Prescient Collision-Free Navigation of Mobile Robots with Iterative Multimodal Motion Prediction of Dynamic Obstacles

Ze Zhang, Hadi Hajieghrary, Emmanuel Dean, and Knut Åkesson

***Abstract*—To explore safe interactions between a mobile robot and dynamic obstacles, this paper presents a comprehensive approach to collision-free navigation in dynamic indoor environments. The approach integrates multimodal motion predictions of dynamic obstacles with predictive control for obstacle avoidance. Multimodal Motion Prediction (MMP) is achieved by a deep-learning method that predicts multiple plausible future positions. By repeating the MMP for each time offset in the future, multi-time-step MMPs are obtained. A nonlinear Model Predictive Control (MPC) solver uses the prediction outcomes to achieve collision-free trajectory tracking for the mobile robot. The proposed integration of multimodal motion prediction and trajectory tracking outperforms other non-deep-learning methods in complex scenarios. The approach enables safe interaction between the mobile robot and stochastic dynamic obstacles.**

***Index Terms*—Collision avoidance, autonomous agents, deep learning methods**

I. Introduction

With mobile robots operating alongside human workers in warehouses and industrial settings, their interaction becomes inevitable. Obstacle avoidance for mobile robots is a natural proposition since the emergence of general mobile robots. Static obstacle avoidance is well-developed [1], whereas dynamic obstacle avoidance is still challenging [2], mainly due to the uncertainty of obstacles' future positions. Industrial Automated Guided Vehicles (AGVs) [3], [4] are normally equipped with sensors detecting nearby obstacles. They decelerate and stop if their paths are blocked. Some AGVs [4] can detour from planned routes. However, most detour strategies are passive and make no distinct difference whether the obstacle is static or dynamic. With the emergence of Autonomous Mobile Robots (AMRs) [5], more proactive approaches to dynamic obstacle avoidance became prominent.

Humans are proficient at sidestepping dynamic obstacles due to the capacity to discern the motion intentions of others, anticipate possible scenarios, and initiate evasive maneuvers [3]. Nevertheless, future motions are difficult to predict due to the inherent uncertainty. A simple prediction assumes a motion model of constant velocity or constant turning rate.

Manuscript received: March 6, 2023; Revised: June 2, 2023; Accepted: July 4, 2023.

This paper was recommended for publication by Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by AIMCoR and AIHURO projects. The computations were enabled by resources provided by Chalmers Centre for Computational Science and Engineering (C3SE).

All authors are with Chalmers University of Technology, 41296 Gothenburg, Sweden {zhze, hadiha, deane, knut}@chalmers.se

Digital Object Identifier (DOI): see top of this page.

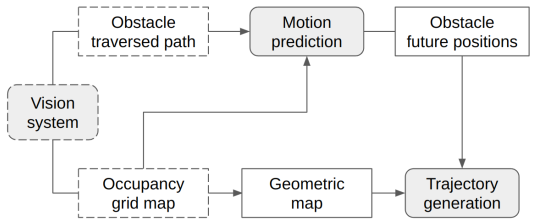

Fig. 1. The proposed pipeline of dynamic obstacle avoidance for mobile robots. The vision system captures the static environment and the motions of dynamic obstacles via deep learning. Gray blocks are system components, and others are transmitted data. Dashed blocks are assumed to be given.

The reality is intricate since the environment constrains how a human moves. Hence, a motion predictor should encode the environmental information and generate multiple estimations of the future position of the target object. The term *multimodal* emphasizes the estimation of multiple alternative future positions, like a pedestrian at an intersection possibly moving straight, turning left, or turning right. In industrial environments, workers normally move in repetitive patterns, which can be exploited to improve the estimation of future positions. After obtaining predictions, a mobile robot can approximate its own future states and steer clear of areas that might be occupied by static or dynamic obstacles. This means the trajectory planner considers both the robot's future states and other dynamic obstacles' possible future positions.

In this paper, we propose the combination of vision-based deep learning for iterative Multimodal Motion Prediction (MMP) with Model Predictive Control (MPC) for trajectory tracking to perform prescient dynamic obstacle avoidance of a mobile robot in factories or warehouses, as shown in Fig. 1. To focus on integrating motion prediction with trajectory tracking, three assumptions are made: (1) The primary sensor is a ceiling-mounted camera system with overlapping views [6]. Feeds from cameras merge into a single bird's-eye-view of the relevant area. (2) An occupancy grid map of the static environment is given. This can be generated using a semantic segmentation neural network as shown in Fig. 2. (3) Dynamic obstacles are captured by the vision system and tracked.

The main contribution of this paper is threefold:

- Integrating of iterative MMP with MPC trajectory tracking to perform collision-free navigation of mobile robots in dynamic and stochastic environments.
- Processing of MMP results to formulate MPC problems.
- Evaluating the proposed integration in robot-warehouse scenarios and providing a comparative analysis.

II. Related Work

Dynamic obstacle avoidance techniques [7] are multifarious with two general classes: *active* and *passive*. Mobile robots with active strategies anticipate potential collisions by considering dynamic obstacles' current and future positions, while robots with passive strategies react only when obstacles are detected in their paths. Traditional motion planning methods [7] can adapt to passive dynamic obstacle avoidance, such as the dynamic window approach and artificial potential field. Besides, trajectory planning methods for dynamic environments, such as the timed-elastic-band [8], can adapt to changing environmental layouts by adjusting the trajectory in real time. However, these approaches are not designed for including MMP in the loop. To design active strategies, an idea is to modify classic motion planning algorithms by considering motions of other agents and obstacles [9]–[11]. Nowadays, more learning-based and optimization-based approaches are proposed owing to the rise of machine learning and increase in computing power, such as reinforcement learning [12], [13], and Model-Predictive Control (MPC) [14], [15]. MPC is an optimization-based approach and can handle nonlinear constraints, which provides a viable strategy for solving the dynamic obstacle avoidance problem.

To achieve a successful proactive dynamic obstacle avoidance, the prerequisite is to *predict the motions* of dynamic obstacles. In earlier studies of dynamic obstacle avoidance, it is assumed that the motions of obstacles are known or modeled by deterministic motion models, such as the constant velocity model [16] and the reciprocal velocity obstacle [17]. To contemplate more complex and uncertain motion patterns, MMP with uncertainty is important. There are increasing studies that use deep learning to make these motion predictions [18]–[20]. The Winner-Takes-All (WTA) loss for multiple hypotheses [18], [21] is a meta-loss for training neural networks. It selectively updates the best hypothesis determined as the closest one to the ground truth based on a specific criterion. A defect is that the WTA loss may discard some hypotheses rather than updating them and the neglected hypotheses cannot be identified. To mitigate this issue, a modified Evolving WTA (EWTA) loss, proposed in [18], updates the k_{top} best hypotheses instead of only the best prediction. The parameter k_{top} decreases according to a predefined schedule during training. This approach is shown to have better performance than WTA but still has the omission of some hypotheses from updates.

In [22], we proposed an Adaptive WTA (AWTA) and a Swarm WTA (SWTA) to mitigate this problem. The AWTA updates hypotheses according to an adaptive range, therefore, more hypotheses are updated. However, the usage of the adaptive range results in a convergence that draws hypotheses to a single point. The SWTA counteracts the convergence by changing the base loss function into the minimal loss among all hypotheses in the range. The AWTA and SWTA are used in this work and are further discussed in Section IV-A. In [18], a mixture density network is used to estimate multimodal probability distributions based on hypotheses generated by the EWTA. However, the generated multimodal probability distributions may have abnormal or redundant components, as discussed in [22]. These abnormal components are not desirable when generating geometric motion predictions as they introduce abnormal dimensions that hinder trajectory tracking and potentially block feasible paths. Thus, we use the WTA with clustering instead.

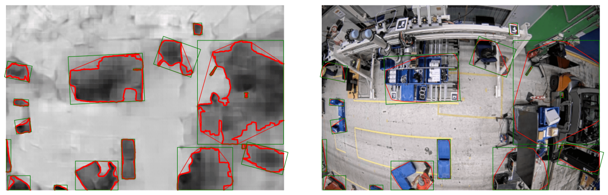
Fig. 2. The occupancy grid map is estimated by BiSeNet [23]. Static obstacles are recognized via contour detection (red curves), from this a convex hull is computed that is then represented by a set of convex polygons (red polygons).

III. Preliminaries

To deploy obstacle avoidance in trajectory tracking, we introduce a receding horizon of the future and check potential collisions at every step in the horizon. Therefore, future positions of dynamic obstacles need to be estimated.

A. Motion Prediction with Uncertainty

In MMP, we use the past trajectory of an object to predict its future positions. Assume that all positions of robots and dynamic obstacles are equidistantly sampled in time. Let $\boldsymbol{p}_k^{(i)} = [x_k^{(i)}, y_k^{(i)}]^\top$ be a position in the Cartesian coordinate system, of obstacle i at time step $k \in \mathbb{Z}$. Then, a *trajectory segment* is a temporal sequence of positions, for $k_{\text{end}} > k_{\text{start}}$,

$$\mathcal{T}_{k_{\text{start}}:k_{\text{end}}}^{(i)} = \langle \boldsymbol{p}_{k_{\text{start}}}^{(i)}, \boldsymbol{p}_{k_{\text{start}}+1}^{(i)}, \dots, \boldsymbol{p}_{k_{\text{end}}}^{(i)} \rangle, \tag{1}$$

which describes the obstacle's motion.

For convenience, a set of non-negative integers, in the closed interval $[a, b]$ and $b > a \geq 0$, is written as $\mathbb{N}_{[a,b]}$. Assuming M alternative predictions are made for a future position τ time steps ahead, a single-time-step MMP (sMMP) is represented by ${}^m\hat{\boldsymbol{p}}_\tau^{(i)}$, where $m \in \mathbb{N}_{[1,M]}$. A multi-time-step MMP (mMMP) includes all future positions $1, \dots, \tau_{\max}$ steps ahead of the current observed position, as in (2).

$${}^M\hat{\mathcal{T}}_{1:\tau_{\max}}^{(i)} = \langle \bigcup_{m \in \mathbb{N}_{[1,M]}} \{{}^m\hat{\boldsymbol{p}}_\tau^{(i)}\} \mid \tau \in \mathbb{N}_{[1,\tau_{\max}]} \rangle. \tag{2}$$

The prediction is based on the past τ_p positions. We call $\tau_{\max}$ the *maximum prediction time offset* and τ_p the *look-back range*. To simplify the notation, and to acknowledge the fact that at time step k the previous τ_p positions are used to estimate the future $\tau_{\max}$ positions, write $\mathcal{T}_{-\tau_p:0}^{(i)}$ to denote the τ_p past positions preceding step k together with the observed position at k and $\mathcal{T}_{1:\tau_{\max}}^{(i)}$ to denote the $\tau_{\max}$ future positions after k.

Commonly, the output of regular neural networks is one estimation of the ground truth. To consider multimodality, a network can generate multiple hypotheses of the future position. In this work, an improved WTA loss is utilized for training networks producing multiple predictions. Given an input $\boldsymbol{x} \in \mathcal{X}$ and its corresponding label $\boldsymbol{y} \in \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are input and label datasets, a neural network $F : \mathcal{X} \to \mathcal{Y}$ mapping from the input to the label, is used to generate Z hypotheses $\{{}^z\boldsymbol{h}\}_{z=1}^Z$. For each hypothesis ${}^z\boldsymbol{h}$, a base-loss ${}^zl$

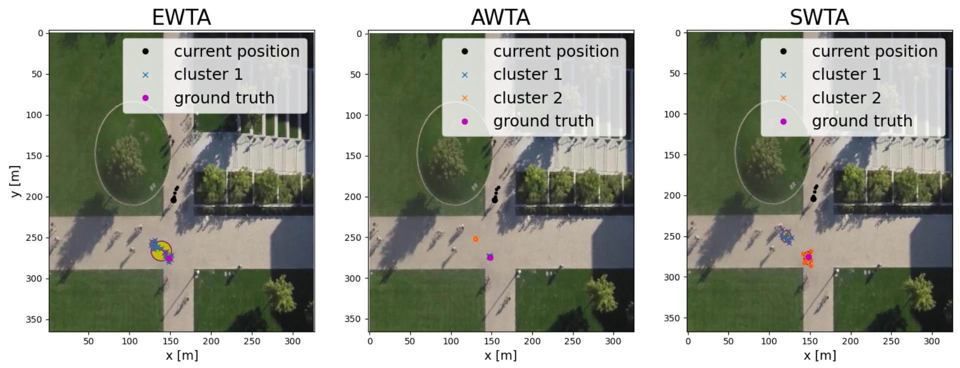

Fig. 3. Comparison between the EWTA, AWTA, and SWTA training results on the Standard Drone Dataset. Large black circles represent the current positions and smaller circles are past positions. The SWTA result produces multimodal prediction clearly and counteracts the clustering effect in AWTA, while the network trained by EWTA can not identify multiple alternatives.

can be computed. The WTA loss is a meta-loss over these Z base-losses as shown in (3), where $\delta(\cdot)$ is the Kronecker delta function and $\mathcal{Z}$ denotes the set $\{1,2,\dots,Z\}$. The WTA loss only updates the winner with the smallest loss.

$$\mathcal{L}_{\text{WTA}} = \sum_{z\in\mathcal{Z}} w_z l(\boldsymbol{y}, {}^z\boldsymbol{h}),\, w_j = \delta(j = \operatorname*{argmin}_{z\in\mathcal{Z}}\ {}^z l). \tag{3}$$

Hypotheses from the network can be regarded as samples from a predicted probability distribution of the future position. If two hypotheses are close enough, they should belong to the same mode. By clustering all hypotheses into different modes, the sMMP ${}^m\hat{\boldsymbol{p}}^{(i)}$ is obtained.

B. Trajectory Tracking Using Model Predictive Control

In trajectory planning, a controller guides a mobile robot to stay within a neighborhood around a reference path, which should be free from obstacles. The controller adjusts the robot's moving direction and may deviate from the reference path to avoid collisions. The generation of collision-free trajectories can be formulated as a receding horizon optimization problem, with the reference deviation error as a loss term, and obstacle avoidance as constraints, as in (4)-(6).

For a mobile robot, let $f : \mathbb{R}^{n_s\times n_u} \to \mathbb{R}^{n_s}$ be its discrete-time nonlinear kinematics or dynamics, where n_s is the dimension of the state and n_u is the dimension of the action, thus $\boldsymbol{s}_{k+1} = f(\boldsymbol{s}_k, \boldsymbol{u}_k)$, where $\boldsymbol{s}_k \in \mathbb{R}^{n_s}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ are the state and input vectors at time k, respectively. Let $\mathcal{O}$ be the set of a-priori known static obstacles and $\mathcal{D}_k$ be the set of dynamic obstacles at time step k. Assuming the finite horizon in the controller is N, the trajectory generation problem is formulated in (4)-(6),

$$\min_{\boldsymbol{u}_{k:k+N-1}} J_{k+N} + \sum_{j=k}^{k+N-1} \left(||\boldsymbol{s}_j - \tilde{\boldsymbol{s}}_j||^2_{\boldsymbol{Q}_s} + ||\boldsymbol{u}_j - \tilde{\boldsymbol{u}}_j||^2_{\boldsymbol{Q}_u}\right), \tag{4}$$

$$\text{s.t.}\quad \boldsymbol{s}_{j+1} = f(\boldsymbol{s}_j, \boldsymbol{u}_j), \tag{5}$$

$$\boldsymbol{p}_j^{(s)} \notin \mathcal{O} \cup \mathcal{D}_j,\, \forall j \in \mathbb{N}_{[k,k+N-1]}, \tag{6}$$

where J_{k+N} is the terminal cost; $\tilde{\boldsymbol{s}}$ and $\tilde{\boldsymbol{u}}$ are reference states and reference actions; $\boldsymbol{Q}_s$ and $\boldsymbol{Q}_u$ are penalizing weights; $\boldsymbol{p}_j^{(s)} = [x_j^{(s)}, y_j^{(s)}]^\top$ is the position of the robot in $\boldsymbol{s}_j$. In this study, a pre-determined reference path $\mathcal{P}_{\text{ref}}$ is provided without considering dynamic obstacles. The reference path is an ordinal list of map points. The reference state $\boldsymbol{s}_j$ is obtained by shifting positions along the path as the robot advances. The reference action $\boldsymbol{u}_j$ is defined by the user.

C. Modeling of Obstacles

Obstacles are described by geometric shapes and obstacle sets are defined by occupied points inside the corresponding shapes. Static obstacles are modeled as polygons represented by sets of inequalities [24]. Let $\mathcal{O} = \cup_i \mathcal{O}^{(i)}$ be the set of static obstacles, where each obstacle set $\mathcal{O}^{(i)}$ is closed and defined by an intersection of some half-spaces. Assuming a static-obstacle set $\mathcal{O}^{(i)}$ has N_E edges, then

$$\mathcal{O}^{(i)} = \{\boldsymbol{p} \in \mathbb{R}^2 \mid \boldsymbol{b}_j^{(i)} - (\boldsymbol{a}_j^{(i)})^\top \boldsymbol{p} > 0,\, \forall j \in \mathbb{N}_{[1,N_E]}\}, \tag{7}$$

where $\boldsymbol{a}_j^{(i)}$ and $\boldsymbol{b}_j^{(i)}$ are coefficients defining a half-space. To indicate if a point $\boldsymbol{p} = [x, y]^\top$ is inside $\mathcal{O}^{(i)}$,

$$\iota_O(\boldsymbol{p} \,|\, \boldsymbol{a}^{(i)}, \boldsymbol{b}^{(i)}) = \prod_{j=1}^{N_E} \max\left\{0, \left[\boldsymbol{b}_j^{(i)} - (\boldsymbol{a}_j^{(i)})^\top \boldsymbol{p}\right]\right\}, \tag{8}$$

where $\boldsymbol{a}^{(i)} = \{\boldsymbol{a}_j^{(i)}, j \in \mathbb{N}_{[1,N_E]}\}$ and $\boldsymbol{b}^{(i)} = \{\boldsymbol{b}_j^{(i)}, j \in \mathbb{N}_{[1,N_E]}\}$. The indicator is zero outside the polygon and positive inside the polygon, and with a higher value if the point is closer to the center of the polygon.

Dynamic obstacles have uncertain future positions, both concerning the multimodality and the local uncertainty within each mode. We use two-dimensional ellipses to represent the different modes of future positions. The time step k is omitted in the following definitions for brevity. For an ellipse centered at $\boldsymbol{p}_\mu = [x_\mu, y_\mu]^\top$ with axes $\boldsymbol{\sigma} = [\sigma_x, \sigma_y]^\top$, a variable ι_D indicating if a point $\boldsymbol{p} = [x, y]^\top$ is inside the ellipse can be defined as (9). The indicator is zero outside the ellipse and positive inside the ellipse, and with a higher value if the point is closer to the center of the ellipse.

$$\iota_D(\boldsymbol{p} \,|\, \boldsymbol{p}_\mu, \boldsymbol{\sigma}) = \max\left\{0, \left[1 - (\frac{x - x_\mu}{\sigma_x})^2 - (\frac{y - y_\mu}{\sigma_y})^2\right]\right\}. \tag{9}$$

Then, the set $\mathcal{D}$ occupied by N_d ellipses can be defined as:

$$\mathcal{D} = \{\boldsymbol{p} \in \mathbb{R}^2 \,|\, \exists\, i \in \mathbb{N}_{[1,N_d]},\, \iota_D(\boldsymbol{p}|\boldsymbol{p}_\mu^{(i)}, \boldsymbol{\sigma}^{(i)}) > 0\}. \tag{10}$$

IV. Collision-free trajectory tracking in dynamic environments

Given (i) static obstacles represented by convex polygons; (ii) MMPs of dynamic obstacles; (iii) a reference path $\mathcal{P}_{\text{ref}}$ without considering dynamic obstacles; (iv) kinematic and dynamic constraints of a mobile robot, the goal is to compute a sequence of control actions that avoid obstacles while staying near the provided reference path.

The motion prediction neural network is trained offline using the SWTA loss. The input $\boldsymbol{x}$ to the network is the occupancy grid map and location masks of dynamic obstacles, and the output is multiple hypotheses of the future position of the dynamic obstacle. During runtime, the neural network infers future positions of a dynamic obstacle for multiple time steps. The network handles one dynamic obstacle at a time. Formally, at time step k, for dynamic obstacle i, we want to predict M future trajectories ${}^{M}\hat{\mathcal{T}}^{(i)}_{1:\tau_{\max}}$ for each step between 1 and $\tau_{\max}$, based on its observed trajectory segment $\mathcal{T}^{(i)}_{-\tau_p:0}$. The predictions of dynamic obstacles are used to generate a sequence of control actions to avoid collisions through a refined MPC formulation, which is described in Section IV-B.

A. Iterative Multimodal Motion Prediction

A challenge of having multiple hypotheses $\{{}^{z}\boldsymbol{h}\}_{z=1}^{Z}$ is to determine which ones to update during training, as updating all hypotheses yields the same behavior as having one hypothesis. The WTA approach handles this by updating the hypothesis with the lowest loss and suffers from abandoned hypotheses. To address this, the EWTA method updates a varying set of hypotheses but still leaves some hypotheses abandoned near equilibrium points, i.e., average positions of ground truths. In [22], the SWTA was proposed to further mitigate this problem.

The training of the proposed MMP approach requires three steps: the evolving step (EWTA), the adaptive step (AWTA), and the swarm step (SWTA). In the first phase, the EWTA loss is used and some hypotheses are close to the ground truths while others are close to the equilibrium points. Next, the AWTA loss is used to continue training. The idea of the AWTA is to find an adaptive range r_{adp} as in (11), defined with a hyperparameter α. All hypotheses within r_{adp} will be updated. In the AWTA defined in (12), all hypotheses with a loss smaller than r_{adp} will be updated during training and gradually converge. With a suitable α, the abandoned hypotheses are updated and converge to other hypotheses close to them. A side-effect of this convergence is that the local uncertainty of each mode diminishes, which is inherently important for trajectory planning. The SWTA loss, depicted in (13), is employed to counteract the convergence caused by the AWTA. It achieves this by giving greater updates to hypotheses within each cluster that are closer to the ground truth, causing them to diverge from the others, thereby representing the local uncertainty. The difference in behavior between the EWTA, AWTA (after EWTA), and SWTA (after EWTA and AWTA) on the Standard Drone Dataset [25] are shown in Fig. 3.

In [22], an iterative manner is proposed to generate mMMP, which appends a prediction time offset channel to the original input $\boldsymbol{x}$. However, including all offset values in the training dataset enlarges its size and slows down the training. Instead, we randomly select offset values from a range $\mathbb{N}_{[1,\tau_{\max}]}$ for each input trajectory. We have observed that the random selection does not affect the MMP performance and significantly decreases the training time.

$$r_{\text{adp}} = \min_{z\in\mathcal{Z}}({}^{z}l) + \alpha\left(\max_{z\in\mathcal{Z}}({}^{z}l) - \min_{z\in\mathcal{Z}}({}^{z}l)\right), \tag{11}$$

$$\mathcal{L}_{\text{AWTA}} = \sum_{z\in\mathcal{Z}} w'_z l(\boldsymbol{y}, {}^{z}\boldsymbol{h}),\; w'_z = \delta({}^{z}l \leq r_{\text{adp}}), \tag{12}$$

$$\mathcal{L}_{\text{SWTA}} = \sum_{z\in\mathcal{Z}} w'_z \min_{i\in\mathcal{Z}} l(\boldsymbol{y}, {}^{i}\boldsymbol{h}),\; w'_z = \delta({}^{z}l \leq r_{\text{adp}}). \tag{13}$$

During inference, each observed trajectory of a dynamic obstacle is transferred into a stack of masks $\langle \boldsymbol{B}_{-\tau_p}, \ldots, \boldsymbol{B}_0 \rangle$, and each mask indicates the location of the obstacle at that time. The occupancy grid map and the time offset channel are appended to the stack of location masks. Specifically, the location mask $\boldsymbol{B}$ is an image of the same dimension as the occupancy grid map and is defined using the bivariate Gaussian distribution with a zero correlation as in (14), where b_{ij} is the pixel value of the i-th row and the j-th column in $\boldsymbol{B}$, u_μ and v_μ are the location coordinates of the dynamic obstacle on the image axes, σ_u and σ_v are preset.

$$b_{ij} = \frac{1}{2\pi\sigma_u\sigma_v}\exp\left(-\frac{1}{2}\left[\left(\frac{j-u_\mu}{\sigma_u}\right)^2 + \left(\frac{i-v_\mu}{\sigma_v}\right)^2\right]\right). \tag{14}$$

The bivariate Gaussian distribution is used to indicate the location of the object and also provides additional information about the distance from a pixel to the ground truth. The time offset channel has the same size as the occupancy grid map, and each entry contains the value of the offset τ.

B. Prescient Dynamic Obstacle Avoidance

The mMMP produces $Z \cdot \tau_{\max}$ hypotheses as in (15), with Z hypotheses for each offset τ of the dynamic obstacle i.

$${}^{Z}\hat{\mathcal{T}}^{(i)}_{1:\tau_{\max}} = \langle \bigcup_{j\in\mathbb{N}_{[1,Z]}} \{{}^{j}\hat{\boldsymbol{p}}^{(i)}_{\tau}\} \mid \tau \in \mathbb{N}_{[1,\tau_{\max}]} \rangle. \tag{15}$$

After training, during runtime, at each time step, the Clustering and Gaussian Fitting (CGF) approach, outlined in [22], is used to cluster all hypotheses related to dynamic obstacles based on their spatial proximity. This clustering operation yields a total of M' distinct hypotheses, as it merges hypotheses associated with multiple dynamic obstacles. This merging is beneficial as it reduces the possibility of obtaining overlapping obstacles.

Given MMPs, the control actions of the mobile robot are expected to be affected. The MPC formulation accounts for both static and dynamic obstacles by including them as constraints. MPC solvers are typically designed to handle both hard and soft constraints. Hard constraints must be satisfied invariably, whereas soft constraints are designed with an expectation of fulfillment, but with room for violation. We use both of them to regulate the robot's behavior. For static obstacles, soft and hard constraints are formulated in (19) via the indicator as in (8). All static obstacles are inflated by the size of the robot for both soft and hard constraints. Similarly, for dynamic obstacles, typically humans, soft and hard constraints are formulated in

(20) via the indicator as in (9). However, due to the uncertainty of MMPs, three different kinds of margins are used for safer navigation. In hard constraints, all dynamic obstacles and their motion predictions are expanded by the size of the robot. On the one hand, the soft constraints for dynamic obstacles at the current time handle an extra margin r_{extra}, which is composed of the safety margin r_{saf} and the social margin r_{soc}. On the other hand, soft constraints for motion predictions of dynamic obstacles handle the extra margin equal to r_{saf}. The purpose of this choice is to provide better feasibility and more safe guarantee for dynamic obstacles.

In the implementation, we use the OpEn engine [26] to solve the MPC problem, which can cope with both hard and soft constraints. The details are discussed in the next section. The use of soft constraints provides several benefits, their flexibility allows for separate treatment of different types of obstacles, and the tolerance to deprioritize certain soft constraints in favor of more crucial requirements. In situations where the mobile robot's path is obstructed by obstacles, yet feasible solutions are easily accessible, soft obstacle-avoidance constraints are often sufficient to prevent collisions. Conversely, when a feasible solution is elusive or non-existent, the robot may violate soft constraints, which may subsequently result in collisions. Therefore, hard obstacle-avoidance constraints become indispensable.

The objective function comprises reference deviation terms as (16)-(18), and soft obstacle avoidance terms as (19) and (20) at time step k, where N_o and N_d are the numbers of static and dynamic obstacles with $\boldsymbol{Q}_{\mathcal{O}}$ and $\boldsymbol{Q}_{\mathcal{D}}$ being the penalty weights. In (20), $\boldsymbol{p}_k^{(i)}$ and $\boldsymbol{\sigma'}_k^{(i)}$ represent the position and axes of the i-th obstacle, and $\boldsymbol{\sigma'}_k^{(i)} = \boldsymbol{\sigma}_k^{(i)} + r_{\text{extra}}$ where $\boldsymbol{\sigma}_k^{(i)}$ is the original axes calculated using the CGF [22] and r_{extra} is the extra margin.

$$J_k^{(s)} = J_s(\boldsymbol{s}_k, \tilde{\boldsymbol{s}}_k) = ||\boldsymbol{s}_k - \tilde{\boldsymbol{s}}_k||^2_{\boldsymbol{Q}_s}, \tag{16}$$

$$J_k^{(u)} = J_u(\boldsymbol{u}_k, \tilde{\boldsymbol{u}}_k) = ||\boldsymbol{u}_k - \tilde{\boldsymbol{u}}_k||^2_{\boldsymbol{Q}_u}, \tag{17}$$

$$J_k^{(a)} = J_a(\boldsymbol{u}_k, \boldsymbol{u}_{k-1}) = ||\boldsymbol{u}_k - \boldsymbol{u}_{k-1}||^2_{\boldsymbol{Q}_a}, \tag{18}$$

$$J_{\mathcal{O}}(\boldsymbol{s}_k) = \sum_{i=1}^{N_o} ||\iota_O(\boldsymbol{p}_k^{(s)} \mid \boldsymbol{a}^{(i)}, \boldsymbol{b}^{(i)})||^2_{\boldsymbol{Q}_{\mathcal{O}}}, \tag{19}$$

$$J_{\mathcal{D}}(\boldsymbol{s}_k) = \sum_{i=1}^{N_d} ||\iota_D(\boldsymbol{p}_k^{(s)} \mid \boldsymbol{p}_k^{(i)}, \boldsymbol{\sigma'}_k^{(i)})||^2_{\boldsymbol{Q}_{\mathcal{D}}}. \tag{20}$$

To make the objective function more compact, let $J_R(k) = J_k^{(s)} + J_k^{(u)} + J_k^{(a)}$. The full MPC formation, using sampling time Δt_s and $\Delta \boldsymbol{u}_j = \boldsymbol{u}_j - \boldsymbol{u}_{j-1}$ (at $k=0$, $\boldsymbol{u}_{k-1} = \boldsymbol{0}$) is

$$\min_{\boldsymbol{u}_{k:k+N-1}} \quad J_N^{(s)} + \sum_{j=k}^{k+N-1} \left[J_R(j) + J_{\mathcal{O}}(\boldsymbol{s}_j) + J_{\mathcal{D}}(\boldsymbol{s}_j)\right], \tag{21}$$

$$\text{s.t.} \quad \boldsymbol{s}_{j+1} = f(\boldsymbol{s}_j, \boldsymbol{u}_j), \tag{22}$$

$$\boldsymbol{u}_j \in [\boldsymbol{u}_{min}, \boldsymbol{u}_{max}], \tag{23}$$

$$\frac{\Delta \boldsymbol{u}_j}{\Delta t_s} \in [\dot{\boldsymbol{u}}_{min}, \dot{\boldsymbol{u}}_{max}], \tag{24}$$

$$\boldsymbol{p}_j^{(s)} \notin \mathcal{O}, \tag{25}$$

$$\boldsymbol{p}_j^{(s)} \notin \mathcal{D}_j, \forall j \in \mathbb{N}_{[k,k+N-1]}. \tag{26}$$

Compared to (4), three terms are added in (21), which are the penalty on the acceleration J_a and the soft constraints of the static and dynamic obstacle avoidance $J_{\mathcal{O}}$ and $J_{\mathcal{D}}$. During runtime, only the first generated action is used at each step. In the implementation, we set $N = 20$ and $J_N^{(s)} = 0$.

V. Implementation

The presented approach has been implemented in Python[1] and extended to ROS. The occupancy grid map is assumed to be given. As illustrated in Fig. 2, the occupancy grid map is transformed into a geometric map, where obstacles are recognized and simplified using contour detection and convex polygons. At runtime, the motion predictor gives mMMPs of dynamic obstacle positions according to their past positions. The MPC controller commands the action of the mobile robot. The proposed pipeline is shown in Fig. 1.

The mobile robot was implemented using a unicycle model (27) assuming no slip of the wheels, with the state $\boldsymbol{s}_k = [x_k^{(s)}, y_k^{(s)}, \theta_k^{(s)}]^\top$ and the action $\boldsymbol{u}_k = [u_{v,k}, u_{\omega,k}]^\top$, specifically, $\boldsymbol{p}_j^{(s)} = [x_k^{(s)}, y_k^{(s)}]^\top$ indicates the x and y position of the robot, $\theta_k^{(s)}$ is the heading of the robot, $u_{v,k}$ and $u_{\omega,k}$ are linear speed and angular velocity of the robot respectively,

$$\boldsymbol{s}_{k+1} = f(\boldsymbol{s}_k, \boldsymbol{u}_k) = \begin{bmatrix} x_k^{(s)} + \Delta t_s u_{v,k}\cos(\theta_k^{(s)}) \\ y_k^{(s)} + \Delta t_s u_{v,k}\sin(\theta_k^{(s)}) \\ \theta_k^{(s)} + \Delta t_s u_{\omega,k} \end{bmatrix}. \tag{27}$$

A. Implementation of the Control Strategy

In the evaluation, the OpEn engine [26] is used to solve nonlinear MPC optimization problems, which can handle inequality constraints using the penalty method. The penalty method treats constraints as extra terms multiplied by penalty coefficients in the objective, which transfers the original constrained problem into an unconstrained problem. Given the uncertainty of the constraints, the way PANOC handles the constraints by lifting them into the objective function is a viable strategy for this application. Since the penalty method requires numerical iterations, a maximum solving time is set to restrict the number of iteration loops.

B. Implementation of the Motion Prediction Strategy

The input to the neural network is a stack of images containing the environment, the observed positions of the target object, and the prediction time offset channel. The backbone is ResNet34 [27] and the output layer is a linear layer with $Z \times D_o$ output units, where Z is the number of hypotheses and D_o is the dimension of each hypothesis. In this case, each hypothesis is a guess of a two-dimensional location, therefore, $D_o = 2$. Hyperparameters of the SWTA loss and training profile are the same as [22].

The neural network is trained on a simulated dataset obtained from the warehouse scene as shown in Fig. 7. Pedestrians are simulated to walk on pre-defined paths around the whole map with random noise on their velocities. There are 580 trajectories collected for training, which gives 265270 training data samples. During the training, the first 14 epochs

[1]Code is available: https://github.com/Woodenonez/DyObAv_MPCnWTA_Warehouse

TABLE I
EVALUATION RESULTS (AVERAGE OVER 100 RUNS). STARRED METHODS ARE DEEP-LEARNING AIDED. METHODS IN BOLD FONT ARE PROPOSED BY US. METRICS IN BOLD FONT ARE THE BEST RESULTS.

Scenario	Method	Smoothness		Clearance (m)		Deviation (m)			Solving time (sec)		Success (%)
		linear	angular	static	dynamic	mean	std	max	mean	max	
Scenario 1	**MPC-WTA***	**0.028**	**0.036**	0.511	0.750	0.429	0.261	0.738	0.061	0.517	**95**
	MPC-WTA-F*	0.030	0.039	0.501	0.786	0.434	0.267	1.095	**0.050**	**0.125**	94
	MPC-CV	0.065	0.075	0.837	0.304	0.270	0.280	1.055	0.191	0.518	22
	DWA	0.041	0.083	0.993	0.279	**0.142**	**0.074**	**0.456**	0.173	0.311	46
	DWA-CV	0.045	0.094	0.989	0.308	0.204	0.142	0.704	0.195	0.413	48
Scenario 2	**MPC-WTA***	0.034	0.044	0.577	0.941	0.211	0.143	0.840	**0.051**	0.517	**94**
	MPC-WTA-F*	0.046	0.042	0.574	0.863	0.224	0.160	1.064	0.052	**0.121**	81
	MPC-CV	**0.031**	0.054	0.544	0.686	0.186	0.126	0.564	0.108	0.516	54
	DWA	0.039	**0.033**	0.632	0.350	**0.183**	**0.102**	0.400	0.206	0.305	58
	DWA-CV	0.042	0.034	0.651	0.298	0.205	0.109	**0.398**	0.207	0.323	62

are trained via the EWTA loss with a decaying k_{top}, and the next three epochs are trained via the AWTA loss, followed by another three epochs trained via the SWTA loss.

C. Integration into ROS

The control and motion forecasting approaches have been integrated into ROS. The proposed method is simulated within ROS and visualized using Gazebo and RViz. The proposed approach plans the trajectory for a mobile robot within the setting of a warehouse, where the robot is operating alongside human workers. This is discussed further in the next section.

VI. EVALUATION

In this section, the proposed method is evaluated both quantitatively and qualitatively. The quantitative analysis considers two aspects: tracking performance and real-time performance. We compare the proposed method against other popular methods for trajectory planning. Additionally, the proposed method is implemented in ROS and validated via Gazebo.

A. Tracking performance

The following metrics are used to evaluate trajectory tracking methods:

- Success rate. A successful run is defined as when the robot traverses the map without any collisions and reaches the goal before a time-out.
- Smoothness (of action). In most cases, the robot is expected to execute smooth movements. The action smoothness is measured as the second derivative of the action, commonly referred to as the *jerk*.
- Clearance. Clearance refers to the minimal distance to obstacles during a trip. We distinguish between clearance from static obstacles and dynamic obstacles. The underlying idea is that the clearance should be large unless the reference path specifically instructs the robot to stay in close proximity to obstacles.
- Deviation. Given that a reference path is provided, the robot should follow it closely, deviating only when necessary to avoid obstacles.

We compare four methods: *MPC-WTA** (ours), *MPC-WTA-F** (ours, a faster version by constraining the solving time), *MPC-CV* (the motion prediction is offered by constant velocity models with Kalman filters, similar to [15]), *DWA* (the

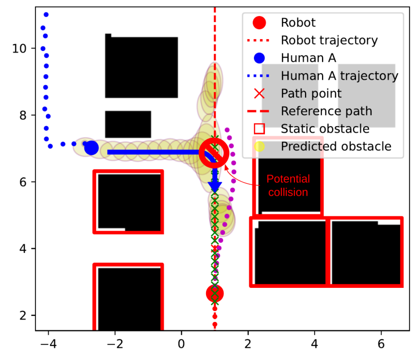

Fig. 4. Illustration of scenario 1 using MPC-WTA. The arrow indicates the movement patterns of the human.

dynamic window approach [7]), and *DWA-CV* (the dynamic window approach with constant velocity motion prediction models, similar to [10]). Note that we do not compare with the MPC method without motion predictions because it generally cannot avoid dynamic obstacles. In order to track the reference path, compared to the original DWA, an extra cost measuring the distance from the candidate trajectory to the reference trajectory is added in the evaluated *DWA*. For the DWA, the linear speed resolution is 0.1 m/s and the angular speed resolution is 0.1 rad/s, which gives 40 candidates. Using the double resolution does not increase the success rate significantly.

All the methods are evaluated in two scenarios in the warehouse environment and the final results are obtained over 100 runs. The reason to have multiple runs is that the simulated agents have stochastic behaviors as shown in Fig. 4 and 5. In the first scenario, as in Fig. 4, a pedestrian and a robot are approaching an intersection. The pedestrian will turn right and potentially collide with the robot. The second scenario, as illustrated in Fig. 5, is a pedestrian walking in parallel with the robot and making a sudden turn at the crossing. As shown in Table I, the proposed approach has the highest success rate and the best comprehensive performance. The evaluation shows the importance of considering MMPs in dynamic obstacle avoidance problems. In comparison, the DWA has less deviation, which means it is not good at dealing with unexpected behaviors of dynamic obstacles.

We also evaluate the influence of different maximum solving times on the performance. In *MPC-WTA**, the maximum solving time is 0.5 seconds while in *MPC-WTA-F** it is 0.1 seconds, which means *MPC-WTA-F** is more likely to give

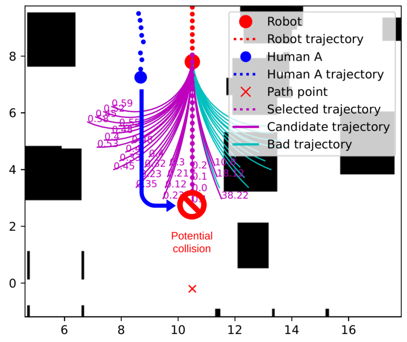

Fig. 5. Illustration of scenario 2 using the DWA. The arrow indicates the movement pattern of the human.

an infeasible solution. In Scenario 1, there is a relatively easy solution where the controller commands the robot to steer right to avoid the potential collision. On the contrary, in Scenario 2, the MMP of the pedestrian completely blocks the robot resulting in no feasible solution to detour but to stop. Thus, in Scenario 1, these two methods have a very close success rate, but *MPC-WTA-F** has a distinctly lower success rate in the second scenario. The solving time in the evaluation is discussed further in the following sub-section.

B. Real-time aspects

For real-time performance, three processes need to be analyzed, including visual sensor measurement, neural network inference, and MPC optimization. In Table I, the solving time only includes the decision-making process without the motion prediction part. The sensor measurement and neural network inference normally cost constant time. Since the sensor measurement is out of the scope of this paper, only the inference time is analyzed. On NVIDIA GTX 1650 Max-Q, the inference time is about 0.3 seconds per object. Note that the deep learning process is not optimized and runs on a low-performance GPU. After optimizing and using better GPUs, this process can be much faster. Meanwhile, in practice, the number of objects should be limited. Only objects close to the robot are considered.

The MPC optimization latency depends on the solving loop. As in TABLE I, on Intel i7-9750H, the solving time is around 0.05 seconds for no obstacle, and may increase if obstacles block the way. To restrict the solving time, a timeout can be added. However, in the penalty method, when the maximum solving time is reached and the solver stops, it doesn't guarantee a feasible solution. To eliminate the risk of collisions, an extra monitor on the loss function can be added such that the robot will stop when the objective value in the MPC is higher than a threshold. Meanwhile, a local sensor/brake system should be used in a physical implementation.

C. Real-world simulation

Apart from the evaluation under Python simulation, the proposed approach is also implemented in ROS for real-world simulation. As shown in Fig. 6, with the help of MMP (yellow ellipses in the Python simulation and purple dots in the ROS simulation), the mobile robot can take evasive action in advance.

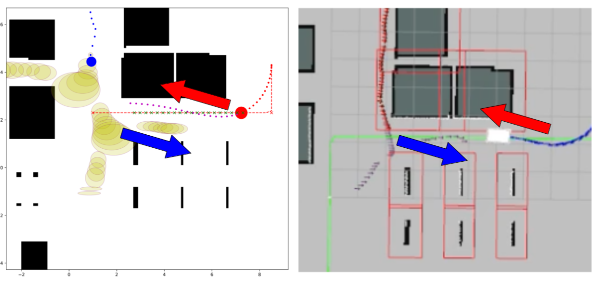

Fig. 6. Illustration of a similar scenario simulated in Python (left) and ROS/RViz (right). The scenario depicts the robot trying to avoid a pedestrian before the pedestrian moves toward the robot. The red arrow indicates the motion direction of the robot (red circle in Python, white rectangle in ROS), while the blue arrow indicates the predicted motion intention of the pedestrian (blue circle in Python, red dot in ROS).

The real-world simulation is illustrated in Fig. 7 which shows a scenario where a human and a robot are moving toward each other. The robot considers all the possible routes the human can take and maneuvers to reduce the possibility of collision. In the specific incident depicted in Fig. 7, the robot estimates several possibilities for the future trajectory of the human based on the map's topology and chooses to steer left until the probability that the human will turn into the left corridor is high. The complete ROS simulation can be found inside the repository mentioned in the last section.

We compare the proposed MPC-WTA method to the Timed-Elastic-Band (TEB) method [8]. In this method, the initial trajectory generated by a global planner is occasionally optimized during runtime to minimize the time-optimal objective, separation from an unforeseen obstacle, and compliance with the kinodynamic constraints of the robot such as maximum velocities and accelerations. This evaluation is implemented in ROS. However, when the robot encounters highly dynamic obstacles such as humans, the TEB method lacks agility in responding to changes in the obstacle's trajectory. This limitation is the result of the compromise between the length of the prediction horizon and the frequency of re-calculation of the trajectory. To improve the robot's ability to react swiftly to avoid a pedestrian, the prediction time horizon should be reduced; this comes at the cost of more frequent optimization. To gain a comparable performance in terms of clearance and smoothness, we adjust the prediction horizon of TEB to achieve a similar result with MPC-WTA. The average solving time of TEB is 0.726 seconds, which is higher than our method with the average solving time of 0.412 seconds.

VII. Conclusion and Future Work

In this article, we proposed a comprehensive dynamic obstacle avoidance solution for mobile robots based on a complete pipeline from a vision system to actual action command for mobile robots. In addition to the high-level pipeline, we integrated a deep learning method for multimodal motion prediction with MPC trajectory generators. The approach is shown to outperform other non-deep-learning approaches and is also evaluated in Python using ROS under warehouse scenes and shows feasibility and safety.

Future work would focus on three aspects: improving the architecture of the neural network for motion prediction to in-

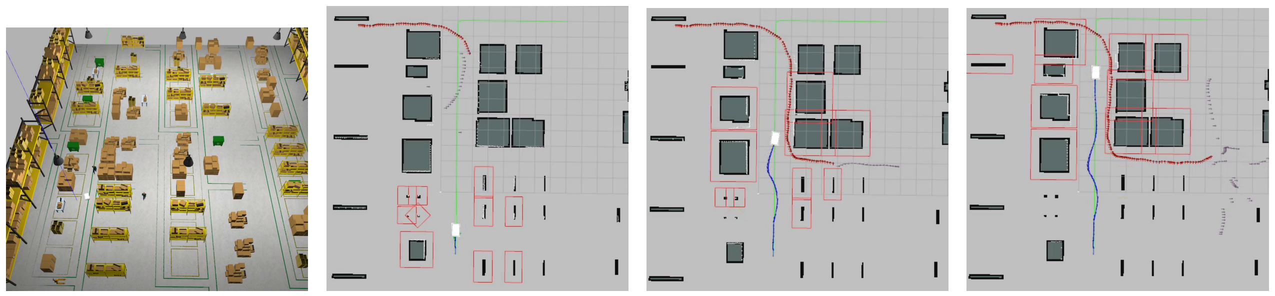

Fig. 7. (a) Gazebo simulation of a Mobile Industrial Robot MiR100 in a warehouse. (b-d) ROS visualization (RViz) of the reference path the mobile robot is trying to follow (in green), the trajectory of the human actor (in red), the predicted future trajectory of the human (in purple), and the actual trajectory of the mobile robot (in blue). The obstacles detected by the robot are inflated (illustrated in red) proportional to the size of the robot. Figures b-d illustrates how the robot predicts the future trajectory of the actor and plans its trajectory accordingly. Full videos: https://shorturl.at/vKV08.

crease the prediction accuracy and shorten the inference time; modifying the reference generator and the MPC formulation to achieve fast convergence in the optimization step; studying the influence of delay from the vision system and how to handle the always present delays. Finally, we plan to evaluate the approach in a physical scenario for a fleet of mobile robots.

References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots (2nd Edition)*. England: MIT Press, 2011.
[2] S. G. Tzafestas, "Mobile robot control and navigation: A global overview," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 1, pp. 35–58, 2018.
[3] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *Journal of Manufacturing Systems*, vol. 54, pp. 152–173, 2020.
[4] V. Digani, F. Caramaschi, L. Sabattini, C. Secchi, and C. Fantuzzi, "Obstacle avoidance for industrial AGVs," in *ICCP*. IEEE, 2014, pp. 227–232.
[5] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020.
[6] Z. Zhang, E. Dean, Y. Karayiannidis, and K. Åkesson, "Motion prediction based on multiple futures for dynamic obstacle avoidance of mobile robots," in *CASE*. IEEE, 2021, pp. 475–481.
[7] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.
[8] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 European Control Conference (ECC)*, 2015, pp. 3352–3357.
[9] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *ICRA*. IEEE, 2007, pp. 1986–1991.
[10] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *ICRA*. IEEE, 2019, pp. 8620–8626.
[11] C. Qixin, H. Yanwen, and Z. Jingliang, "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot," in *IROS*. IEEE/RSJ, 2006, pp. 3331–3336.
[12] L. Huang, H. Qu, M. Fu, and W. Deng, "Reinforcement learning for mobile robot obstacle avoidance under dynamic environments," in *PRICAI: Trends in Artificial Intelligence*. Springer, 2018.
[13] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.
[14] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *RA-L*, vol. 4, no. 4, pp. 4459–4466, 2019.
[15] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *RA-L*, vol. 5, no. 4, pp. 6001–6008, 2020.
[16] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *ICCV*. IEEE/CVF, 2009, pp. 261–268.
[17] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *ICRA*. IEEE, 2008, pp. 1928–1935.
[18] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *CVPR*. IEEE/CVF, 2019.
[19] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," in *ICCV*. IEEE/CVF, 2021.
[20] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann, "The garden of forking paths: Towards multi-future trajectory prediction," in *CVPR*. IEEE/CVF, 2020.
[21] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *ICCV*. IEEE/CVF, 2017, pp. 3611–3620.
[22] Z. Zhang, E. Dean, Y. Karayiannidis, and K. Åkesson, "Multimodal motion prediction based on adaptive and swarm sampling loss functions for reactive mobile robots," in *CASE*. IEEE, 2022, pp. 1110–1115.
[23] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *ECCV*. CVF, 2018, pp. 325–341.
[24] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *ECC*, 2018, pp. 1523–1528.
[25] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *ECCV*. CVF, 2016.
[26] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," in *IFAC World Congress*, 2020.
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*. IEEE/CVF, 2016, pp. 770–778.