Robotica (2024), **42**, pp. 2825–2847
doi:10.1017/S0263574724001176

RESEARCH ARTICLE

Artificial neural network-based model predictive visual servoing for mobile robots

Seong Hyeon Hong[1], Benjamin Albia[2], Tristan Kyzer[2], Jackson Cornelius[3], Eric R. Mark[4], Asha J. Hall[4] and Yi Wang[2]

[1]Florida Institute of Technology, Melbourne, FL, USA
[2]University of South Carolina, Columbia, SC, USA
[3]CFD Research Corporation, Huntsville, AL, USA
[4]Army Research Laboratory, Aberdeen Proving Ground, MD, USA
Corresponding author: Yi Wang; Email: yiwang@cec.sc.edu

Received: 7 December 2023; **Revised:** 16 April 2024; **Accepted:** 6 June 2024; **First published online:** 18 September 2024

Keywords: vision-based control; model predictive control; intelligent control; autonomous robot; path-following

Abstract
This paper presents an artificial neural network (ANN)-based nonlinear model predictive visual servoing method for mobile robots. The ANN model is developed for state predictions to mitigate the unknown dynamics and parameter uncertainty issues of the physics-based (PB) model. To enhance both the model generalization and accuracy for control, a two-stage ANN training process is proposed. In a pretraining stage, highly diversified data accommodating broad operating ranges is generated by a PB kinematics model and used to train an ANN model first. In the second stage, the test data collected from the actual system, which is limited in both the diversity and the volume, are employed to further finetune the ANN weights. Path-following experiments are conducted to compare the effects of various ANN models on nonlinear model predictive control and visual servoing performance. The results confirm that the pretraining stage is necessary for improving model generalization. Without pretraining (i.e., model trained only with the test data), the robot fails to follow the entire track. Weight finetuning with the captured data further improves the tracking accuracy by 0.07–0.15 cm on average.

1. Introduction

With the advent of autonomous mobile robots, there is a significant amount of research in developing control strategies based on computer vision. Visual servoing, also known as vision-based control, utilizes information acquired from imagery data for system control. The continuously growing power of the graphics processing unit (GPU) computing and its application in advanced control and autonomy have vastly boosted the development of visual servoing technology in the past decades. For example, the present robotics applications, such as aerial vehicles [1–3], underwater vehicles [4], robot manipulator [5, 6], humanoid robots [7], and military robots [8] all heavily rely on vision-based control. Among diverse applications in visual servoing, motion control of the mobile robot (MR) has been a popular one and covers a variety of topics, including simultaneous localization and mapping, path planning/following, and self-driving [9, 10].

Recently, model predictive control (MPC) has become one of the popular controllers for visual servoing applications [11–13]. Compared to other control techniques, MPC is intuitive and easy to implement. Perhaps, the foremost advantage is its ability to handle input and output constraints, which is often required in robotics applications. Although vision sensor was not included, Pacheco and Luo proved that MPC outperforms proportional-integral-derivative (PID) control in path-following of MRs and incurs lower tracking errors [14]. Similar results were observed in unmanned aerial vehicles [15], which confirmed the excellent performance of MPC over PID in certain robotics applications. Despite its salient control performance, the MPC implementation is hindered by the substantial computing load for optimization. For linear models, quadratic programing (QP) can be applied to mitigate the burden, while linearization of the states and control inputs incurs approximation, potentially compromising the model prediction accuracy. An effective means to address such a limitation is to use nonlinear models directly in MPC, viz., nonlinear model predictive control (NMPC), which, however, requires sequential quadratic programing (SQP) to find the optimal solution.

Fortunately, with advances in edge computing platforms and numerical methods, NMPC has been successfully demonstrated in MRs. Li et al. reported visual servo-based MPC to control the steering of nonholonomic mobile robot (NMR) with both kinematic and dynamic constraints [16]. Moreover, to lower the computation burden for solving NMPC, the primal-dual neural network was utilized to operate the robot in real-time even with low computing power. Ke et al. further extended this idea and developed a tube-based MPC for MR. It is based on a double-layer control scheme composed of an inner gain-scheduled feedback layer within an outer open-loop NMPC layer to compensate for an environmental disturbance [17]. Implementation of NMPC and primal-dual neural network remains consistent with their prior work [16]. Ribeiro and Conceicao demonstrated NMPC-based visual path-following of the NMR [18]. Their nonlinear model for visual path-following is acquired from a Frenet–Serret (FS) coordinate system to simplify the camera model [19]. The SQP method was used to compute the control signal by a personal computer installed on the robot. The system demonstrated the path-following accuracy of less than 5 cm average distance error at the speed of 10 cm/s.

With convincing evidence on NMPC feasibility for MRs, this paper proposes an ANN-based NMPC method for visual path-following, and the ANN model is trained by both model-generated data and the actual test data. It has been proved in our prior work [20] that the performance of MPC is proportional to the model prediction accuracy. Therefore, it is essential to use an accurate model for horizon predictions. Commonly, physics-based (PB) models are derived from the known kinematics or dynamics of the system. However, insufficient knowledge and parameter uncertainty are inevitable in all PB models and could contribute to modeling errors. For this reason, data-driven models, especially ANNs have been adopted in diverse robotic applications. It is well known that the ANN model can precisely represent a complex system only if the training dataset is sufficiently diverse and covers the entire range of operational parameter space. Unfortunately, this is often difficult to obtain in practice, and therefore, the ANN model is only valid for a limited range due to its poor extrapolation ability. In addition, training must be performed with caution since it is vulnerable to overfitting and often stuck at poor local optima, leading to inconsistent prediction. To address the challenges associated with ANN, in the present effort, the ANN training process is divided into two stages: pretraining and finetuning. In the former, the PB model is utilized to generate a massive amount of data with sufficient diversity to train ANN, which allows the model to represent the system dynamics more robustly in the full range of parameter space. In the finetuning stage, data collected from the actual system operation are then utilized to finetune the trainable weights of the ANN. Since ANN is already pretrained, experimental data for finetuning do not need to be strictly diverse or large in size. This can be considered as a combination of transfer learning and multifidelity modeling techniques. That is, the low-fidelity data generated using the PB model are first utilized to train a model with great generalization. Then high-fidelity data that are captured by the sensors from the MR and have limited availability, are used to update the weight parameters of the model.

The contributions of the paper are summarized as follows: (1) nonlinear model predictive visual path-following is demonstrated with NMR based on quantitative measurements solely extracted from the onboard camera images; (2) an ANN model is used in lieu of the PB model for predictive visual servoing. It accurately represents the system and tackles the issue of poor generalization of the data-driven model by including a pretraining phase that utilizes the PB model to generate low-fidelity data of salient diversity. Thus, the high-fidelity test data directly measured from the system are intended to finetune the ANN parameters to further improve the model accuracy; and (3) a thorough study on path-following is carried out to demonstrate the excellent performance of the proposed method by comparing

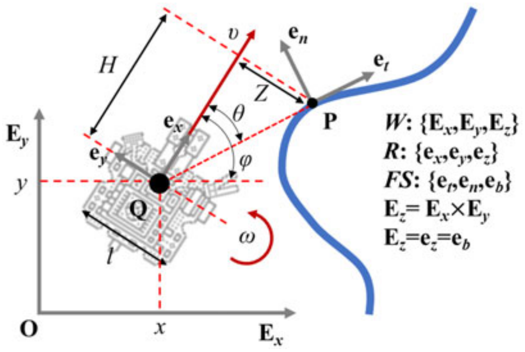


Figure 1. *Schematic of visual path-following for mobile robot.*

it with the MPC based on ANN models trained only with either generated (low-fidelity) or actual (high-fidelity) data.

The remainder of the paper is organized as follows. In Section 2, the PB model for visual path-following of a MR is derived. Model predictive vision-based control is elucidated in Section 3, including stability criterion, control parameters, reference signals, and extracting information from the image. In Section 4, the ANN model representing the MR and the training procedure are described. The details of experimentation regarding the robot platform and the overhead camera tracking system for performance evaluation are given in Section 5. The results of path-following experiments are presented in Section 6, and Section 7 concludes the paper.

2. Visual path-following model

The present research focuses on path-following solely based on visual information of relative position without knowing absolute positions of the MR and the track. The coordinate systems for modeling the visual path-following problem are depicted in Figure 1 [18, 21]. From the figure, E_x, E_y, and E_z are the basis vectors of the world coordinate system (W), and e_x, e_y, and e_z are the basis vectors of the robot coordinate system (R). A point on a track, P, detected from an image at a fixed horizon distance (H) with respect to the MR's center of rotation (Q), is considered as a particle moving along the track. The motion of P is expressed in FS frame and can be transformed to the robot reference frame (R). Since H is fixed, point P with respect to Q can be expressed by Z and θ, which are called as lateral displacement and relative angle, respectively. To be specific, Z is the distance between Q and P along e_y, and θ is the angle between e_x and e_t. Furthermore, υ and ω are linear and angular velocities of MR, and l is the space between left and right motors. Lastly, ϕ is the angle of rotation from reference frame W to R.

From the figure, the position of point P relative to point O can be written as:

$$\begin{aligned} r_P &= r_Q + r_{P/Q} \\ r_P &= xE_x + yE_y + He_x - Ze_y \end{aligned} . \tag{1}$$

The rate of change of r_P relative to an observer in reference frame W can be expressed as:

$$\begin{aligned} {}^W v_P &= {}^W v_Q + {}^R v_{P/Q} + {}^W \omega^R \times r_{P/Q} \\ {}^W v_P &= \dot{x}E_x + \dot{y}E_y - \dot{Z}e_y + \omega e_z \times \left(He_x - Ze_y\right) \\ {}^W v_P &= \upsilon \cos\varphi E_x + \upsilon \sin\varphi E_y - \dot{Z}e_y + \omega He_y + \omega Ze_x \\ {}^W v_P &= (\upsilon + \omega Z)\, e_x + \left(\omega H - \dot{Z}\right) e_y \end{aligned} . \tag{2}$$

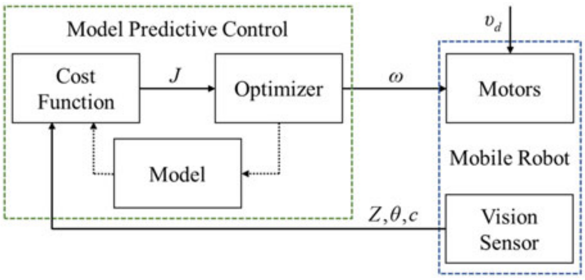


***Figure 2.** Block diagram of model predictive visual servoing for mobile robots.*

By definition in FS, ${}^{W}v_P$ is equal to $\frac{d}{dt}\left({}^{W}s\right)\mathrm{e}_t = \dot{s}\mathrm{e}_t$, where s is the arc length of the track. Using this definition, Eq. (2) can be rewritten in terms of FS basis as:

$$\dot{s}\mathrm{e}_t = (\upsilon + \omega Z)\,(\cos\theta \mathrm{e}_t + \sin\theta \mathrm{e}_n) + \left(\omega H - \dot{Z}\right)(-\sin\theta \mathrm{e}_t + \cos\theta \mathrm{e}_n)\,. \tag{3}$$

Finally, by multiplying basis $\mathrm{e}_t$and $\mathrm{e}_n$to Eq. (3) and using the relation $\dot{\theta} = \omega - \dot{s}c(s)$, the kinematic equations expressed in Z and θ are given in Eq. (4) [18, 21, 22]:

$$\begin{aligned} \dot{Z} &= \omega H + (\omega Z + \upsilon)\tan\theta \\ \dot{\theta} &= \omega - c\,(s)\,\frac{(\omega Z + \upsilon)}{\cos\theta} \end{aligned}\,, \tag{4}$$

where c denotes the path curvature. Consequently Eq. (4) can be translated into a state space form of a visual path-following model:

$$\begin{aligned} \dot{X} &= f\,(X, U) = \begin{bmatrix} \omega H + (\omega Z + \upsilon)\tan\theta \\ \omega - c\,(s)\,\dfrac{(\omega Z + \upsilon)}{\cos\theta} \end{bmatrix}, \\ X &= \begin{bmatrix} Z \\ \theta \end{bmatrix}, U = \begin{bmatrix} \omega \\ \upsilon \end{bmatrix} \end{aligned} \tag{5}$$

where, X and U are state and input vectors, respectively.

3. Model predictive visual servoing

Model predictive control is deployed as a base controller for path-following, which is explained in depth including, cost function, control parameters, stability analysis, and effects of path curvature on reference signals. In addition, vision sensor is the only source of measurements to provide feedback to the controller. Ways to extract the information from the image to measure the states and varying parameters are presented in the latter part of the section.

3.1. Model predictive control

Model predictive control is implemented for visual path-following, and the block diagram of the control scheme is depicted in Figure 2. It is clear from Eq. (5) that for any given linear velocity (υ), angular velocity (ω) can be used to change both states, Z and θ. In other words, for a desired linear velocity (υ_d) prescribed by the user, the controller only needs to synthesize a signal for ω to follow the track. Specifically, the vision sensor (i.e., camera) perceives Z, θ, and c from the image and sends them to the controller. Then the controller computes the optimal ω that minimizes the cost function J based on the finite horizon prediction of the states of the system model.

Table I. *Selected model predictive control parameters.*

Parameter	Value	Parameter	Value
N_p	6	q_Z	1
N_c	2	q_θ	26
N_u	3	ω_{lb}	-1 rad/s
ρ	0.001	ω_{ub}	1 rad/s
μ	10		

The cost function defined for this MPC problem is as follows:

$$\begin{aligned}
&J(k) = \sum_{j=1}^{N_p} \left(X(k+j) - X_r\right)^{\mathrm{T}} Q \left(X(k+j) - X_r\right) + \rho \sum_{j=1}^{N_u} \Delta\omega(k+j-1)^{\mathrm{T}} \Delta\omega(k+j-1) \\
&\Delta\omega(k+j-1) = \omega(k+j-1) - \omega(k+j-2) \\
&X = [Z, \theta]^{\mathrm{T}}, X_r = [Z_r, \theta_r]^{\mathrm{T}} \\
&Q = \begin{bmatrix} q_Z & 0 \\ 0 & q_\theta \end{bmatrix}
\end{aligned}, \tag{6}$$

where N_p and N_u are the prediction and control horizon, respectively, Q is a state weighting matrix, ρ is a control weighing factor, and X_r is a reference state. It can be observed from the cost function that the goal is to minimize the difference between system and reference states, i.e., X and X_r, and to stabilize the changes in angular velocity to control the system efficiently. The linear velocity term is not included in the cost function since it is a user–defined value, as indicated in Figure 2. To ensure the stability, the terminal constraint must be incorporated. The detailed proof of stability for nonlinear MPC has been reported in other MPC literatures [23–26]. Consequently, the cost function can be rewritten as shown in Eq. (7):

$$\begin{aligned}
J(k) = &\sum_{j=1}^{N_p} \left(X(k+j) - X_r\right)^{\mathrm{T}} Q \left(X(k+j) - X_r\right) + \rho \sum_{j=1}^{N_u} \Delta\omega(k+j-1)^{\mathrm{T}} \Delta\omega(k+j-1) \\
&+\mu \sum_{j=1}^{N_c} \left(X\left(k+N_p+j\right) - X_r\right)^{\mathrm{T}} Q \left(X\left(k+N_p+j\right) - X_r\right)
\end{aligned} \tag{7}$$

where μ is a weighing factor of the terminal cost, and N_c is a constraint horizon. The optimization problem of nonlinear MPC based on the cost function in Eq. (7) is summarized as follows:

$$\begin{aligned}
&\omega(k) = \arg\min_{\omega} J \\
&\text{s.t.} \quad \omega_{lb} \le \omega(k+j-1) \le \omega_{ub}, \forall j \in [1, N_u], \\
&\qquad \Delta\omega(k+N_u+j) = 0, \forall j \ge 0
\end{aligned} \tag{8}$$

where ω_{lb} and ω_{ub} are lower and upper bounds of ω, respectively. For this research, the SQP method with a maximum of 50 iterations, and a step size of 1e-3 is implemented to find the optimal control signal. The control parameters selected for this work are shown in Table I. In general, increasing the values of N_p, N_c, and N_u enhances the performance but at larger computational cost of optimization. Therefore, they are selected to be the maximum values that enable the optimizer can compute faster than the control frequency (set to be 20 Hz) without appreciably compromising numerical accuracy. The bounds of the angular velocity, that is, ω_{lb} and ω_{ub}, are chosen based on the specification of the MR. The four weighting parameters: ρ, μ, q_Z, and q_θ, are selected by prior experience and trial and error.

The goal of MPC-based path-following is to minimize the distance between the center of the robot (Q) and the track. Simply minimizing Z and θ, i.e., zero reference state will cause the point P to follow the track, instead of Q. For a curved path, the robot actually deviates from the path if Z and θ are zero,

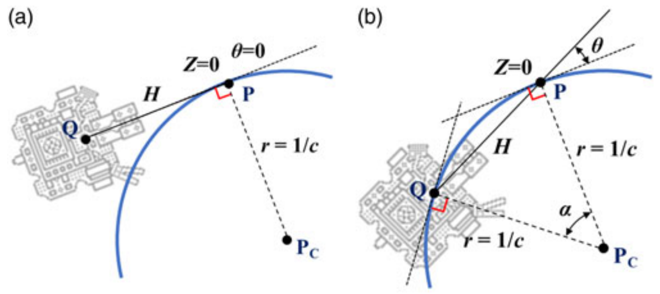


Figure 3. *Path-following of MR on a curved path with: (a) The zero relative angle and (b) The nonzero relative angle.*

as illustrated in Figure 3a. In Figure 3b, it is clearly shown that θ must be a nonzero value to ensure both points Q and p are on the track. Assuming a constant curvature from Q to P, desired θ can be determined, and thus, Q stays on the path while keeping zero lateral distance (i.e., $Z = 0$). An arc with a constant curvature c can be drawn as a circle with center $\mathrm{P_C}$. The distance from P to $\mathrm{P_C}$, and from Q to $\mathrm{P_C}$ are both equal to the radius of the circle r. Since the shortest distance between Q and P is equal to H, the following equation with α can be satisfied

$$\cos\alpha = \frac{H^2 - 2r^2}{-2r^2}. \tag{9}$$

Since our goal is to find θ, which would allow Q and P to be on the path with curvature c, Eq. (9) can be rewritten in terms of θ using the relation $\alpha = 2\theta$. Then with the double–angle formula, θ can be found as:

$$\begin{aligned} \cos 2\theta &= \frac{H^2 - 2r^2}{-2r^2} \\ 1 - 2\sin^2\theta &= \frac{H^2 - 2r^2}{-2r^2} \cdot \\ |\theta| &= \sin^{-1}\frac{cH}{2} \end{aligned} \tag{10}$$

θ from Eq. (10) can be used as the reference angle (θ_r) in Eq. (6) with $Z_r = 0$. There are several points of note. First, θ_r varies with the curvature, and therefore, it must be updated for every c measurement. Second, the sign of θ_r is positive when the curve opens to the right, and vice versa according to the image acquired by the onboard camera. Third, the curvature is assumed to be consistent for the entire prediction horizon, mainly because the image used to predict the curvature includes the path that the robot has not reached yet. In other words, the measured curvature is the average curvature of the path captured in the field of view (FOV) of the onboard camera.

3.2. Image-based measurement

To enable visual servoing, information from the image must be accurately extracted. Before measuring lateral distance (Z), relative angle (θ), and curvature (c), the track must be identified in the acquired image. Figure 4 presents the procedure of track detection using the onboard camera feed. The camera takes a raw image (Figure 4a) of 640×480 pixels, which is then converted into the hue, saturation, and value (HSV) color space (Figure 4b). Every pixel in the HSV color space consists of three values ranging from 0 to 255, respectively, representing HSV. Since the track is in blue, a chroma key filter is applied to detect the track, and thus, the track is represented in white on a black background (Figure 4c). A byproduct of the filter is a 640×480 binary matrix, where 1 represents the path (white) and 0 represents the background (black).

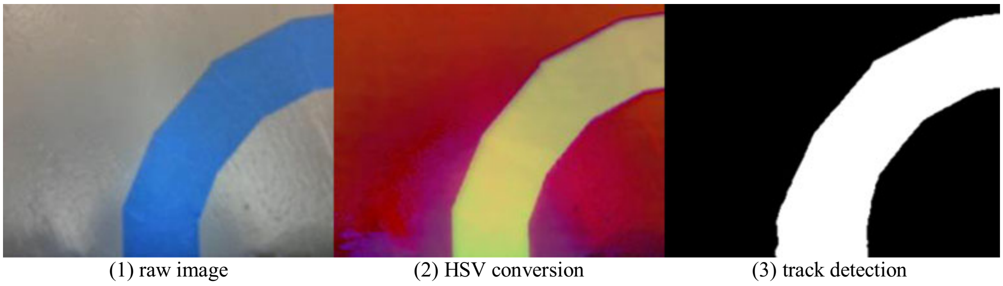


Figure 4. *Track detection procedure of the onboard camera.*

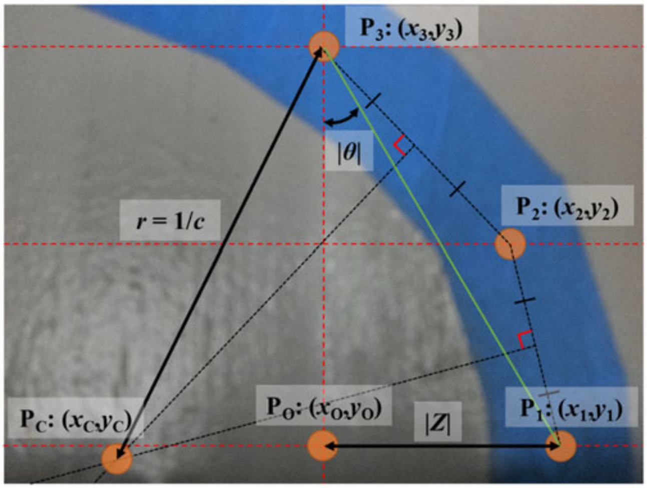


Figure 5. *Schematic of image analysis.*

In order to obtain quantitative values of Z, θ, and c from the binary image, five points need to be used, as shown in Figure 5. Prior to identifying the points, three rows (i.e., the 48th, 240th, and 432th) and a central column are selected as references, indicted by the red-dashed lines. Notably, the central vertical line aligns with e_x in Figure 1. The point P_O is located at the intersection of the central vertical line and the horizontal line at row 432. This point corresponds to P in Figure 1, and Z is measured from this point. Points P_1, P_2, and P_3 are the midpoints of the track at rows 432, 240, and 48, respectively. By assuming that the path has a consistent curvature, the center of the circle P_C that passes through points P_1, P_2, and P_3 can be identified. The equations for Z, θ, and c using the identified points are then given by Eq. (11):

$$\begin{aligned} Z &= \gamma\ (x_1 - x_0) \\ \theta &= -\tan\left(\frac{y_3 - y_1}{x_3 - x_1}\right) \\ c &= \frac{\pm 1}{\gamma\sqrt{(y_3 - y_C)^2 + (x_3 - x_C)^2}} \end{aligned} \tag{11}$$

where γ is the pixel-to-cm conversion factor. Note that c is positive when the curve opens to the left, and vice versa.

Table II. *Data bounds of MR visual path-following.*

states and inputs	lower bound	upper bound
lateral distance Z (m)	-0.08	0.08
relative angle θ (rad)	-1.48	1.48
linear velocity υ (m/s)	0	0.25
angular velocity ω (rad/s)	-1.4	1.4
curvature c (m^{-1})	-50	50

4. Artificial neural network (ANN) mModel

The process of training the ANN model for MR's visual path-following is elucidated in this section. The training is divided in two stages: pretraining and finetuning. During the pretraining stage, the PB model identified in Eq. (5) from Section 2 is used to generate data, and the network is trained with this generated (PB) dataset. After the pretraining stage, test data collected from the actual system are used to finetune the network weights, mitigating internal disturbance of the PB model, such as modeling error and parameter uncertainty.

4.1. ANN pretraining

There are three objectives for this pretraining stage. First, it is expected that the amount of test data collected from the actual system can be reduced by using model data. Collection of the test data is both time-consuming and resource-demanding. Moreover, acquiring a wide variety of data is often difficult due to operational and safety considerations. Therefore, during this pretraining stage, ANN learns to approximate the actual system using the diversified dataset generated by the PB model. Second, with sufficient model-generated data, the structure of the ANN model can be configured precisely, more specifically, selecting the appropriate numbers of hidden layers and neurons. The structure of ANN significantly affects prediction accuracies. Therefore, it is essential to select the structure that can best represent the system, and meanwhile consider its purpose. Although this can also be performed with the test data, they are often limited in range and size, leading to an underperforming architecture and a poorly generalized model [27]. Lastly, by pretraining the network, suboptimal weights can be attained. It is well known that in supervised learning, gradient-based training can easily be trapped in the local minimum using the standard random initialization [28]. One workaround is to repeat the training process multiple times and select the model with the highest prediction accuracy. Therefore, this process can be conducted during the pretraining stage, and the test data are simply used to finetune the weights to eliminate the issue of random weight initialization.

The data for pretraining are generated by integrating the system model in Eq. (5) within the range specified in Table II. The defined ranges of the states and inputs are selected based on the actual MR operation, which will be introduced in Section 5. Random step signals are generated for υ and ω for intervals selected randomly between 0.1 and 0.5 s. Although c is not an input, it also needs to be supplied to the model since it is a geometric parameter that varies with the path profile. Thereby, the random step signal is also generated to represent c. All the generated signals are then supplied to the model for numerical integration, and the states of the system are bounded. In total, 1 h of data are generated with a step size of 0.05 s, equivalent to the control interval chosen for this work.

The number of hidden layers and neurons must be selected to determine the appropriate size of the ANN model. It is important to note that the purpose of the ANN model in this work is to integrate it to the NMPC framework for finite horizon predictions. Since solving NMPC at every timestep is time-consuming, it would be beneficial to select a model with a relatively short inference time. Thus, horizon predictions can be completed rapidly. In other words, shallow networks with a small number of neurons are preferred. The ANN model is trained iteratively by varying the size of hidden layers and neurons to find the model that best predicts the test data. The number of the hidden layer is varied from 1 to 3, and

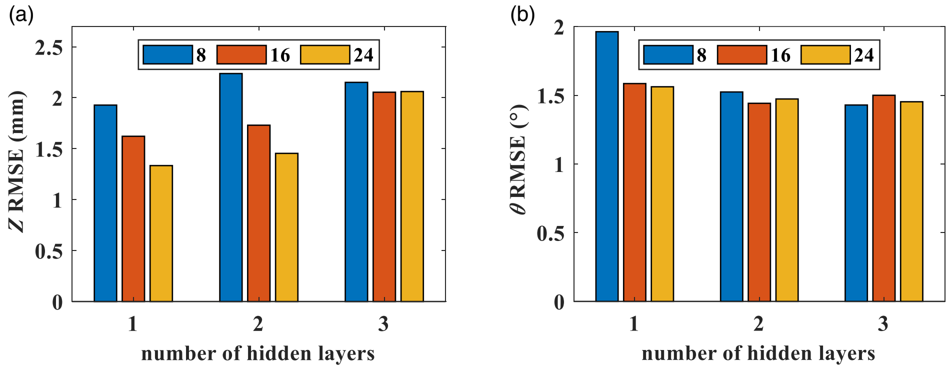


Figure 6. *RMSEs of (a) Z and (b) θ predictions for different ANN architectures.*

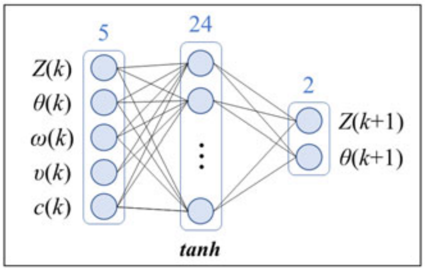


Figure 7. *Structure of ANN model.*

three different numbers of neurons are explored, which are 8, 16, and 24. Larger number of neurons and hidden layers are not considered in this work to limit the size of the ANN model for fast inference. To simplify the search, a consistent number of neurons is applied to all hidden layers, and for all hidden neurons, the hyperbolic tangent activation function is adopted. In total, ANNs with nine different structures are trained, and for each structure, training is repeated ten times to mitigate the aforementioned issue associated with random weight initialization. For all training instances, train and test set ratios are defined to be 0.85 and 0.15, respectively, and 20% of train set is used for validation. The same test set is applied to all models for a fair comparison. Training is performed for 1000 epochs using the mean squared error loss function and the Levenberg–Marquardt optimization algorithm. However, the training is terminated sooner when the loss function no longer decreases, or the gradient drops below a threshold. Out of 10, the best model is selected for each architecture, and root mean square errors (RMSEs) of Z and θ predictions are illustrated in Figure 6. Different color bars represent different numbers of neurons: blue, orange, and yellow colors represent 8, 16, and 24 neurons. As shown in the figure, all models perform well and exhibit RMSEs of less than 2.5 mm and 2° for Z and θ, respectively. Surprisingly, models with three hidden layers perform worst for Z predictions, although a single hidden layer yields best Z prediction. However, increasing the number of the hidden layers improves θ prediction. Generally, both predictions improved as the number of neurons grew. By normalizing the two RMSEs and considering their sum, the ANN structure with a single hidden layer comprised of 24 neurons seems the best structure. The schematic of the selected ANN architecture is portrayed in Figure 7. It is important to note that there are five inputs to the ANN model including two current states (Z and θ), two current inputs (υ and ω), and path curvature (c). Here, curvature must be applied as the extra input since it is a varying parameter determined by the external factor. Moreover, the current state is also implemented as the input since without the initial state, states for the future timesteps cannot be determined.

Table III. *Control parameters used for data collection.*

Control Parameters	Values
υ	5, 10, 15, 20 (cm/s)
ρ	0, 1e-5, 1e-4, 1e-3, 1e-2
N_u	1, 2, 3
N_p	1, 2, 3, 4, 5, 6 ($N_p \geq N_u$)

4.2. ANN finetuning

Although the visual path-following model shown in Eq. (5) may be a good representation of the system, it is still subjected to modeling errors from unknown dynamics and parameter uncertainties. Therefore, actual experimental data of MR must be captured to update the pretrained ANN model through transfer learning. The training condition for finetuning remains consistent with that in the pretraining stage. Since the path must be available within the camera's FOV to extract Z, θ, and c, it would be more convenient to capture the data while MR is under control. The NMPC introduced in Section 3 is implemented for data collection, in which the PB model is used for horizon prediction. It should be noted that regardless of control performance, as long as the path is within the FOV, the data can be used for training since dynamics of the system is independent of the control algorithm. However, to diversify the data collection, control parameters of MPC and linear speed of the robot are varied by randomly selecting the values in Table III.

In total, 1 h data are captured, which are equal to the size of the generated data. To further investigate and compare the captured and generated datasets, distribution plots are depicted in Figure 8. Except for the linear velocity, all other features of the captured dataset are highly concentrated near zero, whereas the generated dataset manifests broader distribution throughout the range. Furthermore, the captured dataset only possesses four distinct linear velocities because the data are produced by selecting one of the four predefined velocities. The distribution histograms between the two datasets verify the necessity of the pretraining stage for training well–generalized ANN models for control synthesis. That is, even though the control parameters and the linear velocity are varied in wide ranges for data collection, the data collected in the experiments can still be limited in diversity.

5. Experiment setup

5.1. Mobile robot

The Turtlebot 3 Waffle Pi (Turtlebot) is selected as the demonstration platform for visual path-following, and is shown in Figure 9. The Turtlebot has two Dynamixel servo motors as actuators to generate linear and angular velocities. To ensure sufficient computing capacity, the NVIDIA Jetson TX2 computing device is used as the main microcontroller unit. The Jetson TX2 product is comprised of 256-core NVIDIA Pascal GPU, Dual-core Denver 2 64-bit CPU and quad-core Arm® Cortex®-A57 MPCore processor, and 8 GB 128-bit LPDDR4 memory. For the vision sensor, the e-con Systems e-CAM 132 camera with a 13 megapixel autofocus feature is used. The images are acquired in video graphics array format with the resolution of 640×480 at 40 frames per second (fps). As shown in the figure, the camera is installed in such a way that its focal axis angle is close to zero, that is, facing down and parallel to the surface to minimize the distortion of the image. The Turtlebot is operated by the robot operating system (ROS), comprised of a set of tools and libraries that enable a convenient environment to operate the robot, and the specific ROS version used in this work is Melodic.

5.2. Overhead camera tracking system

In order to verify control performance, the position of the MR with respect to the path must be determined and recorded. Although the onboard camera provides error estimation, it is not measured relative

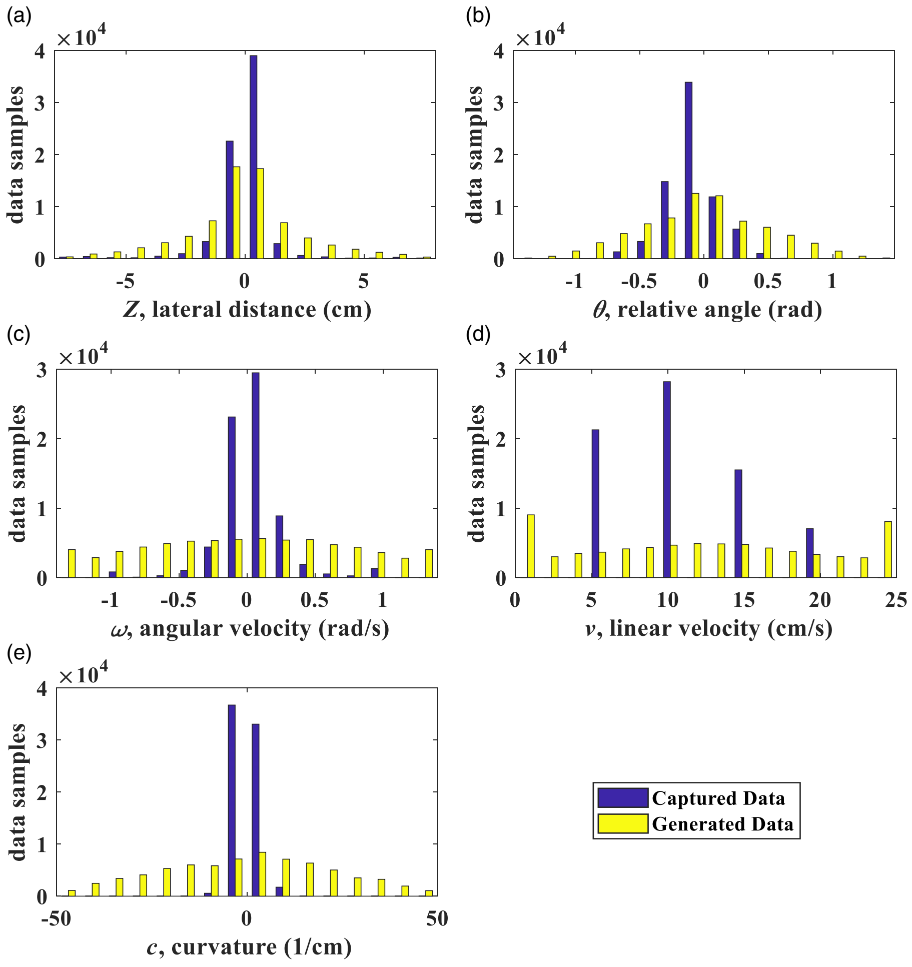


Figure 8. *Distribution plots of captured and generated datasets: (a) Z, (b) θ, (c) ω, (d) υ, and (e) c.*

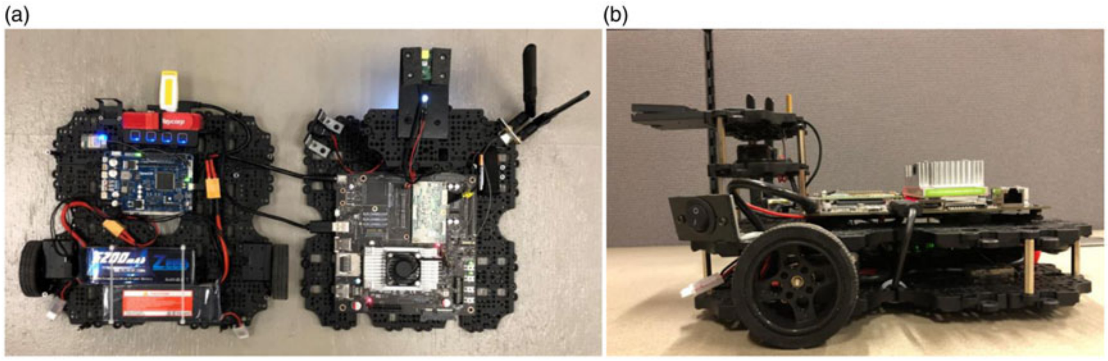


Figure 9. *Images of the mobile robot (Turtlebot 3 Waffle Pi): (a) Top view of each layer, and (b) Side view.*

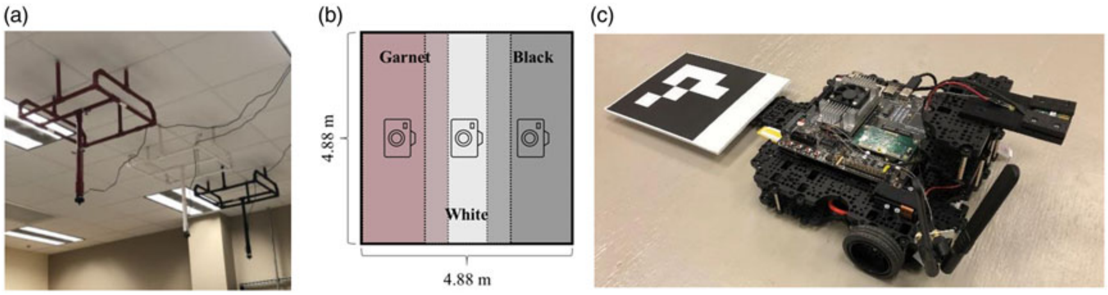


Figure 10. *(a) Image of the overhead camera system, (b) Field of views on the track surface, and (c) Image of the robot with ArUco marker attached.*

to the robot's center but from a point selected in the camera's FOV (i.e., P_O in Figure 5). Therefore, an external means to evaluate the deviation between the MR and the prescribed path is required. In this work, an in-house tracking system comprised of three wide-view overhead cameras is utilized to monitor the entire range of the track surface (4.88 m × 4.88 m) and localize the MR as illustrated in Figure 10a-b. The overhead cameras are attached to 4 degrees of freedom mounts colored in garnet, white, and black. Hereafter, cameras are referred tom the colors of the mounts, and the FOV of each camera that covers the track is illustrated in Figure 10b. In order to identify the robot on the track, the ArUco marker, a binary fiducial marker, is employed. Four corners of the ArUco marker are detected from the image, which allows to estimate position and orientation of the robot. Figure 10c shows the image of the MR with the ArUco marker attached at the rear close to the ground. Since the relative position between the ArUco marker and the center of MR is known, the position of the latter can be postcalculated once the former is found. The marker is placed close to the ground to minimize the height difference with respect to the track, and thus, the lateral distance between the track and the marker can be measured more accurately because of their similar heights. This is important because ArUco tracking does not provide accurate height information in the planar motions, and placing ArUco marker on the top of the robot could cause appreciable errors due to perspective angles of the camera. To further minimize the ArUco tracking error and improve its position measurement, a calibration procedure based on polynomial regression is applied. The track is identified by the overhead cameras in the same manner as the onboard camera, which is shown in Figure 4.

Because the FOV of a single camera is not sufficient to cover the entire track, use of multiple cameras becomes necessary, each covering different parts of the track surface. Thus, a mechanism to switch between different cameras is needed to allow only one camera to track MR at a time. In addition, ArUco tracking localizes the marker and returns zero if the marker is not within the FOV of the camera. This avoids the need for identifying the borders of cameras' FOVs, because the camera whose FOV covers the marker can be determined by examining the returned values of all cameras. As the marker moves from the FOV of one camera to another, a switch can be executed to select the desired camera for tracking. Nonetheless, as depicted in Figure 10b, there are two shared zones of FOVs: garnet/white and white/black. In the shared zones, the marker can be detected by two cameras simultaneously. It is possible to take the average readings from both. However, when the robot just enters the FOV of one camera, it takes some time to receive reliable readings. Therefore, the idea of a hysteresis switch is adopted for more robust and accurate tracking and localization. As illustrated in Figure 11, the robot moves between FOVs of A and B. There exists a shared zone where the MR can be identified from both FOVs, with boundaries S_{BA} and S_{AB}. The switching mechanism is then devised in such a way that when the robot moves from A to B, camera A remains active and the measurement from B is discarded till it crosses S_{AB}. On the other hand, if the robot moves from B to A, the switch is activated when the robot crosses S_{BA}. Thus, rapid switching between FOVs is prevented, allowing sufficient time to stabilize the readings.

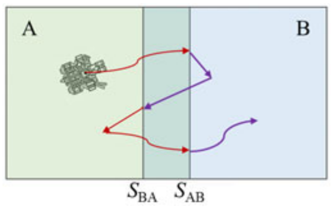
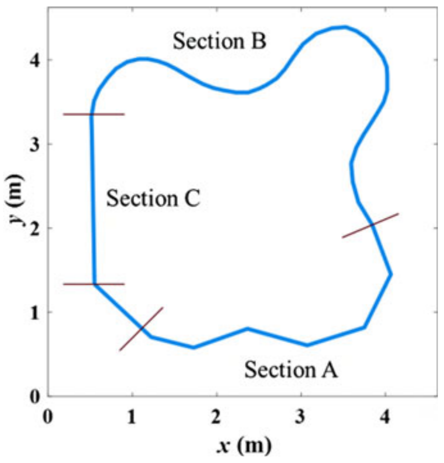


Figure 11. *Switching mechanism for overhead camera system.*



Section	Course
A	Sharp Turn
B	Smooth Turn
C	Straight

Figure 12. *Layout of the track.*

6. Results and Discussion

For performance comparison, two additional ANNs are trained: using either model–generated data or experimentally captured data without going through the two-stage learning above. To be specific, three networks with identical structures are trained: without pretraining (w/o PT), that is, only with captured data; without finetuning (w/o FT), that is, only with model-generated data; and in two-stages (in TS) with model-generated data for pretraining and captured data for finetuning. These models are then compared in control experiments. The MR is placed at the consistent starting location for each trial of the path-following experiment. From the starting point, the robot is under autonomous control to follow the track solely using the vision sensor. It stops at the same designated location unless it loses the path and permanently deviates from the track. As demonstrated in Figure 12, the track consists of three sections with different courses: sharp turn, smooth turn, and straight, represented as A, B, and C, respectively. Experiments are performed (illustrated in Figure 13) with two different specified linear velocities: 15 cm/s and 17.5 cm/s for all three networks. Each trial is repeated ten times to examine the consistency and robustness of MPC.

6.1. Path-following with the linear velocity of 15 cm/s

Experiments with the linear velocity of 15 cm/s was performed because, as shown in Figure 8, and it is one of the four designated velocities used to capture the actual data for model training. Moreover, if the linear velocity is too low, it would become difficult to distinguish the control performance because the model prediction error has less effect on control and can be mitigated immediately within a smaller distance. Both mean absolute error (MAE) and RMSE of the robot's position relative to the track are computed for each trial. The errors are measured by the overhead camera tracking system, as presented in Section 5.2. With the 10 repeated trials, the average, minimum, maximum, and standard deviation

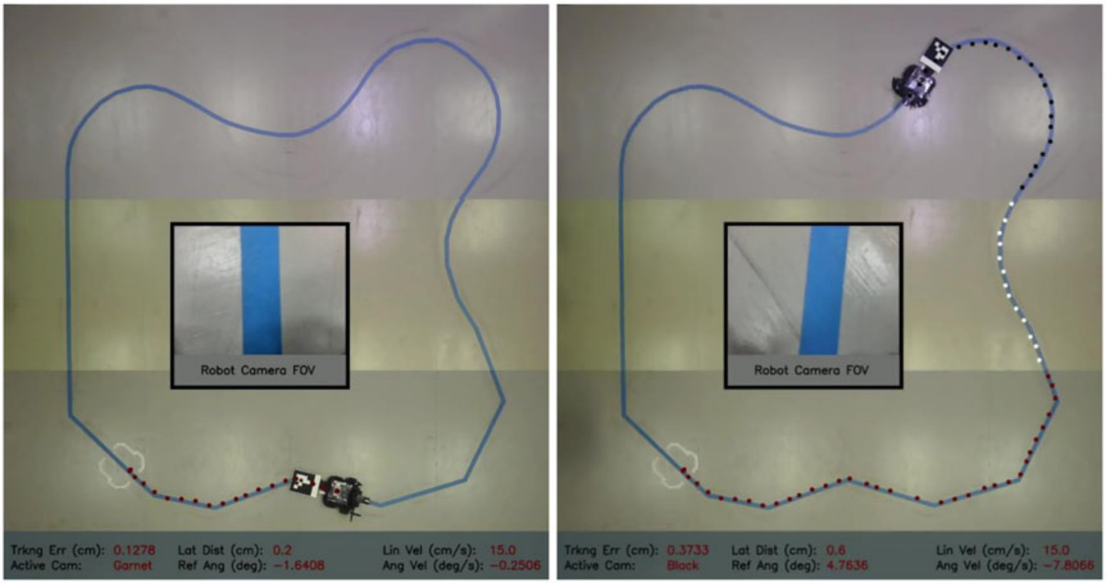


Figure 13. *Experiment snapshots.*

of MAE and RMSE for each network are also calculated and summarized in Table IV. It clearly shows that the network trained without pretraining, that is, only using the captured data for training, exhibits the worst performance. In fact, the robot was not able to follow the track or make it to the end in any of the trials, manifesting large errors and significant deviation. On the other hand, the network trained without finetuning showed reasonable and consistent performance for all ten trials. The average of MAE and RMSE is calculated to be 0.51 cm and 0.74 cm, respectively, which are approximately 6 and 7 times smaller than those of the network without pretraining. This indicates that the PB model in Eq. (5) used to generate pretraining data is a good representation of the actual system. Nonetheless, the network trained with the two-stage approach exhibits the best performance, implying that the model error is compensated through the finetuning stage by updating the model weights with the actual data. On average, compared to the network trained without finetuning, the performance is improved by approximately 0.15 cm.

To further investigate the performance of each network, the measurements: lateral distance (Z), relative angle (θ), angular velocity (ω), and position error (ε) are plotted against time. Furthermore, position snapshots of the robot are provided for visual verification, in which the actual path and the robot's position are denoted by a solid blue curve and a dashed red curve, respectively. The measurements and position snapshots for the three network models trained with different approaches are depicted in Figure 14, Figure 15, and Figure 16, respectively. In all figures, measurements are shown at the top, and the corresponding snapshots are placed below. In addition, the start and end locations of the robot are marked by a square and a circle, respectively. Out of 10 trials, two representative runs: (a) best and (b) worst cases according to the average of MAE are presented. However, for the network without pretraining, since none of the trials completed the run, the best and worst cases are determined by the length of the robot's path before the track loses its presence in the FOV of the onboard camera. To aid in comparison, position errors of best cases for all three models are plotted in the same figure, as shown in Figure 17.

Figure 14 clearly shows that in the best case, the robot deviated from the path toward the end of Section B. In the worst case, the robot did not make through the first sharp corner, terminating the trial in 5 s. The FOV of the onboard camera is approximately 16 cm, and Z ranges from −8 cm to 8 cm. As the line disappears from the vision, Z reaches its maximum value, which terminates the mission. On the other hand, Figure 15 presents the performance of network trained with only model-generated data

Table IV. *MAE and RMSE of path-following experiments with the linear velocity of 15 cm/s.*

	MAE (cm)				**RMSE (cm)**			
	Average	**Min**	**Max**	**STD**	**Average**	**Min**	**Max**	**STD**
w/o Pretraining	3.24	0.61	20.08	6.00	5.06	1.03	22.79	6.68
w/o Finetuning	0.51	0.38	0.59	0.06	0.74	0.61	0.85	0.08
in Two-stages	0.38	0.34	0.43	0.03	0.55	0.51	0.60	0.04

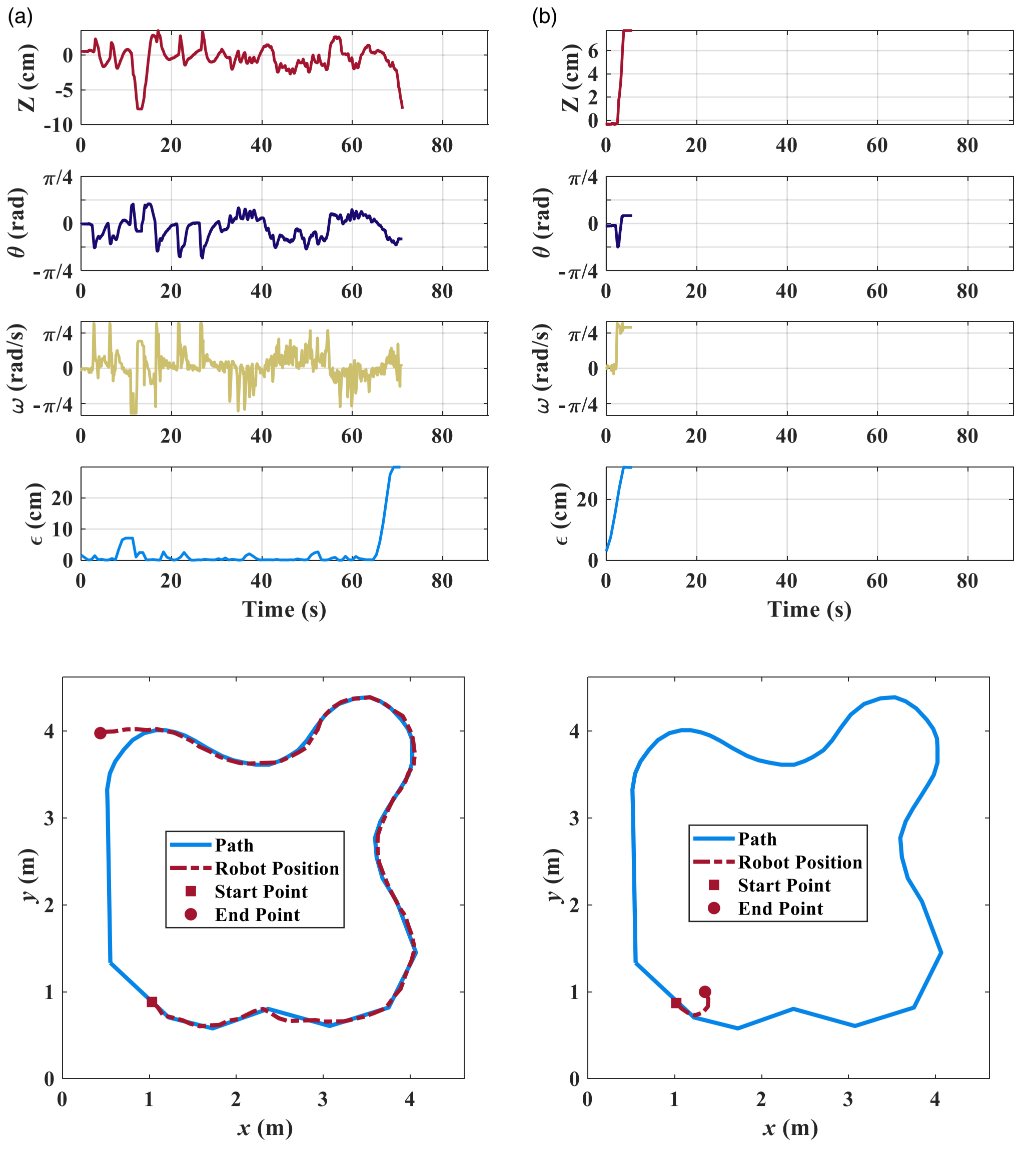


Figure 14. *Performance of model predictive control for path-following with the linear velocity of 15 cm/s and using the model trained without pretraining: (a) Best and (b) Worst performances based on the length of the path traveled by the robot.*

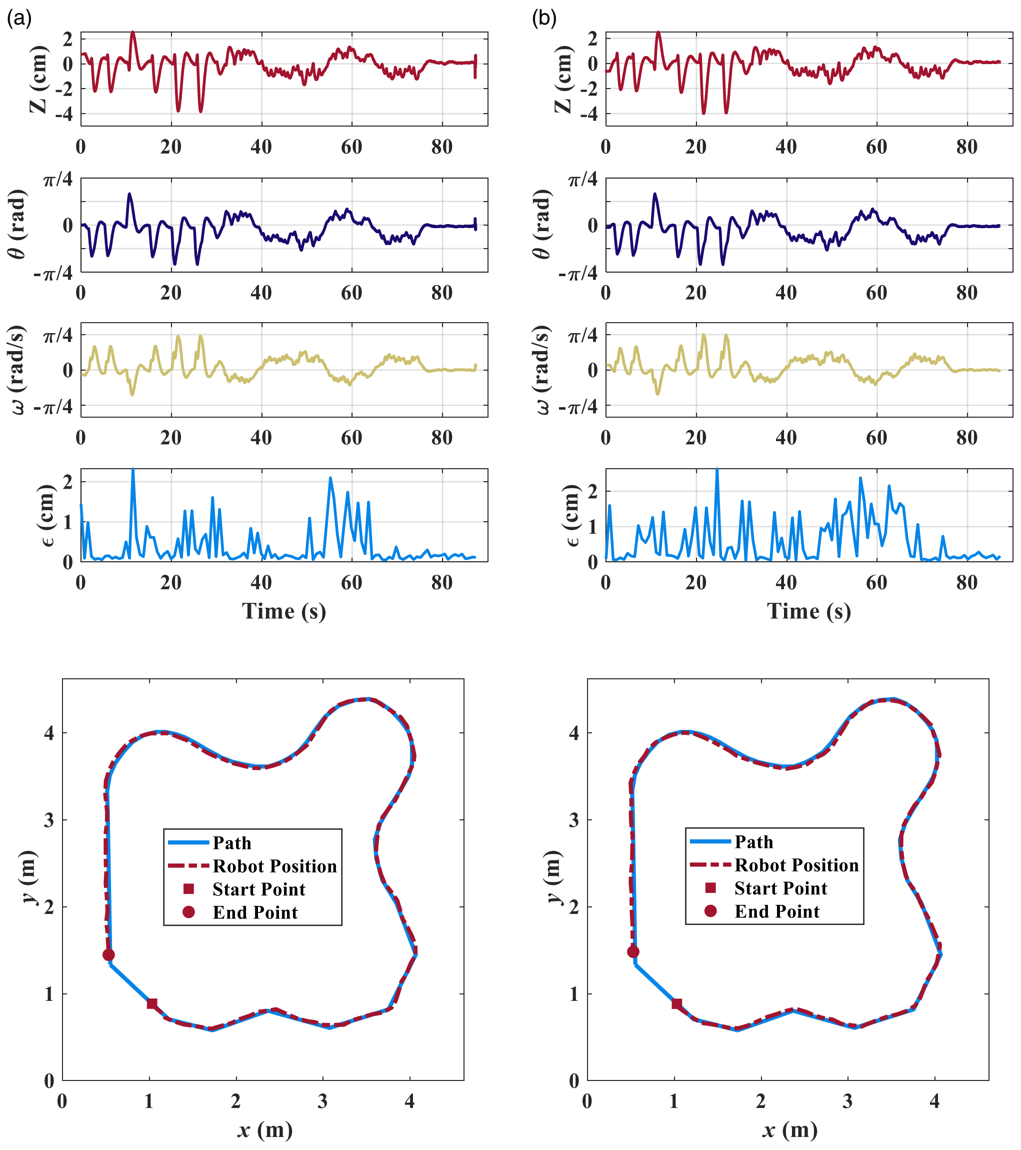


Figure 15. *Performance of model predictive control for path-following with the linear velocity of 15 cm/s and using the model trained without finetuning: (a) Best and (b) Worst performances based on MAE.*

with salient diversity. It is evident from the figure that the generalization of the model directly dictates the robustness of the MPC. With the model trained with the generated data, the robot was able to reach the goal point for all trials. Furthermore, as indicated in Figure 15, the best and the worst cases are not distinctly different from each other, confirming the consistency of the performance. Lastly, Figure 16 illustrates the performance of network trained with both fidelity of data. The position error plots in Figure 16a and Figure 16b exhibit slightly lower errors than that the position error plots in Figure 15a and Figure 15b, implying that the finetuning using the actual data collected in experiments allowed the model to represent the system more precisely. As it was for the case without finetuning, for all trials, the robot reached the destination and the difference between the best and the worst cases is negligible. In all three figures, as long as the track is in the FOV of the onboard camera, Z and θ follow the track

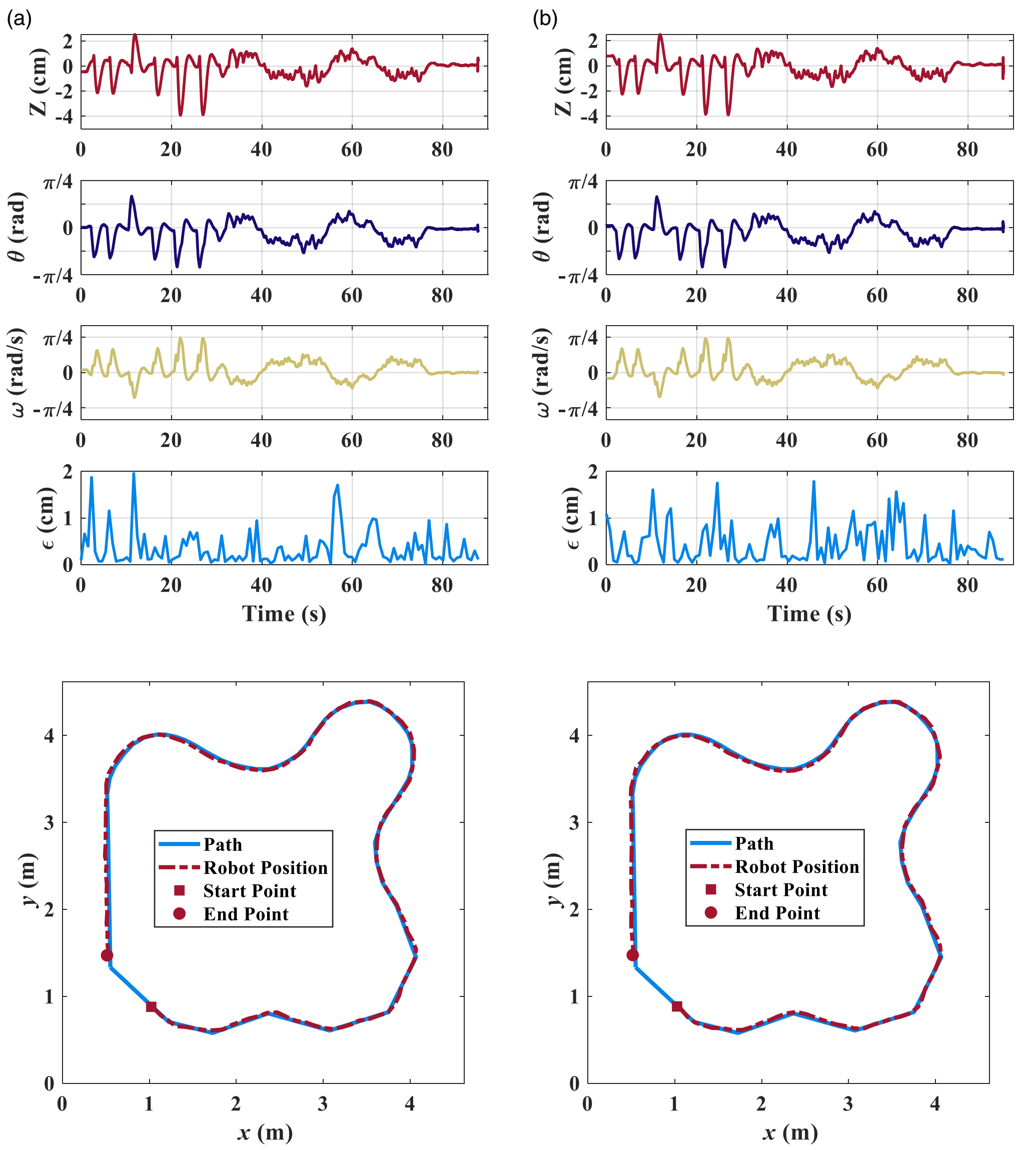


Figure 16. *Performance of model predictive control for path-following with the linear velocity of 15 cm/s and using the model trained in two stages: (a) Best and (b) Worst performances based on MAE.*

profile. In Section A, at the instant of making the sharp turn, Z and θ show sudden increment. After the turn, as the MR continues to follow the track, Z and θ become close to zero again. In Section B, Z and θ are nonzero because the curvature is continuously changing. In the last section of the track, Z and θ remain close to zero since the curvature of a straight path is zero. It is essential to compare the control signal, ω, applied to the robot for different networks. In Figure 15 and Figure 16, the profile of ω is commensurate with the values of Z and θ, and the control signal changes smoothly. However, ω in Figure 14 fluctuates significantly, and its variation does not align with Z and θ, leading to poor control performance. The significant model error yields completely wrong control signals, causing the robot to deviate from the track permanently. Figure 17 simply reflects the result presented in Table IV. The

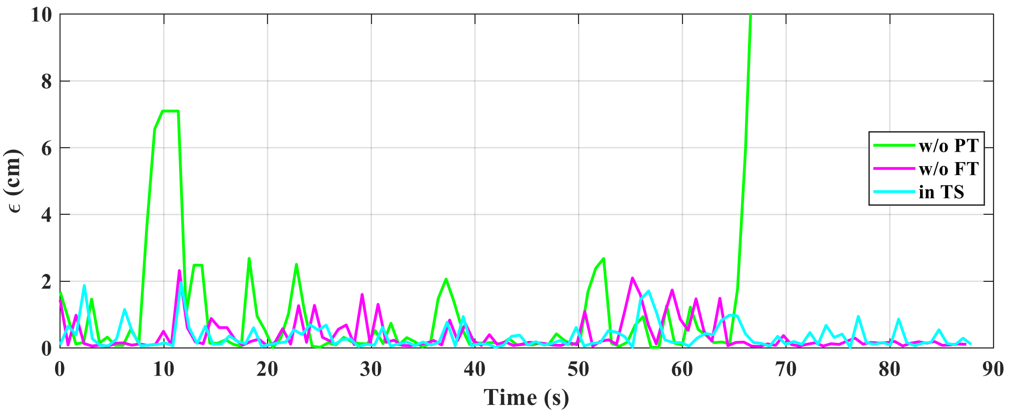


Figure 17. *Position error comparison for path-following with the linear velocity of 15 cm/s.*

ANN trained in two-stages exhibits the smallest error in average, followed by the ANN trained without finetuning.

6.2. Path-following with the linear velocity of 17.5 cm/s

All the experiments were repeated with the linear velocity of 17.5 cm/s. Since the captured dataset does not include this particular linear velocity, it is essential to examine the control performance with unseen data and verify the generality of the model. Equivalent to the previous section, MAE and RMSE are calculated for three different network models for all 10 trials, which are presented in Table V. According to the table, these trials revealed similar results. The network without pretraining shows the worst performance and is not able to completely follow the path in all trials. Surprisingly, their MAE and RMSE are smaller in magnitude compared to those with linear velocity of 15 cm/s. Recall that the center of the robot is always behind the camera's FOV and the lateral distance (Z) and relative angle (θ) are measured within the FOV. However, the robot does not terminate its movement based on the position error, but when the track is no longer detected by the camera. Thus, the position errors right before the terminal may exceed 20 cm according to the orientation of the robot and the path profile. In other words, more significant position errors do not always reflect poor performance in those incomplete trials. In contrast, the network trained without finetuning performs better and exhibits approximately 2 and 3 times lower MAE and RMSE, respectively. Again, the network trained with the two-stage approach surpasses the other two, although the positive error only improves about 0.07 cm when the linear speed is increased to 17.5 cm/s. This is also expected because the data used to finetune the network weights do not contain that with the linear speed equivalent to 17.5 cm/s. Overall, the two-stage training approach outperforms the one only using model-generated data.

Similarly, for more elaborate analysis, states, inputs, and position errors versus time, and the position snapshots are all portrayed in Figure 18, Figure 19, and Figure 20, respectively, for the networks trained without pretraining, without finetuning, and with the two-stage approach. The organization and annotations of these figures are the same as Figure 14, Figure 15, and Figure 16. Similarly, two representative trials: (a) the best and (b) the worst cases, are plotted according to MAE. Again, position errors of best cases for all three models are plotted in the same figure for the sake of comparison, as depicted in Figure 21.

In Figure 18, in the best-performed case, the robot made through half of Section B, and in the worst case the robot lost the track in the middle of Section A. Angular velocity signals from the figure exhibit

Table V. *MAE and RMSE of path-following experiments with the linear velocity of 17.5 cm/s.*

	MAE (cm)				RMSE (cm)			
	Average	Min	Max	STD	Average	Min	Max	STD
w/o Pretraining	1.47	0.58	2.49	0.65	2.92	0.84	5.73	1.57
w/o Finetuning	0.72	0.60	0.80	0.06	1.04	0.96	1.09	0.04
Two-stages	0.67	0.54	0.74	0.05	0.96	0.80	1.04	0.07

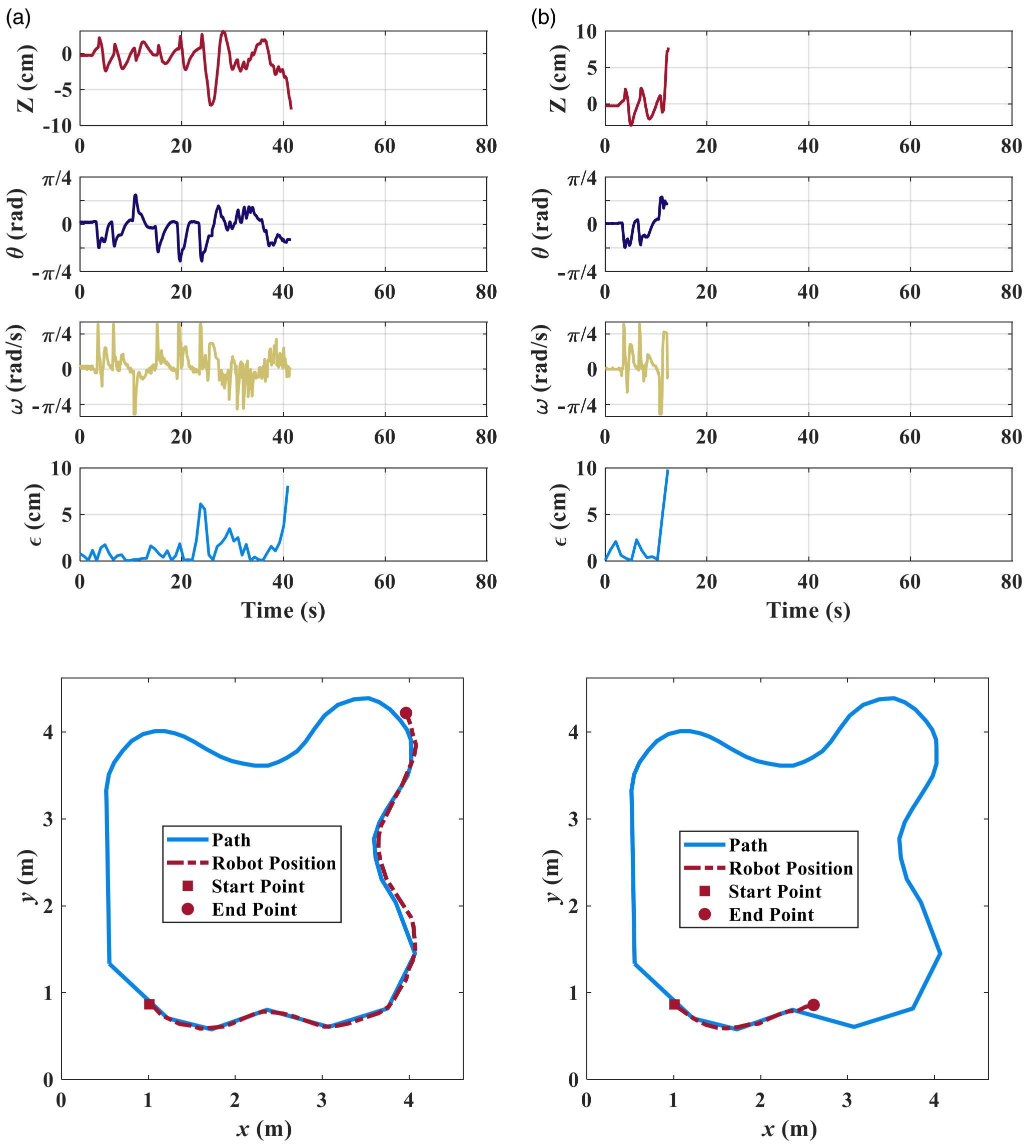


Figure 18. *Model predictive path-following performance with the linear velocity of 17.5 cm/s and using the model without pretraining: (a) The best and (b) The worst performances based on the length traveled by the robot.*

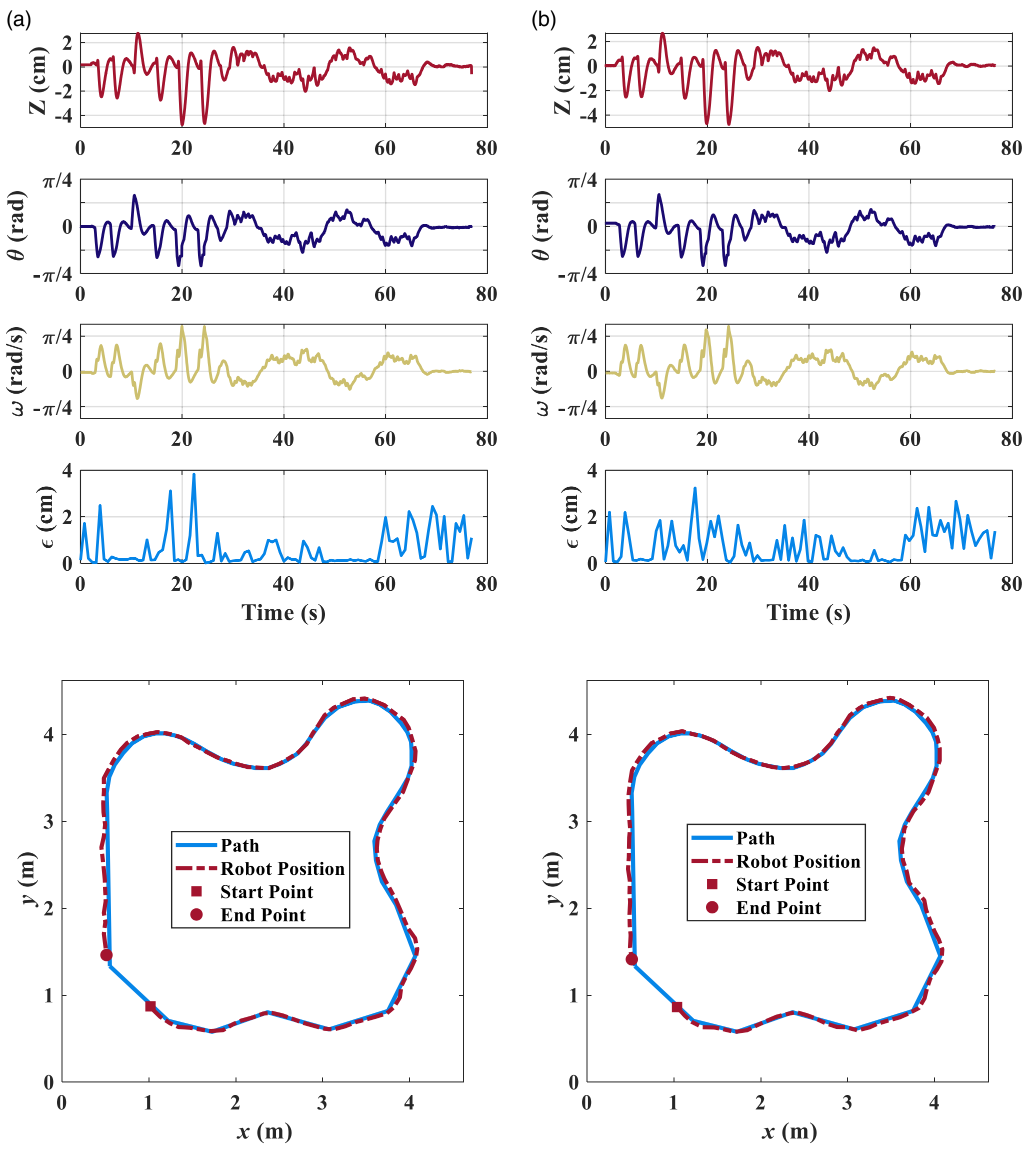


Figure 19. *Model predictive path-following performance with the linear velocity of 17.5 cm/s and using the model without finetuning: (a) Best and (b) Worst performances based on MAE.*

large variations and are inconsistent to the profile of Z and θ, indicating poor model predictions. For the other two models, none of the trials failed, and all reached the designated goal point. As shown in Figure 19 and Figure 20, the robot was able to complete the run within 80 s because of the larger linear velocity. For the same reason, the maximum position error is now close to 4 cm, which occurs more frequently in Section A. Compared to the 15 cm/s case, the position error in Section C seems slightly larger. This is because when the robot traverses from Section B to C, a peak error is introduced. The effect of this peak error becomes more significant at a higher speed, and it takes longer to mitigate the error. Furthermore, the angular velocity (ω) applied to the system matches the profile of Z and θ in Figure 19 and Figure 20. Performances of the best and the worst cases are similar, which implies consistency in control. Figure 20 shows a slightly smaller position error compared to Figure 19, confirming

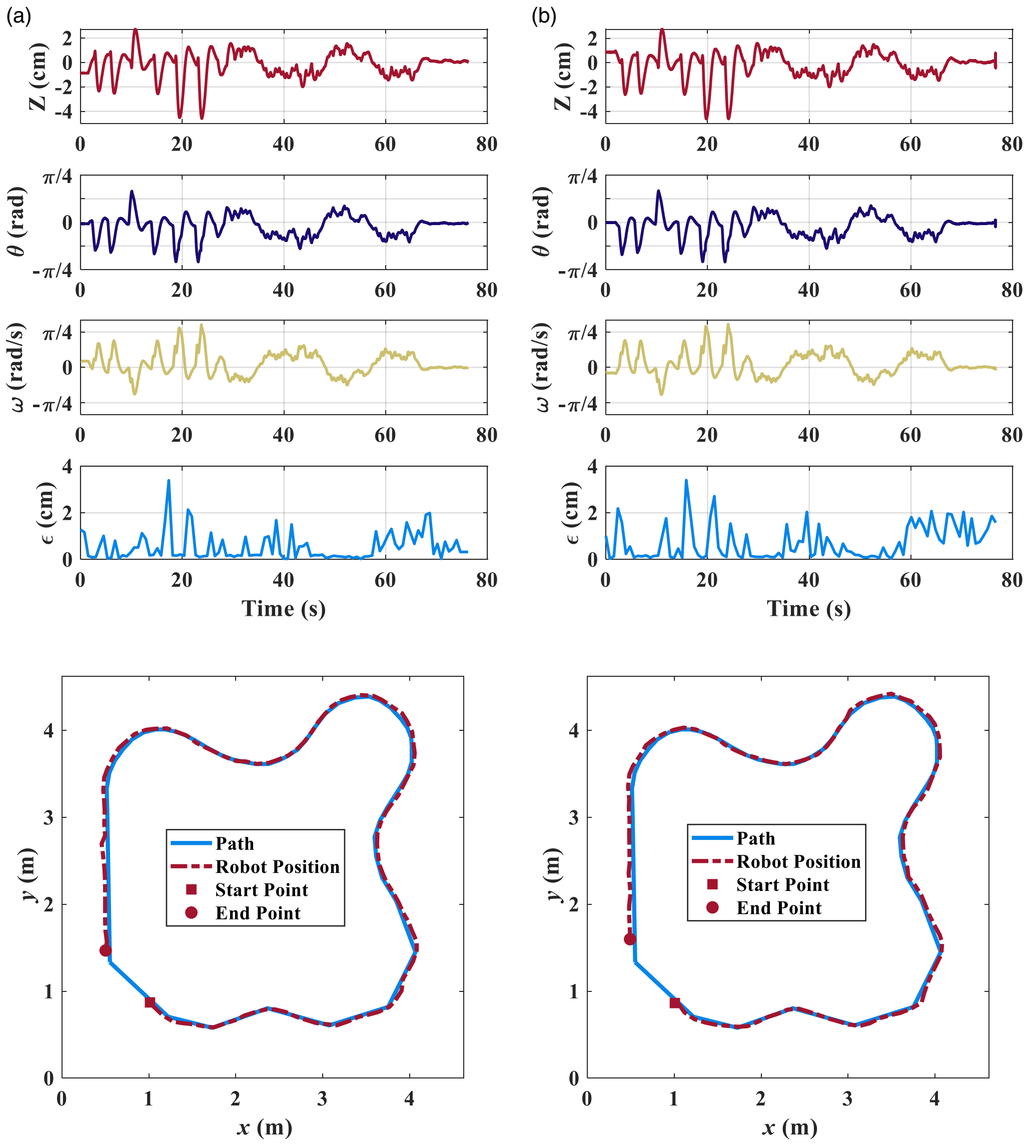


Figure 20. *Model predictive path-following performance with the linear velocity of 17.5 cm/s and using the model trained in two stages: (a) Best and (b) Worst performances based on MAE.*

that the finetuning stage improves the model accuracy, even if the input data has a different linear velocity from the captured data used for finetuning. This is more readily apparent in Figure 21.

7. Conclusion

This paper presents a new approach for model predictive visual servoing of MRs. In contrast to previous efforts, an ANN model is developed to combat challenges associated with PB models for state prediction and control synthesis, such as unknown dynamics and parameter uncertainty. The ANN model is trained

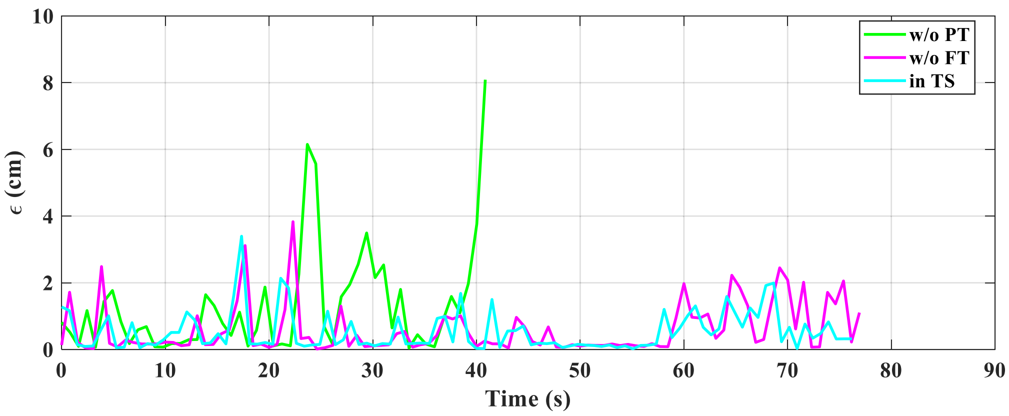


Figure 21. *Position error comparison for path-following with the linear velocity of 17.5 cm/s.*

in two-stages: pretraining and finetuning to utilize both the existing PB model and the test data collected from the actual system. The PB model is employed to generate highly diversified low-fidelity data to train the ANN model and to enhance its generalization. During the finetuning stage, the high-fidelity test data are collected from the actual robot to finetune the model weights obtained from the pretraining stage. Through finetuning, the modeling error of the PB model is mitigated and its accuracy continues to improve.

The experiments are performed to follow the track solely using the camera sensor as the feedback mechanism of the MR. By utilizing the ANN model trained in two stages, the robot is able to follow the track at 15 cm/s and 17.5 cm/s with the MAE of 0.38 cm and 0.67 cm, respectively. Furthermore, it is confirmed that when finetuning is omitted, the accuracy of path following is compromised by 0.07–0.15 cm on average. Importantly, when the model is trained only with actual data, the robot is not able to complete the course for all trials. However, the proposed two-stage training procedure improves both the generalization and accuracy of the ANN model for MPC, and the MR is able to follow the track precisely for both desired velocities.

Despite the salient performance of the proposed approach, two major limitations are identified. First, solving the optimization of the nonlinear MPC is computationally demanding. Therefore, control frequency is limited, and hence, the present control framework is not suitable for high-rate dynamic systems. Second, track information cannot be obtained precisely in the low-light environment. To further enhance the performance of visual path following, we have two specific plans for the future work. First, we will increase the control frequency by accelerating the optimization process within the MPC framework. Second, we will develop an image filter to amplify the contrast of the image, so that the path can be detected more accurately in low-light conditions.

Author contributions. Seong Hyeon Hong: designed the study and wrote the article.
Benjamin Albia and Tristan Kyzer: conducted data gathering and experiments.
Jackson Cornelius: performed data analysis.
Eric R. Mark and Asha J. Hall: provided technical feedback, guidance, and review.
Yi Wang: managed the research and wrote the article.

Financial support. This research was sponsored by the Department of Defense and US Army Combat Capabilities Development Command Army Research Laboratory (ARL) under contract number W911QX-18-P-0180.

Competing interests. The authors certify no conflict of interest for the present work.

Ethical standards. None.

References

[1] H. Xie and A. F. Lynch, “Input saturated visual servoing for unmanned aerial vehicles,” *IEEE/ASME Trans Mechatron* **22**(2), 952–960 (2016).
[2] H. Efraim, S. Arogeti, A. Shapiro and G. Weiss, “Vision based output feedback control of micro aerial vehicles in indoor environments,” *J Intell Robot Syst* **87**(1), 169–186 (2017).
[3] S. Jung, S. Cho, D. Lee, H. Lee and D. H. Shim, “A direct visual servoing-based framework for the 2016 IROS autonomous drone racing challenge,” *J Field Robot* **35**(1), 146–166 (2018).
[4] G. Allibert, M.-D. Hua, S. Krupínski and T. Hamel, “Pipeline following by visual servoing for autonomous underwater vehicles,” *Control Eng Pract* **82**, 151–160 (2019).
[5] G. Palmieri, M. Palpacelli, M. Battistelli and M. Callegari, “A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator,” *J Robot* **2012**, 1–11 (2012).
[6] F. Sadeghi, A. Toshev, E. Jang and S. Levine, “Sim2real Viewpoint Invariant Visual Servoing by Recurrent Control,” **In:** *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018) pp. 4691–4699.
[7] D. J. Agravante, G. Claudio, F. Spindler and F. Chaumette, “Visual servoing in an optimization framework for the whole-body control of humanoid robots,” *IEEE Robot Autom Lett* **2**(2), 608–615 (2016).
[8] P. Ji, A. Song, P. Xiong, P. Yi, X. Xu and H. Li, “Egocentric-vision based hand posture control system for reconnaissance robots,” *J Intell Robot Syst* **87**(3), 583–599 (2017).
[9] Y. Fang, X. Liu and X. Zhang, “Adaptive active visual servoing of nonholonomic mobile robots,” *IEEE Trans Ind Electron* **59**(1), 486–497 (2011).
[10] K. Wang, Y. Liu and L. Li, “Visual servoing trajectory tracking of nonholonomic mobile robots without direct position measurement,” *IEEE Trans Robot* **30**(4), 1026–1035 (2014).
[11] A. Hajiloo, M. Keshmiri, W. F. Xie and T. T. Wang, “Robust online model predictive control for a constrained image-based visual servoing,” *IEEE Trans Ind Electron* **63**(4), 2242–2250 (2015).
[12] G. B. Avanzini, A. M. Zanchettin and P. Rocco, “Constrained model predictive control for mobile robotic manipulators,” *Robotica* **36**(1), 19–38 (2018).
[13] J. Gao, G. Zhang, P. Wu, X. Zhao, T. Wang and W. Yan, “Model predictive visual servoing of fully-actuated underwater vehicles with a sliding mode disturbance observer,” *IEEE Access* **7**, 25516–25526 (2019).
[14] L. Pacheco and N. Luo, “Testing PID and MPC performance for mobile robot local path-following,” *Int J Adv Robot Syst* **12**(11), 155 (2015).
[15] H. Cheng and Y. Yang, “Model Predictive Control and PID for Path Following of an Unmanned Quadrotor Helicopter,” **In:** *2017 12th IEEE conference on industrial electronics and applications (ICIEA)*, (IEEE, 2017) pp. 768–773.
[16] Z. Li, C. Yang, C.-Y. Su, J. Deng and W. Zhang, “Vision-based model predictive control for steering of a nonholonomic mobile robot,” *IEEE Trans Contr Syst Technol* **24**(2), 553–564 (2015).
[17] F. Ke, Z. Li and C. Yang, “Robust tube-based predictive control for visual servoing of constrained differential-drive mobile robots,” *IEEE Trans Ind Electron* **65**(4), 3437–3446 (2017).
[18] T. T. Ribeiro and A. G. Conceição, “Nonlinear model predictive visual path following control to autonomous mobile robots,” *J Intell Robot Syst* **95**(2), 731–743 (2019).
[19] J. Plaskonka, “Different kinematic path following controllers for a wheeled mobile robot of (2, 0) type,” *J Intell Robot Syst* **77**(3-4), 481–498 (2015).
[20] S. H. Hong, J. Cornelius, Y. Wang and K. Pant, “Optimized artificial neural network model and compensator in model predictive control for anomaly mitigation,” *J Dyn Syst Meas Control* **143**(5), 051005 (2021).
[21] I. J. P. B. Franco, T. T. Ribeiro and A. G. S. Conceição, “A novel visual lane line detection system for a NMPC-based path following control scheme,” *J Intell Robot Syst* **101**(1), 1–13 (2021).
[22] J.-B. Coulaud, G. Campion, G. Bastin and M. De Wan, “Stability analysis of a vision-based control design for an autonomous mobile robot,” *IEEE Trans Robot* **22**(5), 1062–1069 (2006).
[23] P. O. M. Scokaert and D. W. Clarke, “Stabilising properties of constrained predictive control,” *IEE Proc Control Theor Appl* **141**(5), 295–304 (1994).
[24] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica* **36**(6), 789–814 (2000).
[25] D. Mayne and P. Falugi, “Generalized stabilizing conditions for model predictive control,” *J Optimiz Theory App* **169**(3), 719–734 (2016).
[26] K. Patan, “Neural network-based model predictive control: Fault tolerance and stability,” *IEEE Trans Contr Syst Technol* **23**(3), 1147–1155 (2014).
[27] S. H. Hong, J. Cornelius, Y. Wang and K. Pant, “Fault compensation by online updating of genetic algorithm-selected neural network model for model predictive control,” *SN Appl Sci* **1**(11), 1–16 (2019).
[28] C. Lu, Z.-Y. Wang, W.-L. Qin and J. Ma, “Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification,” *Signal Process* **130**, 377–388 (2017).

Cite this article: S. H. Hong, B. Albia, T. Kyzer, J. Cornelius, E. R. Mark, A. J. Hall and Y. Wang (2024). “Artificial neural network-based model predictive visual servoing for mobile robots”, Robotica **42**, 2825–2847. https://doi.org/10.1017/S0263574724001176