A Practical Reachability-Based Collision Avoidance Algorithm for Sampled-Data Systems: Application to Ground Robots

Charles Dabadie, Shahab Kaynama, and Claire J. Tomlin

***Abstract*— We describe a practical collision avoidance algorithm that synthesizes provably safe piecewise constant control laws (compatible with the sampled-data nature of the system) for an experimental platform. Our application is formulated in a pursuer-evader framework in which an automated unmanned vehicle navigates its environment while avoiding a moving obstacle that acts as a malicious agent. Offline, we employ reachability analysis to characterize the evolution of trajectories so as to determine what control inputs can preserve safety over every sampling interval. The moving obstacle is considered unpredictable with nearly no restrictions on its control policies (although we do take into account the physical constraints due to limited dynamical and actuation capacities of both robots). Online, the controller executes computationally inexpensive operations based only on an easy-to-store lookup table. The results of the experiment as well as the proposed algorithm are presented and discussed in detail.**

I. Introduction

Collision avoidance is a central problem when dealing with safety of autonomous vehicles such as unmanned aircraft and ground robots. Commonly, a safety distance is defined around the vehicle and the goal is to keep other objects away from this perimeter.

Much research has been done in this domain, mainly through the concept of *velocity obstacles* (basically the set of velocities that would lead to collision with a moving obstacle). In [1], safe straight-line trajectories are built for a host robot that faces multiple obstacles with constant velocity vectors, while in [2] velocity obstacles have been generalized to the case of obstacles that move along arbitrary trajectories. Safety guarantees are obtained for either finite or infinite time horizons. In [3], the concept of velocity obstacle has been extended considering unicycle models for obstacles. The safe paths for the host robot are successions of lines that do not consider the dynamical capacities or physical constraints of the robot during changes of heading or speed.

In [4], the problems of collision avoidance in the presence of moving obstacles and path planning are solved simultaneously in the joint state-time space via probabilistic roadmaps. A similar approach is that of rapidly-exploring random trees [5]. Other related techniques include the body of work on optimal trajectory generation and path planning in dynamic environments, e.g. through trajectory parameterization [6].

A numerical tool for solving the collision avoidance problem is reachability analysis [7], [8]. Indeed, in many applications, the systems under study are modeled by constrained nonlinear differential equations, and therefore analytical solutions for trajectories do not always exist. Reachability analysis allows for controller design via numerical safety verification of trajectories. The backward reachable set of a given set $\mathcal{A}$ is the set of initial states that can be driven into $\mathcal{A}$ by the constrained dynamical system. In the context of collision avoidance, the reachable set formulation can be used to construct control policies that ensure safety of the vehicle despite the actions of a malicious agent. A classical method for numerically computing the reachable set is via the resolution of a terminal value Hamilton-Jacobi-Isaacs (HJI) partial differential equation [7], where one computes the backward evolution of a level set function representing $\mathcal{A}$ using, for example, the method described in [9]. The safety-preserving control inputs are those that optimize the associated Hamiltonian.

In this paper we primarily focus on an application of the reachability-based controller design to an experimental platform with two agents. The problem we consider is to design and implement a practical collision avoidance algorithm that is to be mounted on board an automated unmanned ground vehicle (henceforth simply referred to as the UV) which is being chased by a human-operated robot (henceforth referred to as the moving obstacle, or simply the obstacle). Our testbed consists of two Pioneer robots [10] whose positions are measured by a VICON system. This measurement is in turn fed into the UV and is used online to compute an appropriate course of action. The challenges we face include proposing a provably-safe algorithm that is (a) light enough to store and execute on an embedded microcontroller, (b) compatible with the sampled-data nature of the system (a sampled-data system is one whose evolution is in continuous time, yet it is driven by a digital, and hence discrete-time, controller that has access to state measurements/estimates at a fixed sampling frequency), and (c) flexible enough to allow the UV to follow an objective when safety is not at stake.

Great progress has been made in reachability analysis of constrained, nonlinear sampled-data systems. The common element is to ensure conservatism when the control law is restricted to the class of piecewise constant (PWC) functions (as opposed to the more common, and less stringent, class of Lebesgue measurable functions). For instance, in [11], [12] a reach-avoid problem is considered where the goal is to reach

This work has been supported in part by NSF under CPS:ActionWebs (CNS-931843), by ONR under the HUNT (N0014-08-0696) and SMARTS (N00014-09-1-1051) MURIs and by grant N00014-12-1-0609, by AFOSR under the CHASE MURI (FA9550-10-1-0567).

S. Kaynama and C. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA. {kaynama, tomlin}@eecs.berkeley.edu

C. Dabadie is with Institut Supérieur de l'Aéronautique et de l'Espace (Formation Supaero), Toulouse, France. c.dabadie@isae.fr

a given target set in finite time, and thereafter remain in an invariant subset of the target while also avoiding a given unsafe set. Similarly in [13], a projection-based technique is described that allows one to compute the sampled-data discriminating kernel (loosely speaking, the complement of the reachable set) and its associated control laws.

Here, we build upon these works and propose and apply a collision avoidance algorithm on our experimental platform subject to the challenges described above. Our approach, similar to [11]–[13], is based on quantization of the control set (the set from which the UV draws its input values) which maintains conservatism/safety while reducing the online computational burden. Most computations are handled offline. The online computations, which are performed on the embedded microcontroller, are limited to forming an estimate of the state and executing basic arithmetic based on an easy-to-store lookup table (precomputed offline). Several concepts are introduced to allow simple, practical implementation of the collision avoidance algorithm. For instance, we introduce the notion of *freedom set* which allows us to identify the regions of the state space in which all possible control values can safely be applied to the system. The freedom set serves as an indicator when choosing optimal boundaries for the numerical grid (over which the reachable set has been computed) so as to minimize the data that is needed to be stored on the UV's memory. To that end, we also describe a method to account for the case in which a state estimate falls in between grid points. Doing so will waive the need for employing a dense grid (which could be impossible to store on limited memory), while still ensuring a reasonable degree of conservatism when choosing safety-preserving control values. We also discuss a technique that allows for computation of certain regions of the state space in which any nominal and arbitrary controller with access to the original non-quantized control set could be used without the fear of jeopardizing safety.

In Section III, we present a recursive algorithm for the computation of the sampled-data reachable set. We then describe how this set can be used to design a safety-preserving PWC control law. Section IV quantifies the error introduced when the state estimation does not yield values that can be mapped exactly on the stored grid points. In Section V, we lay out the online algorithm which is realized using the open-source Robot Operating System (ROS) [14], while in Section VI the main experimental results are presented in detail. Section VII discusses (and validates via simulations) additional physically-motivated restrictions that can be placed on the input set of the obstacle so as to reduce the conservatism (as measured by the energy expenditure of the UV's input signal) of our algorithm. Finally, concluding remarks and future works are provided in Section VIII.

II. Problem formulation

Consider a two-player differential game where the dynamics are governed by $\dot{x} = f(x, u, d)$ with $x \in \mathcal{X}$, where $\mathcal{X} \subseteq \mathcal{R}^n$ is the state space. Here, $u(\cdot)$ is the control input of the UV, and $d(\cdot)$ is the control input of the moving obstacle. For simplicity, we shall use the simplified notations u and d to denote both functions as well as their point-wise values; differentiating between these two types should be possible via the context in which they are used.

With a sampling time $T_s > 0$, we define the set of control inputs for the UV over $[-T_s, 0]$ as

$$\mathcal{C}_1 := \{u : [-T_s, 0] \to \mathcal{U} \text{ s.t. } u \text{ constant on } [-T_s, 0]\}, \quad (1)$$

where the input constraint set $\mathcal{U}$ is a compact subset of $\mathbb{R}^{m_u}$. To simplify our algorithm for the purpose of implementation on board the UV, we quantize the set $\mathcal{U}$ (which we denote by $\mathcal{U}^{\mathrm{q}}$) and only consider a finite number of possible input values. Therefore, the new set of control signals for the UV is

$$\mathcal{C}_1^{\mathrm{q}} := \{u : [-T_s, 0] \to \mathcal{U}^{\mathrm{q}} \text{ s.t. } u \text{ constant on } [-T_s, 0]\}. \quad (2)$$

Note, however that the quantization of the input set maintains our desired conservatism in the sense that all generated control laws by the algorithm are, as we shall see shortly, *safety-preserving*.

On the other hand, we define the set of control inputs for the obstacle over $[-T_s, 0]$ as

$$\mathcal{C}_2 := \{d : [-T_s, 0] \to \mathcal{D} \text{ s.t. } d \text{ measurable on } [-T_s, 0]\}, \quad (3)$$

where the input constraint set $\mathcal{D}$ is a compact subset of $\mathbb{R}^{m_d}$. We do not restrict the control inputs of the obstacle as we did for the UV because we want to allow any possible obstacle behavior.

For a given set $\mathcal{A}$, define its backward reachable set for fixed $u \in \mathcal{C}_1^{\mathrm{q}}$ over one sampling interval as

$$\mathcal{BA}(\mathcal{A}, u) := \{x_0 \in \mathcal{X} : \exists d \in \mathcal{C}_2, \exists t \in [-T_s, 0],\, x_{x_0}^{u,d}(t) \in \mathcal{A}\} \quad (4)$$

with $x_{x_0}^{u,d}(t)$ the solution of $\dot{x} = f(x, u, d)$, $x(-T_s) = x_0$.

To model the dynamics of the two agents (our automated UV and the moving obstacle) on a 2D plane, we use the classical pursuer-evader formulation. The UV acts as the evader and the obstacle as the pursuer. The relative dynamics can be represented [7] by the continuous-time system

$$\dot{x} = \frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -u_1 + d_1 \cos(x_3) + u_2 x_2 \\ d_1 \sin(x_3) - u_2 x_1 \\ d_2 - u_2 \end{pmatrix} = f(x\,, u\,, d). \quad (5)$$

The evader's inputs are speed u_1 and yaw rate u_2. The pursuer's inputs d_1 and d_2 are similarly defined. x_1 and x_2 are Cartesian coordinates of the pursuer in the evader's frame (centered at the evader, x-axis aligned with evader's heading) and x_3 is the relative angle between the evader's and the pursuer's headings; see Fig. 1 for a graphical description. The evader's control input is defined and constrained as $u := (u_1, u_2) \in \mathcal{U}^{\mathrm{q}} \subset \mathcal{U} = [0, U_1] \times [-U_2, U_2] \subseteq \mathbb{R}^2$. The pursuer's control input is defined and constrained as $d := (\Delta_1, d_2) \in \mathcal{D} = [-\frac{D_1}{2}, \frac{D_1}{2}] \times [-D_2, D_2] \subseteq \mathbb{R}^2$ where $\Delta_1 = d_1 - \frac{D_1}{2}$ (we shifted d_1 to simplify the Hamiltonian (9)).

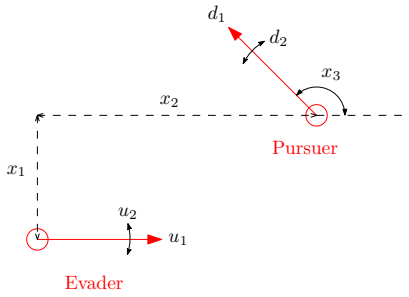


Fig. 1: Graphical description of pursuer-evader game. Arrows represent velocity vectors.

We naturally define the avoid set $\mathcal{A}$ around the evader as $\mathcal{A} := \{x \in \mathbb{R}^3 : \sqrt{x_1^2 + x_2^2} \leq R\}$, where $R > 0$ is the safety radius. The goal is to keep the pursuer out of this set that characterizes collision.

Consider the solution $\Phi : \mathcal{X} \times [-T_s, 0] \to \mathbb{R}$ of the HJI equation

$$\begin{cases} \frac{\partial \Phi}{\partial t} + \min[0, H(x, \frac{\partial \Phi}{\partial x})] = 0 \\ \Phi(x, 0) = \Phi_0(x) \end{cases} \tag{6}$$

over one sampling interval $[-T_s, 0]$ with Φ_0 being a function (e.g., signed distance) whose zero sublevel set represents $\mathcal{A}$ (i.e. $\Phi_0(x) \leq 0 \Leftrightarrow x \in \mathcal{A}$). Let $\Phi_0(x) = \sqrt{x_1^2 + x_2^2} - R$. The Hamiltonian for a fixed control input $u \in \mathcal{U}^{\mathrm{q}}$ for the evader is defined as

$$\forall x \in \mathcal{X},\ \forall p \in \mathbb{R}^3, H(x, p) = \inf_{d \in \mathcal{D}} [p^T f(x, u, d)], \tag{7}$$

where p is a placeholder for the costate. Expanding the Hamiltonian yields:

$$\begin{aligned} H(x,p) = \inf_{d\in\mathcal{D}} [&p_1(-u_1 + (\frac{D_1}{2} + \Delta_1)\cos(x_3) + u_2 x_2) \\ &+ p_2((\frac{D_1}{2} + \Delta_1)\sin(x_3) - u_2 x_1) \\ &+ p_3(d_2 - u_2)] \end{aligned} \tag{8}$$

$$\begin{aligned} = p_1(&-u_1 + \frac{D_1}{2}\cos(x_3) + u_2 x_2) \\ &+ p_2(\frac{D_1}{2}\sin(x_3) - u_2 x_1) - p_3 u_2 \\ &- \frac{D_1}{2}|p_1\cos(x_3) + p_2\sin(x_3)| - D_2|p_3|. \end{aligned} \tag{9}$$

The *unsafe* backward reachable set of $\mathcal{A}$ over $[-T_s, 0]$ is then described by the level set function $\Phi(\cdot, -T_s)$, i.e. $\mathcal{BA}(\mathcal{A}, u) = \{x \in \mathcal{X} : \Phi(x, -T_s) \leq 0\}$.

Through the formulation of $\mathcal{BA}(\mathcal{A}, u)$ the pursuer plays in a worst-case fashion—i.e., it selects input values that are most likely to drive the dynamics into the avoid set $\mathcal{A}$. To be conservative, we will design PWC safety-preserving control laws for the evader with the assumption that the pursuer plays its worst-case control during each sampling interval. In practice, the pursuer does not always behave that aggressively; but if the collision avoidance algorithm is safe in this case, then it will be also safe for any other situation.

III. Reachability-Based Controller Design

A. Unsafe Set Computation

We begin by computing the backward reachable set of the avoid set $\mathcal{A}$ under the dynamics (5), where the evader is assumed to use PWC control laws while the pursuer is allowed to use any admissible Lebesgue measurable control policy. This can be achieved through an iterative process, using the semi-group property of reachable sets, where the control is kept constant on every sampling interval $[-nT_s, -(n-1)T_s]$. Then after N iterations one obtains a PWC control law over $[-NT_s, 0]$ by connecting the subintervals. The backward reachable set that is obtained at iteration n is denoted by $\mathcal{R}_n$ and is the nth unsafe set (if the initial state $x_0 \in \mathcal{R}_n$ then it can be driven into the avoid set within n or less sampling intervals). It is obtained via the recursion

$$\mathcal{R}_0 = \mathcal{A}, \tag{10}$$

$$\mathcal{R}_n = \bigcap_{u \in \mathcal{C}_1^{\mathrm{q}}} \mathcal{BA}(\mathcal{R}_{n-1}, u),\ \forall n \in \mathbb{N}. \tag{11}$$

Intuitively, at iteration n, for every point of the state space $\mathcal{X}$, the intersection implicitly selects the best control $u \in \mathcal{C}_1^{\mathrm{q}}$ to apply over $[-nT_s, -(n-1)T_s]$ (i.e. the input that will make the backward evolution of the $(n-1)$th unsafe set $\mathcal{R}_{n-1}$ the smallest possible). It can be shown that the intersection of sets that are represented by Lipschitz level set functions is also represented by a Lipschitz level set function. Hence, the Lipschitz property that is a necessary requirement of the Level Set Toolbox [9] is preserved through iterations.

By construction of the above recursion, if the initial state $x_0 \notin \mathcal{R}_n$, then there exists a PWC control law over $[-nT_s, 0]$ that will keep the evader safe (i.e. out of the avoid set), regardless of the control law of the pursuer.

B. Convergence of the Reachable Set

If the unsafe reachable set converges, i.e. if $\exists N \in \mathbb{N}$ such that $\forall n \geq N$, $\mathcal{R}_n = \mathcal{R}_N$, then the time dependency of the previous sequence of unsafe sets disappears and the set $\mathcal{R} := \mathcal{R}_N$ becomes (robust) controlled-invariant. Indeed, if $x_0 \notin \mathcal{R}$, then there exists a PWC control law that will keep the dynamics out of $\mathcal{R}$ (and thus out of $\mathcal{A}$) *forever*, despite the worst-case efforts of the pursuer. The existence and design of such a control law is described next. In our particular experiment (details of which will be described later), the reachable set converges after three iterations; see Fig. 2a.

Suppose that the reachable set converges, which we shall refer to as $\mathcal{R}$. At $t = -T_s$, for a given initial state $x_0 \notin \mathcal{R}$, one needs to know what is the value of the constant control to apply over $[-T_s, 0]$ to keep the evader safe. Naturally, we can pick an arbitrary control $u \in \mathcal{C}_1^{\mathrm{q}}$ and compute the one step backward reachable set $\mathcal{BA}(\mathcal{R}, u)$ of $\mathcal{R}$. If $x_0 \in \mathcal{BA}(\mathcal{R}, u)$ then u is not safe to apply, because it will lead the dynamics into $\mathcal{R}$ in at most one sampling interval. On the contrary, if $x_0 \notin \mathcal{BA}(\mathcal{R}, u)$, then u is safe to apply because it keeps the dynamics out of $\mathcal{R}$ over that time interval.

Note that if $x_0 \notin \mathcal{R}$ then such a safe u always exists. Indeed, because $\mathcal{R} = \bigcap_{u\in\mathcal{C}_1^{\mathrm{q}}} \mathcal{BA}(\mathcal{R},u)$, one has $x_0 \notin \bigcap_{u\in\mathcal{C}_1^{\mathrm{q}}} \mathcal{BA}(\mathcal{R},u)$ which implies $\exists u \in \mathcal{C}_1^{\mathrm{q}}$ s.t. $x_0 \notin \mathcal{BA}(\mathcal{R},u)$.

Finally, if $x_0 \notin \mathcal{R}$ one can build a perpetually safe control law by choosing an admissible control at $t = -T_s$, applying it over $[-T_s, 0]$ and reseting the time to $t = -T_s$.

The reachable set has been computed over a numerical grid which is a finite subset of $\mathcal{X}$. We denote this grid by $\mathcal{G}$. For every $u \in \mathcal{C}_1^{\mathrm{q}}$, define Φ_u as the zero level set function of $\mathcal{BA}(\mathcal{R},u)$. We can check the admissibility of a fixed $u \in \mathcal{C}_1^{\mathrm{q}}$ for a given $x_0 \in \mathcal{G}$ via

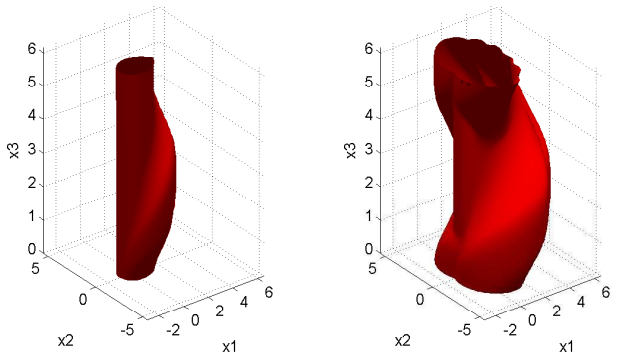
$$u \text{ is admissible } \Leftrightarrow\ \Phi_u(x_0) > 0. \tag{12}$$

C. The Freedom Set

Consider the set of states where all controls are admissible to apply over the current sampling interval. We call this set the *freedom set* and denote it by $\mathcal{F}$. We have that,

$$\mathcal{F} := \Big(\bigcup\nolimits_{u\in\mathcal{C}_1^{\mathrm{q}}} \mathcal{BA}(\mathcal{R},u)\Big)^c. \tag{13}$$

Computing the freedom set can be very useful when dimensioning the numerical grid $\mathcal{G}$ which is to be stored on board the UV. Indeed, $\mathcal{G}$ only needs to contain the complement of the freedom set, $\mathcal{F}^c$. In this case, if a state is out of $\mathcal{G}$, then the evader will not get any information about safety. But this is not a problem because it means that this state belongs to $\mathcal{F}$ and hence all controls belonging to the finite set $\mathcal{U}^{\mathrm{q}}$ are admissible. Computed for our particular application, the complement of the freedom set is shown in Fig. 2b.



(a) Converged unsafe reachable set $\mathcal{R}_3 =: \mathcal{R}$ after 3 iterations.

(b) Complement $\mathcal{F}^c$ of freedom set, enclosing the reachable set.

Fig. 2: Sets of interest for the collision avoidance experiment described in Section VI.

D. More General Control Laws Away From the Pursuer

The boundary of the freedom set $\mathcal{F}$ serves as an indicator of the amount of data that needs to be loaded onto the UV's memory. In the interior of $\mathcal{F}$ any control value in the quantized set $\mathcal{U}^{\mathrm{q}}$ is admissible. Away from the pursuer, however, the evader should have far greater flexibility in its choice of control policy. This allows the UV to be nominally controlled by any desired control strategy (for example, a feedback linearizing mechanism) taking values in the general set $\mathcal{U}$, and only switch back to the collision avoidance controller when safety is of concern.

To achieve this, we would need to identify—somewhat pessimistically—the regions of the state space where there exists a control policy of the evader taking value in $\mathcal{U}$ that, when combined with the action of the pursuer, could drive the evader inside of the unsafe reachable set $\mathcal{R}$ at some time during a sampling interval. That is, we wish to compute

$$\mathcal{BA}_{\mathrm{w}}(\mathcal{R}) := \big\{x_0 \in \mathcal{X} \colon \exists u \in \mathcal{C}_1,\ \exists d \in \mathcal{C}_2,\ \exists t \in [-T_s, 0],\ x_{x_0}^{u,d}(t) \in \mathcal{R}\big\}. \tag{14}$$

If $x_0 \in (\mathcal{BA}_{\mathrm{w}}(\mathcal{R}))^c$, then any control law in $\mathcal{C}_1$ is admissible over $[-T_s, 0]$; otherwise, one would have to be wary of safety and a safety-preserving control law within $\mathcal{C}_1^{\mathrm{q}}$ is needed to keep the evader out of harm's way.

The set $\mathcal{U}$ is infinite. To numerically compute $\mathcal{BA}_{\mathrm{w}}(\mathcal{R})$ we approximate $\mathcal{C}_1$ with

$$\widetilde{\mathcal{C}}_1 := \{u : [-T_s, 0] \to \mathcal{U} \text{ s.t. } u \text{ measurable on } [-T_s, 0]\}. \tag{15}$$

Since the class of Lebesgue measurable functions is a superset of the class of constants (thus, $\widetilde{\mathcal{C}}_1 \supseteq \mathcal{C}_1$), the set $\mathcal{BA}_{\mathrm{w}}(\mathcal{R})$ can now be conservatively approximated by

$$\widetilde{\mathcal{BA}}_{\mathrm{w}}(\mathcal{R}) := \big\{x_0 \in \mathcal{X} \colon \exists u \in \widetilde{\mathcal{C}}_1,\ \exists d \in \mathcal{C}_2,\ \exists t \in [-T_s, 0],\ x_{x_0}^{u,d}(t) \in \mathcal{R}\big\}, \tag{16}$$

ensuring that $\mathcal{BA}_{\mathrm{w}}(\mathcal{R}) \subseteq \widetilde{\mathcal{BA}}_{\mathrm{w}}(\mathcal{R})$. This set can simply be computed as usual with the difference that the corresponding Hamiltonian is now $H(x,p) = \inf_{(u,d)\in\mathcal{U}\times\mathcal{D}} p^T f(x,u,d)$.

IV. State Measurement and Choice of Control

We saw before how to compute and use $\Phi_u(x_0)$ to determine the safety of a control $u \in \mathcal{C}_1^{\mathrm{q}}$ at a given grid point $x_0 \in \mathcal{G}$. However, estimation of the current state yields a measurement x (assumed to be within the grid boundaries) that may not belong to $\mathcal{G}$ (which is only a finite subset of $\mathcal{X}$). The goal here is to approximate $\Phi_u(x)$ based on $\Phi_u(x_0)$ with x_0 being the closest grid neighbor of x.

The first approach for approximating $\Phi_u(x)$ is relatively more precise, although it is heavier to implement: it uses a first order Taylor expansion where partial derivatives are approximated with finite differences.

The second approach, which we will use in our experiments, is less precise but lighter to implement. It simply assumes that $\Phi_u(x) \approx \Phi_u(x_0)$. This is equivalent to saying that a control u is admissible for a state x if it is admissible for its neighboring grid point x_0. An estimate of the error resulting from this approach is described next.

A. Numerical Threshold for the Neighboring Approach

We know that for every $u \in \mathcal{C}_1^{\mathrm{q}}$, Φ_u (which is the level set function of $\mathcal{BA}(\mathcal{R},u)$) is K-Lipschitz. Hence, $|\Phi_u(x) - \Phi_u(x_0)| \le K|x - x_0|$. An analytical expression for K can be formulated based on the relation between the terminal value optimal control problem and the level set function of

the reachable set [15]. For brevity, however, we refrain from discussing these results in the current paper.

Let us assume the grid has constant spacings in every direction. Let Δx_i be the spacing along the x_i direction $\forall i \in \{1,2,3\}$. Then, since x_0 is the closest grid point to x, we have $|x-x_0| \leq \frac{1}{2}\sqrt{(\Delta x_1)^2+(\Delta x_2)^2+(\Delta x_3)^2}$. Denote the numerical error as $\varepsilon := K\frac{1}{2}\sqrt{(\Delta x_1)^2+(\Delta x_2)^2+(\Delta x_3)^2}$ so that for any x that is within the bounds of the grid, for its corresponding closest grid neighbor x_0, and $\forall u \in \mathcal{C}_1^{\mathrm{q}}$ one has $|\Phi_u(x) - \Phi_u(x_0)| \leq \varepsilon$. In (12) we saw a decision criterion based on the sign of $\Phi_u(x_0)$. We can adapt this criterion to take into account the fact that the estimated state x may fall in between grid points:

$$u \text{ is admissible at } x \;\Leftrightarrow\; \Phi_u(x_0) > \varepsilon, \tag{17}$$

where x_0 is the closest neighboring grid point to x.

This new decision threshold is going to reduce the size of the freedom set $\mathcal{F}$. Intuitively, some controls that used to be admissible before are no longer admissible since our new decision criterion is more conservative. This implies that the evader starts having a safety reaction earlier than before. The new freedom set is $\mathcal{F} = \{x \in \mathcal{X} : \min_{u\in\mathcal{C}_1^{\mathrm{q}}} \Phi_u(x) > \varepsilon\}$.

V. Online algorithm

We implemented our collision avoidance algorithm on ground robots via the Robot Operating System (ROS) [14] in C++. Based on the presented offline computations, we form our online algorithm as follows. The main data are the values of $\Phi_u(x_0),\ \forall(u,x_0) \in \mathcal{C}_1^{\mathrm{q}} \times \mathcal{G}$ which we store into an array and load onto the UV's memory.

1) At $t = -T_s$, update measurements, which are positions and headings of the pursuer and the evader. A heading is obtained by measuring two successive positions at $100\,\mathrm{Hz}$. Reconstruct 3D state vector x.
2) If the estimated vector falls within the grid's bounds, find the closest grid neighbor x_0; Otherwise, all controls are admissible and proceed to step 4.
3) For every $u \in \mathcal{C}_1^{\mathrm{q}}$, check the sign of $\Phi_u(x_0) - \varepsilon$. This gives us the set of admissible, safety-preserving controls.
4) Among these controls, choose one that fulfills a given task (such as reaching an objective).
5) Apply the chosen control law during the current time interval ($t = -T_s$ to $t = 0$). Reset the time variable $t \leftarrow -T_s$, and restart from step 1.

We robustify our online algorithm against measurement noise, delay, and numerical errors by increasing the value of the threshold ε. For our experiment we empirically found this value to be $\varepsilon = 0.5\,\mathrm{m}$.

Finally, it may happen that due to experimental approximations the obstacle would come too close to the boundary of the reachable set (though it still remains outside of it). Then at step 3 of the above algorithm, there would not be any $u \in \mathcal{C}_1^{\mathrm{q}}$ verifying $\Phi_u(x_0) - \varepsilon > 0$. In this case the evader should play its best effort control defined as $u_{\text{best}}(x_0) \in \arg\max_{u\in\mathcal{C}_1^{\mathrm{q}}}\{\Phi_u(x_0)\}$.

VI. Experiments

A. Setup

We used two Pioneer ground robots [10]. The first one was automated with the collision avoidance algorithm and played the role of the evader. The second one was remotely controlled by a human operator and played the role of the pursuer. We used VICON system to get position measurements. We set the maximum speeds of the robots to $U_1 = D_1 = 0.15\,\mathrm{m/s}$ and the maximum yaw rates to $U_2 = D_2 = 1\,\mathrm{rad/s}$ (these values were obtained based on the robots' physical capacities). Finally, we assumed that Pioneer ground robots follow unicycle models, i.e. they can be controlled in terms of velocity vector's magnitude and angular rate. This assumption is fairly accurate when considering this type of robot which has a physical heading (the direction from the rear to the front). That being said, the robot is not using a steering mechanism (as one would expect in a vehicle that follows a unicycle model), but instead it is applying differential rotating speeds to the wheels in order to turn (which introduces additional drifting). The kinematics are described by the following simple nonlinear model:

$$\begin{cases} \dot{x} = v\cos(\psi) \\ \dot{y} = v\sin(\psi) \\ \dot{\psi} = \omega. \end{cases} \tag{18}$$

Therefore, we assume that we can directly control the speed v and the yaw rate ω.

The evader's objective is to reach the plane's origin. At every iteration of the algorithm, it synthesizes the set of safe yaw rate values, and picks one that would orient its velocity vector the closest toward the direction of the objective. Finally, to simplify the implementations, we fixed the speed of the evader to its maximum value. Therefore, the only actual control input is the yaw rate. The set of considered control values $\mathcal{U}^{\mathrm{q}}$ that appears in the definition of $\mathcal{C}_1^{\mathrm{q}}$ in (1) is

$$\mathcal{U}^{\mathrm{q}} = \{0.15\} \times \{-1\,, -0.5\,, 0\,, 0.5\,, 1\}, \tag{19}$$

where the speed has been fixed to $0.15\,\mathrm{m/s}$ and the yaw rate values are given in rad/s. As discussed earlier, such quantization maintains conservatism and formalism. Qualitatively, the inputs drawn from $\mathcal{U}^{\mathrm{q}}$ allow the evader to go straight and to turn left or right more or less quickly.

Finally, we fixed the sampling time as $T_s = 1\,\mathrm{s}$. The sampling frequency of the collision avoidance algorithm can be chosen as desired (though it must be the same in both offline and online calculations) as long as it is slower than that of the internal lower-level control unit inside of the robots, as well as the sampling frequency of the measurement sensors used in the state estimator. Also, we fixed the safety radius as $R = 0.8\,\mathrm{m}$. This value has been chosen based on the size of the robots, so that as long as the pursuer's center is further than $0.8\,\mathrm{m}$ away from the evader's center then collision has not occurred.

With the previously described numerical values, the reachable set converges after 3 iterations (Fig. 2a). The grid's size was of 60 points in each direction. The offline computational

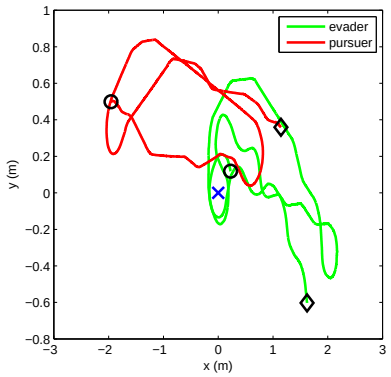


Fig. 3: Absolute trajectories in x-y plane. The objective is represented by a blue cross. Small black circles/diamonds represent the start/end of trajectories.

time of the converged unsafe reachable set was approximately 15 mins.[1] We used 8 MB of on board memory to store the values of the level set function Φ_u on the UV. We apply the collision avoidance algorithm described in Section V.

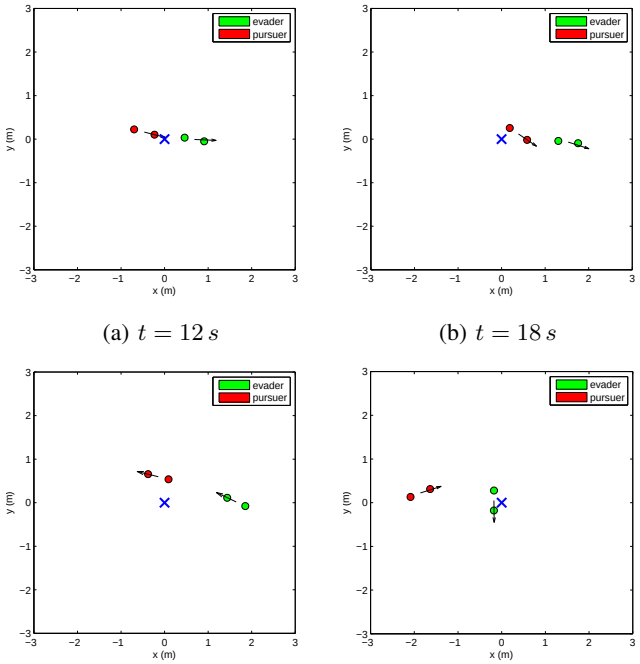
B. Results

The absolute trajectories of the evader and the pursuer are measured and plotted in Fig. 3, while Fig. 4 shows various snapshots of these trajectories so as to highlight the behavior of the evader when the pursuer is coming towards it, and as a comparison, when the pursuer is going away from it. We plotted two points for each object. One is located at the rear end of the vehicle, the other at the front. The arrow represents the direction of the velocity vector (computed for the center point of the vehicle, by measuring two successive positions of the vehicles via sensors) and clearly determines what point is at the rear and what point is at the front. One can observe the safety reaction of the evader that is fleeing when the pursuer is coming close. Also, when the pursuer is going away, the evader attempts to go back to the objective.

Finally these snapshots also highlight that physical and velocity headings (the physical heading is the direction from the rear to the front end, while the velocity heading is represented by an arrow) are not exactly the same, especially during turns. This phenomenon is shown in Fig. 5, which highlights that the ground robot follows a close but not perfect unicycle model (recall that we are controlling the physical heading, although the velocity heading is really of importance). To formalize the effect of this discrepancy on the control algorithm one would need to study the dynamics of the ground robot, hoping to find a relation between the two headings' evolutions. In our experiments, we assume (and empirically validate) that the discrepancy is negligible.

The relative trajectory of the pursuer in evader's frame is plotted in Fig. 6. One can notice that the pursuer never enters the avoid set, which is an expected behavior and validates the proposed collision avoidance algorithm. Various snapshots of this relative trajectory are displayed in Fig. 7.

[1]Computed on Matlab R2009b running on an Intel Core i7 CPU clocked at 2.67 GHz with 8 GB of RAM.



(a) $t = 12\,s$ (b) $t = 18\,s$

(c) $t = 30\,s$ (d) $t = 48\,s$.

Fig. 4: Absolute trajectory snapshots. The objective is represented by a blue cross. Arrows represent velocity vectors.

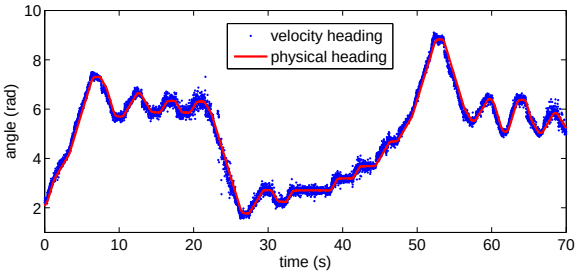


Fig. 5: Physical versus velocity headings of evader.

The inner set is the unsafe reachable set $\mathcal{R}$ and the outer set is $\mathcal{F}^c$, the complement of the freedom set. The arrow represents the relative direction of the pursuer's velocity vector in the evader's frame. When the pursuer enters the outer set (i.e. when it exits the freedom set $\mathcal{F}$), not all controls are admissible for the evader, and the evader has to choose a safety-preserving one. Although the pursuer's relative velocity direction is pointing toward the evader, the pursuer is never able to enter the unsafe set, highlighting the fact that the evader is reacting in an appropriate manner.

Fig. 8 shows the corresponding synthesized control policy (i.e. commands) sent to the evader, and the evader's actual responses to these commands. The green crosses represent the control values chosen when considering safety (i.e. when the state vector belongs to $\mathcal{F}^c$). This plot also highlights the existence of an internal dynamical delay: The synthesized command is PWC, but the actual response is continuous, with connection slopes between constant pieces. Although we do

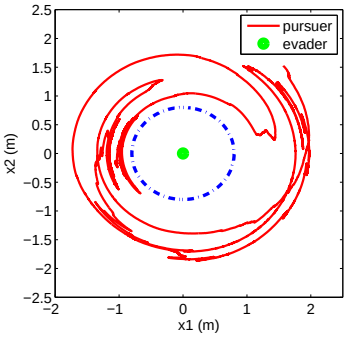
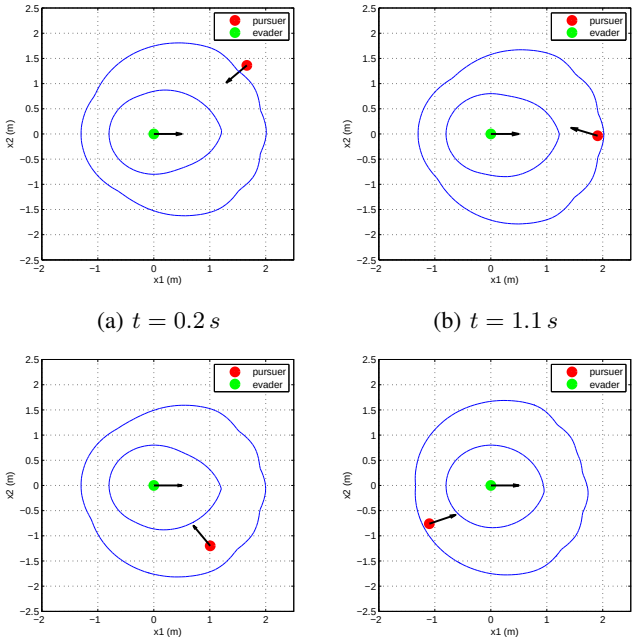


Fig. 6: Relative trajectory of pursuer in evader's frame. The avoid set is represented by the dashed blue circle.



(a) $t = 0.2\,s$ (b) $t = 1.1\,s$

(c) $t = 2.5\,s$ (d) $t = 4.3\,s$.

Fig. 7: Relative trajectory snapshots.

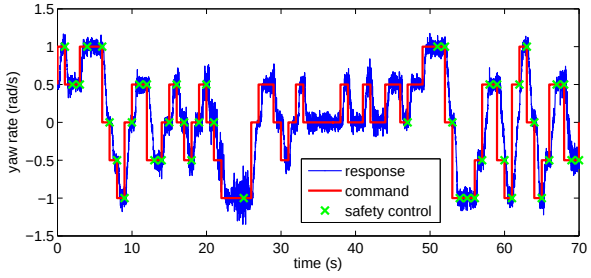


Fig. 8: The evader's synthesized yaw rate command and its actual input response.

not explicitly account for this delay, the chosen sampling time seems to be large enough that the general response during constant yaw rate phases is satisfactory and is close to the given command. In future work, we plan to examine different methods that take into account this dynamical delay, by essentially considering the maximum variational capacities of the robot in terms of yaw rate response.

VII. Online restriction of pursuer's inputs

So far we have let the pursuer's speed d_1 vary as it desires within the set $[0, D_1]$. The lack of any reasonable assumptions on this input during the offline computations of the collision avoidance algorithm may yield an excessively conservative online control policy for the evader. This is due to the fact that the pursuer's input is technically allowed to oscillate/chatter infinitely frequently since the only assumption we impose is for it to be Lebesgue measurable. To be more realistic, one can take into account the maximum acceleration of the pursuer and restrict its set of possible speeds over one sampling interval.

Let us assume that the pursuer has a finite maximum acceleration (speed derivative) $a_{\max} < \infty$. At $t = -T_s$, the evader has to take a control decision to apply during the interval $[-T_s, 0]$. Let us call d_{1_0} the initial speed value of the pursuer at $t = -T_s$ and assume that it verifies $d_{1_0} \in [a\,, b]$, with $0 \leq a \leq b \leq D_1$. Then, over $[-T_s, 0]$, one has a new restricted interval for the control input of the pursuer described by

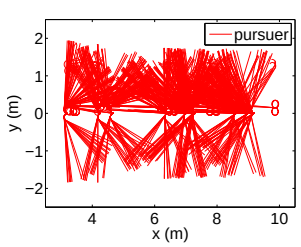
$$d_1 \in [\max(0\,, a - T_s a_{\max})\,, \min(D_1\,, b + T_s a_{\max})]. \quad (20)$$

The restriction of how the pursuer's input is allowed to vary over every sampling interval reduces conservatism of the collision avoidance algorithm. During offline computations of $\mathcal{BA}(\mathcal{R}, u)$, $\forall u \in \mathcal{C}_1^{\mathrm{q}}$, this restriction is accounted for by constraining the set of pursuer's speed input values over which the Hamiltonian is minimized.

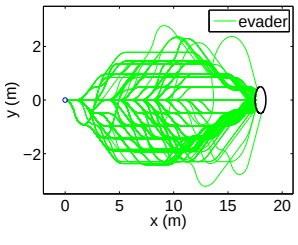
A. Validation via Simulations

To demonstrate the effect of restricting the pursuer's input sets on conservatism of the proposed collision avoidance algorithm, we employ simulations. We implemented the pursuer-evader game and our algorithm in Matlab. The maximum input values are $U_1 = D_1 = 1\,\mathrm{m/s}$ and $U_2 = D_2 = 1\,\mathrm{rad/s}$. The safety radius is $R = 1\,\mathrm{m}$ and the sampling time is $T_s = 1\,\mathrm{s}$. The evader is starting from the origin, and has to reach the location at $(x = 0\,\mathrm{m}, y = 18\,\mathrm{m})$. Throughout the simulation we fixed the pursuer's speed $d_1 = 0.1\,\mathrm{m/s}$, although the evader does not know that this value is constant. At the beginning of every time interval, the evader measures the initial speed of the pursuer and then computes the possible ensuing values for the pursuer's speed over the time interval. We chose $a = 0\,\mathrm{m/s}$ and $b = 0.1\,\mathrm{m/s}$ (values that appear in the restricted interval (20)). The maximum acceleration of the pursuer has been fixed to $a_{\max} = 0.1\,\mathrm{m/s^2}$. We randomly generated 1000 trajectories for the pursuer (Fig. 9a) and obtained the ensuing trajectories of evader (Fig. 9b).

To quantify the effect of the restriction of the pursuer's inputs on the evader's control law, we introduce two cost indicators which are the time to reach the objective for the evader $t_{\max}$ and the energy of its yaw rate control



(a) Random trajectories of pursuer.



(b) Ensuing trajectories of evader. Blue circle is starting point, black ellipse is objective.

Fig. 9: Simulations for pursuer's speed restriction. The collision avoidance algorithm generates control policies that on average require less energy expenditure.

$\int_0^{t_{\max}} u_2^2(t)\,dt$. We average these cost indicators over all evader's trajectories and obtain the following results:

Method	Energy (rad^2/s)	Time (s)
With restriction	1.94	19.18
Without restriction	2.63	19.32

We can see that for equivalent times to reach, the evader requires less energy (on average) to reach the objective when the pursuer's input is restricted as above.

VIII. Conclusions and Future Work

We presented a practical reachability-based collision avoidance algorithm in the framework of a planar two-player pursuer-evader game. We implemented our algorithm on an automated unmanned vehicle (UV) that seeks to reach an objective in a safe fashion despite the actions of a moving obstacle. Our algorithm formally takes into account the sampled-data nature of the underlying system by *not* ignoring the fact that physical systems evolve continuously in time while digital controllers can only make decisions that are based on sensor measurements and are discrete in time. (A provably safe controller must be able to account for the inter-sample behavior of the system.) As such, the presented collision avoidance algorithm generates piecewise constant (PWC) control laws that guarantee safety of the UV, while not making any strict assumptions on the behavior of the obstacle. Costly computations are performed offline, leaving the online calculations manageable on an embedded microcontroller. To achieve this, we introduced the notion of freedom set that is useful in optimally dimensioning the numerical grid (used by the level set methods) which is to be stored on board the UV. Discussions surrounding the ability of the vehicle to apply more general control strategies away from the obstacle were also presented. In addition, our algorithm considers the numerical error introduced by the fact that state estimates could fall in between grid points. This enabled us to work with a relatively coarse grid (and consequently, a smaller online lookup table) while still maintaining formalism to a great extent.

We validated the presented algorithm via experiments on Pioneer ground robots. That being said, this algorithm can be directly used for any other vehicle that is governed by the unicycle model (e.g., a planar fixed wing aircraft). In all other cases (consider the quadrotor, for instance), to be able to use similar analysis based on the unicycle model (which has the advantage of yielding relatively low dimensional dynamics in relative coordinates), an intermediate step would be needed to fill the gap between the vehicle's actual dynamics and the unicycle model. We are currently exploring this avenue.

The presented algorithm generates PWC commands that, in rare occasions, may be difficult to realize by a physical system. This is due to the fact that, without additional constraints, a PWC signal can vary greatly at the discontinuities, while most dynamical systems generally can only tolerate limited variations on their actuation input. This warrants an algorithm that is capable of taking into account the maximum variational capacities of the vehicle. We plan to investigate such an algorithm in future work.

References

[1] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[2] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in *Proc. IEEE Int. Conf. Robot. and Autom.*, vol. 4, 2001, pp. 3716–3721.

[3] A. Wu and J. P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous robots*, vol. 32, no. 3, pp. 227–242, 2012.

[4] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.

[5] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proc. Contr. Theor. Ap.*, vol. 153, no. 5, pp. 575–590, 2006.

[6] Y. Guo and T. Tang, "Optimal trajectory generation for nonholonomic robots in dynamic environments," in *IEEE Int. Conf. Robot. and Autom.*, 2008, pp. 2552–2557.

[7] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Automat. Contr.*, vol. 50, no. 7, pp. 947–957, 2005.

[8] C. J. Tomlin, I. M. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification and control of hybrid systems," *Proc. of the IEEE*, vol. 91, no. 7, pp. 986–1001, 2003.

[9] I. M. Mitchell and J. A. Templeton, "A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems," in *Hybrid Syst.: Comput. Contr.* Springer, 2005, pp. 480–494.

[10] "ADEPT MOBILEROBOTS LLC," http://www.mobilerobots.com/researchrobots/p3at.aspx.

[11] J. Ding, E. Li, H. Huang, and C. J. Tomlin, "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances," in *Proc. IEEE Int. Conf. Robot. and Autom.*, 2011, pp. 2160–2165.

[12] J. Ding and C. J. Tomlin, "Robust reach-avoid controller synthesis for switched nonlinear systems," in *Proc. IEEE Conf. Decis. Contr.*, 2010, pp. 6481–6486.

[13] I. M. Mitchell, S. Kaynama, M. Chen, and M. Oishi, "Safety preserving control synthesis for sampled data systems," *Nonlin. Anal.: Hybrid Syst.*, vol. 10, pp. 63–82, 2013.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[15] L. C. Evans, *Partial Differential Equations*. Providence, RI: American Mathematical Society, 1998, ISBN: 0-8218-0772-2.