NMPC-LBF: Nonlinear MPC with Learned Barrier Function for Decentralized Safe Navigation of Multiple Robots in Unknown Environments

Amir Salimi Lafmejani, Spring Berman, and Georgios Fainekos

Abstract— **In this paper, we present a decentralized control approach based on a Nonlinear Model Predictive Control (NMPC) method that employs barrier certificates for safe navigation of multiple nonholonomic wheeled mobile robots in unknown environments with static and/or dynamic obstacles. This method incorporates a Learned Barrier Function (LBF) into the NMPC design in order to guarantee safe robot navigation, i.e., prevent robot collisions with other robots and the obstacles. We refer to our proposed control approach as NMPC-LBF. Since each robot does not have a priori knowledge about the obstacles and other robots, we use a Deep Neural Network (DeepNN) running in real-time on each robot to learn the Barrier Function (BF) only from the robot's LiDAR and odometry measurements. The DeepNN is trained to learn the BF that separates safe and unsafe regions. We implemented our proposed method on simulated and actual Turtlebot3 Burger robot(s) in different scenarios. The implementation results show the effectiveness of the NMPC-LBF method at ensuring safe navigation of the robots.**

I. Introduction

Collision-free navigation of multiple mobile robots has been extensively studied in the literature for different applications such as autonomous cars [1], warehouse automation [2], planetary exploration [3], and service robots [4]. The existing control approaches for collision-free navigation of multi-robot systems can be categorized as *centralized*, *distributed*, and *decentralized* methods. Although collision and deadlock avoidance of robots can be guaranteed in centralized control approaches, they suffer from the scalability issue since the computational complexity increases with the number of the robots [5], [6].

In distributed control approaches, inter-robot collision avoidance can be established using inter-robot communication [6], [7]. However, inter-robot communication might be unreliable, or even not possible. Thus, decentralized control approaches have been developed to eliminate the aforementioned limitations of centralized and distributed approaches. However, existing decentralized methods are not fully decentralized due to one or more of the following simplifying assumptions: **(1)** the reliance of robots on inter-robot communication; **(2)** a priori knowledge of the robots about the positions of obstacles in the environment; **(3)** dependency of the approach on a global localization system such as a camera or motion capture system; and **(4)** absence of obstacles in multi-robot scenarios.

In traditional control methods for collision-free navigation of mobile robots, collision avoidance has been achieved by incorporating the gradient of an artificial potential field into the controller design, e.g., navigation function-based methods [8] or attraction-repulsion methods [9]. Recently, several methods have been developed to incorporate collision avoidance into the constraints of an optimization problem in order to enforce the kinematics or dynamics of the robot, constraints on the robot's states and actuation, and the dynamics of the environment [5], [6], [10]. For multi-robot systems, collision avoidance is a more challenging task since we need to ensure avoidance of both inter-robot and robot-obstacle collisions. Collision avoidance can be guaranteed by introducing distance functions between each pair of the robots in a centralized framework, as described in [5]. However, using distance functions in a decentralized framework fails to ensure collision avoidance in dynamic environments, since the constraint does not capture the dynamics of unsafe regions. To address this issue, one can use Barrier Functions (BFs) to ensure collision avoidance. These functions separate safe and unsafe regions in the environment and can be included in a constrained minimization problem whose solution minimally deviates from a nominal controller with guaranteed stability [11].

As a promising control approach for collision-free navigation of mobile robots [5], Model Predictive Control (MPC) is a powerful feedback control method that computes optimal control solutions by solving a constrained optimization problem over a prediction horizon [12], [13]. There are several studies that incorporate BFs into constraints or directly into the objective function of the MPC optimization to ensure collision-free navigation of mobile robots. For instance, a safety-critical MPC method with discrete-time BF has been presented in [10] for collision-free navigation of mobile robots in known, static environments. The existing BF-based MPC methods analytically construct the BFs and incorporate them into MPC design to ensure collision-free navigation, which is impractical in real-world applications. Synthesizing BFs is straightforward in a centralized control architecture where the robots have information about the positions and geometry of the obstacles. Given this information, a BF, representing the boundary of the smallest circle that encloses an obstacle, could be defined for each obstacle. On the other hand, it would be challenging for each robot to individually

This work was partially supported by DARPA AMP N6600120C4020, NSF CNS 1932068, NSF IIP-1361926, and the NSF I/UCRC CES.

A. Salimi Lafmejani is with the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), Tempe, AZ, 85287 (asalimil@asu.edu). S. Berman is with the School for Engineering of Matter, Transport and Energy, ASU, Tempe, AZ 85287 (spring.berman@asu.edu). G. Fainekos is with the Toyota Research Institute of North America, Ann Arbor, MI 48105 (georgios.fainekos@toyota.com). The work was initiated when G. Fainekos was with the School of Computing and Augmented Intelligence, ASU, Tempe, AZ, 85281.

construct BFs in a decentralized manner using only its own sensor measurements due to the partial observability of the environment and sparsity of the measured data. To address this challenge, several recent studies have investigated the use of neural networks (NNs) to learn BFs offline or online in both known and unknown environments [14]–[19].

This paper presents a decentralized NMPC method with learned BF for collision-free navigation of multiple nonholonomic mobile robots in unknown environments. A learned BF in the form of a Deep Neural Network (DeepNN) is necessary since the map of the environment is unknown and the BF constraints must be incorporated into the NMPC optimization problem. That is, we need a map from a state of the robot (as used in the NMPC loss function) to the predicted BF value.

The following features of the proposed NMPC-LBF method differentiate it from existing methods for multi-robot navigation: **(1)** Robots do not require a priori information about the locations or geometry of other robots or obstacles. **(2)** There is no inter-robot communication that can be used to avoid collisions between robots. **(3)** Safe navigation is achieved in the presence of non-aggressive dynamic obstacles. **(4)** Due to its decentralization, the proposed method can be duplicated on any number of robots for safe navigation. **(5)** A global localization system is not required since the method can use on-board sensors of the robot. The main novelty of our work compared to approaches for learning BFs in [14]–[19] is to combine the NMPC with the learned BF. Furthermore, through experiments on actual Turtlebot3 Burger robot(s), we demonstrate that our method is feasible in practice (see [20]).

II. Control Problem Formulations

A. Nonlinear Model Predictive Control (NMPC)

Our proposed control approach utilizes a nonlinear MPC (NMPC) method. We use the following modified version of a discrete-time unicycle model that describes the kinematics of a nonholonomic wheeled mobile robot (WMR):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k))T_s \\ \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k)) &= \begin{bmatrix} \cos(\theta(k)) & -a\sin(\theta(k)) \\ \sin(\theta(k)) & a\cos(\theta(k)) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \end{aligned} \quad (1)$$

where k denotes the time step; T_s is the sampling time; a is a small positive constant; the state vector $\mathbf{x}(k) = [x(k) \quad y(k) \quad \theta(k)]^{\mathrm{T}}$ is the robot's pose, i.e., its position $\bar{\mathbf{x}} = [x \quad y]^{\mathrm{T}}$ and heading angle θ in the global coordinate frame at time step k; and the control input vector $\mathbf{u}(k) = [v(k) \quad \omega(k)]^{\mathrm{T}}$ contains the robot's control inputs, which are its linear velocity v and angular velocity ω at time step k. If one uses the standard unicycle kinematic model of a nonholonomic WMR as in [21], then the angular velocity of the robot does not show up in the barrier constraint [22]. Thus, we use the modified kinematic model in Eq. (1) so that the system has relative degree 1.

In an NMPC method, we first solve a nonlinear constrained optimization problem that minimizes a loss function $l(\mathbf{x},\mathbf{u})$ over a prediction horizon of N_p time steps:

$$\begin{aligned} \mathbf{U}^* = \text{argmin}_{\mathbf{u}} & \sum_{k=0}^{N_p-1} l(\mathbf{x}(k),\mathbf{u}(k)) \\ & \mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k))T_s \\ & \mathbf{x}_{\text{min}} \le \mathbf{x}(k) \le \mathbf{x}_{\text{max}}, \quad \mathbf{x}(0) = \mathbf{x}_c \\ & \mathbf{u}_{\text{min}} \le \mathbf{u}(k) \le \mathbf{u}_{\text{max}} \end{aligned} \quad (2)$$

where $\mathbf{x}_c$ is the robot's current odometry measurement of its pose and $\mathbf{U}^* \in \mathbb{R}^{2\times N_p}$ is a sequence of optimal control inputs for the future N_p time steps. The bounds $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$ on the state vector can be imposed to restrict the robot to move within a specific region, and the bounds $\mathbf{u}_{\text{min}}$ and $\mathbf{u}_{\text{max}}$ on the control inputs are determined by the capabilities of the robot's actuators. If the control objective is to drive the robot to a target pose $\mathbf{x}_{\text{ref}}$, then the loss function can be defined as the sum of two quadratic terms that quantify the normed distance of the robot from the target pose and the control effort:

$$l(\mathbf{x}(k),\mathbf{u}(k)) = ||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}||^2_{\mathbf{Q}} + ||\mathbf{u}(k)||^2_{\mathbf{R}}, \quad (3)$$

where $\mathbf{Q}$ and $\mathbf{R}$ are square weighting matrices and $||\mathbf{x}||^2_{\mathbf{A}} \equiv \mathbf{x}^T\mathbf{A}\mathbf{x}$. Given the optimal control inputs $\mathbf{U}^*$, only the first control input $\mathbf{U}^*(0)$ is applied to the robot's actuators. Then, the time step is incremented from k to $k+1$, and optimization problem (2) is solved again. We use the NMPC formulation in Eq. (3) without stabilizing terminal costs or terminal constraints, in order to speed up the convergence rate and reduce the computation time (see [23]).

B. Control Barrier Functions (CBFs)

We define *safety* as a criterion that prevents the control inputs from driving the robot into a collision with other robots or obstacles. Control barrier functions enable safety for control synthesis by providing forward invariance property of a specified *safe set* based on a Lyapunov-like condition. We define a safe set $\mathcal{C}$ that represents the free space of the domain, where the robot can move without colliding with an obstacle. This set is described by the superlevel set of a continuous differentiable function $h(\bar{\mathbf{x}}(k))$, which is known as a barrier function [11], [24]:

$$\mathcal{C} = \{\forall k \in \mathbb{Z}_0,\ \bar{\mathbf{x}}(k) \in \mathbb{R}^n |\ h(\bar{\mathbf{x}}(k)) \ge 0\}. \quad (4)$$

In order to prevent collisions, the control inputs must maintain the robot's position $\bar{\mathbf{x}}$ within the safe set $\mathcal{C}$. In other words, the set $\mathcal{C}$ must be *forward invariant* with respect to $\mathbf{f}(\bar{\mathbf{x}}(k),\mathbf{u}(k))$ defined in Eq. (1). The closed set $\mathcal{C}$ is called forward invariant if for every $\bar{\mathbf{x}}(0) \in \mathcal{C}$, $\bar{\mathbf{x}}(k) \in \mathcal{C}$ for all $k \in \mathbb{Z}_0$. Then, the safety certificate at time step k during robot navigation can be encoded as a Control Barrier Condition (CBC) as follows:

$$h(\bar{\mathbf{x}}(k+1)) - h(\bar{\mathbf{x}}(k)) + \gamma h(\bar{\mathbf{x}}(k)) \ge 0, \quad (5)$$

where γ is a small positive value less than one, i.e., $0 < \gamma \le 1$. In [10], it was proved that adding the CBC (5) to the constraints of the MPC optimization problem (2) ensures the safety of the computed optimal control inputs.

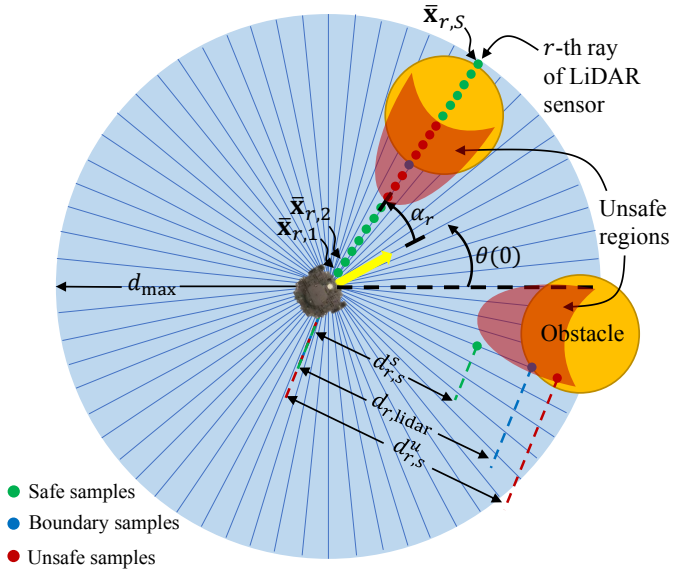

Fig. 1. LiDAR sensor rays and point sampling in safe and unsafe regions within the sensing range of the LiDAR. The unsafe samples are taken from the red shaded regions and the safe samples are in areas excluding the unsafe regions. $d^s_{r,s}$, $d_{r,\text{lidar}}$, $d^u_{r,s}$ show the distances of the robot to the safe, boundary, and unsafe samples on a ray of LiDAR.

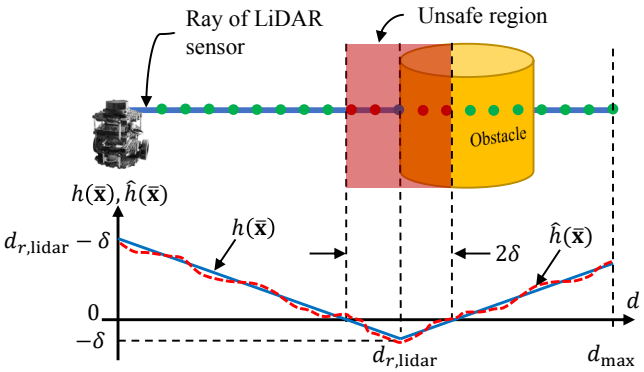

Fig. 2. An illustration of ground-truth BF, $h(\bar{\mathbf{x}})$, and approximation of BF by the DeepNN, $\hat{h}(\bar{\mathbf{x}})$, on a single ray of LiDAR along with a 2D visualization of safe and unsafe samples and a sample on the boundary of an obstacle. We note that the sample points are defined in the $x-y$ plane with a height of zero.

III. Decentralized NMPC-LBF Method

In our NMPC-LBF method, the BF h is learned by training a DeepNN $\hat{h}$ in real-time on each robot. Then, the trained DeepNN $\hat{h}$ is used within the NMPC constraints to provide an approximation of the BF h for the finite prediction horizon of N_p time steps.

A. Data Sampling and Training the DeepNN BF

The DeepNN should be trained to learn an approximation of the BF. To train the DeepNN, we need to collect samples from safe and unsafe regions that are observed by the robot during navigation. Figure 1 demonstrates our method to collect samples at each time instant. These samples are only obtained in real-time based on the LiDAR sensor readings and the current pose of the robot via odometry readings.

We define R as the number of rays on the LiDAR sensor, $r = \{1, 2, ..., R\}$ as the index for the rays, α_r as the angle between the direction of the robot's heading and the r-th ray of the LiDAR, and $d_{r,\text{lidar}}(k)$ as the distance measured by the LiDAR in the direction of the r-th ray at the k-th time step. Then, we sample points within the sensing range of the robot along each ray of the LiDAR as illustrated in Fig. 1. Let us define $d_{\max}$ as the maximum distance that the robot can sense. Then, we take S number of samples on each ray of the LiDAR. We define $\bar{\mathbf{x}}_{r,s}(k) \in \mathbb{R}^2$ as the position in the global coordinate frame of the s-th sample, $s \in \{1, 2, ..., S\}$, on the r-th ray at the k-th time step. Therefore, we have in total $N_{\text{samples}} = R \times S$ samples available in the dataset at each time step to train the DeepNN. By calculating $d_{r,s} = d_{\max} s/S$ as the distance of sample position $\bar{\mathbf{x}}_{r,s}(k)$ to the origin of the robot's local frame, we can readily compute the position of the sample as:

$$\bar{\mathbf{x}}_{r,s}(k) = \bar{\mathbf{x}}_c + \mathbf{R}_z(\theta(k))\mathbf{d}_{r,s}(k)$$
$$\mathbf{R}_z(\theta(k)) = \begin{bmatrix} \cos(\theta(k)) & -\sin(\theta(k)) \\ \sin(\theta(k)) & \cos(\theta(k)) \end{bmatrix} \quad (6)$$
$$\mathbf{d}_{r,s}(k) = d_{r,s} \begin{bmatrix} \cos(\alpha_r(k)) \\ \sin(\alpha_r(k)) \end{bmatrix}$$

where $\theta(k)$ is the heading angle of the robot at the k-th time step and $\mathbf{R}_z(\theta(k))$ is the standard rotation matrix that performs a rotation through angle $\theta(k)$ around about the positive z axis. This rotation matrix transforms vectors described in the local frame of the robot into vectors described in the global frame $x-y$. We note that $\bar{\mathbf{x}}_{r,s}$ is located in the safe region, the boundary of an obstacle, or an unsafe region within the sensing range of the LiDAR. Thus, the input for training the DeepNN is the set of all sampled points, defined as the vector $\mathbf{X}_s \in \mathbb{R}^{N_{\text{samples}} \times 2}$. We describe the ground-truth outputs as the value of the BF at the input samples:

$$h(\bar{\mathbf{x}}_{r,s}(k)) = |d_{r,s} - d_{r,\text{lidar}}(k)| - \delta, \quad (7)$$

where $\delta > 0$ denotes half of the width of the unsafe region around the distance measured by the LiDAR and is set to a positive value greater than the radius of the smallest circle surrounding the robot.

Thus, the input-output set of data for training the DeepNN can be described by $\mathcal{D} = \{\bar{\mathbf{X}}_s, \mathbf{H}_s\}$ in which each row of $\bar{\mathbf{X}}_s \in \mathbb{R}^{N_{\text{samples}} \times 2}$ is the position vector of the sampled point $\bar{\mathbf{x}}_{r,s}$, $s = \{1, 2, ..., S\}$, and each row of $\mathbf{H}_s \in \mathbb{R}^{N_{\text{samples}}}$ is the corresponding ground-truth output value of BF, $h(\bar{\mathbf{x}}_{r,s})$. We employ the *incremental learning* method to train the DeepNN $\hat{h} : \mathbb{R}^2 \to \mathbb{R}$ that provides an approximation of the BF. Figure 2 illustrates samples on a single ray of the robot's LiDAR and the change of the ground-truth BF and its approximation with respect to the distance to an obstacle.

In the current implementation of our method, training data is collected and used for training the DeepNN at every iteration of the control loop. However, other (re-)training

schemes are possible – especially, if we would like to reduce the impact of over-fitting and/or consider NN-based prediction models for the dynamic obstacles.

B. Symbolic Representation of the DeepNN BF

When an analytical representation of the CBC (5) is available along with a map of the environment and motion predictions of the dynamic obstacles, then the BF conditions could be directly incorporated in the constraints of the NMPC (2). However, in our application, we consider an unknown map. Therefore, we need to use a learning-based BF $\hat{h}$ to predict the CBC. As a consequence, a symbolic representation of the DeepNN $\hat{h}$ is necessary so that it can be used in the resulting optimization problem.

Recalling that $\bar{\mathbf{x}}(k)$, where $k \in \{0, 1, ..., N_p - 1\}$, denote the future states of the robot for the NMPC finite horizon, we can obtain a symbolic expression of the approximated BF for each predicted future state, given the activation functions on each node of the DeepNN and the optimal weights after each training. We give the symbolic expression of $\bar{\mathbf{x}}(k)$ as the input to the DeepNN and obtain a symbolic expression for the approximation of the BF, i.e., $\hat{h}(\bar{\mathbf{x}}(k))$, at the output of the DeepNN. The symbolic expression is used to generalize the representation of the approximated BF with respect to $\bar{\mathbf{x}}(k)$ that allows us to compute $\hat{h}(\bar{\mathbf{x}}(k))$ for any possible future state of the robot. The approximated BF can be easily computed by using the Forward Propagation (FP) technique. The required computations in FP to obtain a symbolic expression of the approximated BF are:

$$\begin{aligned} &\mathbf{A}_l = \sigma(\mathbf{Z}_l), \quad \mathbf{Z}_l = {}^{l-1}\mathbf{W}_l^{\mathrm{T}}\mathbf{A}_{l-1} + \mathbf{b}_l \\ &\mathbf{A}_0 = \bar{\mathbf{x}}(k), \ \mathbf{A}_L = \hat{h}(\bar{\mathbf{x}}(k)), \quad l = 1, 2, ..., L \end{aligned} \tag{8}$$

where 0 is the index for the input layer, l is the index for hidden layers, L denotes the number of hidden layers, and σ is the activation function on each node. The weight matrix containing the weights on the connections between two consecutive layers $l-1$ and l is defined by ${}^{l-1}\mathbf{W}_l$. Moreover, $\mathbf{Z}_l$ is an affine transformation on the previous layer's output $\mathbf{A}_{l-1}$, and $\mathbf{A}_l$ describes the output of layer l after applying the activation function on $\mathbf{Z}_l$.

Remark: In some cases, we may want to use the continuous time formulation of BF [24], e.g., in an approximation method. In continuous time, the equivalent CBC condition requires the derivative of the BF. A symbolic representation of the gradient of the approximated BF, i.e., $\nabla\hat{h}(\bar{\mathbf{x}}(k))$, is not as straightforward as deriving a symbolic representation for $\hat{h}(\bar{\mathbf{x}}(k))$. However, it is still possible to compute it. To do so, we can use the Back Propagation (BP) technique to calculate a symbolic expression for the gradient with respect to the DeepNN's inputs, i.e., $\bar{\mathbf{x}}(k)$. For example, if we assume that $\sigma(x) = \tanh(x)$, then through BP, we obtain a symbolic expression for the approximated gradient of the BF with respect to the inputs by:

$$\begin{aligned} &\frac{\partial\hat{h}(\bar{\mathbf{x}}(k))}{\partial\bar{\mathbf{x}}(k)} = \frac{\partial\mathbf{A}_L}{\partial\mathbf{Z}_L}.\frac{\partial\mathbf{Z}_L}{\partial\mathbf{A}_{L-1}} \cdots \frac{\partial\mathbf{A}_1}{\partial\mathbf{Z}_1}.\frac{\partial\mathbf{Z}_1}{\partial\bar{\mathbf{x}}(k)} \\ &\frac{\partial\mathbf{A}_l}{\partial\mathbf{Z}_l} = 1 - \tanh^2(\mathbf{Z}_l), \quad \frac{\partial\mathbf{Z}_l}{\partial\mathbf{A}_{l-1}} = {}^{l-1}\mathbf{W}_l^{\mathrm{T}}. \end{aligned} \tag{9}$$

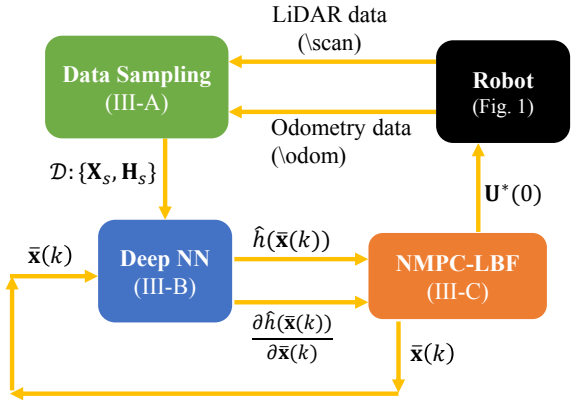

Fig. 3. Block diagram of the NMPC-LBF method described in Eq. (11). \scan and \odom are ROS topics through which the states of the robot and the LiDAR measurements would be available.

C. Incorporating BF into NMPC

Given the BF approximated by the DeepNN, we define sets that correspond to the safe region, its boundary, and the unsafe region of the environment, respectively:

$$\begin{aligned} &\mathcal{C} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) > \delta\}, \quad \partial\mathcal{C} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) = \delta\}, \\ &\mathcal{U} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) < \delta\} \end{aligned} \tag{10}$$

For all future N_p time steps, we calculate $\hat{h}(\bar{\mathbf{x}}(k))$ via FP computations through the DeepNN, which also determines whether each future state will be in the set $\mathcal{C}$, $\partial\mathcal{C}$, or $\mathcal{U}$. The constrained optimization problem of our NMPC-LBF method is formulated as:

$$\begin{aligned} \mathbf{U}^*, \mathbf{X}^* = \operatorname{argmin}_{\mathbf{x},\mathbf{u}} &\sum_{k=0}^{N_P-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\ &\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))T_s \\ &\mathbf{x}_{\min} \le \mathbf{x}(k) \le \mathbf{x}_{\max}, \quad \mathbf{x}(0) = \mathbf{x}_c \\ &\mathbf{u}_{\min} \le \mathbf{u}(k) \le \mathbf{u}_{\max} \\ &\hat{h}(\bar{\mathbf{x}}(k+1)) - \hat{h}(\bar{\mathbf{x}}(k)) + \gamma\hat{h}(\bar{\mathbf{x}}(k)) \ge 0 \ \text{(CBC)} \end{aligned} \tag{11}$$

where $\mathbf{X}^*$ is a sequence of optimal states. In order to reduce the computational complexity in this optimization problem, we use the multiple shooting method and lift up the problem by considering both $\mathbf{x}$ and $\mathbf{u}$ as decision variables in the minimization. We implement and solve this constrained nonlinear optimization problem via CasADi, which is a powerful framework specialized for solving NMPC problems by providing a symbolic expression of the problem. In the decentralized framework of our method, each robot solves the optimization problem described in Eq. (11) independently in which the BF is being learned by the DeepNN online in the loop at each time step. Figure 3 shows a block diagram of the NMPC-LBF method.

D. Implementation and Parameter Tuning

We describe the implementation of our method using the pseudo codes in Algorithms 1 and 2. Algorithm 1 presents the NMPC-LBF method step by step. After initialization of

Algorithm 1 NMPC-LBF Method

Input: $\mathbf{x}(0)$, $\mathbf{x}_{\text{ref}}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{x}_{\min}$, $\mathbf{x}_{\max}$, $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$, T_s, N_p

Output: $\mathbf{U}^*(0)$

1: Initialize ROS node, odometry and LiDAR scan subscribers, and velocity command publishers; symbolically formulate Eq. (11) using CasADi
2: **while** $||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k)|| > e_{\text{ref}}$ **do**
3: Obtain robot's pose and LiDAR measurements
4: Algorithm 2: Data Sampling and DeepNN Training
5: Solve optimization problem in Eq. (11) to obtain $\mathbf{U}^*$
6: Publish $\mathbf{U}^*(0)$ to ROS topics of robot's velocity commands
7: Update initial guess: $\mathbf{U}(0) \leftarrow \mathbf{U}^*(0)$, $\mathbf{X}(0) \leftarrow \mathbf{x}_c$
8: Increment time step: $k \leftarrow k + 1$
9: **end while**

Algorithm 2 Data Sampling and DeepNN Training

Input: $\mathbf{x}_c$, $\mathbf{x}(k)$, $d_{r,\text{lidar}}$, $d_{\max}$, R, S, l_r, n_{epochs}

Output: $\hat{h}(\mathbf{x}(k)), \partial\hat{h}(\mathbf{x}(k))/\partial\mathbf{x}(k)$

1: $\bar{\mathbf{X}}_s = [\,]$, $r = 0$, $s = 0$
2: **for** $r < R$ **do**
3: Compute $d_{r,s}$, α_r given r and s
4: **for** $s < S$ **do**
5: Compute $\mathbf{d}_{r,s}$ in Eq. (6)
6: Obtain sample positions $\bar{\mathbf{x}}_{r,s}$ in Eq. (6)
7: Collect data samples $\bar{\mathbf{X}}_s \leftarrow \bar{\mathbf{x}}_{r,s}$
8: **end for**
9: **end for**
10: Calculate and return $\hat{h}(\bar{\mathbf{x}}(k))$ via FP

the problem and formulating the optimization problem (line 1), the main loop (line 2 to 9) is executed online to collect the samples and train the DeepNN as described in Algorithm 2 given the robot's current pose from odometry and LiDAR measurements. We obtain optimal control inputs after solving Eq. (11) and apply them to the robot. Then, we shift the prediction horizon and initialize the states and control inputs in the optimization problem with a warm start. This loop is executed up to the point that the distance of the robot to its goal becomes less than $e_{\text{ref}} = 0.1$.

There are several parameters in the NMPC-LBF method that should be tuned carefully to achieve the expected performance of the robot to safely navigate in the environment. To implement the NMPC-LBF method, we recommend the following values for parameters to achieve the desired performance: prediction horizon $N_p = 10 \sim 20$, sampling time $T_s = 0.01 \sim 0.05$ s, weight matrices $\mathbf{Q} = \text{diag}(5, 5, 0.05)$ and $\mathbf{R} = \text{diag}(2, 0.5)$, learning rate $l_r = 0.01$, number of samples on each ray of LiDAR $S = 50$, half width of unsafe region $\delta = 0.2$ m, number of epochs $n_{\text{epochs}} = 20$ in the training of DeepNN, and $\beta = 0.1 \sim 0.2$ in the CBC. We use a fully-connected DeepNN that is implemented in TensorFlow [25] and Keras [26] with 2 neurons at the input, and one output. The network architecture for the hidden layers is $n_l = \{32, 32, 16, 16, 8\}$, and we use $\tanh$ as the activation function for all nodes.

IV. Simulation and Experimental Results

To evaluate the effectiveness of our method, we implemented it on simulated TurtleBot3 (TB3) Burger robots [27] in Gazebo and on an actual TB3 Burger robot.

For the simulation analysis, we simulated different scenarios for navigation of single and multiple robots in unknown environments. Due to space limitations, we only present results for two of these scenarios in this paper. Videos of the experiments with the real robot and all simulations, including the simulations not discussed here, are available at [20].

In scenario 1, a single robot should stabilize to a goal position in an environment with six unknown static obstacles. Figure 4 shows a snapshot of the initial configuration of the robot and its trajectory during a simulation of this scenario. The distance of the robot to the goal position and the optimal control inputs over time are presented in Figure 5. In our previous work [21], this control objective was achieved in the same environment using gradient-based feedback controllers that require the robot to have a priori knowledge about the positions and geometry of the obstacles. In contrast, the NMPC-LBF method can perform online learning in an unknown environment with dynamic obstacles and multiple robots. This property of the NMPC-LBF method allows its real-time deployment on actual robots.

In scenario 2, four robots should stabilize to their goal positions while avoiding collisions with one another and two unknown static obstacles in the environment. Figure 6 shows a snapshot of the initial configuration of the robots (R $\#i$, $i = \{1, 2, 3, 4\}$) and their trajectories during a simulation of this scenario. Plots of the distances of the robots to their goal positions over time are shown in Fig. 7.

In experimental tests (see the video [20]), we implemented the NMPC-LBF method on a single robot in three different scenarios in which the robot should stabilize to its goal position in an environment with one or two unknown obstacle(s). Two of the scenarios have static obstacle(s), and the third has a dynamic obstacle. To implement the controller, we used a computer with Linux Ubuntu 16.0, Intel Core i5 processor, 8GB memory, and ROS Kinetic. The average computation time for solving the NMPC-LBF is 0.25 s and the update rate of the DeepNN weights is 0.15 s. The NMPC-LBF code in Python is available at [28].

V. Analysis and Discussion

A. Stability and Feasibility

The stability of the proposed control approach can be established using proofs similar to those in our previous work [5] if we can show that the associated optimization problem in Eq. (11), with CBCs as constraints, has recursive feasibility [29]. Let us denote $\mathbb{X}$ and $\mathbb{U}$ as the feasible sets of states and control inputs for the robot. We use the simplified notation $\mathbb{Z} = \mathbb{X} \times \mathbb{U}$. Suppose that $\bar{\mathbf{x}}(0)$ is known and is

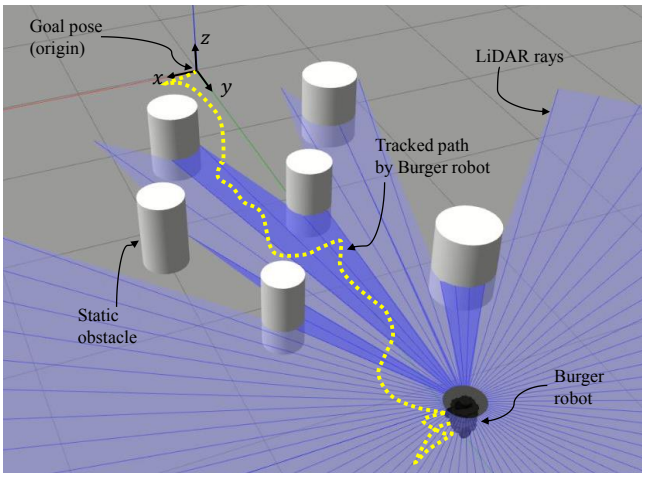

Fig. 4. A snapshot of the simulation of scenario 1 in Gazebo. The robot should stabilize to the goal position at the origin of the global frame.

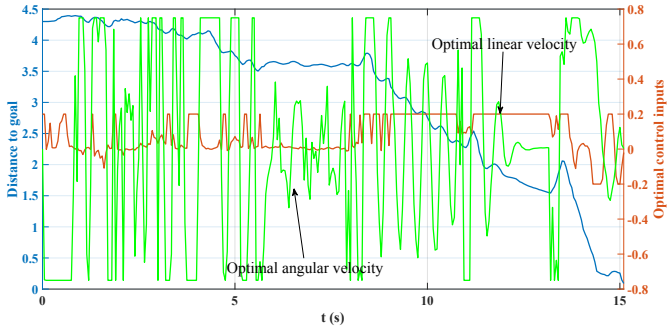

Fig. 5. Distance to goal position and optimal control inputs over time of the robot in scenario 1.

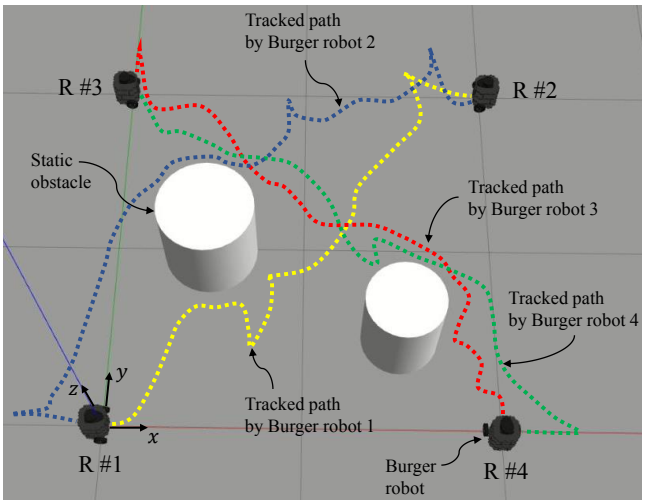

Fig. 6. A snapshot of the simulation of scenario 2 in Gazebo. All robots should stabilize to their goal positions while avoiding collisions with other robots and static obstacles.

in a feasible state, i.e., $\bar{\mathbf{x}}(0) \in \mathbb{X}$. Then, the optimization problem in Eq. (11) has feasible solutions if and only if for all time steps $0, ..., N_p - 1$: **(a)** the optimal control solutions and corresponding predicted states are a subset of the feasible set, i.e., $\mathbf{X}^* \times \mathbf{U}^* \subseteq \mathbb{Z}$, and **(b)** the CBCs in Eq. (5) are

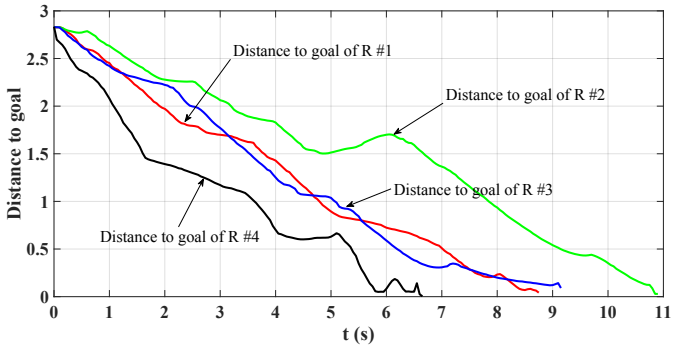

Fig. 7. Distances of the robots to their goal positions over time in scenario 2.

satisfied.

To prove statement **(a)**, we consider the kinematic model of each robot in Eq. (1). If the initial state and initial control input are in the feasible set, i.e., $\bar{\mathbf{x}}(0) \times \mathbf{u}(0) \in \mathbb{Z}$, then the recursive feasibility of the optimization problem in Eq. (11) trivially holds. To establish statement **(b)**, we would need two conditions: (i) $\hat{h}$ is a "good" approximation of h, and (ii) the sampling rate T_s is sufficient. Even though condition (ii) can be guaranteed given the dynamics of the robots and the obstacles, condition (i) remains challenging to demonstrate, in general. Using NN verification methods, e.g., [30], [31], we could establish that the approximation error is bounded. If the error is bounded, then we could guarantee that any control input that satisfies the CBC in the optimization problem in Eq. (11) will also satisfy the CBC (5). This will be the focus of our future work.

B. Limitations of Proposed NMPC-LBF Method

As a limitation of our decentralized method, a powerful computational resource on the robot is required to train the DeepNN and solve the optimization problem of NMPC-LBF in real-time. In turn, this allows us to increase the resolution of sampling points on each ray of the LiDAR to collect more data, which could preempt the over-fitting problem in training of the DeepNN. Another limitation is the challenging task of tuning the parameters of the NMPC-LBF method, which significantly affects the safety and stability of the robot during navigation. Although inter-robot collision avoidance can be ensured in our method, there are some special cases in which collision avoidance between the robot and dynamic obstacles is not guaranteed. If the velocity vector of a dynamic obstacle aligns with the velocity vector of the robot and has a greater magnitude, then they might collide since the robot's control inputs are constrained and there are no known constraints on the obstacle's dynamics.

Due to the generalization error in learning of the BF by the DeepNN, there always exists a difference between the learned BF and its ground-truth values. Given the ground-truth values of the BF, computing the approximation error for the training sample positions would be possible, whereas we are not able to compute it for test sample positions that are not in our dataset. This issue could be addressed by using

an NN verification method, e.g., [30], [31], to bound the approximation error. Lastly, the NMPC-LBF method does not necessarily generate smooth trajectories while approaching the obstacles. This issue could be solved by using a stochastic NMPC method as described in [32] to generate graceful motions of the robot during navigation.

VI. Conclusions and Future Work

In this paper, we presented a decentralized control approach based on an NMPC method leveraged by a DeepNN to learn BF to ensure safety for navigation of mobile robots in unknown environments in the presence of other robots and obstacles. The proposed method does not require inter-robot communication and it can be scaled up to be implemented on any number of robots. Future work includes modifying the method for learning the BF over the prediction horizon, and therefore estimating the unsafe regions at future time steps, by using a history of the robots' LiDAR readings as inputs to the DeepNN. Another direction for future work is to redesign the optimization problem in order to encourage smoothness in the robots' navigation.

References

[1] Y. C. Hou, K. S. Mohamed Sahari, L. Y. Weng, H. K. Foo, N. A. Abd Rahman, N. A. Atikah, R. Z. Homod, Development of collision avoidance system for multiple autonomous mobile robots, International Journal of Advanced Robotic Systems 17 (4) (2020) 1729881420923967.

[2] T. Fan, P. Long, W. Liu, J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios, The International Journal of Robotics Research 39 (7) (2020) 856–892.

[3] K. Otsu, A.-A. Agha-Mohammadi, M. Paton, Where to look? predictive perception with applications to planetary exploration, IEEE Robotics and Automation Letters 3 (2) (2017) 635–642.

[4] K. Majd, S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov, G. Fainekos, Safe navigation in human occupied environments using sampling and control barrier functions, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

[5] A. Salimi Lafmejani, S. Berman, Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots, Robotics and Autonomous Systems 141 (2021) 103774.

[6] L. Wang, A. D. Ames, M. Egerstedt, Safety barrier certificates for collisions-free multirobot systems, IEEE Transactions on Robotics 33 (3) (2017) 661–674.

[7] M. W. Mehrez, T. Sprodowski, K. Worthmann, G. K. Mann, R. G. Gosine, J. K. Sagawa, J. Pannek, Occupancy grid based distributed MPC for mobile robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 4842–4847.

[8] S. Paternain, D. E. Koditschek, A. Ribeiro, Navigation functions for convex potentials in a space with convex obstacles, IEEE Transactions on Automatic Control 63 (9) (2017) 2944–2959.

[9] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 2, 1985, pp. 500–505.

[10] J. Zeng, B. Zhang, K. Sreenath, Safety-critical model predictive control with discrete-time control barrier function, in: American Control Conference (ACC), 2021, pp. 3882–3889.

[11] A. D. Ames, J. W. Grizzle, P. Tabuada, Control barrier function based quadratic programs with application to adaptive cruise control, in: IEEE Conference on Decision and Control (CDC), 2014, pp. 6271–6278.

[12] F. Borrelli, A. Bemporad, M. Morari, Predictive control for linear and hybrid systems, Cambridge University Press, 2017.

[13] M. W. Mehrez, K. Worthmann, J. P. Cenerini, M. Osman, W. W. Melek, S. Jeon, Model predictive control without terminal constraints or costs for holonomic mobile robots, Robotics and Autonomous Systems 127 (2020) 103468.

[14] S. Yaghoubi, G. Fainekos, S. Sankaranarayanan, Training neural network controllers using control barrier functions in the presence of disturbances, in: IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1–6.

[15] K. Long, C. Qian, J. Cortés, N. Atanasov, Learning barrier functions with memory for robust safe navigation, IEEE Robotics and Automation Letters 6 (3) (2021) 4931–4938.

[16] H. Zhao, X. Zeng, T. Chen, Z. Liu, Synthesizing barrier certificates using neural networks, in: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control, 2020, pp. 1–11.

[17] M. Saveriano, D. Lee, Learning barrier functions for constrained motion planning with dynamical systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 112–119.

[18] M. Srinivasan, A. Dabholkar, S. Coogan, P. A. Vela, Synthesis of control barrier functions using a supervised machine learning approach, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 7139–7145.

[19] C. Li, Z. Zhang, A. Nesrin, Q. Liu, F. Liu, M. Buss, Instantaneous local control barrier function: An online learning approach for collision avoidance, arXiv preprint arXiv:2106.05341 (2021).

[20] Autonomous Collective Systems Laboratory YouTube channel, Nonlinear MPC with Learned Barrier Function for Decentralized Safe Navigation of Mobile Robots, https://www.youtube.com/watch?v=I3mAqJKf5dk (2021).

[21] A. S. Lafmejani, H. Farivarnejad, S. Berman, Adaptation of gradient-based navigation control for holonomic robots to nonholonomic robots, IEEE Robotics and Automation Letters 6 (1) (2021) 191–198.

[22] W. Xiao, C. Belta, Control barrier functions for systems with high relative degree, in: 2019 IEEE 58th Conference on Decision and Control (CDC), IEEE, 2019, pp. 474–479.

[23] M. Mehrez, G. Mann, R. Gosine, K. Worthmann, T. Faulwasser, Predictive path following of mobile robots without stabilizing terminal constraints, in: 20th IFAC World Congress, no. CONF, 2017.

[24] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, P. Tabuada, Control barrier functions: Theory and applications, in: 2019 18th European Control Conference (ECC), IEEE, 2019, pp. 3420–3431.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th USENIX symposium on operating systems design and implementation (OSDI), 2016, pp. 265–283.

[26] F. Chollet, et al., Keras, https://github.com/fchollet/keras (2015).

[27] Robots: Turtlebot 3, https://robots.ieee.org/robots/turtlebot3/ (2020).

[28] A. Salimi Lafmejani, NMPC-LBF Python codes, https://github.com/asalimil/NMPC-LBF (2022).

[29] L. Grüne, J. Pannek, Nonlinear model predictive control, in: Nonlinear Model Predictive Control, Springer, 2017, pp. 45–69.

[30] S. Dutta, X. Chen, S. Jha, S. Sankaranarayanan, A. Tiwari, Sherlock - a tool for verification of neural network feedback systems: Demo abstract, in: 22nd ACM HSCC, 2019, pp. 262–263.

[31] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, T. T. Johnson, NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems, in: CAV, Vol. 12224 of LNCS, Springer, 2020, pp. 3–17.

[32] J. J. Park, Graceful navigation for mobile robots in dynamic and uncertain environments., Ph.D. thesis (2016).