Otto-von-Guericke-University Magdeburg
Faculty of Electrical Engineering and Information Technology
Chair of Institute for Automation Technology

Integrated Project

Implementation of a path following controller on Unmanned Ground Vehicle (UGV)

Submitted on: March 1, 2022

Submitted by: Shailesh Kumar (229916)
Arman Ahmed Khan (230601)

Supervised by: M.Sc. Mohamed Soliman
Research Assistant Institute for Automation Technology (IFAT)

Lehrstuhl für
Systemtheorie und Regelungstechnik
Prof. Dr.-Ing. Rolf Findeisen

Integrated Project

Submitted by:

Shailesh Kumar(229916)

Arman Ahmed Khan (230601)

Implementation of a path following controller on Unmanned Ground Vehicle (UGV)

Herr Kumar und Khan haben dieses Forschungsprojekt mit großer Eigenständigkeit und sehr guten Ergebnissen bearbeitet.

Die Betreung erfolgte durch M.Sc. Mohamed Soliman

Magdeburg, March 1, 2022

Prof. Dr.-Ing. Rolf Findeisen
Magdeburg, March 1, 2022

Contents

Objectives of the project:

The main objectives of this project are:

1. Formulate an NMPC path following problem for unmanned ground vehicle in a simulation environment.
2. Validate the simulated results through an experiment validation.

Declaration by the candidate

The project entitled **Implementation of a path following controller on unmanned ground vehicle(UGV)** is conducted under the supervision of Mohamed Soliman, Research Assistant of Institute for Automation Engineering (IFAT) Laboratory for Systems Theory and Automatic Control Otto-von-Guericke University Magdeburg 39106 Magdeburg - Germany.

We hereby declare that this Integrated Project (IP) thesis is our own work and effort and has not been submitted anywhere for any award. Where other sources of information have been used, they have been marked.
The work has not been presented in the same or a similar form to any other testing authority and has not been made public.

Magdeburg, March 1, 2022
Shailesh Kumar
Arman Ahmed Khan

Abstract

Considering the present global scenario, the world is moving drastically towards unmanned ground vehicles. Designing a safe path for an unmanned vehicle is a crucial problem. The path should not be only obstacle-free but also take into account the vehicle dynamics so that the path is feasible and reliable to ensure more safety.
The promising controller design for UGV which takes kinematic bicycle model as the mathematical model makes the path more feasible. An offline planner which approximate the Unmanned Ground Vehicle (UGV) as a double integrator model is used to plan a safe path taking into account all the environmental information. This path is further expressed by a path parameter theta (θ) in an Nonlinear Model Predictive Control (NMPC) path following formulation to allow the vehicle to follow the path safely. After path generation, the controller should execute the path driving the car from a specific point to the final destination. The whole scenario has been executed under a global environment that is calibrated using multiple cameras. The efficacy of the MPC is illustrated via simulation results using kinematic bicycle model. Moreover, simulation results and experimental validation are presented in this project to show the efficacy of the proposed controller.

Nomenclature

u	Control vector
a	Acceleration of the vehicle
T_s	Sample time
x	State
θ	Path parameter
v	Velocity of the vehicle
y	Output
Q, R	Weighting matrices
A	Initial car position
B	Final car position
P_h	Prediction horizon
T_f	Simulation time
$X - Y - Z$	Coordinate system
δ	Steering angle
ψ	Heading angle
β	Slip angle
L	length of the vehicle
l_f	Distance between the front wheel to the center of vehicle
l_r	Distance between the rear wheel to the center of vehicle
w	Yaw angle

List of Acronyms

UGV Unmanned Ground Vehicle

UGVs Unmanned Ground Vehicles

MPC Model Predictive Control

NMPC Non-linear Model Predictive Control

ACADO Automatic Control and Dynamic Optimization

NMPC Nonlinear Model Predictive Control

ROS Robot Operating System

LIDAR Light Detection and Ranging

OCP Optimal Control Problem

IP Integrated Project

LiPo lithium polymer

MPFC Model predictive path following control

WPS WiFi positioning system

List of Figures

List of Tables

1 Introduction

UGV has been in development for several decades. The first UGV was developed in the 1930s decades for military purposes [1]. Over the period, many universities and research institutes found new UGV technologies for commercial use, transportation services, etc. The development of Unmanned Ground Vehicles (UGVs) has a promising application in industry and society, such as traffic congestion, surveillance, military, accidents, and human casualties. Generally, the autonomous vehicle can be classified according to its architecture, capability, environment, or application, such as ground, water, and aerial vehicles. Autonomous cars have recently sparked a surge of interest in the academic, civilian, and research communities as essential solutions to various applications. Below Figure (1.1) shows the various scope of UGV in various fields [2] [3] [4].

Figure 1.1: UGV applications

1.1 Project Structure

This project aims to implement the path following controller on UGV using model-based predictive controller. The project report is structured as follows:

Chapter-[1] covers the brief introduction of UGV with various applications, the main objective of the experiment, problem facing, and solution approach. Also, a short introduction of Model Predictive Control (MPC) has been given. **Chapter-[2]** includes the brief explanation of mathematical formulation and structure of the car model. Moreover, NMPC path following formulation, ACADO solver, and offline planner have also been explained briefly. **Chapter-[3]** presents the environment simulation and the simulation results of the Hamster with the help of the MATLAB-ACADO software package. **Chapter-[4]** depicts the experimental setup for Unmanned Ground Vehicle. A manual designed global environment that is calibrated using a camera system has been presented. Moreover, the step-wise process for camera calibration, rigid body creation, and successfully running the MATLAB-ROS program with motive measurement have been explained. At the end, **Chapter-[5]** summarizes the experimental results , graphs and final conclusion of the project. The error has been presented to show the lag between the actual car movement and the simulated one. The future scope of the project has been briefly discussed with the project's final goal has been achieved.

1.2 Main objective of the experiment

This work has also been motivated by the overwhelming applications of unmanned ground vehicles. The difficulty of designing a safe path for an unmanned vehicle is critical. The path should not only be devoid of obstacles but also take into account vehicle dynamics to be viable. To overcome this, we will be Implementing a path following controller on an unmanned ground vehicle, which has to follow the path and reach a final position under specific conditions. The process of the work is given by:

1. Defining the working environment (start point, end points and obstacles)
2. Creating a matrix of path points from start point to end point using an offline path planner.
3. Considering constraint and to preserve passenger comfort by creating a continuous-differential path from the points using Polyfit linear function and the Sigmoid function.
4. Simulating the NMPC controller on Matlab using ACADO solver on continuous fit path.
5. Extracting the global environment information using camera calibration.
6. Creating rigid body of the Hamster to visualise and fetch the information.

7. Communication is being done using ROS and final Implementation of a path following controller on Hamster has been successfully done.

1.3 Problem facing and solution approach

UGVs are still far from being effective, despite their numerous applications and successes. There could be various reasons for it, but one of the significant challenges is the path following. Designing a safe path for an unmanned vehicle is a crucial problem. The path should be not only devoid of obstacles, but also take into account vehicle dynamics in order to get reliable path. Also, various constraints should be taken into account which can cause discomfort in autonomous driving and apply the controller to get feasible path. This work describes the approach to the problem of path planning for an autonomous vehicle moving in two-dimensional space filled with motionless fix sized obstacles. The problem is based on continuous processing of incoming global information about the environment and the continuous differential optimal path that has to be followed given by an offline path planner. Information about the environment is assumed to be fully known, which means that at every moment, the vehicle knows the coordinates of its target as well as its own coordinates.

1.4 Model Predictive Control (MPC)

Model predictive control (MPC) is a sophisticated form of process control that involves meeting a collection of constraints to achieve the desired result. It is a feedback control algorithm that makes predictions about a process's outputs using a model [5]. It is appeared in the late 1970s and has steadily grown since then. The term model predictive denotes a wide variety of controls rather than a particular form of control technique. It is widely used at the current time in the process industry robot arms, chemical industry, steam generators and applications to regulate a wide range of other procedures, from robotics to clinical anesthesia and many more [5]. MPC can control complex nonlinear systems that are constrained by nonlinear constraints. It operates by using a device model to optimize control signals over a given prediction horizon and then executing a portion of those optimized signals. As a result, it appears to be an exciting and promising solution for overcoming the problems mentioned earlier [6].

1.5 Summary

In this chapter, brief introductory paragraphs have been presented about the UGV with their various applications in various fields. The main objective of the experiment has been briefly summarized. Moreover, the problem faced, and the solution approach taken have been discussed. Furthermore, the short introduction of MPC has been given. In the next chapter, the mathematical formulation and structure of the car model are presented.

2 Mathematical formulation of Car model

2.1 Car Model

An unmanned ground vehicle (UGV) is a piece of mechanized equipment that travels on the ground without the intervention of a human aboard [7].
Hamster is a smart Robot Operating System (ROS) based autonomous ground vehicle for Industry and Academic R&D. It is developed by Congniteam company. It is capable of running Simultaneous Localization and Mapping (SLAM), and Path planning. Hamster robot arrives with Linux (ARM) and ROS pre-installed. Thus, the best method of communication with the robot is through ROS. Hamster platform arrives with various components pre-equipped like: Sensors(360°Light Detection and Ranging (LIDAR), Camera), lithium polymer (LiPo) Battery, 2xRapsberry pi2, DC Motors, WiFi positioning system (WPS) antenna at the back helps to discover the device location and sent it to the motive software [8]. Also, seven-point marker has been placed on the top of the car, which helps to create a rigid body in the motive software, as shown in the Figure (2.1) [8] and (2.2).

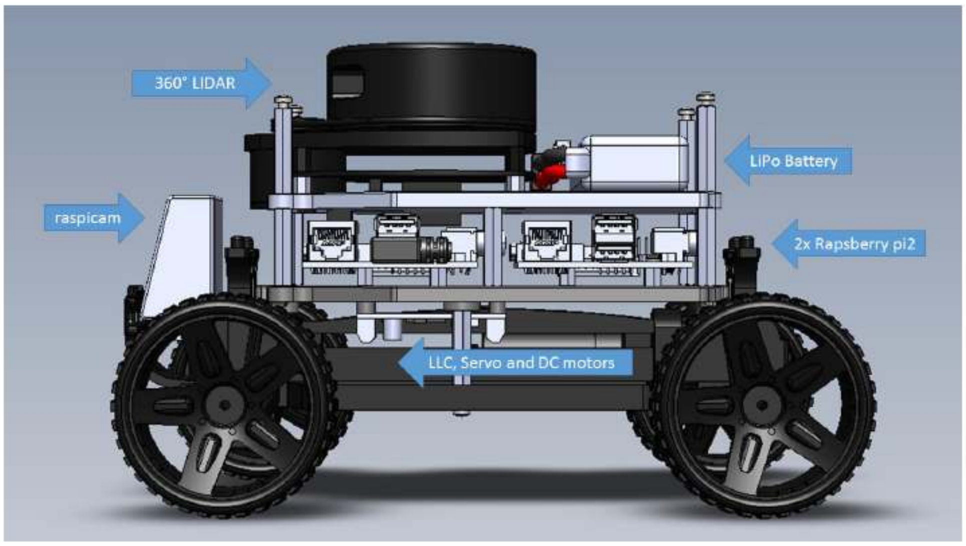


Figure 2.1: Hamster model by Cogniteam

UGVs are being designed for civilian and military applications to perform unique, challenging, and dangerous tasks. They proved to be successful in various situations where using human labor would be too costly, dangerous, or impractical. Research and development of

UGV is often focused on specific applications and are therefore designed accordingly. As shown in Figure (2.2), the hamster with its various components has been depicted.

(a) Hamster

(b) Point marker

(c) WPS antenna

(d) Charger

Figure 2.2: Hamster and their various components

2.2 Mathematical representation of Hamster

The Hamster model can be represented by the kinematic four-wheels model, where the two rear and the two front wheels are lumped together into one rear and one front wheel, respectively. It also follows the mathematical model of kinematic bicycle model represented by the set of equations [9].

$$\dot{x} = v\cos(\psi + \beta) \tag{2.1a}$$

$$\dot{y} = v\sin(\psi + \beta) \tag{2.1b}$$

$$\dot{\psi} = \omega = \frac{v}{l_r}\sin(\beta) \tag{2.1c}$$

$$\dot{v} = a \tag{2.1d}$$

$$\beta = \tan^{-1}[\frac{l_r}{l_f + l_r}\tan(\delta_f)] \tag{2.1e}$$

Where equations (2.1a) and (2.1b) represent the vehicle non-linear translations in x-y plane with respect to the global frame of reference. Equations (2.1c) and (2.1e) represent the heading angle and side-slip angle, respectively. Where l_r and l_f is the distance from the center of the vehicle to the rear wheel and front wheel, respectively. The vehicle's velocity is v, and a is the acceleration of the center of mass, which applies in the same direction as the velocity [9].

Constraints on Hamster

Table 2.1 shows some specific constraints on Hamster provided by the cogniteam company [8].

Table 2.1: Hamsters data

Hamster Parameters	
Parameters	Value
Maximum speed (v_{max}) [m/s]	1.2
Minimum speed (v_{min}) [m/s]	0.1
Steering angle(δ_{fmin})	-0.245rad (-14°)
Steering angle(δ_{fmax})	0.245rad (14°)
Center to rear wheel distance (l_r)[m]	0.15
Center to front wheel distance (l_f) [m]	0.15
Turning degree sensitivity	1°

2.3 NMPC Path following formulation

Nonlinear model predictive control (NMPC) scheme, denoted as Model predictive path following control (MPFC) [10].
We have considered nonlinear system dynamic form : [11]

$$\dot{x} = f(x(t), u(t)) \tag{2.2a}$$

$$y = h(x(t), u(t)) \tag{2.2b}$$

Where $\dot{x} \in \mathbb{R}$, $u \in \mathbb{R}$, & $y \in \mathbb{R}$, represent the state, control vector and the output respectively. The system model is given by the nonlinear continuous time equation in the equation (2.1).
Output path-following refers to the task of tracking a path given by a planner in the output space [12]:

$$P = x, y \in \mathbb{R} || \theta \in \mathbb{R} \Rightarrow x, y = p(\theta) \tag{2.3}$$

Where the scalar variable $\theta \in \mathbb{R}$ is called path parameter.

It is also noted that in path following problems there is typically no strict requirement when to be where on P. In other words, the path parameter (θ) is time dependent, but its time evolution t $\Rightarrow$ θ (t) is not specified a priori.
We use the following general augmented system to formulate the path following problem: [12]

$$\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} f(x,u) \\ l(z,v) \end{pmatrix}, \begin{pmatrix} x(t_0) \\ z(t_0) \end{pmatrix} = \begin{pmatrix} x_0 \\ z_0 \end{pmatrix} \tag{2.4}$$

The path generated by the planner is represented as path parameter (θ) and the formulation is as follows.

$$J = min_u(.) = \int_{t_i}^{t_i+T_p} F(x(\tau), u(\tau) + E(x(t_i + T_p)) \tag{2.5}$$

Cost function J is consider as soft constraints has to be minimized to get an optimal path. The Optimal Control Problem (OCP) is solved repetitively is [12]:

$$J(x(t_k)), z(t_k), \bar{u}(.), \bar{v}(.) \tag{2.6a}$$

subjects to various other constraints:

$$\dot{x} = f(x,u), x(t_i) = X_{sys}(t_i) \tag{2.6b}$$

$$\bar{x}(\tau) \in X, \bar{u}(\tau) \in U \tag{2.6c}$$

$$\bar{z}(\tau) \in Z, \bar{v}(\tau) \in V \tag{2.6d}$$

$$v_{min} \leq v \leq v_{max} \tag{2.6e}$$

$$\delta_{min} \leq \delta \leq \delta_{max} \tag{2.6f}$$

Where equation (2.5) defines the objective function of the Non-linear model based predictive controller path formulation. Equations (2.6) define the various constraints, which have to hold for all t $\in$ [t_k, t_k +T].
MPFC is based on the augmented system described by the equation(2.1). At the core of the MPFC scheme is the repeated solution in a receding horizon fashion at all sampling instants t_k. As common in model predictive control, the system input is obtained via the repetitive solution of an optimal OCP. At each sampling instance t_k = t_o+kδ. Solving (2.6) at time t_k with finite Horizon [t_k, t_k +T] one plans a reference motion x,y(t)$\Rightarrow$ p(θ(t)) $\in P$ also at the same time computes the system inputs to track the formulated path.
The first control signal candidate is plugged into the system, and the system states are measured/predicted in the next simulation step, this process is repeated until certain user defined criterion is achieved. Algorithm 1 explains how MPC works.

Table 2.2: MPC Algorithm

```
Algorithm 1: MPC Algorithm [13]
Result: Optimal Control Sequence $\mathbf{u*} = [u\ (t_k), u(t_{k+1}), ..., u(t_{k+N})]$
for $t_k$ = 0, 1, 2, . . . do ;
    1. Measure/estimate the system state $x(t_k) = x_0$ at the current time $t_k$;
    2. Predict the future behavior using system model $f(x(\cdot); u(\cdot))$ based on the
       current state $x(t_k)$ and the calculated inputs;
    3. Solve a finite horizon OCP by minimize/maximize a cost function respect to
       the system constraints;
    4. Calculate a sequence of optimal control inputs $\mathbf{u*}$, then apply only $u(t_k)$;
    5. Use the receding horizon strategy, whole procedure is repeated, go to step 1.
```

The experiments have been performed using essentially the same parameter as for simulated result. The only difference is in the choice of Weightening Matrices (State and Control parameters).The weightening matrices have been tunned in order to minimize cost function.

2.4 ACADO Solver

In recent years, the number of Linear and Nonlinear MPC applications for fast-dynamic systems has increased significantly as computing power has increased. There are many tools and software packages available to solve linear and NMPC like CasADi, Automatic Control and Dynamic Optimization (ACADO) [14].
ACADO is an open-source platform for dynamic optimization and automatic control. It provides a general framework for implementing a wide range of direct optimal control algorithms, such as model predictive control, state and parameter estimation, and robust optimization. To solve NMPC path formulation of UGV, we proposed to use ACADO Toolkit, which provides a MATLAB interface to generate C++ code for NMPC [15]. It implements the real-time iteration scheme with a different algorithm to solve the OCP and incorporates it with the efficient numerical integrator in auto-generated C-code [14].

2.5 Summary

This chapter includes a brief explanation of the mathematical formulation and structure of the car model by cogniteam. Moreover, NMPC path following formulation and algorithm has been discussed briefly. ACADO solver software package, which has been used to solver NMPC, has also been explained briefly. In the next chapter, simulation results of simulation results of MPC path following problem will be illustrated.

3 Simulation Result

3.1 Introduction

In this chapter, the simulation of the environment and various parameters have been presented. The simulation of the global environment(the environment is entirely known or self-designed) consists of a big prohibited region and three obstacles along the path given by the offline planner. The safe margin from the obstacle point is 0.20 m
As can be seen in the Environment Setup data table [3.1].

Table 3.1: Environment Setup data table

Simulation Environment	
Parameters	Value
Distance in X-direction [m]	8.36
Distance in Y-direction [m]	3.3
Initial position	[0,0]
Final position	[8.36,3.3]
Area of prohibited region [m^2]	6.72x1.45
Position of obstacle 1	[1.52,1.03]
Position of obstacle 2	[3.41,1.3]
Position of obstacle 3	[7.68,3.04]
Safe margin [m]	0.20

3.2 Simulation of Environment

Figure (3.1) shows the simulation environment as per the table [3.1] and a generated green path from an offline planner. This green path is the movement of the point position, which must be followed by our Hamster. First, we try to fit the path in simulation and then implement it on our Hamster.

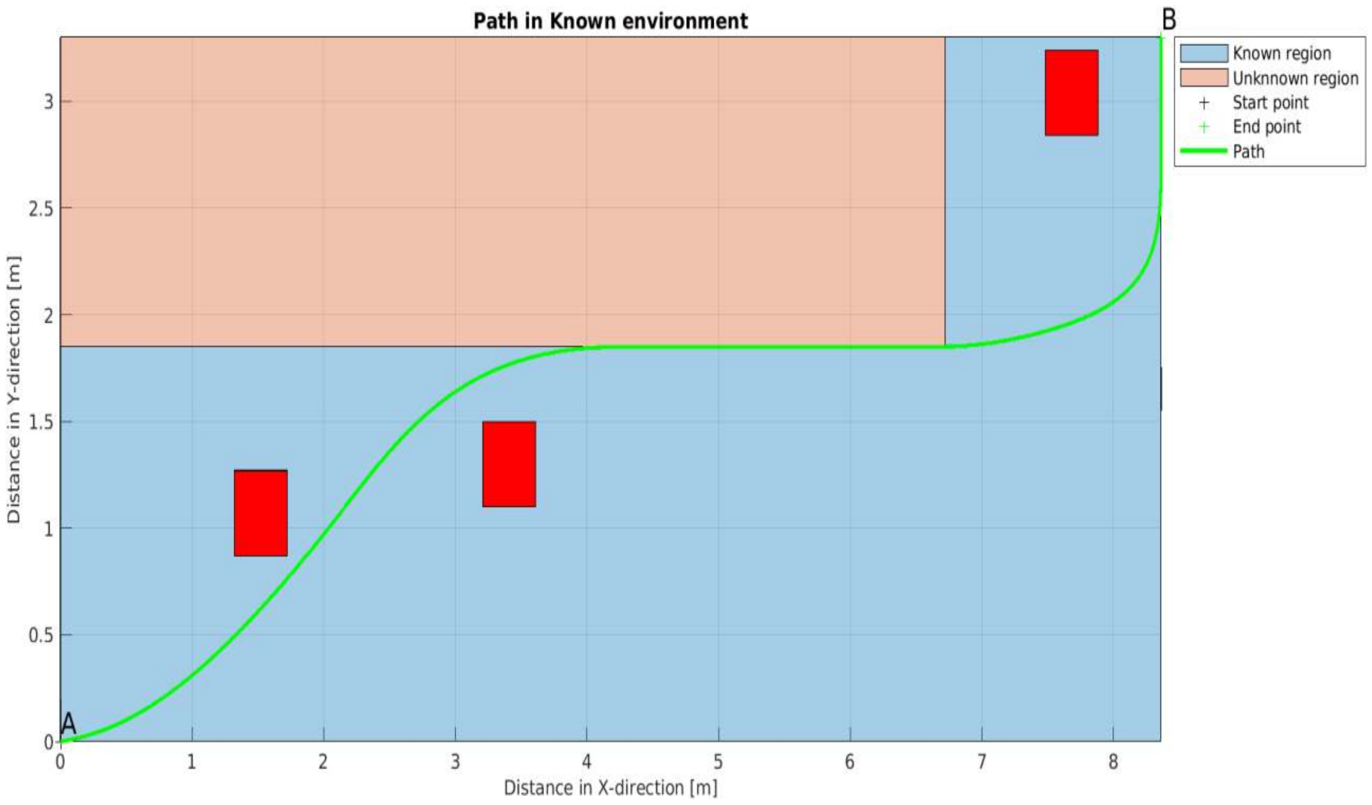


Figure 3.1: Path planner simulated environment

3.2.1 Simulation Results

To achieve the fitting path, we first took the optimal path matrix given by an offline planner and used the poly-fit function to get a set of the linear equation along in x and y direction separately. Using the sigmoid function (Relu function) at the junction of two adjacent linear equation approximations given by the poly-fit function, we made it a continuous differential path which is used as the reference signal while solving Nonlinear MPC problems using an ACADO solver. The final fitted path represented in blue in the Figure (3.2) was achieved by formulation our MPC path following problem. By running the simulation on Matlab and adjusting the weighting matrix to make the simulated result match the optimal path given by the offline planner, we achieved the final blue path.

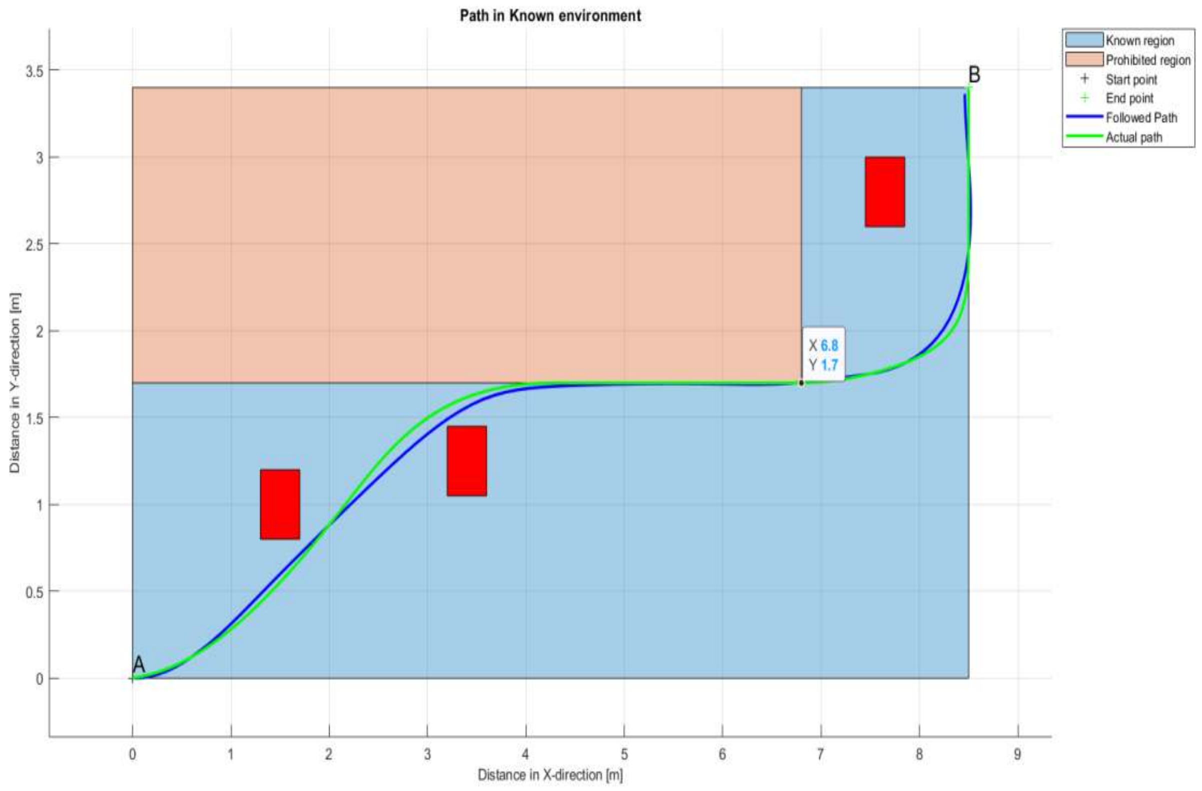


Figure 3.2: Fitted path in simulated environment

Figure (3.3) depicts the evolution of the velocity and time. The velocity of the UGV is gradually changing as time passes. The MPC controller recognizes the constraints and accelerates the UGV to the maximum permissible velocity of 1.2 [m/s]. As the car approaches the target, the velocity decreases until it reaches zero at the target point. Also, we can see that the velocity [m/s] and the steering angle [degree] are two control signals, have to stay within [0, 1.2 m/s] and [-14°, 14°] respectively. The velocity is gradually increases to its maximum allowable velocity and then decreases to zero as the model approaches to its final point. The steering angle has a dramatic change over the journey can be seen in the figure (3.3)
In Figure (3.4) we can see the time it takes for NMPC to calculate the optimal control signal at each time steps.

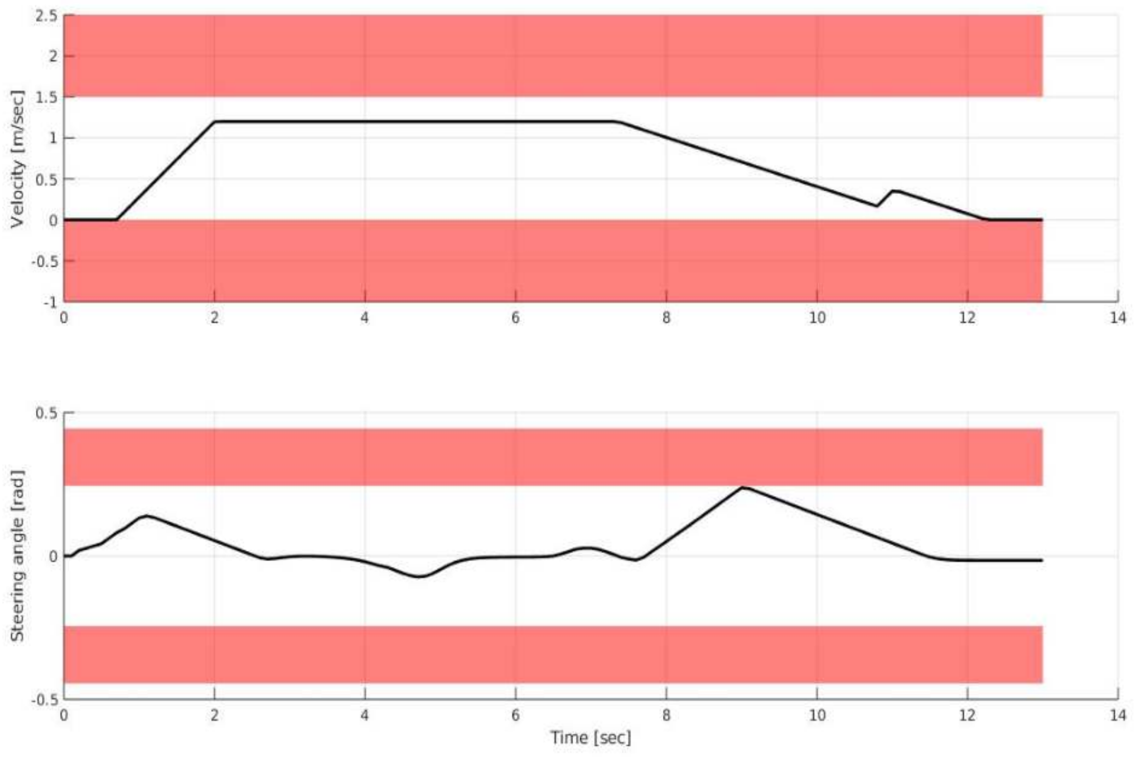


Figure 3.3: Change in Velocity and Steering in simulated time interval

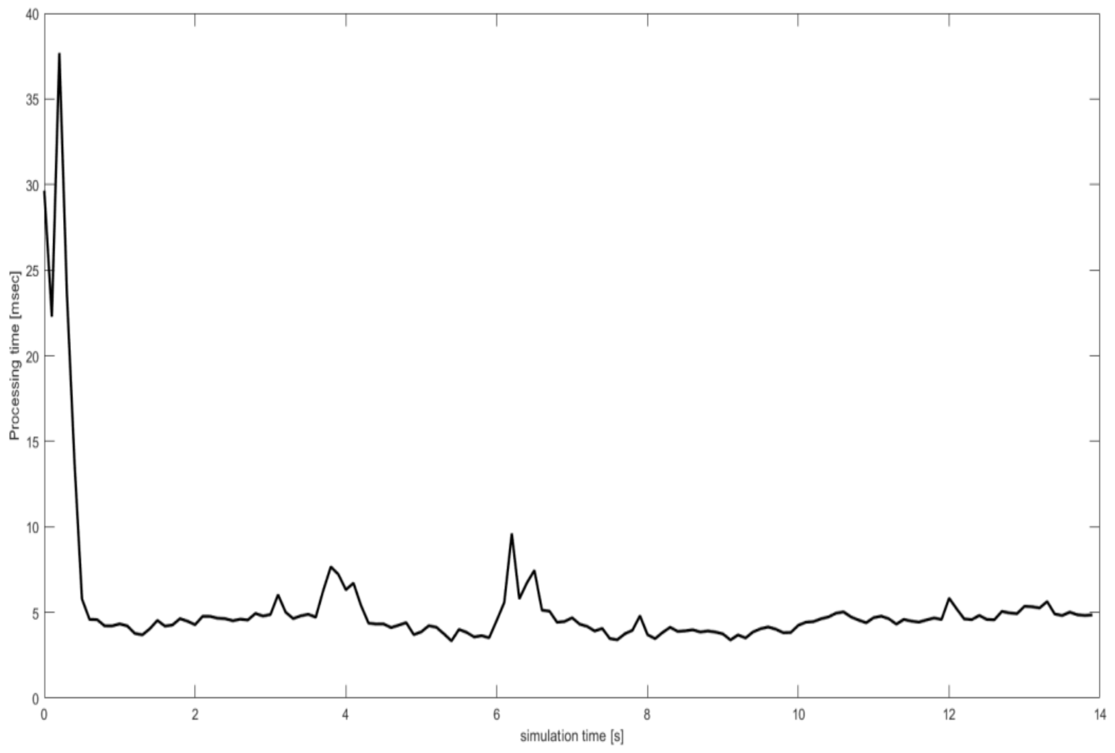


Figure 3.4: Processing time of MPC

3.3 Summary

This chapter presents the simulation of the environment. Also; a simulation results for the adopted MPC path following formulation explained in equation (2.5) were shown. From the aforementioned results, it is clear that the car can follow the path given by the planner while respecting the physical constraints on velocity and steering angle. In the next chapter,the working environment will be presented and calibrated before being used in the experiment. Moreover, several processes of implementing the final output have been explained in detail.

4 Experimental setup for Unmanned Ground Vehicle

4.1 Introduction

Roboticists have developed a range of technologies that could use in industries in recent years. In order to obtain a good grasp of the surroundings, i.e., the industry's surroundings must be captured and known completely. This makes it easier for the robots to navigate across the surroundings. Such environment is called the global environment [16].
Figure (4.1) shows the global environment setup which is designed in the Research lab at Otto von Guericke University in the Chair of Institute for Automation Technology. The design is drawn using tape as per the simulation setup. The environment is fully captured using nine cameras.

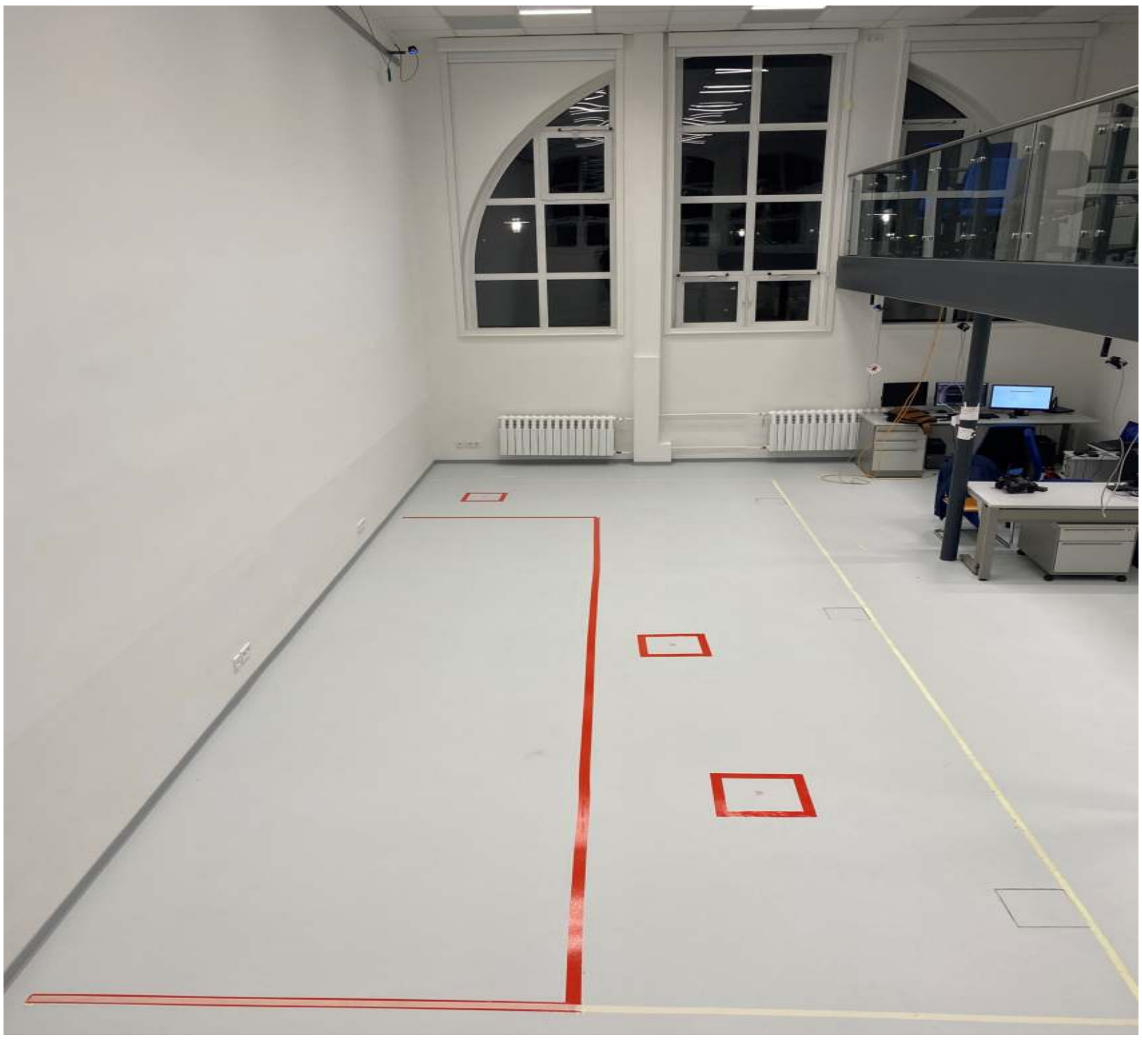
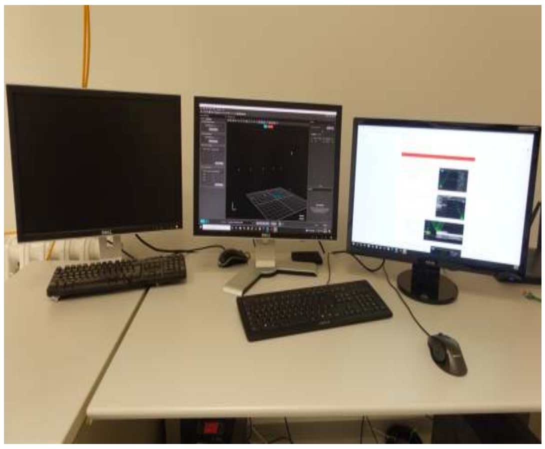
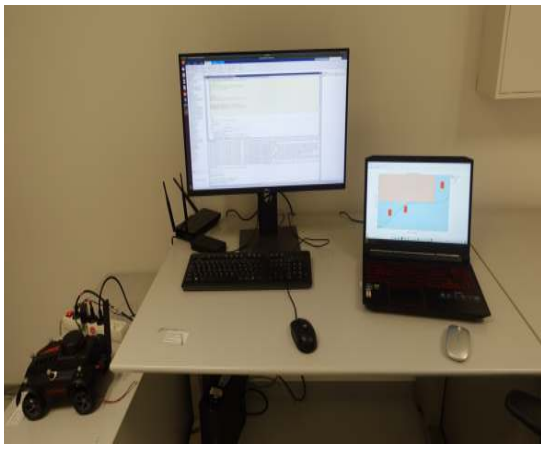


Figure 4.1: Global environment setup

Figure (4.2) shows our ground station, where the two PC are being used. One is where the Motive Software is installed(Window OS) and the other with Robot Operating System (ROS) and Matlab installed(Linux OS).


(a) Positioning unit


(b) Control unit

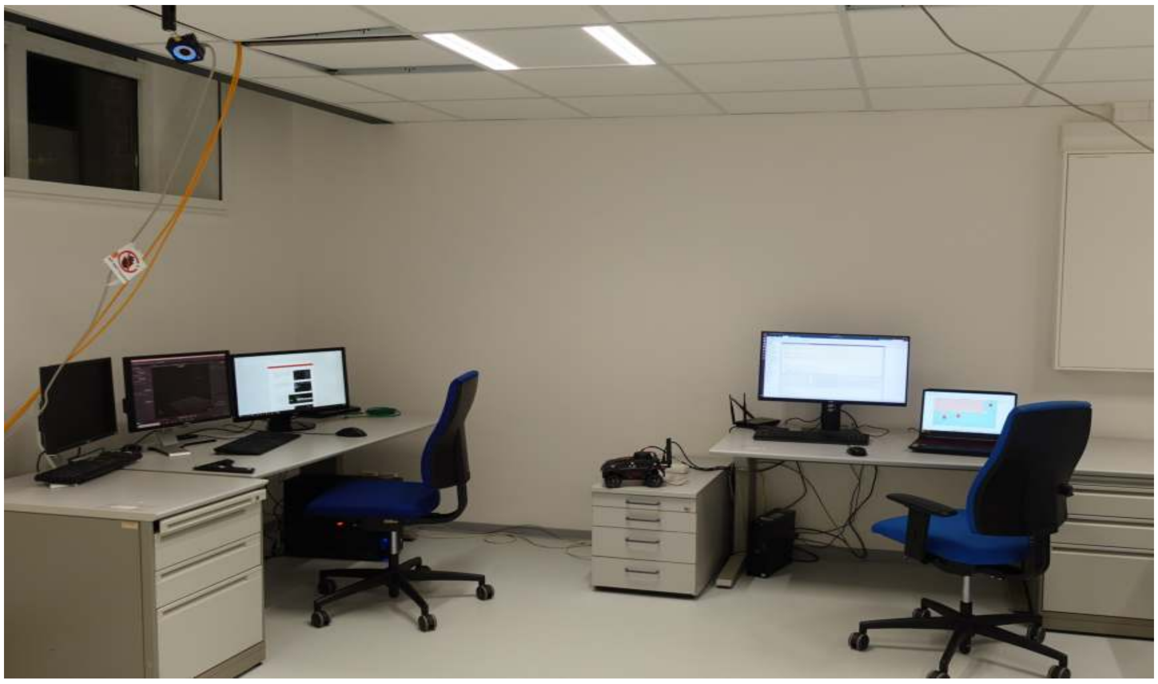


Figure 4.2: Ground station

4.2 Camera system

Figure (4.3) shows the cameras system fully captures the environment, and the data from the cameras is being fetched using motive software. Motive software is the software platform designed by OptiTrack to control motion capture systems for various tracking applications. Motive allows the user to calibrate and configure the system and provides interfaces for both capturing and processing 3D data [17]. The motive software visualizes the live position and coordinates of the cameras as shown in Figure(4.4). Also, the cameras can be seen in the environment picture; they are mounted on the wall, facing the working areas.

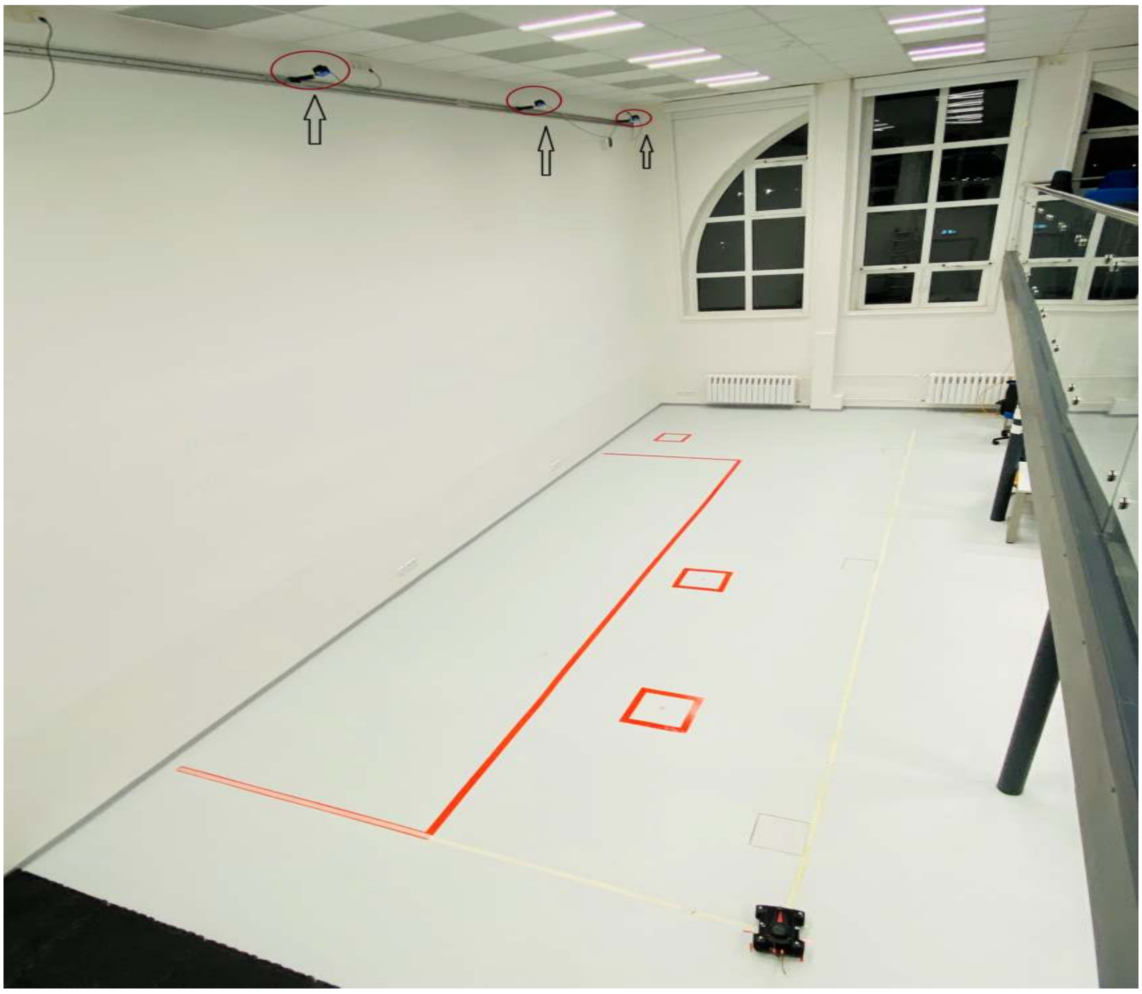
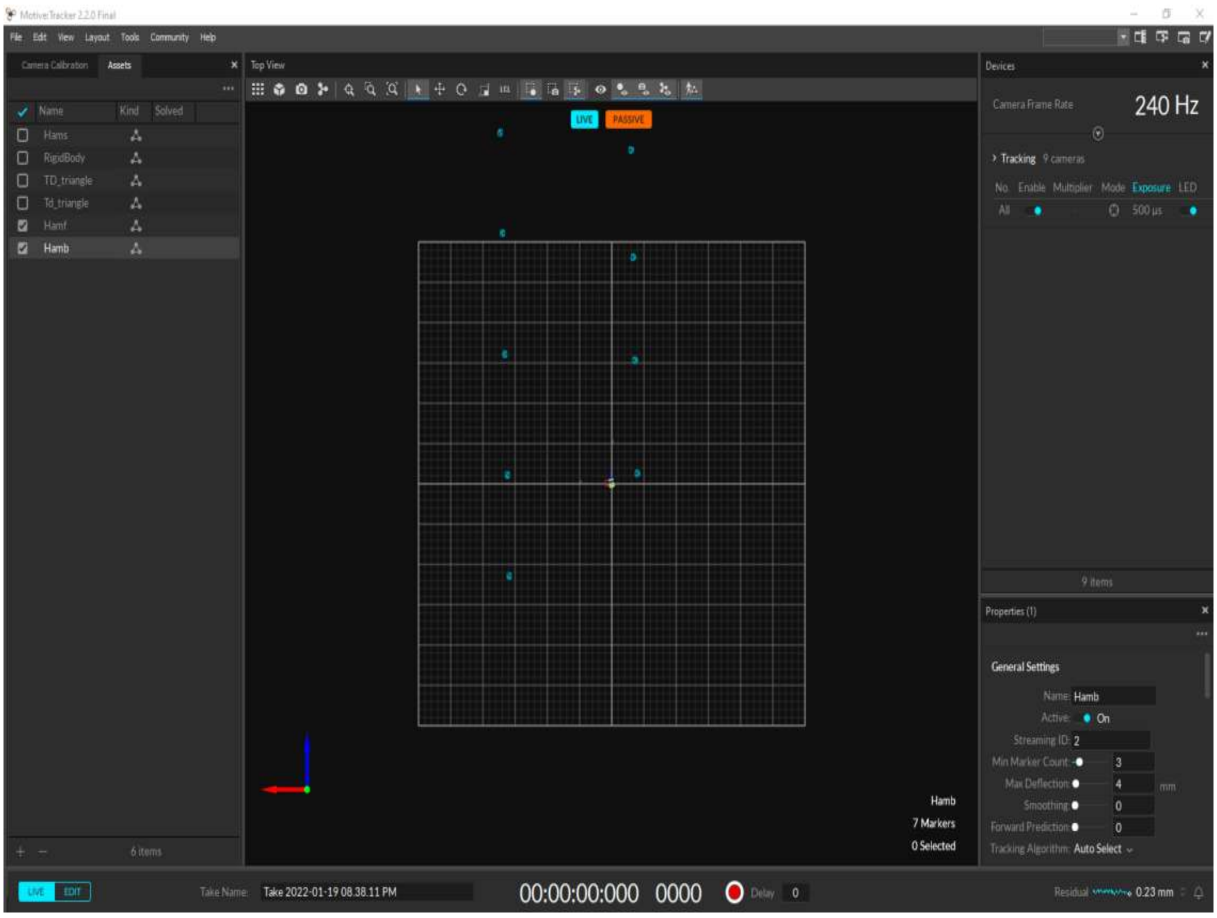


Figure 4.3: Working environment captured by Cameras



✓	Name	Kind	Solved
☐	Hams	⁂	
☐	RigidBody	⁂	
☐	TD_triangle	⁂	
☐	Td_triangle	⁂	
☑	Hamf	⁂	
☑	Hamb	⁂	

6 items

Hamb
7 Markers
0 Selected

Camera Frame Rate 240 Hz

Tracking 9 cameras

No.	Enable	Multiplier	Mode	Exposure	LED
All	●			500 µs	●

9 items

General Settings
Name: Hamb
Active: On
Streaming ID: 2
Min Marker Count: 3
Max Deflection: 4 mm
Smoothing: 0
Forward Prediction: 0
Tracking Algorithm: Auto Select

Take Name: Take 2022-01-19 08.38.11 PM
00:00:00:000 0000
Delay 0
Residual 0.23 mm

Figure 4.4: Camera calibrated view in the Motive software

Camera calibration process

A good estimation of the camera parameters is required to measure the planar objects and surroundings correctly. The images grabbed are affected by radial and/or tangential distortion or could be due to lousy camera calibration. As can be seen in the Figure (4.7), the marker point cloud is detected while doing the calibration, which varies with the points detected by every camera. The steps of calibration are given below.

1. Put the USB Activation key in the PC to activate and open Motive Software.
2. In Motive Software go to View→Camera Calibration Pane and click on Start wandering.
3. Use the Calibration Wand (CW – 500) and move it across the area that you'll use for the experimentation purpose. Take at least 8000 points for each camera and cover all the regions of your experiment.
4. Now, we have to define the Origin for the Calibration file generated. In order to do so, put the Calibration Ground Square (CS – 2000) at the location in the experiment region you want to take as origin.

5. After completing this process, click on calculate and Apply. The Calibrated file generated will have to be saved in your directory.

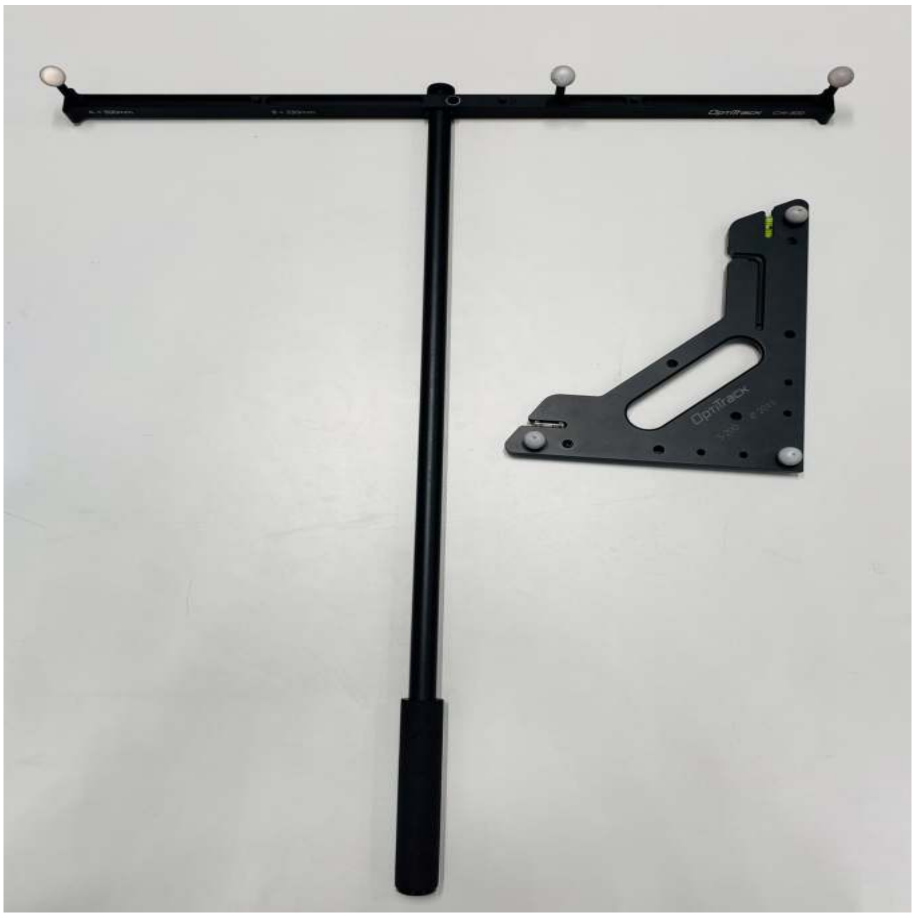
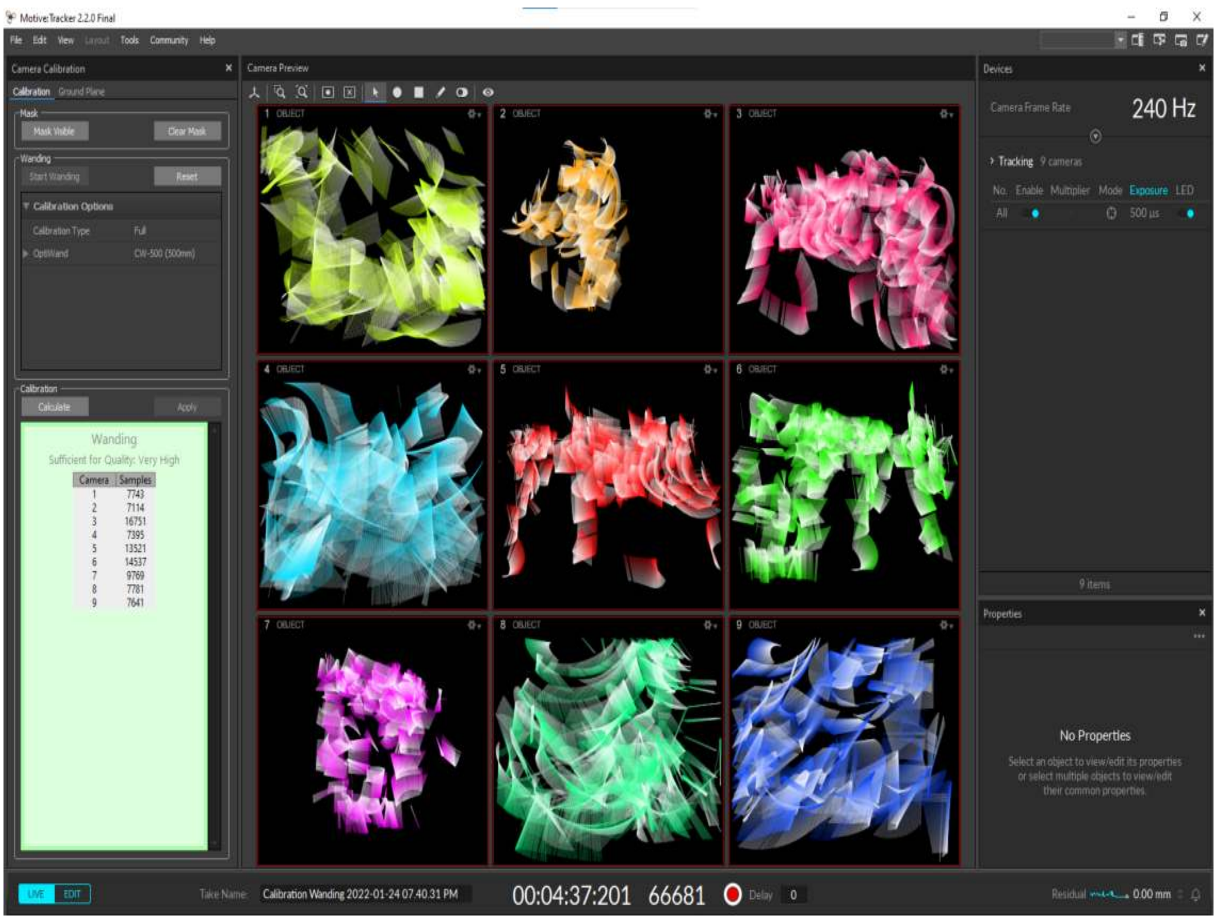


Figure 4.5: Calibration Wand (CW–500) and Calibration Ground Square (CS–2000)

Figure 4.6: USB with Motive files



Calibration Options
Calibration Type Full
OptiWand CW-500 (500mm)

Wanding
Sufficient for Quality: Very High

Camera	Samples
1	7743
2	7114
3	16751
4	7395
5	13521
6	14537
7	9769
8	7781
9	7641

Camera Frame Rate 240 Hz
Tracking 9 cameras
9 items

No Properties
Select an object to view/edit its properties or select multiple objects to view/edit their common properties.

Take Name: Calibration Wanding 2022-01-24 07.40.31 PM 00:04:37:201 66681 Delay 0
Residual 0.00 mm

Figure 4.7: Marker point cloud

4.3 Rigid body creation

Process to create rigid body

1. Put the USB Activation key in the PC to activate and open Motive Software.
2. Select the calibration file you have created in the previous steps from file→ Open.
3. Put the Hamster(UGV) in the Experimental Area and create two rigid bodies from the markers on the Hamster(UGV). The first body created is from the front three markers to represent the front of Hamster(UGV). It can be created by selecting the three markers in motive software and right-clicking the mouse and then selecting create the rigid body. You can give it the name “Hamf”.
4. Similarly, create a rigid body from the remaining markers on the Hamster(UGV) and name it “Hamb” representing the back of the Hamster(UGV).
5. Go to View→Assets Panel and select the two rigid bodies created.

In the below Figure (4.8) we can see the Hamster as rigid body has been created. Also in the Figure (4.9) we can see the Hamster is detected in the camera 1 , 2 and 3.

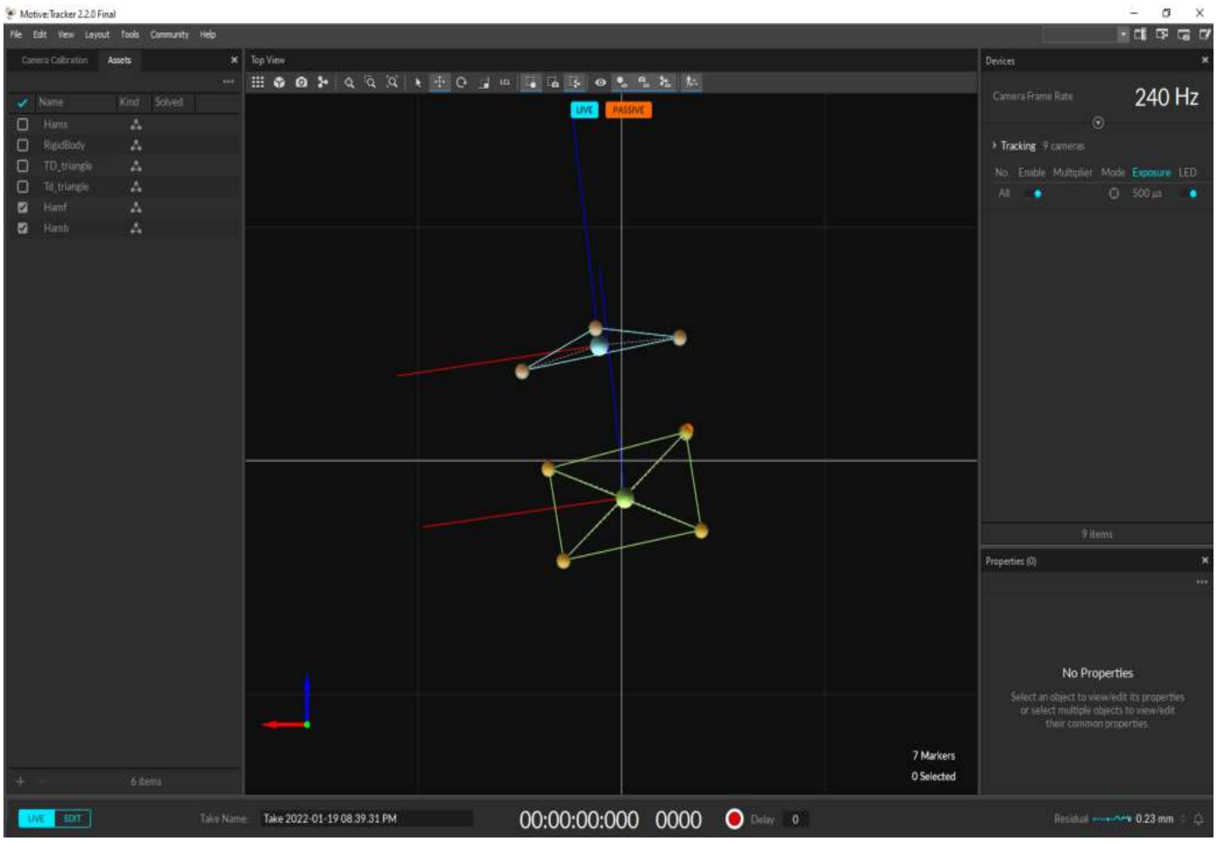

Motive:Tracker 2.2.0 Final
File Edit View Layout Tools Community Help
Camera Calibration Assets
Name Kind Solved
Hams
RigidBody
TD_triangle
Td_triangle
Hamf
Hamb
6 items
Top View
LIVE PASSIVE
7 Markers
0 Selected
Devices
Camera Frame Rate 240 Hz
Tracking 9 cameras
No. Enable Multiplier Mode Exposure LED
All 500 µs
9 items
Properties (0)
No Properties
Select an object to view/edit its properties or select multiple objects to view/edit their common properties
LIVE EDIT
Take Name: Take 2022-01-19 08.39.31 PM
00:00:00:000 0000
Delay 0
Residual 0.23 mm

Figure 4.8: Rigid body of Hamster created in Motive

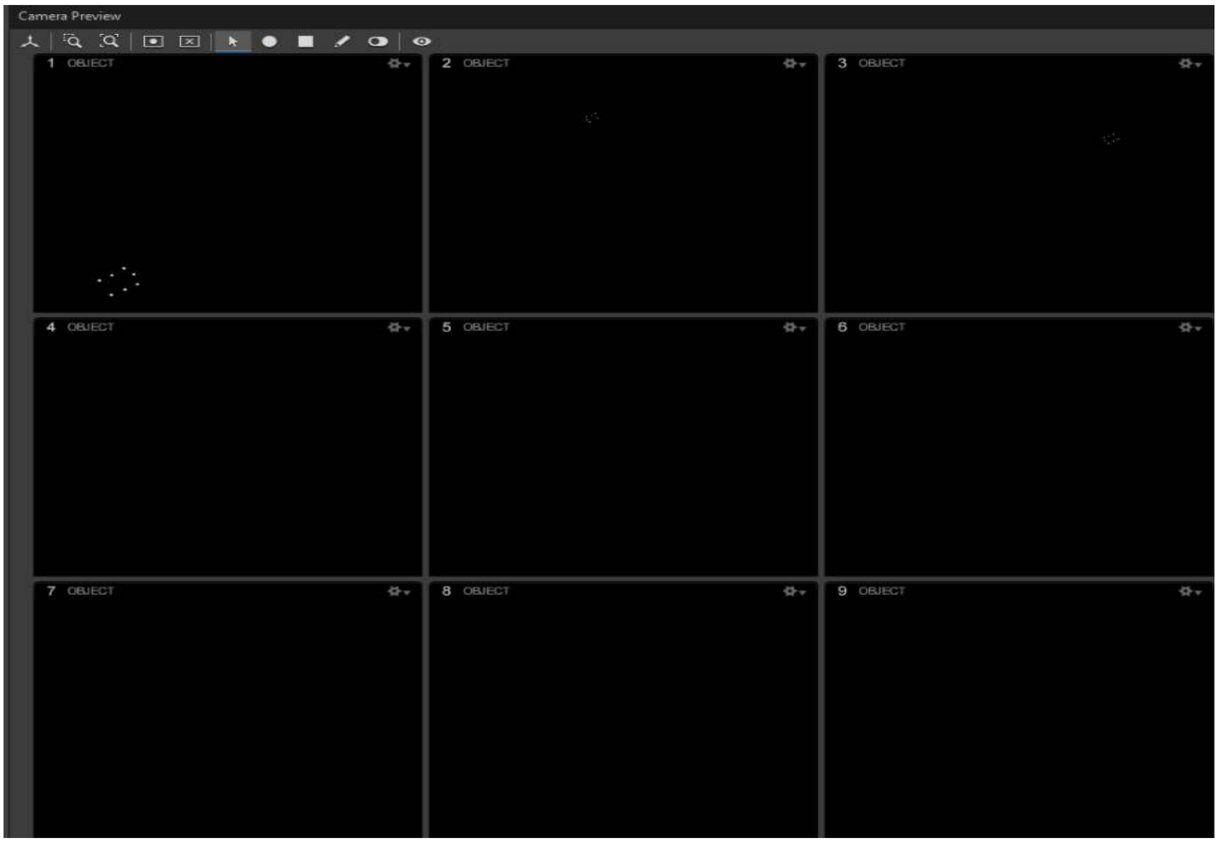


Figure 4.9: Cameras and rigid body in the motive frame

4.4 Summary

This chapter discussed the global environment, how the physical environment has been created and all the parameters of the real environment have been described. A brief introduction of camera system and Motive from OptiTrack described. Moreover, the camera calibration process and the process of creating rigid body has been listed out. In the next chapter, we will be implementing the work in the actual environment on UGV and the final outcomes will be discussed. At end the final path of running Hamster is filmed and the video link will be provided.

5 Experimental Results

5.1 Introduction

The nonlinear mathematical model of UGV and various physical constraints on it have been taken to simulate our model and the Non-linear MPC path following problem has been solved. The aim of the Hamster is to follow the fitted path and reach the target position under the specified position and velocity constraints while avoiding the obstacles and prohibited region. Finally, the behavior of the Hamster during the experiments is documented in the videos available at Otto-von-Guericke-University (at IFAT chair) YouTube channel. Table (5.1) shows the Environment and Hamster data constraints, where the initial and final car velocity is same, given by v. Whereas, the initial and final car positions are stated by A, B. Also, the minimum and maximum speed of the Hamster is given by u.

Table 5.1: Environment and Hamster data constraints

Environment and Hamster data	
Parameters	Value
Initial car position $[A]$ [m]	$[0, 0]$
Final car position $[B]$ [m]	$[8.36, 3.28]$
Velocity $[v]$ [m/s]	$0.1 < v < 1.2$
Initial & final car velocity $[v]$ [m/s]	0
Obstacle_1 position	$[1.52, 1.03]$
Obstacle_2 position	$[3.41, 1.31]$
Obstacle_3 position	$[7.68, 3.04]$
Prohibited region	$[01.75; 6.721.85; 6.723.28; 03.28]$
Prediction horizon P_h	50
Sample time (T_s)[sec]	0.1
Simulation time (T_f)[sec]	14

The system dynamics under various constraints have been simulated using MATLAB-ACADO software packages.The final path of the Hamster has been plotted as can be seen in Figure(5.1).

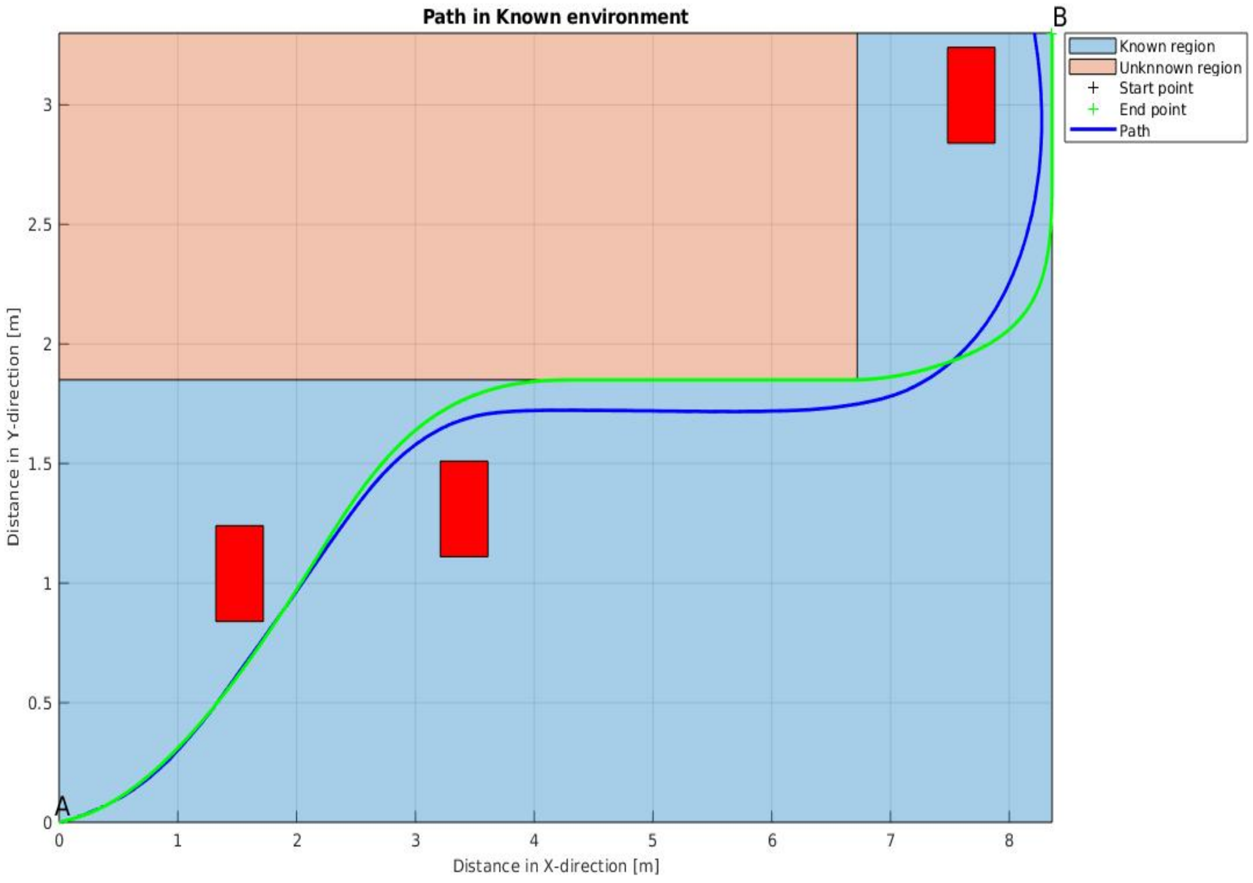


Figure 5.1: Path of hamster in real environment

The Hamster starts from point A and reaches the Final Position at point B. Meanwhile, the actual path taken by the Hamster(UGV) is almost the same as the optimal path generated using an offline path planner except for the slight deviation in the middle. The reason for that is the offline planner used a linear dynamic car model, which considered the vehicle as a point object with no dimension. In contrast, while solving the Nonlinear MPC problem with ACADO, we used a Nonlinear vehicle dynamics model that considered the actual size of the mobile robot. Furthermore, as the optimal path is made for a point object and the real UGV has a rigid body with some dimension hence to avoid the body overlapping the prohibited region, we deliberately kept the reference trajectory slightly below and above the reference path. Despite that, the Hamster(UGV) has perfectly avoided the obstacles and forbidden region.

Figure(5.2) depicts the evolution of the velocity and time. Velocity and Steering angle are constrained control signals variables, they stays within the bound for the entire time. MPC controller is aware of the constraints, the velocity of the UGV is drastically changing as time passes, and it goes to the maximum allowable velocity of 1.2 [m/s], and once the car approaches the target, the velocity reduces and goes to zero when the car reached the target. Moreover, the steering angle has to stay within the limit [-14°, 14°] as time

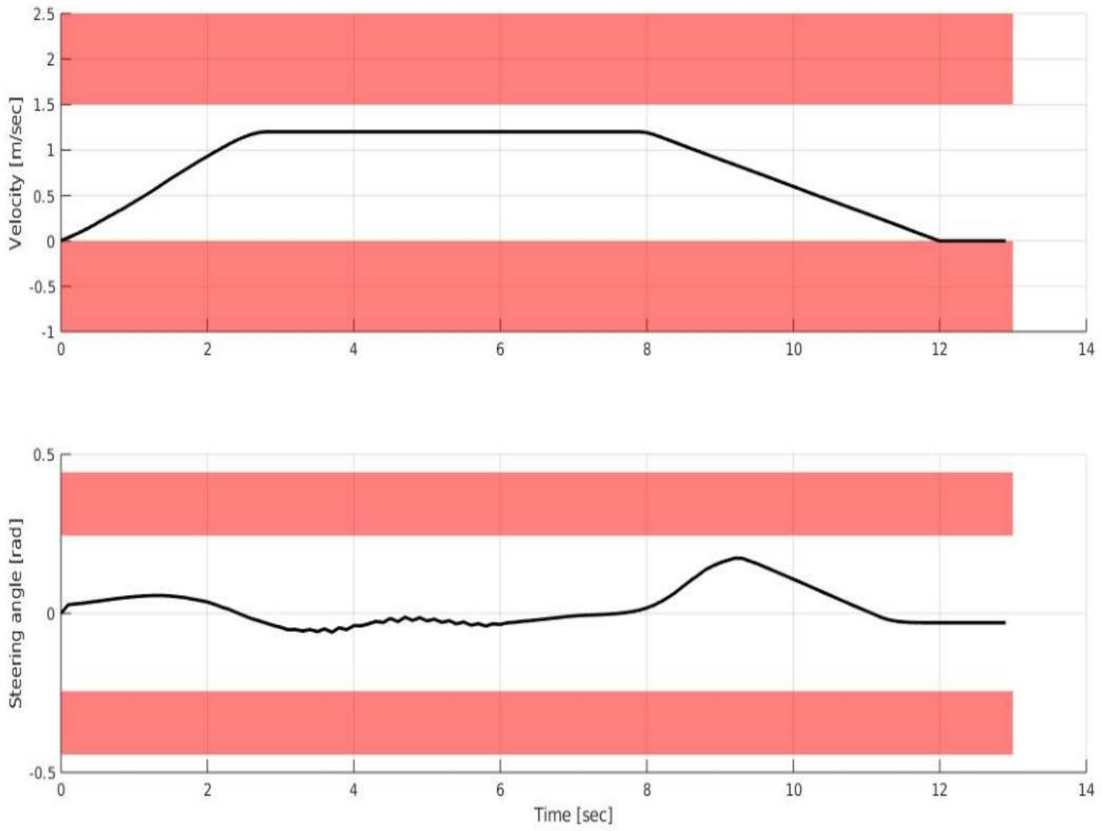


Figure 5.2: Change in Velocity and Steering time interval

passes despite the change in velocity. At time 2 sec and 9 sec there is significant change in steering angle, which is because the at this time the velocity increases or decreases with the maximum rate and Hamster tries to follow the path by adjusting the steering angle. Finally, As shown in the Figure(5.1), the absolute error between the green path, which is the actual path and the blue path, which is the path followed by Hamster. The error between the both is in the range of 0-7 cm, which is about 2%. This is well within the accepted range.

5.2 Summary

This chapter addresses the actual implementation of nonlinear MPC on Hamster considering nonlinear system dynamics and its physical limitation. The actual output path followed by the Hamster and the initial path from the offline planner has been plotted in the real environment graph. It is also shown that the MPC control signals respect the physical constraints provided by the manufacturer. The error has been briefly discussed, and finally, the UGV reached the target point while strictly adhering to the velocity and steering angle limit. With this, the desired output of the project has been achieved.
Finally, given the wide range of applications and growing demand, it is determined that UGVs are the future. As a result, making it more and more feasible would be an incredible feat that would provide a comfortable existence for future generations.

Bibliography

[1] John Ebken, Mike Bruch, and Jason Lum. Applying unmanned ground vehicle technologies to unmanned surface vehicles. In *Unmanned Ground Vehicle Technology VII*, volume 5804, pages 585–596. International Society for Optics and Photonics, 2005.

[2] A UGV robot is patrolling the streets during the covid lockdown,2020 in Tunis, North Africa website. `https://www.inceptivemind.com/tunis-ugv-robot-p-guard-patrolling-streets-lockdown/12805/`. Accessed: 2020-04-20.

[3] Weaponized UGV Demonstrated During Live-Fire Exercise in USA WEBSITE mike ball. `https://www.unmannedsystemstechnology.com/2019/06/weaponized-ugv-demonstrated-during-live-fire-exercise/`. Accessed: 2019-10-19.

[4] Themanufacturer Jonny Williamson. How is slam software powering the next-generation of autonomous industrial robots ?, 2013 https://www.themanufacturer.com/articles/slam-software-powering-next-generation-autonomous-industrial-robots/.

[5] E. F. Camacho and C. Bordons. *Model Predictive control.* Springer London, 2007.

[6] Huckleberry Febbo, Jiechao Liu, Paramsothy Jayakumar, Jeffrey L Stein, and Tulga Ersal. Moving obstacle avoidance for large, high-speed autonomous ground vehicles. In *2017 American Control Conference (ACC)*, pages 5568–5573. IEEE, 2017.

[7] Douglas W Gage. Ugv history 101: A brief history of unmanned ground vehicle (ugv) development efforts. Technical report, Naval command control and ocean surveillance center, 1995.

[8] Cogniteam. V7 Hamster, a car-like-Robot, by cogniteam. `https://www.hit.ac.il/.upload/HamsterDeveloperManual-REV4_v1.pdf`, 2019.

[9] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.

[10] Shuyou Yu, Xiang Li, Hong Chen, and Frank Allgöwer. Nonlinear model predictive control for path following problems. *International Journal of Robust and Nonlinear Control*, 25(8):1168–1182, 2015.

[11] Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[12] Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2016.

[13] PhD thesis , Mohamed Ibrahim. Design and real-time implementation of model predictive control for flight systems in presence of uncertainties. In *Model Predictive Control*, page 170, 2020.

[14] ACADO, Acado toolkit website. `https://acado.github.io/matlab_overview.html`. Accessed: 2014-09-24.

[15] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[16] Hongbin Ren, Sizhong Chen, Lin Yang, and Yuzhuang Zhao. Optimal path planning and speed control integration strategy for ugvs in static and dynamic environments. *IEEE Transactions on Vehicular Technology*, 69(10):10619–10629, 2020.

[17] Motive optitrack. `https://optitrack.com/`. Accessed: © 2022 NaturalPoint, Inc. DBA OptiTrack.