Implementation of Nonlinear Model Predictive Control on all Terrain Mobile Robot

by

©Zohaib H. Farooqi

A Thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

May 2017

St. John s Newfoundland

Abstract

The objective of this thesis is to control a mobile robot with nonholonomic constraints to achieve two control objectives: point stabilization and trajectory tracking. This research adopts Nonlinear Model Predictive Control (NMPC) to achieve these control objectives. The mobile robot platform used in the research is Seekur Jr., which is a skid-steering all terrain mobile robot with nonholonomic constraints. In this study NMPC is developed and tested for both indoor and outdoor navigation. To address the indoor localization issues, two methods have been adopted. In the former approach for indoor localization, a map of the environment is generated using a laser range finder. This map, along with laser range finder, is used to determine the pose (position and orientation) of the mobile robot in the environment. In the second approach, OptiTrack motion capture system has been used, which gives the position data of the mobile robot in the environment and orientation is evaluated through this. For outdoor navigation, Global Positioning System (GPS) is used to obtain the localization. The implementation of NMPC involves solving a dynamic optimization control problem, which makes the evaluation of control command time consuming. Therefore, it is di cult to implement NMPC for mobile robots in real-time applications. To address this issue, an open source toolkit solving Optimal Control Problem (OCP) has been used to implement fast NMPC routine, which provides real-time applicability of the control strategy. Obstacle avoidance feature is also added to the

controller to avoid static obstacles in the trajectory of the mobile robot. The proposed control strategy is evaluated on a number of simulations and experimental studies. The results validate the real-time applicability of the proposed approach in indoor and outdoor navigation.

Acknowledgements

I would first like to thank my supervisors Dr. George K. I. Mann and Dr. Raymond G. Gosine for their advice and guidance towards completion of my Master of Engineering degree. It was matter of privilege to work at Intelligent Systems (IS) Lab, under their supervision. Their precious lessons and guidance enhanced my research skills and enabled me to complete this research work. They consistently allowed this paper to be my own work, but steered me in the right the direction whenever they thought I needed it.

I would also like to thank my lab mates Mr. Thumeera R. Wanasinghe, Mr. Mohamed Mehrez, Mr. Trung Nguyen, Dr. Oscar De Silva, and Mr. Eranga Fernando. I would like to thank them for their great assistances, patience and inspirations.

Finally, I must express my very profound gratitude to my parents, my teachers and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

This research work is supported by Research and Development Corporation (RDC) in NL and Memorial University of Newfoundland for funding this research work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACADO Automatic Control and Dynamic Optimization

ARIA Advanced Robot Interface for Applications

ARNL Advanced Robotics Navigation and Localization

DGPS Di erential Global Positioning System

DMC Dynamic Matrix Control

EKF Extended Kalman Filter

GPS Global Positioning System

IMU Inertial Measurement Unit

LQ Linear Quadratic

MHE Moving Horizon Estimation

MOGS Mobile robot Outdoor Guidance System

MPC Model Predictive Control

MPHC Model Predictive Heuristic Control

NMPC Nonlinear Model Predictive Control

PID Proportional-Integral-Derivative

QP Quadratic Programming

rms root mean square

UKF Unscented Kalman Filter

Chapter 1

Introduction

1.1 Introduction

In the past few decades, the use of mobile robots has been extensively increased for autonomous applications, to achieve task oriented objectives [1]. The examples are in exploration tasks, especially in space, where exploration is constrained for human beings and mobile robots are the only source to accomplish that; typical examples are Mars rovers [2 4]. Mobile robots are being used in the inaccessible and endangered areas for humans, typical examples are in disaster relief operations [5], search and rescue operations [6] and assessing the situation in natural calamity hit areas [7]. Moreover, mobile robots are also being utilized for military applications [8, 9], surveillance [10, 11], domestic applications [12, 13] and entertainment purposes [14].

Mobile robots can be classified in di erent categories by choosing di erent criteria (see [15] for details), based on:

- area of application: ground, ariel, underwater and polar
- locomotion: legged and wheeled

- motion constraints: holonomic and nonholonomic
- scale: large scale, micro and nano

This research is focused on nonholonomic class of mobile robots. The nonholonomic mobile robot platform used in this research is a skid-steered type wheeled mobile robot. In skid steering locomotion, wheels on each side are controlled and can be driven at di erent speeds. There is no explicit steering mechanism and to cause the turning e ect, the wheels on each side are driven at di erent speeds [16, 17].

Development of optimal control strategy is the key requirement to enable the mobile robot for autonomous navigation, and accomplish the tasks mentioned in the beginning of the section. Many techniques have been developed in literature for the control of mobile robots. The classical non-optimization based controls, such as Proportional-Integral-Derivative (PID) controllers [18], have several limitations associated with them. First, the evaluated control command of PID control is solely dependent on the feedback, so it does not cater the real-time constraints, which might come across for such applications such as avoiding obstacles and variable speeds for change in terrain. Secondly, it is di cult to find the tuning parameters, which may e ectively stabilize the system. To fix those issues, several other controllers have been developed with the optimization techniques, such as Linear Quadratic (LQ). These controllers optimize the control signal to be applied while considering the few constraints, which optimize the control signal over the infinite horizon. Then, there comes state-of-the-art controller Model Predictive Controller (MPC), which considers the system model, so it accounts the dynamics of the system. Based on system model and other constraints, which corresponds to real world scenarios, MPC optimizes and evaluates the control command, to drive the mobile robot. Therefore, the controller is more robust and corresponds to real world scenarios, which is the fundamental requirement for autonomous applications. MPC does the optimization over the finite

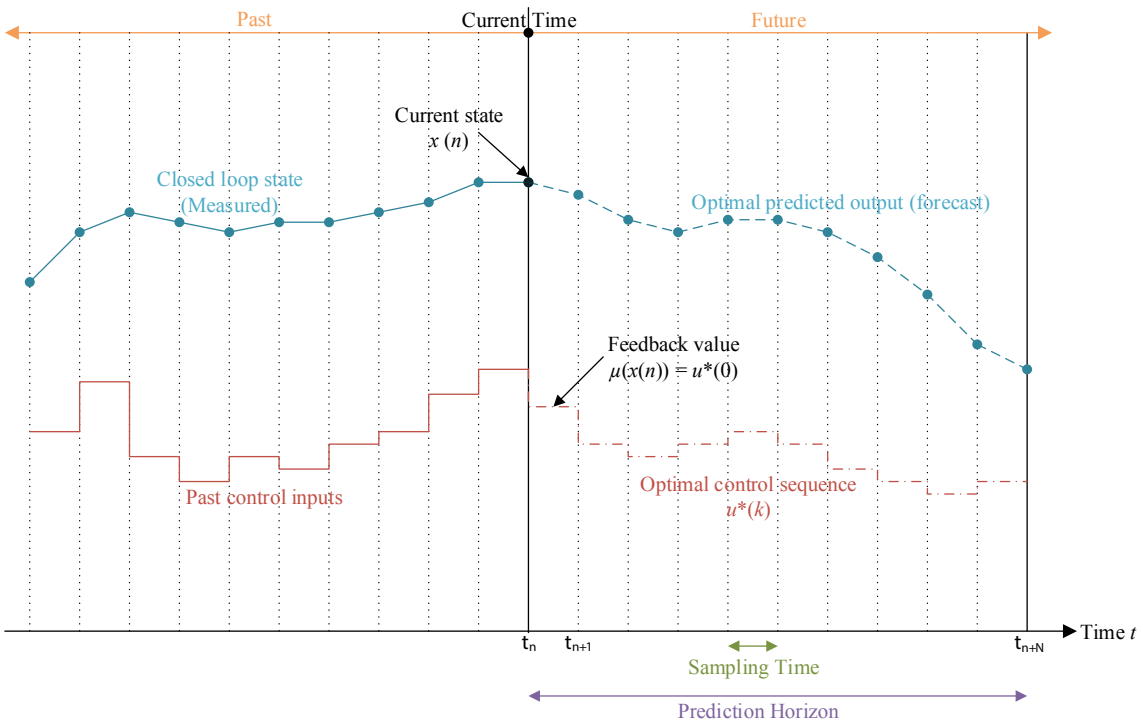


Figure 1.1: Nonlinear Model Predictive Control

horizon and predicts the future control commands and the trajectory [19]. The control strategy is realistic, as it accounts for the constraints and bounds introduced by real world operations.

The core idea of the MPC is to explicitly use a model to predict and optimize the process output at future time instants (horizon), and compute a control sequence over the prediction horizon [19, 20]. This involves the receding horizon strategy, at each instant the horizon is displaced towards the future, which involves the application of the first control signal of the optimized sequence evaluated at each step [19,20]. Figure 1.1 illustrates the MPC strategy. MPC forecasts the optimal predicted output and evaluates the optimal control sequence, based on closed loop state (from measurement) and system model. Future prediction and optimization is done over a finite horizon, called prediction horizon. The key point here is that when it comes to apply the control command, it only applies the first control command from the optimized control

sequence. In the next time step the prediction horizon is shifted by one time step in the future. That is why it is also called receding horizon control [19,20], also depicted in Figure 1.1. For further details see [19 21].

As the model of nonholonomic mobile robot, used in this research work, is nonlinear, so it requires Nonlinear Model Predictive Control (NMPC). There are two variants of NMPC, that are broadly used for the mobile robots.

- In the first method, nonlinear system model is linearized using linearization techniques, followed by application of the linear MPC approaches [22 24]
- In the second method, NMPC is applied using the nonlinear model of the system directly [24 26]

This research work implements the second method. The problem with the linearized version is that the model of the mobile robot is highly nonlinear and linearization is performed under many assumptions. These assumptions are only fulfilled provided the conditions remain the same. Secondly, these assumptions are only true for a confined region around the point of linearization, beyond that region the linearization is not valid. Whereas, when nonlinear system model is used, then we are not performing any linearization and the issues which are associated with the linearization process are avoided. System model used may be continuous, which is the more likely of the real world scenario. However, this imposes a problem that NMPC involves dynamic optimization control problem, which is time consuming and not feasible for real-time applications. To resolve this issue and make it possible for real-time application, this study uses an open source toolkit called Automatic Control and Dynamic Optimization (ACADO) Toolkit [27] which implements real-time NMPC routines.

1.2 Problem Statement

The objective of this thesis is to develop a controller for autonomous navigation of nonholonomic mobile robot for indoor and outdoor places to solve two control problems: point stabilization and trajectory tracking, in the absence or presence of the static obstacles.

To address this problem, this thesis implements NMPC for nonholonomic mobile robot in indoor and outdoor terrain and achieves two control objectives: point stabilization (regulation) and trajectory tracking. Mobile robot platform used is Seekur Jr. [28] (shown in figure 1.2), which is equipped with Laser range finder, Di erential Global Positioning System (DGPS) module, Inertial Measurement Unit (IMU), on board computer and Wi-Fi module.

Figure 1.2: Seekur Jr., Mobile robot research platform

1.3 Objectives and Contributions

The main objectives and contributions of this thesis are listed below:

Objective 1 Implementation of NMPC to solve point stabilization problem of non-holonomic mobile robot

- **Contribution 1**: Real-time implementation of NMPC to solve point stabilization problem for indoor navigation
- **Contribution 2**: Real-time implementation of NMPC to solve point stabilization problem for outdoor navigation
- **Contribution 3**: Real-time implementation of NMPC to solve point stabilization problem with obstacle avoidance

Objective 2 Implementation of NMPC to solve trajectory tracking problem of non-holonomic mobile robot

- **Contribution 1**: Real-time implementation of NMPC to solve trajectory tracking problem for indoor navigation
- **Contribution 2**: Real-time implementation of NMPC to solve trajectory tracking problem for outdoor navigation
- **Contribution 3**: Real-time implementation of NMPC to solve trajectory tracking problem with obstacle avoidance

1.4 Thesis Overview

Chapter 1 introduces the focused area of research. It also outlines the objectives and contributions of the thesis.

Chapter 2 studies the background of NMPC and its applicability for mobile robots. Furthermore, it presents some mobile robot control techniques emphasizing their relative advantages and disadvantages.

Chapter 3 models the tracking control problem with presentation of robot kinematic model used for navigation. NMPC formulation and ACADO implementation is also presented. Overall control strategy for methods, adopted in this research work, is also explained.

Chapter 4 presents the point stabilization results of the mobile robot. Simulation and experimental results are being presented for both indoor and outdoor navigation. Point stabilization with obstacle avoidance is also demonstrated.

Chapter 5 presents the trajectory tracking results of the mobile robot. Simulation and experimental results are being presented for both indoor and outdoor navigation. Trajectory tracking with obstacle avoidance is also demonstrated.

Chapter 6 presents the concluding remarks by mentioning the research contributions and limitations. Future directives are also highlighted.

Chapter 2

Background

About this Chapter: This chapter studies the background for control of nonholonomic mobile robots. It presents predictive and non-predictive control techniques used for mobile robot emphasizing their relative advantages and disadvantages. Furthermore it puts forward the history of NMPC and its applicability for mobile robots.

2.1 Control Problems of Non-holonomic Mobile Robots

Feedback is one of the fundamental concept deployed for automation and control. The basic control problems studied in literature widely for nonholonomic mobile robots includes point stabilization, trajectory tracking and path following [29]. In order to explain these control problems, let s take a general nonlinear system, expressed as [30]

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad , \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.1}$$

where $t \in \mathbb{R}$ is the time, which is sampled at defined sampling rate, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are the n dimensional system state vector and m dimensional system control input, respectively. Moreover, state constraints $\mathbb{X} \in \mathbb{R}^m$ and control constraints $\mathbb{U} \in \mathbb{R}^n$ can be defined for this system.

In point stabilization control problem, a feedback control $\mu(\mathbf{x}(t)) : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\mu(\mathbf{x}(t)) \in \mathbb{U}$ is designed such that the 2.1 starting at the initial condition, $\mathbf{x}_0 \in \mathbb{R}^n$ driven by the feedback, stays close to the desired set point, $\mathbf{x}_s \in \mathbb{X}$, and converges, i.e.

$$\lim_{t \to \infty} \|\mathbf{x}(t) - \mathbf{x}_s\| = 0 \tag{2.2}$$

The important point to notice here is that in point stabilization, the set point $\mathbf{x}_s$ is a constant reference, but in case of trajectory tracking this reference is time dependent, i.e. $\mathbf{x}_r(t) : t \mapsto \mathbb{R}^n$. In trajectory tracking problem, the feedback control $\mu(\mathbf{x}(t)) : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\mu(\mathbf{x}(t)) \in \mathbb{U}$ is designed such to steer the solution of 2.1 to track the time varying reference trajectory, such that

$$\lim_{t \to \infty} \|\mathbf{x}(t) - \mathbf{x}_r(t)\| = 0 \tag{2.3}$$

It can be observed here that for constant reference, trajectory tracking problem becomes point stabilization problem. The third control problem is path following, but this thesis only focuses on point stabilization and trajectory tracking problems.

2.2 Non Predictive Controllers for Non-holonomic Mobile Robots

The nonholonomic constraints on the mobile robots makes the point stabilization problem di cult, due to Brockett s condition [31] a smooth time-invariant feedback

control law cannot be used to stabilize the mobile robot at a given posture. This implies that the linearized nonholonomic mobile robot model is not asymptotically stable and the problem is nonlinear. Asymptotic stabilization can be achieved by using time-varying or/and discontinuous controllers . Smooth time-varying stabilization was first presented in [32] and discontinuous feedback controller for point stabilization was proposed by [33]. Other work presented to solve point stabalization problem used dynamic feedback linearization [34, 35], Lyapunov control [36] and state space exact feedback linearization [37]. For further study see [1, 38].

Unlike point stabilization, trajectory tracking is more straight forward for non holonomic mobile robots as the kinematic constraints are implicitly considered by the reference trajectory. The trajectory tracking problem was globally solved by using a nonlinear feedback control in [39] and by using dynamic feedback linearization in [40, 41]. Other methods adopted for trajectory tracking includes Lyapunov stable time-varying tracking control law [42], back stepping technique [43] and sliding mode techniques [44]. Controllers with simultaneous regulation and tracking capabilities include di erential kinematic controller [45] and dynamic feedback linearization control [46]. The controllers developed in these two studies are not single controller and require switching between regulation and tracking controllers. For further details see [1, 38, 47, 48].

2.3 Model Predictive Control

2.3.1 History

MPC is not a specific control strategy, but it refers to the optimization based control methods which explicitly use a model of system to obtain the control command by minimizing an objective function [49]. The concept of MPC was first introduced in

middle of twentieth century [20]. Richalet *et al.* presented Model Predictive Heuristic Control (MPHC) [50, 51] and Cutler and Ramakter with Dynamic Matrix Control (DMC) [52] are considered as the pioneers to use MPC in process industry. In both algorithms, process model has been used to predict the future control commands by minimizing the predicted error, while considering the operational constraints. It also predicts the output of the system with future control commands. For further details of the early work in area of linear MPC see [53, 54]. As most of the industrial processes are nonlinear so there was a need of nonlinear MPC which would make use of a nonlinear dynamic model and nonlinear constraints [49]. The NMPC algorithm presented by [55] is considered as the first in this field [20]. Unlike the current versions of NMPC which only applies the first control command from the control sequence evaluated, this work applies the whole control sequence on optimization horizon to the process plant [20]. The NMPC, which applies only the first optimal control command from the optimized control sequence evaluated, for discrete time was first presented in [56] and for continuous time in [57]. For further details on the historical background on model predictive control see [20, 49, 58].

2.3.2 NMPC for Non-holonomic Mobile Robots

NMPC using linearization uses the linearized version of robot model which enables it to implement for trajectory tracking [59] while NMPC uses the explicit nonlinear robot model which enables it to implement for both point stabilization and trajectory tracking problems [25, 48]. Di erent strategies and early development for motion control of non holonomic wheeled mobile robots using MPC can be studied in [60 62] Some work studied in literature, for both approaches, is presented in the following sections.

2.3.2.1 Linear MPC

In [22] successive linearization approach is adopted, which yields a linear description of the nonlinear error model of the nonholonomic wheeled mobile robot. Then the control problem is solved through linear MPC. After following this linearization approach, the control inputs are assumed as decision variables, which makes it possible to convert optimization problem to Quadratic Programming (QP) problem. Quadratic Programming (QP) is solved by using numerical robust solver to get global optimal solution. Simulation results for trajectory tracking are presented here and it is shown that real time implementation is possible for the di erential drive mobile robot.

In [23] an actual real-time implementation of trajectory tracking problem is presented. It also uses the successive linearization approach as used in [22] and solves the Quadratic Programming (QP) to find global optimal solution. The optimization problem is solved by using library OOQP (Object Oriented software package for Quadratic Programming) [63]. Real-time results have been implemented on a di erential drive mobile robot with standard of the shelf computer to run MPC based controller. The experimental results reveal that the computational time in real-time scenario is was more than the estimated one in [22].

In [24] both linear and nonlinear approaches of MPC have been presented for trajectory tracking. Simulation results have been presented and real-time applicability is evaluated. The nonlinear variant of the MPC leads to the nonconvex optimization problem whereas linear variant leads towards the convex problem. The advantage of linear approach is that it is possible to reduce the computational e ort as it solves the QP.But on the other hand, linear approach has also the disadvantage that linearization is only valid near the point of linearization, i.e., linearized model is only valid for points near the reference trajectory.

2.3.2.2 Nonlinear MPC

The linear variant of MPC is only utilized for tracking control. In order to achieve regulation problem nonlinear variant is required. In [64] point stabilization for nonholonomic wheeled mobile robots has been achieved using NMPC. Simulation results have been presented and real-time applicability for nonholonomic wheeled mobile robot is evaluated. Two di erent cost functions has been utilized, a standard quadratic cost function and a modified cost function. Modified cost function exploits the idea of exponentially increasing state weighting, such that it increases the state penalty over the prediction horizon and the states converge to an acceptable solution. A terminal cost has been added to the cost function to be minimized. The modified cost function gives the better convergence as compare to the standard cost function.

In [25] both point stabilization and trajectory tracking results have presented on skid steered wheeled mobile robots, Pioneer 3-AT. It uses the open source toolkit ACADO to solve the optimization problem. Real time experimental results for indoor navigation have been presented which shows the satisfactory performance of the controller. The scope of the paper is limited to the indoor navigation and localization is map based localization

2.3.2.3 Contributions and problems addressed in thesis

According to best of author s knowledge, no previous work has been presented which implements NMPC on Seekur Jr. This work addresses both indoor and outdoor navigation to solve point stabilization and trajectory tracking problems. Moreover, for indoor navigation the use of motion capture system, to get the robot posture information, is not present in literature for these two problems using NMPC.

Chapter 3

Methodology of the Implementation

About this chapter: The overall methodology for the implementation of NMPC for mobile robot is presented in this chapter. Preliminaries are presented before the NMPC formulation, which includes the robot kinematics and tracking control modelling. After modelling the tracking control problem, NMPC is formulated and its implementation through ACADO toolkit is also presented. The chapter is concluded by discussing the overall system diagram to highlight the methodology developed for this research.

3.1 Robot Kinematic Model

If the motion of a nonholonomic mobile robot is described by its position (x, y) and orientation θ in Cartesian plane as shown in figure 3.1, then the kinematic equation

is as follow [48]:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.1}$$

where $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the state vector and $\mathbf{u} = \begin{bmatrix} v & \omega \end{bmatrix}^T$ is the control signal vector with v as the linear velocity and ω as the angular velocity. This model is called unicycle model, which represents the di erential drive mobile robots and used in this research work. The model is su cient to describe the nonholonomic constraints of this class of robots.

3.2 Tracking Control

In order to demonstrate the tracking control problem a reference robot, shown in figure 3.1, is defined with the reference state vector $\mathbf{x}_r = \begin{bmatrix} x_r & y_r & \theta_r \end{bmatrix}^T$ and reference control vector $\mathbf{u} = \begin{bmatrix} v_r & \omega_r \end{bmatrix}^T$. Reference robot has the same constraints as (3.1). So, its kinematic model can be expressed as [48]:

$$\dot{\mathbf{x}}_r = \begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\theta_r} \end{bmatrix} = \begin{bmatrix} v_r\cos\theta_r \\ v_r\sin\theta_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{3.2}$$

The fundamental control objective is to control (3.1) and track (3.2). Therefore, the error state vector $\mathbf{x}_e$ is defined as follow [48]:

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{3.3}$$

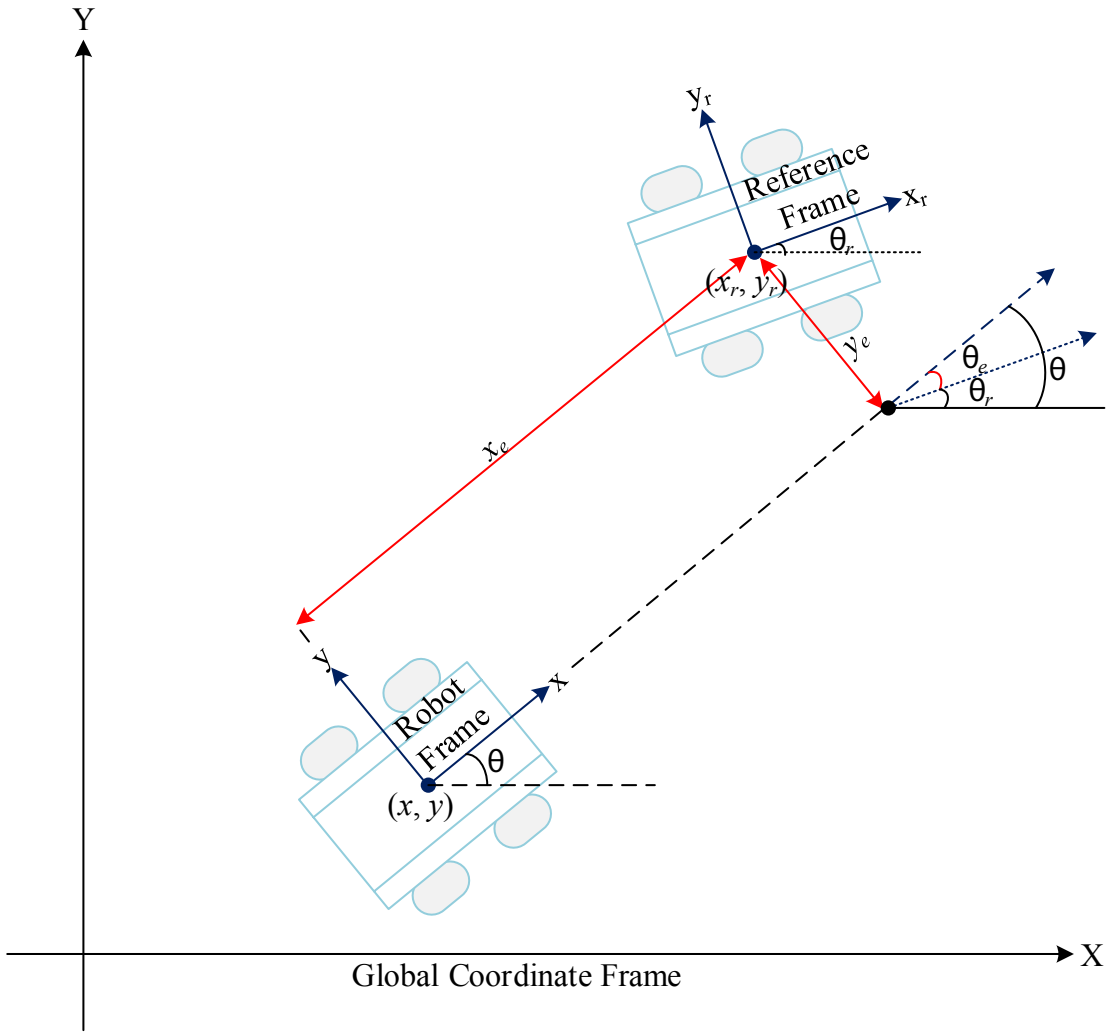


Figure 3.1: Mobile robot in global coordinate frame

Using the error state vector, the tracking control problem is converted into a regulation problem, where a constant reference is used to track the desired trajectory. From (3.3) it can be seen that tracking control objective can be achieved by driving the error state vector $\mathbf{x}_e$ to zero. By di erentiating (3.3), the error state dynamic model is obtained as follow [48]:

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{bmatrix} = \begin{bmatrix} \omega y_e - v + v_r \cos\theta_e \\ -\omega x_e + v_r \sin\theta_e \\ \omega_r - \omega \end{bmatrix} \tag{3.4}$$

For this error state model, redefining the error control signals as [48]:

$$\mathbf{u}_e = \begin{bmatrix} u_{1e} \\ u_{2e} \end{bmatrix} = \begin{bmatrix} v_r \cos\theta_e - v \\ \omega_r - \omega \end{bmatrix} \tag{3.5}$$

Then, the error state dynamic model (3.4) becomes

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{bmatrix} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} u_{1e} \\ v_r \sin\theta_e \\ u_{2e} \end{bmatrix} \tag{3.6}$$

A linearized version of error model of (3.4) can be obtained as follow:

$$\dot{\bar{x}}_e = \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{bmatrix} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{\mathbf{x}}_e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_e \tag{3.7}$$

This linearized version of error state model is controllable, but the controllability of this model is lost when reference linear and angular velocities approach zero. This thesis does not use this linear version, instead it uses nonlinear version directly.

3.2.1 Point Stabilization

In point stabilization, the reference state vector $\mathbf{x}_r$ for (3.2) has a fixed value, which corresponds to the pose of the desired goal position; and the reference control vector $\mathbf{u}_r$ for (3.2) is the null vector, which means that reference linear and angular velocities

are zero at the desired goal pose.

3.2.2 Trajectory Tracking

In trajectory tracking, the reference state vector $\mathbf{x}_r$ and the reference control vector $\mathbf{u}_r$ for (3.2) are time varying values corresponding to the reference trajectory.

3.3 Nonlinear Model Predictive Control Formulation

In this section, a brief description of NMPC scheme for mobile robots with nonholonomic constraints and its implementation in ACADO toolkit is presented.

3.3.1 Nonlinear Model Predictive Control Design

The general nonlinear control system, like in (3.1), can be expressed as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.8}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ are the n dimensional state and m dimensional control vectors, respectively. The function f is assumed to be continuous. If the model is expressed in error form, like (3.4), then the general expression will be [48]:

$$\dot{\mathbf{x}}_e(t) = f(\mathbf{x}_e(t), \mathbf{u}_e(t)) \tag{3.9}$$

where $\mathbf{x}_e(t) \in \mathbb{R}^n$ and $\mathbf{u}_e(t) \in \mathbb{R}^m$ are the n dimensional error state vector and m dimensional error control vector, respectively. The control objective is to compute a suitable control input $\mathbf{u}(t)$ to drive the system in (3.8) towards the equilibrium

position ($\mathbf{x}_e(t) = 0$ and $\mathbf{u}_e(t) = 0$). The constraints employed normally are the control signal saturation constraints that can be expressed as [48]:

$$\mathbf{u}_e(t) \in U \tag{3.10}$$

where $0 \in U \in \mathbb{R}^m$ is a compact and convex set.

The objective of the tracking control algorithm is to minimize a weighted cost function expressed as [48]:

$$J(t, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) = \int_t^{t+T} l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau))\, d\tau \tag{3.11}$$

where $l(\tau, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) = \mathbf{x}_e(\tau)^\tau \mathbf{Q}\mathbf{x}_e(\tau) + \mathbf{u}_e(\tau)^\tau \mathbf{R}\mathbf{u}_e(\tau)$ is the running cost and $\mathbf{Q}$ and $\mathbf{R}$ are the positive definite symmetric weight matrices and T is the prediction horizon. At time t, the open loop optimization problem (OP) to be solved online can be formulated as follow [48]:

$$\begin{aligned} \underset{\mathbf{u}_e}{\text{minimize}} \quad & J(t, \mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ \text{subject to:} \quad & \\ & \dot{\mathbf{x}}_e(\tau) = f(\mathbf{x}_e(\tau), \mathbf{u}_e(\tau)) \\ & \mathbf{u}_e(\tau) \in U, (\tau \in [t, t+T]) \end{aligned} \tag{3.12}$$

where $0 \in U \in \mathbb{R}^m$ is a compact and convex set as in (3.10)

3.3.2 ACADO Toolkit

ACADO (Automatic Control and Dynamic Optimization) Toolkit [27] is a package, which implements real-time NMPC routines [65]. The key features of ACADO toolkit, which make it superior than other packages available are [66, 67]:

- open source
- user-friendliness, i.e., interface close to mathematical syntax
- code extensibility, as it is easy to link existing algorithms and also serve as the platform for new developments
- self-contained, i.e., only need C++ compiler
- design for closed loop MPC implementation

Moreover, this toolkit provides MATLAB interface. This interface brings ACADO integrators and algorithms for direct optimal control, model predictive control and parameter estimation to MATLAB. It uses the ACADO Toolkit C++ code base and implements methods to communicate with this code base [68]. As the algorithm in this study is implemented on MATLAB, so this feature was worthy to use. It complies the code and generates mex (MATLAB Executable) file, which solves the NMPC problem in the run time.

The general form of the NMPC problem solved by ACADO toolkit is given by [65]:

$$
\begin{aligned}
\underset{x(.),u(.),p(.)}{\text{minimize}} \quad & \int_t^{t+T} \left\| h\left(x\left(t\right), u\left(t\right), p\right) - \eta\left(t\right) \right\|_Q^2 dt \\
& + \left\| m\left(x\left(t_o + T\right), p, t_o + T\right) - \mu \right\|_P^2 \\
\text{subject to:} & \\
& x\left(t_o\right) = x_o \\
\forall t \in [t_o, t_o + T] \quad & 0 = f\left(t, x\left(t\right), \dot{x}\left(t\right), u\left(t\right), p\right) \\
\forall t \in [t_o, t_o + T] \quad & 0 \geq s\left(t, x\left(t\right), u\left(t\right), p\right) \\
& 0 = r\left(x\left(t_o + T\right), p, t_o + T\right)
\end{aligned}
\tag{3.13}
$$

where $x(.)$ denote the system states, $u(.)$ the control input, p a time constant parameter (optional), T the prediction horizon time, $f(.)$ the model equation, $s(.)$ the path constraints, and $r(.)$ the terminal constraints. Moreover, the objective function is given in the least square form, where η and μ specify the tracking and terminal reference. The control problem (3.13), written in scalar notation, can be easily matched with the robot control problem given by (3.12). For the mobile robot unicycle model three system states are present, so $x(.)$ contains three entities x-position, y-position and orientation, θ. There are two control inputs, linear velocity (v) and angular velocities (ω). Therefore, $u(.)$ comprises of two elements v and ω. Prediction Horizon (T) is selected such that the system performance is satisfactory. The value is specified in problem formulation subsection of each problem sections, individually. Constraints have been put on velocities and specified in the experimental setup sections. No terminal constraint has been implemented in this work.

3.4 System Specifications

The mobile robot platform used for this study is Seekur Jr. [28], which is a four-wheel, skid-steer di erential drive mobile robot with an embedded computer, client-server communications capability, laser range scanner and many other useful features. MATLAB codes have been written which integrates the NMPC routine implemented by ACADO toolkit, Seekur Jr. input and output files and communication files. Following softwares have been used in this research work:

- MATLAB : Platform to connect all the softwares and control code.
- ARNL [28]: ARNL (Advanced Robotics Navigation and Localization) server runs on the mobile robot to run the robot according to the command velocity received for indoor navigation. It uses the map of the environment and laser

range finder to perform localization task.

- MOGS [28]: MOGS (Mobile Robot Outdoor Guidance System) server runs on the mobile robot to run the robot according to the command velocity received for outdoor navigation. It uses DGPS to perform the localization task.
- ARIA [28]: ARIA (Advanced Robot Interface for Applications) is the platform which provides the client service for ARNL/MOGS server running on the mobile robot.
- ACADO Toolkit : Toolkit with MATLAB interference to implement the NMPC routine
- Mobile Eyes [28]: MobileEye was used as the graphical user interface for navigation and monitoring of mobile robot.
- MobileSim [28]: Simulator used for indoor map based navigation to check the applicability of controller for real time experimentation
- Mapper3 [28]: The map of environment for indoor navigation was created by using the Mapper3 tool.
- Motive [69]: Software platform used for the OptiTrack [69] tracking system. It tracks the mobile robot for indoor environment by using trackable markers and provides the position of markers in the tracked environment.

3.5 Overall Control Strategy

The overall control strategy is represented in figure 3.2. NMPC has two major blocks, optimizer and predictor. If the reference pose is given, NMPC controller computes the optimal control command to track reference pose, while considering the cost function

and constraints on the system. NMPC predictor predicts the future output states and control inputs over the given prediction horizon using the current robot state and control sequence, while considering the system model and prediction horizon. The optimizer optimizes the cost function to compute the control input sequence using the predicted output states, reference pose and control input, while considering the constraints.

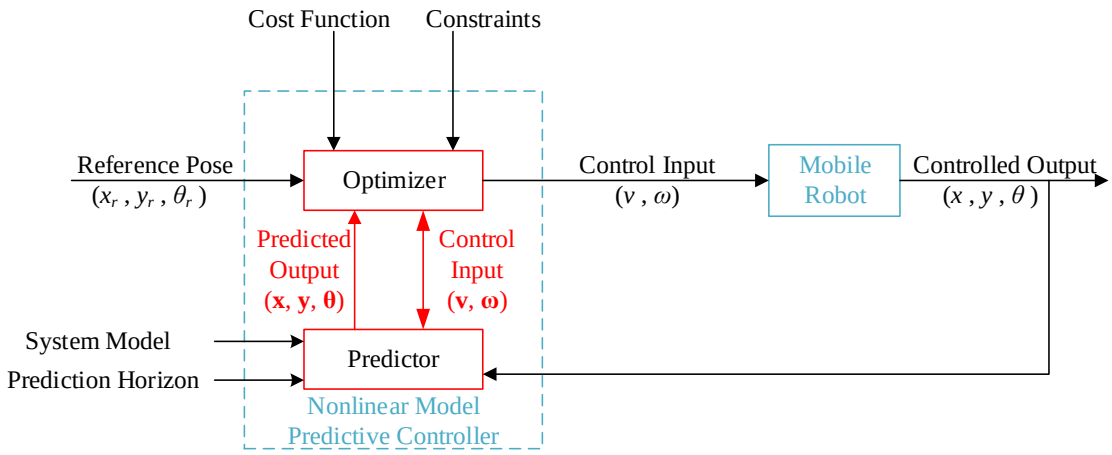


Figure 3.2: Overall Control Strategy

3.5.1 Indoor Navigation based on Map Based Localization

The overall system architecture for indoor navigation of mobile robot using map based localization is shown in figure 3.3. The system has four major blocks.

- *Server:* ARNL (Advanced Robotics Navigation and Localization) server runs on mobile robot (for real time experiment) and on MobileSim (for simulation). This is the modified version of ARNL server, which enables it to receive the velocity command through client which connects to MATLAB through TCP/IP connector. It drives the mobile robot on the basis of velocity command (v, ω)

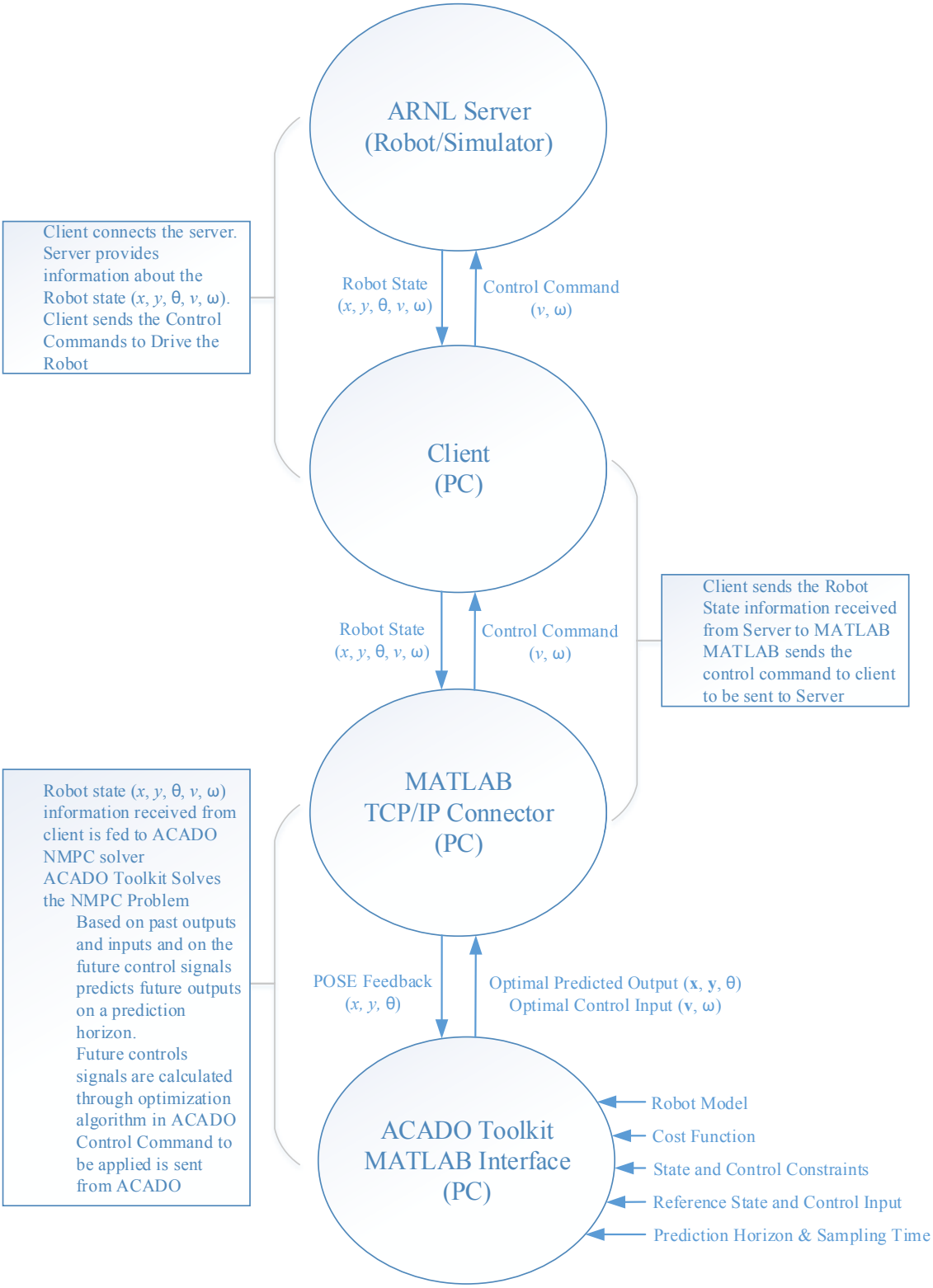


Figure 3.3: Overall Control Strategy for Indoor Navigation using map based localization

received from client. Localization of mobile robot is also done by ARNL server. It uses the map data and laser range finder to perform localization task. It provides the robot status $(x, y, \theta, v, \omega)$ information to client.

- *Client:* ArNetworking Client connects with ARNL server through wireless connection. This is the modified version which enables MATLAB to connect to ARNL server. It receives the robot status $(x, y, \theta, v, \omega)$ from ARNL server and sends command velocity (v, ω) to server which drives mobile robot. This velocity command is being received from MATLAB TCP/IP connector. To this end, it acts as server, which receives velocity commands and sends the robot status to MATLAB. Therefore, the client acts as a middle layer, which provides connection between MATLAB and ARNL Server.

- *MATLAB:* This is the core of processing part of the navigation process. It receives the robot status $(x, y, \theta, v, \omega)$ from client. It passes the pose (x, y, θ) information to NMPC controller for the evaluation of velocity command, which is then passed to client. NMPC is implemented through ACADO toolkit, which has a MATLAB interface. After solving NMPC problem by ACADO, MATLAB receives optimal predicted states $(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the given prediction horizon. These are the vectors having length equal to the number of prediction steps. From the optimal control sequence, the first set is selected as velocity command and is fed to the robot. All the plotting and storing of results are also done here.

- *ACADO Toolkit:* ACADO toolkit solves the fast NMPC by taking the pose feedback (x, y, θ), mobile robot model, state and control constraints, cost function, reference state and control inputs, prediction horizon and sampling time. ACADO toolkit runs the online optimization and gives optimal predicted states

$(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the specified prediction horizon.

Figure 3.4 shows the map of the environment for indoor navigation. The map is constructed through Laser range finder by using Mapper3 software. This is the map of Intelligent System Lab in Faculty of Engineering and Applied Sciences. The black lines represent the walls or boundaries of the objects scanned by laser. The clay colored area represents the forbidden area for the mobile robot. The central white region is the area where mobile robot can navigate.

Figure 3.4: Map of the environment for Indoor Navigation

3.5.2 Indoor Navigation based on Tracking System

The overall system architecture for indoor navigation of mobile robot using tracking system is shown in figure 3.5. The system has four major blocks.

- *Server:* ARNL (Advanced Robotics Navigation and Localization) server runs on mobile robot (for real time experiment). This is the modified version of ARNL server, which enables it to receive the velocity command through client which connects to MATLAB through TCP/IP connector. It drives the mobile robot on the basis of velocity command (v, ω) received from client. It provides the robot speed (v, ω) information to client. Now, the localization data is not provided by ARNL server, instead it is obtained through tracking system.

- *Client:* ArNetworking Client connects with ARNL server through wireless connection. This is the modified version, which enables MATLAB to connect to the ARNL server. It receives the robot speed (v, ω) from ARNL server and sends command velocity (v, ω) to server, which drives the mobile robot. This velocity command is being received from MATLAB TCP/IP connector. To this end, it acts as server, which receives velocity commands and sends the robot status to MATLAB. Therefore, the client acts as a middle layer, which provides connection between MATLAB and ARNL Server.

- *Tracking System:* The software platform used fot OptiTrack tracking system is Motive. Motive broadcasts the tracked markers data. These markers are placed on the mobile robot, so data obtained corresponds the mobile robot position in the tracked environment. It uses the NatNet SDK, which is a Client/Server networking for sending and receiving NaturalPoint data across networks. Server is the host machine, where Motive is running and broadcasting the tracked position data of rigid body and markers. Client is the local machine where data

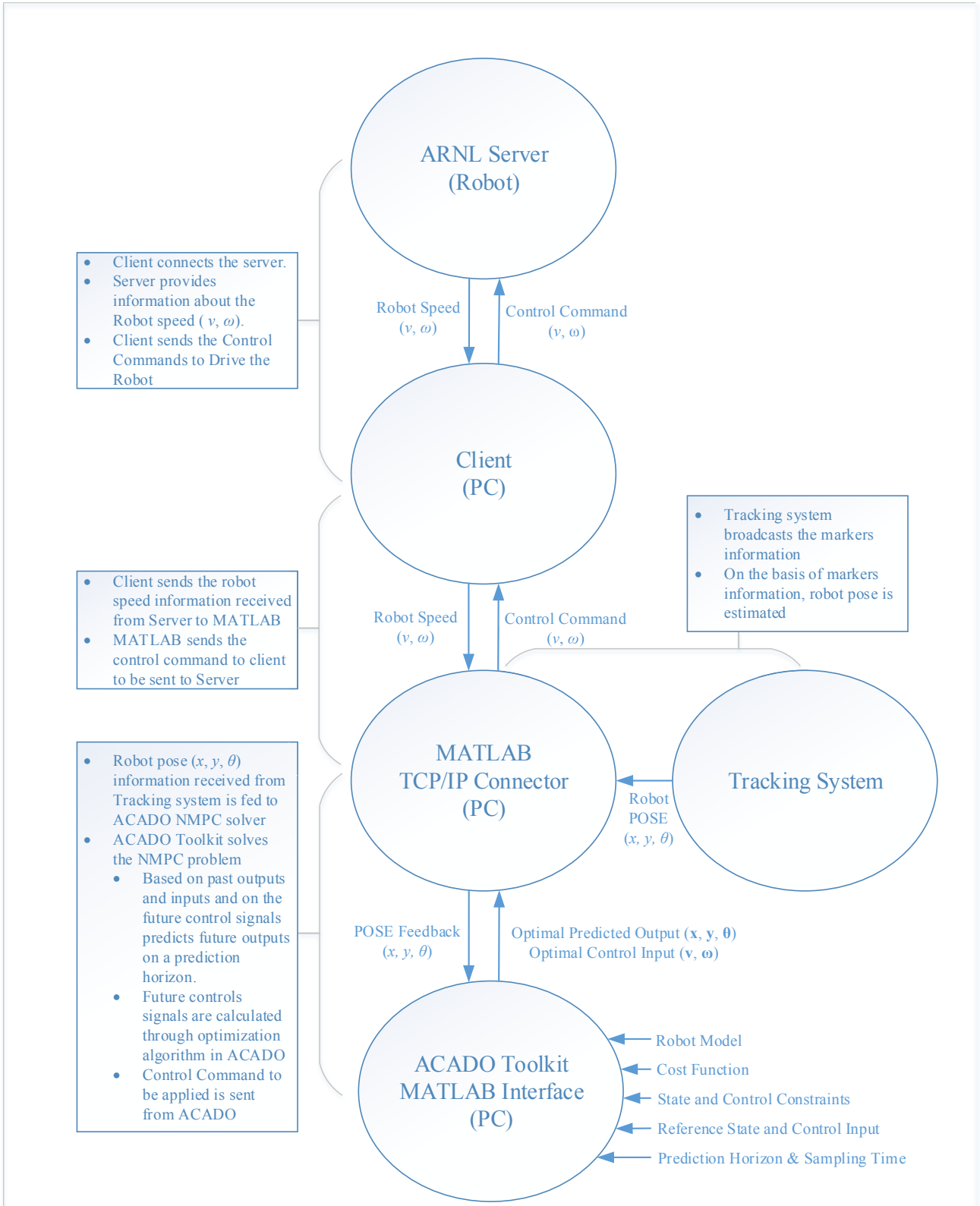


Figure 3.5: Overall Control Strategy for Indoor Navigation using tracking

is obtained in MATLAB. Client is connected with MATLAB to get the markers information. From this markers position data, pose (x, y, θ) of mobile robot is evaluated.

- *MATLAB:* This is the core processing part of the navigation process. It receives the robot pose (x, y, θ) from tracking system through NatNet SDK and mobile robot speed (v, ω) from client. It is necessary to mention that in this method robot pose is not obtained through map based localization, rather it is obtained through tracking. MATLAB passes the POSE (x, y, θ) information to NMPC controller for the evaluation of velocity command, which is then passed to client. NMPC is implemented through ACADO toolkit, which has a MATLAB interface. After solving NMPC problem by ACADO, MATLAB receives optimal predicted states $(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the given prediction horizon. From the optimal control sequence, the first set is selected as velocity command and is fed to the robot. All the plotting and storing of results are also done here.

- *ACADO Toolkit:* ACADO toolkit solves the fast NMPC by taking the POSE feedback (x, y, θ), mobile robot model, state and control constraints, cost function, reference state and control inputs, prediction horizon and sampling time. ACADO toolkit runs the online optimization and gives optimal predicted states $(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the prediction horizon.

The tracked environment is shown in figure 3.6. The environment is of Intelligent System Lab in Faculty of Engineering and Applied Sciences. Four cameras have been used to track the mobile robot.

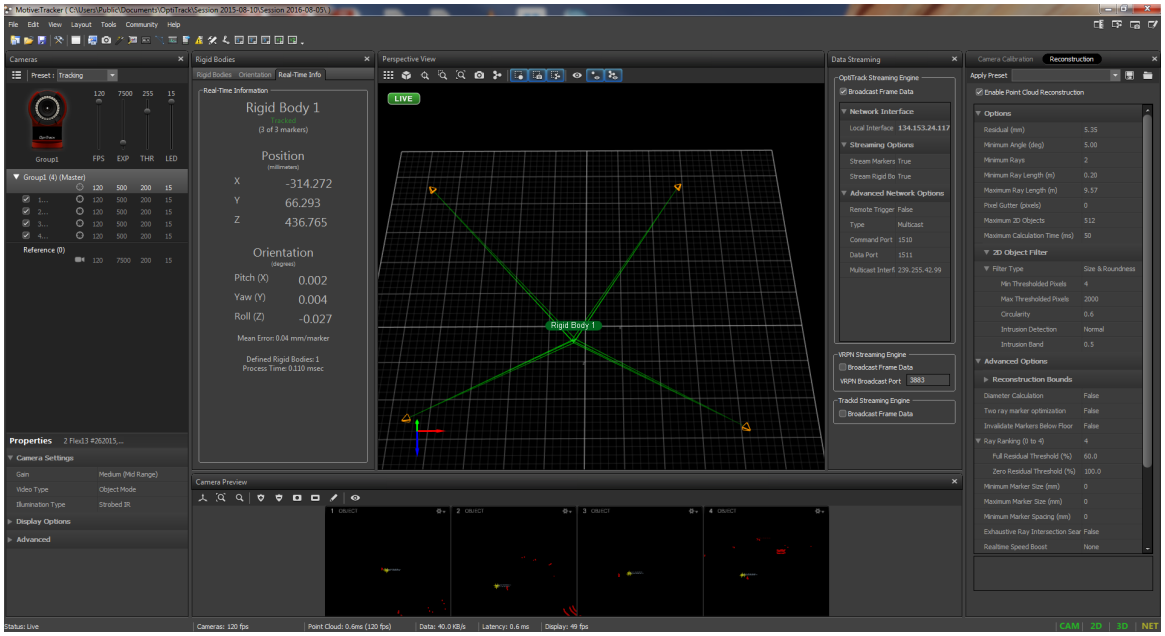


Figure 3.6: Motive (Software) for OptiTrack motion capture system

3.5.3 Outdoor Navigation based on DGPS

The overall system architecture for outdoor navigation of mobile robot is shown in figure 3.7. The system has four major blocks.

- *Server:* MOGS (Mobile robot Outdoor Guidance System) runs on Mobile robot (for real time experiment) and on MobileSim (for simulation). This is the modified version of MOGS server, which enables it to receive the velocity command through client which connects to MATLAB through TCP/IP connector. It drives the mobile robot on the basis of velocity command (v, ω) received from client. MOGS performs the localization task based on DGPS. It uses the map of the environment to navigate, positioning of the mobile robot is determined through DGPS. It provides the robot status $(x, y, \theta, v, \omega)$ information to client. Pose is calculated from the GPS, robot gyro/IMU and odometry sensors.

- *Client:* ArNetworking Client connects with MOGS server. It receives the Robot status $(x, y, \theta, v, \omega)$ from MOGS server and sends Command velocity (v, ω) to

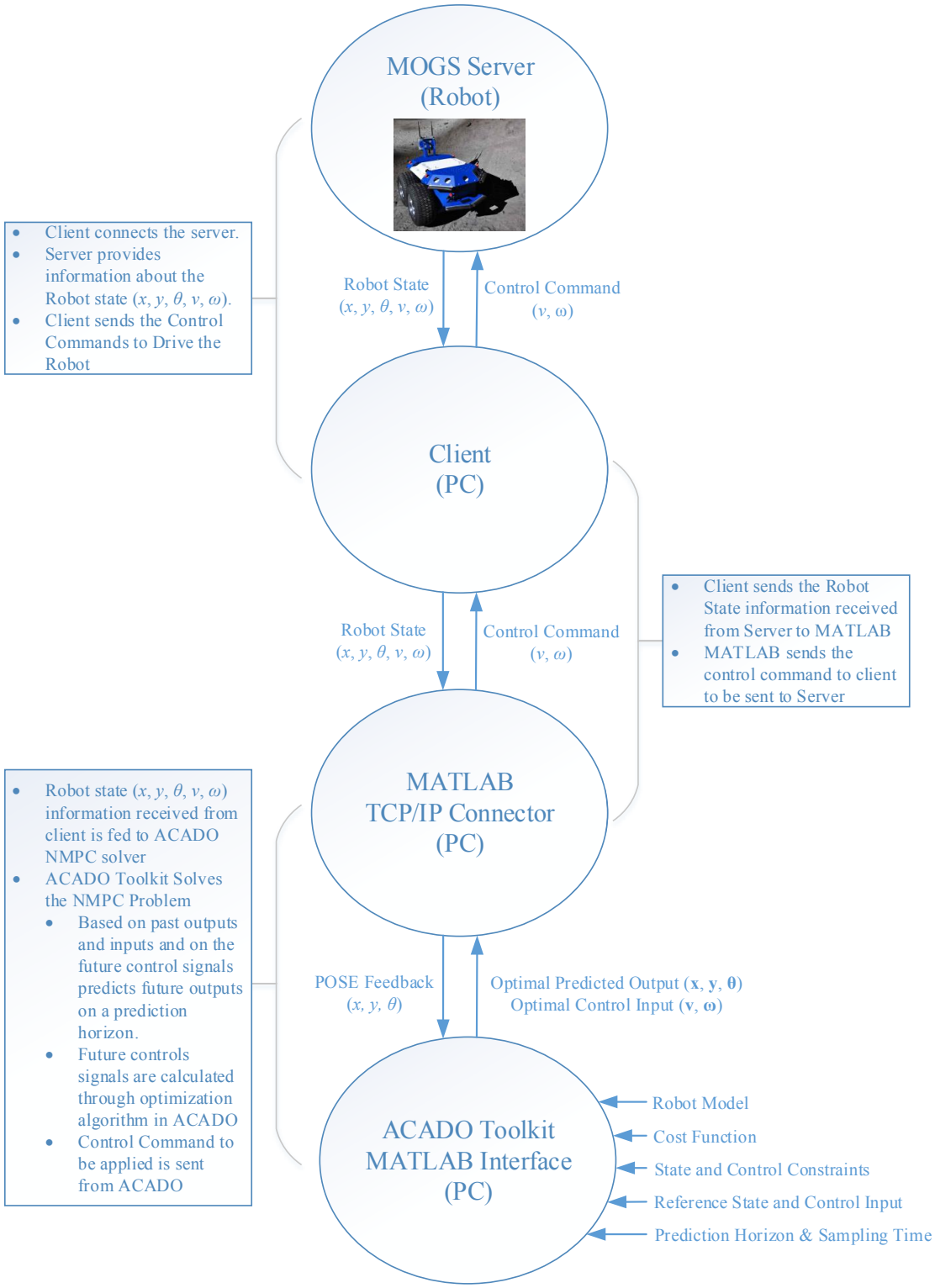


Figure 3.7: Overall Control Strategy for Outdoor Navigation

server, which drives mobile robot. This velocity command is being received from MATLAB TCP/IP connector. To this end, it acts as server, which receives velocity commands and sends the robot status to MATLAB. Therefore, the client acts as a middle layer, which provides connection between MATLAB and MOGS Server.

- *MATLAB:* This is the core of processing part of the navigation process. It receives the robot status $(x, y, \theta, v, \omega)$ from client. It passes the POSE (x, y, θ) information to NMPC controller for the evaluation of velocity command, which is then passed to client. NMPC is implemented through ACADO toolkit, which has a MATLAB interface. After solving NMPC problem by ACADO, MATLAB receives optimal predicted states $(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the given prediction horizon. From the optimal control sequence, the first set is selected as velocity command and is fed to the robot. All the plotting and storing of results are also done here.

- *ACADO Toolkit:* ACADO toolkit solves the fast NMPC by taking the POSE feedback (x, y, θ), mobile robot model, state and control constraints, cost function, reference state and control inputs, prediction horizon and sampling time. ACADO toolkit runs the online optimization and gives optimal predicted states $(\mathbf{x}, \mathbf{y}, \theta)$ and optimal control input $(\mathbf{v}, \omega)$ over the prediction horizon.

3.5.4 Obstacle Avoidance

In this research work, the obstacles considered are static and their location is pre-known in the mapped environment. As NMPC is well able to cope with the constrained optimization problem, so this idea has been exploited for obstacle avoidance problem. The controller considers these obstacles as the constraints in optimization

problem. The objective is to minimize

$$\sqrt{(x - x_o)^2 + (y - y_o)^2} \geq r_o \tag{3.14}$$

where (x_o, y_o) represents the location of the obstacle in the environment, (x, y) depicts the current location of mobile robot and r_o represents the radius of the obstacle. Here, the radius of the obstacle also considers the safe margin for the mobile robot navigation. So, the mobile robot keeps the minimum safe distance from the obstacle, while avoiding that obstacle. r_o includes the radius of the obstacle, radius of the environment and minimum safe distance from the obstacle.

Chapter 4

Point Stabilization

About this Chapter: This chapter presents the point stabilization results for wheeled mobile robot. The chapter has been divided into four sections. First section presents the results for indoor navigation using map based localization. Second section discusses the indoor navigation results using tracking system. Point stabilization results for outdoor navigation have been presented in third section. Last section presents the point stabilization results with obstacle avoidance.

4.1 Indoor Navigation based on Map based Localization

4.1.1 Problem Formulation

In the mapped environment, the robot starts from the initial pose

$$\mathbf{x}_i = \begin{bmatrix} 0.338 & 0.990 & 0 \end{bmatrix}^T \text{(m, m, rad)}$$

and desired goal is to stabilize at the pose

$$\mathbf{x}_r = \begin{bmatrix} 4.735 & 3.620 & 0 \end{bmatrix}^T \text{ (m, m, rad)}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

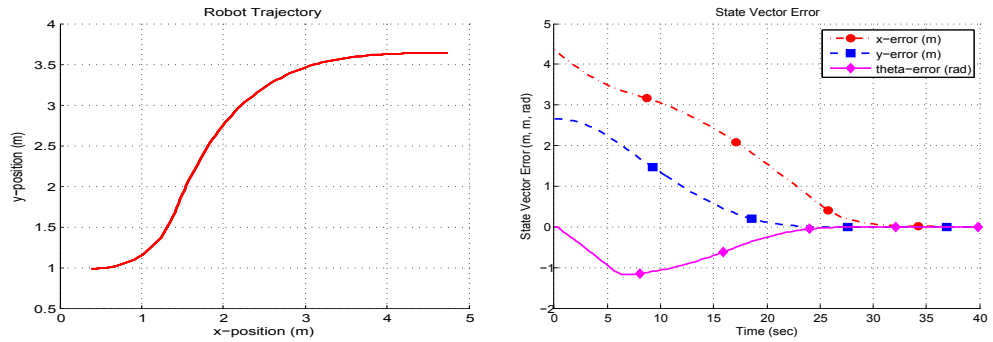
$$v = \begin{bmatrix} -0.3 & 0.3 \end{bmatrix} \quad \text{m/s}$$
$$\omega = \begin{bmatrix} -0.3 & 0.3 \end{bmatrix} \quad \text{rad/s}$$

In order to achieve the above mentioned task, the controller update time step is chosen to be 0.3 s with number of prediction steps equal to 10. Therefore, the prediction horizon time is 3 s. The weight matrices $\mathbf{Q}$ and $\mathbf{R}$ of the cost function (3.11) are chosen as diagonal matrices with diagonal elements defined as (1, 10, 1) and (2, 2) for $\mathbf{Q}$ and $\mathbf{R}$, respectively.

4.1.2 Simulation Results

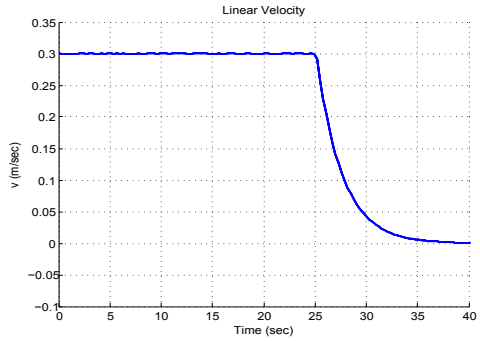
Series of simulations were carried out in MATLAB and MobileSim (Simulator). The former is used to test the proper functionality of controller, while later is used to evaluate the applicability for the mobile robot for real time experimentation. MATLAB simulations do not use map and robot states are simulated over the prediction horizon. MobileSim simulations use the map to perform localization task and controller receives the state feedback same as it does for real-time scenario.

Figure 4.1 shows the MATLAB simulation results. It can be seen from Figure 4.1(a), robot trajectory, that robot started from position (0.338 m, 0.990 m) and stopped at (4.74 m, 3.62 m). Figure 4.1(b) shows that error state vector $\mathbf{x}_e$ converges to null

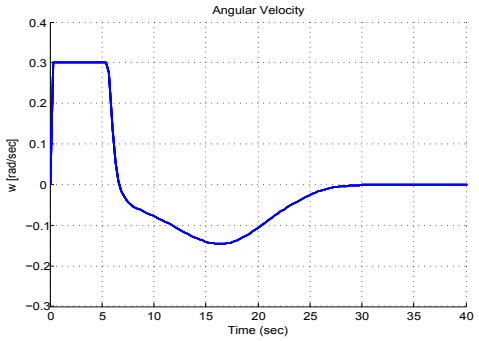


(a) Mobile Robot Trajectory

(b) State Vector Error



(c) Linear Velocity Profile

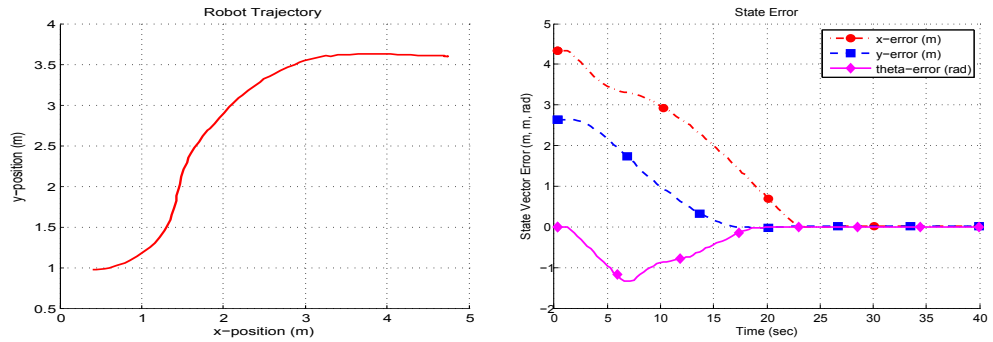


(d) Angular Velocity Profile

Figure 4.1: Simulation(ACADO) results of point stabilization for indoor navigation using map based localization

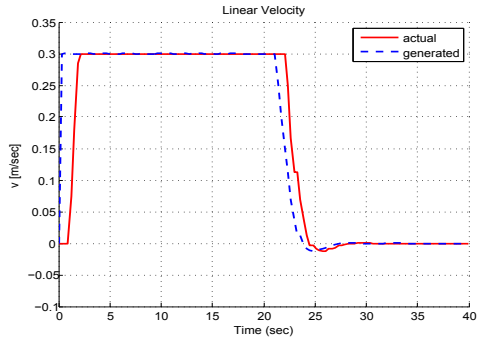
vector. The steady state absolute error in x-position is 0.0013 m, y-position is 0.0064 m, and theta is 0.0002 rad. This leads to overall error state vector magnitude 0.0065. Figure 4.1(c) and 4.1(d), shows that both linear and angular velocities are always within the specified range as mentioned in the constraints to the controller. These results achieve the proper working of the controller developed for this problem.

Figure 4.2 shows the simulation results from MobileSim. It can be seen from Figure 4.2(a), robot trajectory, that robot started from position $(0.396\text{ m}, 0.974\text{ m})$ and stopped at $(4.726\text{ m}, 3.610\text{ m})$. Figure 4.2(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state absolute error in x-position is 0.0090 m, y-position is 0.0010 m, and theta is 0.0 rad. Therefore, overall error state vector magnitude 0.0014. The overall error is more than the ACADO case because of two

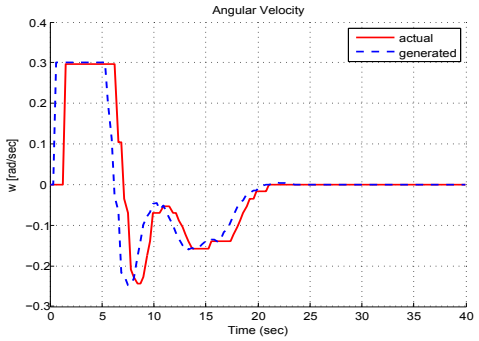


(a) Mobile Robot Trajectory

(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile

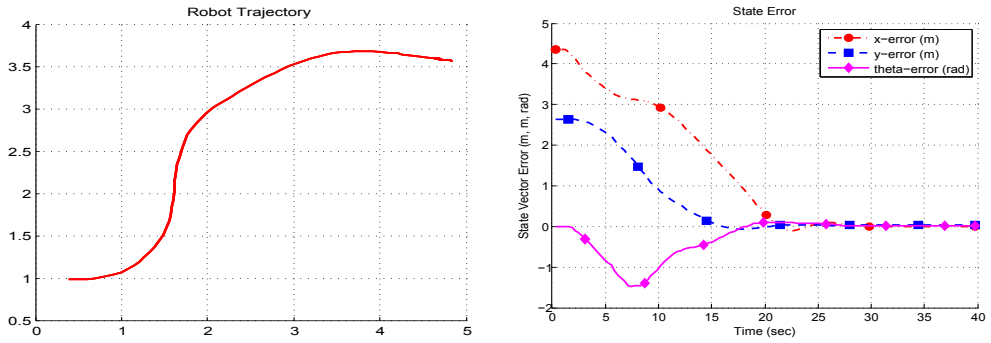
Figure 4.2: Simulation(MobileSim) results of point stabilization for indoor navigation using map based localization

reasons. Firstly, MobileSim uses the localization data by using map, whereas ACADO simulation only uses the predicted states from the controller. Secondly, MobileSim provides the simulation environment with realistic robot model. Figure 4.2(c) shows the linear velocity of mobile robot. It can be noticed, that there is time lag between the robot actual velocity and velocity command generated by the controller. This can be attributed to two reasons: (a) Actual velocity is considered after one sampled time, which the robot achieved when the current generated velocity command is applied (b) Only kinematic model of robot has been considered and there is always inertia associated with the real robot, which is not considered in the kinematic model. Figure 4.2(d) shows angular velocities, actual and generated. It has the same lag as appearing in the linear velocity case. From both figures, it is clear that velocities are

always with in the specified range as mentioned in the constraints to the controller. These simulations lead to the real-time applicability of controller developed for this problem.

4.1.3 Real-Time Experiment

Figure 4.3 shows the real time experimental results run on Seekur Jr. for point stabilization problem for indoor navigation using map based localization.



(a) Mobile Robot Trajectory

(b) State Vector Error

(c) Linear Velocity Profile

(d) Angular Velocity Profile

Figure 4.3: Realtime Experimental results of point stabilization for indoor navigation using map based localization

It can be seen from Figure 4.3(a), robot trajectory, that robot starts from position $(0.389\text{ m}, 0.982\text{ m})$ and stops at $(4.731\text{ m}, 3.587\text{ m})$. Figure 4.3(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state absolute error in x-position

is 0.0040 m, y-position is 0.0330 m, and theta is 0.0175 rad. Therefore, the overall error state vector magnitude is 0.0375. This error is larger than the simulation results. It was expected, because the model of system doesn t use the dynamics of robot and localization data is not accurate. In MobileSim simulation localization is always considered to be accurate, whereas in real-time scenario it can never achieve comparable localization as simulation. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 4.3(c) shows the linear velocity of mobile robot. Again, there is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned above. It can be observed that from time 0 to 20 s, linear velocity command generated by the controller remain constant, but robot actual velocity has spikes in it. This is because of the real time dynamics of robot. Figure 4.3(d) shows the actual angular velocity of robot and angular velocity generated by the controller. It has the same lag as appearing in the linear velocity case. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. These result shows the satisfactory performance of developed controller for the point stabilization problem of mobile robot. Table 4.1 summarizes the steady state errors for robot states.

Steady State Error			
Parameter	**Simulation**		**Real-Time Experiment**
	MATLAB	**MobileSim**	
x-position (m)	0.0013	0.0090	0.0040
y-position (m)	0.0064	0.0010	0.0330
Heading angle (rad)	0.0002	0.0	0.0175

Table 4.1: Steady state errors of point stabilization problem for indoor navigation based on map based localization

4.2 Indoor Navigation based on Tracking System

4.2.1 Problem Formulation

The robot starts from the initial pose

$$\mathbf{x}_i = \begin{bmatrix} 1.1508 & 1.5571 & 0.0195 \end{bmatrix}^T \text{(m, m, rad)}$$

and desired goal is to stabilize at the pose

$$\mathbf{x}_r = \begin{bmatrix} 3.8061 & -2.0550 & -1.5378 \end{bmatrix}^T \text{(m, m, rad)}$$

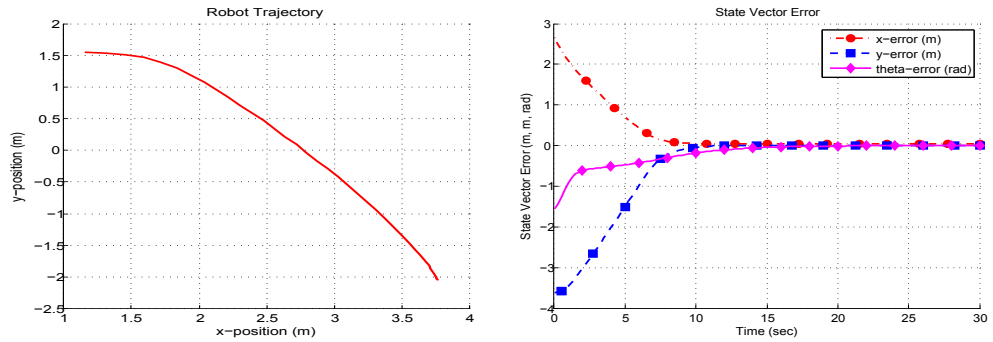
In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

$$v = \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \quad \text{m/s}$$
$$\omega = \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \quad \text{rad/s}$$

The controller update time step is chosen to be 0.25 s with number of prediction steps equal to 20, leading to a prediction horizon time T = 5 s. The weight matrices **Q** and **R** of the cost function (8) are chosen as diagonal matrices with diagonal elements defined as (3, 2, 0.5) and (4, 5) for **Q** and **R**, respectively.

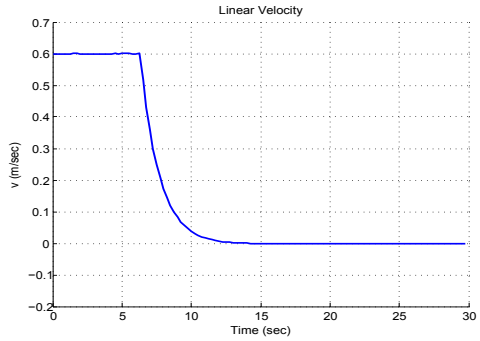
4.2.2 Simulation Results

Series of simulations were carried out in MATLAB using ACADO toolkit. The purpose is to test the proper functionality of controller. MATLAB simulations do not use tracking system and robot states are simulated over the prediction horizon. Figure 4.4 shows the simulation result obtained from ACADO. It can be seen from Figure 4.4(a),

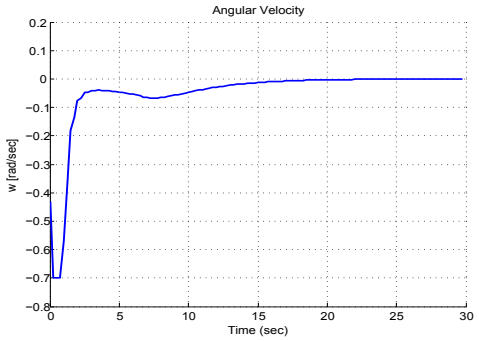


(a) Mobile Robot Trajectory

(b) State Vector Error



(c) Linear Velocity Profile



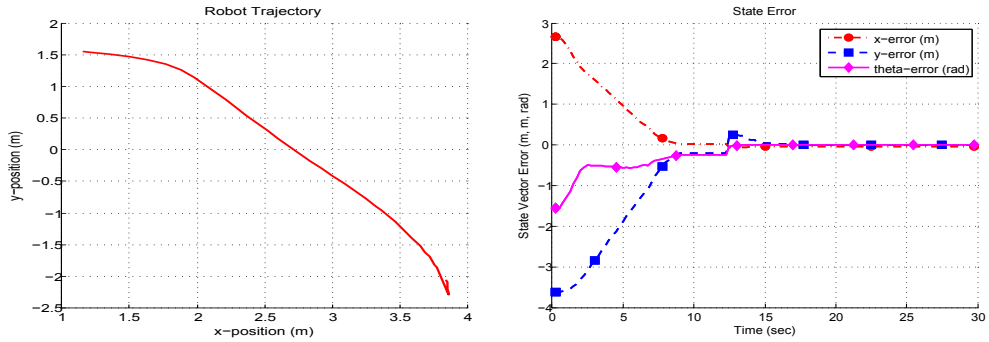
(d) Angular Velocity Profile

Figure 4.4: Simulation results of point stabilization for indoor navigation using tracking system

robot trajectory, that robot starts from position $(1.1508\text{ m}, 1.5571\text{ m})$ and stops at $(3.7660\text{ m}, -2.0571\text{ m})$. Therefore, the steady state absolute error in x-position is 0.0401 m, y-position is 0.0021 m, and theta is 0.0006 rad. So, the overall error state vector magnitude 0.0401. Figure 4.4(b) plots the state errors. It is clear that error state vector $\mathbf{x}_e$ converges to null vector. Figure 4.4(c) and 4.4(d), shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. Though Linear velocity and angular velocities touches the maximum allowed values but never exceeds the limits. So the controller generates the velocity commands under constrained values in order to achieve point stabilization of the mobile robot. This leads to the proper working of the controller developed for this problem.

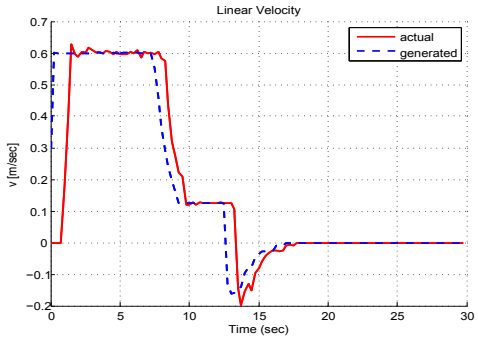
4.2.3 Real-Time Experiment

Figure 4.5 shows the real time experimental results run on Seekur Jr. for indoor navigation using tracking system.

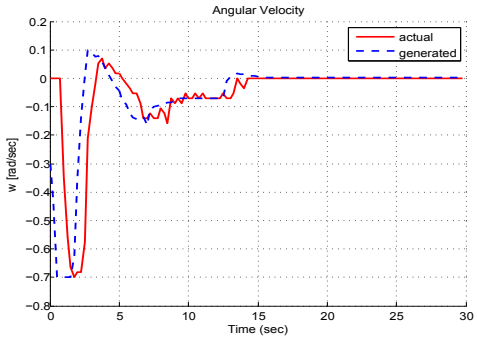


(a) Mobile Robot Trajectory

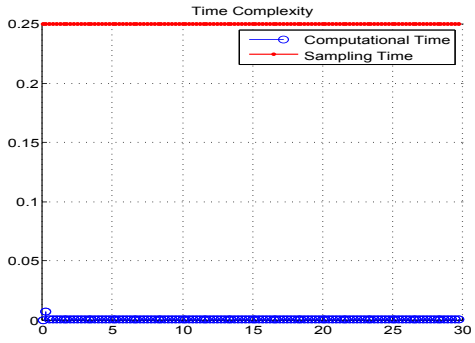
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 4.5: Realtime Experimental results of point stabilization for indoor navigation using tracking system

Figure 4.5(a) shows the robot trajectory, where robot starts from position (1.1508 m, 1.5571 m) and stops at (3.8496 m, −2.0550 m). At the final pose robot adjusts for y-position. This is also clear from figure 4.5(b) where y state error becomes positive and then converges to zero. Figure 4.5(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state absolute error in x-position is 0.0044 m, y-position is 0.0002 m, and theta is 0.0040 rad. This leads to overall error state vector magnitude 0.0437. The overall error state vector is really close to the simulation results as compare to the results obtained by mapped based localization. The reason behind is that, now robot is using being tracked by motion capture system which has more precision as compare to localization achieved through map based localization. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 4.5(c) shows the linear velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in Section 4.1.2. It can be observed that actual linear velocity of the robot momentarily above the maximum specified velocity 0.6 m/s, but velocity generated by controller is still in the limits at that point also. The reason behind this is that, robot can have velocity more than 0.6 m/s. But in order to be in safe region, maximum velocity specified is 0.6 m/s so that robot always follow the velocity command generated by the controller. Figure 4.5(d) shows angular velocities, actual velocity of robot and angular speed generated by the controller. It has the same lag as appearing in the linear velocity case. From both figures, it is clear that velocities are always with in the specified range,for both generated and actual ones, as mentioned in the constraints to the controller. Figure 4.5(e) shows that the computation time, for obtaining the robot status, solving NMPC, evaluating velocity command and saving results, is in the order of fraction of milliseconds. The sampling time selected for this experiment is 0.25 s.

So, there is enough time to apply the velocity command to mobile robot, which is still equivalent to sampling time. These result shows the satisfactory performance of developed controller for the point stabilization problem of mobile robot. Table 4.2 summarizes the absolute values of steady state errors for robot states.

Steady State Error		
Parameter	**Simulation**	**Real-Time Experiment**
x-position (m)	0.0401	0.0044
y-position (m)	0.0021	0.0002
Heading angle (rad)	0.0006	0.0040

Table 4.2: Steady state errors of point stabilization problem for indoor navigation based on tracking system

4.3 Outdoor Navigation based on DGPS

4.3.1 Problem Formulation

The robot starts from the initial pose

$$\mathbf{x}_i = \begin{bmatrix} -2.533 & -87.248 & 2.3911 \end{bmatrix}^T \text{(m, m, rad)}$$

and desired goal is to stabilize at the pose

$$\mathbf{x}_r = \begin{bmatrix} -18.984 & -78.170 & 2.7402 \end{bmatrix}^T \text{(m, m, rad)}$$

These poses correspond to the positioning data obtained through MOGS server on the map of the outdoor environment. In order to move from initial position and stabilize itself at the desired pose, the mobile robot is constrained on its linear and angular

speed, specified as:

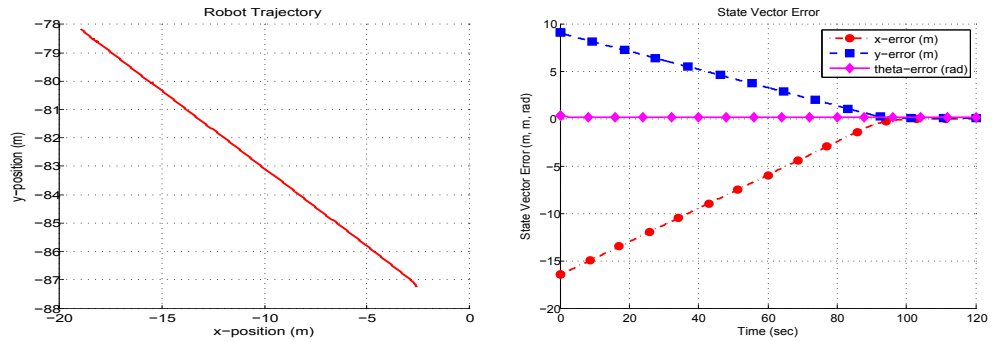
$$v = \begin{bmatrix} -0.3 & 0.3 \end{bmatrix} \quad \text{m/s}$$
$$\omega = \begin{bmatrix} -0.3 & 0.3 \end{bmatrix} \quad \text{rad/s}$$

The controller update time step is chosen to be 0.3 s with number of prediction steps equal to 10. Therefore, prediction horizon time is 3 s. The weight matrices **Q** and **R** of the cost function (8) are chosen as diagonal matrices with diagonal elements defined as (10, 10, 0.1) and (50, 80) for **Q** and **R**, respectively.

4.3.2 Simulation Results

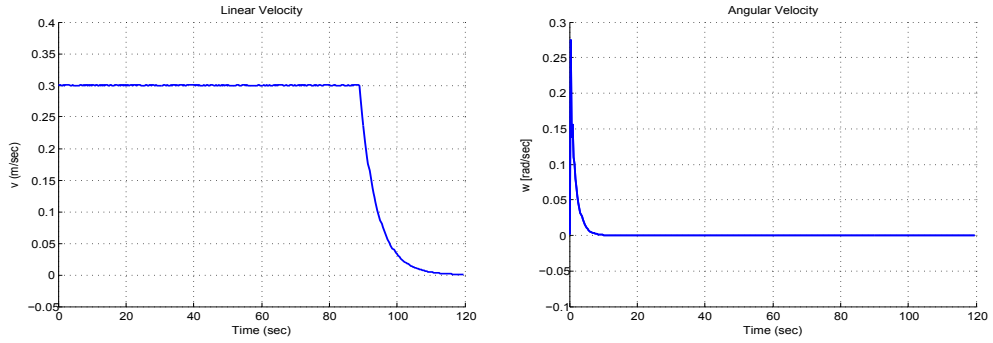
Simulation results are obtained from MATLAB using ACADO. Simulations do not use GPS based localization. So, in order to get mobile robot state feedback, robot states are simulated over the prediction horizon. Simulation results are shown in Figure 4.6, which includes (a) mobile robot trajectory, (b) state vector error, (c) linear velocity and (d) angular velocity.

It can be seen from Figure 4.6(a) that robot starts from position (-2.533 m, -87.248 m) and stops at (-18.9842 m, -78.1747 m). Therefore, the absolute values of steady state error in x-position is 0.0002 m, y-position is 0.0047 m, and the theta error is 0.0936 rad. So, the overall error state vector magnitude is 0.0937. Figure 4.6(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. Figure 4.6(c) and 4.6(d), shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. These results achieve the proper working of the controller developed for this problem.



(a) Mobile Robot Trajectory

(b) State Vector Error



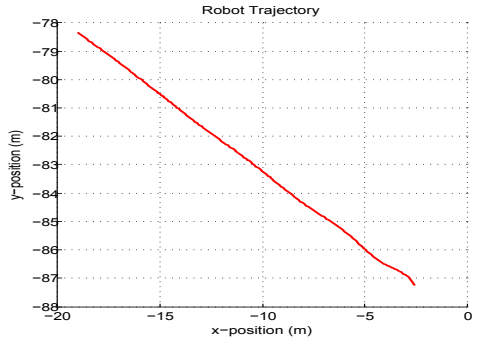
(c) Linear Velocity Profile

(d) Angular Velocity Profile

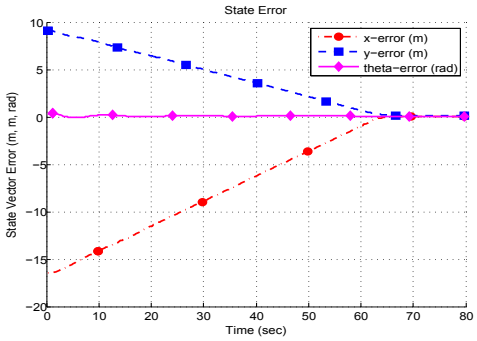
Figure 4.6: Simulation results of point stabilization for outdoor navigation

4.3.3 Real-Time Experiment

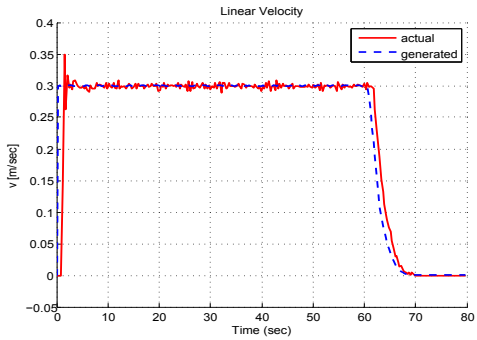
Figure 4.7 shows the real-time experimental results run on Seekur Jr. for outdoor navigation. It can be seen from figure 4.7(a), robot trajectory, that robot starts from position $(-2.5330 \text{ m}, -87.2480 \text{ m})$ and stops at $(-19.0640 \text{ m}, -78.3290 \text{ m})$. Figure 4.7(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state absolute error in x-position is 0.0800 m, y-position is 0.1590 m, and theta is 0.0698 rad. This leads to overall error state vector magnitude 0.1912. The overall error state vector is quite high as compare to the simulation results. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 4.7(c) shows the linear velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by



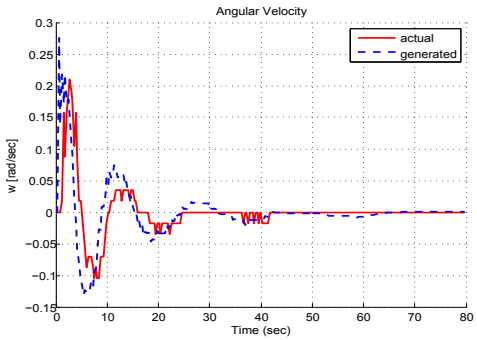
(a) Mobile Robot Trajectory



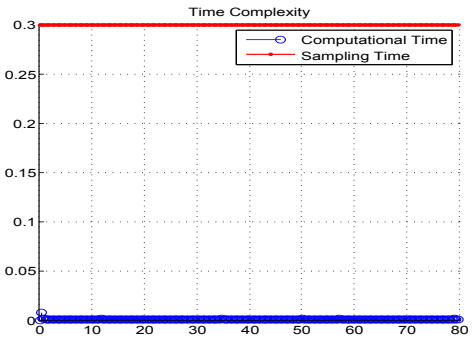
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 4.7: Realtime Experimental results of point stabilization for outdoor navigation

the controller, because of same reason, as mentioned in section 4.1.2. For outdoor rough terrain, there is so frequent spikes in the velocity, but the mean value is same as generated by controller. Figure 4.7(d) shows angular velocities, actual velocity of robot and angular speed generated by the controller. It has the same lag as appearing in the linear velocity case. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. Figure 4.7(e) shows that the computation time for obtaining the robot status, solving NMPC, evaluating velocity command and saving results is in order of milliseconds whereas sampling time selected for this experiment is 0.3 s. So, there is enough time to apply the velocity command to mobile robot which is still equal to sampling time. These result shows the satisfactory performance of developed controller for the point stabilization problem of mobile robot. Table 4.3 summarizes the absolute values of steady state errors for robot states.

Steady State Error		
Parameter	**Simulation**	**Real-Time Experiment**
x-position (m)	0.0002	0.0800
y-position (m)	0.0047	0.1590
Heading angle (rad)	0.0936	0.0698

Table 4.3: Steady state errors of point stabilization problem for outdoor navigation based on DGPS

4.4 Obstacle Avoidance

4.4.1 Problem Formulation

The robot starts from the initial pose

$$\mathbf{x}_i = \begin{bmatrix} 0.9782 & 1.5937 & 0.0779 \end{bmatrix}^T \text{(m, m, rad)}$$

and desired goal is to stabilize at the pose

$$\mathbf{x}_r = \begin{bmatrix} 3.8061 & -2.0550 & -1.5378 \end{bmatrix}^T \text{(m, m, rad)}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

$$\begin{aligned} v &= \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \quad \text{m/s} \\ \omega &= \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \quad \text{rad/s} \end{aligned}$$

While moving from initial position to final goal position, the robot is also supposed to avoid the obstacles in its path. The obstacles are positioned at:

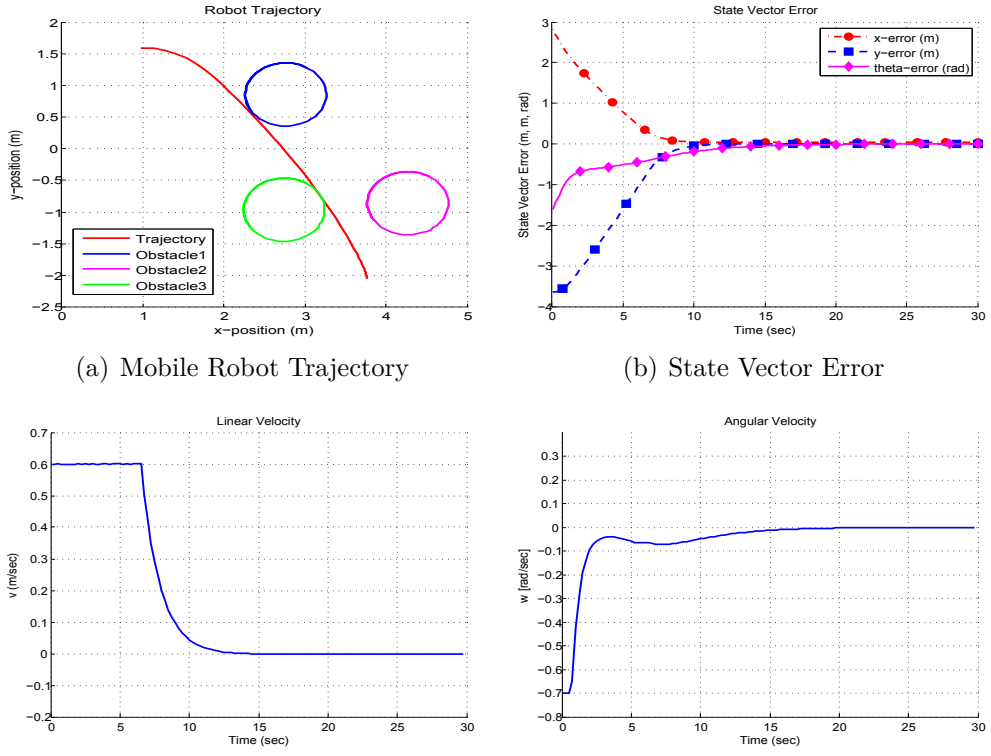
$$\begin{aligned} (x_{o_1}, y_{o_1}) &= (2.763 \text{ m}, 0.863 \text{ m}) \\ (x_{o_2}, y_{o_2}) &= (4.270 \text{ m}, -0.862 \text{ m}) \\ (x_{o_3}, y_{o_3}) &= (2.744 \text{ m}, -0.964 \text{ m}) \end{aligned}$$

In order to achieve the above mentioned task, the controller update time step is chosen to be 0.25 s with number of prediction steps equal to 20. Therefore, the prediction horizon time is equal to 5 s. The weight matrices $\mathbf{Q}$ and $\mathbf{R}$ of the cost function (3.11)

are chosen as diagonal matrices with diagonal elements defined as (3, 2, 0.5) and (4, 5) for **Q** and **R**, respectively. The problem is similar to the problem stated in Section 4.2, but now there are obstacles in the robot path which it needs to avoid. All the coordinate points correspond to the locations in the tracked environment.

4.4.2 Simulation Results

Simulations are run in MATLAB using ACADO toolkit. The purpose is to test the proper functionality of controller. Simulations do not use tracking system and robot states are simulated over the prediction horizon. Results are shown in figure 4.8, including (a) mobile robot trajectory, (b) state vector error, (c) linear velocity and (d) angular velocity.



(a) Mobile Robot Trajectory

(b) State Vector Error

(c) Linear Velocity Profile

(d) Angular Velocity Profile

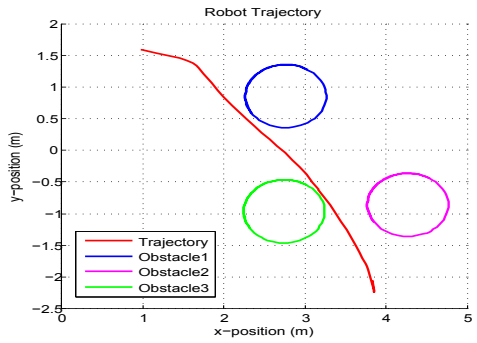
Figure 4.8: Simulation results of point stabilization with obstacle avoidance

It can be seen from figure 4.8(a) that robot starts from position (0.9782m, 1.5937m) and stops at (3.8061m, -2.0550m) and avoids the above mentioned obstacles at the specified locations. Figure 4.8(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The absolute value of steady state error in x-position is 0.0408 m, y-position is 0.0021 m, and the theta error is 0.0006 rad. This leads to overall error state vector magnitude 0.0408. Figure 4.8(c) and 4.8(d), clearly shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. These results conclude that leads the controller developed for this problem is working, properly.

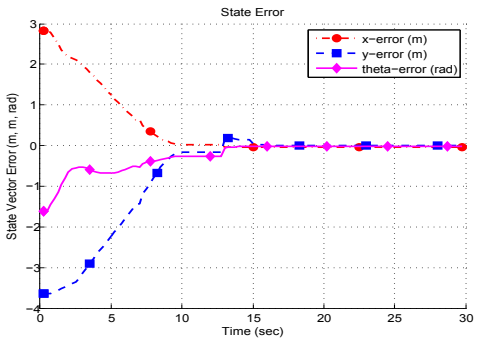
4.4.3 Real-Time Experiment

Figure 4.9 shows the real time experimental results run on Seekur Jr. for point stabilization in the presence of obstacles.

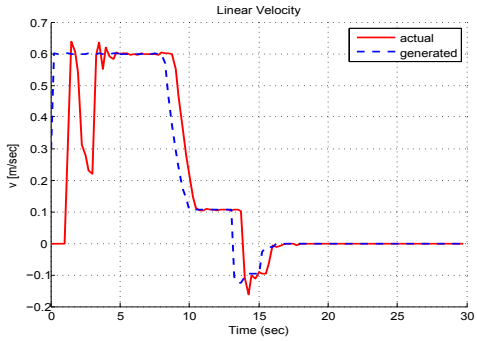
It can be seen from figure 4.9(a), robot trajectory, that robot starts from position $(0.9782\text{m}, 1.5937\text{m})$ and stops at $(3.8393\text{m}, -2.0539\text{m})$. While moving from initial pose to final pose, robot also avoids the obstacles at the above mentioned locations. In the plot obstacles are shown, it is important to mention here that these circles includes the radius of circle and robot and safe margin for navigation. Figure 4.9(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state absolute error in x-position is 0.0332 m, y-position is 0.0011 m, and theta is 0.0216 rad. This leads to overall error state vector magnitude 0.0396. The overall state vector error is really close to the simulation results. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 4.9(c) shows the linear velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in section 4.1.2. It can be observed that actual linear velocity of



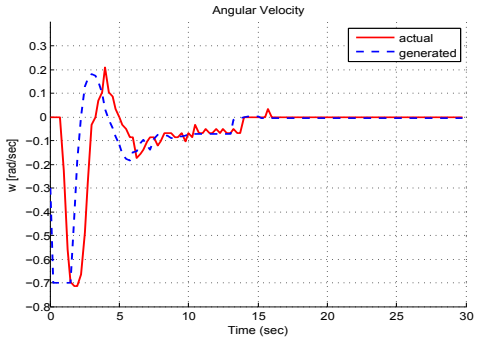
(a) Mobile Robot Trajectory



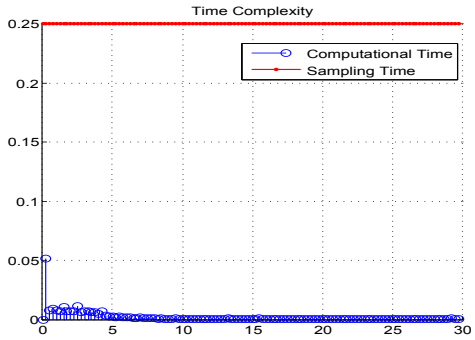
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 4.9: Realtime Experimental results of point stabilization with obstacle avoidance

the robot is momentarily above the maximum specified velocity 0.6 m/s, but velocity generated by controller is still in the limits at that point also. The reason behind this is same as mentioned in section 4.1.2. When robot sees the first obstacle at that moment, there is dip in actual velocity of robot as robot slows down and slightly

change the orientation as seen in figure 4.9(b) also where theta error oscillate at that time step. Figure 4.9(d) shows angular velocities, actual velocity of robot and angular speed generated by the controller. It has the same lag as appearing in the linear velocity case. From both figures it is clear that velocities are always with in the specified range,for both generated and actual ones, as mentioned in the constraints to the controller. Figure 4.9(e) shows that the computation time, for obtaining the robot status, solving NMPC, evaluating velocity command and saving results, is in order of fraction of milliseconds. The sampling time selected for this experiment is 0.25 s. So, there is enough time to apply the velocity command to mobile robot which is still equal to sampling time. There are obstacles present and specified as the constraints for the optimal control problem. Therefore, computational time is more than that of section 4.2 around the location of obstacles, which can been seen from the initial spikes. These result shows the satisfactory performance of developed controller for the point stabilization with obstacle avoidance problem of mobile robot. Table 4.4 summarizes the absolute values of steady state errors for robot states.

Steady State Error		
Parameter	**Simulation**	**Real-Time Experiment**
x-position (m)	0.0408	0.0332
y-position (m)	0.0021	0.0011
Heading angle (rad)	0.0006	0.0216

Table 4.4: Steady state errors of point stabilization problem with obstacle avoidance

Chapter 5

Trajectory Tracking

About this Chapter: This chapter presents the trajectory tracking results for non-holonomic wheeled mobile robot. The chapter has been divided in four sections. First section presents the trajectory tracking results for indoor navigation based on map based localization. Second section discusses the indoor navigation results based on tracking system. Trajectory tracking results for outdoor navigation have been presented in third section. Last section presents the trajectory tracking results in the presence of obstacles.

5.1 Indoor Navigation based on Map based Localization

5.1.1 Problem Formulation

The reference trajectory is a circular trajectory defined by:

$$x_r(t) = 2.66 + 1.5 * \sin(0.133 * t) \text{ m}$$

$$y_r(t) = 2.44 + 1.5 * \cos(0.133 * t) \text{ m}$$

$$v_r = 0.2 \text{ m/s}$$

$$\omega_r = -0.133 \text{ rad/s}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

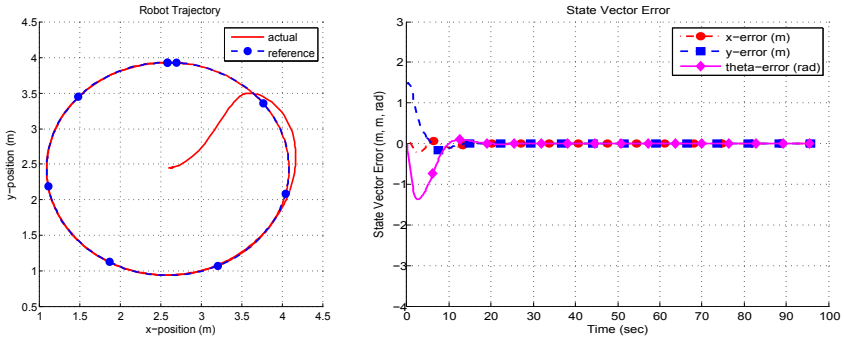
$$v = \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \text{ m/s}$$
$$\omega = \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \text{ rad/s}$$

The controller update time step is chosen to be 0.2 s with number of prediction steps equal to 20, leading to a prediction horizon time 4 s. The weight matrices **Q** and **R** of the cost function (3.11) are chosen as diagonal matrices with diagonal elements defined as (30, 30, 1) and (50, 40) for **Q** and **R**, respectively.

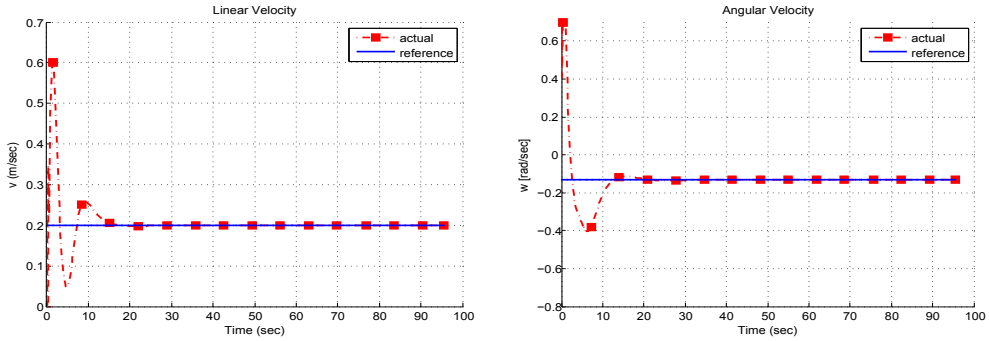
5.1.2 Simulation Results

Series of simulations were carried out in MATLAB and MobileSim (Simulator). The former is used to test the proper functionality of controller, while later is used to evaluate the applicability for the mobile robot for real time experimentation. MATLAB simulations do not use map and robot states are simulated over the prediction horizon. MobileSim simulations use the map to perform localization task and controller receives the state feedback same as it does for real-time scenario.

Figure 5.1 shows the MATLAB simulation results. It can be seen from Figure 5.1(a),



(a) Mobile Robot Trajectory (b) State Vector Error



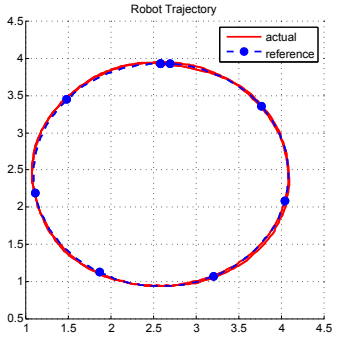
(c) Linear Velocity Profile (d) Angular Velocity Profile

Figure 5.1: Simulation(ACADO) results of trajectory tracking for indoor navigation using map based localization

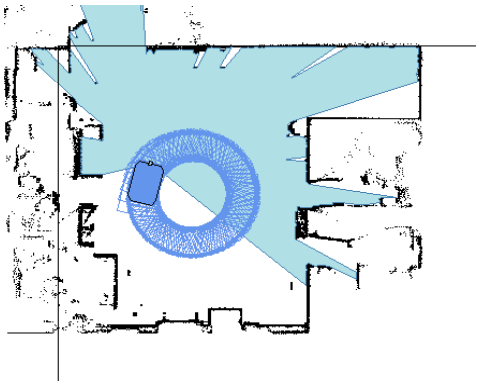
robot trajectory, that robot started from position (2.59 m, 2.45 m) and makes a circle of radius 1.5 m centered at (2.585 m, 2.436 m). Figure 5.1(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The root mean square (rms) values of steady state error in x-position is 0.00020 m, y-position is 0.00047 m, and theta is 0.0016 rad. This leads to overall error state vector magnitude 0.0016. The steady state mean values with standard deviation for x-position error is 0.00006 ± 0.00019 m, y-position error is 0.00007 ± 0.00047 m, and theta error is -0.00027 ± 0.00150 rad. Figure 5.1(c) and 5.1(d), shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. Both linear and angular velocities converge to reference velocities. The rms value of steady state error in linear velocity is 0.00052 m/s, and for angular velocity it is 0.00054 rad/s.

Overall error control vector magnitude is 0.00075. The steady state mean values with standard deviation for linear velocity error is -0.00050 ± 0.00014 m/s, and angular velocity error is -0.00014 ± 0.00053 rad/s.These results achieve the proper working of the controller developed for this problem.

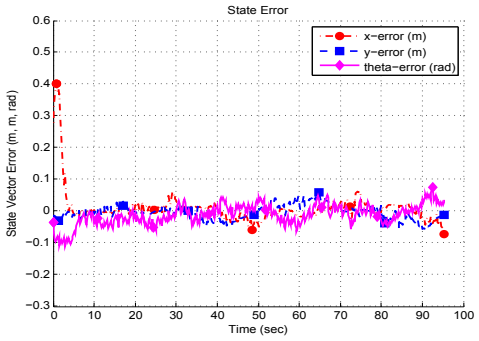
Figure 5.2 shows the simulation results from MobileSim. It can be seen from Figure 5.2(a), robot trajectory, that robot started from position (2.308 m, 3.949 m) and makes a circle of radius 1.5 m centered at (2.585 m, 2.436 m). Figure 5.2(c) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state rms error in x-position is 0.0213 m, y-position is 0.0254 m, and theta is 0.0439 rad. This leads to overall error state vector magnitude 0.0551. The steady state mean values with standard deviation for x-position error is -0.0040 ± 0.0210 m, y-position error is -0.0027 ± 0.0253 m, and theta error is -0.0366 ± 0.0244 rad. The overall error is more than the ACADO case, because MobileSim provides the simulation environment with realistic robot model. Figure 5.2(d) shows the linear velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in Section 4.1.2. Both generated and actual linear velocities converge to reference velocity. The rms value of steady state error for linear velocity is 0.0185 m/s. Figure 5.2(e) shows angular velocities, actual velocity of robot and angular speed generated by the controller. It has the same lag as appearing in the linear velocity case. The rms value of steady state error for angular velocity is 0.0234 rad/s. The steady state mean values with standard deviation for linear velocity error is 0.0133 ± 0.0129 m/s, and angular velocity error is -0.0209 ± 0.0105 rad/s. From both figures, it is clear that velocities are always with in the specified range as mentioned in the constraints to the controller. These simulations lead to the real-time applicability of controller developed for this problem.



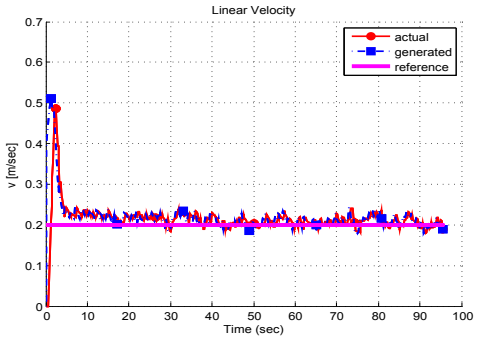
(a) Mobile Robot Trajectory



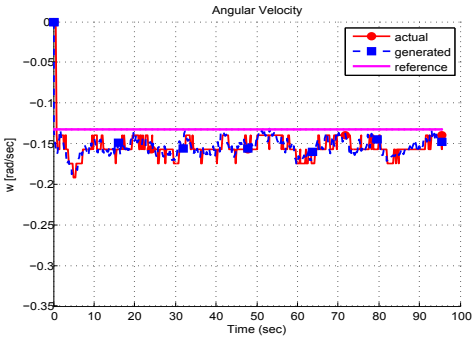
(b) Mobile Robot Trajectory in Mapped Environment



(c) State Vector Error



(d) Linear Velocity Profile

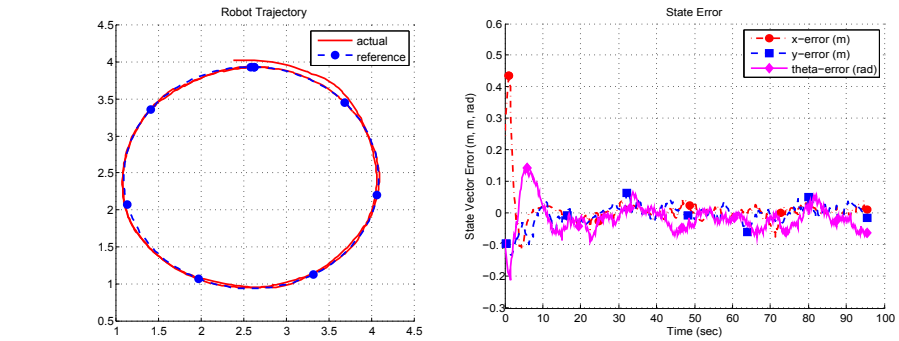


(e) Angular Velocity Profile

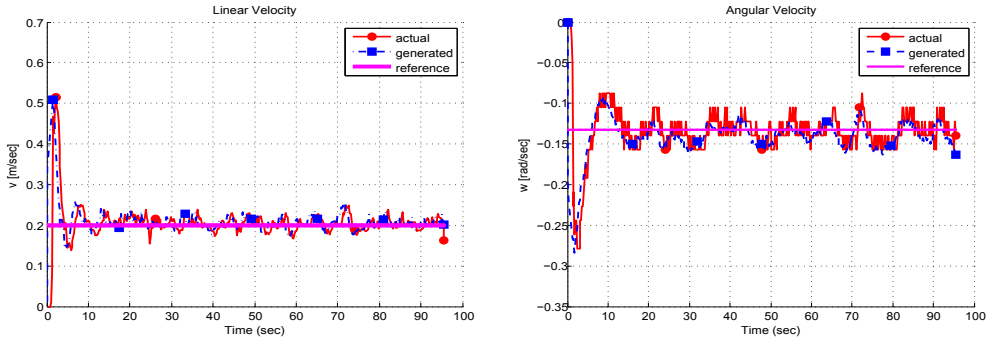
Figure 5.2: Simulation(MobileSim) results of trajectory tracking for indoor navigation using map based localization

5.1.3 Real-Time Experiment

Figure 5.3 shows the real time experimental results run on Seekur Jr. for trajectory tracking problem for indoor navigation based on map based localization.



(a) Mobile Robot Trajectory (b) State Vector Error



(c) Linear Velocity Profile (d) Angular Velocity Profile

Figure 5.3: Realtime Experimental results of trajectory tracking for indoor navigation using map based localization

It can be seen from Figure 5.3(a), robot trajectory, that robot started from position (2.380 m, 4.027 m) and makes a circle of radius 1.5 m centered at (2.585 m, 2.436 m). Figure 5.3(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state rms values of error in x-position is 0.0370 m, y-position is 0.0454 m, and theta is 0.0574 rad. This leads to overall error state vector magnitude 0.0820. The steady state mean values with standard deviation for x-position error is -0.0042 ± 0.0368 m, y-position error is -0.0071 ± 0.0449 m, and theta error is -0.0412 ± 0.0400 rad. The error is larger than the simulation results. It was expected, because the model of system doesn t use the dynamics of robot. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 5.3(c) shows the linear velocity of mobile robot. Again, there is time lag between

the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in section 4.1.2. Both generated and actual linear velocities converge to reference velocity. The steady state rms error magnitude for linear velocity is 0.0182 m/s. Figure 5.3(d) shows the actual angular velocity of robot and angular velocity generated by the controller. It has the same lag as appearing in the linear velocity case. The steady state rms error magnitude for angular velocity is 0.0178 rad/s. The steady state mean values with standard deviation for linear velocity error is 0.0047 ± 0.0176 m/s, and angular velocity error is 0.00024 ± 0.0178 rad/s. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. These result shows the satisfactory performance of developed controller for the point stabilization problem of mobile robot.

Table 5.1 summarizes the rms values of the steady state errors for the mobile robot states and controls.

Steady State Error			
Parameter	**Simulation**		**Real-Time**
	MATLAB	**MobileSim**	**Experiment**
x-position (m)	0.00020	0.0213	0.0370
y-position (m)	0.00047	0.0254	0.0454
Heading angle (rad)	0.00160	0.0439	0.0574
Linear Velocity (m/s)	0.00052	0.0185	0.0182
Angular Velocity (rad/s)	0.00054	0.0234	0.0178

Table 5.1: Steady state errors of trajectory tracking problem for indoor navigation based on map based localization

5.2 Indoor Navigation using Tracking System

5.2.1 Problem Formulation

The reference trajectory is a circular trajectory defined by:

$$x_r(t) = 0.3301 + 1.5 * \sin(0.15 * t) \text{ m}$$
$$y_r(t) = -0.4489 + 1.5 * \cos(0.15 * t) \text{ m}$$
$$v_r = 0.225 \text{ m/s}$$
$$\omega_r = -0.15 \text{ rad/s}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

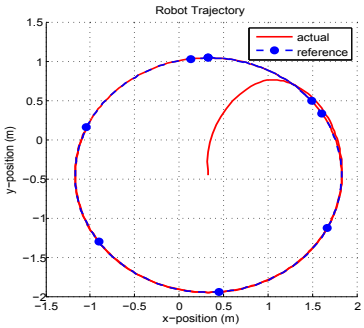
$$v = \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \text{ m/s}$$
$$\omega = \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \text{ rad/s}$$

The controller update time step is chosen to be 0.2 s with number of prediction steps equal to 20, leading to a prediction horizon time 4 s. The weight matrices **Q** and **R** of the cost function (3.11) are chosen as diagonal matrices with diagonal elements defined as (30, 30, 1) and (20, 10) for **Q** and **R**, respectively.

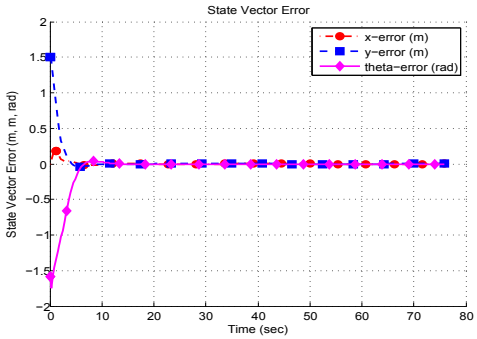
5.2.2 Simulation Results

Series of simulations were carried out in MATLAB using ACADO toolkit. The purpose is to test the proper functionality of controller. MATLAB simulations do not use tracking system and robot states are simulated over the prediction horizon.

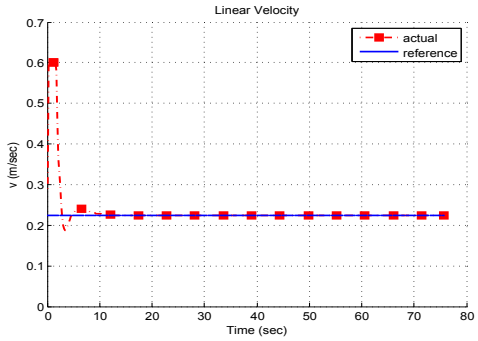
Figure 5.4 shows the MATLAB simulation results. It can be seen from Figure 5.4(a),



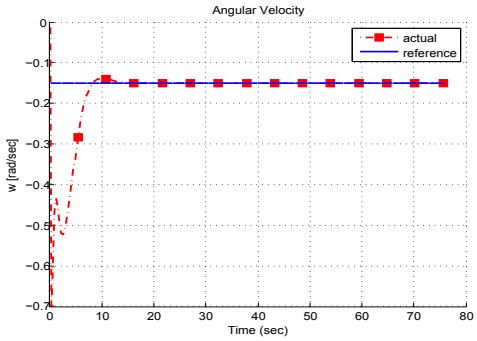
(a) Mobile Robot Trajectory



(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile

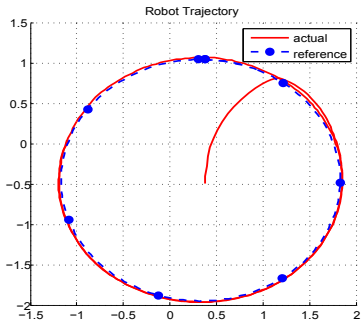
Figure 5.4: Simulation(ACADO) results of trajectory tracking for indoor navigation based on tracking system

robot trajectory, that robot started from position (0.3301 m, −0.4489 m) and makes a circle of radius 1.5 m centered at (0.3301 m, −0.4489 m). Figure 5.4(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The root mean square (rms) values of steady state error in x-position is 0.000063 m, y-position is 0.000065 m, and theta is 0.000057 rad. This leads to overall error state vector magnitude 0.00011. The steady state mean values with standard deviation for x-position error is -0.000009 ± 0.000062 m, y-position error is 0.000013 ± 0.000064 m, and theta error is -0.000057 ± 0.000006 rad. Figure 5.4(c) and 5.4(d), shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. Both linear and angular velocities converge to reference velocities. The rms value of steady state error in linear velocity is 0.000003 m/s, and for angular velocity it is

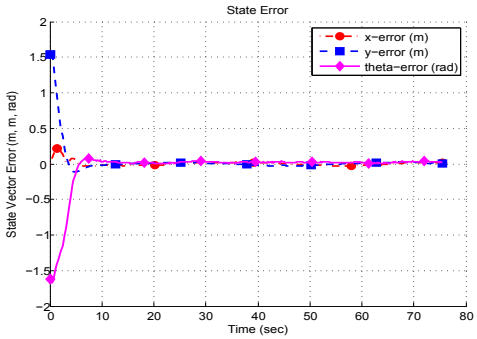
0.000003 rad/s. Overall error control vector magnitude is 0.000005. The steady state mean values with standard deviation for linear velocity error is -0.000003 ± 0.0000007 m/s, and angular velocity error is 0.0000008 ± 0.000003 rad/s. These results achieve the proper working of the controller developed for this problem.

5.2.3 Real-Time Experiment

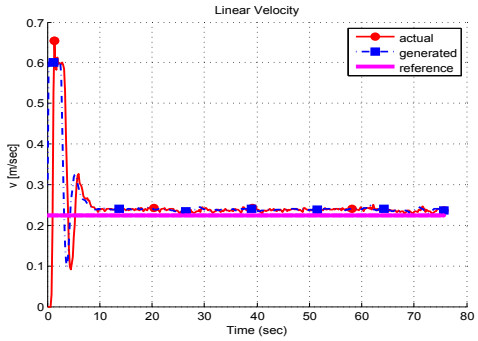
Figure 5.5 shows the real-time experimental results run on Seekur Jr. for outdoor navigation. It can be seen from figure 5.5(a), robot trajectory, that robot starts from position (0.3301 m, -0.4489 m) and makes a circle of radius 1.5 m centered at (0.3301 m, -0.4489 m). Figure 5.5(b) shows that error state vector error $\mathbf{x}_e$ converges to null vector. The steady state rms error in x-position is 0.0175 m, y-position is 0.0182 m, and theta is 0.0240 rad. This leads to overall state vector error magnitude 0.0348. The steady state mean values with standard deviation for x-position error is 0.0017 ± 0.0174 m, y-position error is 0.0017 ± 0.0182 m, and theta error is 0.0228 ± 0.0075 rad. This error is larger than the simulation results. It was expected, because the model of system doesn t use the dynamics of robot. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 5.5(c) shows the linear velocity of mobile robot. Again, there is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in section 4.1.2. Both generated and actual linear velocities converge to reference velocity. The rms value of steady state error for linear velocity is 0.0129 m/s. Figure 5.5(d) shows the actual angular velocity of robot and angular velocity generated by the controller. It has the same lag as appearing in the linear velocity case. The rms value of steady state error for angular velocity is 0.0121 rad/s. The steady state mean values with standard deviation for linear velocity error is 0.0124 ± 0.0034 m/s, and angular velocity error is



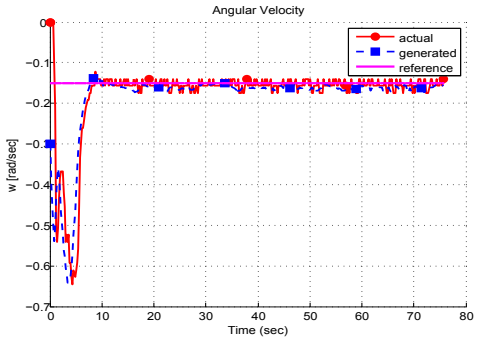
(a) Mobile Robot Trajectory



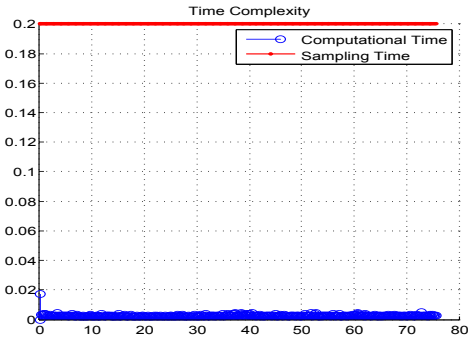
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 5.5: Realtime Experimental results of trajectory tracking for indoor navigation based on tracking system

-0.0056 ± 0.0108 rad/s. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. Figure 5.5(e) shows that the computation time, for obtaining the robot status, solving NMPC, evaluating velocity command and saving results, is in

order of fraction of milliseconds whereas sampling time selected for this experiment is 0.2 s. So, there is enough time to apply the velocity command to mobile robot which is still equivalent to sampling time. These result shows the satisfactory performance of developed controller for the point stabilization problem of mobile robot. Table 5.2 summarizes the rms values of steady state errors for robot states and controls.

Steady State Error		
Parameter	**Simulation**	**Real-time Experiment**
x-position (m)	6.27e-05	0.0175
y-position (m)	6.49e-05	0.0182
Heading angle (rad)	5.71e-05	0.0240
Linear Velocity (m/s)	3.40e-06	0.0129
Angular Velocity (rad/s)	3.05e-06	0.0121

Table 5.2: Steady state errors of trajectory tracking problem for indoor navigation based on tracking system

5.3 Outdoor Navigation based on DGPS

5.3.1 Problem Formulation

The reference trajectory is a circular trajectory defined by:

$$x_r(t) = -4.783 + 1.5 * \sin(0.133 * t) \text{ m}$$

$$y_r(t) = -89.306 + 1.5 * \cos(0.133 * t) \text{ m}$$

$$v_r = 0.2 \text{ m/s}$$

$$\omega_r = -0.133 \text{ rad/s}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

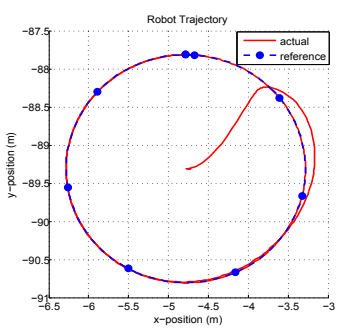
$$v = \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \quad \text{m/s}$$
$$\omega = \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \quad \text{rad/s}$$

The controller update time step is chosen to be 0.22 s with number of prediction steps equal to 20, leading to a prediction horizon time T = 4 s. The weight matrices **Q** and **R** of the cost function (8) are chosen as diagonal matrices with diagonal elements defined as (30, 30, 1) and (50, 50) for **Q** and **R**, respectively.

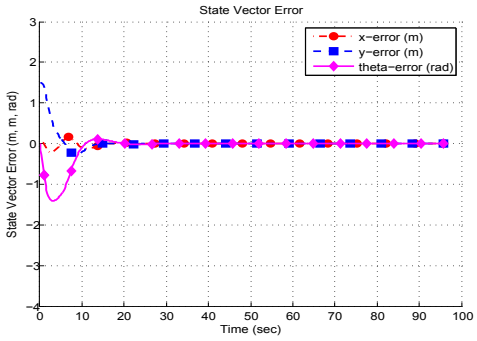
5.3.2 Simulation Results

Simulation results are obtained from MATLAB using ACADO. Simulations do not use GPS based localization. So, in order to get mobile robot state feedback, robot states are simulated over the prediction horizon.

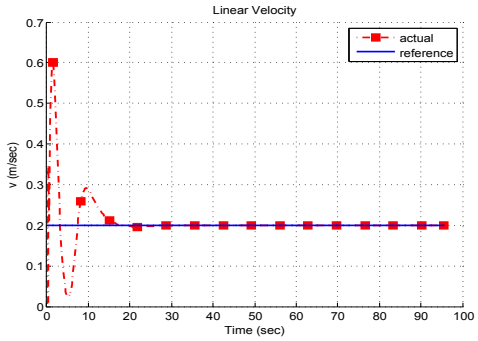
Figure 5.6 shows the simulation results. It can be seen from figure 5.6(a), robot trajectory, that robot starts from position $(-4.78 \text{ m}, -89.31 \text{ m})$ and makes a circle of radius 1.5 m centered at $(-4.78 \text{ m}, -89.31 \text{ m})$. Figure 5.6(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. The steady state rms error in x-position is 0.00057 m, y-position is 0.00066 m, and theta is 0.00095 rad. This leads to overall error state vector magnitude 0.0013. The steady state mean values with standard deviation for x-position error is 0.00010 ± 0.00056 m, y-position error is 0.00018 ± 0.00063 m, and theta error is 0.00027 ± 0.00092 rad. Figure 5.6(c) and 5.6(d), shows that both linear and angular velocities are always with in the specified range as mentioned in the constraints to the controller. Both linear and angular velocities converge to reference velocities. The rms value of steady state error in linear velocity is 0.00048 m/s,



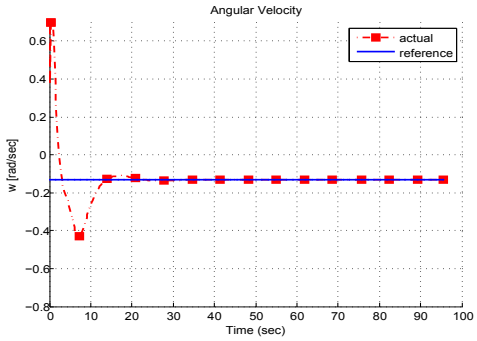
(a) Mobile Robot Trajectory



(b) State Vector Error



(c) Linear Velocity Profile



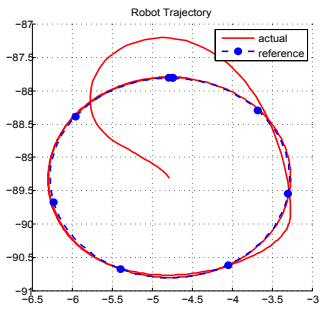
(d) Angular Velocity Profile

Figure 5.6: Simulation results of trajectory tracking for outdoor navigation

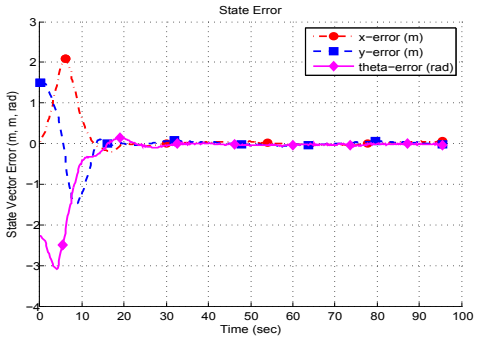
and for angular velocity it is 0.00039 rad/s. Overall control vector error magnitude is 0.00062. The steady state mean values with standard deviation for linear velocity error is -0.00045 ± 0.00017 m/s, and angular velocity error is -0.00001 ± 0.00039 rad/s. These results achieve the proper working of the controller developed for this problem.

5.3.3 Real-Time Experiment

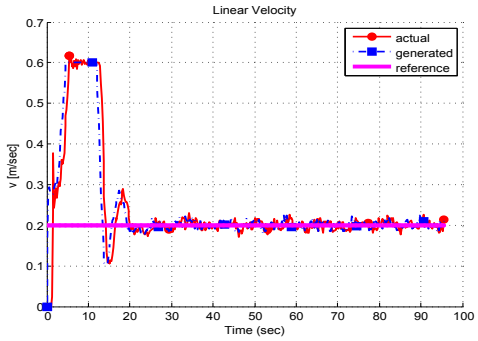
Figure 5.7 shows the real-time experimental results run on Seekur Jr. for outdoor navigation. It can be seen from figure 5.7(a), robot trajectory, that robot starts from position $(-4.78 \text{ m}, -89.31 \text{ m})$ and makes a circle of radius 1.5 m centered at $(-4.78 \text{ m}, -89.31 \text{ m})$. Figure 5.7(b) shows that the error state vector $\mathbf{x}_e$ converges to null vector. The steady state rms error in x-position is 0.0315 m, y-position is



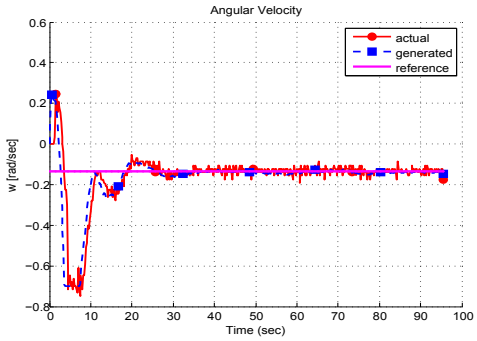
(a) Mobile Robot Trajectory



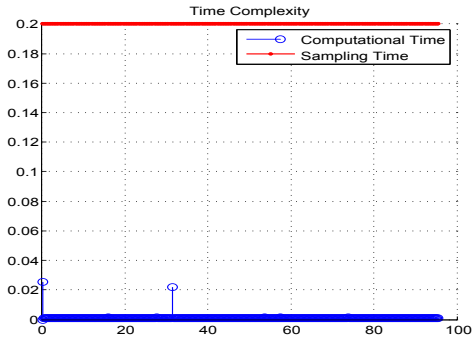
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 5.7: Realtime Experimental results of trajectory tracking for outdoor navigation

0.0377 m, and theta is 0.0277 rad. This leads to overall error state vector magnitude 0.0564. The steady state mean values with standard deviation for x-position error is 0.0072 ± 0.0307 m, y-position error is 0.0085 ± 0.0368 m, and theta error is $-0.0218 \pm$ 0.0171 rad. Overall trend in the robot trajectory and state vector error is similar for

both simulation and real time experimental cases. Figure 5.7(c) shows the linear velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in Section 4.1.2. Both generated and actual linear velocities converge to reference velocity. The steady state rms error magnitude for linear velocity is 0.0098 m/s. The steady state mean values with standard deviation for linear velocity error is 0.0002 ± 0.0098 m/s. Figure 5.7(d) shows angular velocities, actual velocity of robot, velocity generated by the controller and the reference velocity. It has the same lag as appearing in the linear velocity case. The steady state rms error magnitude for angular velocity is 0.0171 rad/s. The steady state mean values with standard deviation for angular velocity error is -0.0010 ± 0.0171 rad/s. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. Overall control vector error magnitude is 0.0197, which is well within acceptable range. Figure 5.7(e) shows that the computation time, for obtaining the robot status, solving NMPC, evaluating velocity command and saving results, is in order of fraction of milliseconds whereas sampling time selected for this experiment is 0.2 s. So, there is enough time to apply the velocity command to mobile robot which is still equal to sampling time. These result show the satisfactory performance of the developed controller for the trajectory tracking problem of the mobile robot for outdoor navigation. Table 5.2 summarizes the rms values of steady state errors for robot states and controls.

Steady State Error		
Parameter	**Simulation**	**Real-time Experiment**
x-position (m)	0.0006	0.0315
y-position (m)	0.0007	0.0377
Heading angle (rad)	0.0009	0.0277
Linear Velocity (m/s)	0.0005	0.0098
Angular Velocity (rad/s)	0.0004	0.0171

Table 5.3: Steady state errors of trajectory tracking problem for outdoor navigation based on DGPS

5.4 Obstacle Avoidance

5.4.1 Problem Formulation

The reference trajectory is a circular trajectory defined by:

$$x_r(t) = 0.3301 + 1.5 * \sin(0.15 * t) \text{ m}$$

$$y_r(t) = -0.4489 + 1.5 * \cos(0.15 * t) \text{ m}$$

$$v_r = 0.225 \text{ m/s}$$

$$\omega_r = -0.15 \text{ rad/s}$$

In order to move from initial position and stabilize itself at the desired pose, robot is constrained on its linear and angular speed, specified as:

$$v = \begin{bmatrix} -0.3 & 0.6 \end{bmatrix} \text{ m/s}$$

$$\omega = \begin{bmatrix} -0.7 & 0.7 \end{bmatrix} \text{ rad/s}$$

While tracking the reference trajectory, the mobile robot is also supposed to avoid the obstacle in its trajectory present at

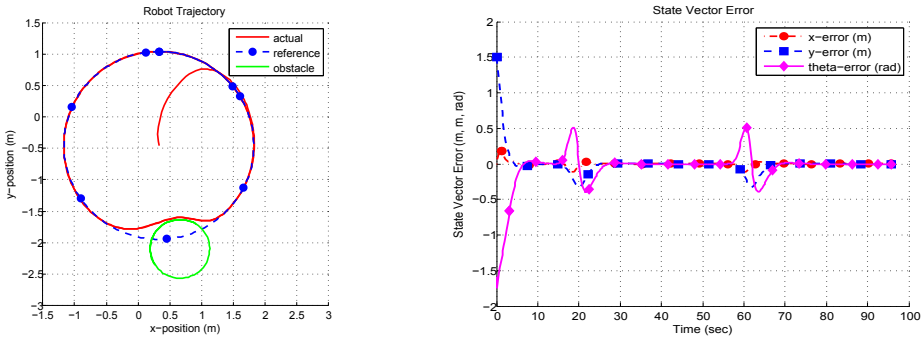
$$(x_o, y_o) = (0.663 \text{ m}, -2.103 \text{ m})$$

The controller update time step is chosen to be 0.2 s with number of prediction steps equal to 20, leading to a prediction horizon time 4 s. The weight matrices $\mathbf{Q}$ and $\mathbf{R}$ of the cost function (3.11) are chosen as diagonal matrices with diagonal elements defined as (30, 30, 1) and (20, 10) for $\mathbf{Q}$ and $\mathbf{R}$, respectively. The problem is similar to the problem stated in Section 5.2, but now there are obstacles in the robot path which it needs to avoid. All the coordinate points correspond to the locations in the tracked environment.

5.4.2 Simulation Results

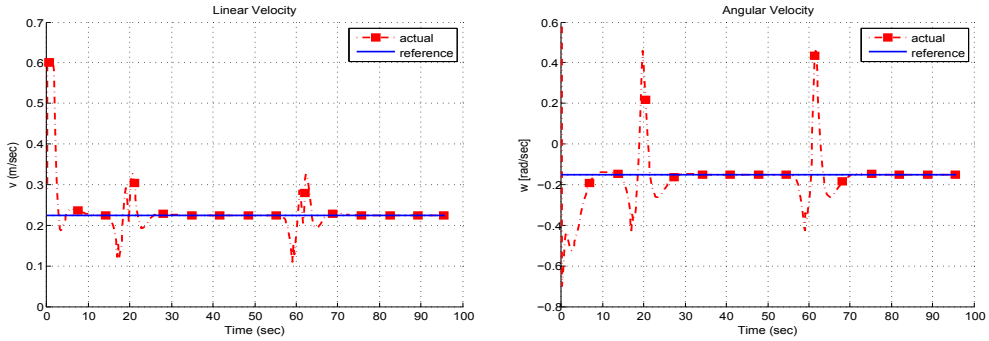
Simulations are run in MATLAB using ACADO toolkit. The purpose is to test the proper functionality of controller. Simulations do not use tracking system and robot states are simulated over the prediction horizon. Simulation results are shown in figure 5.8, including (a) mobile robot trajectory, (b) state vector error, (c) linear velocity and (d) angular velocity.

Figure 5.8 shows the simulation results. It can be seen from figure 5.8(a), robot trajectory, that robot starts from position (0.3301 m, −0.4489 m) and makes a circle of radius 1.5 m centered at (0.3301 m, −0.4489 m). The mobile robot successfully avoids the obstacle, which comes in the reference trajectory. Once obstacle is avoided, robot again starts to track the reference trajectory. Figure 5.8(b) shows that error state vector $\mathbf{x}_e$ converges to null vector. It can be noted while robot is avoiding the obstacle, the state errors are present. Figure 5.8(c) and 5.8(d), shows that both linear and



(a) Mobile Robot Trajectory

(b) State Vector Error



(c) Linear Velocity Profile

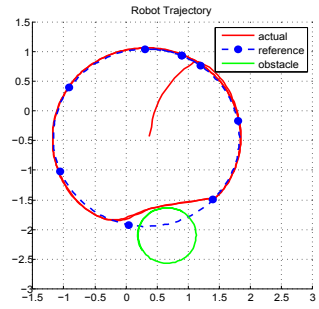
(d) Angular Velocity Profile

Figure 5.8: Simulation results of trajectory tracking with obstacle avoidance

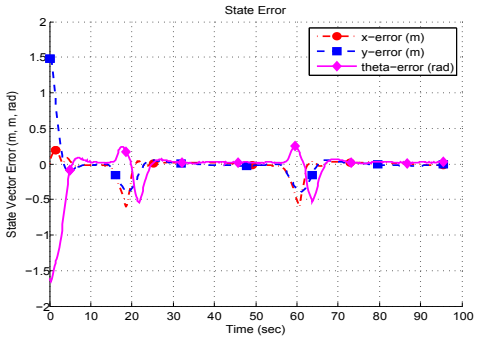
angular velocities are always with in the specified range as mentioned in the constraints to the controller. Both linear and angular velocities converge to reference velocities. At the time of obstacle avoidance, both linear and angular velocities deviates from the reference values. Once obstacle is avoided, both linear and angular velocities again converge to reference values. These results achieve the proper working of the controller developed for this problem.

5.4.3 Real-Time Experiment

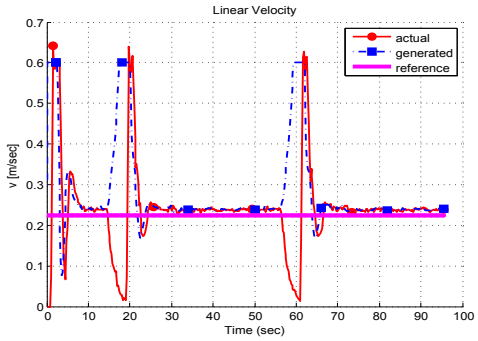
Figure 5.9 shows the real-time experimental results run on Seekur Jr. for outdoor navigation. It can be seen from figure 5.9(a), robot trajectory, that robot starts from position $(0.3760\text{ m}, -0.4340\text{ m})$ and makes a circle of radius 1.5 m centered



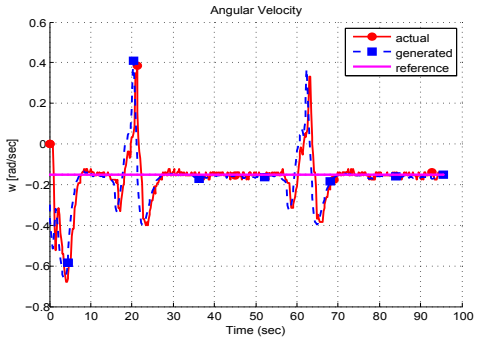
(a) Mobile Robot Trajectory



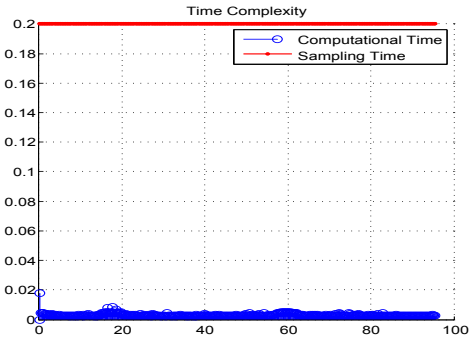
(b) State Vector Error



(c) Linear Velocity Profile



(d) Angular Velocity Profile



(e) Time Complexity

Figure 5.9: Realtime Experimental results of trajectory tracking for obstacle avoidance

at $(0.3301\text{ m}, -0.4489\text{ m})$. The mobile robot successfully avoids the obstacle, which comes in the reference trajectory. Once obstacle is avoided, robot again starts to track the reference trajectory. Figure 5.9(b) shows that the error state vector $\mathbf{x}_e$ converges to null vector. Overall trend in the robot trajectory and state vector error is similar for both simulation and real time experimental cases. Figure 5.9(c) shows the linear

velocity of mobile robot. There is time lag between the robot actual velocity and velocity command generated by the controller, because of same reason, as mentioned in Section 4.1.2. Both generated and actual linear velocities converge to reference velocity. Actual linear velocity converges to zero when obstacle is encountered and then tracks the linear velocity command generated by the controller. Once the obstacle is avoided linear velocity starts to track the reference value. Figure 5.9(d) shows angular velocities, actual velocity of robot, velocity generated by the controller and the reference velocity. It has the same lag as appearing in the linear velocity case. From both figures, it is clear that velocities are always within the specified range, for both generated and actual ones, as mentioned in the constraints to the controller. Figure 5.9(e) shows that the computation time, for obtaining the robot status, solving NMPC, evaluating velocity command and saving results, is in order of fraction of milliseconds whereas sampling time selected for this experiment is 0.2 s. So, there is enough time to apply the velocity command to mobile robot which is still equal to sampling time. At the time instants of obstacle avoidance, computational time is more, but still well below the sampling time. These result show the satisfactory performance of the developed controller for the trajectory tracking problem of the mobile robot with obstacle avoidance.

Chapter 6

Conclusion and Future Work

About this Chapter: This chapter presents the discussion of the implementation of the control strategy used. It concludes by mentioning the research contributions and limitations of the thesis. Future directives are also highlighted

6.1 Discussion

6.1.1 Conclusion

This thesis successfully achieves its objectives defined in Chapter 1. It implements NMPC for indoor and outdoor navigation, to solve point stabilization and trajectory tracking problem. From the results presented in Chapter 4 for point stabilization and Chapter 5 for trajectory tracking, it is clear that steady state errors for robot states and inputs converge to zero. The mobile robot always followed the smooth trajectory. In real time experiments, the robot velocities (both linear and angular) follow the same velocities as generated by the controller. These velocities are always in the constrained input limits. There is small time delay exists in the actual velocity and velocity command generated by the controller. This delay is cause of only considering

the kinematics in robot model, but overall trend is same for both. The computational time for generation of velocity command and doing all processing is in the order of fraction of milliseconds. Therefore, computational time is always less than the sampling time for the controller. These leads to the successful implementation of NMPC on mobile robot. Two methods have been used for indoor navigation to address the localization issue. Among these two methods the results are better for the tracking control, rather than mapped based localization. Moreover, obstacle avoidance feature, for static obstacles, is successfully implemented. The results show the satisfactory performance of the developed controller.

6.1.2 Research contributions

The research contributions are as follow:

- **Contribution 1**: Real-time implementation of NMPC to solve point stabilization problem in indoors using mapped based localization and also with tracking system
- **Contribution 2**: Real-time implementation of NMPC to solve trajectory tracking problem for indoor navigation using mapped based localization
- **Contribution 3**: Real-time implementation of NMPC to solve point stabilization problem for outdoor navigation using DGPS
- **Contribution 4**: Real-time implementation of NMPC to solve trajectory tracking problem for outdoor navigation using DGPS
- **Contribution 5**: Real-time implementation of NMPC with obstacle avoidance for both point stabilization and trajectory tracking

6.1.3 Research limitations

This research work has following limitations:

- **Obstacle Avoidance**: This thesis addresses the obstacle avoidance problem, which comes across in many cases of real time navigation of mobile robots. Obstacles considered in this research are static and whose location is already known to the controller. But in some real-world scenarios, the location of the obstacles is not pre-known and in number of cases obstacles are moving also. But this research work does not consider those cases.

- **Localization**: In this research, control of wheeled mobile robot is considered for indoor and outdoor navigation. Navigation consists of localization and control, but this thesis only considers the control aspect and takes the localization as it receives from the server. For indoor navigation two methods are used to get feedback data from the mobile robot: mapped based localization and indoor tracking which doesn t consider the localization instead it uses motion capture system. Then for outdoor, it uses DGPS based localization which converts the DGPS coordinates into the mapped pose information. Inaccuracy of localization e ects the results, but its not the research focus area for this thesis.

6.2 Future Directions

This research work can be extended in following aspects, while considering the research limitations presented in Section 6.1.3.

- **Unknown and Dynamic Obstacle Avoidance**: In future, the obstacle avoidance feature would be extended to the unknown obstacles, the obstacles which are not known to the controller and comes across during navigation, and dy-

namic obstacles, the obstacles which are in motion and not static at fixed location. For this purpose, a high-level controller can be developed, like behaviour based controller. This high-level controller will be doing the obstacle avoidance and NMPC will be taking care of the robot trajectory and formulating velocity command to follow the desired trajectory.

- **State Estimation**: In this research localization data was available, but in future, when localization data is not su cient, state estimation can be added. Even now, in order to improve the localization state estimation can be added which will eventually improve the performance of system. Di erent state estimation techniques are studied in literature, like basic Extended Kalman Filter (EKF) techniques [70], Unscented Kalman Filter (UKF) [71], and then we have the Moving Horizon Estimation techniques (MHE) [26] in which the prediction horizon is moving each step, just like NMPC.

- **Universal Platform for server**: In this thesis, three di erent methods have been developed to do the indoor and outdoor navigation of wheeled mobile robots; based on mapped based localization, tracking system for indoor navigation and DGPS based localization for outdoor navigation. For these methods two di erent server files have been run on the mobile robot, ARNL server and MOGS server. This server connects with client in PC and which connects MATLAB. In future this work can be extended in which a single server file would be run in the mobile robot which would select in the indoor and outdoor, and which method of localization is used. Secondly, rather than using PC and having TCP/IP connection, a executable MATLAB code can run in the onboard computer of Seekur Jr. This will reduce the time computation as one layer of communication will be reduced and algorithm will become more e cient.

Bibliography

[1] G. Klan ar and I. ' krjanc, Tracking-error model-based predictive control for mobile robots in real time, *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460 469, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889007000140

[2] B. K. Muirhead, Mars rovers, past and future, in *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, vol. 1, March 2004, p. 134 Vol.1.

[3] M. Bajracharya, M. Bajracharya, M. Bajracharya, M. Bajracharya, M. W. Maimone, M. W. Maimone, M. W. Maimone, M. W. Maimone, D. Helmick, D. Helmick, D. Helmick, and D. Helmick, Autonomy for mars rovers: Past, present, and future, *Computer*, vol. 41, no. 12, pp. 44 50, Dec 2008.

[4] T. A. Fuad, I. Ridwan, M. I. H. Raju, A. S. M. Mohiuddin, S. S. Saumik, M. M. H. Polash, P. Saha, M. Rahman, S. R. Dipta, M. I. Abedin, S. S. C. Puspo, W. M. Abdullah, and M. A. Hossain, Maya: A fully functional rover designed for the mars surface, in *2015 18th International Conference on Computer and Information Technology (ICCIT)*, Dec 2015, pp. 423 428.

[5] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, Cooperative air and ground surveillance, *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 16 25, Sept 2006.

[6] M. Bernard, K. Kondak, I. Maza, and A. Ollero, Autonomous transportation and deployment with aerial robots for search and rescue missions, *Journal of Field Robotics*, vol. 28, no. 6, pp. 914 931, 2011.

[7] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida *et al.*, Collaborative mapping of an earthquake-damaged building via ground and aerial robots, *Journal of Field Robotics*, vol. 29, no. 5, pp. 832 841, 2012.

[8] D. Voth, A new generation of military robots, *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 2 3, Jul 2004.

[9] J. Khurshid and H. Bing-rong, Military robots - a glimpse from today and tomorrow, in *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, vol. 1, Dec 2004, pp. 771 777 Vol. 1.

[10] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, Cooperative large area surveillance with a team of aerial mobile robots for long endurance missions, *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 329 345, 2013.

[11] D. Kingston, R. W. Beard, and R. S. Holt, Decentralized perimeter surveillance using a team of uavs, *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394 1404, Dec 2008.

[12] J. Palacin, J. A. Salse, I. Valganon, and X. Clua, Building a mobile robot for a oor-cleaning operation in domestic environments, *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 5, pp. 1418 1424, Oct 2004.

[13] R. Fathzadeh, V. Mokhtari, F. Abdollahi, N. Nabavi, F. Abazari, and H. Bagheri, Mobile robot in domestic environment, in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, vol. 2, April 2014, pp. 918 922.

[14] A. S. Conceção, A. P. Moreira, and P. J. Costa, Design of a mobile robot for robocup middle size league, in *2009 6th Latin American Robotics Symposium (LARS 2009)*, Oct 2009, pp. 1 6.

[15] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, ser. Intelligent robotics and autonomous agents. MIT Press, 2011.

[16] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation, *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1087 1097, Oct 2009.

[17] H. Wang, J. Zhang, J. Yi, D. Song, S. Jayasuriya, and J. Liu, Modeling and motion stability analysis of skid-steered mobile robots, in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 4112 4117.

[18] J. E. Normey-Rico, I. Alcala, J. Gómez-Ortega, and E. F. Camacho, Mobile robot path tracking using a robust pid controller, *Control Engineering Practice*, vol. 9, no. 11, pp. 1209 1214, 2001.

[19] E. Camacho and C. Alba, *Model Predictive Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2013. [Online]. Available: https://books.google.ca/books?id=tXZDAAAAQBAJ

[20] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, ser. Communications and Control Engineering. Springer London, 2011. [Online]. Available: https://books.google.ca/books?id=kBnz6OFxO8wC

[21] F. Allgower, R. Findeisen, Z. K. Nagy *et al.*, Nonlinear model predictive control: from theory to application, *Journal of the Chinese Institute of Chemical Engineers*, vol. 35, no. 3, pp. 299 315, 2004.

[22] F. Kuhne, W. F. Lages, and J. M. G. da Silva Jr, Model predictive control of a mobile robot using linearization, in *Proceedings of mechatronics and robotics*, 2004, pp. 525 530.

[23] W. F. Lages and J. A. V. Alves, Real-time control of a mobile robot using linearized model predictive control, *IFAC Proceedings Volumes*, vol. 39, no. 16, pp. 968 973, 2006.

[24] F. Künhe, J. Gomes, and W. Fetter, Mobile robot trajectory tracking using model predictive control, in *II IEEE latin-american robotics symposium*, 2005.

[25] M. W. Mehrez, G. K. I. Mann, and R. G. Gosine, Stabilizing nmpc of wheeled mobile robots using open-source real-time software, in *2013 16th International Conference on Advanced Robotics (ICAR)*, Nov 2013, pp. 1 6.

[26] A. Jayasiri, S. Gros, and G. K. I. Mann, Tracking control and state estimation of a mobile robot based on nmpc and mhe, in *2016 American Control Conference (ACC)*, July 2016, pp. 1999 2004.

[27] D. Ariens, B. Houska, H. Ferreau, and F. Logist, Acado: Toolkit for automatic control and dynamic optimization, Optimization in Engineering Center (OPTEC). [Online]. Available: http://www.acadotoolkit.org/

[28] (2016) The Omron Adept MobileRobots website. [Online]. Available: http://www.mobilerobots.com/

[29] K. Kanjanawanishkul, Coordinated path following control and formation control of mobile robots, Ph.D. dissertation, University of Tübingen, Tübingen, Germany, 2010. [Online]. Available: https://publikationen.uni-tuebingen.de/xmlui/handle/10900/49446

[30] T. Faulwasser, Optimization-based solutions to constrained trajectory-tracking and path-following problems, Ph.D. dissertation, Otto-von-Guericke University Magdeburg, Magdeburg, Germany, 2012. [Online]. Available: https://infoscience.ep .ch/record/184941

[31] R. W. Brockett *et al.*, Asymptotic stability and feedback stabilization, *Differential geometric control theory*, vol. 27, no. 1, pp. 181 191, 1983.

[32] C. Samson, Control of chained systems application to path following and time-varying point-stabilization of mobile robots, *IEEE transactions on Automatic Control*, vol. 40, no. 1, pp. 64 77, 1995.

[33] C. C. De Wit and O. Sordalen, Exponential stabilization of mobile robots with nonholonomic constraints, *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1791 1797, 1992.

[34] A. De Luca, G. Oriolo, and M. Vendittelli, Stabilization of the unicycle via dynamic feedback linearization, in *6th IFAC Symp. on Robot Control*, 2000, pp. 397 402.

[35] G. Oriolo, A. De Luca, and M. Vendittelli, Wmr control via dynamic feedback linearization: design, implementation, and experimental validation, *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835 852, 2002.

[36] G. Indiveri, Kinematic time-invariant control of a 2d nonholonomic vehicle, in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3. IEEE, 1999, pp. 2112 2117.

[37] K. Park, H. Chung, and J. G. Lee, Point stabilization of mobile robots via state-space exact feedback linearization, *Robotics and Computer-Integrated Manufacturing*, vol. 16, no. 5, pp. 353 363, 2000.

[38] A. De Luca, G. Oriolo, and M. Vendittelli, Control of wheeled mobile robots: An experimental overview, in *Ramsete*. Springer, 2001, pp. 181 226.

[39] A. De Luca and M. D. Di Benedetto, Control of nonholonomic systems via dynamic compensation, *Kybernetika*, vol. 29, no. 6, pp. 593 608, 1993.

[40] B. d Andréa Novel, G. Campion, and G. Bastin, Control of nonholonomic wheeled mobile robots by state feedback linearization, *The International journal of robotics research*, vol. 14, no. 6, pp. 543 559, 1995.

[41] C. Samson and K. Ait-Abderrahim, Feedback control of a nonholonomic wheeled cart in cartesian space, in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1136 1141.

[42] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, A stable tracking control method for an autonomous mobile robot, in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 384 389.

[43] Z.-P. JIANGdagger and H. Nijmeijer, Tracking control of mobile robots: a case study in backstepping, *Automatica*, vol. 33, no. 7, pp. 1393 1399, 1997.

[44] D. Chwa, Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates, *IEEE transactions on control systems technology*, vol. 12, no. 4, pp. 637 644, 2004.

[45] W. E. Dixon, D. M. Dawson, F. Zhang, and E. Zergeroglu, Global exponential tracking control of a mobile robot system via a pe condition, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 1, pp. 129 142, 2000.

[46] G. Oriolo, A. De Luca, and M. Vendittelli, Wmr control via dynamic feedback linearization: design, implementation, and experimental validation, *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835 852, 2002.

[47] D. Gu and H. Hu, A stabilizing receding horizon regulator for nonholonomic mobile robots, *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1022 1028, 2005.

[48] D. Gu and et al., Receding horizon tracking control of wheeled mobile robots, *IEEE Transactions on control systems technology*, vol. 14, no. 4, pp. 743 749, 2006.

[49] E. F. Camacho and C. Bordons, *Nonlinear Model Predictive Control: An Introductory Review*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1 16. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72699-9_1

[50] J. Richalet, A. Rault, J. Testud, and J. Papon, Algorithmic control of industrial processes, in *Proceedings of the 4th IFAC symposium on identification and system parameter estimation*, 1976, pp. 1119 1167.

[51] J. Richalet and et al., Model predictive heuristic control: Applications to industrial processes, *Automatica*, vol. 14, no. 5, pp. 413 428, 1978.

[52] C. R. Cutler and B. L. Ramaker, Dynamic matrix control - a computer control algorithm, in *joint automatic control conference*, no. 17, 1980, p. 72.

[53] C. E. Garcia, D. M. Prett, and M. Morari, Model predictive control: theory and practice a survey, *Automatica*, vol. 25, no. 3, pp. 335 348, 1989.

[54] S. J. Qin and T. A. Badgwell, An overview of nonlinear model predictive control applications, in *Nonlinear model predictive control*. Springer, 2000, pp. 369 392.

[55] C. Chen and L. Shaw, On receding horizon feedback control, *Automatica*, vol. 18, no. 3, pp. 349 352, 1982.

[56] S. a. Keerthi and E. G. Gilbert, Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations, *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265 293, 1988.

[57] D. Q. Mayne and H. Michalska, Receding horizon control of nonlinear systems, *IEEE Transactions on automatic control*, vol. 35, no. 7, pp. 814 824, 1990.

[58] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, Constrained model predictive control: Stability and optimality, *Automatica*, vol. 36, no. 6, pp. 789 814, 2000.

[59] F. Kuhne, W. F. Lages, and J. M. G. da Silva Jr, Model predictive control of a mobile robot using linearization, in *Proceedings of mechatronics and robotics*, 2004, pp. 525 530.

[60] K. Kanjanawanishkul, Motion control of a wheeled mobile robot using model predictive control: A survey, *KKU Research Journal*, vol. 17, no. 5, pp. 811 837, 2012.

[61] L. Pacheco and N. Luo, Mobile robot local trajectory tracking with dynamic model predictive control techniques, *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 6, pp. 3457 3483, 2011.

[62] L. Pacheco, N. Luo, and X. Cufí, *Using Model Predictive Control for Local Navigation of Mobile Robots*. INTECH Open Access Publisher, 2011.

[63] E. M. Gertz and S. J. Wright, Object-oriented software for quadratic programming, *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 1, pp. 58 81, 2003.

[64] F. Kuhne, W. F. Lages, and J. G. Da Silva, Point stabilization of mobile robots with nonlinear model predictive control, in *IEEE International Conference Mechatronics and Automation, 2005*, vol. 3. IEEE, 2005, pp. 1163 1168.

[65] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, Autogenerating microsecond solvers for nonlinear {MPC}: A tutorial using {ACADO} integrators, *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 685 704, 2015. [Online]. Available: http://dx.doi.org/10.1002/oca.2152

[66] B. Houska, H. Ferreau, and M. Diehl, ACADO Toolkit An Open Source Framework for Automatic Control and Dynamic Optimization, *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298 312, 2011.

[67] B. Houska, H. Ferreau, M. Vukov, and R. Quirynen, *ACADO Toolkit User's Manual*, 1st ed., Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, K. U. Leuven, 2014.

[68] D. Ariens, B. Houska, H. Ferreau, and F. Logist, *ACADO for Matlab User's Manual*, 1st ed., Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, K. U. Leuven, June 2010. [Online]. Available: http://www.acadotoolkit.org/

[69] (2016) The OptiTrack website. [Online]. Available: http://optitrack.com/

[70] V. M. Becerra, P. Roberts, and G. Gri ths, Applying the extended kalman filter to systems described by nonlinear di erential-algebraic equations, *Control Engineering Practice*, vol. 9, no. 3, pp. 267 281, 2001.

[71] S. J. Julier and J. K. Uhlmann, Unscented filtering and nonlinear estimation, *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401 422, Mar 2004.