

# Optimal Predictive Neuro-Navigator Design for Mobile Robot Navigation with Moving Obstacles

First Mahsa Mohaghegh,<sup>1</sup>

*School of Engineering, Computing and Mathematical Sciences, Auckland University of Technology (AUT), New Zealand*

*AUT, Faculty of Design and Creative Technologies, Auckland, New Zealand*

Second Samaneh-Asadat Saeedinia<sup>2</sup>

*School of Electrical Engineering, University of Science and Technology(IUST), Tehran,Iran*  
and

Third Zahra Roozbehi<sup>3</sup>

*School of Engineering, Computing and Mathematical Sciences, Auckland University of Technology(AUT), New Zealand*

**The challenge of navigating a Mobile robot in dynamic environments has grasped significant attention in recent years. Despite the available techniques, there is still a need for efficient and reliable approaches that can address the challenges of real-time near optimal navigation and collision avoidance. This paper proposes a novel Log-concave Model Predictive Controller (MPC) algorithm that addresses these challenges by utilizing a unique formulation of cost functions and dynamic constraints, as well as a convergence criterion based on Lyapunov stability theory. The proposed approach is mapped onto a novel recurrent neural network (RNN) structure and compared with the CVXOPT optimization tool. The key contribution of this study is the combination of neural networks with model predictive control to solve optimal control problems locally near the robot, which offers several advantages, including computational efficiency and the ability to handle nonlinear and complex systems. The major findings of this study include the successful implementation and evaluation of the proposed algorithm, which outperforms other methods such as RRT, A-Star, and LQ-MPC in terms of reliability and speed. This approach has the potential to facilitate real-time**

---

<sup>1</sup> \*Senior Lecturer and Director of Women in Tech in AUT's School of Computer, Engineering, and Mathematical Sciences. e-mail: mahsa.mohaghegh@aut.ac.nz.

<sup>2</sup> Ph.D. Candidate, Control Engineering Department, IUST. e-mail: s\_saeedinia@elec.iust.ac.ir.

<sup>3</sup> Ph.D, Candidate, Computer and Mathematical department, AUT. e-mail: zahra.roozbehi@aut.ac.nz.

**navigation of mobile robots in dynamic environments and ensure a feasible solution for the proposed constrained-optimization problem.**

**Keywords: Navigation, optimization, MPC, Stability, Neural Network, Dynamic Environment.**

### **Nomenclature**

(Nomenclature entries should have the units identified)

$M$ =Mass

$r$ =Wheels' radius

$L$ =Distance to wheels

$I_z$ =Inertia of the Robot

$J$ =Inertia of the Wheels

$d$  = maximum possible movement

$d_\theta$  = maximum possible rotation angle

$V_{max}$  = maximum possible velocity for the robot

$\omega_{max}$  = maximum possible angular velocity

$\hat{\gamma}^T$  = the estimated model parameter by Kalman Filter

$\hat{\phi}_w(k)$  =the estimated robot angle

$\hat{Y}_{k+1}$  =the one-step-ahead prediction of  $Y_k$

$Y_f$  =the (x-y) position vector of the target destination

$Y_d(k)$  =the generated path at time step  $k$

$S_{ij}$  = the Boolean number 0 or 1 of the screen state in  $(i,j)$ <sub>th</sub> matrix index

$I_n$  = the input synaptic current on the neuron n (mA)

$\rho_n$  = the adaption potential (mV)

$V_n$  = the neuron membrane function (mV)

*Subscripts*

$q = [x_w \quad y_w \quad \varphi_w]^T$  = position of the robot in the global coordinate  $(X_w, Y_w)$  and angle of the robot vector

$\dot{q}$  = robot velocity

$\ddot{q}$  = robot acceleration

$\tau \in R^3$  = torque input

$M(q) \in R^3$  = inertial matrix

$C(q, \dot{q})q$  = Coriolis

$B(q) \in R^{3 \times 3}$  = centrifugal force vector

$x_{oi(t-1)}$  = the  $i$ th obstacle position

$v_{oi(t-1)}$  = the  $i$ th obstacle velocity

$J(t)$  = the cost of the MPC at time t

$Y_k = Y(t) = [q \quad \dot{q}]^T$  = sample of position and velocity vector of the robot dynamic at time  $t = k\Delta t$

$U_k$  = the control law, i.e., the generated path

$\hat{Y}_{k+1}$  = the one-step-ahead prediction of  $Y_k$

$F$  = discrete-time error-feedback gain

## I.Introduction

NEURAL networks and model predictive control (MPC) are two powerful techniques in robotics, especially in the path planning of mobile robots. Neural networks are computational algorithms, mimicking human brain functions, while MPC is a control technique that predicts the future behavior of a system based on a mathematical model. Both approaches are popular methods for planning the path of a mobile robot with the aim of optimizing the path and avoiding obstacles. However, both of them have different strengths and weaknesses. One of the highlighted capabilities of neural networks, in the field of path planning, is their ability to determine the optimal path by training on a set of data from previous successful paths and generalizing its pattern to a new environment based on its input, such as sensory data from the robot's sensors. On the other hand, MPC uses a mathematical model of the robot's dynamics and constraints, such as speed limits and obstacle avoidance, to predict the optimal path. This matter improves the performance of MPCs in comparison to neural networks in more complex dynamics and constraints. Although, achieving this requires modeling and accurate calculations [3].

Depending on the specific requirements, engineers may find both neural networks and MPC to be highly useful. In fact, a combination of these methods can provide robotics experts with even greater capabilities to enhance the efficacy and reliability of path planning [4,5].

In the realm of path planning for mobile robots, it is imperative to not overlook the significance of model-free techniques, even while capitalizing on model-based approaches such as MPCs. Although model-free control methods, exemplified by Q-learning and its derivatives, have been extensively studied in mobile robot navigation [6], their computational intensity may render them unsuitable for systems with stringent constraints or safety prerequisites. In contrast, MPC provides a systematic framework for addressing constraints and objectives, making it a versatile tool for navigating mobile robots in intricate and dynamic environments [7]. Moreover, MPC can be

integrated with learning-based techniques, such as neural networks, to enhance prediction model accuracy and attain high-performance path tracking [8]. Additionally, incorporating exploration-exploitation-based adaptive principles into policy-based and value-based methods constitutes an additional strategy for model-free management [9]. Nonetheless, these techniques can be exceedingly computational and may necessitate vast amounts of data for training.

The literature review reveals that most of the methods in recent research have utilized neural networks to predict the environment for robot navigation [10][11][12][13][14]. However, in this study, we propose a novel approach that combines neural networks with model predictive control to solve optimal control problems locally near the robotic system. Specifically, we utilize neural networks as a function approximator to estimate the system dynamics and cost function, which are then used by the model predictive control algorithm to solve the optimal control problem.

In a recent study a learning-based nonlinear model predictive control (LB-NMPC) algorithm is presented to improve vision-based mobile robot path-tracking in challenging outdoor environments [15]. This approach combines learning-based techniques with MPC to enhance the accuracy of the prediction model and achieve high-performance path tracking. Saedinia et al proposed a new navigation control scheme for a three-wheel omnidirectional robot that takes into account the dynamic constraints to achieve stable and efficient navigation, combining multi-modal MPC and Q-learning to improve the accuracy of the prediction model and handle system constraints in the control process[16]. Additionally, there have been recent breakthroughs in deep reinforcement learning-based MPC controllers for mobile robots that have the capability of moving omnidirectionally. These controllers utilize neural networks to estimate the robot's future state and generate optimal control inputs using MPC. Experimental validation on real robots has demonstrated improved performance compared to traditional MPC controllers [11][17]. As

another work, we can refer to a real-time neural MPC has been proposed for quadrotors and agile robotic platforms, which makes use of neural networks as dynamic model to optimize control inputs in real-time [18].

While most of the existing literature focuses on using neural networks to predict the environment, our work emphasizes utilizing neural networks to solve optimal solutions locally near the robot. This approach confers advantages such as increased accuracy and efficacy in the navigation of robotic systems.

Other recent research has also been conducted regarding path planning and navigation for mobile robots. For example, a study proposed an improved deep reinforcement learning algorithm for path planning in large-scale dynamic environments, considering the avoidance of dynamic obstacles [19]. Another study explored the use of object detection using CNNs for model-predictive control of omnidirectional mobile robots in logistic environments [20]. These studies highlight the ongoing research efforts in developing advanced techniques for mobile robot navigation.

On the other hand, the process of identifying the most efficient trajectory for a mobile robot is inherently characterized by non-linearity and occasionally non-convexity, thereby restricting the applicability of standard convex optimization tools in the context of MPCs applications. In this regard, a MPC that is based on neural networks (Neuro-MPC) can prove to be particularly advantageous when navigating complex environments in which dynamic obstacles cannot be readily modeled using convex optimization tools [21].

Although the combined method mentioned above offers numerous advantages, it is crucial not to overlook the importance of system stability. This issue cannot be neglected and must be taken into consideration. The stability of Neuro\_MPC is a critical issue in path planning because instability can lead to unexpected behavior of the autonomous system, which may cause accidents or damage

to the environment [22,23]. Several methods have been proposed to guarantee the stability of Neuro\_MPC, such as Lyapunov-based stability analysis and robust control techniques. These methods ensure that the MPC controller generates control actions that steer the system towards a safe and stable trajectory. Thus, ensuring the stability of Neuro\_MPC is essential to prevent unexpected behavior and ensure the safety of both the autonomous system and its environment.

In order to tackle the challenges at hand, we have developed a novel heuristic log-concave MPC navigation problem that takes into account dynamic and kinematic constraints, and guarantees convergence based on Lyapunov stability theory. To solve this problem, we have designed a new structure for a recurrent neural network that is inspired by the NN constrained formulation presented in [24], and utilizes a screen searching based approach.

Our exploration revolves around a three-wheeled omnidirectional robot that uses a differential drive scheme to control its motion. This distinctive robot architecture confers greater dexterity and accuracy in traversing intricate environments, particularly when confronted with mobile obstacles. Our work employs the built-in tracking controller, which is based on the controller devised by Peñaloza-Mejía et al [25]. This controller is a PID controller, effectively steers the robot's locomotion while accommodating velocity constraints. Nevertheless, in order to ensure compatibility with the environment and surmount limitations associated with mobile obstacles, we have implemented MPC to govern the entire system, in conjunction with the integrated tracking controller. In our research, we have employed MPC to optimize the robot's movement and positioning in real-time, taking into account the existence of mobile obstacles within the surroundings. We have furnished further explication on the functionality of the MPC controller in the following sections.

This paper is organized into several sections to provide a comprehensive overview of our approach. In Section 2, we introduce the Materials and Methods, which includes a detailed description of the navigation strategy, a new definition of the log-concave navigation MPC problem, and proposing a novel Recurrent Neural Network structure to solve the complex nonlinear problem. In Section 3, we compare the results of our proposed navigation strategy with those of well-known classical and reactive methods, namely, RRT, A-star and LQ-MPC methods. Finally, in Section 4, we discuss the simulation results and draw conclusions.

## II. Materials and Methods

The general navigation system for robots involves various factors such as the positions of the target and the robot, the environment's characteristics and arrangements, and the constraints and objectives. (See Figure 1 for a visual representation of these interactions.)

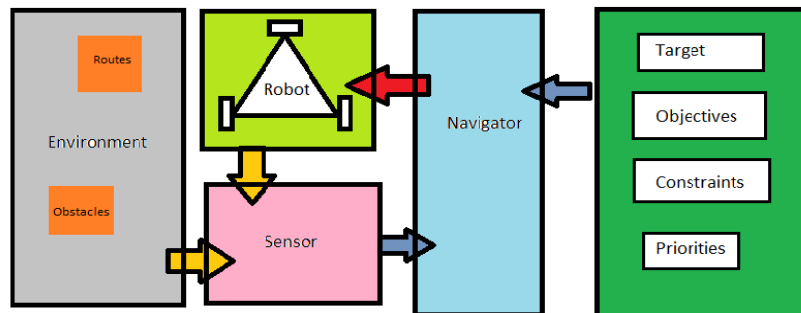


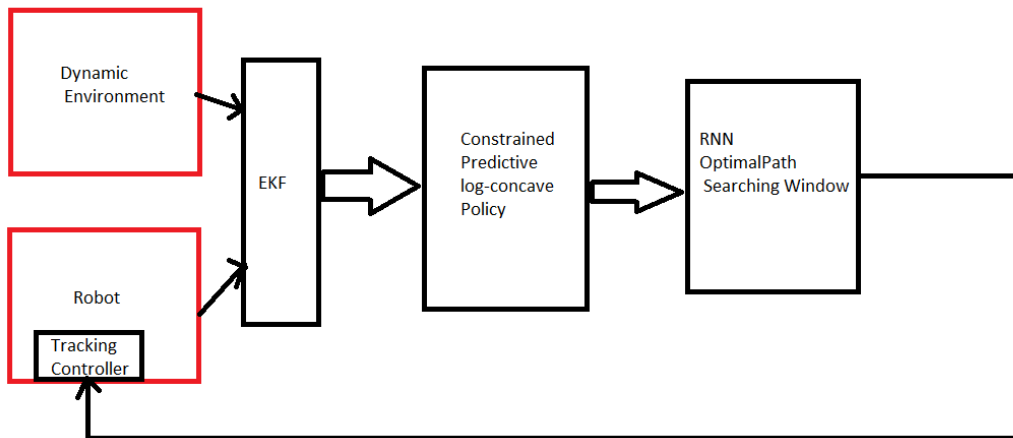
Figure 1 A general navigation system interactions

When it comes to global path planning, having knowledge of the environment and its characteristics is crucial in developing optimal strategies. This is why the environment is

monitored globally in optimal approaches to reduce distance costs. However, local information is sufficient in local navigation methods to reduce the cost of time response for real-time implementation. Unfortunately, this may not always consider distance costs appropriately, which is why a trade-off between local and global advantages is necessary.

To address this issue, researchers have developed heuristic algorithms that benefit both strategies. Our paper proposes a combination of local predictive path planning methods with global information about the environment. (See Figure 2 for the internal structure of our proposed path planning system.).

The term "local information" refers to data collected around the robot's current position, while "global environment data" refers to information about the robot's overall environment. For predicting the behavior of moving obstacles, both local and global information are used. The local data predicts immediate behavior, while global environment data predicts future behavior. This tactic reduces risk, minimizes computational expenses, and presents a complete comprehension of the system.



**Figure 2 Internal Structure of the proposed path planning system**

Our approach involves processing the received global image to label obstacle positions as 1 and the rest as 0. We then estimate the locations of dynamic obstacles in future time intervals to avoid collisions using multiple EKF estimators. The constrained predictive log-concave policy block is the center of rewarding the policy, designed to maximize the award when the path is optimal and minimize it in the presence of a high risk of collision.

We also designed an Recurrent Neural Network ( RNN ) block to receive updated information on the position classified data and evaluate the next state policy cost of the robot's neighborhood, considering a conservative collision avoidance constraint compatible with finite state prediction steps. By combining local and global information, our proposed path planning system offers a more comprehensive and effective approach to robot navigation.

In order to assess the effectiveness of our navigation strategy, we utilized Python 3.10 to simulate three randomly moving obstacles, as well as a real three-wheeled omnidirectional mobile robot (OMR) dynamic model with a built-in tracking controller introduced by [25]. Our goal was to create a realistic indoor environment and robot dynamic behavior simulation, allowing us to thoroughly investigate the proposed navigator performance. To achieve this, we utilized the robot dynamic model and the Kalman Filter updating equations [26] as the real robot dynamics indicator and the observer component of the navigation system, respectively.

By and large, we used the EKF algorithm for obstacle tracking and estimation. The EKF algorithm is used to estimate the position and velocity of obstacles and the robot in the environment, which are utilized as inputs to the MPC algorithm to generate control actions for the robot. The EKF algorithm considered the position and velocity of the obstacles and predicted their next position by predicting one step ahead.

The MPC algorithm was responsible for generating control actions that steer the robot towards the target while avoiding collisions with obstacles. The MPC algorithm used the estimated positions and velocities of obstacles obtained from the EKF algorithm as inputs to generate control actions for the robot. The cost function used in the MPC algorithm was designed to balance the tradeoff between achieving the desired goal and avoiding obstacles while minimizing energy consumption. The RNN structure is designed as a local optimization solver for the nonlinear MPC problem. Specifically, the RNN is used to estimate the system dynamics and cost function, which are then used as inputs to the MPC algorithm to solve the optimal control problem. The RNN is trained to minimize the cost function in the Lagrangian form by adjusting the input control to the system at each time step, such that the constraints are satisfied and the state of the system approaches the desired reference trajectory.

#### **A. Environment Simulation and Data Collection**

This section provides a comprehensive description of the preprocessing and analysis of the sensor data utilized in our simulation study. In consideration of the EKF converging process and RNN learning and training time samples, we utilized 70 sampling epochs as the training dataset, using data recursively, and the remaining epochs for testing.

The sensor data is simulated as flag data, which detects obstacles from a distance of 0.4 meters. If obstacles are located in front of the sensor, the flag is set to 1, otherwise, it is set to 0.

To generate the input data for the learning algorithm, we used a camera mounted on the ceiling of the room to detect the position and color of obstacles in the environment. The position of the robot and obstacles were labeled as global positions, and the obstacles were represented as binary values, with 1 indicating the presence of an obstacle and 0 indicating no obstacle at that position. We also

added normal random noise to the positions of the robot and obstacles to simulate the measuring noise in the sensor data.

To analyze the sensor data, we first pre-processed the data to remove noise and outliers and normalized the data to ensure that the learning algorithm could effectively learn from the data. We also used data augmentation techniques to increase the size of the dataset and improve the generalization ability of the learning algorithm.

Next, we used the Extended Kalman Filter (EKF) algorithm to estimate the position and velocity of obstacles in the environment and the robot. The EKF algorithm considered the position and velocity of the obstacles and estimated their next position by predicting one step ahead. Obstacles were designed with random velocity and shape to simulate a realistic environment. If any new obstacles were detected, a new EKF was started to estimate its position. If the predicted location and sensor data were not compatible, the sensor data was given priority, and the robot changed its direction by 45 degrees in each step while the sensor flag was set to zero to show the lack of obstacles nearby. Finally, we used the pre-processed sensor data as input to the learning algorithm and trained the algorithm to predict the position and velocity of obstacles in the environment.

## **B. Mobile Robot Model**

To simulate realistic dynamic behavior of a mobile robot, a real three-wheeled OMR Dynamic model parameters and internal tracking control law, given by [25] are utilized. The dynamic model of OMR is described as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = B(q)\tau \quad (1)$$

The inertial matrix, the Coriolis and centrifugal force vector, and the input matrix, defined as follows:

$$\begin{aligned}
M(q) &= \begin{bmatrix} m + \frac{3J}{2r^2} & 0 & 0 \\ 0 & m + \frac{3J}{2r^2} & 0 \\ 0 & 0 & I_z + \frac{3JL^2}{r^2} \end{bmatrix}, \\
C(q, \dot{q}) &= \begin{bmatrix} 0 & \frac{3J}{2r^2} \dot{\varphi}_w & 0 \\ -\frac{3J}{2r^2} \dot{\varphi}_w & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\
B(q) &= \begin{bmatrix} -\frac{1}{2r}(\sin(\varphi_w) - \sqrt{3} \cos(\varphi_w)) & \frac{1}{r} \sin(\varphi_w) & -\frac{1}{2r}(\sin(\varphi_w) + \sqrt{3} \cos(\varphi_w)) \\ \frac{1}{2r}(\sin(\varphi_w) + \sqrt{3} \cos(\varphi_w)) & -\frac{1}{r} \cos(\varphi_w) & \frac{1}{2r}(\sin(\varphi_w) - \sqrt{3} \cos(\varphi_w)) \\ \frac{L}{r} & \frac{L}{r} & \frac{L}{r} \end{bmatrix}
\end{aligned} \tag{2}$$

The parameters of the model are stated in Table 1, and The tracking control law is described in [25].

**TABLE 1**

*The Real Robot Model Parameters, taken by [25]*

Parameters	Units	Quantities
m	kg	11.83
r	m	0.0625
L	m	0.287
$I_z$	kg.m <sup>2</sup>	0.0127
J	kg.m <sup>2</sup>	5.82E-4

### C. Prediction of the Future States

This paper employs  $m+1$  Kalman filters concurrently to forecast the future states of  $m$  obstacles, as well as the robot's position and velocity, based on the mean square error of the estimation. The Kalman Filter Parameter estimation laws are derived from [26], and the regressor for the obstacles is defined as follows:

$$X_{oi}(k) = [x_{oi(k-1)} \ v_{oi(k-1)}] \quad (3)$$

The estimated future states of the robot are utilized to predict the future screen of the environment. Moreover, to predict future states of the robot, a Kalman filter is implemented to estimate the parameters of an Auto-Regressive Exogenous Model, which is considered as the dynamic constraint of the robot in the proposed MPC navigator.

### D. The log-Convex Navigation MPC definition

This section addresses a common challenge in obstacle avoidance, which involves dynamic and kinematic constraints within a log-convex MPC framework. In order to tackle this issue, we have identified several key rules that are typically applied in navigation problems. These rules have been translated into mathematical expressions that can be incorporated into the MPC cost function and constraints.

Rule one emphasizes the importance of taking the shortest route possible, while rule two highlights the value of maintaining a safe distance from obstacles. Rule three prioritizes the use of sensor information over image processing for obstacle detection, and rule four stresses the need for accurate tracing at all times. Rule five reminds us to adhere to the maximum allowable speed, and rule six emphasizes the importance of ensuring system stability.

To incorporate these rules into our MPC framework, we have developed a set of linear constraints and kernel functions that allow us to evaluate their relative importance on a scale of zero to one.

Rules one to five are incorporated into a constrained MPC structure, which is defined as follows:

$$\min_{\varepsilon, g} J_{\infty} = \sum_{n=k}^{\infty} \underbrace{\int_{n\Delta t}^{(n+1)\Delta t} J(t)dt}_{J_n} = \sum_{n=k}^{k+N} J_n + \sum_{n=k+N+1}^{\infty} J_n = \sum_{n=k}^{k+N} J_n + I((N+1)\Delta t) \quad (4-1)$$

*S.t.*

$$\begin{aligned} \hat{Y}_{k+1} &= \hat{\gamma}^T \begin{bmatrix} Y_k \\ U_k \end{bmatrix}, \\ |P_k - O_{ik}| &> 2(2r), \\ \begin{cases} O_{ik} = [I_{3 \times 3} & 0_{3 \times 3}] [X_{oi}(k+1) \cdots X_{oi}(k+N)]_{6 \times N} \\ Y_p = [\hat{Y}_{k+1} \cdots \hat{Y}_{k+N}]_{6 \times N} \\ P_k = [I_{3 \times 3} & 0_{3 \times 3}] Y_p \end{cases} & \quad (4-2) \\ |\hat{\varphi}_w(k)| &\geq Flag \times \left( \frac{\pi}{4} \right) \\ \left[ \begin{array}{c} \bar{0}_{3 \times 3} \quad \bar{1}_{3 \times 3} \\ \bar{1}_{3 \times 3} \end{array} \right] Y_{k_{6 \times 1}} &\leq \begin{bmatrix} V_{\max} \cdot \cos(\varphi_w(k)) \\ V_{\max} \cdot \sin(\varphi_w(k)) \\ \omega_{\max} \end{bmatrix} \end{aligned}$$

where  $J(t)$  is defined as follows:

$$\begin{aligned} J(t) &= 1 - \exp\left(-0.5(Y_{k\Delta t} - Y_f)^T \Lambda^{-1}(Y_{k\Delta t} - Y_f)\right) \\ &+ \exp\left(\left(-0.5(Y_{k\Delta t} - X_{oi})^T \Lambda^{-1}(Y_{k\Delta t} - X_{oi})\right)\right) \\ &+ 1 - \exp\left(-0.5(U_{k\Delta t}^T) \Lambda^{-1}(U_{k\Delta t})\right) \end{aligned} \quad (5)$$

where  $J(t)$  is defined as follows:

$$\begin{aligned}
J(t) &= 1 - \exp\left(-0.5(Y_{k\Delta t} - Y_f)^T \Lambda^{-1}(Y_{k\Delta t} - Y_f)\right) \\
&+ \exp\left(\left(-0.5(Y_{k\Delta t} - X_{oi})^T \Lambda^{-1}(Y_{k\Delta t} - X_{oi})\right)\right) \\
&+ 1 - \exp\left(-0.5(U_{k\Delta t}^T) \Lambda^{-1}(U_{k\Delta t})\right)
\end{aligned} \tag{5}$$

where  $Y(t) = Y(k\Delta t) = Y_k$  is discrete-time sampling. In which  $\Lambda^{-1}$  is defined based on the constraint of the robot dynamic and environment limitations.

$$\Lambda^{-1} = \begin{bmatrix} \begin{bmatrix} \frac{1}{d_x} & 0 & 0 \\ 0 & \frac{1}{d_y} & 0 \\ 0 & 0 & \frac{1}{d_\theta} \end{bmatrix} & & \\ & 0_{3 \times 3} & \\ & & \begin{bmatrix} \frac{1}{V_{\max}} & 0 & 0 \\ 0 & \frac{1}{V_{\max}} & 0 \\ 0 & 0 & \frac{1}{\omega_{\max}} \end{bmatrix} \end{bmatrix} \tag{6}$$

Where  $d$  indicates maximum possible movement in the Environment toward the (x-y) axis. In addition, the maximum possible rotation angle is shown by  $d_\theta$ .  $V_{\max}$  denotes the maximum possible velocity for the robot, according to the environment and robot dynamic.  $\omega_{\max}$  is the maximum possible angular velocity.  $Y_k = [q \ \dot{q}]^T$  indicates position and velocity vector of the robot dynamic at time step  $k$ .  $U_k$  is the control law, i.e., the generated path.  $\hat{\gamma}^T$  is the estimated model parameter by KF.  $\hat{\phi}_w(k)$  is the estimated robot angle and  $\hat{Y}_{k+1}$  is the one-step-ahead prediction of  $Y_k$ . *Flag* is 1 if the built-in sensors detect obstacles around the robot, otherwise, it is zero, to deal with uncertainties and possible faults of environment predictions. This item is considered to avoid the robot collision with obstacles, in the KF failure situations.

The cost function used in this study is a key component of the model predictive control (MPC) algorithm that controls the navigation of the robot. It is designed to balance the tradeoff between achieving the desired goal (i.e., reaching the target point) and avoiding obstacles while minimizing the energy consumption of the robot. The position cost function is given by the term

$1 - \exp\left(-0.5(Y_{k\Delta t} - Y_f)^T \Lambda^{-1}(Y_{k\Delta t} - Y_f)\right)$ . This term penalizes the distance between the robot's current position  $Y_{k\Delta t}$  and the target point  $Y_f$ . The collision cost function is given by  $\exp\left(-0.5(Y_{k\Delta t} - X_{oi})^T \Lambda^{-1}(Y_{k\Delta t} - X_{oi})\right)$ , and the energy cost function is given by  $1 - \exp\left(-0.5(U_{k\Delta t}^T) \Lambda^{-1}(U_{k\Delta t})\right)$ , here is the command requests to the robot to change its position and orientations. Constraints of the navigation problem, given in equation (4-2) are break down to system dynamic constraint, local and global collision avoidance, using sensor data, and kinematic constraint.

This paper utilizes ARMAX model to estimate velocity and position of the robot as follows:

$$\begin{aligned}
 Y_{k+1} &= aY_k + bU_k + e_k, \quad e_k = N(0,1) \\
 E\{Y_{k+1}\} &= \underset{\gamma}{\begin{bmatrix} a \\ b \end{bmatrix}}^T \underset{\varphi}{\begin{bmatrix} Y_k \\ U_k \end{bmatrix}} = \gamma\varphi
 \end{aligned} \tag{7}$$

Where  $Y_k$  denotes the position and velocity of the robot at  $k$ th sample time, and  $e_k$  is a disturbance with normal distribution.  $\varphi$  is the input regressor, and the next states of the robot are calculated as below:

$$C1: \hat{Y}_{k+1} = \gamma(:,1)^T Y_k + \gamma(:,2)^T U_k \tag{8}$$

Considering constant control input in each sample time, we define N-horizon prediction of states vector as follows:

$$\begin{aligned}
 Y_p &= [Y_{k+1} \quad \cdots \quad Y_{k+N}]_{6 \times N} \\
 P_k &= [I_{3 \times 3} \quad 0_{3 \times 3}] Y_p
 \end{aligned} \tag{9}$$

Where  $P_k$  is the prediction of N next positions of the robot.

Suppose that  $D_{oi} = [x_{oi}, v_{oid}]^T$  denotes to the  $i$ th obstacle's position and velocity vector in the environment at the global coordination. Obstacles' states predications are calculated below:

$$\begin{aligned} X_{oi}(k+1) &= \begin{bmatrix} I_{3 \times 3} & \Delta t \times I_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} X_{oi}(k) \\ O_{ik} &= [I_{3 \times 3} \quad 0_{3 \times 3}] [X_{oi}(k+1) \quad \cdots \quad X_{oi}(k+N)]_{6 \times N} \end{aligned} \quad (10)$$

To show the constraint of the robot and obstacles collision avoidance, we define the below criteria:

$$C2: |P_k - O_{ik}| > \Omega, \quad \Omega > 0 \quad (11)$$

Where  $\Omega$  is the conservative distance parameter, it should be determined based on the maximum estimated velocity of the robot, and obstacles to handling prediction's uncertainties. Where  $\Omega$  is the conservative distance parameter, it should be determined based on the maximum estimated velocity of the robot, and obstacles to handling prediction's uncertainties, here is considered  $2(2r)$ .

Due to possible prediction faults and uncertainties, the importance of checking robot obstacle detection sensors increases, so we consider a Flag to detect an obstacle. Flag=1 indicates an obstacle is nearby the robot, and Flag=0 reveals the lack of obstacle around the robot.

$$C3: |Y(3)| \geq Flag \times \left( \frac{\pi}{4} \right) \quad (12)$$

This inequality can also be considered as a fitness neighborhood searching function, instead of intending it as another constraint of the MPC problem. This substitution helps to preserve convexity of the problem, when convex optimization solver is utilized.

Kinematic constraints should be regarded in the navigation strategy. This study considers a mere velocity constraint.

$$C5: |[0 \ 1]Y| \leq \begin{bmatrix} V_{\max} \cdot \cos(\theta) \\ V_{\max} \cdot \sin(\theta) \\ \omega_{\max} \end{bmatrix} \quad (13)$$

### E. Designing control law with stability guarantee

This section provides an error feedback control law,  $U(t) = F(Y(t) - Y_f)$  which is able to ensure the stability of MPC problem, (4). To this aim, consider the candidate Lyapunov function as follows:

$$J_{\infty} = \int_t^{\infty} \left( \begin{array}{l} 1 - \exp\left(-0.5(Y(t) - Y_f)^T \Lambda^{-1}(Y(t) - Y_f)\right) \\ + \exp\left(\left(-0.5(Y(t) - X_{oi})^T \Lambda^{-1}(Y(t) - X_{oi})\right)\right) \\ + 1 - \exp\left(-0.5(Y(t) - Y_f)^T F^T \Lambda^{-1} F(Y(t) - Y_f)\right) \end{array} \right) dt \quad (14)$$

$F$  is discrete-time error-feedback gain. In order to ensure stability in a navigation system, it is necessary to determine the criterion for stability guarantee. This can be achieved by calculating the derivative of the defined Lyapunov function, which should be negative. By simplifying the Lyapunov inequality constraint of stability, it is possible to determine whether the navigation system is stable.

Stability in navigation is crucial for ensuring that a robot can reach its intended destination without any collisions. To simplify the stability criterion, a linear approximation of the cost function is used, taking into account the limited arguments of the kernel functions, which are restricted to values of 0 and 1. By following these steps, it is possible to ensure that the navigation system is stable and reliable, allowing for safe and efficient navigation.

$$\begin{aligned}
\frac{\partial J_{\infty}(t)}{\partial t} &= H + H^T \\
H &= 0.5(\dot{Y})^T \left( -\Lambda^{-1}(X_{oi} - Y_f) + F^T \Lambda^{-1} F (Y - Y_f) \right) \\
\frac{\partial J_{\infty}(t)}{\partial t} &\leq \int_t^{\infty} \underbrace{(H(t) + H^T(t)) \partial t}_{I(t)} < 1 - \underbrace{\left( -Y^T \Lambda^{-1}(X_{oi} - Y_f) + (Y^T) F^T \Lambda^{-1} F (Y - Y_f) \right)}_{I(t)} < 0
\end{aligned} \tag{15}$$

$$\begin{aligned}
I(t) &= 1 - \begin{bmatrix} -Y(t) \\ Y(t) \end{bmatrix}^T \begin{bmatrix} \Lambda^{-1} & 0 \\ 0 & F^T \Lambda^{-1} F \end{bmatrix} \begin{bmatrix} X_{oi} - Y_f \\ Y - Y_f \end{bmatrix} < 0, \\
\Lambda^{-1} &> 0, F^T \Lambda^{-1} F > 0
\end{aligned}$$

#### F. Recurrent Neural network design to seek the near optimal path

This section is assigned to map the newly defined optimization problem into the searching-based recurrent Neural Network structure. To this aim, consider a general Lagrangian constrained optimization problem as follows [17]:

$$\begin{aligned}
L(x_1, x_2, \dots, x_n, \lambda_1, \dots, \lambda_m) &= f(x_1, x_2, \dots, x_n) + \sum_{i=1}^m \lambda_i r_i \\
\frac{\partial L}{\partial x_i} &= 0, \quad i = 1, \dots, n \\
\frac{\partial L}{\partial \lambda_j} &= 0, \quad j = 1, \dots, m
\end{aligned} \tag{16}$$

A new and innovative RNN structure has been proposed to find a near-optimal solution in possible movement directions. This structure is made up of newly designed Gaussian spiking neurons that construct the cost function criteria concept in their membrane function. The synaptic neurons are defined in such a way that the network can search for the best position.

The main idea behind this proposed structure is to solve the Lagrangian problem by considering synaptic current as a term that can indicate the weighted constraints. This synaptic current is then inputted to the synaptic neurons.

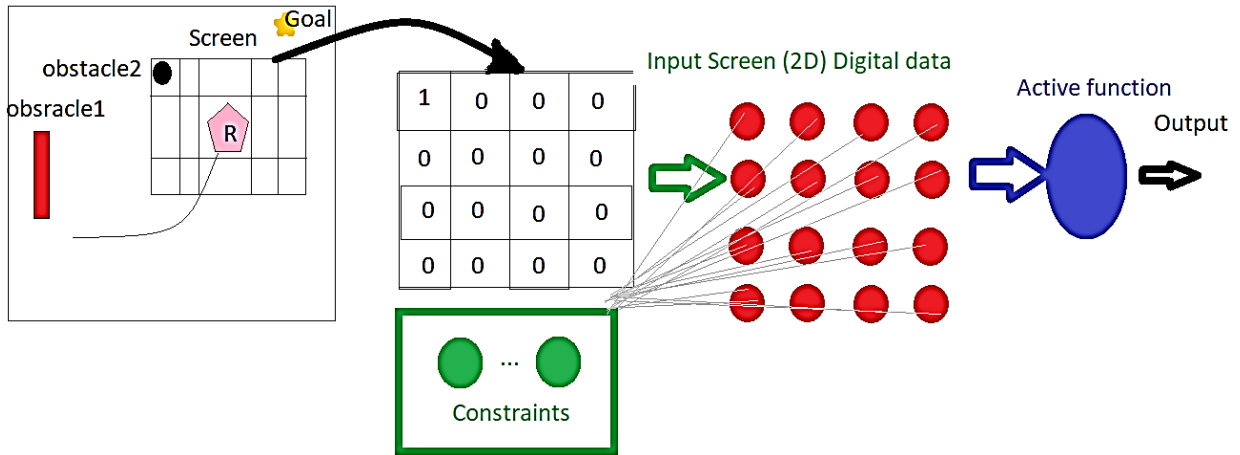


Figure 3: Proposed Neural Navigator Structure

Figure 3 demonstrates how the searching screen window is selected based on the location of the robot and the prediction horizon. Additionally, the figure indicates the constraints inputted to the reservoir recurrent neural network beside the selected window. The amount of four-sided shapes was based on the desired forecast range, with a single quadrangle corresponding to a forecast one step ahead, two quadrangles corresponding to a forecast two steps ahead, and so on. This facilitated the capture of pertinent data pertaining to the automaton's immediate surroundings at various prediction intervals, which was employed as input to our RNN-based optimization methodology. The dimensions of the display were determined in accordance with the number of quadrangles required to capture the sought-after prediction horizon.

After analyzing the low-risk position corresponding to the minimum membrane potential, the activation function generates the desired path according to the calculated feedback gain

corresponding to the low-risk possible location. In other words, the minimum  $V_n$  indices indicate the path error feedback indices, and consequently, the desired path is calculated.

The proposed network algorithm is defined in Eq.17. This new RNN structure has the potential to revolutionize the way we approach movement directions and find optimal solutions.

$$\begin{aligned}
V_n &= \sum_{i=x_r-1}^{x_r+1} \sum_{j=y_r-1}^{y_r+1} s_{ij} e^{-(x_r-i-1)^2-(y_r-j-1)^2} + (2s_{ij} - 1) e^{-(x_f-i)^2-(y_f-j)^2} + I_n \\
\rho_n &= \sum_{k=1}^m W_{nk} r_k + \theta_n \\
I_n &= \tanh(\rho_n) \\
r_1 &= \Delta x_r - V_{x_{\max}} \cdot \Delta t \\
r_2 &= \Delta y_r - V_{y_{\max}} \cdot \Delta t \\
r_3 &= x_r - \gamma \phi[1] \\
r_4 &= y_r - \gamma \phi[2] \\
r_6 &= \tan^{-1}(y/x) - Flag \times \frac{\pi}{4} \\
r_7 &= 1 - \left( -Y^T \overbrace{\Lambda^{-1}(X_{oi} - Y_f)}^Z + (Y^T) \overbrace{F^T \Lambda^{-1} F}^\Phi (Y - Y_f) \right)
\end{aligned} \tag{17}$$

where  $S_{ij}$  is a Boolean value that represents the state of a grid cell in the global screen. The screen is divided into a grid based on the scale of the robot's dimensions, treating the robot as a single point in the encoded matrix of the environment. The values of  $x_r$  and  $y_r$  represent the robot's position in the (x-y) coordinate system. The variable  $V_n$  in the equation represents the cost of each state in the neighborhood of the robot, as well as the cost of searching around it, considering one-step movements and calculating the cost of the distance from the target.  $S_{ij}$  is used to increase or decrease the cost if an obstacle is present or absent at that position. The first term of  $V_n$  searches for low-risk positions around the robot, while the next term evaluates the risk of one-step movements.

This searching structure is designed with the aim of reducing map size dependency and consequently its computational cost.  $I_n$  is the input synaptic current on the neuron  $n$  and  $Nv$  is the neuron membrane function, which indicates the proposed cost function and 1 step prediction of moving environment screen. The combination of the searching-based idea with the reservoir

structure can reduce sensitivity of the proposed navigator to uncertainties and improve its robustness.

The input neurons and the size of proposed network can be considered constant and relevant to the steps of prediction horizon. Synaptic weights ( $W_{ij}$ ) are updated based on minimizing  $V_n$ , as an indicator of the  $(i, j)$ th Lagrangian function of the proposed algorithm.

$$\begin{aligned}
\Delta W_{nk} &= -\eta \frac{\partial V_n}{\partial W_{nk}} = -\eta(1 - \rho_n^2)r_k \\
\Delta \theta_n &= -\eta(1 - \rho_n^2) \\
\Delta F &= -\mu \frac{\partial r_7}{\partial \Phi} = \mu \Lambda^{\frac{1}{2}}(YY^T - YY_f^T)^{\frac{1}{2}}
\end{aligned} \tag{18}$$

The desired error feedback gain is calculated based on the minimum- $V_n$  neuron location. Consequently, the desired path is calculated according to the following active function:

$$Y_d = \begin{cases} Y_{robot} + F(Y_f - Y_{robot}) & |Y_f - Y_d| > 2V_{max} * dt \\ Y_f & else \end{cases} \tag{19}$$

### III. Simulation Results

In order to assess the effectiveness of our proposed navigation strategy, we conducted simulations using Python 3.10. We simulated a real three-wheeled omnidirectional mobile robot dynamic model, which included a built-in tracking controller as introduced by [25]. These simulations were conducted in various dynamic and static environments. The field of robotics necessitates intricate path planning that encompasses a multitude of factors leading to indeterminate outcomes, including sensor noise, modeling inaccuracies, extraneous interferences, and environmental disturbances. These uncertainties can be typified based on their origin, nature, conduct, and scope. The study considers white noise with normal (0,1) for the sensors' noise, prediction error between 0 to 2 times greater than robot diameter, and random velocity in the obstacles' movements.

We compared the performance of our newly designed log-concave strategy, which utilizes a recurrent neural network, with the results obtained from the CVXPY optimization library, as well as the RRT, LQR, and A-star navigation methods. Through these simulations, we aimed to answer the following questions and evaluate the performance of our proposed strategy:

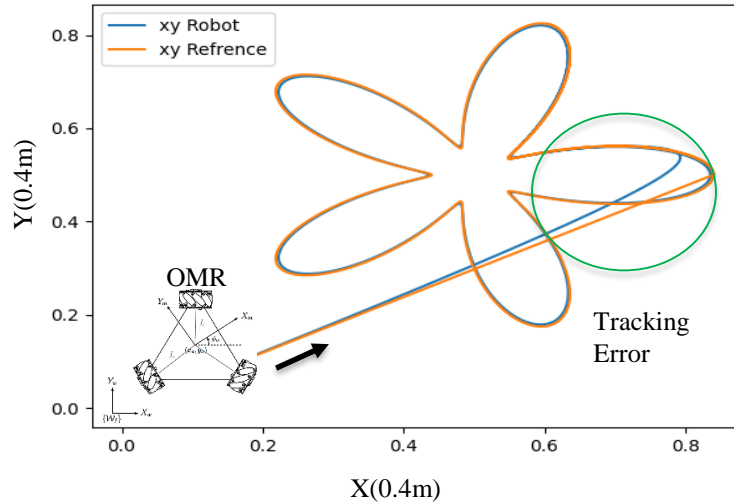
- How does our proposed strategy compare to existing navigation methods?
- How does our strategy perform in dynamic and static environments?
- What are the strengths and weaknesses of our proposed strategy?

The comparison was made to provide a benchmark for the performance of the proposed algorithm in terms of computational efficiency and path quality. While the approaches are fundamentally different, both methods have the ability to search in the neighborhood for feasible paths. A\*/RRT use a search algorithm to explore neighboring nodes to find a suitable path, while the proposed RNN structure uses a searching screen window to identify low-risk positions for path planning.

To evaluate the performance of each method, we conducted a series of simulations in dynamic and static environments. We compared the computational efficiency and path quality of the proposed RNN structure with that of A\*, RRT, LQR, and the proposed log-concave MPC using CVXPY optimization library. The metrics used to evaluate performance included the computation time, path length, and collision avoidance ability.

### **G. Impact of Dynamic Constraints on Path Planning Efficiency**

The dynamics of a robot can limit its ability to follow a planned path precisely, as shown in Figure 3.



**Figure 4 Path Tracking and Errors due to the robot’s dynamic and internal controller performance introduced by [25]**

Figure 4 exemplifies the plausible path tracking predicaments that mobile robot may face, encompassing inaccuracies and aberrations. These predicaments may culminate in the fiasco of collision evasion schemes, especially in fluctuating surroundings where barriers approach the automaton. To ensure the safe and efficacious operation of mobile automatons, it is crucial to ponder over the impact of dynamic constraints on the efficiency of path plotting. By apprehending the restrictions imposed by a robot's dynamics, we can formulate more sturdy and dependable path plotting algorithms that take into account the possibility of inaccuracies. This approach can decrease the probability of collisions and other safety perils while augmenting the comprehensive efficiency and performance of mobile robots.

#### **H. Performance of the Proposed Log-Concave MPC in Sample Environments with Static and Dynamic Obstacles**

To assess the effectiveness of the proposed navigation strategy, we utilized the CVXOPT library to solve the constrained MPC problem. Our evaluation included sample environments with both static and dynamic obstacles, all within a 20x20 screen size.

Figure 5 provides a visual representation of the robot's movements and the obstacles encountered every 0.3 seconds. As demonstrated in this figure, the robot successfully navigated to its target destination without any collisions.

To further evaluate the efficiency of our proposed strategy, we utilized Figure 6, which provides a global overview of the environment. Through these evaluations, we were able to demonstrate the effectiveness of our log-concave MPC approach in navigating mobile robots through environments with both static and dynamic obstacles.

#### **I. Difference between the Designed RNN Planner and the Optimization of Log-MPC Strategy**

This section evaluates the performance of RNN searching window and its efficiency in comparison to utilize CVXOPT optimization tool in another example of environment.

CVXOPT is a convex optimization library that provides solvers for a wide range of convex optimization problems, including nonlinear MPC. The specific algorithm used by CVXOPT for solving the nonlinear MPC problem depends on the formulation of the problem and the specific options selected by the user. In general, CVXOPT uses a primal-dual interior-point method for solving convex optimization problems [27]. This method iteratively solves a sequence of linear systems and updates the primal and dual variables until convergence. For nonlinear MPC problems, CVXOPT can use either a direct or an indirect method to solve the problem.

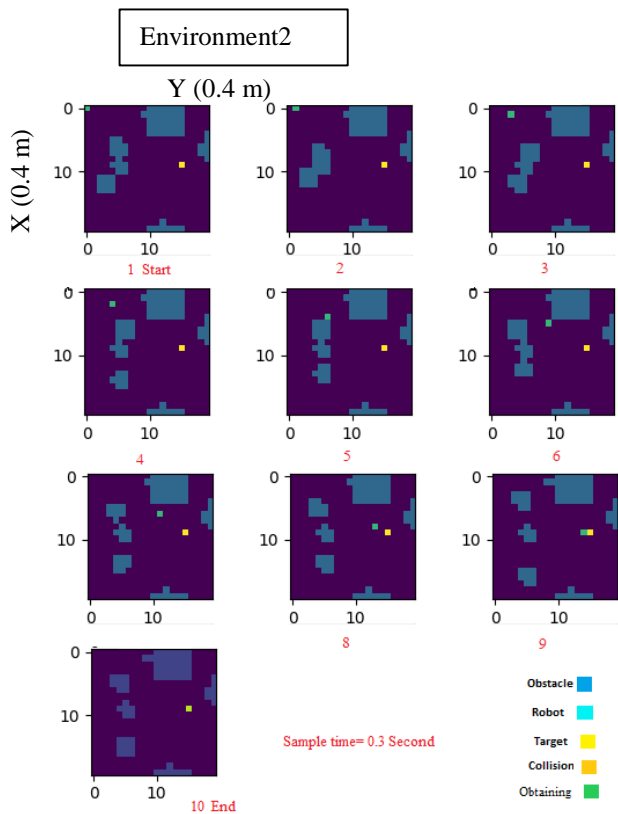


Figure 6 Log-Concave MPC path planning in dynamic Environment2

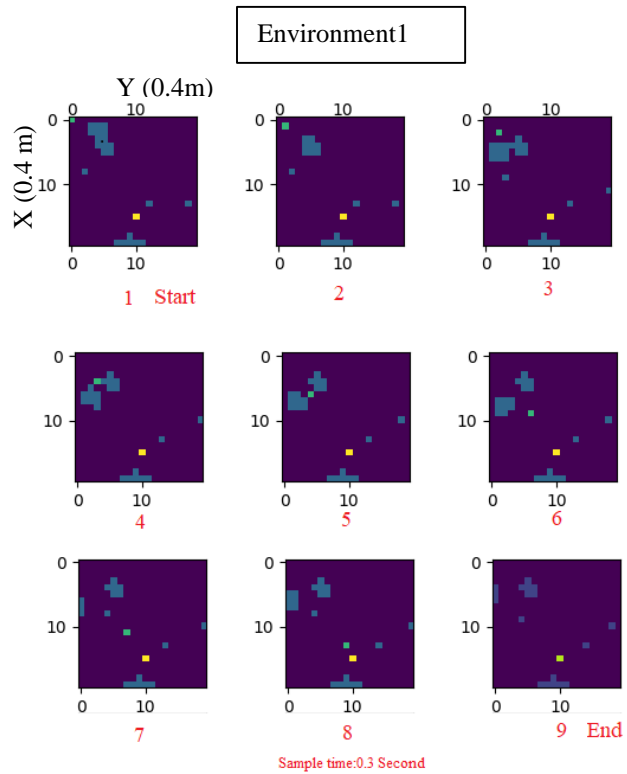


Figure 5 Log-Concave MPC path planning in dynamic Environment1

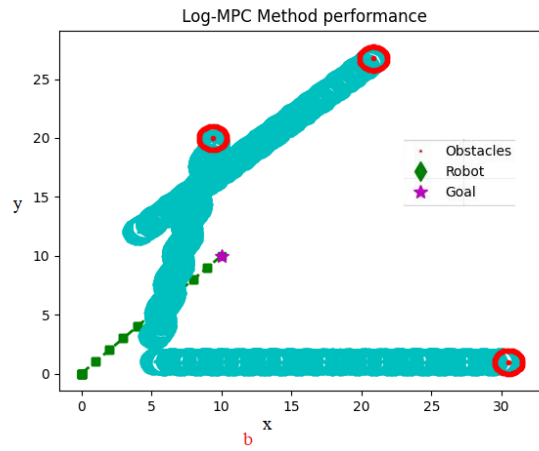
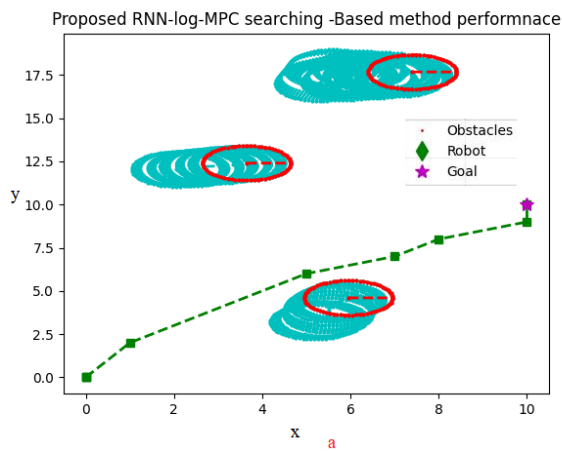
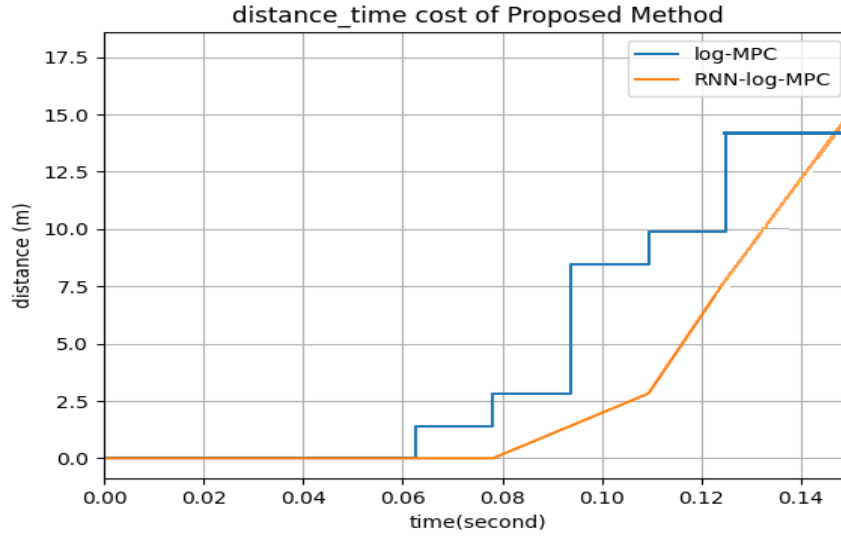


Figure 7 a) Proposed RNN-Log-MPC navigation performance in a dynamic environment b) Proposed Log-MPC navigation performance in a dynamic environment, utilizing CVXOPT optimization tool. Unit is 0.4 m, corresponding to the robot dimension.

Figure 7 reveals how both proposed path planning methods navigate the robot to destination, safely. This comparison has been conducted in the same dynamic environment, in which three dynamic environments start moving from the same initial points in both a and b figures and the obstacle's velocity are also considered the same.

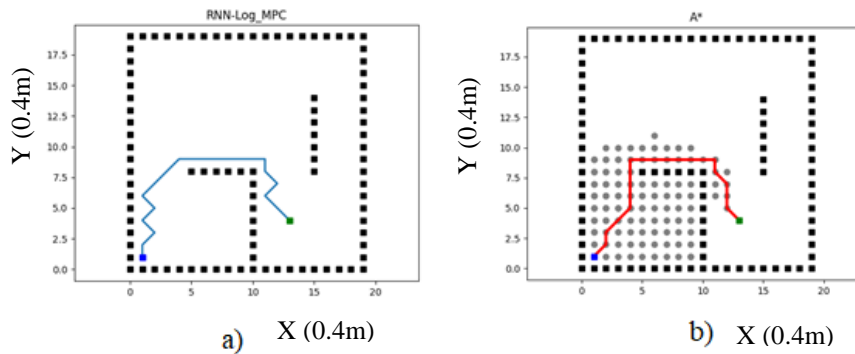
In Fig.7, the path passed by the obstacles is depicted by light blue circles. The current positions of obstacles are depicted in red circles. The robot passes through a green –square dash line, and the star is the target destination. Upon scrutinizing both Figure 7-a and 7-b, it becomes conspicuous that the simulation circumstances are strikingly alike. The motion of obstacles and their velocity remains consistent in both illustrations. However, a noticeable dissimilarity arises when contemplating the obstructions illustrated in Figure 7-b, as they progressively shift further away from the intended target locale in comparison to Figure 7-a. Therefore, the robot in Figure 7-b is coerced to traverse for an extended duration in order to attain the desired position. This incongruity can be credited to the inferior pace of the robot in Figure 7-b, which, in turn, is instigated by the path planner dispensing a smaller displacement by CVXOPT. In this situation, some generated paths of movements are in the range of 0.03 which cannot empower the robot efficiently, therefore, the robot stays at its position for several sequential sample times. In contrast, the initial training time of the RNN takes almost 0.05 (second) longer than the CVXOPT tool. Then, the robot is continuously navigated to the target position without any long time stopping in a particular position. Of course, we cannot regard stopping in a particular position as a disadvantage for a path planner, while in social environments, waiting for other people to pass the way and then continuing the optimal path or safe route is considered an advantage. On the other hand, this feature is not recommended in automated vehicles on highways, in air traffic, and other applications. In this case, RNN-Log-MPC can be more efficient. Figure.7 indicates the time-distance cost comparison.



**Figure 8 Comparison of Distance-time cost between Log-MPC and RNN-Log-MPC**

**J. Performances of the Proposed Log-MPC and RNN-Log-MPC Strategies in Comparison of RRT, A-star and LQ-MPC Methods**

Classic methods are not able to handle path planning programming in dynamic environments efficiently. Therefore, RRT and A-star as the classic methods are recommended to be utilized in static environments. On the other hand, reactive and heuristic methods are mostly utilized in dynamic environments. To evaluate the performance of the proposed Log-MPC method, the following table 2 depicts the comparison of RNN-Log-MPC and A-star performances in the given below static environment (Fig.9).



**Figure 9 a) RNN-Log-MPC path planning method in a static environment b) A\* method performance in a static environment (the left figure)**

Figure 9 illustrates a static environment example in which RNN-log-MPC method is compared with A\* strategy. The proposed strategy is considered with one prediction step as a global planner. The costs are compared in Table 2. Table 2 provides a comparison of both strategies in the same environment and with the same processor, significant efficiency of the proposed method, especially in the sense of time costs. In addition, the results illustrate that RRT may fail in providing a free-collision path in a dynamic environment, while the proposed method can navigate the robot efficiently.

**Table 2 RRT, A-star , LQ-MPC, and the Proposed Method (RNN-Log-MPC) Comparison in, including 3 moving obstacles and the same target point**

<b>Methods</b>	<b>Distance cost</b>	<b>Time cost</b>	<b>Collision with obstacle</b>	<b>Concern</b>
RNN-Log-MPC	15	0.14	No	Dependency on accurate prediction
Log-MPC using CVXPY	15	0.16	No	Infeasibility and Dependency on accurate prediction
RRT	24	0.2	Yes	No solution in some cases and failing in dynamic environment
A*	17	0.157	Yes	Collision and high computational cost in real time applications
LQ-MPC[28]	15.8	0.153	Yes	Cannot find solution in long distance target situations

The parameters of simulated RNN is given in the table 3:

**Table 3 RNN Parameters**

Items	Quantity
$\rho_n$	0.1
$\eta$	0.7
$\mu$	0.8

Another sample environment includes two sinusoidal shape movements of the obstacles and one random walker, with the target position at the point (12,15). The robot movement path is simulated under the following conditions:

1. When the utilized path planning algorithm is RRT
2. When the utilized path planning algorithm is the new proposed Navigator

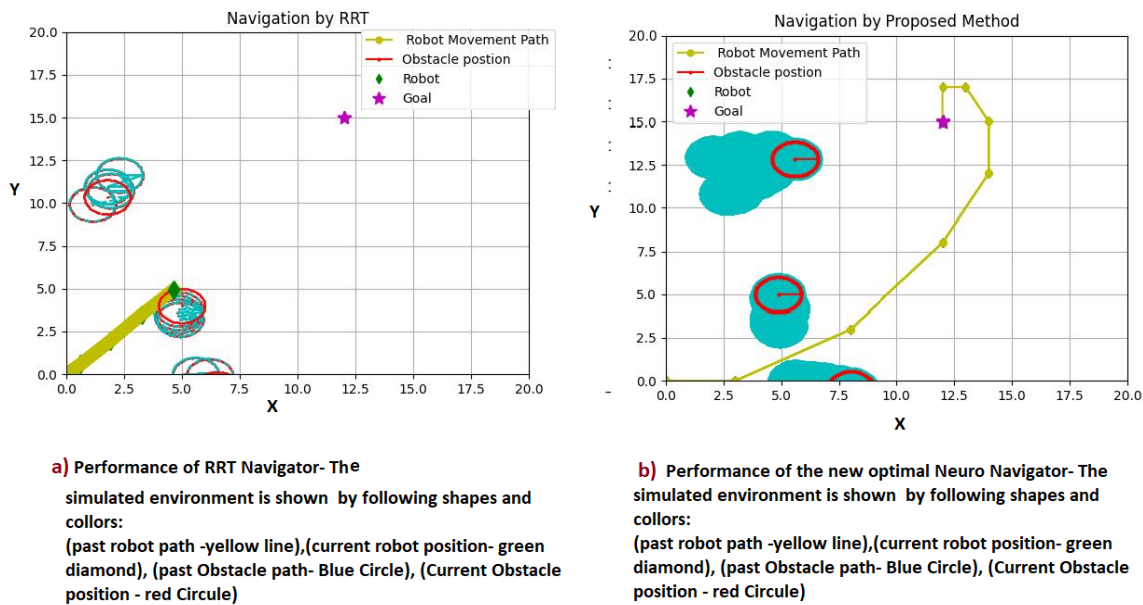
The global screen of the environment, including the movement path of the robot and obstacles are shown in Figure 10.

Figure 11 presents a more comprehensive comparison of our proposed method and RRT in a simulated environment with varying target positions. The figure showcases the success of our method and the failure of RRT. The target is represented by a violet star shape, while the robot is depicted as a green diamond. The obstacles are shown as red circles, and the passed path is indicated by the yellow line. The left side of both parts of Figure 10-a demonstrates the crucial scenes, while the right side shows the entire passed trajectory.

It is evident that generating a path using RRT can be time-consuming, especially if a unique solution is obtainable in this structure. To better understand this issue, we have provided a table below that compares our proposed method with RRT in terms of distance versus time. The simulation screen size is 20\*20 with a scale of 0.45 meters, and the arrangement of obstacles differs

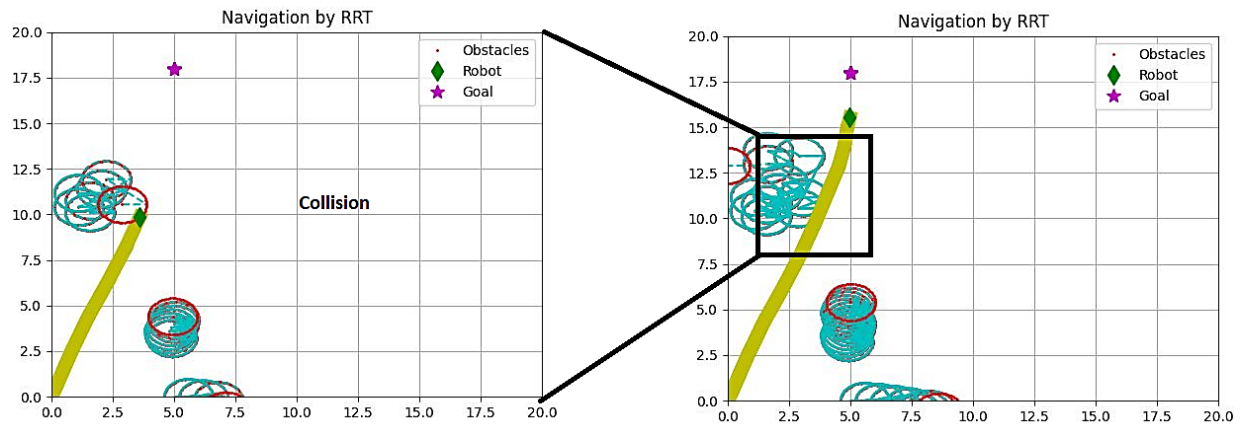
in at least five similar simulation examples. The target position is located at [12,15] in Figures 10 and 11.

Our proposed method offers a more efficient and effective solution for navigation in a simulated environment. With our method, the robot can navigate through the obstacles and reach the target in a shorter amount of time compared to RRT. This not only saves time but also reduces the risk of errors and increases the overall success rate of the navigation process.

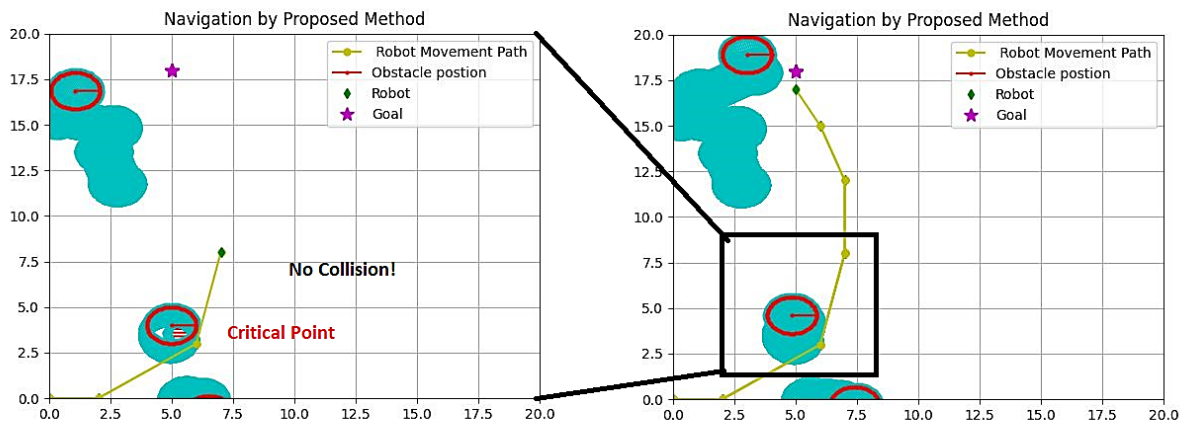


**Figure 10 Proposed method and RRT Path Planning in a dynamic environment (fourth Example)**

Figure (10-a & b) demonstrates a comparison of the RRT performance (10-a) with the suggested strategy (10-b). The results show that the RRT failed in providing a safe path, and consequently collides with the obstacle in a dynamic environment.



a) Performance of the RRT Navigator in the pervious Environment with different target position (5,17)



b) Performance of the Optimal Neuro Navigator in thepervious Environment with different target positon (5,17)

**Figure 11 Proposed method and RRT Path Planning in the same previous dynamic environment with different target**

Figure 11 presents a more comprehensive comparison of our proposed method and RRT in a simulated environment with varying target positions. The figure showcases the success of our method and the failure of RRT. The target is represented by a violet star shape, while the robot is depicted as a green diamond. The obstacles are shown as red circles, and the passed path is

indicated by the yellow line. The left side of both parts of Figure 10-a demonstrates the crucial scenes, while the right side shows the entire passed trajectory.

It is evident that generating a path using RRT can be time-consuming, especially if a unique solution is obtainable in this structure. To better understand this issue, we have compared our proposed method with RRT in terms of distance versus time in table 2.. The simulation screen size is 20\*20 with a scale of 0.45 meters, and the arrangement of obstacles differs in at least five similar simulation examples. The target position is located at [12,15] in Figures 10 and 11.

Our proposed method offers a more efficient and effective solution for navigation in a simulated environment. With our method, the robot can navigate through the obstacles and reach the target in a shorter amount of time compared to RRT. This not only saves time but also reduces the risk of errors and increases the overall success rate of the navigation process.

#### **K. Potential Challenges and Future Directions**

The proposed RNN-Log-MPC algorithm presents a notable advantage by utilizing both sensor and camera data to diminish the sensitivity of the navigation system to momentary data loss or prediction error by EKF. Nonetheless, this does not eliminate the exigency to tackle potential issues such as prediction errors, sensor disconnection, and lack of accuracy in the command follower controller. Therefore, a conservative approach to adjusting and evaluating restrictions may be deemed necessary, which could constrain optimization, particularly in unfamiliar environments where speed changes occur, and there is a possibility of collision risk.

Moreover, the precision of obstacle detection, which is processed via a central server and communicated wirelessly to the robot's internal controller, can also influence the system's performance. Hence, it is crucial to scrutinize the algorithm in various scenarios to identify

potential errors and devise solutions to rectify them. Constant monitoring of the system can also enhance the accuracy of performance and reliability of the navigation system.

#### **IV. Conclusion**

This study proposes a groundbreaking navigation algorithm that employs a log-concave strategy and a visual-based model predictive control (MPC) framework. The tactic is innately near-optimal and restricted, and it is tackled by employing two optimization resources, namely a newly proposed network configuration and the CVXOPT optimization library. The suggested recurrent neural network (RNN) architecture adeptly resolves diverse constrained navigation predicaments. The constraints, which include the velocities of the robot and the nearby obstacles, are given careful consideration to enhance the reliability of the navigation system. Our simulations demonstrate promising results, but further research is imperative to scrutinize the applicability of our methodology in pragmatic settings or using tangible mechanisms such as Gazebo or VREP to validate our approach and provide additional insights.

This investigation furnishes a resolution to the optimal visual navigation technique, where the structure size remains constant in various environments with different numbers of obstacles, by utilizing a window-based searching approach. Nonetheless, the Extended Kalman Filter, which is utilized to localize objects, is a highly favored stochastic method in this area, but it operates based on the linearization method. Consequently, there is still a concern about its failing events, which can trigger low-accurate predictions of the environment.

To summarize, our proposed RNN-based optimization methodology for circumventing obstacles showcases auspicious outcomes and harbours the potentiality of being expanded to more intricate scenarios and environments. For instance, our methodology may be implemented in pragmatic situations such as self-governing motoring, wherein the robot necessitates traversing through a multifarious and dynamic milieu while evading hindrances. Furthermore, our approach has the capacity to amalgamate with alternative state estimation and prognostication techniques to

advance the performance and resilience of the system. We are of the conviction that our study lays a robust groundwork for ensuing research in the establishment of RNN-based techniques for circumventing obstacles in autonomous robots.

### Acknowledgement

This research is underwritten by the Auckland University of Technology and is endowed through a summer project scholarship.

### Reference

- [1] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks—a review," *Pattern recognition*, vol. 35, no. 10, pp. 2279-2301, 2002.
- [2] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019: IEEE, pp. 3965-3972.
- [3] P. Yao, H. Wang, and Z. Su, "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerospace Science and Technology*, vol. 47, pp. 269-279, 2015.
- [4] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448-468, 2021.
- [5] I. Askari, B. Badnava, T. Woodruff, S. Zeng, and H. Fang, "Sampling-Based Nonlinear MPC of Neural Network Dynamics with Application to Autonomous Vehicle Motion Planning," in *2022 American Control Conference (ACC)*, 2022: IEEE, pp. 2084-2090.
- [6] H. Du, B. Hao, J. Zhao, J. Zhang, Q. Wang, and Q. Yuan, "A path planning approach for mobile robots using short and safe Q-learning," *Plos one*, vol. 17, no. 9, p. e0275100, 2022.
- [7] X. Xiao et al., "Learning model predictive controllers with real-time attention for real-world navigation," arXiv preprint arXiv:2209.10780, 2022.
- [6] P. Stano et al., "Model predictive path tracking control for automated road vehicles: A review," *Annual Reviews in Control*, 2022.

- [7] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020: IEEE, pp. 7629-7636.
- [10] M. H. Haider et al., "Robust mobile robot navigation in cluttered environments based on hybrid adaptive neuro-fuzzy inference and sensor fusion," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9060-9070, 2022.
- [11] M.-F. R. Lee and S. H. Yusuf, "Mobile robot navigation using deep reinforcement learning," *Processes*, vol. 10, no. 12, p. 2748, 2022.
- [12] Y.-H. Kim, J.-I. Jang, and S. Yun, "End-to-end deep learning for autonomous navigation of mobile robot," in 2018 IEEE International Conference on Consumer Electronics (ICCE), 2018: IEEE, pp. 1-6.
- [13] J. Wang, X. Ding, H. Xia, Y. Wang, L. Tang, and R. Xiong, "A LiDAR based end to end controller for robot navigation using deep neural network," in 2017 IEEE International Conference on Unmanned Systems (ICUS), 2017: IEEE, pp. 614-619.
- [14] H. Quan, Y. Li, and Y. Zhang, "A novel mobile robot navigation method based on deep reinforcement learning," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, p. 1729881420921672, 2020.
- [15] D. Limon, J. Calliess, and J. M. Maciejowski, "Learning-based nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7769-7776, 2017.
- [16] S. A. Saeedinia and M. Tale Masouleh, "The synergy of the multi-modal MPC and Q-learning approach for the navigation of a three-wheeled omnidirectional robot based on the dynamic model with obstacle collision avoidance purposes," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 236, no. 17, pp. 9716-9729, 2022.
- [17] M. Ramezani, H. Habibi, J. L. Sanchez-Lopez, and H. Voos, "UAV Path Planning Employing MPC-Reinforcement Learning Method Considering Collision Avoidance," in 2023 International Conference on Unmanned Aircraft Systems (ICUAS), 2023: IEEE, pp. 507-514.
- [18] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397-2404, 2023.
- [19] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24884-24900, 2021.

- [20] S.-D. Achirei, R. Mocanu, A.-T. Popovici, and C.-C. Dosofoei, "Model-Predictive Control for Omnidirectional Mobile Robots in Logistic Environments Based on Object Detection Using CNNs," *Sensors*, vol. 23, no. 11, p. 4992, 2023.
- [21] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in 2014 IEEE-RAS international conference on humanoid robots, 2014: IEEE, pp. 279-28
- [22] K. Yang, S. K. Gan, and S. Sukkarieh, "An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 101-122, 2010.
- [23] S. Cheng, L. Li, H.-Q. Guo, Z.-G. Chen, and P. Song, "Longitudinal collision avoidance and lateral stability adaptive control system based on MPC of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2376-2385, 2019.
- [24] G. Villarrubia, J. F. De Paz, P. Chamoso, and F. De la Prieta, "Artificial neural networks used in optimization problems," *Neurocomputing*, vol. 272, pp. 10-16, 2018.
- [25] O. M.-M. L. A. A. J.-C. M. G. & G.-H. R. Peñaloza-Mejía, "Motion control design for an omnidirectional mobile robot subject to velocity constraints," *Mathematical Problems in Engineering*, 2015
- [26] E. D. a. S. A. a. S. C. Blanchard, "Parameter estimation method using an extended Kalman filter," *Proceedings of the Joint North America, Asia-Pacific ISTVS Conference*, 2007.
- [27] J. Dahl and E. D. Andersen, "A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization," *Mathematical Programming*, vol. 194, no. 1-2, pp. 341-370, 2022.
- [28] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*, 2010: IEEE, pp. 987-993.