2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
July 15-19, 2024. Boston, MA, USA

# A Convex Optimization Based Differentially Driven Mobile Robot Planner for Crowd Navigation

Leixin Chang Haoran Yuan Tengyue Wang Haonan Mai Liangjing Yang

***Abstract*—Safe navigation in a pedestrian-rich environment has gained a lot of attention in robotics research. Unlike classical motion planning with a static environment, pedestrian-rich scenarios are associated with a highly dynamic environment and safety risk. In this paper, we propose a novel local planner designed for differentially driven wheeled robots, directly inspired by the Dynamic Window Approach (DWA). Our model leverages convex optimization and differential drive kinematics to efficiently determine optimal velocity inputs as the robot moves in the human crowd. Our approach does not only account for the position but also the velocity of the pedestrians in the planning framework to facilitate safer navigation through dynamic pedestrian-dense environments. Through extensive simulation experiments, we demonstrate the superior effectiveness and safety of our method compared to DWA, showcasing significant enhancements in collision-free navigation success rates and computational efficiency through the use of convex optimization techniques. Code release: https://github.com/Leixinjonaschang/convex_op_planner.**

***Index Terms*—Mobile robot navigation, convex optimization**

## I. Introduction

As robots become increasingly prevalent, the symbiotic integration of robots and humans has garnered significant research attention. As a result, the effort to ensure safe robot autonomy in a human-shared workspace has become imperative to the feasibility of its deployment. Robot autonomous navigation is an important and classical problem in robotics. Generally, the navigation of mobile robots involves two steps, namely mapping, and planning. Initially, a map of the environment is usually created either online or offline through simultaneous localization and mapping (SLAM) techniques. Subsequently, planning algorithms generate a feasible and collision-free global path based on the map. Safe autonomous navigation in pedestrian-rich areas, also called crowd navigation, is still an open challenge due to the highly uncertain and dynamic movements of humans and the high demand for safety. The robots most widely used in human living environments, such as cleaning robots and delivery robots, predominantly utilize the differential drive kinematic model. Inspired by DWA[1], a widely used, simple, and effective algorithm, we propose a novel mobile robot trajectory planner specifically for differentially driven mobile robots to deal with the highly dynamic human movements in crowd navigation, which integrates the motion information of the pedestrians into planning and employs the convex optimization to determine the optimally safe trajectory efficiently.

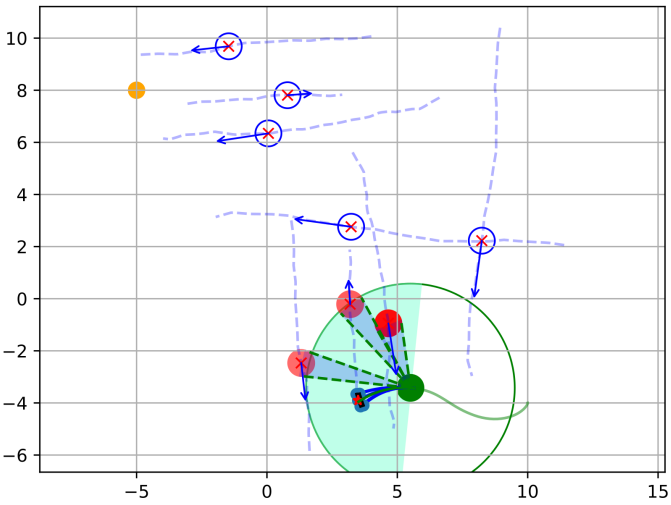



Fig. 1. Simulation of navigation with the proposed method. The blue circles represent moving pedestrians and the light blue lines represent their trajectories. The green solid circle represents the robot and the light green lines represent its trajectory. The hollow green circle is the local map. The transparent green semi-circle is the region of interest for navigation. The yellow dot is the goal point. The perceived pedestrians will be marked as red-filled circles, and the closer to the robot it is, the redder it will be filled. The black polygon ahead of the robot is the state window and the red star in the polygon is the optimal potential position.

### A. Related Work

A prevalent research approach to crowd navigation is to model crowd behavior. Social Force (SF) model[2] is a pioneering work in the crowd behavior modeling field, which has applied widely to robot crowd navigation. Optimal Reciprocal Collision Avoidance (ORCA)[3] is another method to model mutual interactions among multiple agents. However, these two approaches for modeling the mutual interactions of a crowd have many hyperparameters that need to be tuned for reliable performance, which makes them time-consuming to deploy. In addition, ORCA employs the concept of velocity obstacles, which leads to overly conservative agent behavior and low spatial utilization in crowded environments.

This work was supported in part by the Industry-University Educational Collaboration Project 230904701283901 under the Ministry of Education, China, and by the Dynamic Research Enterprise for Multidisciplinary Engineering Sciences (DREMES) at Zhejiang University and the University of Illinois at Urbana-Champaign, led by Principal Supervisor Liangjing Yang.

Leixin Chang, Haoran Yuan, Tengyue Wang, Haonan Mai and Liangjing Yang are with the ZJU-UIUC Institute and School of Mechanical Engineering, Zhejiang University, Zhejiang, China (e-mail: {leixin.23, haoran1.20, tengyue.21, haonan1.22, liangjingyang}@intl.zju.edu.cn).

Reinforcement Learning (RL) based approaches are also popular in research on safe crowd navigation topics. The deep V-Learning method tackles this problem by using the neural network to model the interaction rules[4], [5] which employs OCRA as an expert to get the supervision and uses RL to learn a value function for motion planning. Hybrid-MRCA[6] employs the hybrid control centered on RL to solve the collision avoidance among distributed agents. CADRL[7] approach employs RL to train a policy for collision avoidance. The uniqueness of this work lies in the absence of introducing the dynamics of other agents. However, the common drawback of the learning-based methods is the high computational cost, which requires additional hardware resources and can be a barrier to deployment in low-resource environments.

Trajectory prediction-based approaches involve the idea of pedestrian trajectory prediction, which has progressed rapidly recently[8], [9]. The main idea of this method is to integrate the prediction of human movements into the motion planning framework to take the potential collisions into consideration. One line of this work is to utilize the learning-based pedestrian trajectory prediction model to estimate the future states of pedestrians. This kind of method can produce a very good performance of collision avoidance[10]–[13]. However, this learning-based prediction involved ways to inherit high computational costs and can be limited in low-resource environments. Another line of this work uses model-based approaches, like the Kalman Filter (KF) model[14], to predict the trajectories of pedestrians[15].

### B. Contribution

Fully exploiting the kinematic features of differentially driven mobile robots and inspired by the concept *Dynamic Velocity Window* in DWA[1], we integrate the idea of convex optimization into robot trajectory planning, which enables robots to find the optimal trajectory efficiently considering the motion of pedestrians in a crowded environment. Our main contributions can be summarized as follows:

- we propose an efficient convex optimization based trajectory planner for differentially driven mobile robots' safe navigation in pedestrian-rich environments.
- conduct a series of simulation experiments to validate the effectiveness of the proposed method and verify the superiority with respect to the DWA method.

### C. Content Organization

Section II describes the differentially driven robot kinematic model and analyzes the kinematic features that are employed in the planning. Section III elaborates on the proposed convex optimization based planning algorithm, consisting of algorithm pipeline, constraints formation, and cost function design. Section IV displays the simulation results and analysis And Section V offers the concluded remarks and future direction.

## II. Analysis for Differentially Driven Kinematics

### A. Differentially Driven Kinematic Model

Figure 2 illustrates the differentially driven kinematic model of the mobile robot. Consider this kinematic model with state

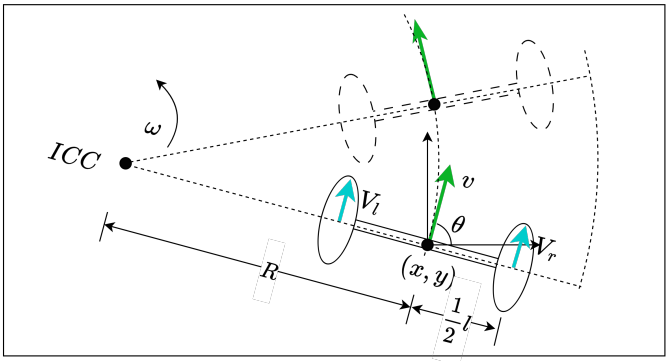



Fig. 2. Two-wheeled differential driven robot kinematic model.

$[x_t, y_t, \theta_t]$, where $x_t$ and $y_t$ represent the position of the robot and $\theta_t$ represents its orientation. In Figure 2, $ICC$ means the instantaneous center of curvature and $R$ and $l$ represent the radius of the turning circle and the distance between two wheels, respectively. $V_l$ and $V_r$ are the left and right wheel speeds. We can formulate the kinematic model as below[16],

$$\begin{cases} x_{t+1} = x_t - R\sin(\theta_t) + R\sin(\theta_t + \omega_t \Delta t) \\ y_{t+1} = y_t - R\cos(\theta_t) + R\cos(\theta_t + \omega_t \Delta t) \\ \theta_{t+1} = \theta_t + \omega_t \Delta t \end{cases} \tag{1}$$

where $R = \dfrac{v_t}{\omega_t}$, $v_t = \dfrac{V_{lt} + V_{rt}}{2}$ and $\omega_t = \dfrac{V_{rt} - V_{lt}}{l}$. And it is worth noting that the linear velocity $v_t$ and angular velocity $\omega_t$ are decoupled.

### B. Velocity Scope and State Window

Due to the dynamics constraints of actuators mounted on the robot and the robot's mechanical structure, there will be a velocity scope $V_{d,t}$ for the robot concerning current robot velocity $(v_t, \omega_t)$ in every time period, which is illustrated as a light blue rectangle in Fig. 3. The robot inherent dynamics constraints shown in Table I result in a maximum and minimum for both angular velocity and linear velocity given the current velocity $(v_t, \omega_t)$, namely $\omega_{t+1,max}$, $\omega_{t+1,min}$, $v_{t+1,max}$ and $v_{t+1,min}$. These velocity boundaries are defined as (2) and then the velocity scope at time step $t$ is defined as (3).

TABLE I
Dynamics Constraints of Robot Hardware

| Symbol | Meaning |
|---|---|
| $a_{max}$ | Maximum Linear Acceleration |
| $\alpha_{max}$ | Maximum Angular Acceleration |
| $v_{max}$ | Maximum Linear Velocity |
| $v_{min}$ | Minimum Linear Velocity |
| $\omega_{max}$ | Maximum Angular Velocity |
| $\omega_{min}$ | Minimum Angular Velocity |

$$\begin{cases} v_{t+1,min} = max\{v_{min}, v_t - a_{max} \cdot \Delta t\} \\ v_{t+1,max} = min\{v_{max}, v_t + a_{max} \cdot \Delta t\} \\ \omega_{t+1,min} = max\{\omega_{min}, \omega_t - \alpha_{max} \cdot \Delta t\} \\ \omega_{t+1,max} = min\{\omega_{max}, \omega_t + \alpha_{max} \cdot \Delta t\} \end{cases} \tag{2}$$

$$V_{d,t} = \{(v_{t+1}, \omega_{t+1}) | (v_{t+1}, \omega_{t+1}) \in V_{dv,t} \times V_{d\omega,t}\} \quad (3)$$
$$V_{dv,t} = [v_{t+1,min}, v_{t+1,max}], V_{d\omega,t} = [\omega_{t+1,min}, \omega_{t+1,max}]$$

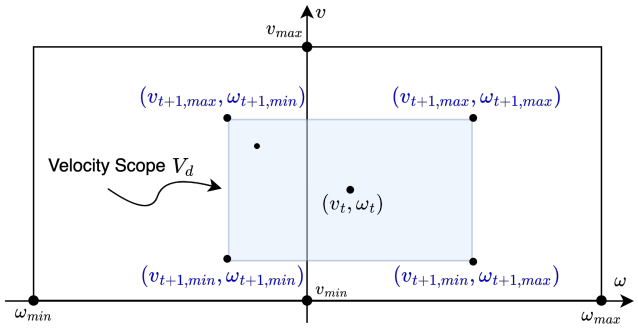



Fig. 3. Velocity scope $V_{d,t}$. The light blue rectangle represents the velocity scope based on current velocity pair $(v_t, \omega_t)$, and velocity input pairs $(v_{t+1}, \omega_{t+1})$ in this velocity scope are admissible within $\Delta t$ for the robot under the dynamics constraints.

Given the differential kinematic model (1), each admissible velocity input pair $(v_{t+1}, \omega_{t+1})$ in the velocity scope shown in Fig. 3 corresponds to a unique trajectory, and the endpoints of these trajectories form a region which can be called **state window** as illustrated in the Fig. 4.

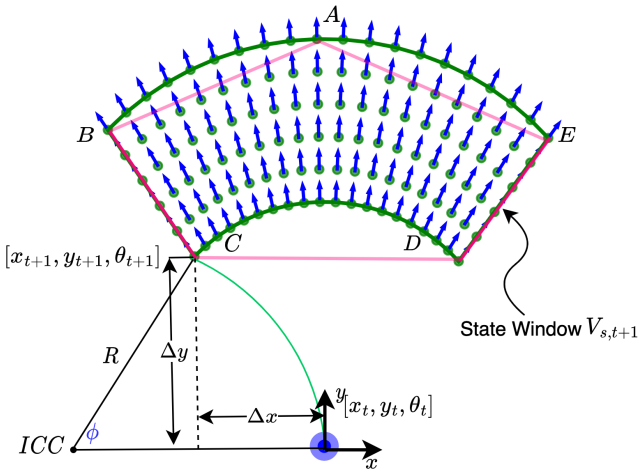



Fig. 4. State window $V_{s,n=1}$. The current robot state is represented as the blue dot and the potential robot positions at $t+n$ time step given the current robot state compose the state window, which looks like a sector ring but is not for the upper bound and the lower bound is not strictly a circular arc. The potential robot positions at $(t+1)$ step are represented by the green dots while the attached blue arrows are the corresponding orientations.

Fig. 4 shows the configuration of the state window, which is defined in the Cartesian space and contains all the possible positions at the $(t+1)$ step given the state at $t$ step. The green circular arc is a representative trajectory of the robot given a state pair $[x_t, y_t, \theta_t]$ and $[x_{t+1}, y_{t+1}, \theta_{t+1}]$. The state window is actually defined by the kinematic model (1). In Fig. 4 $\Delta y = R\sin\phi$, $\Delta x = R(1-\cos\phi)$, $\phi = \omega_t \Delta t$ and $R = \frac{v_{t+1}}{\omega_{t+1}}$ with respect to the coordinate attached to the robot. From (6), it's easy to notice that velocity inputs with constant $v_{t+1}$ result in a noncircular arc around the current robot position. So it can be inferred that the upper and lower boundaries of the state window are noncircular arcs.

$$\Delta x^2 = R^2(1 + \cos^2\omega_{t+1}\Delta t - 2\cos\omega_{t+1}\Delta t) \quad (4)$$
$$\Delta y^2 = R^2 \sin^2\omega_{t+1}\Delta t \quad (5)$$
$$\Delta x^2 + \Delta y^2 = 2\frac{v_{t+1}}{\omega_{t+1}}(1 - \cos\omega_{t+1}\Delta t) \quad (6)$$

Velocity inputs with maximum and minimum $\omega_{t+1}$ correspond to the left and right boundaries, respectively. Given (7), $\frac{\Delta y}{\Delta x}$ is a constant when $\omega_{t+1}$ is a constant. Furthermore, it's easy to infer that the slopes of the left and right boundaries are constant with $\omega_{t+1,max}$ and $\omega_{t+1,min}$ respectively. So the left and right boundaries are the line segments whose extension pass through the current robot position $[x_t, y_t]$.

$$\frac{\Delta y}{\Delta x} = \frac{1 + \cos\omega_{t+1}\Delta t}{\sin\omega_{t+1}\Delta t} \quad (7)$$

However, it is important to note that the state window appears as a sector ring-like shape only when minimum linear velocity $v_{min} >= 0$.

Given the kinematic model in (1), we can know that only $[x_{t+1}, y_{t+1}]$ is sufficient to uniquely describe the motion given the current state $[x_t, y_t, \theta_t]$. So the state format $[x_{t+1}, y_{t+1}, \theta_{t+1}]$ is redundant to describe the motion. In other word, if the $[x_{t+1}, y_{t+1}]$ is known given known current state, $\theta_{t+1}$, which indicates that$[x_{t+1}, y_{t+1}]$ is more dense state format to define the state window. Finally, we can define the state window $V_s$ at time step $t + \Delta t$ given the velocity input $(v_{t+1}, \omega_{t+1})$ as (8).

$$\begin{aligned} V_{s,n=1} = \{&(x_{t+1}, y_{t+1})| \\ &x_{t+1} = x_t - \frac{v_{t+1}}{\omega_{t+1}}\sin\theta_t + \frac{v_{t+1}}{\omega_{t+1}}\sin(\theta_t + \omega_{t+1}\Delta t), \\ &y_{t+1} = y_t - \frac{v_{t+1}}{\omega_{t+1}}\cos\theta_t + \frac{v_{t+1}}{\omega_{t+1}}\cos(\theta_t + \omega_{t+1}\Delta t), \\ &\forall v_{t+1} \in V_{dv} \wedge \forall \omega_{t+1} \in V_{d\omega}\} \end{aligned} \quad (8)$$

In addition, we want to emphasize that the shape of the state window in the $(t+n)$ would still be like a sector ring as shown in Fig. 4 if the velocity input $(v, \omega)$ keep constant in these $n$ steps.

## III. Method

We aim to implement efficient navigation for the widely used differentially driven mobile robot in a highly dynamic pedestrian-rich environment. Based on the analysis for the differential kinematics in Section II-A and concepts of velocity scope and state window that are established in Section II-B, we can formulate the problem as finding the optimal admissible state $(x_{t+1}, y_{t+1})$ at the subsequent time step in the state window given the current state $(x_t, y_t, \theta_t)$ under the dynamics constraints that the velocity input $(v_{t+1}, \omega_{t+1})$ is confined in velocity scope $V_{d,t}$ shown in Fig. 3.

For the problem above, we propose a novel local motion planner that employs the convex optimization technique and incorporates the highly dynamic motion of moving pedestrians in the optimization-based planning framework to implement efficient and safe crowd navigation. The primary idea is as follows:

1) Define the state window $V_{s,t+n}$ based on the current state and velocity scope $V_{d,t}$ formed by dynamic constraints, whereby $n$ is the prediction horizon.
2) Find the optimal point $(x^*_{t+n}, y^*_{t+n})$ in the state window $V_{s,t+n}$ as the optimal potential state which is the best state for the aim of approaching the goal and avoiding the collision with moving pedestrians.
3) Inversely calculating the optimal velocity input $(v^*_{t+1}, \omega^*_{t+1})$ in the velocity scope $V_{d,t}$ from the already found optimal subsequent position $(x^*_{t+1}, y^*_{t+1})$, which will be the input to navigate the robot.

In step 1) shown above, we construct the state window at $t+n$ time step, which is the mapping of velocity scope onto cartesian space given the current robot state and $n$ prediction horizon. In other words, each point in the state window $V_{s,t+n}$ represents the end position of the robot after running n steps with each admissible velocity pair in the velocity scope $V_{d,t}$. In step 2), we approximate the state window as a convex polygon represented by a series of linear constraints while designing a quadratic cost function that aims to navigate the robot to approach the goal and avoid colliding with moving human with their motion considered. This approach leads to a convex optimization problem formulation, which means it could be solved very efficiently.

### A. Optimization Problem Formulation

*1) Constraints:* From Fig. 4, we can notice that the state window, which represents the constraints of the optimization problem, is non-convex. To convexify the constraints, we approximate the sector ring-like state window to a polygon $V_{c,t+\Delta t}$, which is shown as a pink pentagon in Fig. 4 and can be formulated as a series of linear inequalities. We pick five points $A-E$ from the boundary of the state window to form a convex hull $V_c$. Point $B$, $C$, $D$, $E$ are the end of $n$ step trajectories with velocity input $(v_{t+1,max}, \omega_{t+1,max})$,$(v_{t+1,min}, \omega_{t+1,max})$,$(v_{t+1,min}, \omega_{t+1,min})$ and $(v_{t+1,max}, \omega_{t+1,min})$, respectively. And point $A$ is the end of $n$ step trajectories with input velocity $(v_{t+1,max}, \frac{\omega_{t+1,max}+\omega_{t+1,min}}{2})$. As mentioned above, the constraints can be formulated as a series of linear inequalities as (9), (10) and (11). In (11), $\omega_{t+1,mid} = 0.5(\omega_{t+1,max} + \omega_{t+1,min})$. Notice that we set the prediction horizon $n = 1$ in the Fig. 4 to make the concept of state window easier to understand. This approximation will lead to some loss compared to the original state window, but the loss is trivial. So The optimal solution obtained is, in fact, suboptimal for the original state window. If the solved solution fell outside the original state window $V_{s,t+n\Delta t}$, the inversely calculated velocity input would be out of the corresponding velocity scope $V_{d,t}$. On such occasions, we will confine the input into the velocity scope $V_{d,t}$.

$$V_{c,t+n} = \{\boldsymbol{x} | \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{B}\}, \tag{9}$$

$$\boldsymbol{A} = \begin{bmatrix} y_B - y_A & x_A - x_B \\ y_C - y_B & x_B - x_C \\ y_D - y_C & x_C - x_D \\ y_E - y_D & x_D - x_E \\ y_A - y_E & x_E - x_A \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} (y_B - y_A)x_A + (x_A - x_B)y_A \\ (y_C - y_B)x_B + (x_B - x_C)y_B \\ (y_D - y_C)x_C + (x_C - x_D)y_C \\ (y_B - y_D)x_D + (x_D - x_E)y_D \\ (y_A - y_E)x_E + (x_E - x_A)y_E \end{bmatrix} \tag{10}$$

$$\begin{cases} \boldsymbol{p}_A = \begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{v_{t+1,max}}{\omega_{t+1,mid}} \begin{bmatrix} \sin(\theta_t + \omega_{t+1,mid} n\Delta t) - \sin\theta_t \\ \cos(\theta_t + \omega_{t+1,mid} n\Delta t) - \cos\theta_t \end{bmatrix} \\ \boldsymbol{p}_B = \begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{v_{t+1,max}}{\omega_{t+1,max}} \begin{bmatrix} \sin(\theta_t + \omega_{t+1,max} n\Delta t) - \sin\theta_t \\ \cos(\theta_t + \omega_{t+1,max} n\Delta t) - \cos\theta_t \end{bmatrix} \\ \boldsymbol{p}_C = \begin{bmatrix} x_C \\ y_C \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{v_{t+1,min}}{\omega_{t+1,max}} \begin{bmatrix} \sin(\theta_t + \omega_{t+1,max} n\Delta t) - \sin\theta_t \\ \cos(\theta_t + \omega_{t+1,max} n\Delta t) - \cos\theta_t \end{bmatrix} \\ \boldsymbol{p}_D = \begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{v_{t+1,min}}{\omega_{t+1,min}} \begin{bmatrix} \sin(\theta_t + \omega_{t+1,min} n\Delta t) - \sin\theta_t \\ \cos(\theta_t + \omega_{t+1,min} n\Delta t) - \cos\theta_t \end{bmatrix} \\ \boldsymbol{p}_E = \begin{bmatrix} x_E \\ y_E \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \frac{v_{t+1,max}}{\omega_{t+1,min}} \begin{bmatrix} \sin(\theta_t + \omega_{t+1,min} n\Delta t) - \sin\theta_t \\ \cos(\theta_t + \omega_{t+1,min} n\Delta t) - \cos\theta_t \end{bmatrix} \end{cases} \tag{11}$$

*2) Cost Function Design:* We design a quadratic cost function to guide the robot to approach the goal and avoid collisions with moving pedestrians.

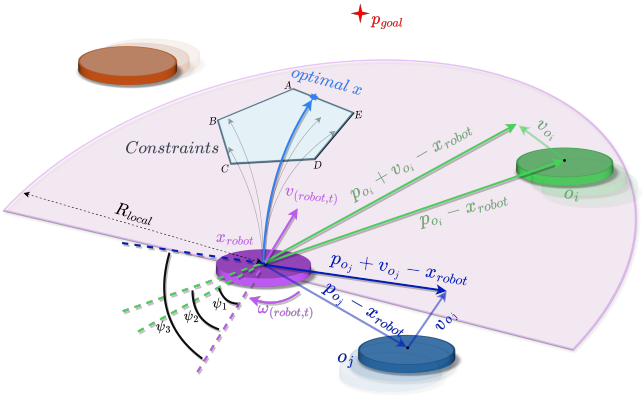



Fig. 5. Planning scene. The purple circle represents the robot and the pink semicircle is the local map for planning and $R_{local}$ represents the local map radius. The dark blue circle represents pedestrian $o_j$ whose relative velocity vector end $\boldsymbol{p}_{o_j} + \boldsymbol{v}_{o_j} - \boldsymbol{x}_{robot}$ is located in the local map. The green circle represents pedestrian $o_i$ whose velocity vector end $\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - x_{robot}$ is located in the local map. The orange circle represents the pedestrian of no interest in the planning. The light blue pentagon represents the approximated state window with the prediction horizon which forms the constraints of this optimization problem. The red star represents the position of the goal point and the blue star in the constraints is the optimal position solved in the approximated state window.

In (12) $\boldsymbol{x} \in \mathbb{R}^2$ is optimization variable representing the point in the state window, $\boldsymbol{p}_{goal} \in \mathbb{R}^2$ is the goal position. $\boldsymbol{p}_{o_i}$ and $\boldsymbol{v}_{o_i}$ are position and velocity vectors of pedestrian $i$. $\boldsymbol{p}_{robot} \in \mathbb{R}^2$ represents current robot position $(x_t, y_t)$. $w_1$ and $w_2$ are hyperparameters representing the weights of each linear term. $\sigma_i$ and $\gamma_i$, as defined in (13) and (14), are the signs of the cross-product terms, ensuring that the absolute values of these terms increase throughout the minimization process.

$$
\begin{aligned}
\mathrm{f}(\boldsymbol{x}) = & w_1 \|\boldsymbol{x} - \boldsymbol{p}_{goal}\|^2 \\
& + w_2 \sum_{o_i \in \mathcal{O}_v} \sigma_i \frac{(\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot})}{\|\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{p}_{robot}\|} \\
& + w_2 \sum_{o_j \in \mathcal{O}_p} \gamma_j \frac{(\boldsymbol{p}_{o_j} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot})}{\|\boldsymbol{p}_{o_j} - \boldsymbol{p}_{robot}\|}
\end{aligned} \tag{12}
$$

$$
\sigma_i = \begin{cases} -1, \ (\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot}) \geq 0 \\ \ \ 1, \ (\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot}) < 0 \end{cases} \tag{13}
$$

$$
\gamma_j = \begin{cases} -1, \ (\boldsymbol{p}_{o_j} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot}) \geq 0 \\ \ \ 1, \ (\boldsymbol{p}_{o_i} - \boldsymbol{p}_{robot}) \times (\boldsymbol{x} - \boldsymbol{p}_{robot}) < 0 \end{cases} \tag{14}
$$

$\mathcal{O}_v$ is a set consisting of pedestrians whose position vector relative to the robot plus velocity vector relative to the robot end in the local map, as defined in (15). Similarly, $\mathcal{O}_p$ is a set consisting of pedestrians whose relative position vector end is located in the local map, as defined in (16). For example, as illustrated in Fig. 5, the dark blue pedestrian $o_j$ is in the set $\mathcal{O}_v$ and the green pedestrian $o_i$ is in both sets, $\mathcal{O}_p$ and $\mathcal{O}_v$.

$$
\begin{aligned}
\mathcal{O}_v = \{o | \cos^{-1} \left( \frac{(\boldsymbol{p}_o + \boldsymbol{v}_o - \boldsymbol{p}_{robot}) \cdot \boldsymbol{v}_{robot}}{\|(\boldsymbol{p}_o + \boldsymbol{v}_o - \boldsymbol{p}_{robot})\| \|\boldsymbol{v}_{robot}\|} \right) \leq \frac{\pi}{2}, \\
\|\boldsymbol{p}_o + \boldsymbol{v}_o - \boldsymbol{p}_{robot}\| \leq R_{local}\}
\end{aligned} \tag{15}
$$

$$
\begin{aligned}
\mathcal{O}_p = \{o | \cos^{-1} \left( \frac{(\boldsymbol{p}_o - \boldsymbol{p}_{robot}) \cdot \boldsymbol{v}_{robot}}{\|(\boldsymbol{p}_o - \boldsymbol{p}_{robot})\| \|\boldsymbol{v}_{robot}\|} \right) \leq \frac{\pi}{2}, \\
\|\boldsymbol{p}_o - \boldsymbol{p}_{robot}\| \leq R_{local}\}
\end{aligned} \tag{16}
$$

In equation (15) and (16), $\boldsymbol{v}_{robot}$ represent the current velocity vector. The quadratic term $\|\boldsymbol{x} - \boldsymbol{goal}\|^2$ in the cost function (12) primarily serves to guide the robot closer to the target point. The cross-product terms $(\boldsymbol{x} - \boldsymbol{p}_{robot}) \times (\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{x}_{robot})$ and $(\boldsymbol{x} - \boldsymbol{p}_{robot}) \times (\boldsymbol{p}_{o_j} - \boldsymbol{x}_{robot})$ in the cost function (12) are mainly responsible for steering the robot away from pedestrian to avoid the collision. It is worth noting that we integrate the motion information of moving pedestrians by introducing the pedestrian velocities $v_{o_i}$ into the cost function. The predicted pedestrian velocities are obtained by the KF running on the robot, which is detailed in Section III-B. When the cost function is minimized, the absolute value of cross-product terms will be maximized for the negative coefficients. This will bring two aspects of influences to the navigation: 1) $\|\boldsymbol{x} - \boldsymbol{p}_{robot}\|$, which is proportional to the linear velocity input $v_{t+1}$, will be optimized as large as possible. This will lead the robot to move as efficiently as it can. 2) And the angles $\psi_1$, $\psi_2$, and $\psi_3$ as illustrated in Fig. 5 will be enlarged in the optimization process to avoid collisions with pedestrians. In addition, the two adaptive terms $\frac{1}{\|\boldsymbol{p}_{o_i} + \boldsymbol{v}_{o_i} - \boldsymbol{p}_{robot}\|}$ and $\frac{1}{\|\boldsymbol{p}_{o_i} - \boldsymbol{p}_{robot}\|}$ in (12) are used to differentiate the level of danger each pedestrian poses to the robot, and to assign higher weights to the cross-product terms related to more hazardous pedestrian. In other words, with the adaptive terms, the closer to the robot the pedestrian is, the more dangerous the one is, and the cross-product term related to this pedestrian will be assigned a higher weight. To avoid scenarios where the robot is very close to the target but is repelled by nearby stationary or slow-moving obstacles that are non-threatening, we apply a factor of $0.5$ to the obstacle weight $\omega_2$ as the robot approaches.

*3) Overall Optimization Problem Formulation:*

$$
\min_{\boldsymbol{x}} \ \mathrm{f}(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{B} \tag{17}
$$

In (17) the $\mathrm{f}(\boldsymbol{x})$ is (12) and the constraints are (9). The cost function in (17) is in quadratic form and the constraints are composed of a series of linear inequalities. So this optimization problem is a Quadratic Programming (QP) problem, which can be solved very efficiently. After the optimal state at $(t+n\Delta t)$ is obtained by solving the QP problem above, we can inversely calculate the corresponding optimal velocity input $(v_{t+1}, \omega_{t+1})$ uniquely. The overall process of the proposed algorithm is shown as Algorithm 1. The local obstacle set $\mathcal{O}_l$ in Algorithm 1 consists of the obstacles whose distance to the robot is less than the local map radius.

**Algorithm 1:** The Proposed Planner

**Input:** $[x_t, y_t, \theta_t, v_t, \omega_t]$, $\boldsymbol{p}_{goal}$,$V_{d,t}$,prediction horizon $n$, local obstacle set $\mathcal{O}_l$
**Result:** Optimal control inputs $(v^*_{t+1}, \omega^*_{t+1})$
// Initialization
$\mathcal{O}_v$, $\mathcal{O}_p$ *constraint*, *objective*;
**for** *obs in* $\mathcal{O}_l$ **do**
  **if** $\langle \boldsymbol{p}_{obs} - \boldsymbol{p}_{robot}, \boldsymbol{v}_{robot} \rangle \leq \frac{\pi}{2}$ **then**
    $\mathcal{O}_p.append(obs)$
  **if** $\langle \boldsymbol{p}_{obs} + \boldsymbol{v}_{obs} - \boldsymbol{p}_{robot}, \boldsymbol{v}_{robot} \rangle \leq \frac{\pi}{2}$ **then**
    $\mathcal{O}_v.append(obs)$
// Optimization Prolem Solving
$\hat{x}, \hat{y} \leftarrow OptiVariable()$;
Calculate $A_n, B_n, C_n, D_n, E_n$ from $VelocityScope$;
$V_c = ConvexHull(A_n, B_n, C_n, D_n, E_n)$;
**for** *edge of* $V_c$ **do**
  $constraint.append(edge$ inequality$)$
**for** *o in* $\mathcal{O}_v$ **do**
  $objective.append(o$ related term$)$
**for** *o in* $\mathcal{O}_v$ **do**
  $obj.append(o$ related term$)$
$objective.append(\boldsymbol{p}_{goal}$ related term$)$;
$(\hat{x}^*_{t+n}, \hat{y}^*_{t+n}) \leftarrow solve(constraint, obj)$;
$(v^*_{t+1}, \omega^*_{t+1}) \leftarrow inverseCalc(\hat{x}^*_{t+n}, \hat{y}^*_{t+n}, n)$;
$v^*_{t+1} \leftarrow min\left(v_{max}, max(v^*_{t+1}, v_{min})\right)$;
$\omega^*_{t+1} \leftarrow min\left(\omega_{max}, max(\omega^*_{t+1}, \omega_{min})\right)$;
**return** Optimal control input $(v^*_{t+1}, \omega^*_{t+1})$;

### *B. Pedestrian Velocity Prediction*

To capture the dynamic movement information of pedestrians for the planning, we employ Kalman Filter[14] to predict the velocity of pedestrians that are in the local map, which refers to pedestrians whose distance from itself to the robot is less than the local map radius $R_{lcoal}$. The state of pedestrians can be represented as $\boldsymbol{X} = [x, y, \dot{x}, \dot{y}]^T$. Due to the small

velocity change between every two small time steps, we can model the motion of pedestrians as uniform motion as shown in (18). Based on the assumption above, we can formulate the problem of pedestrian motion prediction as below,

$$\boldsymbol{X}_{t+1} = \boldsymbol{A}\boldsymbol{X}_t + \boldsymbol{W}_t \quad (18)$$

$$\boldsymbol{Z}_{t+1} = \boldsymbol{H}\boldsymbol{X}_{t+1} + \boldsymbol{V}_t \quad (19)$$

where (18) is the state transition equation and (19) is the observation equation. The $\boldsymbol{X}_t$ is the pedestrian state in time step $t$, and $\boldsymbol{Z}_{t+1}$ is the observation vector which contains the predicted position information. $\boldsymbol{W}_t$ and $\boldsymbol{V}_t$ is the noise vector characterized by a Gaussian distribution. The state transition matrix $\boldsymbol{A}$ and observation matrix $\boldsymbol{H}$ are defined as follows.

$$\boldsymbol{A}_{4\times4} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{H}_{2\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

With the pedestrian velocity prediction module, we can monitor the pedestrian positions and predict the velocity once the pedestrian steps into the local map according to the monitored historical positions. The predicted velocity will be integrated into the planning frame to enhance the capability of dynamic collision avoidance and empower safe crowd navigation.

## IV. Experiments

### A. Experiment Setup

We carried out 10 sets of experiments: half were conducted using the classical DWA method with 3, 6, 9, 12, and 15 pedestrians respectively, while the remaining sets employed our proposed method, also with different pedestrian counts. Each experimental set comprised 100 trials featuring pedestrian trajectories randomly chosen from the ETH BIWI dataset. Table II shows the geometry parameters setting for the series of experiments. The robot radius is 0.5 m and the radius of the pedestrians is 0.5 m. As for the hyperparameters of our proposed method, we set the prediction horizon $n = 15$. The weight of quadratic term $w_1 = 1$, and the weight of collision avoidance-related cross-product term $w_2 = 20$. In addition, the control frequency is $10Hz$ which means the time period $\Delta t = 0.1\ s$. Code implementation of simulation refers to [17].

TABLE II
Geometry parameters and robot dynamics parameters

| | Parameters | Value |
|---|---|---|
| Geometry | Robot radius | $0.5\ m$ |
| | Pedestrians radius | $0.5\ m$ |
| | Environment size | $16\ m \times 14\ m$ |
| | Local map radius | $6\ m$ |
| Robot Dynamics | Max Linear Velocity | $1.5\ m \cdot s^{-1}$ |
| | Min Linear Velocity | $0\ m \cdot s^{-1}$ |
| | Max Angular Velocity | $0.22\ \pi\ rad \cdot s^{-1}$ |
| | Min Angular Velocity | $-0.22\ \pi\ rad \cdot s^{-1}$ |
| | Max Linear Acceleration | $0.2\ m \cdot s^{-2}$ |
| | Max Angular Acceleration | $0.22\ \pi\ rad \cdot s^{-2}$ |

### B. Pedestrian Movements Simulation

To simulate the pedestrian movements as close to scenes as possible, we employ the ETH BIWI Walking Pedestrians dataset[18] consisting of walking pedestrians in busy scenarios from a bird's eye view. In each experimental group, we select a fixed quantity of pedestrian trajectories from the dataset at random, aiming to mimic pedestrian movements with high fidelity. Pedestrian trajectories in the ETH BIWI dataset are generally aligned along the y-direction. In each experiment, some trajectories were rotated by 90 degrees to simulate pedestrian motion in an open world better. This approach ensures the realism of pedestrian behavior while maintaining the variability of trajectories across trials within each experimental group. The speed of walking pedestrians stays in the range of $0.8\ m/s \sim 1.5\ m/s$, and there are some stationary pedestrians in the dataset.

### C. Experiment Results

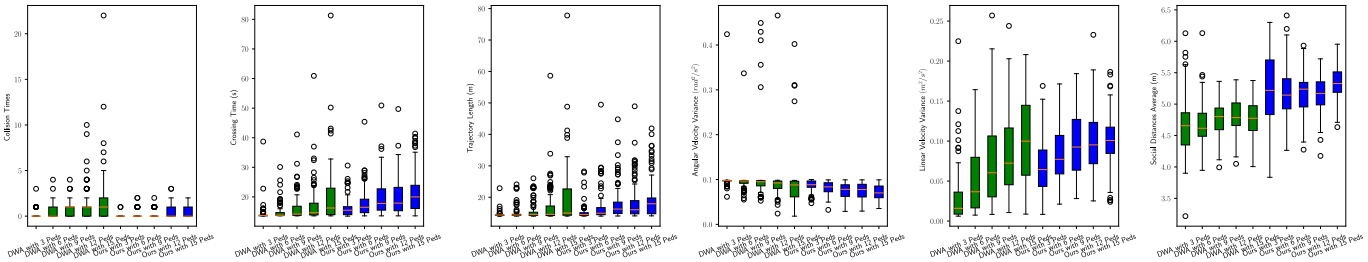
Table III presents a quantitative comparison between experiments using the DWA and the proposed method under different conditions, specifically with varying numbers of pedestrians (Peds.), whereby the success rate is the ratio of collision-free trials over all the trials, and the number of collisions is the sum of all the collisions in the 100 trials. Travel time refers to the time that the robot takes to travel from the starting point to the destination in every single trial. Similarly, travel distance is the length of the final trajectory. In addition, we propose a concept of the indicator called *Average Social Distance* generally represents how far from the robot to the obstacles is and it is defined as (20), where $sd$ represents the social distance and $\mathcal{O}_{l,t}$ represent the set of obstacles in the local map at time step $t$. The angular velocity variance and linear velocity variance here refer to the variance of angular velocity data and linear velocity data of all time steps during each travel.

TABLE III
Quantitative results of comparison between experiments with DWA and proposed method

| | No. of Peds. | 3 | 6 | 9 | 12 | 15 |
|---|---|---|---|---|---|---|
| Success Rate | DWA | 84% | 63% | 49% | 42% | 28% |
| | **Ours** | 98%(16.7% ↑) | 89%(41.3% ↑) | 77%(57.1% ↑) | 71%(69.0% ↑) | 65%(132.1% ↑) |
| No. of Collisions | DWA | 18 | 49 | 71 | 119 | 176 |
| | **Ours** | 2 | 13 | 24 | 35 | 40 |
| Mean of Travel Time ($s$) | DWA | 14.22 | 14.86 | 16.23 | 17.33 | 19.64 |
| | **Ours** | 16.19 | 18.06 | 19.78 | 19.77 | 21.36 |
| Mean of Travel Distance ($m$) | DWA | 14.46 | 14.97 | 15.72 | 17.13 | 19.12 |
| | **Ours** | 15.23 | 16.86 | 18.09 | 18.40 | 19.34 |
| Mean of Angular Velocity Variance ($10^{-2}\ rad^2/s^2$) | DWA | 9.94 | 9.51 | 10.41 | 8.90 | 8.85 |
| | **Ours** | 8.70(12.5% ↑) | 8.05(15.4% ↑) | 7.51(27.9% ↑) | 7.55(15.17% ↑) | 7.08(20.0% ↑) |
| Mean of Linear Velocity Variance ($10^{-2}\ m^2/s^2$) | DWA | 2.99 | 5.36 | 7.58 | 8.41 | 10.07 |
| | **Ours** | 6.93 | 8.35 | 9.68 | 9.93 | 9.99 |
| Mean of Average Social Distance ($m$) | DWA | 4.66 | 4.69 | 4.76 | 4.81 | 4.77 |
| | **Ours** | 5.23(12.2% ↑) | 5.17(10.2% ↑) | 5.18(8.8% ↑) | 5.17(7.5% ↑) | 5.33(11.7% ↑) |

$$sd = \frac{1}{T}\sum_{t}^{T}\frac{\sum_{o_i \in \mathcal{O}_{l,t}} \|\boldsymbol{p}_{o_i} - \boldsymbol{p}_{robot}\|}{|\mathcal{O}_{l,t}|} \quad (20)$$

From Table III, our method demonstrates a substantial increase in success rate across all the experiment groups compared to DWA. Our method outperforms more significantly with the number of pedestrians increasing, as the dark blue line



(a) Collisions (b) Travel time (c) Trajectory length (d) Angular velocity variance (e) Linear velocity variance (f) Average social distance

Fig. 6. Distribution of the quantitative results. The green box and blue box show the results of experiments with DWA and our method, respectively.

and blue line shown in Fig. 7. As the environment becomes more challenging with 12 and 15 pedestrians, the proposed algorithm's success rates are 71% and 65%, respectively, compared to significantly lower rates of 42% and 28% for DWA, which shows 60.9% and 132.1% improvements. In addition, our method consistently results in fewer collisions across all tested scenarios with varying pedestrian densities and shows more significant improvements in more challenging environments. Our proposed method's superiority is also evident in the aspect of average social distance as indicated by *Mean of Average Social Distance* in Table III, demonstrating 7.5%∼12.2% improvements. It generally shows that the robot employing our method will try to keep more distance from the pedestrian which makes the pedestrians feel safer. Besides, our proposed method appears more stable in angular velocity and demonstrates approximately 12.5%∼20.0% improvement compared to DWA across the experiments.

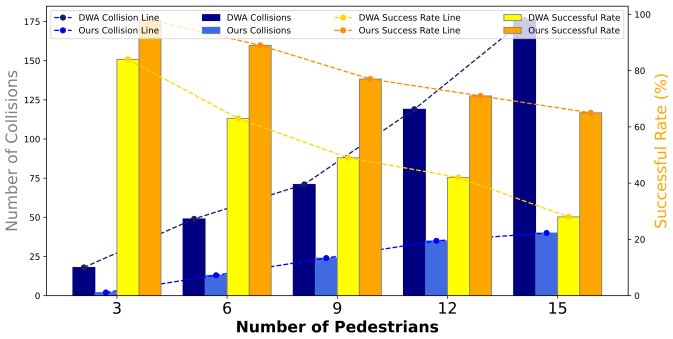



Fig. 7. Comparison for collision times and success rate between DWA and proposed method

The set of boxplots in Fig. 6 illustrates the distribution of quantitative results from experiments employing DWA and our proposed method across scenarios with varying numbers of pedestrians, whereby the green boxes represent results from experiments with DWA and the blue ones represent results from experiments with the proposed method. Fig. 6(a) shows that our proposed method has fewer collisions and more stable performance, indicating a more effective collision avoidance capability. Fig. 6(c) displays that our method produces more consistent trajectory lengths, particularly noticeable in scenarios with more pedestrians, suggesting stable performance in complex environments. Fig. 6(f) depicts the average social distance maintained during navigation. Our method not only maintains a larger average distance across all scenarios but also exhibits less variance, indicating a consistent ability to keep safe distances from pedestrians. Overall, Fig. 6 shows that our proposed method can implement more effective safe navigation with more consistent and stable performance in environments with a high density of pedestrians.

### D. Analysis for Computational Burden

DWA finds the optimal velocity by evaluating the sampled admissible velocity pairs with a cost function, which leads to a heavy computational burden. Also, there is a dilemma

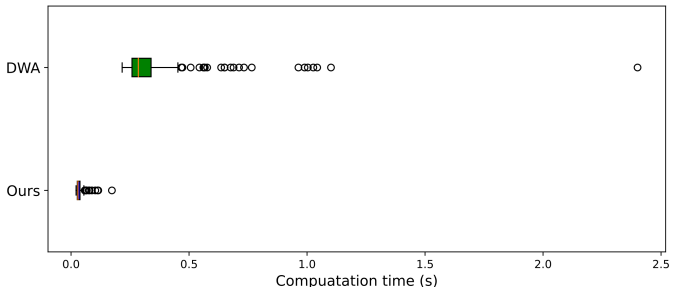



Fig. 8. The distribution of computation time for each optimal input-solving process with DWA and our proposed method. The means of the computational time of DWA and the proposed method are $0.371\ s$ and $0.035\ s$, the standard deviations of both are $0.018\ s$ and $0.254\ s$ respectively. respectively.

between efficiency and accuracy, which means finer resolution results in lower efficiency. However, our method approximates the original problem as a convex optimization problem that can be solved very efficiently. In the experiments, the convex optimization problem is solved by the ECOS solver of CVXPY package[19]. We conduct a simulation experiment on MacOS with 2.3GHz Intel Core i5 CPU, and Fig. 8 shows the distribution of the computational time of each solving for optimal velocity input with DWA and our methods respectively, which illustrates the computational burden of our method is much lower and the distribution of that is denser and the computation time has decreased by an order of magnitude.

Overall, our proposed method is much more computationally efficient compared to DWA.

## V. Conclusion And Future Work

In this work, we introduced a novel local motion planning algorithm for differentially driven mobile robots, inspired by DWA, yet significantly overcoming safety risks in pedestrian-rich environments. Utilizing the features of differential drive kinematics, our approach employs convex optimization to efficiently find optimal velocity inputs that account for pedestrian movements predicted by the KF. Through rigorous and extensive simulation experiments, including scenarios with varying pedestrian densities using data from the ETH BIWI pedestrian dataset, our method consistently outperformed DWA, showcasing higher success rates in dense environments and reduced collisions. Furthermore, it maintained greater average social distances, enhancing pedestrian safety. We also demonstrated a substantial reduction in computational time compared to DWA, emphasizing our method's efficiency.

Despite outperforming DWA in crowd navigation data, our approach has limitations, including the absence of a safety guarantee and the potential loss of admissible velocity inputs due to the convexification of the state window. Future efforts will aim to incorporate safety guarantees and consider pedestrians' reactions within the motion planning framework.

## References


[1] 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.

[2] T. Kretz, J. Lohmiller, and P. Sukennik, "Some indications on how to calibrate the social force model of pedestrian dynamics," *Transportation research record*, vol. 2672, no. 20, pp. 228–238, 2018.

[3] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[4] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.

[5] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*, IEEE, 2019, pp. 6015–6022.

[6] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 856–892, 2020.

[7] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[8] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[10] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6272–6281.

[11] K. Matsumoto, A. Kawamura, Q. An, and R. Kurazume, "Mobile robot navigation using learning-based method based on predictive state representation in a dynamic environment," in *2022 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2022, pp. 499–504.

[12] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 007–10 013.

[13] S. Liu, P. Chang, Z. Huang, *et al.*, "Intention aware robot crowd navigation with attention-based interaction graph," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 12 015–12 021.

[14] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems [j]," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[15] Z. Jian, Z. Yan, X. Lei, *et al.*, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3679–3685.

[16] T. Hellström, *Kinematics equations for differential drive and articulated steering*. Department of Computing Science, Umeå University, 2011.

[17] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, *Pythonrobotics: A python code collection of robotics algorithms*, 2018. eprint: arXiv:1808.10703.

[18] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th international conference on computer vision*, IEEE, 2009, pp. 261–268.

[19] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.