Fall 2021

Nonlinear Intelligent Model Predictive Control of Mobile Robots

Benjamin Albia

NONLINEAR INTELLIGENT MODEL PREDICTIVE CONTROL OF MOBILE ROBOTS

by

Benjamin Albia

Bachelor of Science
College of Charleston, 2012

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Aerospace Engineering

College of Engineering and Computing

University of South Carolina

2021

Accepted by:

Yi Wang, Director of Thesis

Bin Zhang, Reader

Tracey L. Weldon, Interim Vice Provost and Dean of the Graduate School

DEDICATION

To my wife Emmie, thank you for daring me to better myself, and facing the resulting trials with me; and my parents, thank you for your endless support.

ACKNOWLEDGEMENTS

I would like to take this opportunity to recognize a few of the most important sources of help during the completion of this Thesis.

Thanks to the Army Research Laboratory for the opportunity to work on this project, which has proven as enjoyable as it has been challenging.

To my Advisor Dr. Yi Wang, thank you for giving me both the opportunity and the confidence to succeed academically. I admire your passion for your research, but more than anything I admire your ability to help your students realize the importance of passion. My ultimate professional goal is to one day unlock someone else's potential as you have done for me and so many other students.

To my Mentor Dr. Seong H. Hong, thank you for guiding every aspect of my development, constantly demonstrating healthy research habits and principles. I can always count on you to both encourage and challenge me, and you are more responsible than anyone for helping me turn my ideas into reality.

To my colleague Tristan Kyzer, thank you for struggling alongside me. It has been a privilege to work with you, but I place even more value on your friendship, and your support that I have counted on in areas outside of the work.

To my colleague Christian Fletcher Turner, thank you for cultivating an atmosphere of healthy competition, and pushing our research group to be its best.

Thanks for contributions from Michail Kalaitzakis, Junlin Ou, Jacob Rains and Thomas Johnson.

ABSTRACT

This thesis presents a framework for an artificial neural network (ANN) model-based nonlinear model predictive control of mobile ground robots. A computer vision analysis module was first developed to extract quantitative position information from onboard camera feed with respect to a prescribed path. Various strategies were developed to construct nonlinear physical plant models for model predictive control (MPC), including the physics-based model (PBM), the ANN trained on PBM-generated data, the ANN trained on test-captured data, and the ANN initially trained on PBM-generated data and then retrained with captured data. All the models predict physical states and positions of the robot in the future horizon using the current control signals and the information obtained by the computer vision analysis. Model predictive controllers based on these models and real-time optimization were also developed, and were able to determine optimal control signals in the future horizon, enabling the robot to follow the designated path. Path following experiments were then conducted on a test track to evaluate nonlinear MPC performance. It is found that both the PBM and the ANN model allow accurate path following through nonlinear MPC with an error metric of 1 cm (i.e., the average deviation of the robot from the designated path). More importantly, incorporating test-collected data into the ANN retraining to consider non-ideal factors not captured by PBM further improves the path following accuracy by 30.0%. The developed framework paves the way towards a new paradigm to develop autonomous robots with anomaly mitigation and system resilience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

α	Ground robot center of rotation
Z	Ground robot line error (distance from prescribed path at horizon)
θ	Ground robot angle error (orientation relative to prescribed path)
v	Ground robot linear velocity
ω	Ground robot angular velocity
c	Path curvature
H	Horizon distance (distance from ground robot center of rotation to path following horizon)
ε	Overhead camera error (distance between ground robot center of rotation and nearest point on prescribed path)

LIST OF ABBREVIATIONS

ANN ..... Artificial Neural Network

MPC ..... Model Predictive Control

PBM ..... Physics-based Model

RMS ..... Root-mean-square Error

CHAPTER 1

INTRODUCTION

Artificial neural networks (ANNs) are models which can make classification and/or regression predictions, given specific input data. They were originally inspired to replicate the neural network found in the human brain, and have seen widespread use in machine learning. Computer vision has an established practice for robotic path following applications (Surya Prakash et al, 2012). The use of ANNs has the potential to further enhance robotic visual servoing. This project utilizes ANNs to control a ground robot for path following.

The objective of this research is to develop an ANN which can replace the Physics-based Model (PBM) for model predictive control (MPC) in ground robot path following with the ultimate goal of improving accuracy by using real data captured by the ground robot. To this end, ANNs are trained using data generated by the PBM, as well as data captured during operation of the robot.

The motivation for this research is to examine the viability of neural network models in control applications. A possibility is the use of ANNs to detect and mitigate physical anomalies, which cannot be accounted for by physical models.

The remainder of this thesis is organized as follows. Chapter 2 covers the robotic hardware, the PBM for path following, the computer vision analysis, and both the robot's onboard camera as well as an overhead camera used to measure path following accuracy

and assess control performance. Chapter 3 presents structure, training and data of the various ANNs. Chapter 4 discusses the specifics of MPC used for path following. Chapter 5 compares the path following performance of the ANNs against the baseline PBM. Chapter 6 concludes the thesis with a technical summary and future work.

CHAPTER 2

COMPUTER VISION ANALYSIS

The ground robot is equipped with a camera, which provides it real time visual information necessary to follow a prescribed path. Path following is accomplished by minimizing the distance between the ground robot and the path, and by orienting the robot in the correct direction relative to the path. This requires a physical model which predicts the changes in the robot's distance from and relative orientation to the prescribed path given various commanded control signals.

2.1 GROUND ROBOT BUILD

The ground robot build is a modified TurtleBot3 Waffle Pi kit. It is controlled by a Jetson TX2 computer, which communicates with its servo motors via the OpenCR microcontroller, an Open-source Control Module for ROS. The ground robot is equipped with an e-CAM132_TX2 camera and is powered by two lithium polymer batteries.

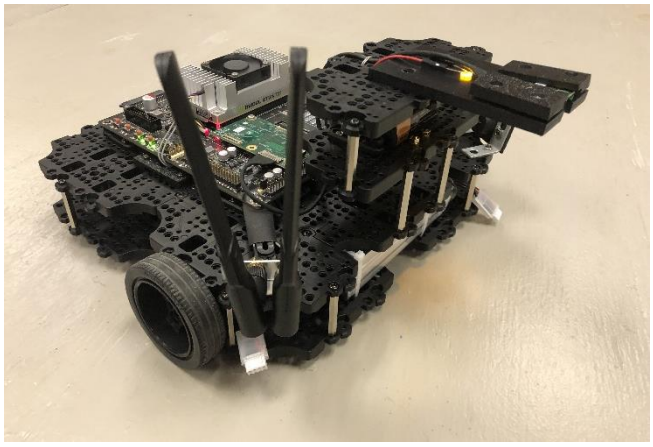


Figure 2.1 Path following ground robot

The robot's two separable layers are shown in Figure 2.2: the bottom layer (blue) houses the fore left and right servo motors and wheels, the two aft ball casters, the computer and motor batteries and the OpenCR, and the top layer (orange) houses the onboard computer, camera and headlight equipment.

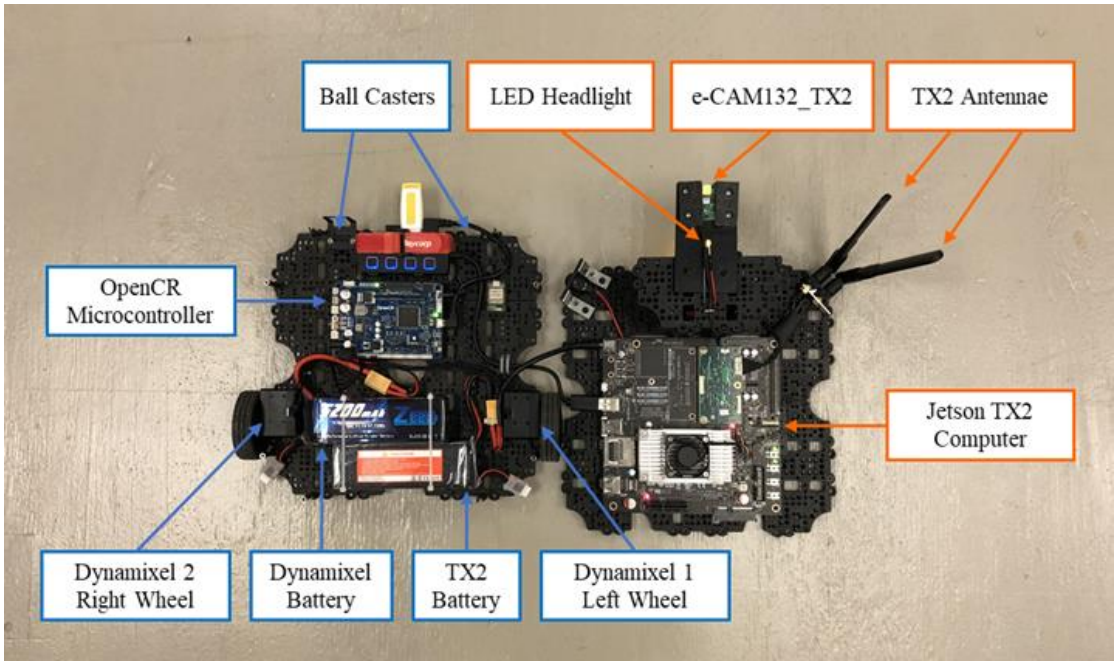


Figure 2.2 Two-tier ground robot assembly

The predominant structure of the robot is formed by TurtleBot3 waffle-plates, uniform plastic pieces which can connect in various orientations to assemble into different configurations, and to which components can be fastened at multiple points. The versatility of the waffle-plate is conducive to practical customization, and the two-tier modular assembly allows for easy maintenance and rearrangement. More importantly, adding or removing waffle-plate layers to the camera mount allows for adjustment of the camera's height and field of view.

2.2 PATH FOLLOWING MODEL

The path following model calculates the changes in the robot's two deviations from the path, henceforth referred to as line error Z and angle error θ. The system is dictated by the following kinematics equations (Ribeiro & Conceição 2019):

$$\frac{dZ}{dt} = \omega H + (\omega Z + v)\tan(\theta) \tag{2.1}$$

$$\frac{d\theta}{dt} = \omega - c\frac{(\omega Z + v)}{\cos(\theta)} \tag{2.2}$$

where v and ω are the robot's linear and angular velocities, respectively, c is the path curvature, and H is the horizon distance. The variables of the kinematics equations are illustrated in Figure 2.3.

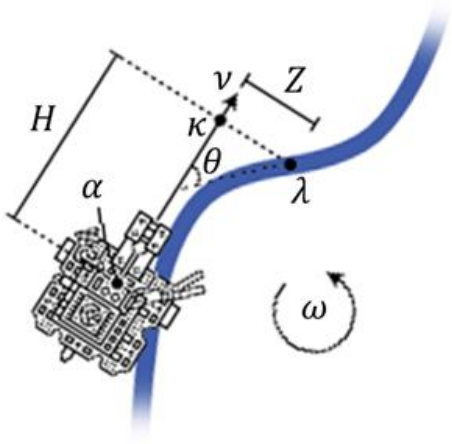


Figure 2.3 Path following model schematic

Consider α the robot's center of rotation, and κ to be the point on the ground in front of the robot which corresponds to the point at a distance of H from α along the spine of the robot. Consider λ the intersection of the path and a line perpendicular to the spine of the robot at κ. The line error Z represents the distance from κ to λ. The angle error θ

represents the angle formed by the intersection of the spine of the robot and a line tangent to the path at λ. The assumed sign conventions for Z, θ and ω are given in Figure 2.4.

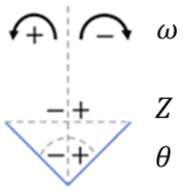


Figure 2.4 Path following sign conventions

A clockwise (left) turn is considered positive for ω. Z is considered positive when λ lies to the right of the robot's spine. θ is considered positive on the right side of the robot's spine (as in Figure 2.3).

2.3 CAMERA READINGS

The camera feed from the robot is processed in Python by the OpenCV computer vision library. For a given image, the robot's initial state and curvature can be calculated by analyzing a binary matrix representation of that image. The process of converting a single image into its representative binary path matrix is shown in Figure 2.5:

(a) *(b)* *(c)* *(d)*

Figure 2.5 Binary path matrix extraction process

The camera supplies a 640 pixel wide × 480 pixel tall full color image (Figure 2.5-a) of the prescribed path, which is constructed with blue painter's tape on a grey wooden surface. This image is converted into the hue, saturation, value (HSV) color space (Figure 2.5-b). Every pixel in the HSV color space consists of three values ranging from $0 - 255$,

each representing the intensity of that pixel's hue, saturation and value. A chroma key filter can be applied to convert the HSV image into a white track on a black background (Figure 2.5-c). To produce this image, all pixels within the ranges $[40,255]$ for hue, $[50,255]$ for saturation and $[10,255]$ for a value become white, while all pixels outside of these ranges become black. These ranges are found empirically by measuring the HSV values for path pixels (blue) and non-path pixels (non-blue) from test images taken from the robot's camera. A byproduct of this conversion is a 480 row $\times$ 640 column binary matrix (Figure 2.5-d) (not depicted to scale). In this matrix a 1 represents a white pixel (path) and a 0 represents a black pixel (no path).

To calculate Z, θ and c, three reference points extracted from the binary path matrix are analyzed. The reference points (x_1, y_1), (x_2, y_2) and (x_3, y_3) are always taken from the same three matrix rows: 432, 240, 48, respectively. The three reference points for a sample image are shown in Figure 2.6.

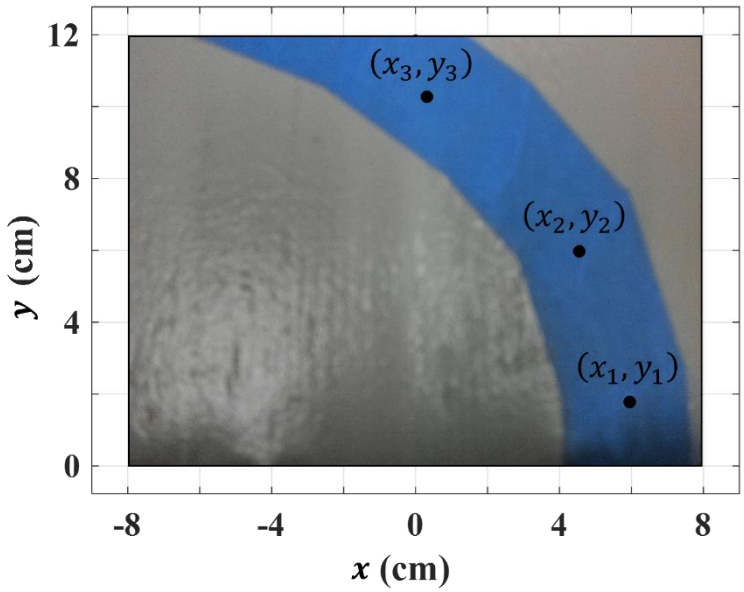


Figure 2.6 Binary path matrix analysis – reference points

The x-coordinate of a reference point is found by taking the mean column number of nonzero elements in its matrix row, and subtracting half of the total number of matrix columns:

$$x_P = \gamma \left[\frac{\sum_{i=1}^{n} I_i}{n} - 320 \right] \tag{2.3}$$

where i is the column number of nonzero element I, n is the total number of nonzero elements and γ is a pixel-to-cm conversion factor, found experimentally. 320 is the pixel number of the centerline along the *x*-direction.

The y-coordinate of a reference point is found by subtracting its matrix row number R_P from the total number of matrix rows (recall $R_1 = 432$, $R_2 = 240$, $R_3 = 48$):

$$y_P = \gamma[480 - R_P] \tag{2.4}$$

The line error Z is equal to the x-coordinate of the first reference point, as illustrated in Figure 2.7:

$$Z = x_1 \tag{2.5}$$

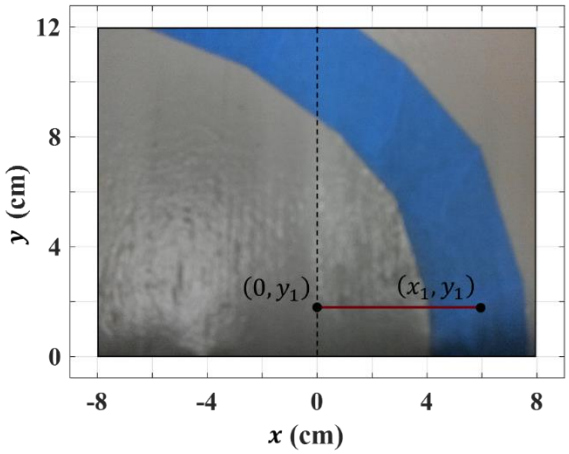


Figure 2.7 Binary path matrix analysis - Z

The angle error θ is equal to the opposite of the tangent of the slope between the first and third reference points, as illustrated in Figure 2.8:

$$\theta = -\tan\left\{\frac{y_3 - y_1}{x_3 - x_1}\right\} \tag{2.6}$$

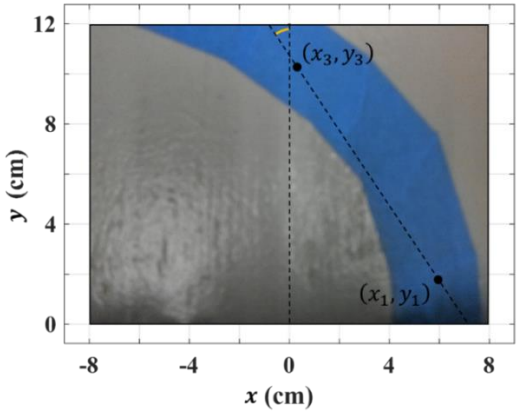


Figure 2.8 Binary path matrix analysis - θ

If the three reference points are collinear, then the curvature $c = 0$. Otherwise, c is calculated by considering the circle passing through the three reference points, as illustrated in Figure 2.9. The magnitude of c is equal to the inverse of the radius r of this circle, and the sign of c is determined by (x_C, y_C), the position of the center of said circle:

$$\text{If } x_C < x_2, \text{ then } c = \frac{1}{r}$$
$$\text{If } x_C > x_2, \text{ then } c = -\frac{1}{r} \tag{2.7}$$

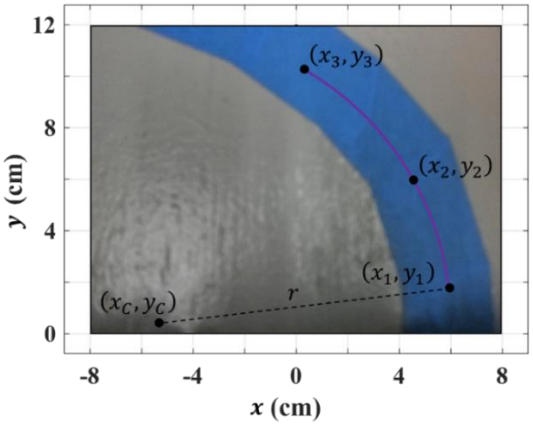


Figure 2.9 Binary path matrix analysis – c

The pixel-to-cm conversion factor γ is determined using the test image from the robot's camera feed shown in Figure 2.10:

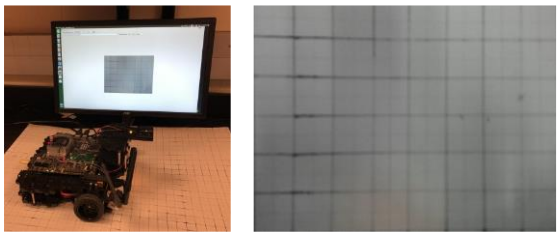


Figure 2.10 Camera feed test image

In the test image, the dark lines create 2 cm $\times$ 2 cm squares. At several locations in the image, the pixels spanning various horizontal and vertical 2 cm segments are counted, corresponding to a mean of 2 cm $= 80$ pixels for a pixel-to-cm conversion factor of $\gamma = 0.025$. The same test image is used to calculate the horizon distance H:

$$H = B + (480 - R_1)\gamma \tag{2.8}$$

where B is the distance in cm from the robot's center of rotation α to the point on the ground corresponding to the bottom of the test image (determined by measuring the distance from α to the point at the center of the bottom of the test image), and κ is $480 - R_1$ rows from the last row of the binary path matrix, as illustrated in Figure 2.11:

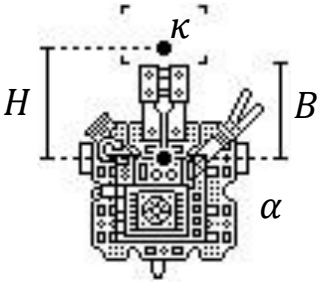


Figure 2.11 Horizon distance calculation

2.4 OVERHEAD CAMERA TRACKING

The robot can be tracked via overhead camera by attaching to it a fiducial marker. The type of fiducial marker used is an ArUco marker, a binary square marker, shown in Figure 2.12. The simplicity of the marker allows for fast, accurate detection, and an existing module in OpenCV supports ArUco tracking.

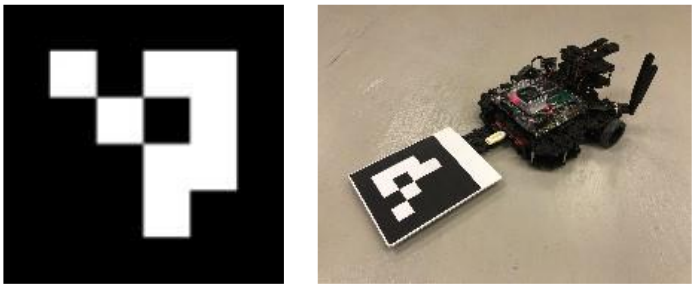


Figure 2.12 ArUco marker and trailing apparatus

The point on the robot targeted for tracking is its center of rotation α. The ArUco marker trailing apparatus keeps the distance between the center of the marker and α constant, so the recorded position of the marker can be used to find the position of the robot. The overhead camera records five reference points of the marker: its center F and its four corners $F^{(1)}$, $F^{(2)}$, $F^{(3)}$, $F^{(4)}$, illustrated in figure 2.13:

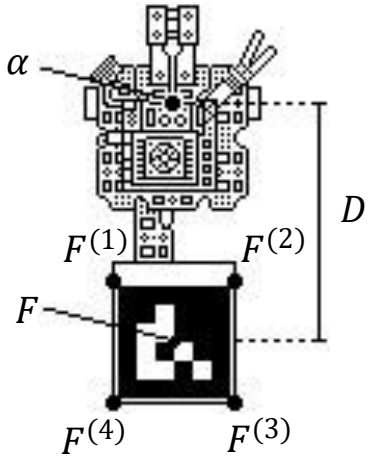


Figure 2.13 ArUco marker tracking reference points

The basic method of extracting the position of α from the marker reference points is to find the equation of a line between F and α, and to travel a distance of $D = 36$ cm (the constant distance between the center of the ArUco marker and the robot's center of rotation) along that line.

First, it is necessary to find the midpoint between the front two marker corners, which in theory aligns with the marker center and robot center. Equation (2.9) defines the midpoint M of $F^{(1)}$ and $F^{(2)}$:

$$\left(M_x, M_y\right) = \left(\frac{F_x^{(1)} + F_x^{(2)}}{2}, \frac{F_y^{(1)} + F_y^{(2)}}{2}\right) \tag{2.9}$$

The position of M relative to F dictates which method is used to find the position of α. If the robot and marker centers are aligned vertically, Equation (2.10) is used. Otherwise, Equation (2.11) is used).

$$\text{If } M_x = F_x, \text{ then } \left(\alpha_x, \alpha_y\right) = \left(F_x, F_y + D\frac{M_x - F_x}{|M_x - F_x|}\right) \tag{2.10}$$

$$\text{If } M_x \neq F_x, \text{ then } \left(\alpha_x, \alpha_y\right) = \left(F_x + \frac{D}{\sqrt{1+m^2}}\frac{M_x - F_x}{|M_x - F_x|}, F_y + \frac{Dm}{\sqrt{1+m^2}}\frac{M_x - F_x}{|M_x - F_x|}\right) \tag{2.11}$$

where m is the slope of the line between M and F:

$$m = \frac{M_y - F_y}{M_x - F_x} \tag{2.12}$$

Because tracked positions of the marker corners do not always form the expected perfect square, Equations (2.9)-(2.12) are applied in a similar fashion to the back two marker corners, and the robot positions found from the two corner pairs are averaged for a more accurate result.

The same method used to convert the robot's camera feed into a representative binary path matrix can be applied to the overhead camera feed to produce the binary path matrix shown in Figure 2.14:

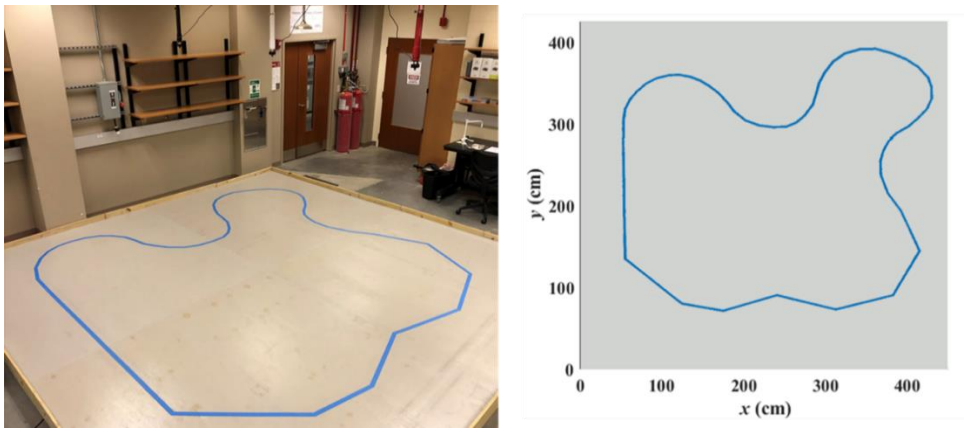


Figure 2.14 Overhead binary path matrix

The overhead binary path matrix can be treated as a set of points to which the position of α can be compared. The distance ε between α and the nearest point on the path provides a metric with which to measure path following accuracy.

CHAPTER 3

ARTIFICIAL NEURAL NETWORK

An artificial neural network (ANN) model can be trained to replace the physics-based model (PBM) in model predictive control (MPC). An ANN model learns the behavior of the system using physical data, and then predict the behavior where the data is not available. Both models have the same inputs: the initial line and angle errors $Z(n-1)$ and $\theta(n-1)$ respectively, and the current curvature, linear velocity, and angular velocity $c(n)$, $v(n)$ and $\omega(n)$, respectively. Both models also have the same output: predictions for the resultant line and angle errors $Z(n)$ and $\theta(n)$.

3.1 NETWORK STRUCTURE

Multiple ANNs are used for MPC, but each has the structure illustrated in Figure 3.1. Each ANN is a single-layer perceptron with 16 neurons, and a hyperbolic tangent activation function.

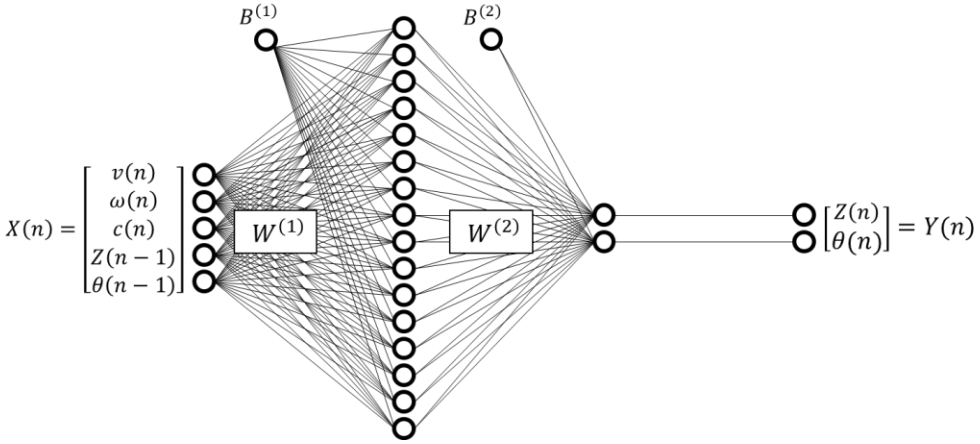


Figure 3.1 Neural network architecture

A single network is used to predict both $Z(n)$ and $\theta(n)$, as opposed to predicting each with a separate network, because the two values are related by the kinematics equations of the PBM. The output of the ANN can be represented mathematically by Equation 3.1:

$$Y(n) = W^{(2)} \tanh\{W^{(1)}X(n) + B^{(1)}\} + B^{(2)} \quad (3.1)$$

where, X is the input; Y is the output; $W^{(1)}$, $B^{(1)}$, $W^{(2)}$, $B^{(2)}$ are weights and biases.

3.2 TRAINING DATA

The ANN is trained using two different data sets: one generated by the PBM, and the other physically captured. The generated data set is created by supplying random inputs to the PBM without considering the realistic interference and is used to produce a baseline ANN that establishes the relationship between the input and the output. The captured data set is created by recording realistic operation of the robot and is used to either create a new model or compensate the difference between the baseline model and the realistic operation. The two data sets are used in different ways to train different ANNs – separately, as well as in tandem to measure the effect of introducing captured data.

The generated data set is created using the algorithm shown in Figure 3.2:

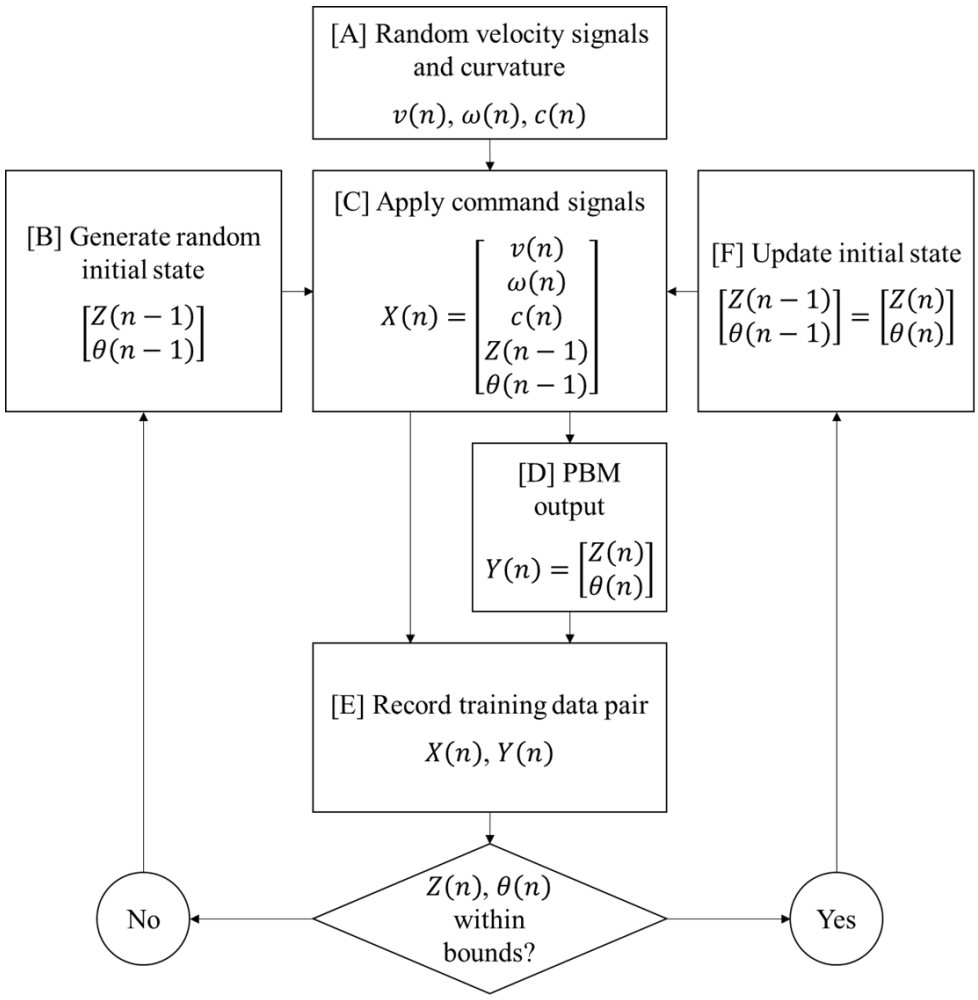


Figure 3.2 Generated training data algorithm

First, a random sequence of linear and angular velocity commands, as well as curvatures, is generated ([A] in Figure 3.2). Next, a random initial state (line and angle error) is generated ([B] in Figure 3.2), and supplied to the PBM ([C] in Figure 3.2), along with a curvature and velocity signals from the aforementioned sequence. The resultant state is calculated ([D] in Figure 3.2), and the initial and resultant state are recorded to become one training data point ([E] in Figure 3.2). The process is then repeated using one of two initial states: if the resultant state is within prescribed bounds, it becomes the initial state for the

next iteration ([F] in Figure 3.2); if not, a new random initial state is generated for the next iteration ([B] in Figure 3.2).

The bounds for each variable in the generated data set are given in Table 3.1. For v, ω, and c, a random value is generated for a random duration. The random value is set between the flag bounds given for each variable. The random duration is set between 0.1 and 0.5 seconds. The random shift in value when each variable changes is restricted as well, to no more than 40% of the total range of the flag bounds. Z and θ are considered out of bounds if either variable is outside of its flag bounds. Whenever a new initial state is generated, Z and θ are set inside the central 25% of their respective flag bounds.

Table 3.1 Generated training data bounds

	v	ω	c	Z	θ
Flag Bound	[0,20.0]cm/s	$\pm$1.4 rad/s	$\pm$1.0 cdpt	$\pm$8.0 cm	$\pm\frac{\pi}{2}$ rad
Initial Bound	[0,20.0]cm/s	$\pm$1.4 rad/s	$\pm$1.0 cdpt	$\pm$2.0 cm	$\pm\frac{\pi}{8}$ rad
Duration	[0.1,0.5] s	[0.1,0.5] s	[0.1,0.5] s	-	-
Max Shift	$\pm$ 8.0 cm/s	$\pm$0.56 rad/s	$\pm$0.1 cdpt	-	-

A total of one hour of data is generated, a five minute sample of which is shown in Figure 3.3:

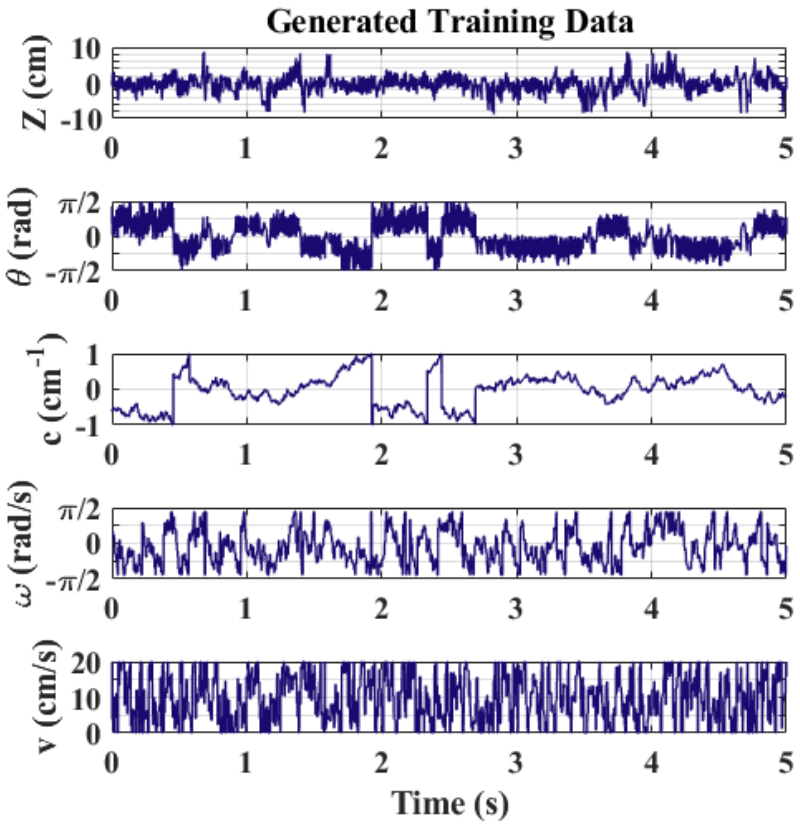


Figure 3.3 Sample of generated data set

The captured data set is created by operating the robot while recording its camera readings Z, θ and c and its velocity commands v and ω. The robot is operated in one of two ways: on a circuit track with path following control or on a straightaway supplying random velocity commands.

The controlled captured data set is created using the algorithm shown in Figure 3.4:

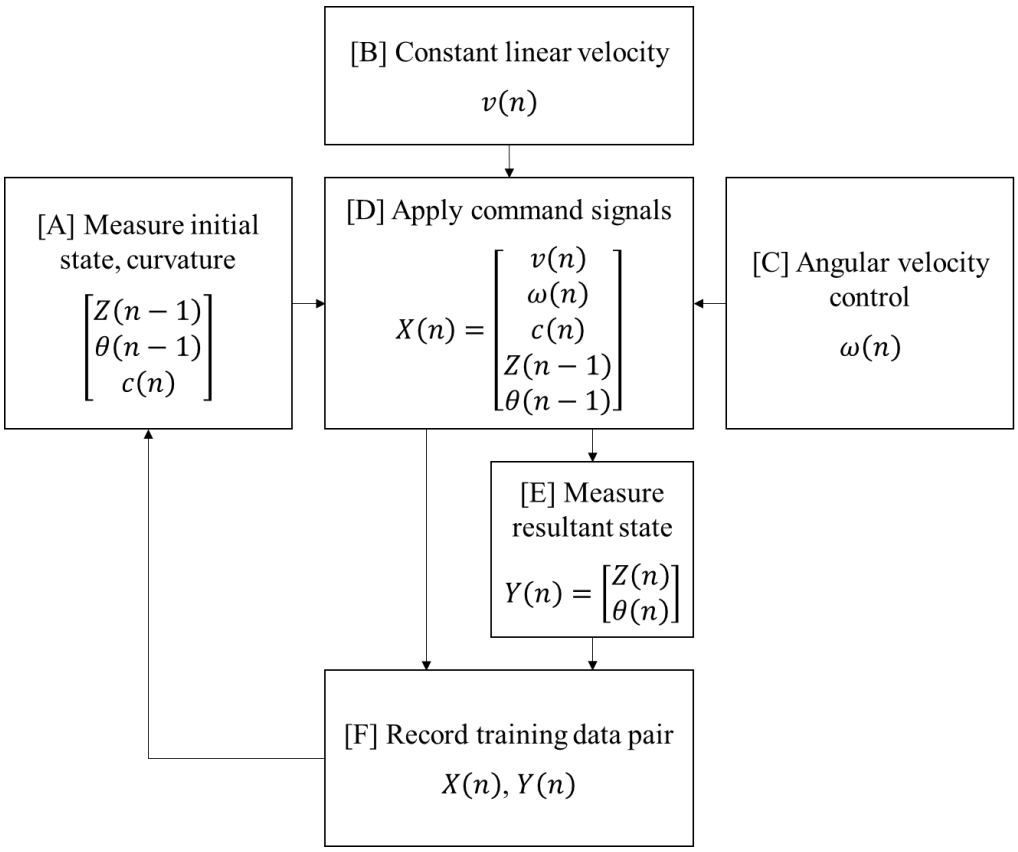


Figure 3.4 Controlled captured training data algorithm

First, the initial state and curvature are measured ([A] in Figure 3.4). The robot's linear velocity is constant ([B] in Figure 3.4), and an angular velocity which will cause the robot to follow the path is selected ([C] in Figure 3.4) using MPC. The linear and angular velocity command signals are applied to the real robot ([D] in Figure 3.4), the resultant state is measured ([E] in Figure 3.4) and becomes the output of a training data point ([F] in Figure 3.4). The robot's linear velocity is varied for different laps around the circuit track. A total of thirty minutes of controlled data is captured, a five minute sample of which is shown in Figure 3.5:

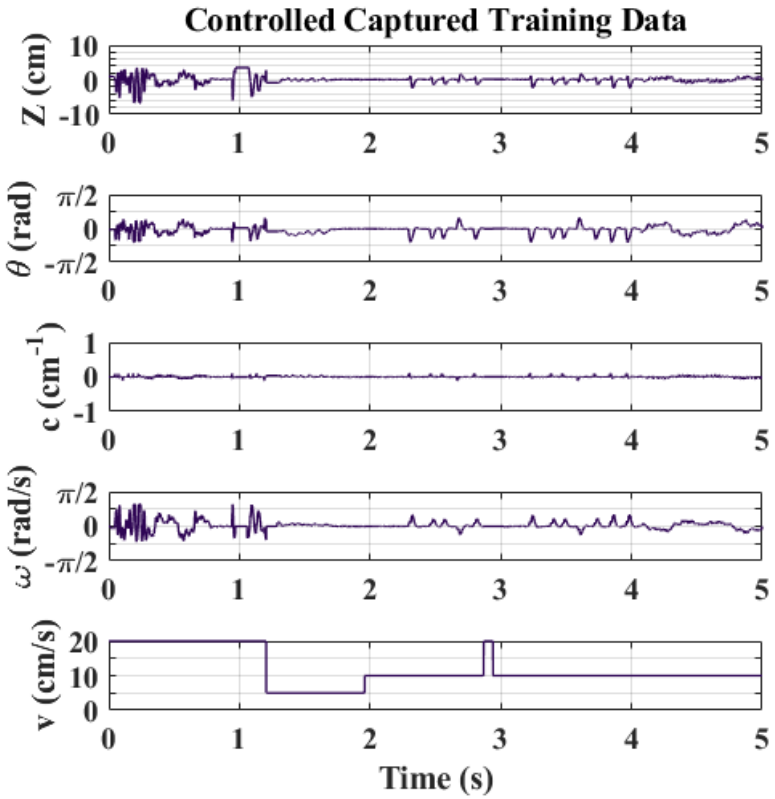


Figure 3.5 Sample of MPC captured data set

As seen in Figures 3.3 & 3.5, the recorded reading (Z, θ and c values) are of smaller magnitudes than those randomly generated.

3.3 PRETRAINING AND FINETUNING

Three ANNs are trained: one trained only with generated data (ANN_{gen}), one trained only with captured data (ANN_{cap}), the other initially trained on generated data then retrained with captured data (ANN_{combo}). The ANNs are trained in MATLAB using the *feedforwardnet* function. The training method is Levenberg-Marquardt. The validation metric is mean squared error (MSE). Training ends if validation does not improve after 6 consecutive epochs, or if the maximum iteration count of 1000 is reached.

The training performance for ANN_{gen} is shown in Figure 3.6. The best validation was reached after 270 epochs.

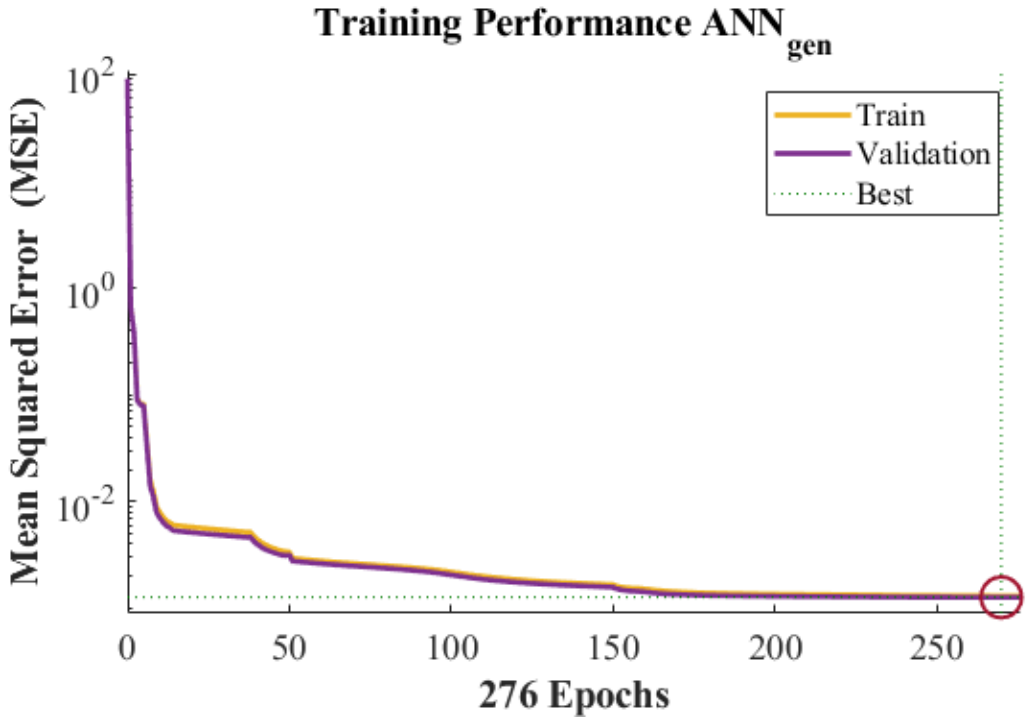


Figure 3.6 Training performance for ANN_{gen}

The offline performance on the test set for ANN_{gen} is shown in Figure 3.7. The ANN is able to predict the test set with an accuracy of $Z_{RMS} = 0.036$ cm and $\theta_{RMS} = 0.0025$, where subscript RMS denotes the root mean squared errors for Z and θ.

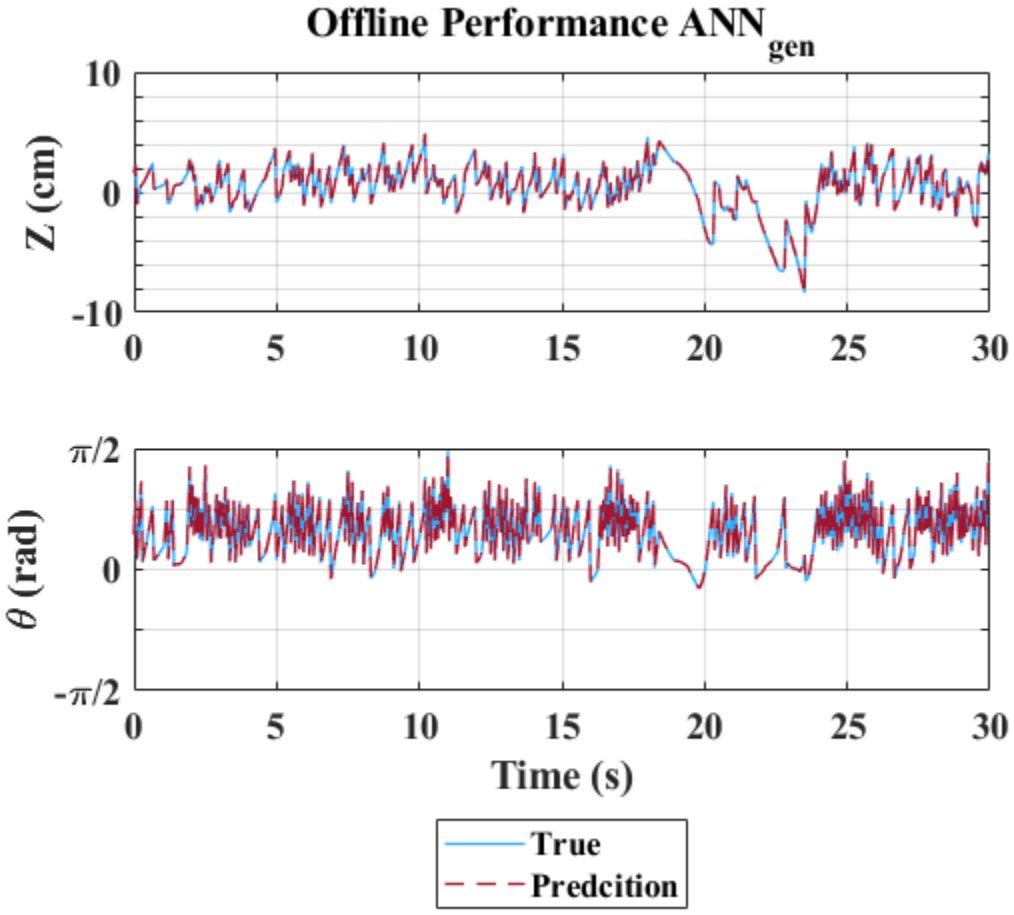


Figure 3.7 Offline performance for ANN_{gen}

CHAPTER 4

MODEL PREDICTIVE CONTROL

The ground robot is controlled using a method called model predictive control (MPC), the model being either the physics-based model (PBM) or an artificial neural network model (ANN). The model is used to predict the state of the robot given different control signals. The control signals which yield the desired state are supplied to the robot.

4.1 OVERVIEW OF MODEL PREDICTIVE CONTROL

The overall MPC framework for path following is shown in Figure 4.1:

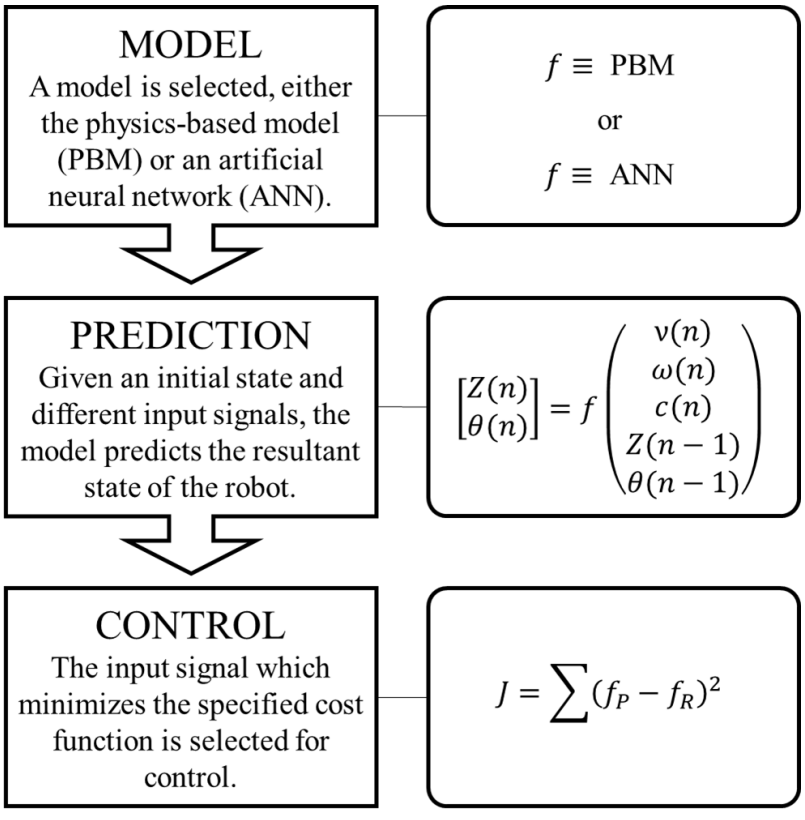


Figure 4.1 Model predictive control framework

The same control framework (shown in Figure 4.1) is used for both the PBM and ANN. The model is simply a function to which inputs are supplied and outputs are given, and both models take the same inputs: $v(n)$, $\omega(n)$, $c(n)$, $Z(n-1)$, $\theta(n-1)$. The outputs of both models are also the same: $Z(n)$, $\theta(n)$.

MPC is implemented on the ground robot as shown in Figure 4.2:

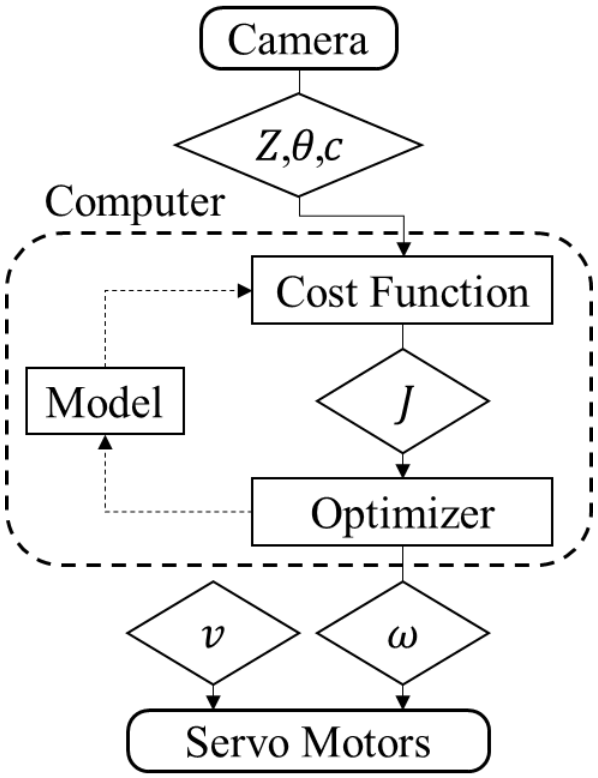


Figure 4.2 Model predictive control implementation on ground robot

$\omega(n)$ is the variable used for control, and is thus the target of optimization. The other four variables remain unchanged as different values of $\omega(n)$ are supplied to the model (the robot travels at a constant linear velocity ($v(n)$ is the same value for all n), and each time step has unique values for $c(n)$, $Z(n-1)$ and $\theta(n-1)$, measured using computer vision analysis). The different values of $\omega(n)$ yield different predictions for $Z(n)$ and $\theta(n)$.

4.2 COST FUNCTION

The path following cost function is dictated by the following equation:

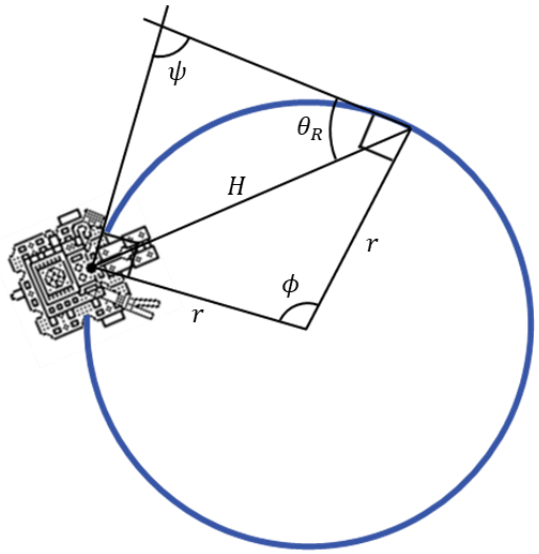
$$J = \underbrace{\sum_{j=1}^{N2} Z(j)^2 + \left(Q(\theta(j) - \theta_R)\right)^2}_{\text{error term}} + \underbrace{\rho \sum_{i=1}^{Nu} \left(\omega(i) - \omega(i-1)\right)^2}_{\text{stabilizer term}} \tag{4.1}$$

where $Q = \frac{16\text{ cm}}{\pi\text{ rad}}$ normalizes $\theta(j)$ and θ_R to the same range as $Z(j)$, θ_R is the target reference angle, ρ is the stabilizer weight, tuned experimentally (the stabilizer cost will be explained in further detail), Nu is the control horizon and $N2$ is the costing horizon. The control horizon is the number of time steps for which the control variable ω is selected, and the costing horizon is the number of time steps for which the resultant state is predicted. Note that $N2 \geq Nu$, and in Equation (4.1), for all $j \geq Nu$, $\omega(j) = \omega(Nu)$.

The first half of the path following cost function is the error term. Minimizing this term minimizes the difference between the predicted and reference states. The reference state for the line error is zero; the reference state for the angle error is given by:

$$\theta_R = -\sin^{-1}\left(\frac{Hc}{2}\right) \tag{4.2}$$

the derivation of which is given in Figure 4.3:



$$\cos(\phi) = \frac{H^2 - (r^2 + r^2)}{-2r^2}$$

$$\cos(2\theta_R) = \frac{H^2 - (r^2 + r^2)}{-2r^2}$$

$$1 - 2\sin^2(\theta_R) = \frac{H^2 - (r^2 + r^2)}{-2r^2}$$

$$\sin^2(\theta_R) = \frac{H^2}{4r^2}$$

$$\sin^2(\theta_R) = \frac{H^2c^2}{4}$$

$$\theta_R = \pm \sin^{-1}\left(\frac{Hc}{2}\right)$$

Figure 4.3 Derivation of reference angle error

The derivation assumes a path of constant curvature (i.e. a circle). Note that the sign of the θ_R is the opposite of the sign of c, and that when $c = 0$ (i.e. a line), $\theta_R = 0$. The nonzero reference angle is necessary because for a path with nonzero curvature, Z and θ cannot simultaneously be zero (unless $H = 0$).

The second half of the path following cost function is the stabilizer term. Minimizing this term minimizes the change in ω from one time step to the next. This prevents the robot from changing direction too drastically too quickly at a given time step. The value of the stabilizer weight ρ is selected based on the value which yields the best path following accuracy.

4.3 OPTIMIZATION

The signal frequency needs to be greater than the control frequency. The robot's camera has limited speed; the maximum frequency of signal calculation (image intake, analysis and calculation of Z, θ and c) is 40 Hz. Thus the selected control frequency is 20

Hz. To reach this frequency, an angular velocity command must be generated every 0.05 s.

The minimization of the cost function is performed by the Python optimization package *scipy.optimize.minimize*. The arguments for this function are the definition of the function to be minimized, an initial estimate and bounds for the control variable, the optimization method, the learning rate, the tolerance, and a cap on the number of iterations to be performed if the tolerance is not reached.

The function to be minimized is Equation (4.1). The initial estimate for $\omega(1)$ is $[0 \quad 0 \quad 0]$, and the initial estimate for every subsequent time step $\omega(n)$ is $[\omega(n-1) \quad \omega(n-1) \quad \omega(n-1)]$; it is assumed that the optimal control for the ground robot is most likely a minor adjustment from its current trajectory. The bounds for the control variable are $[-1.0 \quad 1.0]$ rad/s. The optimization method is Sequential Least Squares Programming (SLSQP). The learning rate is 10^{-3}, the tolerance is 10^{-11} and the maximum number of iterations is 50.

4.4 CONTROL PARAMETER TUNING

Increasing the horizons Nu and $N2$ in general, increases the accuracy of path following, however the frequency with which command signals must be generated limits the usable values since increasing Nu and $N2$ increases the time necessary to minimize the cost function and thus generate a command signal. The optimization times for trial runs using $Nu = 3$ and various values of $N2$ are shown in Figure 4.4:

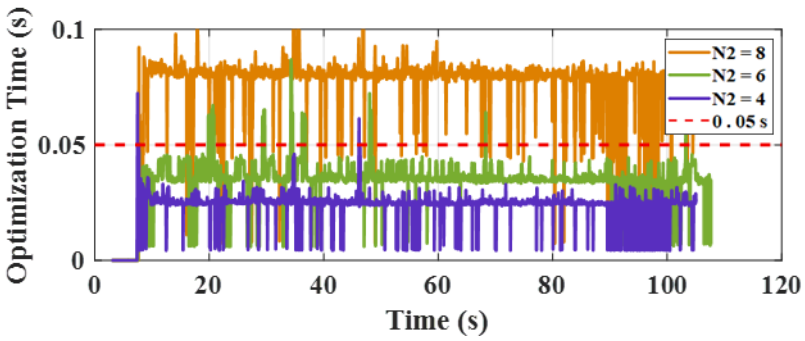


Figure 4.4 Optimization times for various costing horizons

As seen in Figure 4.4, the costing horizon $N2 = 6$ is selected because it is the highest value which completes optimization in the allotted time interval. The horizons $Nu = 3$, $N2 = 6$ provide sufficiently quick and accurate optimization.

A similar search is conducted to determine the optimal value of the stabilizer weight ρ. Trials of MPC are performed with select values of ρ at various linear velocities v and the overhead camera error ε is recorded, the results of which are shown in Figure 4.5:

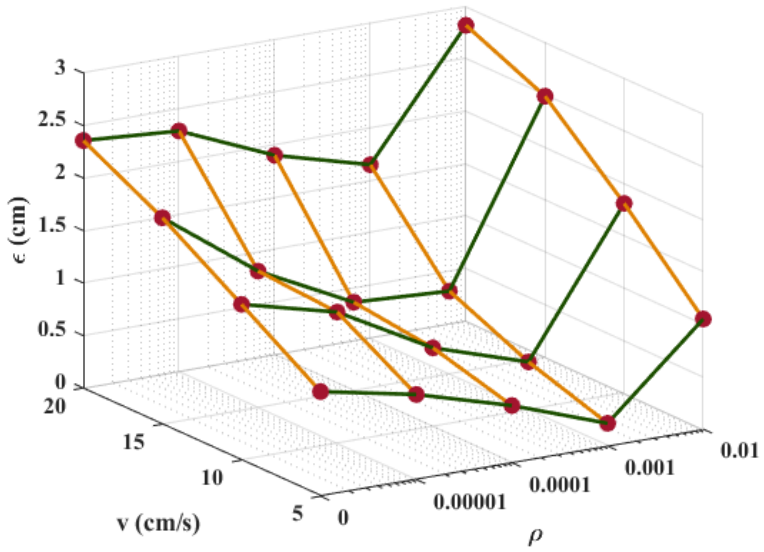


Figure 4.5 Overhead camera error for various stabilizer weights, linear velocities

As seen in Figure 4.5, the stabilizer weight $\rho = 0.001$ is selected because it consistently yields the lowest overhead camera error across the range of linear velocities.

CHAPTER 5

RESULTS & DISCUSSION

The accuracy of path following using model predictive control (MPC) is compared for different models: the physics-based model (PBM), an artificial neural network trained only on PBM-generated data (ANN_{gen}), an artificial neural network trained only on captured data (ANN_{cap}), and an artificial neural network initially trained on generated data then retrained with captured data (ANN_{combo}).

5.1 PERFORMANCE METRICS

The test track, shown in Figure 5.1, is designed to examine the robot's ability to follow a variety of paths. It is composed of a sharp turn section (A to B), a smooth curve section (B to C) and a straightaway section (C to D). Figure 5.1 is generated by cropping and stitching together feeds from three separate cameras, and is used only for display purposes (Kyzer 2021). The robot's deviation from the path is tracked using only one camera at a time. When the robot enters a camera's field of view, two sets of points are generated: the x and y-coordinates of the robot's center of rotation α, and the x and y-coordinates of the path. The distance ε between every ArUco point and the path point nearest to that point is calculated using the Python module *dsearchn*.

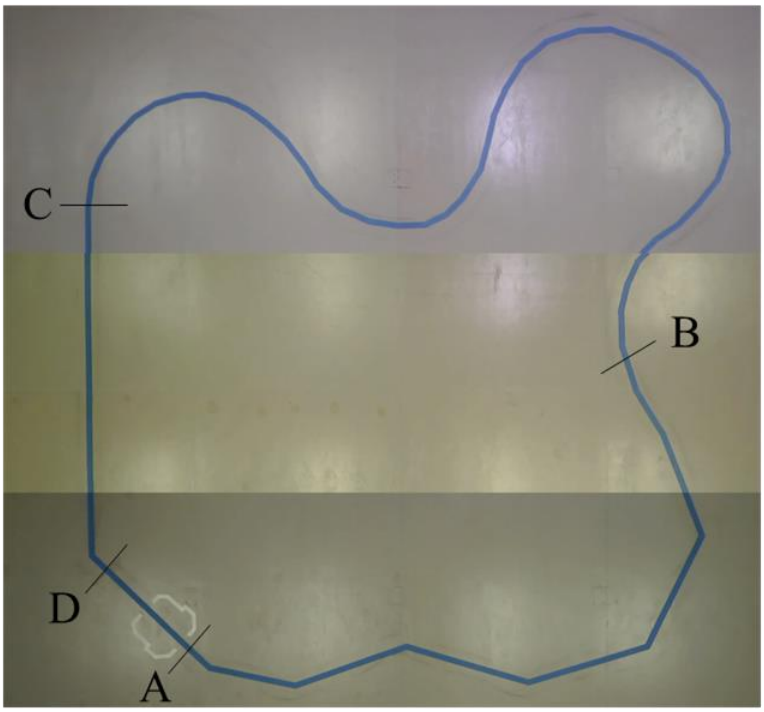


Figure 5.1 Test track

Three metrics are used to compare path following performance for different models: the root mean squared error of the line error measured by the onboard camera (Z_{RMS}), the root mean squared error of the angle error measured by the onboard camera (θ_{RMS}), and the root mean squared distance between the robot and the track measured by the overhead camera (ε_{RMS}).

5.2 RESULTS

For each of the four different models, three trials of MPC are conducted, beginning at point A and traveling anticlockwise along the path to point D. For each trial the linear velocity v is set to a constant 15 cm/s; path following is achieved by optimizing the angular velocity ω. The results of these trials are shown in Table 5.1:

Table 5.1 MPC trials results

Model	Trial	Z_{RMS}		θ_{RMS}		ε_{RMS}	
PBM	1	1.0057	Mean: 0.9953	0.2099	Mean: 0.2096	1.2862	Mean: 1.1571
	2	0.9822		0.2081		1.0180	
	3	0.9980		0.2109		1.1672	
ANN_{gen}	1	0.9894	Mean: 1.0200	0.2077	Mean: 0.2120	1.6139	Mean: 1.8185
	2	0.9947		0.2077		1.8646	
	3	1.0760		0.2076		1.9771	
ANN_{cap}*	1	3.1754	Mean: 3.0084	0.2044	Mean: 0.2231	3.8925	Mean: 3.7800
	2	2.7054		0.2343		3.0066	
	3	3.1443		0.2305		4.4408	
ANN_{combo}	1	0.7794	Mean: 0.7743	0.2075	Mean: 0.2069	0.8219	Mean: 0.8102
	2	0.7740		0.2065		0.8045	
	3	07694		0.2066		0.8043	

**The robot is unable to complete the test track in three trials of MPC using the ANN_{cap} model.*

The robot is able to complete the test track circuit in every trial using MPC with all models except for ANN_{cap}. The PBM gives a baseline mean performance of 1.1571 cm for ε_{RMS}. The ANN_{gen} yields an accuracy of an ε_{RMS} 6.6 mm greater than that of the PBM. Retraining the ANN_{gen} with captured data improves the performance to 3. 5 mm better than the PBM. Z_{RMS} was improved by 2.2 mm and θ_{RMS} was improved by 0.002 rad from the PBM to ANN_{combo}.

The ROSbag files and overhead camera tracking from the best trial for each model are shown in Figures 5.2-5.5. The best trial is determined by the minimum ε_{RMS}, except for ANN_{cap}, where the trial considered the best is the one in which the robot makes the farthest progression along the path before losing it completely.

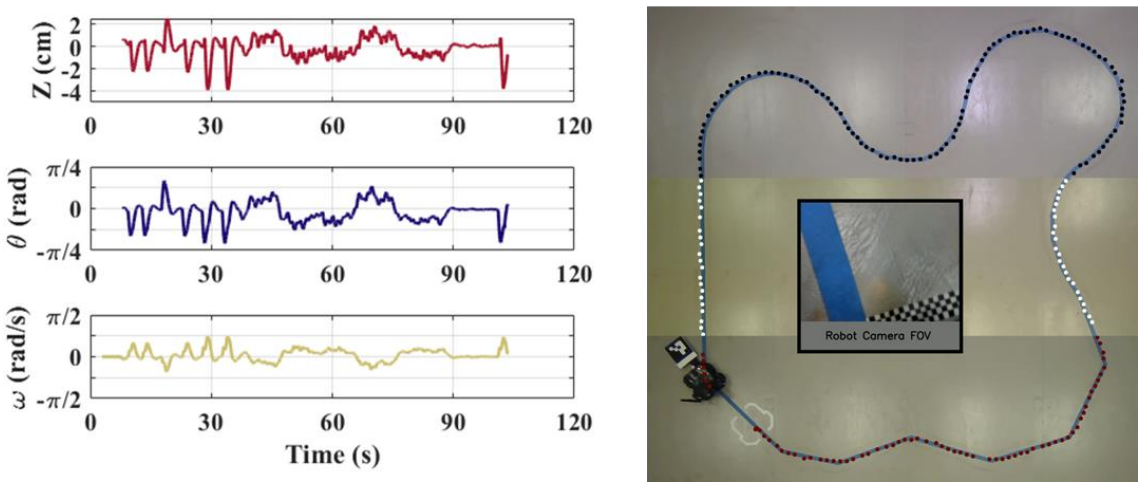


Figure 5.2 MPC performance with PBM

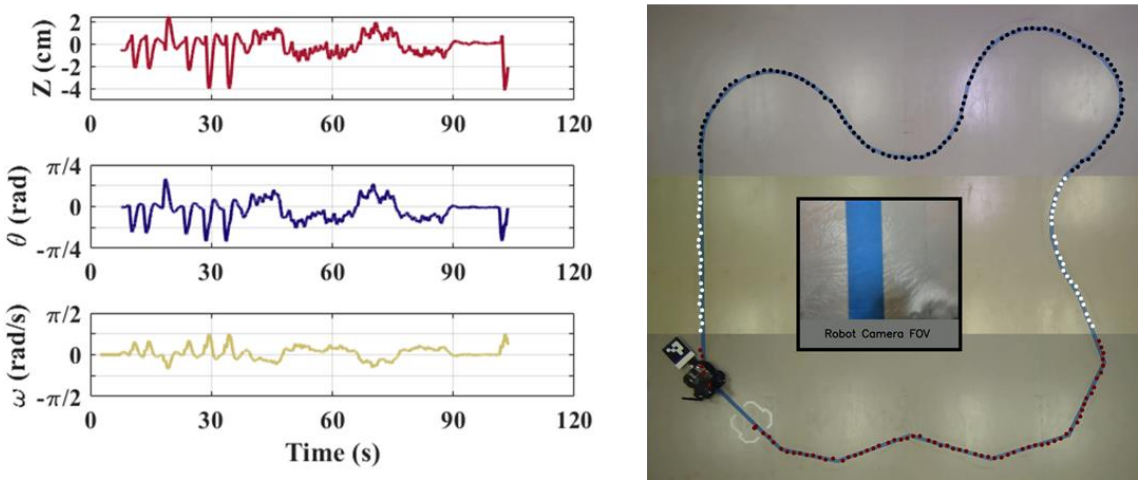


Figure 5.3 MPC performance with ANN_{gen}

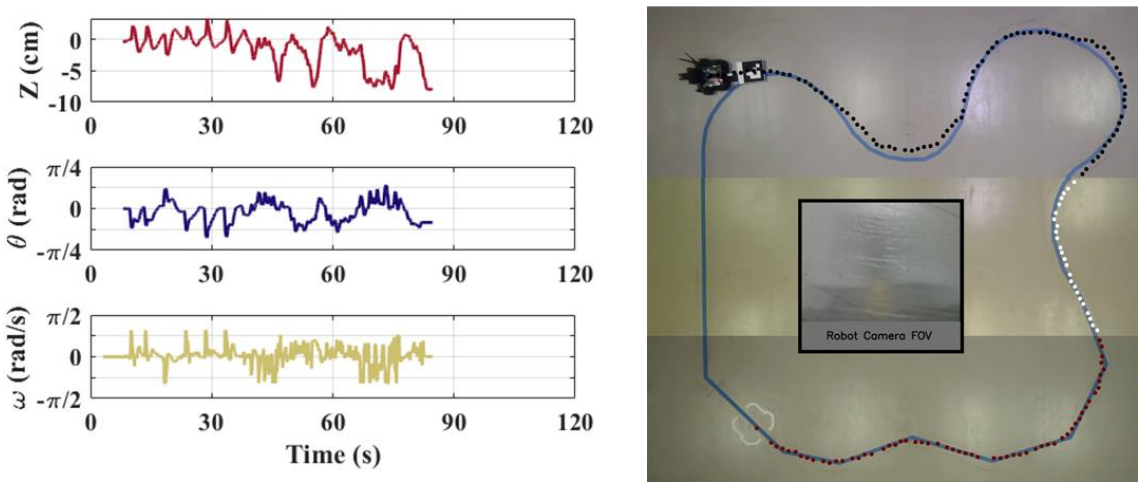


Figure 5.4 MPC performance with ANN_{cap}

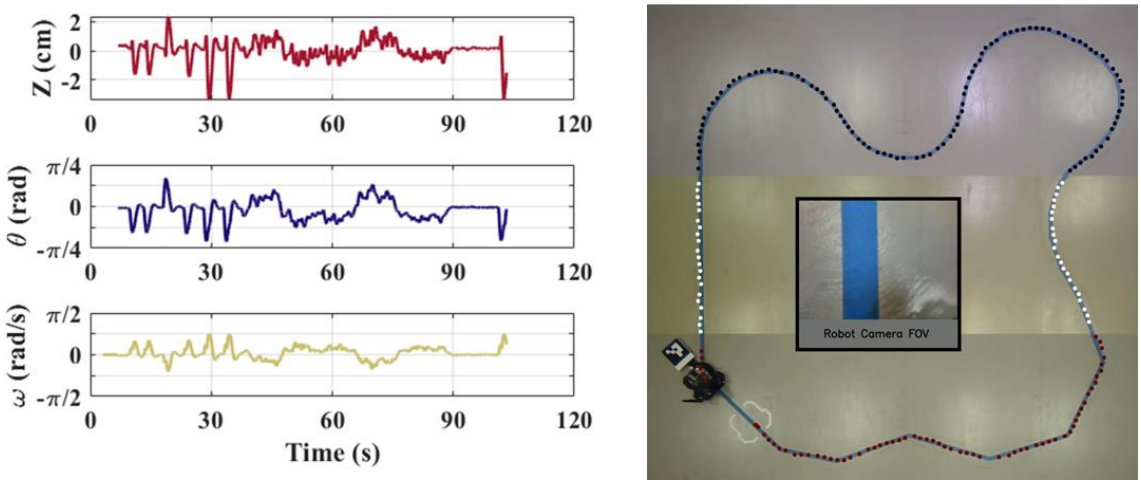


Figure 5.5 MPC performance with ANN_{combo}

The PBM, ANN_{gen} and ANN_{combo} consistently give similar values for ω for each section of the test track, which yielded similar values for Z and θ . The largest magnitudes of ω, as well as Z and θ, occur in the sharp turn section.

CHAPTER 6

CONCLUSION

Path following with the ground robot is achieved using its onboard camera. Computer vision tools allow the blue path on a grey background to be converted to a binary matrix. Treating the row and column indices of nonzero elements of that matrix allows for analysis of the track in a Cartesian coordinate system. A known physics-based model (PBM) dictates changes in the system given information taken from the camera in this way. Model predictive control (MPC) determines which command signals will keep the robot following a path by selecting those which minimize a cost function. The associated cost is higher when the robot is farther away from the path, as well as when changes in command signals between time steps is greater.

The path following PBM can be replicated with reasonable accuracy by training an artificial neural network (ANN) with data randomly generated using that PBM. Path following performance can be further improved by introducing captured data, though captured data alone is insufficient for path following. The ANN is a viable replacement for the PBM and offers potential to accommodate varying dynamics under realistic circumstance.

Overhead camera tracking shows that using an ANN initially trained on generated data and retrained on captured data for MPC can follow a prescribed path with an average

distance 3.5 mm closer to the path than using the PBM (based on the overhead camera measurement).

Future work will focus on the use of an ANN to detect and mitigate anomalies in the robotic platform, such as a misaligned wheel(s) or camera. The robot will collect data from its onboard camera in real time, and unexpected values will indicate the presence of an anomaly. The online data will be used to retrain an ANN to enable accurate path following in spite of the presence of anomalies.

REFERENCES

Kyzer, T. Instrumentation and Experimentation Development for Robotic Systems. *University of South Carolina, Proquest Dissertations Publishing*, 2021. https:www.proquest.com/docview/2572554310

Ribeiro, T., & Conceição, A. (2019). Nonlinear Model Predictive Visual Path Following Control to Autonomous Mobile Robots. *Journal of Intelligent & Robotic Systems, 95*, 731-743. https://doi.org/10.1007/s10846-018-0896-3

Surya prakash, M., Ajay Vignesh, K., Shyamsunthar, J., Raman, K., Senthil Raju, J., & Raju, N. Computer Vision Assisted Line Following Robot. *Procedia Engineering, 38*, 1764-1772. https://doi.org/10.1016/j.proeng.2012.06.215