Safe Navigation of Mobile Robots in Dense Human Environments using Control Barrier Functions

by

Nanami Hashimoto

Department of Cognitive Robotics

To obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly on Monday August 5, 2024 at 10:00 AM.

Student number:	4779495	
Project duration:	December 1, 2023 – August 5, 2024	
Thesis committee:	Dr. Chris Pek	TU Delft Cognitive Robotics, supervisor
	Dr. Luka Peternel	TU Delft Cognitive Robotics
	Dr. Meichen Guo	TU Delft Systems & Control

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Safe Navigation of Mobile Robots in Dense Human Environments using Control Barrier Functions

Nanami Hashimoto
Department of Cognitive Robotics
Delft University of Technology, the Netherlands
Email: N.Hashimoto@student.tudelft.nl

***Abstract*—Autonomous mobile robots are increasingly performing tasks in our daily environments, e.g., cleaning offices or order picking in supermarkets. In such human-populated scenarios, it is crucial that these robots always navigate safely when performing their task. Existing state-of-the-art methods can ensure safety, but often require deterministic motions of humans and may lead to conservative behavior in dense human-populated environments. This study investigates the use of time-varying control barrier functions (CBFs) and time-based rapidly exploring random trees (RRTs) to combine local safety with global task performance. The proposed new cost function improves the trade-off of safety and task progress in densely populated scenarios. Our results show that time-varying CBFs can perform better in terms of both task performance and safety compared to normal CBFs. Furthermore, our real-world robot experiments validate our approach to a physical nonholonomic robot.**

I. Introduction

With the rapid progression of autonomous robot technology, autonomous robots increasingly coexist with humans. Examples are settings like hospitals [7], retail [4] and catering [9]. These robots are anticipated to yield beneficial impacts on society, such as addressing labour shortages and collaborating on intricate tasks. Since these robots are expected to share their workspace with humans, possibly at a relatively close distance, safe navigation is of paramount importance. An example scenario where a mobile robot navigates around multiple humans is illustrated in Fig. 1.

Traditional approaches for safe robot navigation in human-populated environments focused on collision avoidance methods, such as the dynamic window approach [17], artificial potential fields [20], and velocity obstacles [6]. Although these approaches showed the successful deployment of robots in applications such as museum tour guides [21], their behaviours often come at the cost of task performance. For example, in a crowded or confined environment, a pessimistic robot might choose to freeze, assuming that all space could be occupied by humans. This issue is known as the freezing robot problem [22] and prevents the robot from completing its task execution.

Furthermore, when the robot interacts with humans, merely preventing harm to humans by avoiding collision (i.e. ensuring physical safety) is not sufficient. For the robot to be accepted in a human environment, the robot has to also be perceived as safe. Perceived safety in human-robot interaction is described as the "user's perception of the level of danger when interacting with a robot, and the user's level of comfort during the interaction" [3]. Even when physical risks are minimal or absent, people's perceived safety can be influenced by various factors, including the context of use, comfort, familiarity, predictability, transparency, a sense of control, and trust [1]. One of the pioneering works by [18] first modelled situations where people feel uncomfortable with the robot's navigation, such as appearing from a hidden area, and proposed a planner avoiding paths that could lead to such situations. However, this study is limited to static humans, and ensuring the overall safety of humans in a dynamic environment remains a challenge.

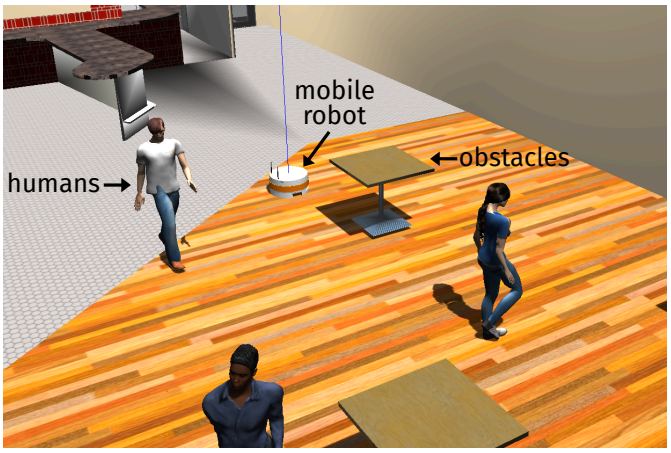


Fig. 1: Example scenario, where a mobile robot is tasked to safely navigate around multiple moving humans.

An emerging tool to address this safety challenge of robot control is CBFs [2]. CBFs encode safety requirements, such as distance to obstacles, to define a safe set for the system. Then, by permitting only inputs that maintain the system within the safe set, synthesizing the controller with CBFs guarantees the safety of the system. CBFs are computationally light and scalable compared to other safe set-based methods such as reachability analysis [24]. It has been applied to various applications such as adaptive cruise control, bipedal robot walking, dynamic balancing of Segways, and swarm control [2] [5]. Furthermore, the recent work [25] has developed a promising approach to synthesize CBFs that simultaneously ensure physical safety and optimize perceived safety in human-drone interaction applications by learning from human feedback. However, the application of CBFs to human-robot

interaction requires further research. Application for mobile robots in a dynamic human environment is a challenge due to the complexity of the human movement and the nonholonomic dynamics of many mobile robots.

In this study, I investigate the use of CBFs to ensure safe navigation of nonholonomic mobile robots in dense human-populated environments, such as office spaces. I build this work upon the state-of-the-art CBF time-based RRT (CBF-TB-RRT) planners. In this paper, I

1) investigate the performance of CBF-TB-RRTs in dense human-populated environments;
2) propose a trade-off function to balance safety and performance in dense environments;
3) compare normal CBFs, time-varying CBFs and baseline algorithm from ROS1 navigation stack in simulated human-populated environment;
4) validate the algorithm in real-world TIAGo robot

II. Control Barrier Functions

This study investigates the use of CBFs for safe navigation of mobile robots around humans. This section introduces the concept of CBFs and how they can be used to ensure safety.

A. Fundamentals of CBFs

Consider a robot motion model with nonlinear affine control dynamics shown in Equation 1:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1}$$

with f and g are locally Lipschitz functions, $x(t) \in D \subset \mathbb{R}^n$ is the state of robot at time t and $u \in U \subset \mathbb{R}^m$ is the set of admissible control inputs. Consider set $\mathcal{C}$ defined as the superlevel set of a continuously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$, in other words $\mathcal{C} = \{x \in D \subset \mathbb{R}^n : h(x) \geq 0\}$. When $h(x)$ encodes desired safety criteria and all unsafe states x have $h(x) < 0$, the set $\mathcal{C}$ is referred to as the safe set. Having the safe set defined, the system is safe with respect to the set $\mathcal{C}$ if the set $\mathcal{C}$ is forward invariant, i.e. not leaving the safe set once the system's state is in that set.

Definition of CBFs. Let $\mathcal{C} \coloneqq \{x \in D : h(x) \geq 0\}$ be the superlevel set of continuously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$, then h is a CBF if there exists an extended class $\mathcal{K}_\infty$ function α such that for the control system (1):

$$\sup_{u \in U} [L_f h(x) + L_g h(x) u] \geq -\alpha(h(x)) \tag{2}$$

for all $x \in D$. The extended class $\mathcal{K}_\infty$ function α is an extension of class $\mathcal{K}$ functions defined in $\alpha : (-\infty, \infty) \to (-\infty, \infty)$ and additionally satisfies $\lim_{r \to \infty} \alpha(r) = \infty$.
Therefore, the set of control values that renders $\mathcal{C}$ invariant is:

$$K_{\text{cbf}}(x) = \{u \in U : L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0\} \tag{3}$$

To ensure system safety, the controller is synthesized with the CBF. Finding the optimal control input, which minimally affects the task performance while guaranteeing safety, entails formulating the control task as an optimization problem having the CBF requirement as its constraint. Since the safety condition given in Equation 3 is affine, this can be given by quadratic programming (QP). :

$$\begin{aligned} u(x) = \underset{(u,\delta)^T \in \mathbb{R}^{m+1}}{\operatorname{argmin}} \quad & \frac{1}{2}(u - u_{ref})^T H(x)(u - u_{ref}) \\ \text{s.t.} \quad & L_f h(x) + L_g h(x) u \geq -\alpha(h(x)) \end{aligned} \tag{4}$$

where u_{ref} is the reference input that drives the robot toward the goal and $H(x)$ is any positive definite matrix. The method to obtain u_{ref} is explained in section IV.

B. Time-varying control barrier functions

A variant of CBFs that is useful in a dynamic environment is a time-varying CBFs [12]. The formulation of time-varying CBFs is presented in 5, which updates the formulation of the normal CBFs given in equation 2.

$$\sup_{u \in U} [L_f h(x,t) + L_g h(x,t) u + \frac{\partial h(x,t)}{\partial t}] \geq -\alpha(h(x,t)) \tag{5}$$

The difference between the normal CBFs and time-varying CBFs is the added term $\frac{\partial h(x,t)}{\partial t}$. This term shows how the change in the position of the obstacle affects the allowed control input. In the context of obstacle avoidance, this added term works in a way that it relaxes the constraint when the obstacle is moving away from the robot, while it tightens the constraint when the obstacle is approaching the robot. For the control barrier function used in this paper as shown in Equation 10, $\frac{\partial h(x,t)}{\partial t}$ is given as follows.

$$\frac{\partial h(x,t)}{\partial t} = 2(x_p - x_o(t))(-\frac{\partial x_o(t)}{\partial t}) + 2(y_p - y_o(t))(-\frac{\partial y_o(t)}{\partial t}) \tag{6}$$

III. CBFs for Nonholonomic Mobile Robots

In many real-world applications, nonholonomic robots are commonly employed due to their practical utility in navigation and maneuvering within various environments. In this implementation, the unicycle model is considered for the robot dynamics:

$$\dot{x}(t) = g(x(t))u(t) = \begin{bmatrix} cos(\theta(t)) & 0 \\ sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} u(t), \tag{7}$$

where states are $x(t) = [p_x(t), p_y(t), \theta(t)]^T \in \mathcal{X} \subseteq \mathbb{R} \times [-\pi, \pi))$ and control input vector is $u(t) = [v(t), \omega(t)]^T \in \mathcal{U} \subseteq \mathbb{R}^2$. The state $x(t) = [p_x(t), p_y(t), \theta(t)]^T$ denote the position and the orientation of the robot, and the control inputs $v(t)$ and $\omega(t)$ represent the robot's linear and angular velocity.

Given the robot's dynamics and that the distance between the robot and humans must be positive, the continuously differentiable function $h(x)$ can be designed as:

$$h(x) = \|[p_x, p_y]^T - x_o\|_2^2 - (r_o + r_r)^2 \tag{8}$$

where p_x, p_y is the position of the robot, $x_o(t)$ is the state of the obstacle, r_o is the radius of the obstacle and r_r is the radius of the robot.

The function $h(x)$ has a relative degree of one with respect to the input $v(t)$ and a relative degree of two with respect to the input $\omega(t)$. Thus, when deriving the safety constraint shown in Equation 2, $\omega(t)$ does not appear either in term $L_f h(x)$ and $L_g h(x)$ of the equation. In such a case, angular velocity cannot contribute to satisfy the constraint, resulting in a situation where obstacle avoidance is solely done by changing the linear velocity. Since this is not a desirable approach for robot control, motivated by the approach in [14], the following coordinate transformation is introduced:

$$\tilde{x}(t) = x(t) + lR(\theta(t))e_1, \tag{9}$$

where l is a small positive constant, $e_1 = [1, 0, 0]^T$, and

$$R(\theta(t)) = \begin{bmatrix} cos(\theta(t)) & -sin(\theta(t)) & 0 \\ sin(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Then, because the maximum possible deviation between x and $\tilde{x}$ is length l, the Equation 8 is modified to Equation 10 to account for the error.

$$h(\tilde{x}) = \|[p_x, p_y]^T - x_o\|_2^2 - (r_o + r_r + l)^2 \tag{10}$$

IV. CBF-BASED TB-RRT

Inspired by the work [14], a time-based RRT method is employed for robot planning. Time-based RRTs (TB-RRTs) are a variation of RRT motion planning algorithms that are particularly useful in dynamic environments. This algorithm expands a tree in real-time during the specified time or until specified numbers of nodes are obtained within each cycle. From the grown tree, they extract a partial path iteratively until the robot reaches the goal. In each cycle, the algorithm selects the best vertex from the grown tree according to its cost and executes the velocity command of the corresponding path from the current position to the selected vertex. The cost of the vertex depends on the distance to the goal and $h(x)$ value defined in Equation 10 at that vertex.

Algorithm 1 CBF-TB-RRT

Input: $\mathcal{X}_g$, T_s, N_s
Output: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, σ

$t_0 \leftarrow$ CLOCK()
$\mathbf{x}(t_0) \leftarrow$ ROBOT-POSE()
$\sigma \leftarrow \mathbf{x}(t_0)$, $\mathcal{G} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$
while $\mathbf{x}(t_0) \notin \mathcal{X}_g$ **do**
  while CLOCK() $< t_0 + T_s$ **do**
    GROW($\mathcal{G}, \mathcal{X}_o^d(t), N_s$)
  end while
  $\mathbf{u}(t_0) \leftarrow$ EXTRACT-CONTROL($\mathcal{G}$)
  EXECUTE-CONTROL($\mathbf{u}(t_0)$)
  $t_0 \leftarrow$ CLOCK()
  $\mathbf{x}(t_0) \leftarrow$ ROBOT-POSE()
  $\sigma \leftarrow \mathbf{x}(t_0)$
end while

Algorithm 2 GROW

Input: $\mathcal{G}, \mathcal{X}_o(t), N_s$

$\mathcal{V} \leftarrow \emptyset$, $\mathcal{E} \leftarrow \emptyset$
$v_v \leftarrow$ VERTEX-SAMPLE($\mathcal{G}, p_v$)
$\mathbf{x}_{rand} \leftarrow$ STATE-SAMPLE(v_v, p_s)
$\mathbf{u}_{ref} \leftarrow$ REF-SAMPLE($\mathbf{x}_{rand}, p_u$)
$\mathbf{U}_{seg}, \mathbf{X}_{seg}, \mathbf{x}_{new}, t_{new} \leftarrow$ STEER($\mathbf{x}_{rand}, \mathbf{u}_{ref}, N_s, v_v$)
if $\mathbf{x}_{new} \neq \emptyset$ **then**
  $z_{new} \leftarrow$ COST-CALCULATOR($\mathbf{x}_{new}$)
  $\mathcal{V} \leftarrow (\mathbf{x}_{new}, z_{new}, t_{new})$, $\mathcal{E} \leftarrow (\mathbf{X}_{seg}, \mathbf{U}_{seg})$
  $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$
end if

Algorithms 1 and 2 summarize the planning algorithm and tree growth procedure. For a duration T_s (or until the number of maximum nodes is reached), the algorithm grows a tree from the robot's current position to obtain a partially grown tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of vertices and edges, respectively. In TB-RRT, each vertex of the tree contains the robot's state $x(t)$, the cost of the vertex z, and the vertex timestamp t. Within the state $x(t) = [p_x(t), p_y(t), \theta(t)]^T$, $p_x(t)$ and $p_y(t)$ (position of the robot) are fixed to each vertex, but the heading angle $\theta(t)$ is a free state variable, meaning that the vertex is not dependent on the heading angle. The tree expansion in TB-RRT is done by sampling over this free variable of the state. This process takes the following steps:

1) VERTEX-SAMPLE($\cdot$) samples a vertex v_v over the existing vertices of tree $\mathcal{G}$ using uniform distribution p_v. This step essentially determines the vertex of the existing tree from which the new edge extends.
2) STATE-SAMPLE($\cdot$) samples an updated state x_{rand} of the vertex v_v by sampling the new free state variable θ. Gaussian distribution $p_s(\theta)$ given in Equation 11 is used for this sampling. θ_g is the heading angle toward the goal and Ω_θ is the standard deviation, which determines how exploratory the tree is.

$$p_s(\theta) = \frac{1}{\sigma_\theta \sqrt{2\pi}} \text{exp}(-\frac{(\theta - \theta_g)^2}{2\sigma_\theta^2}), \tag{11}$$

3) REF-SAMPLE($\cdot$) samples a reference input $u_{ref} = [v_{ref}, \omega_{ref}]^T$. v_{ref} is a maximum allowed speed and ω_{ref} is selected as $\omega_{ref} = a_\omega(\theta - \theta_g)$, where a_ω is the weight of angular difference between the current theta and sampled theta.
4) STEER($\cdot$) solves a sequence of quadratic programming problems (QPs) (Equation 4) to find a sequence of control $\mathbf{U}_{seg}$ that satisfies the CBF-based safety constraints while minimally deviating from the reference input u_{ref}. This ensures that each control input u adheres to the safety requirements while following the desired trajectory as much as possible. The input sequence $\mathbf{U}_{seg}$ consists of N_s steps of input u obtained from the sequence of QPs. An illustration of u_{ref} and the sequence of u is provided in Figure 2. $\mathbf{U}_{seg}$ creates a safe and dynamically feasible path segment $\mathbf{X}_{seg}$

that has N_s intermediate steps. The new state x_{new} is obtained at the end of the segment $\mathbf{X}_{seg}$.

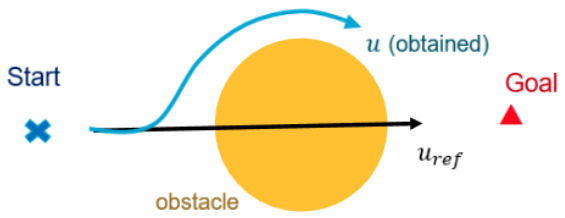


Fig. 2: Illustration of the STEER function, which finds the safe and dynamically feasible input u from the reference input

Once the new state x_{new} is obtained, the COST-CALCULATOR($\cdot$) calculates the cost of that vertex using Equation 12. dist($\cdot$) represents the distance between $\mathbf{x}_{new}$ and the goal set $\mathcal{X}_g$, $h(\mathbf{x}_{new})$ is the safety value of the state, and a_{cost} is the weight determining the trade-off of between the safety and closeness to the goal. This cost function rewards proximity to the goal and penalizes proximity to obstacles.

$$z_{new} = \text{dist}(\mathbf{x}_{new}, \mathcal{X}_g) - a_{cost} h(\mathbf{x}_{new}) \tag{12}$$

This calculated cost, as well as the state and the time at the new vertex, the path segment $\mathbf{X}_{seg}$ and the corresponding control input sequence $\mathbf{U}_{seg}$ are stored in the tree.

V. RESULT

This section presents the experimental results of static obstacle avoidance in a simulated environment, navigation in a simulated dynamic human environment, and testing in a static real-world environment. The algorithm is implemented in Python and the simulation is run by a laptop equipped with a 12th Gen Intel Core i7-12650H processor and NVIDIA GeForce RTX 4050 GPU.

A. CBF-constrained tree expansion

The result of tree growth under the CBF constraints over a single cycle is shown in Figure 3. The robot's initial position and orientation are set at $(p_x, p_y, \theta) = (0, 0, 0)$ (the robot heading right in the figure) and the centre of the goal set is positioned at $(p_x, p_y) = (6, 6)$. The blue dots indicate the tree's vertices and the green lines and small dots show the path segments with the intermediate points, respectively. The algorithm grows a tree toward the goal direction while avoiding the obstacle regions.

In this simulation, the time interval between each change in control input is set to 0.1 [s] and each path segment has $N_s = 10$ intermediate steps. This tree contains 30 vertices, and the time taken to generate the tree was 0.189 [s]. One can see that the algorithm grows the tree while avoiding the obstacle region. The α in Equation 4 is set to $\alpha = 10$ for this tree. It is also verified that the higher α value leads to the more aggressive yet still safe path generation, allowing the tree to put the vertices closer to the obstacles.

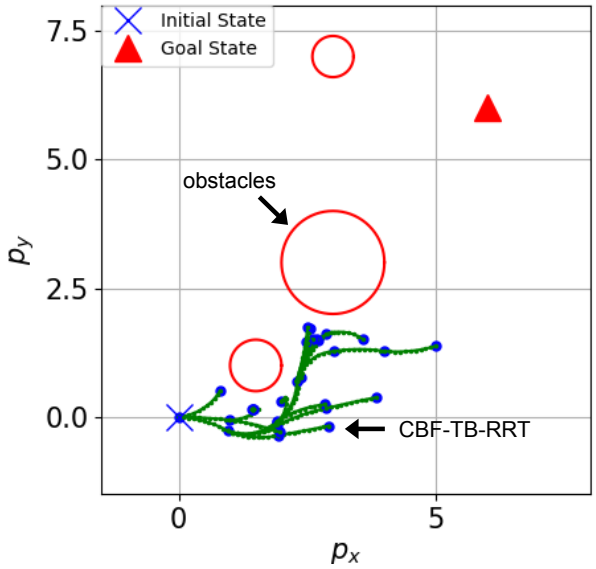


Fig. 3: Visualization of the CBF-TB-RRT for an example scenario, where the robot grows a tree avoiding obstacle regions from the initial state towards the goal.

B. Robot simulation in dynamic environment

The CBF-TB-RRT navigation algorithm is implemented and executed in a high-fidelity pmb2 (TIAGo base) simulator [16]. The simulator is based on ROS1 and Gazebo. The testing scenario in a dynamic human environment is shown in Figure 4. Pedsim ROS [19] was used to simulate pedestrian movement, which is modelled by the social force model [11]. The robot's start is set at $(p_x, p_y, \theta) = (0, 0, 0)$ (with the robot heading upward in the figure), and the goal region is a circle with a radius of 0.5 [m], positioned at $(p_x, p_y) = (15, 0)$ named as "goal 1", and $(p_x, p_y) = (4.5, 4.5)$ named as "goal 2". These two scenarios represent the most common situations the robot would encounter in the real world: a straight path and a cornering path. A total of 20 humans were spawned in the environment. The humans' trajectories over the past five seconds and the force directions are also shown in the figure.

In experiment 1 (the case navigating to goal 1), the normal CBF and time-varying CBF (t-v CBF) were compared with the baseline algorithm from ROS1 navigation stack. The simulation experiments are conducted 12 times per algorithm at the same condition to account for the randomness of the algorithm. Table I shows the mean and standard deviation of the evaluation metrics. For the baseline algorithm, the values shown in the table are results from 8 trials because the robot failed to reach the goal in 4 trials. For the other two algorithms, the robot managed to reach the goal in all 12 trials.

The top five metrics indicate how efficiently the robot approaches the goal, and the bottom six metrics show how safely the robot navigates among humans. These metrics were chosen to evaluate performance in terms of both task efficiency and safety level, which are both crucial for robot navigation and often in a trade-off relationship. The definition of metrics

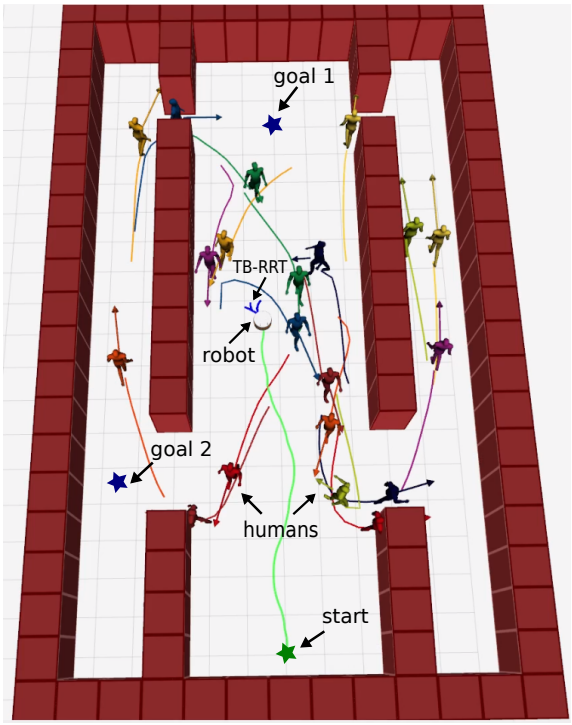


Fig. 4: Visualization of the testing scenario in the dynamic human environment.

TABLE I: Metrics for simulation experiment 1

Metrics	Value (Mean ± SD)		
	CBF	T-v CBF	Baseline
Time to reach the goal [s]	43.9 ± 5.7	41.2 ± 4.6	47.7 ± 10.9
Path length [m]	15.5 ± 0.3	15.4 ± 0.3	16.1 ± 0.9
Avg. robot linear speed [m/s]	0.34 ± 0.05	0.36 ± 0.05	0.33 ± 0.06
Cum. heading changes [rad]	4.0 ± 0.5	3.7 ± 0.4	4.0 ± 1.5
Time not moving [s]	4.8 ± 2.2	3.3 ± 1.7	16.9 ± 9.5
Avg. dist. closest human [m]	1.24 ± 0.10	1.33 ± 0.16	1.02 ± 0.08
Intimate space intrusion [%]	25.4 ± 3.7	22.4 ± 5.1	29.1 ± 7.5
Personal space intrusion [%]	39.2 ± 4.6	39.0 ± 5.8	43.2 ± 10.1
Social space intrusion [%]	29.4 ± 6.4	31.8 ± 4.7	24.0 ± 3.9
Robot to person collision [-]	0.9 ± 1.0	0.3 ± 0.5	0.9 ± 0.8
Person to robot collision [-]	2.1 ± 1.4	1.9 ± 1.1	1.4 ± 0.7

for personal space intrusion is based on Hall's proxemics theory [10], which classifies different zones of personal space as follows: intimate space is less than 0.45 [m], personal space ranges from 0.45 to 1.2 [m], and social space extends from 1.2 to 3.6 [m]. These metrics serve as indicators of how safe the robot is perceived, given the reported relationship between the proximity of the robot and a person's perceived safety. Robot on person collision is the number of times the robot collides with a person, while person on robot collision is the number of times a person collides with the robot. This distinction of collision is determined by the angle of the collision and the linear velocity of each agent [23]. The former case mainly occurs when the robot fails to stop in time to avoid the collision, and the latter case mainly occurs when the human has a stronger attraction to the next waypoint than to avoid the robot. These metrics are commonly used in the social robot navigation domain [15, 8].

The comparison in Table I shows that time-varying CBF performs better than the normal CBF in terms of both task performance and safety. The time-varying CBF allows the robot to reach the goal faster, with a shorter path and less time of not moving, while maintaining a larger distance from humans and with fewer collisions. This outcome is expected because the time-varying CBF restricts more control input when humans are approaching the robot, while it allows more input when humans are moving away from the robot. As a result, the robot is forced to be more conservative when humans are approaching and is allowed to have more aggressive control when humans are moving away.

Additionally, the proposed algorithms outperform or equally perform compared to the baseline algorithm in all the listed metrics in scenario 1 on average. However, the baseline algorithms have much more deviation in the time to reach the goal, cumulative heading changes, and the time not moving. My observation of this difference during the experiment is that the CBF-based navigation allows the robot to keep moving towards the goal if even a little movement is possible, while the baseline algorithm opts to stop if it cannot find the path to the goal due to the presence of too many surrounding humans. This is also the main cause of the four failed trials where the robot did not manage to reach the goal. This implies the limitation of global and local costmap-based navigation of ROS1 navigation stack in dense and dynamic environments. The CBF-based approach demonstrates improved robustness and adaptability in the given scenario.

The collisions caused by the robot are considered to be due to the discrepancies in the commanded velocity input and the actual velocity executed by the robot. These discrepancies likely stem from the limitations in acceleration and suboptimal controller tuning. By improving the tuning of the controller and considering the acceleration limit when determining the velocity command, the robot on human collision should be further minimized.

In experiment 2, time-varying CBF is compared to the ROS1 navigation stack baseline algorithm, given the fact that the time-varying CBF performs better than the normal CBF in experiment 1. The experiments are conducted 6 times per algorithm. The result can be found in Table II. The values shown for time-varying CBF are the result of 5 trials as it did not reach the goal in one trial. The robot managed to reach the goal in all 6 trials in the baseline algorithms.

The comparison in experiment 2 shows that the baseline algorithm is overall better in task efficiency and exhibits slightly superior safety metrics. The main reason for the difference in task performance is that the baseline algorithm plans a path from the start to the goal from the beginning, while the CBF-based algorithm only considers the direction towards the goal and the surrounding obstacles at the current position. This approach made it challenging for the CBF-based navigation to avoid walls and navigate through narrow sections at the corner, resulting in less efficient task performance and

TABLE II: Metrics for simulation experiment 2

Metrics	Value (Mean ± SD)	
	T-v CBF	Baseline
Time to reach the goal [s]	20.7 ± 5.0	12.8 ± 1.9
Path length [m]	7.9 ± 0.7	6.5 ± 0.1
Avg. robot linear speed [m/s]	0.24 ± 0.06	0.38 ± 0.04
Cum. heading changes [rad]	2.1 ± 0.4	0.8 ± 0.1
Time not moving [s]	6.2 ± 5.2	1.1 ± 0.9
Avg. dist. to closest human [m]	1.95 ± 0.46	2.68 ± 0.56
Intimate space intrusion [%]	9.9 ± 10.8	6.1 ± 10.1
Personal space intrusion [%]	25.5 ± 9.7	18.3 ± 17.5
Social space intrusion [%]	56.3 ± 17.3	42.5 ± 25.0
Robot to person collision [-]	0.0 ± 0.0	0.0 ± 0.0
Person to robot collision [-]	0.8 ± 1.3	0.0 ± 0.0

one failed trial. Additionally, the struggle in manoeuvring at the narrow part forced the robot to encounter humans in closer proximity for a longer time, leading to worse safety metrics, such as a shorter average distance to the nearest human and more frequent intrusions into personal space. The difficulty with non-straight paths is a major weakness of the proposed algorithm. Therefore, in more complex scenarios, it may be beneficial to combine CBF-TB-RRT with a global planner that provides intermediate waypoints, creating a route from the start to the goal through multiple straight segments.

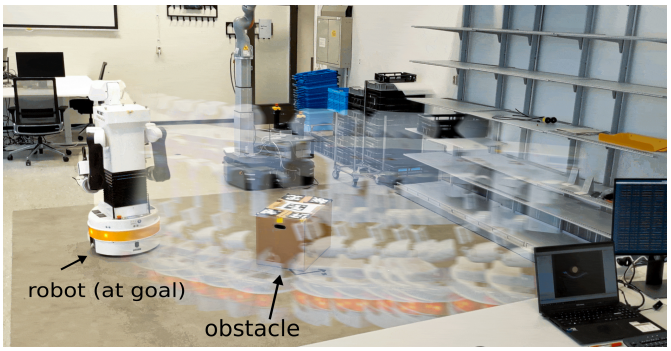
The parameters listed in table Table III are used for these experiments. These parameters are provisionally adjusted, but may be further tuned for better performance. Also, the optimal value depends on the nature of the environment, such as population density, dynamics of humans, and scenarios.

TABLE III: Parameters for robot simulation.

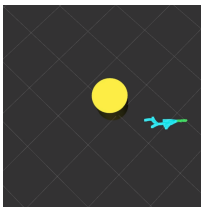
Parameter	Value
α	10
$v_{\max}$	0.8 [m/s]
$\omega_{\max}$	2.0 [rad/s]
N_s	6
a_{cost}	0.3
Ω_θ	1.5
a_ω	0.4
r_r	0.26

C. Experiment with a physical robot

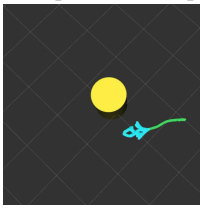
Finally, the algorithm is tested on the physical TIAGo robot within a static environment to evaluate its real-world performance. Figure 5 show the pictures from the experiment. The obstacle in the environment is a static box with a radius of 0.4 [m] from its centre. A motion capture (Mocap) system is employed to localize both the robot and the obstacle throughout the experiment. Due to the small testing area, v_{max} is limited to 0.3 [m/s]. One can see that the robot steers itself away from the obstacle to avoid collisions and navigates itself to the goal. This result validates that the CBF-RRT algorithm can be used to safely navigate the nonholonomic mobile robot not only in simulation, but also in a real-world robot. The video of the experiment is available in the link: https://youtu.be/sJSye8xWiSg.



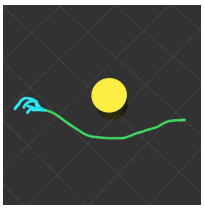
(a) Experiment setup



(b) At t = 3 [s]



(c) At t = 18 [s]



(d) At t = 55 [s]

Fig. 5: (a): A picture showing the sequence of robot motions during the experiment conducted in our lab space. The non-transparent robot is at the goal. (b)-(d): Visualizations of the executed path (light green) and the grown trees (light blue) at different time steps. Each square in the grid measures 1 [m] on each side.

VI. Conclusions

The simulation results show that the proposed algorithm can safely navigate a mobile robot through dense, dynamic human-populated environments.

The major difference from the prior work [14] is that this paper focused on how well the CBF-TB-RRT can perform in human-populated environment without accurate prediction of the future human motion. The result in the simulated dynamic environment shows that the collisions caused during the experiment were mostly caused by the humans and not by the robot, which resulted from the limitation in human pedestrian simulation. This encourages the validity of this algorithm for robot navigation in dense environments, where robots and humans cooperatively avoid collisions. Also, I proposed a new cost function for choosing the best vertex in the grown tree, see (12), that provides a better tradeoff between safety and closeness to the goal. While the previously proposed cost function prevented the robot from advancing in dense areas due to the higher cost of new nodes compared to the current node, this new cost function is more robust and enables the robot to navigate effectively through dense environments. Moreover, I integrated time-varying CBFs to account for the uncertain future motion of humans. By incorporating the time derivative of $h(x)$ into the CBF constraints into the CBF constraints, the

[1]The code of the navigation algorithm is available at https://github.com/nanaminh/navigation_CBF_RRT

[2]The code of the pedestrian simulation is available at https://github.com/nanaminh/pedsim_ros

algorithm considers the dynamics of humans around the robot. This approach provides more adapted constraints depending on whether the human is approaching the robot or is moving away. As a result, it performs better in terms of both task performance and safety in the tested scenario. Additionally, the experiment with a physical robot validates our navigation algorithm in the real-world nonholonomic robot.

In future work, the algorithm could be compared with more advanced baselines, such as ROS2 navigation stack [13]. This comparison will provide an indication of how the CBF-based navigation algorithm performs against other state-of-art algorithms in the context of social robot navigation. Regarding the major weakness of the proposed algorithm in the cornering scenario, combining it with another global planner could be advantageous. A global planner can provide additional strategic planning, offering a broader perspective on the route and enabling the robot to handle intricate path segments. Such a combination would enhance the algorithm's ability to navigate more complex scenarios, which are often encountered in real-world environments. Furthermore, I am keen to conduct an experiment with a physical robot in a dynamic human environment. This experiment will be useful to see how the proposed method performs in a situation that is closer to a real-world scenario and contains more uncertainties such as human movements, communication latency, and slippage of the robot wheels. Lastly, safety in human environments involves more than just avoiding collisions; it also includes ensuring that the robot's movement is perceived as safe. Therefore, my vision is to extend the CBFs to enhance perceived safety by considering the proximity, velocity, and acceleration of the robot around humans and by incorporating the humans' feedback.

REFERENCES

[1] Neziha Akalin, Annica Kristoffersson, and Amy Loutfi. Do you feel safe with your robot? Factors influencing perceived safety in human-robot interaction based on subjective and objective measures. *International Journal of Human-Computer Studies*, 158:102744, February 2022. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2021.102744. URL https://www.sciencedirect.com/science/article/pii/S1071581921001622.

[2] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control Barrier Functions: Theory and Applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, Naples, Italy, June 2019. IEEE. ISBN 978-3-907144-00-8. doi: 10.23919/ECC.2019.8796030. URL https://ieeexplore.ieee.org/document/8796030/.

[3] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. *International Journal of Social Robotics*, 1(1):71–81, January 2009. ISSN 1875-4805. doi: 10.1007/s12369-008-0001-3. URL https://doi.org/10.1007/s12369-008-0001-3.

[4] Robert Bogue. Strong prospects for robots in retail. *Industrial Robot: the international journal of robotics research and application*, 46(3):326–331, January 2019. ISSN 0143-991X. doi: 10.1108/IR-01-2019-0023. URL https://doi.org/10.1108/IR-01-2019-0023. Publisher: Emerald Publishing Limited.

[5] Urs Borrmann, Li Wang, Aaron D. Ames, and Magnus Egerstedt. Control Barrier Certificates for Safe Swarm Behavior. *IFAC-PapersOnLine*, 48(27):68–73, January 2015. ISSN 2405-8963. doi: 10.1016/j.ifacol.2015.11.154. URL https://www.sciencedirect.com/science/article/pii/S240589631502412X.

[6] Paolo Fiorini and Zvi Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17(7):760–772, July 1998. ISSN 0278-3649. doi: 10.1177/027836499801700706. URL https://doi.org/10.1177/027836499801700706. Publisher: SAGE Publications Ltd STM.

[7] Giuseppe Fragapane, Hans-Henrik Hvolby, Fabio Sgarbossa, and Jan Ola Strandhagen. Autonomous Mobile Robots in Hospital Logistics. In Bojan Lalic, Vidosav Majstorovic, Ugljesa Marjanovic, Gregor von Cieminski, and David Romero, editors, *Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems*, IFIP Advances in Information and Communication Technology, pages 672–679, Cham, 2020. Springer International Publishing. ISBN 978-3-030-57993-7. doi: 10.1007/978-3-030-57993-7_76.

[8] Anthony Francis, Claudia Pérez-D'Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, Hao-Tien Lewis Chiang, Michael Everett, Sehoon Ha, Justin Hart, Jonathan P. How, Haresh Karnan, Tsang-Wei Edward Lee, Luis J. Manso, Reuth Mirksy, Sören Pirk, Phani Teja Singamaneni, Peter Stone, Ada V. Taylor, Peter Trautman, Nathan Tsoi, Marynel Vázquez, Xuesu Xiao, Peng Xu, Naoki Yokoyama, Alexander Toshev, and Roberto Martín-Martín. Principles and Guidelines for Evaluating Social Robot Navigation Algorithms, September 2023. URL http://arxiv.org/abs/2306.16740. arXiv:2306.16740 [cs].

[9] Juan Miguel Garcia-Haro, Edwin Daniel Oña, Juan Hernandez-Vicen, Santiago Martinez, and Carlos Balaguer. Service Robots in Catering Applications: A Review and Future Challenges. *Electronics*, 10(1):47, January 2021. ISSN 2079-9292. doi: 10.3390/electronics10010047. URL https://www.mdpi.com/2079-9292/10/1/47. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[10] Edward Twitchell Hall. *The Hidden Dimension*, volume 609. Anchor, 1969.

[11] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995. ISSN 1063-651X, 1095-3787.

doi: 10.1103/PhysRevE.51.4282. URL https://link.aps.org/doi/10.1103/PhysRevE.51.4282.

[12] Motoi Igarashi, Issei Tezuka, and Hisakazu Nakamura. Time-varying Control Barrier Function and Its Application to Environment-Adaptive Human Assist Control. *IFAC-PapersOnLine*, 52(16):735–740, January 2019. ISSN 2405-8963. doi: 10.1016/j.ifacol.2019.12.050. URL https://www.sciencedirect.com/science/article/pii/S2405896319318804.

[13] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725, October 2020. doi: 10.1109/IROS45743.2020.9341207. URL http://arxiv.org/abs/2003.00368. arXiv:2003.00368 [cs].

[14] Keyvan Majd, Shakiba Yaghoubi, Tomoya Yamaguchi, Bardh Hoxha, Danil Prokhorov, and Georgios Fainekos. Safe Navigation in Human Occupied Environments Using Sampling and Control Barrier Functions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5794–5800, Prague, Czech Republic, September 2021. IEEE. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636406. URL https://ieeexplore.ieee.org/document/9636406/.

[15] Noé Pérez-Higueras, Roberto Otero, Fernando Caballero, and Luis Merino. HuNavSim: A ROS 2 Human Navigation Simulator for Benchmarking Human-Aware Robot Navigation, September 2023. URL http://arxiv.org/abs/2305.01303. arXiv:2305.01303 [cs].

[16] PAL robotics. pmb2_simulation. URL https://github.com/pal-robotics/pmb2_simulation.

[17] Masahiro Shiomi, Francesco Zanlungo, Kotaro Hayashi, and Takayuki Kanda. Towards a Socially Acceptable Collision Avoidance for a Mobile Robot Navigating Among Pedestrians Using a Pedestrian Model. *International Journal of Social Robotics*, 6(3):443–455, August 2014. ISSN 1875-4805. doi: 10.1007/s12369-014-0238-y. URL https://doi.org/10.1007/s12369-014-0238-y. Number: 3.

[18] E.A. Sisbot, L.F. Marin-Urias, R. Alami, and T. Simeon. A Human Aware Mobile Robot Motion Planner. *IEEE Transactions on Robotics*, 23(5):874–883, October 2007. ISSN 1552-3098. doi: 10.1109/TRO.2007.904911. URL http://ieeexplore.ieee.org/document/4339546/.

[19] srl{-}freiburg. Pedsim ROS. URL https://github.com/srl-freiburg/pedsim_ros.

[20] Mikael Svenstrup, Thomas Bak, and Hans Jørgen Andersen. Trajectory planning for robots in dynamic human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4293–4298, October 2010. doi: 10.1109/IROS.2010.5651531. URL https://ieeexplore.ieee.org/document/5651531/citations?tabFilter=papers#citations. ISSN: 2153-0866.

[21] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: a second-generation museum tour-guide robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 3, pages 1999–2005, Detroit, MI, USA, 1999. IEEE. ISBN 978-0-7803-5180-6. doi: 10.1109/ROBOT.1999.770401. URL http://ieeexplore.ieee.org/document/770401/.

[22] Pete Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, March 2015. ISSN 0278-3649. doi: 10.1177/0278364914557874. URL https://doi.org/10.1177/0278364914557874. Publisher: SAGE Publications Ltd STM.

[23] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S. Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W. Gupta, Mubbasir Kapadia, and Marynel Vázquez. SEAN 2.0: Formalizing and Generating Social Situations for Robot Navigation. *IEEE Robotics and Automation Letters*, 7(4):11047–11054, October 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3196783. URL https://ieeexplore.ieee.org/document/9851501/?arnumber=9851501. Conference Name: IEEE Robotics and Automation Letters.

[24] Spencer van Koevering, Yiwei Lyu, Wenhao Luo, and John Dolan. Provable Probabilistic Safety and Feasibility-Assured Control for Autonomous Vehicles using Exponential Control Barrier Functions. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 952–957, Aachen, Germany, June 2022. IEEE. ISBN 978-1-66548-821-1. doi: 10.1109/IV51971.2022.9827424. URL https://ieeexplore.ieee.org/document/9827424/.

[25] Sanne van Waveren, Rasmus Rudling, Iolanda Leite, Patric Jensfelt, and Christian Pek. Increasing Perceived Safety in Motion Planning for Human-Drone Interaction. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pages 446–455, Stockholm Sweden, March 2023. ACM. ISBN 978-1-4503-9964-7. doi: 10.1145/3568162.3576966. URL https://dl.acm.org/doi/10.1145/3568162.3576966.