Safe Receding Horizon Motion Planning with Infinitesimal Update Interval

Inkyu Jang, Sunwoo Hwang, Jeonghyun Byun, and H. Jin Kim

***Abstract*— Safety verification in motion planning is known to be computationally burdensome, despite its importance in robotics. In this paper, we investigate the behavior of safe receding horizon motion planners when the update interval becomes infinitesimal. By requiring the trajectory parameters to evolve continuously in time, the trajectory optimization problem is reformulated into a time-derivative form, whose decision variables are their rate of change. This results in a quadratic programming problem which directly provides safe input, and can be regarded as a real-time safety filter. The input expressivity is also enhanced by leveraging the differentiable structure of the parameter space. The proposed safety filter is experimentally validated using a wheeled ground robot in obstacle-cluttered environments. The result shows that the safety filter is capable of generating safe inputs in real-time, while addressing hundreds of constraints simultaneously.**

I. Introduction

Safety assurance is a very important problem in generating robot motion. Mathematically, the safety requirement can be interpreted as persistent satisfaction of a number of inequality constraints on the state space. The burden of checking constraint satisfaction over an infinite time horizon makes direct safety verification intractable, and many motion planners therefore choose to plan trajectories in a *receding horizon* manner. In such methods, at every update, the robot repeatedly generates trajectories within the prediction horizon by formulating an optimization problem that searches for the best suitable set of trajectory parameters satisfying the safety and motion constraints, e.g., collision avoidance, actuation limits. By doing so, the complex trajectory generation problems are transformed into finite-dimensional optimization problems solvable by computers, but they still remain nonlinear and nonconvex in general, making online safety verification computationally burdensome.

Suppose that the trajectory update occurs periodically with an interval of $\tau_{\text{plan}} > 0$. It is advantageous to keep the interval τ_{plan} as short as possible, in that it enables the system to respond more promptly to changing environmental conditions such as discovery of previously unseen obstacles or state drift due to external disturbances. At the same time, τ_{plan} also serves as the *time budget* to complete searching for the next trajectory, i.e., solving the optimization problem. Since a failure to find a feasible trajectory within this timeframe may put the robotic system at a safety risk, blindly choosing a short τ_{plan} in the current form is not practicable.

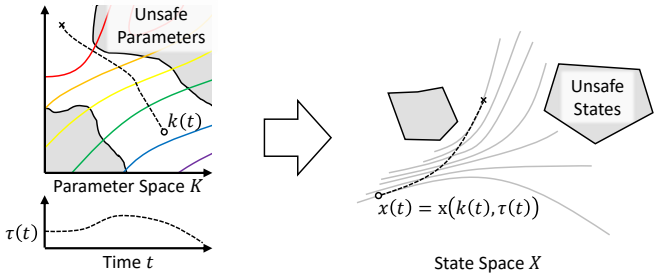

Fig. 1. Let k and τ be the trajectory parameter and the timepoint on the trajectory, respectively. Continuously changing k and τ gives a uniquely-determined trajectory in the state space. In this paper, we reformulate the trajectory parameter optimization problem into a time-derivative form, which enables real-time safety assurance.

In this work, we consider the limit $\tau_{\text{plan}} \searrow 0$. The trajectory parameters are required to change *continuously* with respect to time (Fig. 1), and the optimization problem is translated into a derivative form whose decision variables are the parameters' rate of change. The nonlinear, nonconvex constraints are transformed into linear inequalities that can be efficiently handled. The objective function is replaced with a convex quadratic one, such that the resulting motion planning problem is a quadratic program (QP) which can be solved in real-time on onboard computers. The resulting optimization may be also regarded as a *safety filter*, since it directly gives the optimal safe input. The proposed safety filter takes advantage of the differentiable structure of the parameter space, and can even generate inputs that are not in the trajectory library, while providing the same level of safety guarantee.

We validate the proposed safety filter through an experiment scenario using a differential-drive ground rover with five-dimensional state space and two-dimensional input space. Even under aggressive reference input signal, the proposed method succeeded in generating inputs in real-time that satisfy hundreds of safety requirements.

II. Related Work

Many prior works adopted receding horizon motion planning (RHP) methods to address safety for many different

This work was supported by the Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and the Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, Republic of Korea, under Grant NRF-2020M3C1C1A010864.

The authors are with the Department of Aerospace Engineering, Automation and Systems Research Institute (ASRI), Seoul National University, Seoul, Korea. {leplusbon, swsw0411, quswjdgus97, hjinkim}@snu.ac.kr

types of robotic systems, including unmanned ground/aerial vehicles, a swarm of drones, humanoid robots, and robotic manipulators [1]–[4]. Safety verification of the receding horizon trajectories is often done using Hamilton-Jacobi (HJ) reachability analysis [5]. For example, using this technique, [6] computed a library consisting of funnel-shaped tubes that encompass all possible tracking errors to ensure safety even in the worst case scenario. The computational complexity of HJ reachability analysis, however, scales exponentially with respect to the system's dimension, also known as the *curse of dimensionality*. Many works recognized this issue and proposed ways to reduce the dimension by using, for example, a low-fidelity *planning* model [7], [8], or a system decomposition technique [9]. There were also attempts to reduce computation by utilizing polynomial optimization [10]–[12], zonotopes [13], or system linearization techniques [14]. Still, their online computation times are nonnegligible and they often require a high-performance onboard computer to achieve a reasonable τ_{plan}.

This work is also well-aligned with recent advances in the topic of control barrier functions (CBFs) [15], such as CBF-QP, a real-time QP-based safety filter using a CBF [16]. Although handcrafting a valid CBF is often very hard for systems with actuation limits, there were a number of meaningful attempts to systematically synthesize one. For example, [17], [18] used polynomial bases to parametrize CBFs. In [19], an optimal control problem, whose value function becomes a CBF, was formulated and solved. Unfortunately, these methods also are not free from the curse of dimensionality and require a tremendous amount of computation. To address this, [20], [21] used deep neural networks, and [22] used multiple CBFs based on Lyapunov functions to efficiently cover the safe space. The work [23] introduced a way to construct a CBF in a more computationally efficient manner using evading maneuvers. As we discuss in Remark 3, the proposed safety filter can be understood as a generalization of [23], with the parametrized trajectories being the evading maneuvers. Our work relieves the mentioned issues of CBF-based methods in terms of computation speed and satisfaction of actuation limits by utilizing a library of feasible trajectories.

III. Preliminaries

A. Notation

The letter t is used to denote the (physical) time, and the Greek letter τ is used to indicate a specific timepoint on a trajectory. Let $\beta : \zeta(\in D) \mapsto \beta(\zeta)(\in Y)$ be a differentiable mapping where D and Y are differentiable manifolds. We use the notation $\partial_\zeta \beta(\zeta) : T_\zeta D \to T_{\beta(\zeta)} Y$ to denote the pushforward map of β: i.e., if $\zeta(t)$ is a differentiable curve parametrized by t on D, $\frac{d}{dt}\beta(\zeta(t)) = \partial_\zeta \beta(\zeta(t)) \cdot \dot{\zeta}(t)$, where the dot notation $\dot{(\cdot)}$ is used to denote the total derivative with respect to time t. The dot notation should not be confused with $\partial_t(\cdot)$ or $\partial_\tau(\cdot)$. The function β is said to be C^1 if it is differentiable and its derivative is continuous. An inequality symbol between two equidimensional vectors denotes that it is satisfied in an elementwise manner.

B. Dynamics and Safety Requirements

We consider a robot with continuous-time nonlinear time-invariant control-affine dynamics

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

where $x \in X$ is the state, $u \in U$ is the input. The state space X is a differentiable manifold, and we assume U is a convex closed subset of $\mathbb{R}^m$.

Safety requirements for this system are in general written in the form of nonlinear inequality constraints on the state variable x. For some mapping $d : X \to \mathbb{R}^m$, the state x is instantaneously safe if every element of $d(x)$ is nonnegative, i.e., $d(x) = [d_1(x), \cdots, d_m(x)]^\top \geq 0$

IV. Receding Horizon Motion Planning

A. Receding Horizon Motion Planning

A general RHP problem is written in the following form.

$$\begin{aligned}
\min_{\mathrm{x},\mathrm{u}} \quad & \mathcal{J}[\mathrm{x},\mathrm{u}] && \text{(2a)}\\
\text{s.t.} \quad & \mathrm{x}(0) = x(t) && \text{(2b)}\\
& \partial_\tau \mathrm{x}(\tau) = f(\mathrm{x}(\tau)) + g(\mathrm{x}(\tau)) \cdot \mathrm{u}(\tau) && \text{(2c)}\\
& d(\mathrm{x}(\tau)) \geq 0 && \text{(2d)}\\
& \partial_\tau \mathrm{x}(T) = 0 && \text{(2e)}\\
& \mathrm{u}(\tau) \in U && \text{(2f)}
\end{aligned}$$

Each constraint being represented as a function of τ means that it should be satisfied for all $\tau \in [0, T]$, where $T > 0$ is the prediction horizon. In this optimization problem, we aim to find the tuple of feasible state and input trajectories $\mathrm{x} : [0, T] \to X$ and $\mathrm{u} : [0, T] \to U$ that minimizes the cost functional $\mathcal{J}$. The bracket $[\cdot]$ in (2a) is used to emphasize that $\mathcal{J}$ is a *functional* rather than a normal function. That is, it is a mapping from the set of all feasible trajectories, which is typically infinite-dimensional. The first condition (2b) is the initial condition, i.e., the starting point of the receding horizon trajectory should match the current state, (2c) is the dynamic feasibility, i.e., the trajectory x should be *trackable*, and (2d) is the safety requirement that should be satisfied throughout the interval $\tau \in [0, T]$. The *final stop* condition (2e) is employed to ensure *recursive feasibility*.

B. RHP using Parametrized Library of Trajectories

Since the aforementioned RHP (2) is an infinite-dimensional optimization problem, we need to parametrize the set of possible trajectories and search for the optimal parameter. Let $\mathrm{x}(k, \cdot) : [0, \infty) \to X$ and $\mathrm{u}(k, \cdot) : [0, \infty) \to U$ be the state and input trajectories parametrized by $k \in K$, where K is the parameter set. For all $k \in K$, the trajectories are required to satisfy the dynamics, final stop condition and input feasibility, i.e.,

- $\partial_\tau \mathrm{x}(k, \tau) = f(\mathrm{x}(k, \tau)) + g(\mathrm{x}(k, \tau)) \cdot \mathrm{u}(k, \tau)\ \forall \tau \geq 0$,
- $\partial_\tau \mathrm{x}(k, \tau) = 0\ \forall \tau \geq T(k)$, and
- $\mathrm{u}(k, \tau) \in U\ \forall \tau \geq 0$,

where $T(k)$ is the length of the receding horizon trajectory.

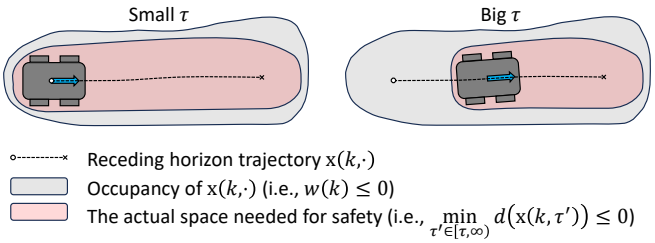

Fig. 2. Consider a ground rover controlled by (5), and suppose the function d indicates the robot's occupancy. When the value of τ is kept small (left), the robot stays near the origin of the trajectory, and its occupancy well matches the actual space needed for the robot to brake. With big τ (right) on the other hand, the trajectory's occupancy is excessively bigger than the actual space needed, resulting in conservatism.

Let the function $w : K \to \mathbb{R}^m$ conservatively indicate whether the parametrized trajectory is safe, i.e.,

$$w_i(k) \le \min_{t \in [0, T(k)]} d_i(\mathrm{x}(k,t)) \quad \forall i \in \{1, \cdots, m\}, \tag{3}$$

where $w(k) = [w_1(k), \cdots, w_m(k)]^\top$. Hence, if $w(k) \ge 0$, the trajectory $\mathrm{x}(k,\cdot)$ is safe. We assume that w_i-s are at least piecewise C^1.

Considering this safety constraint, the optimal RHP problem (2) is rewritten in the following form:

$$\begin{aligned} \min_{k} \quad & J(t,k) && \text{(4a)} \\ \text{s.t.} \quad & \mathrm{x}(k,0) = x(t) && \text{(4b)} \\ & w(k) \ge 0 && \text{(4c)} \\ & k \in K, && \text{(4d)} \end{aligned}$$

where $J(t,k)$ is the (possibly time-varying) trajectory cost. The constraint (4b) indicates that the initial condition $\mathrm{x}(k,0)$ should match the current state, (4c) is the safety constraint, and (4d) is the feasibility condition for the parameter k. Although the final stop condition is employed, the above optimization problem sometimes can run into an infeasibility due to the conservative definition of w. In that case, the robot is commanded to continue following the previous step's receding horizon trajectory. In summary, the planning is done using the following procedure.

1) Initialize: $t = 0$, let $x(0)$ be the robot's current state.
2) If $t = j\tau_{\mathrm{plan}}$ for some integer j, solve (4) to find the optimal trajectory parameter. If one exists, update k using the obtained optimal argument, and let $\tau = 0$. Else, reuse the previous step's parameter k, and $\tau \leftarrow \tau + \tau_{\mathrm{plan}}$.
3) During $t \in (j\tau_{\mathrm{plan}}, (j+1)\tau_{\mathrm{plan}}]$, apply the input $u = \mathrm{u}(k, t - j\tau_{\mathrm{plan}} + \tau)$.
4) At $t = (j+1)\tau_{\mathrm{plan}}$, the state is now at $x(t) = \mathrm{x}(k, \tau + \tau_{\mathrm{plan}})$. Go to step 2).

Given that the optimization problem (4) is feasible at the initial step $t = 0$, this algorithm provides persistent feasibility.

C. Parametrized RHP in a Recursively Feasible Form

Let t_{before} be the start time of the trajectory given from the most recent successful trajectory optimization, and let $\tau = t - t_{\mathrm{before}}$ be the elapsed time on that trajectory. In this optimal RHP problem, it is favorable to keep τ as small as possible, because big τ values (that result from frequent infeasibilities) increase conservatism, as shown in Fig. 2.

With that to keep in mind, we slightly modify the optimization problem (4) as follows:

$$\begin{aligned} \min_{\tau,k} \quad & J(t,k) + \lambda\tau && \text{(5a)} \\ \text{s.t.} \quad & \mathrm{x}(k,\tau) = x(t) && \text{(5b)} \\ & \tau \ge 0 && \text{(5c)} \\ & \text{(4c), (4d),} && \text{(5d)} \end{aligned}$$

where $\lambda > 0$ is a parameter that penalizes big τ. Note that (5) is not exactly equivalent to the algorithm introduced in the previous section, since we have blended the recursive-feasibility-ensuring logic into the objective function as a soft constraint. Nevertheless, with sufficiently large λ, the optimization will favor $\tau = 0$ if possible, yielding a similar result to (4). The constraint (5c) was added because the trajectories are defined only for $\tau \ge 0$.

Suppose that the optimization in the previous step was feasible and yielded τ_{before}, k_{before}. Then, (5) is always feasible (i.e., recursively feasible), because $\tau = \tau_{\mathrm{before}} + \tau_{\mathrm{plan}}$ and $k = k_{\mathrm{before}}$ is a feasible solution to the current step's optimization.

V. RHP with Infinitesimal Update Interval

A. Trajectory Parameter as Controllable Internal State

Now we discuss how to construct a real-time safety filter by taking the limit $\tau_{\mathrm{plan}} \searrow 0$. With infinitesimal τ_{plan}, k and τ now should be represented as *trajectories* in the spaces they live in. For the sake of numerical stability, we require k and τ to be continuous with respect to time. For that, the following two assumptions are added:

- The parameter space K is a connected subset of a finite-dimensional Riemannian manifold M, described by
$$K = \{k \in M : \rho(k) = [\rho_1(k), \cdots, \rho_{m_k}(k)]^\top \ge 0\},$$
where $\rho : M \to \mathbb{R}^{m_k}$ is an elementwise C^1 function. This assumption allows us to perform calculus on the parameter space and evaluate the change rate of the parameter.
- The parametrized spectrum of trajectories $\mathrm{x}(k,\tau)$ is continuous and piecewise C^1 with respect to k and τ.

Since the state should always stay on the parametrized trajectory, it is constrained through the relation

$$x(t) = \mathrm{x}(k(t), \tau(t)), \tag{6}$$

and given the above two assumptions, the resulting trajectory $x(t)$ is continuous and piecewise C^1.

To ensure continuity of the parameter, we regard k and τ as internal states of the motion planner, and require them to obey the following first-order *dynamics*:

$$\dot{k} = v, \quad \dot{\tau} = s \tag{7}$$

where $v \in T_k M$, $s \in \mathbb{R}$ are the virtual inputs. The variables k and τ are now *controlled* through v and s.

B. Constraints

First, to match the initial condition (5b) using the virtual inputs v and s, we take the time derivative of both sides of (6) to obtain

$$\dot{\mathrm{x}}(k,\tau,v,s) = \dot{x}(x,u) = f(x) + g(x)\cdot u, \tag{8}$$

where $\dot{\mathrm{x}}(k,\tau,v,s)$ can be evaluated as

$$\dot{\mathrm{x}}(k,\tau,v,s) = \partial_k \mathrm{x}(k,\tau)\cdot v + \partial_\tau \mathrm{x}(k,\tau)\cdot s. \tag{9}$$

To satisfy the safety and feasibility constraints through v, we use the following constraints inspired by CBF:

$$\dot{w}(k,v) = \partial_k w(k)\cdot v \geq -\alpha(w(k)), \tag{10}$$

$$\dot{\rho}(k,v) = \partial_k \rho(k)\cdot v \geq -\gamma(\rho(k)), \tag{11}$$

where α and γ are elementwise extended class-$\mathcal{K}_\infty$ functions.[1] Similarly, to keep τ nonnegative through s, we replace the constraint (5c) with

$$\dot{\tau} = s \geq -\sigma(\tau), \tag{12}$$

where σ is also an extended class-$\mathcal{K}_\infty$ function.

C. Objective Function

Cost minimization can be achieved by minimizing its time derivative. Namely, we want to minimize the objective

$$\frac{d}{dt}\left(J(t,k)+\lambda\tau\right) = \partial_t J(t,k) + \partial_k J(t,k)\cdot v + \lambda s. \tag{13}$$

Here, the term $\partial_t J(t,k)$ does not depend on the optimization variables v and s and therefore can be omitted. For the sake of the solution's uniqueness and numerical stability, we add the following quadratic regularization term

$$\frac{1}{2}\mu_v \|v\|^2_{T_k M} + \frac{1}{2}\mu_u \|u\|^2, \tag{14}$$

to ensure boundedness and uniqueness of the solution to the optimization. Here, μ_v and μ_u are positive weighting parameters, $\|v\|^2_{T_k M}$ is the squared norm associated with the Riemannian metric on M, $\|u\|^2$ is the squared norm on $\mathbb{R}^m$. This enhances the continuity of the optimal solution with respect to the parameters and reduces the jerkiness of the resulting trajectory.

D. Safety Filter

With the above constraints and the objective, the optimization problem (5) is reformulated into

$$\begin{aligned}
\min_{u,v,s}\quad & \partial_k J(t,k)\cdot v + \lambda s + \frac{1}{2}\mu_v\|v\|^2_{T_k M} + \frac{1}{2}\mu_u\|u\|^2 \\
\text{s.t.}\quad & \partial_k \mathrm{x}(k,\tau)\cdot v + \partial_\tau \mathrm{x}(k,\tau)\cdot s = f(x(t)) + g(x(t))\cdot u \\
& \partial_k w(k)\cdot v + \alpha(w(k)) \geq 0 \\
& \partial_k \rho(k)\cdot v + \gamma(\rho(k)) \geq 0 \\
& s + \sigma(\tau) \geq 0 \\
& u \in U.
\end{aligned} \tag{15}$$

[1] A function $\beta : \mathbb{R} \to \mathbb{R}$ is an extended class-$\mathcal{K}_\infty$ function if it is continuous, strictly increasing, unbounded, and $\beta(0) = 0$.

The advantage of this formulation is that, regardless of which J we take, it is a QP as long as the functions J, x, w and ρ are differentiable. Compared to the original setting (5), which is a nonlinear, nonconvex optimization problem, (15) can be solved in a significantly shorter time window using off-the-shelf convex optimization solvers, and therefore permits real-time online computation.

E. Invariance Property and Recursive Feasibility

We conclude this section by examining the invariance property and recursive feasibility of the proposed safety filter. We first show that the the feasible set of (5) is a control invariant set, controlled by the feedback controller given from (15).

Proposition 1 (Invariance)**.** *The input (u,v,s) obtained from (15), should one exist, renders the feasible set of (5) invariant.*

Proof. It can be easily seen that, for any active constraint of (5), the feasible solution for (15) will always push the optimization variable into the feasible set. For example, if the equality holds for the safety condition (i.e., $w(k) = 0$) then (10) will ensure $\dot{w} \geq 0$. □

Now, we show that (15) is always feasible within that set.

Proposition 2 (Feasibility of (15))**.** *For some $t \geq 0$, suppose τ, k is a feasible solution for (5). Then, the optimization (15) is feasible.*

Proof. If τ, k is a feasible solution for (5), $w(k) \geq 0$, $\rho(k) \geq 0$, and $\sigma(\tau) \geq 0$. It is straightforward to find out that $u = \mathrm{u}(k,\tau)$, $v = 0$, $s = 1$ is one feasible solution for (15). It is a continuous-time analogy of the robot's choice to stay on the previously planned trajectory in case (4) is infeasible. □

From the above two propositions, we conclude that the proposed control strategy is recursively feasible.

VI. Remarks

In this section, we mention some notable points about the proposed safety filter.

Remark 1 (Input expressivity)**.** In existing discrete-time safe RHP algorithms such as [6], [7] which are in the form of (4), it is very important to retain *expressivity* of the trajectory library, since the system is only allowed to utilize inputs contained in the library. In the current work on the other hand, the input u need not match $\mathrm{u}(\cdot,\cdot)$ and the differentiable structure of the parameter space allows a wider range of control input selection. For instance, the robot can accelerate aggressively when safety risks are not present, i.e., ρ and w values are sufficiently large. Owing to this strength, the dimension of the parameter space can be reduced (hence resulting in faster online computation speed) while having a similar level of expressivity with the discrete-time setting.

Remark 2 (Discussion on τ and s)**.** Suppose the following:

- The equality holds for (3).

TABLE I

THE VALUES OF CONSTANT PARAMETERS USED IN THE EXPERIMENT.

Symbol	Value	Symbol	Value	Symbol	Value
$v_{\max}$	0.26 m/s	μ_v	0.5	r	0.2 m
$\omega_{\max}$	1.82 rad/s	μ_u	0.5	b	0.065 m
$\dot{v}_{\max}$	0.1 m/s^2	ϵ	0.5	$\alpha(x) =$	$x\ (w_i)$
$\dot{\omega}_{\max}$	0.5 rad/s^2	δ	0.02 m		$2x\ (w_v, w_\omega)$

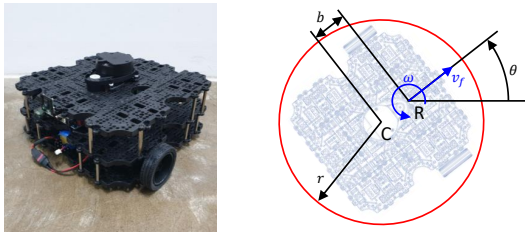

Fig. 3. Left: The Turtlebot 3 hardware used in the experiment. Right: The shape of the robot is assumed to be a circle with a radius of r, whose center (point C) is located away from the center of rotation (point R) by an offset of b. The robot's position p_x and p_y, its heading angle θ, linear and angular velocities v_f and ω are all measured with respect to R.

- There exists a function $v(k,\tau) \in T_k M$, such that $\partial_\tau \mathrm{x}(k,\tau) = \partial_k \mathrm{x}(k,\tau) \cdot v(k,\tau)$, $\forall k \in K, \tau \in [0,\infty)$.

The first assumption states that the safety filter does not *overestimate* the safety risk, and the second one indicates that for any time offset $\tau_0 > 0$, the subtrajectory defined as $\mathrm{x}(k, \tau+\tau_0)$ where $\tau \in [0,\infty)$ is also in the trajectory library. If so, we can eliminate s from the optimization variable by fixing $\tau = 0$, (thus $s = 0$) and rewrite the equality constraint of (15) as

$$\partial_k \mathrm{x}(k,0) \cdot v = f(x) + g(x) \cdot u. \tag{16}$$

This is a continuous-time analogy to the situation where (4) is always feasible.

Remark 3 (Connection to CBF-QP)**.** Suppose:

- The trajectories are parametrized by their initial conditions, i.e., $\mathrm{x}(k,0) = k$.
- The conditions of Remark 2 hold, so we can fix $\tau = 0$.
- $K = M = X$, i.e., $m_k = 0$. A braking trajectory is defined everywhere on the state space.

Then, with an appropriate choice of J, the optimization (15) takes the following form:

$$\begin{aligned} \min_{u \in U} \quad & \left\| u - u_{\mathrm{ref}}(t) \right\|^2 \\ \text{s.t.} \quad & \partial_k w(x) \cdot (f(x) + g(x) \cdot u) + \alpha(w(x)) \geq 0, \end{aligned} \tag{17}$$

which is identical to CBF-QP using a CBF synthesized using nominal evading maneuvers [23]. Thus, the proposed safety filter can be regarded as its generalization.

VII. EXPERIMENT

To validate the proposed safety filter, we conduct an experiment using Turtlebot 3 [24], a differential-drive wheeled robot (Fig. 3). The values of the constant parameters used in the experiment are summarized in Table I.

A. Experiment Setting

The dynamics model of Turtlebot 3 is written as

$$\dot{x} = (v_f \cos\theta, v_f \sin\theta, \omega, \dot{v}_f, \dot{\omega}), \tag{18}$$

where $x = (p_x, p_y, \theta, v_f, \omega) \in X = SE(2) \times \mathbb{R}^2$ is the state and $u = [\dot{v}_f, \dot{\omega}]^\top \in \mathbb{R}^2$ is the input bounded by $-\dot{v}_{\max} \leq \dot{v}_f \leq \dot{v}_{\max}$, $-\dot{\omega}_{\max} \leq \dot{\omega} \leq \dot{\omega}_{\max}$. Here, p_x and p_y represent the 2D coordinates of the robot's position, θ is the heading angle, v_f its forward velocity, and ω is the turning rate. We assume that the robot has a circular shape of radius r, where the center of the circle is located away from the center of rotation by an offset of b, as depicted in Fig. 3.

We set a spectrum of trajectories for this system parametrized by their initial conditions, i.e., $K = X$. With $k = (k_1, \cdots, k_5)$, the input trajectory is parametrized as

$$\mathrm{u}(k,\tau) = \begin{cases} [-k_4/T(k), -k_5/T(k)]^\top & (0 \leq \tau \leq T(k)) \\ [0,0]^\top & (\tau > T(k)), \end{cases} \tag{19}$$

and the state trajectory $\mathrm{x}(k,\tau)$ is given as the solution to the following initial value problem:

$$\begin{aligned} \partial_\tau \mathrm{x}(k,\tau) &= f(\mathrm{x}(k,\tau)) + g(\mathrm{x}(k,\tau)) \cdot \mathrm{u}(k,\tau), \\ \mathrm{x}(k,0) &= k. \end{aligned} \tag{20}$$

The braking time $T(k) \in \mathbb{R}$ is defined as

$$T(k) = \frac{1}{1-\epsilon} \cdot \max\left\{ \frac{|k_4|}{\dot{v}_{\max}}, \frac{|k_5|}{\dot{\omega}_{\max}} \right\}, \tag{21}$$

where $\epsilon \in [0,1)$ is a constant parameter. Nonzero ϵ provides some level of robustness to numerical error and disturbances.

The safety requirements for this experiment are as follows:

- Speed limit: $|v_f| \leq v_{\max}$ and $|\omega| \leq \omega_{\max}$.
- Collision avoidance: For every 2D LiDAR point $p_{\mathrm{obs}} \in \mathbb{R}^2$, $\left\| p_{\mathrm{obs}} - [p_x - b\cos\theta, p_y - b\sin\theta]^\top \right\| \geq r$.

To address the first requirement, we define

$$w_v(k) = [v_{\max} - v_f, v_{\max} + v_f]^\top, \tag{22}$$

$$w_\omega(k) = [\omega_{\max} - \omega, \omega_{\max} + \omega]^\top, \tag{23}$$

so that the speed limits can be satisfied by enforcing $w_v(k) \geq 0$ and $w_\omega(k) \geq 0$. To address the collision avoidance constraint, we let $D(p_{\mathrm{obs}}, k)$ be the distance of the obstacle point p_{obs} from the trace of the center of robot (point C in Fig. 3) given the state trajectory $\mathrm{x}(k,\cdot)$. The values of $D(\cdot,k)$ for various k values are shown in Fig. 4. The D function is continuous and piecewise C^1 with respect to the trajectory parameter k, and the its value and gradient can be obtained using simple hand-doable geometric calculations. Obstacle detection is done by an onboard 2D LiDAR sensor that scans the environment at 5 Hz. We assume that coordinates of the LiDAR points are stationary and regard them as static obstacles. Hence, for every LiDAR point $p_{\mathrm{obs},i}$ where $i \in \{1, \cdots, N\}$, we let $w_i(k) = D(p_{\mathrm{obs},i}, k) - r - \delta$, so that the system is collision-free if $w_i(k) \geq 0$ for all i. Here, $N \in \mathbb{N}$ is the number of LiDAR points[2], and $\delta > 0$ is the

[2]With the bundle LiDAR of Turtlebot3, the number of sensed obstacles N ranges up to 360, although it varies from time to time.

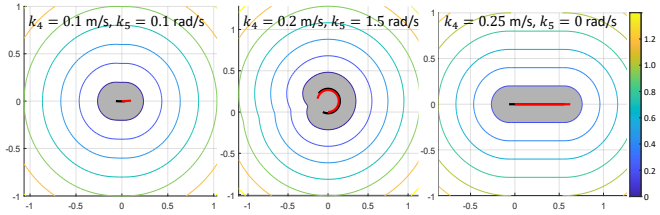

Fig. 4. The values of $D(p_{\text{obs}}, k)$, plotted as a function of p_{obs} for given $k = (0, 0, 0, k_4, k_5)$. The horizontal and vertical axes represent the position of the LiDAR point p_{obs}, the contours represent the values of $D(\cdot, \cdot)$, and the gray shaded regions denote the set of $p_{\text{obs},i}$ points that make $w_i(\cdot) \leq 0$. The black and red curves denote the trace of the center of robot and the center of rotation (points C and R in Fig. 3), respectively.

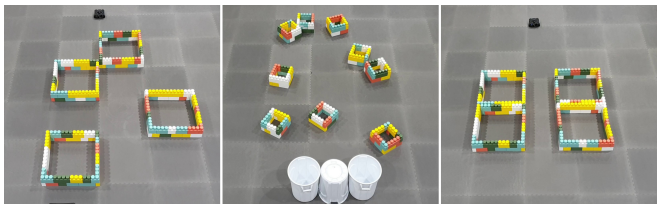

Fig. 5. The obstacle environments used in the experiment.

safety distance margin. Since $w_{(\cdot)}$-s satisfy the conditions mentioned in Remark 2, we choose to fix $\tau = 0$.

The trajectory cost J is set as

$$J(t, k) = -u_{\text{ref}}(t)^\top \pi_{\mathbb{R}^2}(\mathrm{x}(k, 0)), \tag{24}$$

where the map $\pi_{\mathbb{R}^2} : X \to \mathbb{R}^2$ extracts the linear and angular velocity components from the state. This cost J drives the robot to align its velocity and angular acceleration to the given u_{ref}. After differentiating, with $\mu_v + \mu_u = 1$, it results in the CBF-QP-style quadratic objective $\left\|u - u_{\text{ref}}(t)\right\|^2$. The reference input $u_{\text{ref}}(t) \in \mathbb{R}^2$ is given manually by a human-operated remote controller. To thoroughly test safety, the operator is instructed to transmit aggressive reference inputs towards the obstacle.

The experiment was conducted in an indoor environment with different obstacle configurations, as shown in Fig. 5. We used the C++ programming language and OSQP [25], an open-source QP solver, to implement and solve the optimization problem (15).

B. Results

The state variable, input (reference and filtered), and the values of w_i are plotted in Fig. 6 as a function of time. At almost all times, the w_i values were successfully kept nonnegative, i.e., collision did not occur, except for very small portions caused by measurement noise of the LiDAR sensor and the uncertainty in the model. Even in the case of negative w_i, (6) was able to drive the system back into the safe region, which is similar to the advantage offered by CBF-based safety critical controllers [15]. It can be also seen that the speed limits were satisfied through w_v and w_ω. Although the trajectory library $\mathrm{x}(\cdot, \cdot)$ consists of *decelerating* maneuvers only (see (19)), the safety filter was able to generate accelerating motions as well, as mentioned in Remark 1.

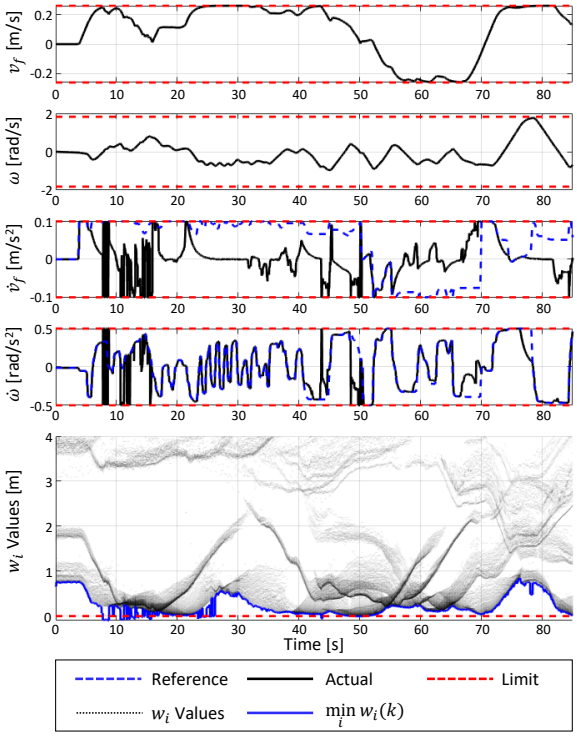

Fig. 6. The plots showcase that the safety constraints are met throughout the robot's deployment: linear and angular velocities between the limit, and nonnegative w_i values. Negative w_i values during a very short interval mainly result from dynamic model mismatch, and noise of the sensor measurements. Addressing these is left as a future research direction.

The safety filter was able to run at a rate faster than 50 Hz on an onboard computer with 2.4 GHz CPU and 16 GB RAM, which is fast enough to be considered real-time. It is also notable that it could handle hundreds of safety constraints (including LiDAR data points) simultaneously.

VIII. Conclusion

This paper discussed taking an infinitesimal update interval for a safe RHP problem. Using a spectrum of differentiably parametrized dynamically feasible trajectories, the trajectory parameter optimization problem was reformulated into a time-derivative form by requiring the parameters to vary continuously with respect to time. This results in a safety filter written in a form of a QP which can be solved in real-time. We observed that the proposed safety filter is capable of generating input that is not in the library by leveraging the differentiable structure of the parameter space. An experiment using a differential-drive ground robot was carried out to validate the proposed method. The results show that, even with hundreds of safety constraints, the proposed safety filter can generate safe inputs in real-time.

Although the proposed method exhibits good performance in terms of computation speed and safety assurance, it still has some limitations. Firstly, the equality constraint was (5b) only satisfied *indirectly* through its time derivative, resulting in numerical error building up. Various kinds of model uncertainty should also be addressed for the method's deployment to a wider range of real-world robots. Therefore, for future work, we would like to develop an improved version addressing these issues.

REFERENCES

[1] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[2] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.

[3] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 922–938, 2021.

[4] J. Park, Y. Lee, I. Jang, and H. J. Kim, "DLSC: Distributed multi-agent trajectory planning in maze-like dynamic environments using linear safe corridor," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3739–3758, 2023.

[5] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[6] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[7] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[8] M. Chen, S. L. Herbert, H. Hu, Y. Pu, J. F. Fisac, S. Bansal, S. Han, and C. J. Tomlin, "Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 5861–5876, 2021.

[9] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[10] S. Kousik, B. Zhang, P. Zhao, and R. Vasudevan, "Safe, optimal, real-time trajectory planning with a parallel constrained Bernstein algorithm," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 815–830, 2020.

[11] H. Seo, C. Y. Son, and H. J. Kim, "Fast funnel computation using multivariate Bernstein polynomial," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1351–1358, 2021.

[12] I. Jang, H. Seo, and H. J. Kim, "Fast computation of tight funnels for piecewise polynomial systems," *IEEE Control Systems Letters*, vol. 6, pp. 2234–2239, 2021.

[13] J. Liu, Y. Shao, L. Lymburner, H. Qin, V. Kaushik, L. Trang, R. Wang, V. Ivanovic, H. E. Tseng, and R. Vasudevan, "REFINE: Reachability-based trajectory design using robust feedback linearization and zonotopes," *arXiv preprint arXiv:2211.11997*, 2022.

[14] H. Seo, D. Lee, C. Y. Son, I. Jang, C. J. Tomlin, and H. J. Kim, "Real-time robust receding horizon planning using Hamilton–Jacobi reachability analysis," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 90–109, 2022.

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

[16] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[17] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1216–1229, 2017.

[18] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 585–590.

[19] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier–value functions for safety-critical control," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6814–6821.

[20] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 3717–3724.

[21] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2289–2307, 2023.

[22] I. Jang and H. J. Kim, "Safe control for navigation in cluttered space using multiple Lyapunov-based control barrier functions," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2056–2063, 2024.

[23] E. Squires, P. Pierpaoli, and M. Egerstedt, "Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 1656–1661.

[24] "Turtlebot 3," https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/, accessed: 2023-09-15.

[25] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.