2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Safe Robot Navigation in a Crowd Combining NMPC and Control Barrier Functions

Veronica Vulcano, Spyridon G. Tarantos, Paolo Ferrari, Giuseppe Oriolo

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9993397

***Abstract*— We propose a sensor-based scheme for safe robot navigation in a crowd of moving humans. It consists of two modules, i.e., the crowd prediction and motion generation module, which run sequentially during every sampling interval. Using information acquired online by an on-board sensor, the crowd prediction module foresees the future motion of the humans in the robot surroundings. Based on such prediction, the motion generation module produces feasible commands to safely drive the robot among the humans by combining a nonlinear Model Predictive Control (NMPC) algorithm with collision avoidance constraints formulated via discrete-time Control Barrier Functions (CBFs). We show the effectiveness of the proposed approach via simulations obtained in CoppeliaSim on the Pioneer 3-DX mobile robot in scenarios of different complexity.**

I. Introduction

Mobile robots used in service applications are required to perform tasks – such as parcel delivery, patrolling, cleaning and many others – in environments that are typically crowded by moving humans. To successfully accomplish these tasks, robots must be able to safely navigate among humans, which poses as critical requirement the avoidance of collisions with them. Fig. 1 illustrates an example of such scenario.

A variety of navigation methods in dynamic environments exists in the literature, such as the artificial potential fields method [1], the dynamic window approach [2], online versions of randomized planners [3], and techniques based on the concept of velocity obstacles [4]. The latter category also includes methods specifically designed for crowded environments (see [5], [6]).

An appealing alternative to tackle the safe navigation problem in dynamic environments consists in using Model Predictive Control (MPC), which solves at each control cycle an Optimal Control Problem (OCP) throughout a finite horizon. MPC is indeed ideal thanks to its prediction capabilities, the possibility of including constraints, and the recent availability of efficient tools (such as `acados` [7]) for solving the OCP in real-time, even in case of nonlinear MPC (NMPC). For these reasons, MPC has been adopted in many methods for safe navigation. Methods proposed in the context of structured environments leverage these features to formulate simple bounds on the robot position that guarantee collision avoidance [8], [9]. In unstructured environments, sensor-based methods like [10] rely on decomposition of the perception area in multiple convex obstacle-free regions which, under the assumption that all obstacles

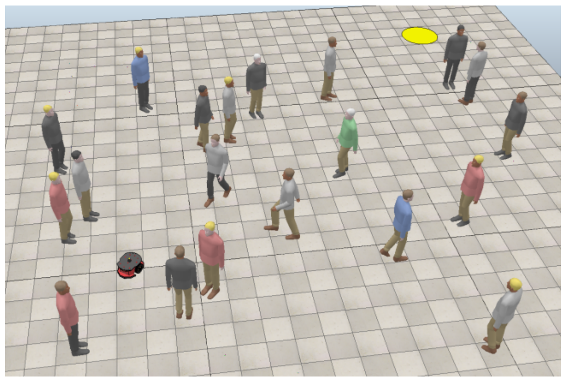
Fig. 1. An instance of the considered problem. The robot must safely navigate in the crowd of moving humans to reach the goal region (in yellow).

are stationary, confine the solution of the OCP. Other, more general, methods impose full-fledged collision avoidance constraints, by enveloping the robot and/or the obstacles (see, for example, [11], [12]) with ellipsoids that are possibly enlarged throughout the prediction horizon. Similar strategies have been adopted also in the context of safe navigation for nonholonomic vehicles [13] and humanoid robots [14] in environments crowded by humans, which are approximated as disc-shaped obstacles.

All the aforementioned MPC-based approaches include collision avoidance constraints that depend only on information about the distance from the obstacles. A constraint of this kind is actually active (i.e., it influences the OCP solution) only when an obstacle is in the range of the prediction horizon. As a consequence, the robot reacts to the presence of the obstacle only in its proximity, which is clearly dangerous especially in case of moving obstacles. To mitigate this issue, a straightforward option consists in using a long prediction horizon, at the expense of dramatically increasing the time needed to solve the OCP. Other approaches use collision avoidance constraints that consider, in addition to the distance information, the robot-obstacle relative velocities and the robot actuation capabilities [15]. Recently, Control Barrier Functions (CBFs) [16] have been introduced as a tool to enforce forward invariance of a safe set, which in the context of safe navigation is the set of robot states where a safety clearance from the obstacles is guaranteed. Only few works (e.g., [17], [18], [19]) that include collision avoidance constraints formulated via (continuous or discrete-time) CBFs in an MPC scheme have been introduced. A constraint of this kind considers not only the distance from the obstacle, but also the rate at which the robot approaches

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy. E-mail: {tarantos,ferrari,oriolo}@diag.uniroma1.it.

it. As a result, the MPC exhibits the remarkable ability of initiating avoidance maneuvers even when the obstacle is not in the range of the prediction horizon, which can therefore be chosen relatively short.

In order to endow the robot with full autonomy, any safe navigation framework invariably needs a method for predicting the future motions of the surrounding humans based on sensory information. In particular, this is essential to fully exploit the prediction capabilities of an MPC scheme. Existing human prediction algorithms are based on either social-force (e.g., [20]) or data-driven models (e.g., [21]). With both approaches, a training phase is required; this makes their generalization often difficult. Moreover, learned models are typically not robust to sensor noise.

In this paper, we propose a sensor-based scheme for safe robot navigation in a crowd. It consists of two modules, i.e., the crowd prediction and motion generation module, which run sequentially during every sampling interval. Using information acquired during motion by an on-board sensor, the crowd prediction module foresees the future motion of the humans in the robot surroundings. Based on such prediction, the motion generation module produces feasible commands to safely drive the robot among the humans. The crowd prediction module relies on a simple, yet effective, technique based on Kalman filters (KFs), while the motion generation module combines a real-time NMPC algorithm and CBF-based collision avoidance constraints. To our knowledge, this is the first work in which NMPC is combined with discrete-time CBFs to devise a complete framework (including crowd prediction) for safe robot navigation in a crowd. We show via extensive simulations the superior performance of our method over the typical approach of including distance-based collision avoidance constraints in the NMPC.

The paper is organized as follows. Section II formulates the considered problem. Section III gives an overview of the proposed approach. The crowd prediction and motion generation modules are described in detail in Sects. IV and V, respectively. Simulation results are presented in Sect. VI, while Sect. VII offers few concluding remarks.

II. Problem Formulation

The situation of interest is shown in Fig. 1. A wheeled mobile robot is assigned a *navigation task*, i.e., it must reach a desired goal region $\mathcal{G}$, in a workspace $\mathcal{W}$ populated by a crowd of moving humans. In the following, we discuss the considered problem referring to a differential drive robot (see Fig. 2); however, the proposed approach can be easily extended to other types of wheeled robots (e.g., car-like).

Since a differential drive robot is kinematically equivalent to a unicycle [22], we choose as $\boldsymbol{q} = (x, y, \theta)$ the configuration vector of the robot, where (x, y) are the Cartesian coordinates of a point B located along the sagittal axis at a distance $b > 0$ from the midpoint of the segment joining the centers of the actuated wheels, and θ is their common orientation. Assuming that the robot wheels roll without slippling and taking as control inputs the angular accelerations of the right and left wheel, denoted by $\dot{\omega}_R$ and $\dot{\omega}_L$, the kinematic model of the robot is given by

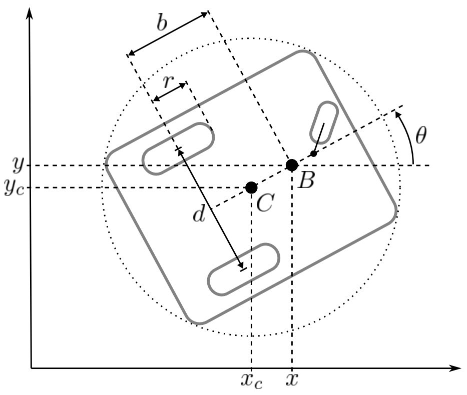

Fig. 2. The differential drive robot with the bounding circle considered for collision avoidance (see Sect. V).

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{u}) = \begin{pmatrix} v\cos\theta - \omega b \sin\theta \\ v \sin\theta + \omega b \cos\theta \\ \omega \\ \frac{r}{2}(\dot{\omega}_R + \dot{\omega}_L) \\ \frac{r}{d}(\dot{\omega}_R - \dot{\omega}_L) \end{pmatrix} \tag{1}$$

where $\boldsymbol{\xi} = (x, y, \theta, v, \omega)$ is the state vector taking values in a set $\mathcal{D}$, collecting the configuration $\boldsymbol{q}$ and the robot driving and steering velocities v and ω, $\boldsymbol{u} = (\dot{\omega}_R, \dot{\omega}_L)$ the control input vector, r the radius of the wheels and d the distance between their centers. We assume that the robot is always aware of its current state $\boldsymbol{\xi}$.

Denote by $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$ the volume occupied by the robot at configuration $\boldsymbol{q}$ and by $\mathcal{H}(t) \subset \mathcal{W}$ the volume occupied by the humans at time t. The robot is equipped with an on-board sensor – in particular, a laser rangefinder – through which it continuously acquires information about its surrounding. This information can be readily transcribed into a set $\mathcal{S}$ of measured relative position of points on the humans detected within the sensor field of view (FOV).

Then, the problem of safe navigation in a crowd consists in generating in real-time a robot motion, based on the available sensory information $\mathcal{S}$, that

1) is feasible w.r.t. the robot structure, i.e., it is consistent with model (1) and respects existing limitations on both state and input variables;
2) always avoids collisions with humans, i.e., $\mathcal{R}(\boldsymbol{q}(t)) \cap \mathcal{H}(t) = \emptyset$ at all time instants t;
3) eventually reaches the goal region $\mathcal{G}$, i.e., point B of the robot is brought inside $\mathcal{G}$.

We emphasize that, for safe navigation the knowledge about the humans future motion is essential. Obviously, in the considered context, this is not available and has thus to be predicted based on the available sensory information. This aspect is therefore part of the considered problem.

III. Proposed Approach

To solve the problem described in Sect. II, we propose the safe navigation framework outlined in the block scheme of

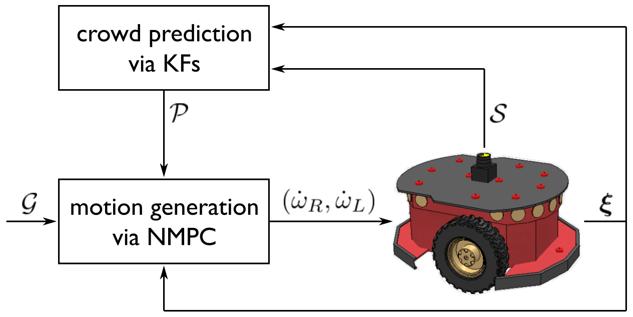

Fig. 3. Block scheme of the proposed safe navigation framework.

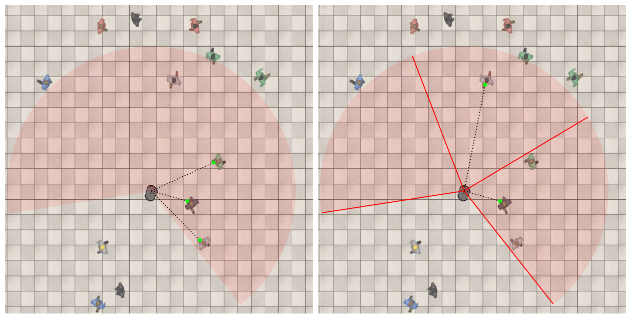

Fig. 4. Humans selected for collision avoidance by the K-*Neighbors* (left) and K-*Cones* (right) strategy for $K = 3$. Green dots indicate the closest point of the selected humans.

Fig. 3. It works in a digital fashion over sampling intervals of duration δ and is constituted by two main modules, i.e., the crowd prediction and motion generation module.

At the generic time instant $t_k = k\delta$, the available sensory information $\mathcal{S}_k$ and the current robot state $\boldsymbol{\xi}_k$ are passed to the crowd prediction module. Based on this information, it produces the set

$$\mathcal{P}_k = \{\boldsymbol{P}_k^1, \dots, \boldsymbol{P}_k^M\}$$

of the predicted motion of M humans over the time interval $[t_k, t_k + T_p]$, with $T_p = N\delta$ its duration and N the number of sampling intervals within it. In particular, the generic element of $\mathcal{P}_k$, i.e., the predicted motion of the generic human, is defined as

$$\boldsymbol{P}_k^j = (\boldsymbol{p}_{0|k}^j, \dots, \boldsymbol{p}_{N|k}^j),$$

where $\boldsymbol{p}_{i|k}^j$ is the predicted absolute position of the human closest point at $t_{k+i} = (k+i)\delta$. Moreover, M represents the number of humans that will be considered for collision avoidance and is a byproduct of the crowd prediction module.

The predicted humans motion $\mathcal{P}_k$ are fed to the motion generation module, together with the goal region $\mathcal{G}$ and the current robot state $\boldsymbol{\xi}_k$. This module relies on a real-time NMPC algorithm to produce wheel acceleration commands $(\dot\omega_R, \dot\omega_L)$ for the robot. It uses a prediction horizon of T_p (over which $\mathcal{P}_k$ is defined) and includes a CBF-based collision avoidance constraint for each of the M humans accounted in $\mathcal{P}_k$.

In the following, we describe in detail the two modules.

IV. Crowd Prediction via KFs

This module receives in input the sensory information $\mathcal{S}_k$ available at t_k and the current robot state $\boldsymbol{\xi}_k$. It outputs the predicted motion $\mathcal{P}_k$ of M humans over $[t_k, t_k + T_p]$.

To obtain this, it is essential to first choose which humans in the crowd will be considered for collision avoidance, whose future motion must therefore be predicted. We propose two possible strategies to make this choice, which are described in the following and illustrated in Fig. 4. Let K be a user-specified maximum number of humans to be considered.

- K-*Neighbors*. This strategy considers for collision avoidance the K nearest humans within the sensor FOV.
- K-*Cones*. This strategy conceptually divides the sensor FOV in K cones of equal detection angle and considers the nearest human within each of them.

The state at t_k of the selected K humans is estimated by means of an array of K KFs, called KF-1, ..., KF-K. In particular, consider the generic l-th human and collect in a vector $\boldsymbol{\chi}_k^l = (\boldsymbol{p}_k^l, \dot{\boldsymbol{p}}_k^l)$ his state at t_k, with $\boldsymbol{p}_k^l$ and $\dot{\boldsymbol{p}}_k^l$ the position and velocity of his closest point, which will be estimated by the associated KF-l. Assume this closest point moves omnidirectionally with constant velocity over the sampling intervals; moreover, notice that the measurements include only position information. Then, the state-transition and output models can be represented by the discrete-time stochastic system

$$\boldsymbol{\chi}_{k+1}^l = \underbrace{\begin{pmatrix} \boldsymbol{I}_{2\times2} & \delta\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{pmatrix}}_{\boldsymbol{A}} \boldsymbol{\chi}_k^l + \boldsymbol{v}_k \tag{2}$$

$$\boldsymbol{\zeta}_k^l = \underbrace{\begin{pmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} \end{pmatrix}}_{\boldsymbol{C}} \boldsymbol{\chi}_k^l + \boldsymbol{w}_k \tag{3}$$

where $\boldsymbol{v}_k$, $\boldsymbol{w}_k$ are white gaussian noises with zero mean and covariance matrices $\boldsymbol{V}_k$, $\boldsymbol{W}_k$, respectively; $\boldsymbol{\zeta}_k^l$ is the measured human closest point which needs to be carefully extracted from $\mathcal{S}_k$. Note that, since the available sensory information is non-specific (i.e., it does not embed the notion of human identity), the same KF can, in principle, estimate the state of different humans at different time instants.

The full crowd prediction procedure, outlined in Fig. 5, consists of the three stages described in the next subsections.

A. Data Association

Measurements extracted from $\mathcal{S}_k$ are associated to the KFs according to the adopted selection strategy.

When adopting the K-*Neighbors* strategy, an iterative procedure is involved to extract from $\mathcal{S}_k$ a subset $\mathcal{Z}_k$ containing the measured absolute position of the closest points on the K nearest humans. Such procedure relies on the assumption that the planar projection of the occupancy volume of any human in the crowd is always contained in a bounding circle having radius ρ_H. At the l-th iteration, the measured relative position of the point that is the closest to the robot is extracted from

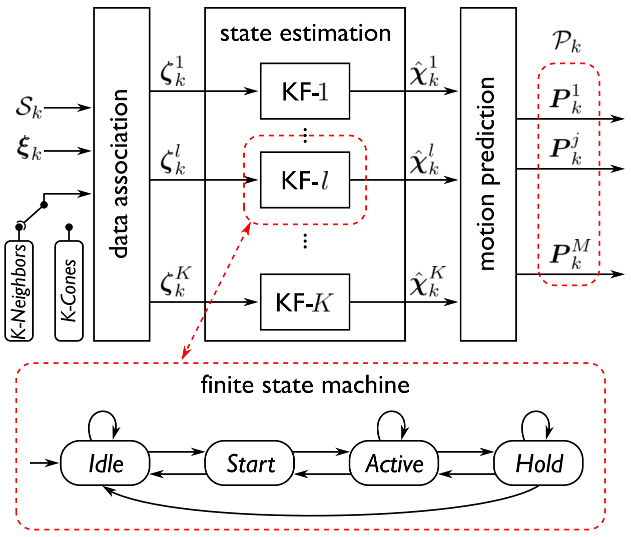

Fig. 5. Conceptual scheme of the crowd prediction module.

$\mathcal{S}_k$, expressed in absolute coordinates (using $\boldsymbol{\xi}_k$), and added to $\mathcal{Z}_k$; then, the center of the bounding circle of the l-th human is computed via simple geometrical considerations and all positions of points lying within it are removed from $\mathcal{S}_k$. The procedure terminates as soon as the number of elements in $\mathcal{Z}_k$ reaches K ($|\mathcal{Z}_k| = K$) or $\mathcal{S}_k$ becomes empty ($\mathcal{S}_k = \emptyset$). Finally, the extracted $|\mathcal{Z}_k|$ measurements are associated to as many KFs of the array using the maximum likelihood technique (see [23]), while a void measurement $\zeta_k = \emptyset$ is associated to each of the remaining $K - |\mathcal{Z}_k|$ KFs.

When adopting the *K-Cones* strategy, each cone is separately considered. Denote by $\mathcal{S}_k^l$ the subset of $\mathcal{S}_k$ containing the measured relative position of human points lying within the l-th cone. If this is not empty ($\mathcal{S}_k^l \neq \emptyset$), the measured relative position of the point that is the closest to the robot is extracted; then, expressing it in absolute coordinates (using $\boldsymbol{\xi}_k$) produces the measurement ζ_k^l to be associated to KF-l. Otherwise, a void measurement $\zeta_k^l = \emptyset$ is associated to KF-l.

B. State Estimation

The l-th KF receives the associated measurement ζ_k^j at t_k and has memory of the previous state estimate χ_{k-1}^l, the last valid measurement $\bar{\zeta}^l$ that it received and the time instant $\bar{t}^l$ at which this was received. Both ζ_k^j and χ_{k-1}^l can be either valid ($\zeta_k^j \neq \emptyset$, $\chi_{k-1}^l \neq \emptyset$) or void ($\zeta_k^j = \emptyset$, $\chi_{k-1}^l = \emptyset$). Moreover, KF-l has an associated finite state machine (FSM), depicted in Fig. 5, that governs its operation. It consists of four states, i.e., *Idle*, *Start*, *Active*, *Hold*, described in the following, together with the actions to be taken according to the received measurement and the KF memory in order to produce the state estimate $\hat{\chi}_k^l$.

- *Idle*. The KF is inactive, i.e., $\hat{\chi}_{k-1}^l = \emptyset$.
  - If $\zeta_k^l \neq \emptyset$, $\hat{\chi}_k^l$ is initialized as $\hat{\chi}_k^l = (\zeta_k^l, \boldsymbol{0})$ and the FSM state becomes *Start*.
  - Otherwise, $\hat{\chi}_k^l = \emptyset$ and the FSM state remains *Idle*.
- *Start*. The KF is initializing the state estimate.
  - If $\zeta_k^l \neq \emptyset$, $\hat{\chi}_k^l$ is set as $\hat{\chi}_k^l = (\zeta_k^l, \frac{1}{\delta}(\zeta_k^l - \hat{\boldsymbol{p}}_{k-1}^l))$ and the FSM state becomes *Active*.
  - Otherwise, $\hat{\chi}_k^l = \emptyset$ and the FSM state becomes *Idle*.
- *Active*. The KF is actively generating the state estimate based on valid measurements.
  - If $\zeta_k^l \neq \emptyset$, $\hat{\chi}_k^l$ is generated applying the standard prediction-correction procedure. First, the state prediction and covariance matrix are generated as

$$\begin{aligned}\hat{\chi}_{k|k-1}^l &= \boldsymbol{A}\hat{\chi}_{k-1}^l \\ \boldsymbol{\Sigma}_{k|k-1}^l &= \boldsymbol{A}\boldsymbol{\Sigma}_{k-1}^l\boldsymbol{A}^T + \boldsymbol{V}_k.\end{aligned} \tag{4}$$

    Then, the innovation $\boldsymbol{\nu}_k^l$, i.e., the difference between the measured and the predicted output, is computed as

$$\boldsymbol{\nu}_k^l = \zeta_k^l - \boldsymbol{C}\hat{\chi}_{k|k-1}^l \tag{5}$$

    and it is checked whether its norm does not exceed a predefined threshold $\bar{\nu}$.
    - If $\left\|\boldsymbol{\nu}_k^l\right\| < \bar{\nu}$, the state estimate and covariance matrix are corrected as

$$\begin{aligned}\hat{\chi}_k^l &= \hat{\chi}_{k|k-1}^l + \boldsymbol{G}_k^l\boldsymbol{\nu}_k^l \\ \boldsymbol{\Sigma}_k^l &= \boldsymbol{\Sigma}_{k|k-1}^l - \boldsymbol{G}_k^l\boldsymbol{C}\boldsymbol{\Sigma}_{k|k-1}^l,\end{aligned} \tag{6}$$

      with $\boldsymbol{G}_k^l$ the Kalman gain matrix computed as

$$\boldsymbol{G}_k^l = \boldsymbol{\Sigma}_{k|k-1}^l\boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{\Sigma}_{k|k-1}^l\boldsymbol{C}^T + \boldsymbol{W}_k\right)^{-1}$$

      and the FSM state remains *Active*.
    - Otherwise, $\hat{\chi}_k^l$ is reset as $\hat{\chi}_k^l = (\zeta_k^l, \hat{\dot{\boldsymbol{p}}}_{k-1}^l)$ and the FSM state becomes *Start*.
  - Otherwise, $\hat{\chi}_k^l$ is generated using (4–6) by setting $\zeta_k^l = \bar{\zeta}^l$ in (5) and the FSM state becomes *Hold*.
- *Hold*. The KF is temporarily generating the state estimate based on the last valid measurement $\bar{\zeta}^l$.
  - If $\zeta_k^l \neq \emptyset$, $\hat{\chi}_k^l$ is generated using (4–6) and the FSM state becomes *Active*.
  - Otherwise:
    - if $t_k \leq \bar{t}^l + \bar{T}$, $\hat{\chi}_k^l$ is generated using (4–6) by setting $\zeta_k^l = \bar{\zeta}^l$ in (5) and the FSM state remains *Hold*;
    - else, $\hat{\chi}_k^l = \emptyset$ and the FSM state becomes *Idle*.

C. Motion Prediction

Let $\hat{\mathcal{X}}_k = \{\hat{\chi}_k^1, \ldots, \hat{\chi}_k^M\}$ be the set of the M valid state estimates, i.e., $\hat{\chi}_k^j \neq \emptyset$ ($j = 1, \ldots, M$), among the K produced by the KFs. Then, the set $\mathcal{P}_k = \{\boldsymbol{P}_k^1, \ldots, \boldsymbol{P}_k^M\}$ is generated by computing each vector $\boldsymbol{P}_k^j = (\boldsymbol{p}_{0|k}^j, \ldots, \boldsymbol{p}_{N|k}^j)$ under the assumption that the corresponding human closest point will move with constant velocity $\hat{\dot{\boldsymbol{p}}}_k^j$ over the time interval $[t_k, t_k + T_p]$, i.e., $\boldsymbol{p}_{i|k}^j = \hat{\boldsymbol{p}}_k^j + i\delta\hat{\dot{\boldsymbol{p}}}_k^j$ ($i = 0, \ldots, N$).

V. Motion Generation via NMPC

This module receives in input the predicted humans motion $\mathcal{P}_k$ produced by the crowd prediction module, the goal region $\mathcal{G}$ and the current robot state $\boldsymbol{\xi}_k$. To generate the control inputs $(\dot{\omega}_R, \dot{\omega}_L)$ to be applied to the robot, it relies on a real-time NMPC algorithm that enforces a discrete-time CBF-based collision avoidance constraint throughout the prediction horizon T_p for each of the humans accounted in $\mathcal{P}_k$.

In the following, we first present the NMPC algorithm and then describe the adopted collision avoidance constraints.

A. NMPC Algorithm

At each control cycle, the NMPC solves a finite horizon constrained OCP. For computational efficiency, the latter is formulated as a finite dimensional Nonlinear Program (NLP). To this end, the NLP uses the discrete-time version

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{F}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \tag{7}$$

of the robot kinematic model, obtained by numerically integrating (1) assuming piecewise constant control inputs.

Let $\boldsymbol{\xi}_{i|k}$ and $\boldsymbol{u}_{i|k}$ be, respectively, the predicted robot state and control input at t_{k+i} computed at t_k. Collect in vectors

$$\begin{aligned}\Xi_k &= (\boldsymbol{\xi}_{0|k}, \dots, \boldsymbol{\xi}_{N|k})\\ \boldsymbol{U}_k &= (\boldsymbol{u}_{0|k}, ..., \boldsymbol{u}_{N-1|k})\end{aligned}$$

the NLP decision variables at t_k. Moreover, since the task is to drive the point B of the robot to the goal region $\mathcal{G}$ using the least possible control effort, let $\boldsymbol{\eta}_{i|k}$ be the predicted position of B at t_{k+i}, $\boldsymbol{\eta}_d$ the position of a representative point of $\mathcal{G}$ and $\boldsymbol{e}_{i|k} = \boldsymbol{\eta}_d - \boldsymbol{\eta}_{i|k}$ the predicted task error at t_{k+i}. Then, the running and terminal cost are

$$\begin{aligned}V_{i|k}(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k}) &= \boldsymbol{e}_{i|k}^T \boldsymbol{Q} \boldsymbol{e}_{i|k} + \dot{\boldsymbol{\eta}}_{i|k}^T \boldsymbol{R} \dot{\boldsymbol{\eta}}_{i|k} + \boldsymbol{u}_{i|k}^T \boldsymbol{S} \boldsymbol{u}_{i|k}\\ V_{N|k}(\boldsymbol{\xi}_{N|k}) &= \boldsymbol{e}_{N|k}^T \boldsymbol{Q}_N \boldsymbol{e}_{N|k} + \dot{\boldsymbol{\eta}}_{N|k}^T \boldsymbol{R}_N \dot{\boldsymbol{\eta}}_{N|k}\end{aligned}$$

where $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ are weighting matrices of appropriate dimensions for the predicted task error, velocity of B and control effort along the prediction horizon; $\boldsymbol{Q}_N$ and $\boldsymbol{R}_N$ are weighting matrices for the predicted task error and velocity of B at the final time instant of the prediction horizon.

At this point, the NLP can be formulated as

$$\min_{\Xi_k, \boldsymbol{U}_k} \sum_{i=0}^{N-1} V_{i|k}(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k}) + V_{N|k}(\boldsymbol{\xi}_{N|k}) \tag{8a}$$

subject to

$$\boldsymbol{\xi}_{0|k} - \boldsymbol{\xi}_k = \boldsymbol{0} \tag{8b}$$

$$\boldsymbol{\xi}_{i+1|k} - \boldsymbol{F}(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k}) = \boldsymbol{0}, \quad i = 0, ..., N-1 \tag{8c}$$

$$\boldsymbol{\xi}_{\min} \leq \boldsymbol{\xi}_{i|k} \leq \boldsymbol{\xi}_{\max}, \quad i = 0, ..., N \tag{8d}$$

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_{i|k} \leq \boldsymbol{u}_{\max}, \quad i = 0, ..., N-1 \tag{8e}$$

collision avoidance constraints (8f)

where (8b) enforces the initial condition, (8c) the discrete-time system dynamics, (8d–8e) the state and input bounds, and (8f) collision avoidance throughout the prediction horizon. The latter constraints are presented in the following.

Once the NLP is solved, the first control samples $\boldsymbol{u}_{0|k}$ are extracted from the obtained $\boldsymbol{U}_k$ and applied to the robot.

B. CBF-based Collision Avoidance

To guarantee collision avoidance, the proposed NMPC incorporates constraints formulated via discrete-time CBFs.

Consider the smallest circle bounding the planar projection of the robot occupancy volume $\mathcal{R}(\boldsymbol{q})$ at any configuration $\boldsymbol{q}$ (see Fig. 2). Let ρ be the radius of this circle and C its center, whose Cartesian coordinates $\boldsymbol{c}(\boldsymbol{\xi})$ depend on the robot configuration $\boldsymbol{q}$ contained in the state vector $\boldsymbol{\xi}$.

We say that a state $\boldsymbol{\xi}$ is safe, i.e., it is collision-free, w.r.t. a generic human if the distance between the robot bounding circle centered at $\boldsymbol{c}(\boldsymbol{\xi})$ and the human closest point $\boldsymbol{p}$ is at least equal to a safety clearance $d_s > 0$, i.e., $\|\boldsymbol{c}(\boldsymbol{\xi}) - \boldsymbol{p}\| \geq \rho + d_s$. Accordingly, we define the safe set of robot states w.r.t. a generic human as

$$\mathcal{C} = \{\boldsymbol{\xi} \in \mathcal{D} : h(\boldsymbol{\xi}) \geq 0\}$$

where

$$h(\boldsymbol{\xi}) = \|\boldsymbol{c}(\boldsymbol{\xi}) - \boldsymbol{p}\|^2 - (\rho + d_s)^2.$$

It can be shown [18] that function $h(\boldsymbol{\xi})$ is a CBF and, in the discrete-time domain, it satisfies the condition

$$\Delta h(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \geq -\gamma h(\boldsymbol{\xi}_k) \tag{9}$$

with $\Delta h(\boldsymbol{\xi}_k, \boldsymbol{u}_k) = h(\boldsymbol{\xi}_{k+1}) - h(\boldsymbol{\xi}_k)$ and $0 < \gamma \leq 1$.

By imposing condition (9) throughout the prediction horizon for all M humans accounted in $\mathcal{P}_k$, the CBF-based collision avoidance constraints in (8f) can be explicitly written as

$$\Delta h^j(\boldsymbol{\xi}_{i|k}, \boldsymbol{u}_{i|k}) \geq -\gamma h^j(\boldsymbol{\xi}_{i|k}), \quad \begin{aligned} i &= 0, \dots, N-1,\\ j &= 1, \dots, M,\end{aligned} \tag{10}$$

where

$$h^j(\boldsymbol{\xi}_{i|k}) = \|\boldsymbol{c}(\boldsymbol{\xi}_{i|k}) - \boldsymbol{p}_{i|k}^j\|^2 - (\rho + d_s)^2,$$

with $\boldsymbol{p}_{i|k}^j$ extracted from element $\boldsymbol{P}_k^j$ of $\mathcal{P}_k$.

Note the following points.

- Since the value of M can vary at each control cycle between 0 and K, in a concrete implementation one shall set up K collision avoidance constraints. During operation, only the M constraints required at every sampling interval will be activated. For sake of illustration we omit discussion about the activation mechanism.
- The identity of the closest point of a certain human might be different at different time instants. Such aspect is not explicitly considered by our collision avoidance constraints which, instead, involve only information about the predicted future position of the point that is the closest at t_k. However, as we will show via simulation results in Sect. VI, this is not an issue for the proposed scheme thanks to (i) the real-time capabilities of our implementation that allows high control frequency and (ii) the use of a relatively small value for γ that improves the ability to promptly react to sudden changes in the robot surroundings.

VI. Simulations

The proposed safe navigation method was implemented in the CoppeliaSim environment as a C++ plugin. The latter wraps a library implementing the NMPC algorithm generated using the Python interface of `acados` [7]. In particular, the corresponding NLP is solved via Real-Time Iteration [24].

The targeted platform is the Pioneer 3-DX mobile robot for which $r = 9.75$ cm and $d = 38.1$ cm. It is equipped with an Hokuyo URG-04LX laser rangefinder whose detection area is aligned with the robot heading direction and has range 5 m and angle 240°, thus leaving a 120° blind zone behind the robot. According to the robot physical characteristics, its driving and steering velocities are bounded as, respectively, $0 \leq v \leq 1.2$ m[1] and $|\omega| \leq 5.24$ rad/s, while the angular accelerations of the wheels as $|\dot{\omega}_{R,L}| \leq 70$ rad/s^2.

To assess the performance of the proposed method we performed two simulation campaigns in which the environment is populated by, respectively, a *robot-friendly* and *robot-unfriendly* crowd depending on whether the humans are aware of the presence of the robot (and try to avoid collisions with it) or not. To simulate the motion of the crowd, we assumed that each human moves along a sequence of viapoints (locations of the environment) in a unicycle-like fashion under the action of an artificial potential field that attracts him towards his currently targeted viapoint while repulsing him from the other humans (and from the robot in the first campaign). Every time a human reaches a viapoint, he stands there for a certain pause period. Moreover, each human has his own maximum linear velocity. We emphasize that the robot is not aware of such crowd behavior, but only relies on sensory information.

In both campaigns, our aim was to evaluate the effect of the choice regarding the selection strategy adopted in the crowd prediction module, and also to compare our CBF-based method with a purely distance-based (DB) approach that is obtained by using

$$h^j(\boldsymbol{\xi}_{i|k}) \geq 0, \quad i = 0, \ldots, N, \quad j = 1, \ldots, M$$

in (8f) instead of (10). To this purpose, for each campaign, we considered 12 simulation scenarios obtained by varying (i) the number (5, 10 or 20) of humans in a 15×15 m environment, (ii) the adopted selection strategy (*K-Neighbors* or *K-Cones*), and (iii) the formulation of the collision avoidance constraints (CBF or DB). For each of these scenarios, we run 50 different simulations, each time randomly generating the initial configuration of the robot, goal region $\mathcal{G}$, sequence of viapoints, associated pause periods and maximum linear velocity of each human.

All the simulations were performed on an Intel Core i7-8700K CPU running at 3.7 GHz. In all of them, the whole safe navigation scheme worked with a sampling time $\delta = 0.05$ s, while the prediction horizon was set to $T_p = 2$ s; also, we used $K = 3$, $b = 0.15$ m, $\rho_H = 0.8$ m, $\gamma = 0.3$ and $d_s = 1$ m. Whenever the NLP solved by the motion generation module proved to be infeasible, we extracted the control inputs for the current time instant from the last available NMPC solution. Although in our simulations we have rarely observed infeasibility issues, a deep investigation of this aspect is part of our future work (see Sect. VII).

[1]We chose to not use a negative value for the minimum driving velocity in order to prevent the robot moving back in its blind zone.

# of humans	selection strategy	collision avoidance constraint	success rate (%)	maximum computation time (ms)
5	*K-Neighbors*	CBF	100	31
		DB	92	28
	K-Cones	CBF	98	34
		DB	90	29
10	*K-Neighbors*	CBF	96	30
		DB	86	33
	K-Cones	CBF	98	29
		DB	72	34
20	*K-Neighbors*	CBF	88	36
		DB	64	36
	K-Cones	CBF	86	35
		DB	48	41

TABLE I

Performance data in case of robot-friendly crowd

# of humans	selection strategy	collision avoidance constraint	success rate (%)	maximum computation time (ms)
5	*K-Neighbors*	CBF	92	30
		DB	90	31
	K-Cones	CBF	92	32
		DB	84	27
10	*K-Neighbors*	CBF	74	28
		DB	62	28
	K-Cones	CBF	80	27
		DB	68	31
20	*K-Neighbors*	CBF	60	32
		DB	38	30
	K-Cones	CBF	58	29
		DB	40	31

TABLE II

Performance data in case of robot-unfriendly crowd

The results of the two simulation campaigns are reported in Tables I–II, respectively, where we show the success rate (we recorded a failure whenever a collision occurred) and maximum computation time (including both crowd prediction and motion generation modules) averaged over the 50 runs in each simulation scenario. Video clips showing examples of generated motions are available at the link `https://youtu.be/iDdM6Ud9I4c`.

In the case of robot-friendly crowd (see Table I), as expected, for both our and DB method the success rate decreases as the number of humans populating the environment increases. However, our method maintains higher success rate even in the most challenging environment containing 20 humans, where we achieve a success rate higher than 85%, independently of the adopted selection strategy, while the DB method fails in more than half of cases when using the *K-Cones* strategy. Moreover, the table confirms the real time capabilities of the proposed scheme, as maximum computation time is always lower than the sampling time. We

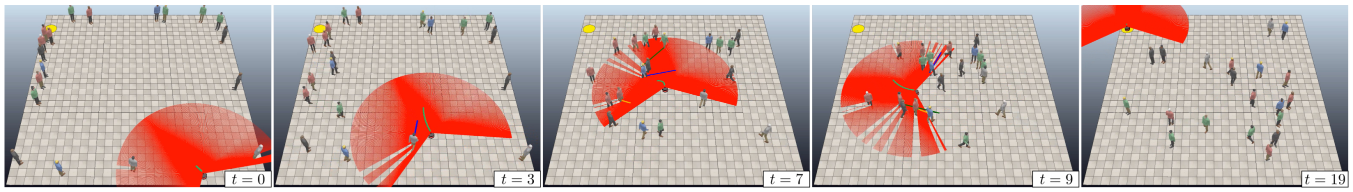

Fig. 6. Safe navigation in a robot-unfriendly crowd of 20 humans using the *K-Neighbors* strategy. Snapshots from a simulation.

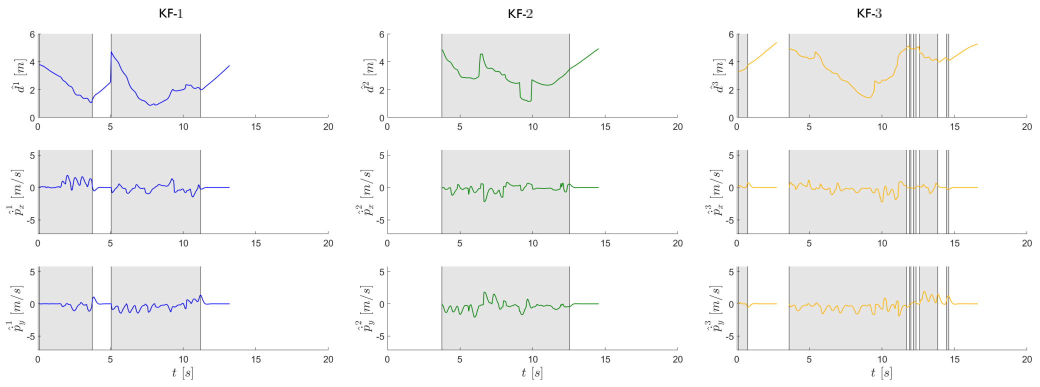

Fig. 7. Safe navigation in a robot-unfriendly crowd of 20 humans using the *K-Neighbors* strategy. Plots of $\hat{d}^l$, $\hat{\dot{p}}_x^l$, $\hat{\dot{p}}_y^l$ $(l = 1, 2, 3)$, i.e., distance and x,y-components of the velocity of a human estimated by KF-l. Gray zones indicate time intervals in which the corresponding KF was in the *Active* state.

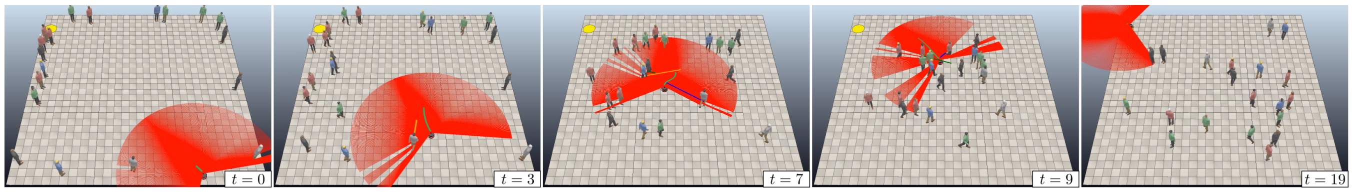

Fig. 8. Safe navigation in a robot-unfriendly crowd of 20 humans using the *K-Cones* strategy. Snapshots from a simulation.

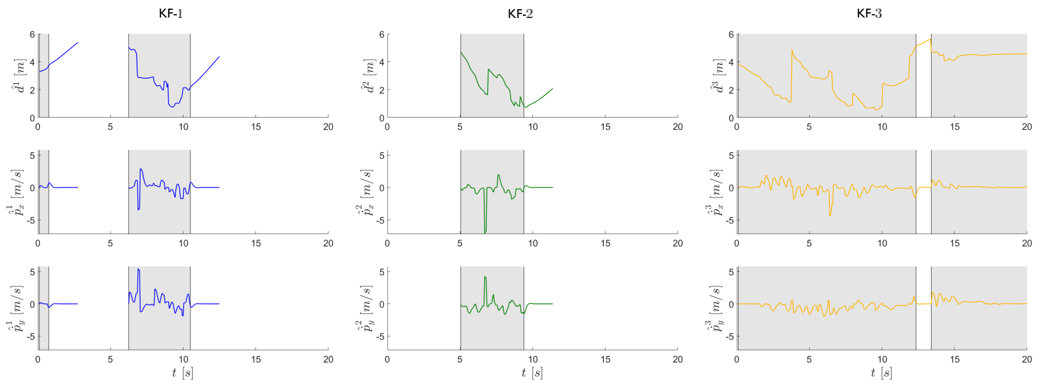

Fig. 9. Safe navigation in a robot-unfriendly crowd of 20 humans using the *K-Cones* strategy. Plots of $\hat{d}^l$, $\hat{\dot{p}}_x^l$, $\hat{\dot{p}}_y^l$ $(l = 1, 2, 3)$, i.e., distance and x,y-components of the velocity of a human estimated by KF-l. Gray zones indicate time intervals in which the corresponding KF was in the *Active* state.

also report that, in our observations, most of the computation time is spent by the motion generation module, while only a negligible portion is used by the crowd prediction module. This highlights the efficiency of the latter which, differently from other existing methods, does not even need a training phase.

The case of robot-unfriendly crowd (see Table II) is clearly more challenging. In environments containing 10 and 20 humans, the success rate is significantly reduced compared to the case of robot-friendly crowd. This is reasonable if we consider that in this case a human can either approach the robot from its blind zone and collide with it without any possibility for the robot to react, or suddenly turn towards it without giving the robot the chance to react in time. However, the table reveals that, even in the presence of a robot-unfriendly crowd, our method is clearly more effective than the DB approach. Also in this case the real time requirement is respected, as proven by the reported maximum computation times.

Figures 6 and 8 show snapshots from two simulations obtained using, respectively, the *K-Neighbors* and *K-Cones* strategy in a robot-unfriendly crowd of 20 humans. Figures 7 and 9 show associated plots. Clips are included in the video. Comparing the snapshots, taken at same time instants, it is possible to appreciate the identical placement of the crowd and the different one of the robot, whose travelled path results from the chosen selection strategy.

Although in terms of both success rate and computation time we found no significant differences between the two selection strategies, in practice the *K-Cones* strategy (as can be seen from the plots corresponding to KF-1 and KF-2 in Fig. 9) might consider for collision avoidance a number of humans smaller than K even if K humans are actually present in the sensor detection area, which could obviously be disadvantageous. On the other hand, the *K-Cones* strategy is a valid option when dealing with multiple sensors having reduced detection angle. In this case, each cone will correspond to a certain sensor and the presented scheme can be applied without any significant modification.

Note that, even if we presented simulation results obtained using $K = 3$, our approach can work with generic values of K provided that this still guarantees real time performance. The video also shows simulations obtained with $K = 6$.

VII. Conclusion

In this paper, we presented a sensor-based scheme for safe robot navigation in a crowd of moving humans. At each control cycle, the crowd prediction module foresees the future motion of the humans in the robot surroundings using the available sensory information. Then, the motion generation module produces feasible commands for the robot to safely drive it among the humans until a desired goal region of the workspace is reached. For crowd prediction we employ a simple, yet effective, technique based on Kalman filters, while our motion generation strategy combines NMPC and collision avoidance constraints formulated via discrete-time CBFs.

Simulation results show that this combination can produce sensible results in environments crowded by a varying number of moving humans. Moreover, we demonstrated that our approach outperforms the typical one consisting in adopting purely distance-based collision avoidance constraints.

Future work will address (i) the investigation of feasibility properties of the NMPC and design of recovery strategies in case of possible infeasibilities, (ii) the extension of our method to environments containing (possibly unknown) static obstacles, (iii) an in depth performance analysis of the effect of the various parameters (e.g., K, T_p, γ).

References

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[3] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *2007 IEEE Int. Conf. on Robotics and Automation*. IEEE, 2007, pp. 1603–1609.

[4] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The Int. J. of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[5] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[6] D. J. Gonon, D. Paez-Granados, and A. Billard, "Reactive navigation in crowds for non-holonomic robots with convex bounding shape," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4728–4735, 2021.

[7] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, pp. 1–37, 2021.

[8] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[9] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *European Control Conference (ECC)*, 2013, pp. 4136–4141.

[10] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017.

[11] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[12] H. Febbo, P. Jayakumar, J. L. Stein, and T. Ersal, "Real-Time Trajectory Planning for Automated Vehicle Safety and Performance in Dynamic Environments," *Journal of Autonomous Vehicles and Systems*, vol. 1, no. 4, 12 2021.

[13] W. Jin, P. Salaris, and P. Martinet, "Proactive-cooperative navigation in human-like environment for autonomous robots," in *Int. Conf. on Informatics in Control, Automation and Robotics*, 2020.

[14] N. Bohórquez, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Safe navigation strategies for a biped robot walking in a crowd," in *2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 379–386.

[15] S. G. Tarantos and G. Oriolo, "A dynamics-aware NMPC method for robot navigation among moving obstacles," in *Int. Conf. on Intelligent Autonomous Systems*, 2022, pp. 129–143.

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

[17] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *IEEE Conf. on Decision and Control (CDC)*, 2019, pp. 7623–7628.

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

[19] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions," in *2022 IEEE Int. Conf. on Robotics and Automation*, 2022, pp. 286–292.

[20] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *IEEE Int. Conf. on Computer Vision*. IEEE, 2009, pp. 261–268.

[21] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[22] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, London, 2009.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[24] M. Diehl, H. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.