the controller used for the robot. [7] is the basic source that is required to complete the design of the controller for the unicycle type robot. It gives a good insight into the various techniques that can be used and the basics of nonlinear control design.
In [2] a backstepping controller is used for the trajectory tracking of a unicycle type mobile robot. It uses the Lyapunov theorem, and a kinematic tracking controller is used to provide the linear and angular velocities for the given trajectory. Here both the linear and the angular velocities are used as control inputs to the system and the kinematic and dynamic behavior of the robot is studied. Similarly, in [9] the method of backstepping is used in addition with the LaSalle's invariance principle.
Another way to achieve path following and autonomous navigation is to combine multiple controllers as shown in [1], where the navigation architecture combines go-to-goal, follow the wall and avoid obstacle into a full navigation system by using algorithms for moving to a point, moving to a pose, moving on an arc and avoiding obstacles. This paper also hints to the fact that a kinematic controller can be used for the unicycle type robots. For our project, we have a separate planner to give the path which already incorporates collision avoidance.
A time-invariant, discontinuous control law which utilizes the technique of feedback linearization can also be used as shown in [19]. This paper also shows an approach that is based on the invariant manifold theory for the purpose of the path following controller. The kinematic controller developed here is then extended to a dynamic controller. The kinematic controller showed good results for path following and the performance improved with increasing the velocity feedback gain.
In [3] tracking and path following strategies are based on inner and outer loop structure and are formulated over the kinematic and dynamic model of the unicycle type mobile robot. They also compare the difference between two different types of controllers, namely controller A which is a nonlinear controller for tracking and path following and controller B which represents a unified controller as is proposed in [18]. In this paper the linear velocity is set to a non-zero constant value and thus the system has only one input variable that needs to be controlled that is the angular velocity to the robot. The error dynamics is then used to form the path following controller where

the error is the positioning of the robot in space from the desired path. Thus, if the error is made to converge to zero the robot will be moving on the desired path.

2.2 Sensor Fusion

Autonomous vehicles are heavily dependent on the sensors that are installed to have a good sense of their surroundings. But the sensors have uncertainties and errors in determining the states they are measuring, such as the position, velocity, acceleration or the orientation. Thus, it is wiser to observe the same parameter through different sensors to better estimate the states of the object that may be used later by the controller. Sensor Fusion is basically using multiple sensors to read a single parameter and to estimate the parameter to a more accurate value than with just one sensor. And sensor fusion not only increases the accuracy but also increases the robustness against sensor failures as multiple sensors observe a single object. This research is limited to sensor fusion to minimize localization error, that is to get the position of the observed object with minimal error.
Data fusion can be done by the use of an Extended Kalman Filter (EKF) and adaptive fuzzy logic to make the laws for the algorithm for sensor fusion as shown in [16]. Here the paper shows the use of sensor fusion for mobile robot navigation using the odometry and sonar signals. The adaptive fuzzy logic system is used in the paper for the measurements from sensors which cannot be considered as white noise, because colored signals cause the EKF to diverge which fails the point of sensor fusion. The observations made in this paper are that the output signal from the sensor fusion block is more accurate than the original signals separately. It was concluded that using this the real-time operation of the vehicle/robot can be reduced.
Indoor localization using Wi-Fi received signal strength and pedestrian dead reckoning are used together with a sensor fusion algorithm in [4]. Similar to [16] this paper uses an EKF for the purpose of sensor fusion which is a light computation approach to sensor fusion and desirable for real time computations. The method implied uses developing a measurement model which enables the accurate Wi-Fi localization and then combining it with the one obtained by the method of dead reckoning. The paper concluded that the approach of using an EKF for localization achieves improved accuracy at the same time has a lower computational complexity.

Chapter 3

System Architecture

The initial step is to develop an architecture scheme for the system using the sensors available. The main aim of the robot is to cover the entire office space in a systematic and planned manner. The sensors available to the robot are a Lidar, the UWB system and the internal measurement system in the robot which consists of the encoders, accelerometers, and a gyroscope. To achieve the goal, he system should be able to get its accurate location in the office space and also determine the path that it needs to take so as to achieve rea coverage along with avoiding all the obstacles in the office space. The sensors mentioned are enough to realize this goal.

For the purpose of area coverage, obtaining the accurate position of the robot in space is of utmost importance. Using the fore mentioned sensors there are two separate ways to obtain the position of the robot in space. The first method is the UWB system which comprises of the pozyx board on the robot and the six range measurement beacons that are placed in the office space. Here one of the beacons is considered as the origin and the system gives the absolute position of the robot in space. The second method is to use the odometery to get the relative position of the robot. This can be achieved by the method known as dead reckoning which is very commonly used by sailors to calculate their positon at sea. It is a method which gives the position of the robot with respect to the previous position. Thus, basically the encoders and accelerometer can be used to determine the distance traveled by the robot and the gyroscope gives the direction and the position of the robot is determined from the initial position. The sensors specially the UWB beacons have uncertainties in their measurements and thus cannot always be reliable due to errors in measurements or reflections of metallic surfaces. To overcome this both the methods for localization can be used and sensor fusion can be performed using an extended Kalman filter. In this manner, the uncertainties in the measurements can be reduced and an accurate measurement for the position of the robot is obtained. Once the position is obtained the next step would be to obtain information about the surroundings so as to plan a path which would avoid the obstacles and help achieve the goal. The Lidar is used to generate a map of the surroundings.

The system architecture that was designed for the system, keeping the above mentioned points in mind, is as

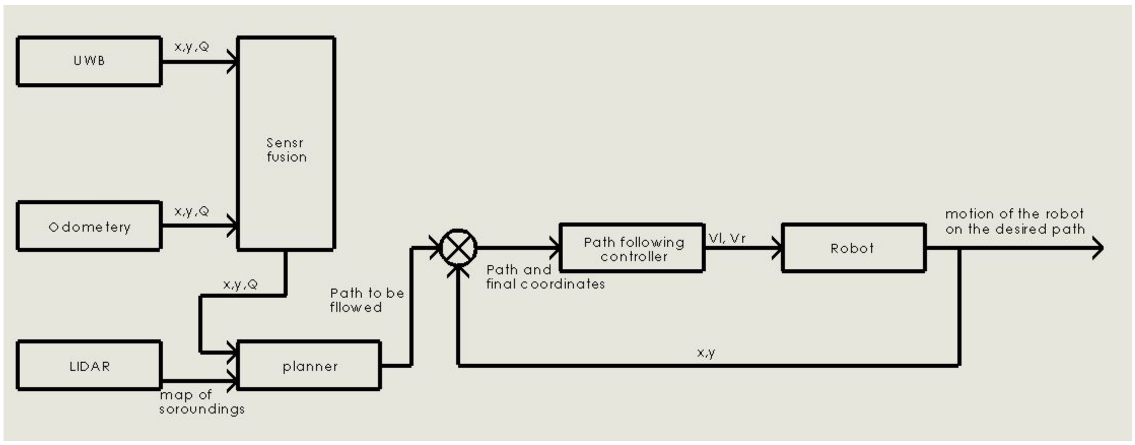

Figure 3.1: System architecture

The value for the position that is obtained from the odometery and the UWB beacons are sent as inputs to the extended Kalman filter. The input to the Kalman filter is of the form (x, y, θ) which is the positon of the robot in the x and y direction and the orientation of the robot in space(where it is facing). The filter then performs sensor fusion and gives a more accurate measurement of the position and the orientation of the robot in the office space. As it can also be observed form the figure (3.1) the output of the Kalman filter along with the output of the Lidar is sent as the input to the planner. The planner uses this information to determine a path for the robot in the form of set points (i.e. the x and y coordinates of the destination). The set points can be joined together to give the desired path to be followed by the robot. These set points are then given as the desired path to the controller which then calculates the speed for each of the two wheels and accordingly gives a voltage input (Vl an Vr as shown in the fiure) to the motors such that the desired set points are reached and the robot follows the path developed by the planner.
The output of the Kalman filter was chosen to be sent as the input to the planner instead of the controller, to simplify the work of the controller. This is because the pose of the robot is provided in a coordinate system with the origin as one of the beacons which is decided when the hardware is setup for use. But the Lidar uses the robot as the origin and thus gives the map of the surroundings in a completely different coordinate system than the pose of the robot. Thus, if the planner gives a path just based on the Lidar information then the controller will need to do the coordinate transformation from the set used for the path to that of the position of the robot, for every set point that makes the path. But if the planner gets the output of the Kalman filter as its input as well as the Lidar output as its second input. Then the planner does the coordinate

Chapter 4

Controller Design

The neato XV vacuum cleaning setup used for the presented study is as shown in figure (4.1). The mobile robot consists of a rigid body and a set of non-deforming wheels. The robot is designed in such a way by the manufacturers that none of the wheels loses traction. If the wheels were to lose traction, then the robot might deviate from the path because of slipping of the wheels. This is overcome by the manufacturer of the robot by giving both the wheels freedom in the z direction, i.e. the wheels can move along the vertical axis. This basically acts as a simple suspension system for the robot and thus helps to retain the contact of the wheels to the ground even when moving on irregular surfaces. The design of the robot is such, that if any of the wheels loses contact to the ground the robot returns an error and stops moving. The robot can move with a maximum speed of $0.3m/s$, which is due to the design constraints. The design of the robot and the low maximum speed of the robot can help us assume that the chances of slip will be minimal.

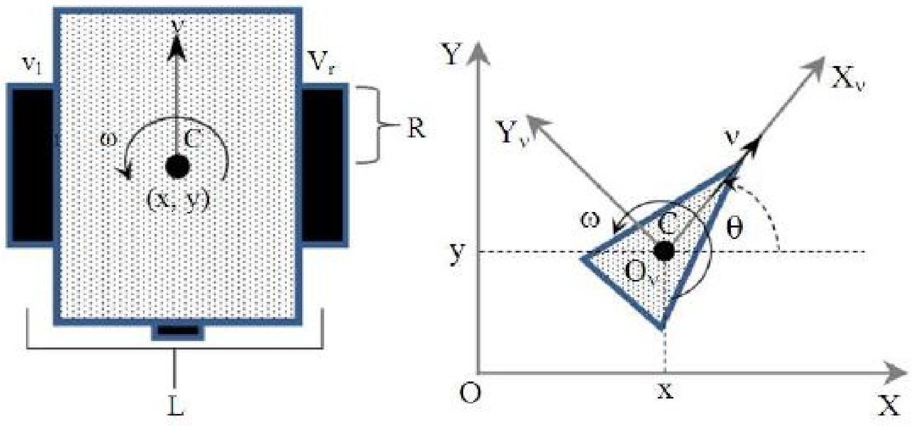

Figure 4.1: Neato XV model considered for the project

The robot is represented by the generalized coordinates $q = (x, y, \theta)$, where (x, y) is the position of the robot in space and θ is the orientation of the robot (heading), of the center of the line joining the two wheels, the point depicted as C in 4.1, with respect to a global inertial frame O, X, Y. The frame of the robot is taken as O_v, X_v, Y_v the vehicles velocity is defined as v in the vehicles direction (X_v) and L is the distance between the two wheels and R is the radius of the wheels. v_r and v_l represent the angular velocities of the right and left wheels respectively and ω is the heading rate. The kinematic model can be represented as

$$\dot{x} = \frac{R}{2}(v_r + v_l)cos\theta \tag{4.1}$$

$$\dot{y} = \frac{R}{2}(v_r + v_l)sin\theta \tag{4.2}$$

$$\dot{\theta} = \frac{R}{L}(v_r - v_l) \tag{4.3}$$

The model is based on the angular velocities of the left and right wheels and the heading rate represented by v_l, v_r and ω respectively. Here the system has three inputs, which can be reduced to two inputs by changing the angular velocities of each of the wheels to the translational velocity of the robot, using the relation $v = \omega r$ where the translational velocity is the product of the angular velocity and the radius of the wheel. Thus, the relation between v_r, v_l, v and ω is given as

$$v_r = \frac{v + \frac{\omega L}{2}}{R} \tag{4.4}$$

$$v_l = \frac{v - \frac{\omega L}{2}}{R} \tag{4.5}$$

The above relations are substituted into the equations (4.1), (4.2), (4.3) and the set of equations found are equivalent to the mathematical model for the unicycle. Thus, the unicycle model is used for the robot. The model is given as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.6}$$

The basic task of the controller is the position estimation of the robot and to make sure that the robot is at the desired position. The robot has an internal controller present which monitors the force and torque provided to the wheels (voltage to the actuators). From the above two statements a kinematic model was deemed sufficient to solve the problem. The robot design and the low maximum speed of the robot helped to solidify the decision of going ahead without the need for a dynamic model. Another factor influencing the decision is that the kinematic controller is generally simpler and can be later worked on towards the dynamic model, by using the methods as shown in [7]. Thus, only a kinematic model is considered for this project.

The kinematic model (4.6) is considered, where $x \in R^3$ the input $v \in R$ is the translational speed and $\omega \in R$ is the angular velocity of the steering angle. The position of the robot in space is taken as the output of (4.6) and is given as

$$Y = h(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{4.7}$$

As an example the desired path $\alpha(x)$ is chosen as a unit circle and is given as a regular parameterized curve by a path parameter Λ

$$\Lambda \mapsto \begin{bmatrix} Cos\Lambda \\ Sin\Lambda \end{bmatrix} \tag{4.8}$$

There exists a smooth map $s : R^2 \rightarrow R^1$ where 0 is a regular value of s and the manifold $\sigma = s^{-1}(0)$ thus the submanifold for the desired trajectory can be defined as $\Gamma := (soh)^{-1} = \{x \in R^3 : s(h(x)) = 0\}$. The function $\alpha(x)$ can be defined as

$$\alpha(x) = soh(x) = x_1^2 + x_2^2 - 1 \tag{4.9}$$

In [12] the path following problem is solved by fixing the translational speed $v \neq 0$. This is done so that the system becomes a single input system and is easy to tackle. Note that the only input to the robot is the angular velocity (rate of change of the steering angle x_3). Thus the input $u = \omega$. Thus the system transforms to

$$\dot{x} = \begin{bmatrix} vcosx_3 \\ vsinx_3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{4.10}$$

The above system is of the form $\dot{x} = f(x) + g(x)u$ where

$$f(x) = \begin{bmatrix} vcosx_3 \\ vsinx_3 \\ 0 \end{bmatrix}; g(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.11}$$

As the system is of the form $\dot{x} = f(x)+g(x)u$ the method of input-output feedback linearization can be performed to control the nonlinear system, i.e. the unicycle type mobile robot. The main aim of the input-output feedback linearization is to give a transformation whose states are the output $y = h(x)$, and the first $(n-1)$ derivatives of the output function where n is the relative degree of the system. To do this the Lie derivatives are used [7]. Differentiating the reference trajectory function $\alpha(x)$ with respect to time

$$\begin{aligned} \dot{\alpha} &= \frac{\delta\alpha(x)}{\delta x}\dot{x} \\ &= \frac{\delta\alpha(x)}{\delta x}f(x) + \frac{\delta\alpha(x)}{\delta x}g(x)u \\ &= L_f\alpha(x) + L_g\alpha(x)u, \end{aligned} \tag{4.12}$$

Computing the equation (4.12) the following values are obtained

$$L_f\alpha(x) = 2v(x_1cosx_3 + x_2sinx_3), L_g\alpha(x) = 0. \tag{4.13}$$

Now taking the derivative of the function $\dot{\alpha}(x)$ with respect to time

$$\begin{aligned} \ddot{\alpha} &= \frac{\delta\dot{\alpha}(x)}{\delta x}\dot{x} \\ &= \frac{\delta L_f\alpha(x)}{\delta x}f(x) + \frac{\delta L_f\alpha(x)}{\delta x}g(x)u \\ &= L_f^2\alpha(x) + L_gL_f\alpha(x)u \end{aligned} \tag{4.14}$$

Computing the equation (4.14) the following values are obtained

$$L_f^2\alpha(x) = 2v^2, L_gL_f\alpha(x) = 2v(x_2cosx_3 - x_1sinx_3). \tag{4.15}$$

The next step is to understand how the input u enters the system, which can be done by finding the relative degree of the system. To find the relative degree of the system using the Lie derivatives, count the number of times the output function was differentiated till the input u appears explicitly in the result. Form equation 4.15 it can be said that the relative degree of the system is 2 because the input to the system appears in the second derivative of the function $\alpha(x)$ [7] . By the assumption that $v \neq 0$, the only possibility that the function $L_gL_f\alpha(x)$ would be equal to zero is if and only if

$$tan(x_3) = \frac{x_2}{x_1} \tag{4.16}$$

This condition means that the unicycle is oriented 1normal to the desired path. Suppose by way of contradiction, that the function $L_gL_f\alpha(x)$ on Γ. Then $\alpha(x) = 0$, the following holds

$$\begin{bmatrix} cos(x_3) & sin(x_3) \\ -sin(x_3) & cos(x_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0 \tag{4.17}$$

The above 2×2 matrix in equation (4.17) is nonsingular, thus the only solution to the equation (4.17) is that $x_1 = x_2 = 0$. This is not possible since the robot needs to be on the unit circle $x_1^2 + x_2^2 - 1$ on γ. This shows that the function $\alpha(x)$ yields a well-defined relative degree of 2 at each point for the path following manifold denoted by the set Γ. The function α can be used to stabilize the circle. The output function (4.7) of the system (4.11) can be made to converge to the given reference trajectory by simply making $\alpha(x)$ and $\dot{\alpha}(x)$ converge to zero. Thus, the path following problem transforms into an output zeroing problem, which is a well-known problem in control [17]. Now a coordinate transform can be partially performed using the functions $\alpha(x)$ and $\dot{\alpha}(x)$ as the first two functions of the transformation because the relative degree was found to be two. To complete the coordinate transformation, a third function is assumed. As shown in [13] and [6] the choice for the third function for circular paths is as

$$\pi(x) := \tan^{-1} \frac{x_2}{x_1} \tag{4.18}$$

The transformation then can be written as

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = T(x) = \begin{bmatrix} \alpha(x) \\ \dot{\alpha}(x) \\ \pi(x) \end{bmatrix} \tag{4.19}$$

This basically transforms the trajectories from the original x coordinate system into the newly defined z coordinate system. If the transformation is a diffeomorphism, then the smooth trajectories in the original coordinate system will have smooth unique counterparts in the newly defined z coordinate system. To show that a local diffeomorphism is determined by these functions ($\alpha(x)$, $\dot{\alpha}(x)$ and $\pi(x)$), the inverse function theorem is used. Proof of the diffeomorphism is done in the appendix. Using the coordinate transformation T, in the neighborhood of any point $x^* \in \Gamma$, the system transforms to

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} = \begin{bmatrix} z_2 \\ L_f^2\alpha + L_g L_f \alpha u|_{x=T^{-1}(z)} \\ f^0(z) \end{bmatrix} \tag{4.20}$$

When $z_1 = z_2 = 0$, that just implies that $\alpha(x) = \dot{\alpha}(x) = 0$, that means that the system is restricted to evolve on the path following manifold(Γ). Stabilizing the states z_1 and z_2 is equivalent to get the robot onto the desired path with the heading velocity as a tangent to the path to be followed. When the robot is on the path following manifold (i.e. $z_1 = z_2 = 0$) then z_3 determines the position of the robot on the path. Under this condition the dynamics of z_3 are basically the zero dynamics of the system which are restricted to the path following manifold and are given as

$$\dot{z}_3|_\Gamma = f^0(0,0,z_3) = -\frac{v(x_2 cos x_3 - x_1 sin x_3)}{x_1^2 + x_2^2}|x = T^{-1}(0,0,z_3) \tag{4.21}$$

From [7] the regular feedback transformation is given as

$$u = \frac{1}{L_g L_f \alpha}(-L_f^2\alpha + u_{new}) \tag{4.22}$$

Where u_{new} is the auxiliary control input. The controller is well defined in the vicinity of $x^* \in \Gamma$, this is due to the fact that $L_g L_f \alpha(x) \neq 0$. Around x^* the closed loop system is as

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} = \begin{bmatrix} z_2 \\ u_{new} \\ f^0(z_1, z_2, z_3) \end{bmatrix} \tag{4.23}$$

Note that on the path following manifold Γ

1. $x_1^2 + x_2^2 = 1$
2. $x_1 cos(x_3) = -x_2 sin(x_3)$

Solving the above two mentioned relations for the values of x_1 and x_2 the following two solutions are obtained

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -sin(x_3) \\ cos(x_3) \end{bmatrix} \tag{4.24}$$

or

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} sin(x_3) \\ -cos(x_3) \end{bmatrix} \tag{4.25}$$

Using these solutions (4.24) and (4.25) into the expression for $\dot{z}_3$ in the equation (4.21) the following is obtained

$$\dot{z}_3 = \pm v \tag{4.26}$$

In both the cases, the dynamics restricted to the desired trajectory are unstable. In this case, it is desirable because it means that the unicycle will traverse the entire path (the unit circle). Since the translational speed v is constant, the motion of the robot on the desired trajectory cannot be affected, the plus and minus signs correspond to the clockwise or anticlockwise path following respectively.

The control input u_{new} is referred to as the transversal input as it is used to control the transversal subsystem. The transversal subsystem is LTI and controllable so there are many techniques that can be used to stabilize z_1 and z_2. The one chosen is the input-output linearization (tracking) as shown in [7]. Let us suppose the output required is 'y' converges to some constant reference value 'y_d'. Then the new input u_{new} can be defined as

$$u_{new} = -a_{n-1} y^{(n-1)} - ...... - a_1 \dot{y} - a_0 (y - y_d) \tag{4.27}$$

Where n is the relative degree of the system thus the control input for the given case can be written as

$$u_{new} = -a_1 \dot{y} - a_0 (y - y_d) \tag{4.28}$$

It is known that the error in the desired output is given by $e = y - y_d$, thus the error dynamics for a system with a relative degree two is given as

$$\ddot{e} + a_1 \dot{e} + a_0 e = 0 \tag{4.29}$$

The characteristic equation can be written down for the system as

$$\lambda^2 + a_1 \lambda + a_0 = 0 \tag{4.30}$$

It is shown in [7] that $e(t) \to 0$ and $t \to \infty$ if the poles of the character equation 4.30 are in the open left half plane. The values of the constants a_1 and a_0 are calculated using pole placement and the values found are found to be as $a_1 = 0.6$ and $a_0 = 0.1$. Thus the control input $u_n ew$ transforms as $u_n ew = 0.6(\dot{x}_3 - \dot{x}_{3d}) - 0.1(x_3 - x_{3d})$ Using the control input, the input to the system is updated (4.22). The input is as

$$u = \frac{2v^2 + u_{new}}{2v(x_2 cos x_3 - x_1 sin x_3)} = \frac{2v^2 - 0.6(\dot{x}_3 - \dot{x}_{3d}) - 0.1(x_3 - x_{3d})}{2v(x_2 cos x_3 - x_1 sin x_3)} \tag{4.31}$$

The controller described above was designed on matlab and simulated for different starting points of the robot that were located away from the unit circle. It was observed that the controller is able to follow the desired path irrespective of the starting postion, i.e. the controller is able to guide the robot to go towards the unit circle and follow the unit circle. This is shown in figure (4.2) It can be infered that the controller is able to make the robot follow the desired path.

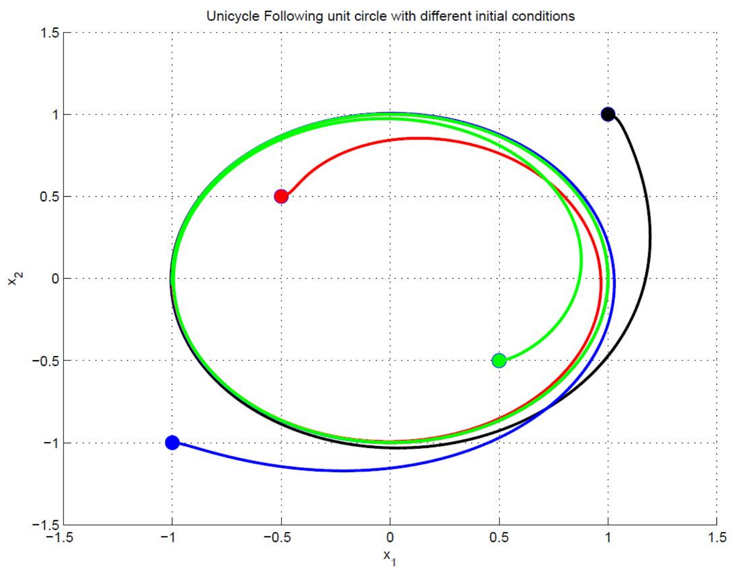

Figure 4.2: Simulation results following a unit circle with different starting positions

The example of the reference trajectory that is discussed in this chapter is that of a circle. The controller can be applied to other paths both closed and non-closed ones. For example a straight linewhich is shown in figure 4.3

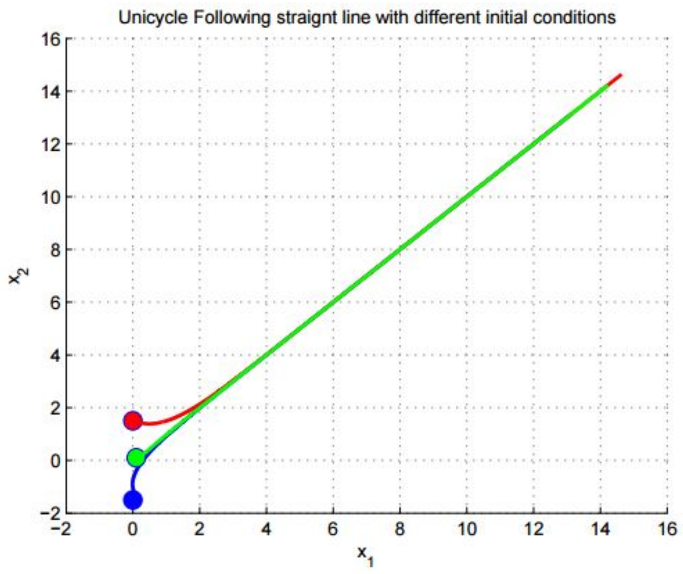

Figure 4.3: Simulation results following a straight line with different starting positions

Thus it can be concluded that the controller can be applied to various different trajectories just by changing the function α and following the steps that are defined above.

Chapter 5

Implementation and results

Implementing the controller on matlab and implementing it onto the hardware are two completely different things. On the software one doesnt consider all the functionalities of the robot and only concentrates on the part to be designed, which might lead to problems when implementing it onto the hardware. This chapter lists some of the problems that were faced when performing the task of path following onto the actual robot and what was done to solve these problems and the observed results.
Before the path-following controller was implemented, the robot was first checked if it is able to move in accordance with the given velocity inputs. This was done by giving velocity inputs to the wheels and observing the orientation of the robot through the heading sensor and the amount of distance moved by each of the wheels using the encoder readouts. It was observed that the robot had a slight drift to the left side, that means the robot has a tendency to deviate slightly towards the left than its intended direction. This can be observed as shown in figure (5.1)

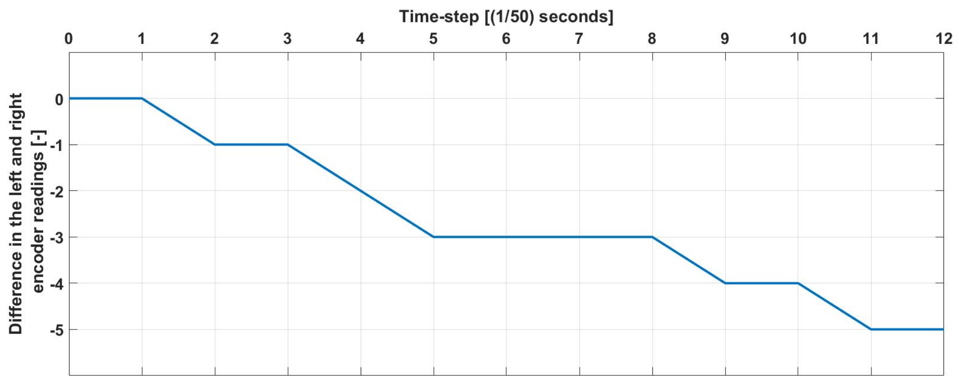

Figure 5.1: Difference in the left and right encoder values when the robot is to move in a straight line

The figure depicts the difference between the values obtained from the encoders on the left and right wheel when both the wheels were given the same velocity. The x axis depicts the time step (The encoders run at a frequency of $50Hz$, thus the time step is $0.02s$) and the y axis is the difference between the two encoder readouts which is calculated as $encoder_{left} - encoder_{right}$. In a situation where the wheels are given the same velocity the difference between the values obtained from the left and the right encoders should be 0 as both the wheels need to move the same distance to make the robot move in a straight line. But it is observed that the difference in the values of the left and the right encoders grows more negative over time. That means that

the right wheel of the robot is travelling more distance than the left wheel, which means that the robot is gradually moving left. It can also be interpreted as that the robot is deviating towards the left from its desired trajectory. This is considered to be a problem for the system because even with the path following controller the robot will have a drift which would accumulate over time and lead to the performance of the path following controller being affected, and the robot deviating away from the desired end coordinates. There can be several reasons for this problem to occur. The ones that were thought of are the wheels having a slight difference in their diameters, which might cause them to move differently, but in this case the change in the difference between the encoder readouts will be a smooth curve and not plot with steps where the difference does not change. It is not possible in practice to make two motors to rotate exactly at the same speed even when they are given the same voltage as input for a long period time. This leads to the second reason, why there might be a drift as it is possible that even though the input to the two motors is the same, the output may be a little different leading the robot to have different velocities at both the wheels. There is an inbuilt controller in the robot provided by the manufacturers which sends the voltage and power to the robots depending on the velocity set points that are given as the input. It was not determined with absolute certainty that this was the cause of the drift but it can be a possibility.
To solve this problem a simple PD controller was designed, where the difference in the encoder readouts was taken as the error and the rate of change of this error was also considered. The controller took the form as

$$v_{left} = v - K_1(p_{l1} - p_{r1}) - K_2((p_{l0} - p_{r0}) - (p_{l1} - p_{r1})) \tag{5.1}$$

In the above equation v_{left} is the velocity of the left wheel, v is the constant forward velocity that is set for the robot, p_{l0} and p_{r0} are the encoder readouts of the previous time step,p_{l1} and p_{r1} are the encoder readouts of the current time step and K_1 and K_2 are the gains. A similar equation is set up for the velocity of the right wheel and then implemented onto the robot. The result of implementing this controller can be seen in figures (6.1) and (5.3)

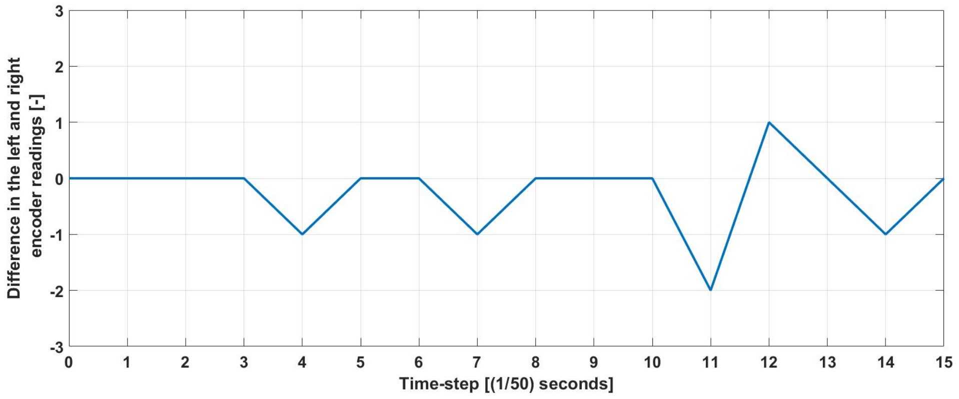

Figure 5.2: difference in the encoder readouts after the implementation of the pd controller

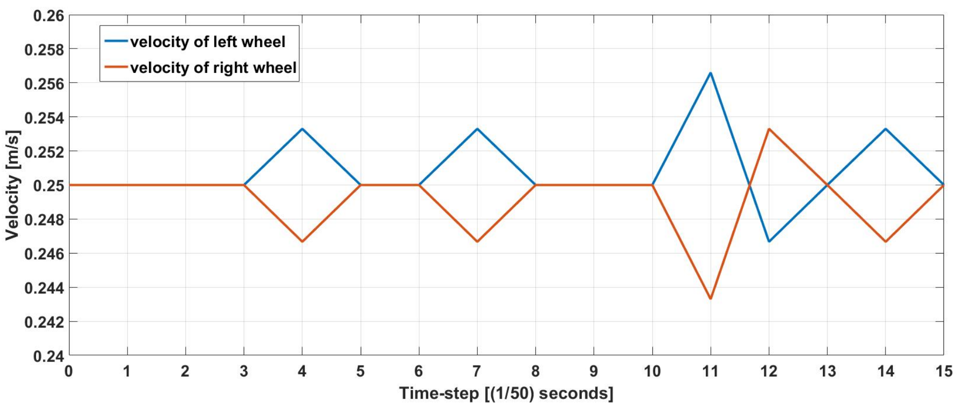

Figure 5.3: speeds of the wheels given by the controller depending on the difference in encoder readouts

In figure (6.1) it is observed that the difference in the encoder readouts is always returning to zero. This is because the controller is tuning the velocities on each wheel, as shown in figure (5.3) so that the difference always returns to zero, such that the drift to the left that was observed earlier is removed. In figure (6.1) it is observed that the difference goes to a positive value which was not observed when the controller was not used. The encoders are stet to work at a frequency of 50 HZ and thus there is a small overshoot in the correction of the difference in the encoder values. This can be removed by increasing the operating frequency or changing the gains to the controller.

The controller can thus be divided into three parts that is the one that is already present in the robot, then the simple PD controller which was implemented to remove the drift and finally the path following controller which was designed in the previous chapter. Thus the controller architecture can be represented as shown in figure (5.4)

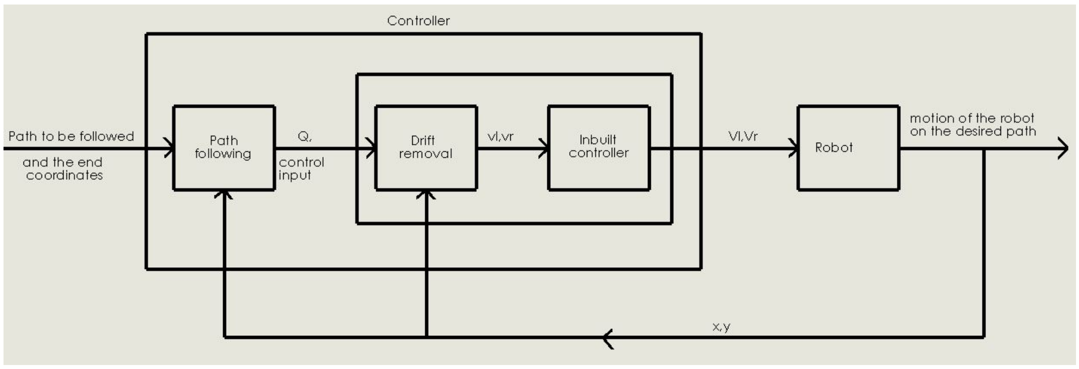

Figure 5.4: Control loop architecture

Here the input to the path following controller is the path to be followed which Is given by the planner. The path following controller processes that data and gives ahead the orientation θ and the control input u_new to the above designed PD controller which takes care of the drift, that was observed in the first half of this chapter. Then the drift removal part uses the information to compute the velocities for each of the wheels and then gives those velocities, vl and vr for the left and right wheel, as inputs to the inbuilt controller of the robot, which translates the velocities into voltages Vl for the left actuator and Vr for the right actuator for each wheel and actuates the actuators. The position (coordinates) of the robot in space, denoted by x and y is given back

to the system for the next time step.
As described in the previous chapter the path following controller makes sure that the robot stays on the desired path and reaches the final destination. In the simulations the refence trajectory chosen was that of a unit circle, but in practice the robot will be following straight lines and arcs. The unit circle was chosen for the purpose of simulations because it is easier to show the result on that trajectory. For the practical application the trajectory of a line was chosen and the accordingly the changes were made in the design to incorporate the new trajectory. The necessary changes were made in the design, the main changes were the Lie derivatives as they are the derivatives of the reference trajectory, the input u also changes as it depends on the Lie derivatives and finally adjusting the gains for the control input u_{new}. Once this was done the feedback linearization path-following controller was implemented onto the robot. But the robot can move in straight lines and arcs, the controller that needs to be implemented needs to be a switching controller which should be able to switch between the two trajectories depending on the path that is provided to it by the planner. That would complicate the controllers process, Thus a simpler solution would be to design the planner in a way such that whenever the path would have an arc the planner chooses the set points closer than normal which would result in the path from one set point to the other to be a straight line. It is depicted in figure (5.5)

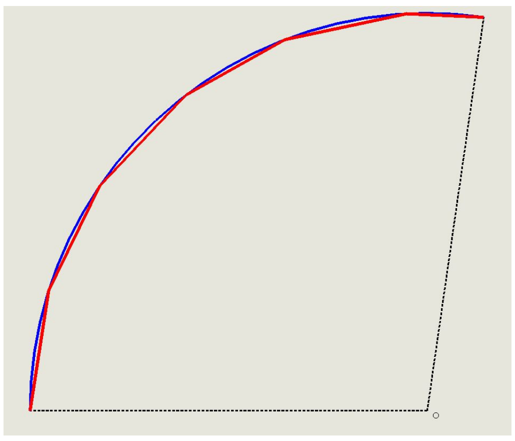
Figure 5.5: arc made up of multiple small lines

It can be inferred from the figure that the arc can be made from multiple straight lines. This is also possible for the robot because it has a wide vacuum cleaning mouth at the front, and effectively it will cover the same amount of area if it goes on the arc in the figure (5.5) which is shown as the blue curve. Or if it goes on the red colored path, which is made up of multiple lines. Thus for this project it was chosen that the planner will be designed in a manner such that all the paths obtained by the controller as input are straight lines. For this project the planner is not available as it has not been designed yet, thus the path given to the robot for testing is predefined and is not updated continuously by the system. Thus the verification of the designed controller is done on a pre-defined path. The UWB system for the localization was also not functional due to system upgrades and other reasons. Thus the method used to verify the controller is not the UWB system which would confirm the position of the robot as desired, but to find the angle between the current position of the robot and the desired position, and then using the current orientation and finding the difference between the two angles and using that to determine if the robot is moving in the desired direction. But this method is not 100 percent accurateaccurate as the position is not verified, thus the system needs to be checked once the UWB system and the planner are both online and functional.

Chapter 6

Sensor fusion

Sensor fusion for this project is used to get a more accurate readout of the pose of the robot, in simpler words it means to minimize the error in the localization of the robot. The basic idea is to get the value of the pose (basically the position and the orientation) of the robot using the wireless ultra-wideband (UWB)system, which is basically the indoor positioning system, and the inertial navigation system (basically calculated using odometery). The two set of inputs for the pose are then to be passed through a sensor fusion algorithm to get a more accurate readout of the localization of the robot in the office space. Due to the lack of time and some hardware issues the sensor fusion part could not be implemented and tested on the robot. But in this report a method to apply the Kalman filter to achieve sensor fusion on the two methods for localization is discussed.

The robot is equipped with a pozyx board which has a UWB tag and is responsible for communicating with the 6 UWB beacons placed in the office space (the fixed world). The beacons are basically range measurement devices to the UWB tag. Using all the six range measurements the system is able to compute a position of the robot with respect to the fixed origin, which in this case is one of the beacons. This is a low frequency module (10 Hz), that is we approximately get a reading of the pose about 2 times every second, because the tag sends a signal to the beacons and computes the distance to one beacon. The pozyx board is also equipped with a heading sensor which gives the orientation of the robot. Whereas the accelerometer and the gyroscope work at a much higher frequency of $1kHz$. As it is observed that the two methods are not working at the same frequency and thus a simple sensor fusion algorithm cannot work. A multi rate extended Kalman filter can be used in this case for the purpose of sensor fusion. The basic idea is to use a delay compensator and predict the values of the slower rate sensor and use the predictions in the sensor fusion algorithm. The framework of the working of the sensor fusion part can be expressed as shown in figure (6.1)

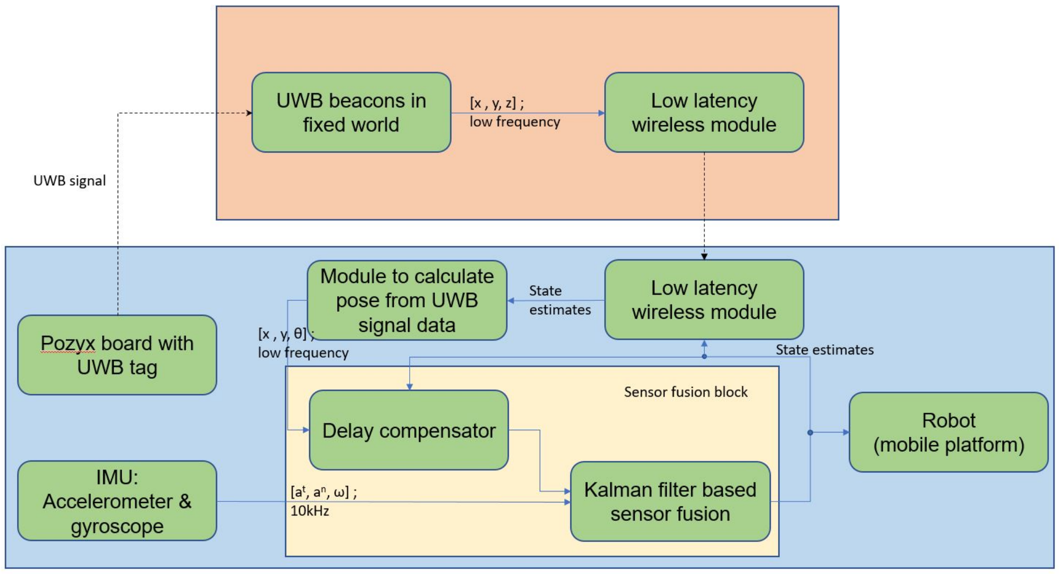
UWB beacons in fixed world
[x , y, z] ; low frequency
Low latency wireless module
UWB signal
Module to calculate pose from UWB signal data
State estimates
Low latency wireless module
Pozyx board with UWB tag
[x , y, θ] ; low frequency
State estimates
Sensor fusion block
Delay compensator
Robot (mobile platform)
IMU: Accelerometer & gyroscope
[aᵗ, aⁿ, ω] ; 10kHz
Kalman filter based sensor fusion

Figure 6.1: Block diagram of the sensor fusion framework

The robot is equipped with a UWB tag which gives the coordinates of the robot and the orientation directly in $[x, y, \theta]$. The measurements from the internal navigation system is used to get the second set of estimates of the robots position and orientation. The relation between the position, orientation, velocity and acceleration can be expressed as

$$x = x_0 + Tv^x + \frac{1}{2}T^2a^x \tag{6.1}$$

$$y = y_0 + Tv^y + \frac{1}{2}T^2a^y \tag{6.2}$$

$$\theta = \theta_0 + T\omega + \frac{1}{2}T^2a^\omega \tag{6.3}$$

$$v^x = v_0^x + Ta^x \tag{6.4}$$

$$v^y = v_0^y + Ta^y \tag{6.5}$$

$$\omega = \omega_0 + Ta^\omega \tag{6.6}$$

In the above equations the values x_0, y_0, θ_0 are the initial position and orientation of the robot. This initial position will always be from the UWB system because the internal navigation system always requires an initial reference from which it can start to calculate the positions in the following time steps. In simpler words, when the robot is started, the first set of values for $[x, y, \theta]$ will always be taken from the UWB system and the following ones will be taken from both. That is at the first time step the measurement from the UWB system is the only measurement received, and from the next time step the system gets two values of the pose and from here on the sensor fusion part is important. In the above equations T represents the integration time (basically the time of one time step, the accelerometer works at a frequency of 1kHz thus the time step is $0.001s$). v_0^x and v_0^y are the initial velocities in x and y directions respectively and ω_0 initial angular velocity. a^x and a^y are the linear accelerations in x and y directions respectively, and a^ω is the angular acceleration. The output for the accelerometer is the accelerations in the tangential direction, denoted by a^t, and the normal direction, denoted by a^n. The relation between the tangential, normal accelerations to the accelerations in the x and y directions is

$$a^x = a^t cos\theta - a^n sin\theta \tag{6.7}$$

$$a^y = a^t sin\theta + a^n cos\theta \tag{6.8}$$

The same concept is also applied to get the relationship between the velocities in x,y and t,n directions and is

$$v^x = v^t cos\theta - v^n sin\theta \tag{6.9}$$

$$v^y = v^t sin\theta + av^n cos\theta \tag{6.10}$$

The UWB system works over a low latency wireless network, and the fact that the algorithm for its working is based on contacting each of the 6 beacons individually and then computing the position of the robot makes it a slow process. Due to these facts, it can be argued that there would be some delays in the process namely communication delays between the tag and the beacons and the way back to the tag and also computation delays for the calculation of the position of the robot from all the separate range measurements. The delay compensation block designed as the part of the sensor fusion block in the framework block diagram in figure (6.1) is made to take care of this particular problem of the UWB system. The first step is to calculate the communication delay between the tag and one of the beacons and then for all of them. Then the computation delay and sum the delays and have a set value for it, for this case, it was not possible to run the tests thus it is just a proposed method to do the sensor fusion for localization. Thus, let us assume the value of the delay to be represented by the variable d. The basic idea of the delay compensation block is to use the value coming in from the UWB system at time k as the initial position (which actually represents the position of the robot at time k-d) and it also takes the state estimates as another set of inputs, which are the output of the sensor fusion block, basically $[\hat{\theta}, \hat{v}^t, \hat{v}^n, \hat{\omega}, \hat{a}^t, \hat{a}^n, \hat{a}^\omega]$. Then the values of the state estimates are integrated from time $k-1$ until time $k-d$ and then stored in a buffer. Adding these values to the initial position at k-d which is the input from the UWB system can be used to reconstruct the trajectory between the times $k-d$ and $k-1$ to have an accurate prediction of k. Mathematically this computation in the delay compensation block can be represented as

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \sum_{n=k-d}^{n=k-1} \left(\begin{bmatrix} T\hat{v}_n^x + \frac{1}{2}T^2\hat{a}_n^x \\ T\hat{v}_n^y + \frac{1}{2}T^2\hat{a}_n^y \\ T\hat{\omega}_n + \frac{1}{2}T^2\hat{a}_n^\omega \end{bmatrix} \right) \tag{6.11}$$

Where T represents the sampling time, while x_k,y_k and θ_k are the predicted position and the orientation at time step $t = k$ while x_{k-d},y_{k-d} and θ_{k-d} are the delayed position and orientation that are received from the UWB system at the time step $t = k$. In theory, the velocities and accelerations for the delay compensation block can also be directly used from the data from the accelerometer and the gyroscope, but it would be better to use the output of the Kalman filter, because the signal has lower amount of noise as it is basically the filtered data from the internal navigation system. Thus it is advised to use the output of the sensor fusion block as the state estimates used in the delay compensation block, as shown in figure (6.1)

The second part of the sensor fusion block is the Kalman filter which is responsible for the actual process. For this case the use of a multi-rate extended Kalman filter should be preferred, since the UWB system works at a very slow rate (around 5 to 10 Hz) which is much slower than the internal navigation system measurements (1 kHz). The extended Kalman filter basically uses one prediction model where it predicts all of the state estimates $[\hat{x}, \hat{y}, \hat{\theta}, \hat{v}^t, \hat{v}^n, \hat{\omega}, \hat{a}^t, \hat{a}^n, \hat{a}^\omega]$ and two sets of measurement models. The UWB system works at a very low rate, thus if the measurement form the UWB system is not available for the Kalman filter to use, it only uses the set of measurements from the internal navigation system which is working at a high rate. But if the measurement from the UWB is available the system moves to the second measurement model where it uses the values from both the sources and gives the state estimates. This model works at a low rate which is the same as that of the UWB system measurements. During all of this the prediction model works at a high rate as it is the same model used for both the measurement models. The full calculations for the Kalman filter could not be done in this project due to the lack of time, but the mathematical model for the prediction and the measurement models were designed.

The prediction model of the EKF is implemented in continuous time in Kalman-Bucy form [20] [21] while the model of measurement update is implemented in discrete time. A generic prediction model is used and is given as

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t) + B(t)u(t) + w(t) \tag{6.12}$$

In equation (6.12) $\hat{x}(t)$ is a column vector consisting of all the predicted state estimates, $F(t)$ is the dynamic coefficient matrix, $u(t)$ is the input vector matrix. The input vector matrix for this case is zero because the system does not have any control input as the system is just an observer which is used for the prediction of the state estimates. Finally $w(t)$ is the process noise which is generally assumed to be Gaussian random variable with a zero mean and covariance Q, $w(t)\ N(0, Q)$.
The prediction model is giving the following states $[\hat{x}, \hat{y}, \hat{\theta}, \hat{v}^t, \hat{v}^n, \hat{\omega}, \hat{a}^t, \hat{a}^n, \hat{a}^\omega]$ thus a jerk is modeled to be in the output as well. The prediction model in this case can be expressed as

$$\begin{bmatrix} \dot{\hat{x}}(t) \\ \dot{\hat{y}}(t) \\ \dot{\hat{\theta}}(t) \\ \dot{\hat{v}}^t(t) \\ \dot{\hat{v}}^n(t) \\ \dot{\hat{\omega}}(t) \\ \dot{\hat{a}}^t(t) \\ \dot{\hat{a}}^n(t) \\ \dot{\hat{a}}^\omega(t) \end{bmatrix} = F(t) \begin{bmatrix} \hat{x}(t) \\ \hat{y}(t) \\ \hat{\theta}(t) \\ \hat{v}^t(t) \\ \hat{v}^n(t) \\ \hat{\omega}(t) \\ \hat{a}^t(t) \\ \hat{a}^n(t) \\ \hat{a}^\omega(t) \end{bmatrix} + w(t) \tag{6.13}$$

Where the dynamic coefficient matrix is derived from the kinematic model of the robot and is expressed as

$$F(t) = \begin{bmatrix} 0 & 0 & 0 & cos\hat{\theta}(t) & -sin\hat{\theta}(t) & 0 & dt.cos\hat{\theta}(t) & -dt.sin\hat{\theta}(t) & 0 \\ 0 & 0 & 0 & sin\hat{\theta}(t) & cos\hat{\theta}(t) & 0 & dt.sin\hat{\theta}(t) & dt.cos\hat{\theta}(t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.14}$$

The generic measurement model in the discreet time domain can be expressed as

$$z_k = H.x_k + v_k \tag{6.15}$$

with z_k being a column vector of the state measurements from the sensors, H is defined as the measurement sensitivity matrix and v_k is defined as the measurement noise of the system. which can be assumed as a Gaussian random variable with zero mean and a covariance J, $v_k`N(0, J)$. The first measurement model where the measurements are attained only from the internal navigation system, thus the measurements available are namely (ω, a^t, a^n). This model runs at a high sampling rate and is expressed as

$$\begin{bmatrix} \omega \\ a^t \\ a^n \end{bmatrix} = H \begin{bmatrix} x_k \\ y_k \\ \theta_k v_k^t \\ v_k^n \\ \omega_k \\ a_k^t \\ a_k^n \\ a_k^\omega \end{bmatrix} + v_k \tag{6.16}$$

Where the measurement sensitivity matrix can be expressed as shown in equation (6.17) which is basically the matrix which relates the state estimates to the values that are coming in from the sensors.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.17}$$

The measurement noise can be represented as

$$J = \begin{bmatrix} \sigma_\omega^2 & 0 & 0 \\ 0 & \sigma_{a^t}^2 & 0 \\ 0 & 0 & \sigma_{a^n}^2 \end{bmatrix} \tag{6.18}$$

In equation (6.18) the variables σ_ω^2, $\sigma_{a^t}^2$ and $\sigma_{a^n}^2$ are the noise variances of the internal navigation system measurement ω, a^t and a^n respectively. These values can be measured from the measured sensor signals. This can generally be done by making the robot run over a fixed distance multiple times and recording the values and then computing the standard deviation and the variances. Which in case of the internal navigation system is harder because the sensors do not log their own covariances. Whereas in the case of the UWB beacons they log the standard deviation of the measurements and finding the variances is much easier in that case.
The second measurement model is with both the UWB system and the internal navigation system. Thus the equation (6.15) transforms to

$$\begin{bmatrix} x \\ y \\ \theta \\ \omega \\ a^t \\ a^n \end{bmatrix} = H \begin{bmatrix} x_k \\ y_k \\ \theta_k v_k^t \\ v_k^n \\ \omega_k \\ a_k^t \\ a_k^n \\ a_k^\omega \end{bmatrix} + v_k \tag{6.19}$$

Here the measurement sensitivity matrix develops into

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.20}$$

and the noise matrix includes the noise variances of the measurements of x, y, θ from the UWB system and is as

$$J = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 & 0 & 0 \\ 0 & 0 & 0\sigma_\omega^2 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \sigma_{a^t}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{a^n}^2 \end{bmatrix} \tag{6.21}$$

In this chapter a theoretical model for the prediction and measurement models for the extended Kalman filter are derived. These models are used for the multi-rate sensor fusion of the two different localization methods. For these models to be implemented onto the robot, the covariance matrix for the UWB system (R) needs to be obtained from the UWB beacons once the system is running and functional again (for the measurement model) and he process noise matrix, ($w(t)$) once all the sensors are functional (for the prediction model).

Chapter 7

Conclusions and future work

The main aim of the project was to develop a control algorithm such that the robot is able to reach the desired coordinates (path following) that are provided to it by the planner and also to provide a sensor fusion algorithm for localization of the robot through the UWB system and the inertial measurement system of the robot.

7.1 Conclusions

A path following controller is developed which is able to drive the robot to the given set of co-ordinates. The controller was tested on a pre-defined path and not on a path that is autonomously generated by the planner, since the path planner has not yet been designed. Thus, the full capability of the controller could not be tested.
A theoretical model for multi-rate sensor fusion using an extended Kalman filter for the two localization process was developed. Due to System upgrades for the UWB system the algorithm could not be tested as the system was not available for use.

7.2 Recommendations

- **Controller**
  - Verify the controller with the UWB system.
  - Test the controller with the area coverage planning algorithm once the planner is designed and implemented.
- **Sensor Fusion**
  - To find the covariance matrix of the UWB system once it is up and running.
  - To find the system noise matrix using the UWB beacons.
  - Implementing the prediction and measurement model.
  - Tuning the Kalman gains while testing the sensor fusion on the robot.

Chapter 8

Bibliography

[1]. Stephen Armah, Sun Yi and Taher Abu-Lebdeh. Implementation of autonomous navigation algorithm on two wheeled ground mobile robot. American journal of engineering and applied sciences, 2014.
[2] Ibari Benaoumeur, Benchikh Laredj, H. E. A. Reda and Ahmed-foitih Zoubir. Backstepping aproach for autonomous mobile robot trajectory tracking. Indonesian journal of electrical engineering and computer Science, june 2016.
[3]. Ricardo Carona, A. Pedro Aguiar, Jose Gaspar. Control of unicycle type robots, tracking, path following and point stabalization. November 2008
[4]. Zhi-An Deng, Ying Hu, Jianguo Yu and Zhenya Na. Extended kalman filter for Real time Indoor localization by fusing WiFi and smartphone inertial sensors. Michromachines ISSN 2072-666X. 2015.
[5]. Yassen Doborev, Sergio Flores, Martin Vossiek. Multi-modal sensor fusion for indoor mobile robot pose estimation. FAU Erlangen, germany.
[6]. A. Hladio, C. Nielsen, and D. Wang. Path following controller design for a class of mechanical systems. In 18th World Congress of the International Federation of Automatic Control, Milano, Italy, August 2011. Accepted.
[7]. H. K. Khalil. Nonlinear systems. Prentice Hall, 2002.
[8]. D. Kostic, S. Adinandra, J. Caarls and H. Nijmeijer. Collision-free motion coordination of unicycle multi-agent systems. American control conference. 2009.
[9]. Ti-Chung Lee, Kai Tai Song, Ching-Hung Lee and Ching-ching Teng. Tracking control of unicycle modeld mobile robot using a saturation feedback controller. IEEE transactions on control systems technology. March 2001.
[10]. David McNeil Mayhew. Multi-rate fusion for GPS navigation using kalman filtering. Master thesis, Virginia Polytechnic Institute and State University. 1999.
[11]. Rudy negenborn. Robot localization and kalman filters on finding your position in a noisy world. master thesis, Utrecht University. 2003.
[12]. C. Nielsen. Maneuver regulation, transverse feedback linearization and zero dynamics. Masters thesis, Department of Electrical and Computer Engineering University of Toronto, 2004.
[13]. C. Nielsen, C. Fulford, and M. Maggiore. Path following using transverse feedback linearization: Application to a maglev positioning system. Automatica, 46:585590, March 2010.
[14]. C. C. Pugh. Real mathematical analysis. Springer, New York, 2002.
[15]. Joanna Plaskonka. Different kinematic path following controllers for a wheeled mobile robot of (2,0)type. 2013.
[16]. J.Z. Sasiadek and P. Hartana. Sensor Data Fusion using Kalman Filter. Carleton university.
[17]. S. Sastry. Nonlinear systems: analysis, stability, and control. Springer, New York, 1999.
[18]. Francesco Vanni, A. Pedro Aguiar, and Antonio M. Pascoal. Cooperative path-following of underactuated autonomous marine vehicle with logic-based communication. In Proc. of NGCUV08 - IFAC Workshop on Navigation, Guidance and control of underwater vehicles, pages 16, Lisbon, Portugal, Apr 2008.

[19]. David De Von and Timothy Bretl. Kinematic and dynamic control of a wheeled mobile robot. University of Illinois. IEEE, November 2007.
[20] R. E. Kalman, A new approach to linear filtering and prediction problems, ASME Journal of Basic Engineering, Vol. 82, pp. 34-45, 1960.
[21]R. E. Kalman, R. S. Bucy, New results in linear linear filtering and prediction theory, ASME Journal of Basic Engineering, Series D, Vol. 83, pp. 95-108, 1961.

Appendix A

Inverse function theorum

Theorem 5.1 (Inverse function theorem [14]) *Let U be an open subset o R^n and $f : U \to R^n$, a C^∞ mapping. If the Jacobian, df_{x^*}, is nonsingular at some x^* in U, then there exists an open neighborhood V of x^* in U such that $W = f(U)$ is open in R^n and $f|_v$ is a diffeomorphism onto W*

Corollary 5.2 Let $x^* \in \Gamma$. There exists a neighbored $U \subset R^3$ containing x^* such that the mapping $T : U \subset R^3 \to T(U) \subset R^3$ which is defined in the equation 4.19 The equation 4.19 is a diffeomorphism onto its image.
Proof: Let $x^* \in \Gamma$. By the direct calculations the Jacobian $dT_{x^*} := \frac{\delta T}{\delta x}|_{x=x^*}$ is given by

$$dT_{x^*} = \begin{bmatrix} \frac{\delta z_1}{\delta x_1} & \frac{\delta z_1}{\delta x_2} & \frac{\delta z_1}{\delta x_3} \\ \frac{\delta z_2}{\delta x_1} & \frac{\delta z_2}{\delta x_2} & \frac{\delta z_2}{\delta x_3} \\ \frac{\delta z_3}{\delta x_1} & \frac{\delta z_3}{\delta x_2} & \frac{\delta z_3}{\delta x_3} \end{bmatrix} = \begin{bmatrix} -x_2 & x_1 & 0 \\ 2vcosx_3 & 2vsinx_3 & 2v(x_2cosx_3 - x_1sinx_3) \\ 2x_1 & 2x_2 & 0 \end{bmatrix} \tag{A.1}$$

The determinant of the matrix in A.1 is given as shown in A.2

$$det(dT_{x^*} = -4v(x_2cosx_3 - x_1sinx_3) \tag{A.2}$$

As shown before that on the path following manifold Γ this determinant is equal to zero only under the condition that $v = 0$. The condition set is that $v \neq 0$, thus the Jacobian dT_{x^*} is nonsingular. In accordance with Theorem(5.1), T is diffeomorphism onto its image