Periodically Controlled Hybrid Systems

Verifying A Controller for An Autonomous Vehicle

Tichakorn Wongpiromsarn[1], Sayan Mitra[2],
Richard M. Murray[1], and Andrew Lamperski[1]

[1] California Institute of Technology
[2] University of Illinois at Urbana Champaign

Abstract. This paper introduces Periodically Controlled Hybrid Automata (PCHA) for describing a class of hybrid control systems. In a PCHA, control actions occur roughly periodically while internal and input actions may occur in the interim changing the discrete-state or the setpoint. Based on periodicity and subtangential conditions, a new sufficient condition for verifying invariance of PCHAs is presented. This technique is used in verifying safety of the planner-controller subsystem of an autonomous ground vehicle, and in deriving geometric properties of planner generated paths that can be followed safely by the controller under environmental uncertainties.

1 Introduction

Alice, an autonomous vehicle built at Caltech, successfully accomplished two of the three tasks at the National Qualifying Event of the 2007 DARPA Urban Challenge [4], [17], [5]. In executing the third task, which involved making left-turns while merging into traffic, its behavior was unsafe and almost led to a collision. Alice was stuck at the corner of a sharp turn dangerously stuttering in the middle of an intersection.

This behavior, it was later diagnosed, was caused by bad interactions between the *reactive obstacle avoidance subsystem (ROA)* and the relatively slowly reacting *path planner*. The planner incrementally generates a sequence of waypoints based on the road map, obstacles, and the mission goals. The ROA is designed to rapidly decelerate the vehicle when it gets too close to (possibly dynamic) obstacles or when the deviation from the planned path gets too large. Finally, for protecting the steering wheel, Alice's low-level controller limits the rate of steering at low speeds. Thus, accelerating from a low speed, if the planner produces a path with a sharp left turn, the controller is unable to execute the turn closely. Alice deviates from the path; the ROA activates and slows it down. This cycle continues leading to stuttering. For avoiding this behavior, the planner needs to be aware of the constraints imposed by the controller.

Finding this type of design bugs in hybrid control systems is important and challenging. While real world hybrid systems are large and complex, they are also engineered, and hence, have more structure than general hybrid automata [1].

Although restricted subclasses that are amenable to algorithmic analysis have been identified, such as rectangular-initialized [6], o-minimal [8], planar [13], and stormed [15] hybrid automata, they are not representative of restrictions that arise in engineered systems. With the motivation of abstractly capturing a common design pattern in hybrid control systems, such as Alice, and other motion control systems [11], in this paper, we study a new subclass of hybrid automata. Two main contributions of this paper are the following:

First, we define a class of hybrid control systems in which certain *control actions* occur roughly periodically. Each control action sets the *controlling input* to the plant or the physical process. In the interval between two consecutive control actions, the state of the system evolves continuously and discretely, but the control input remains constant. In particular, discrete state changes triggered by an external source may changes the waypoint or the set-point of the controller, which in turn may influence the computation of the next control input. For this class of *periodically controlled hybrid systems*, we present a sufficient condition for verifying invariant properties. The key requirement in applying this condition is to identify subset(s) C of the candidate invariant set $\mathcal{I}$, such that if the control action occurs when the system state is in C, then the subsequent control output guarantees that the system remains in $\mathcal{I}$ for the next period. The technique does not require one to solve the differential equations, instead, it relies on checking conditions on the periodicity and the subtangential condition at the boundary of $\mathcal{I}$. We are currently exploring the possibility of automating such checks using quantifier elimination [3] and optimization [14].

Secondly, we apply the above technique to verify a sequence of invariant properties of the planner-controller subsystem of Alice. From these invariants, we are able to deduce safety. That is, the deviation —distance of the vehicle from the planned path—remains within a certain constant bound. In the process, we also derive geometric properties of planner paths that guarantee that they can be followed safely by the vehicle.

The remainder of the paper is organized as follows: In Section 2 we briefly present the key definitions for the hybrid I/O automaton framework. In Section 3 we present PCHA and a sufficient condition for proving invariance. In Sections 4 and 5 we present the formal model and verification of Alice's Controller-Vehicle subsystem. Owing to limited space, complete proofs for identifying the class of safe planner paths appear in the full version of the paper available from [16].

2 Preliminaries

We use the Hybrid Input/Output Automata (HIOA) framework of [9,7] for modeling hybrid systems and the state model-based notations introduced in [10]. A Structured Hybrid I/O Automaton (SHIOA) is a non-deterministic state machine whose state may change instantaneously through a transition, or continuously over an interval of time following a *trajectory*.

Let V be a set of variables. Each variable $v \in V$ is associated with a *type*. The set of valuations of V is denoted by $val(V)$. For a valuation $\mathbf{v} \in Val(V)$

of set of variables V, its restriction to a subset of variables $Z \subseteq V$ is denoted by $\mathbf{v} \lceil Z$. A variable may be *discrete* or *continuous*. A *trajectory* for a set of variables V models continuous evolution of the values of the variables over an interval of time. Formally, a trajectory τ is a map from a left-closed interval of $\mathbb{R}_{\geq 0}$ with left endpoint 0 to $val(V)$. The domain of τ is denoted by $\tau.dom$. The *first state* of τ, $\tau.\mathsf{fstate}$, is $\tau(0)$. A trajectory τ is *closed* if $\tau.dom = [0, t]$ for some $t \in \mathbb{R}_{\geq 0}$, in which case we define $\tau.\mathsf{ltime} \triangleq t$ and $\tau.\mathsf{lstate} \triangleq \tau(t)$. For a trajectory τ for V, its restriction to a subset of variables $Z \subseteq V$ is denoted by $\tau \downarrow Z$.

For given sets of input U, output Y, and internal X variables, a *state model* $\mathcal{S}$ is a triple $(\mathscr{F}, Inv, Stop)$, where (a) $\mathscr{F}$ is a collection of Differential and Algebraic Inequalities (DAIs) involving the continuous variables in U, Y, and X, and (b) Inv and $Stop$ are predicates on X called *invariant condition* and *stopping condition* of $\mathcal{S}$. Components of $\mathcal{S}$ are denoted by $\mathscr{F}_{\mathcal{S}}$, $Inv_{\mathcal{S}}$ and $Stop_{\mathcal{S}}$. $\mathcal{S}$ defines a set of trajectories, denoted by $traj(\mathcal{S})$, for the set of variables $V = X \cup U \cup Y$. A trajectory τ for V is in the set $trajs(\mathcal{S})$ iff (a) the discrete variables in $X \cup Y$ remain constant over τ; (b) the restriction of τ on the continuous variables in $X \cup Y$ satisfies all the DAIs in $\mathscr{F}_{\mathcal{S}}$; (c) at every point in time $t \in dom(\tau)$, $(\tau \downarrow X)(t) \in Inv$; and (d) if $(\tau \downarrow X)(t) \in Stop$ for some $t \in dom(\tau)$, then τ is closed and $t = \tau.\mathsf{ltime}$.

Definition 1. *A* Structured Hybrid I/O Automaton (SHIOA) $\mathcal{A}$ *is a tuple* $(V, Q, Q_0, A, \mathcal{D}, \mathscr{S})$ *where (a) V is a set of variables partitioned into sets of* internal X, output Y *and* input U *variables; (b) $Q \subseteq val(X)$ is a set of* states *and $Q_0 \subseteq Q$ is a nonempty set of* start states*; (c) A is a set of actions partitioned into sets of* internal H, output O *and* input I *actions; (d) $\mathcal{D} \subseteq Q \times A \times Q$ is a set of* discrete transitions*; and (e) $\mathscr{S}$ is a collection of* state models *for U, Y, and X, such that for every $\mathcal{S}, \mathcal{S}' \in \mathscr{S}$, $Inv_{\mathcal{S}} \cap Inv_{\mathcal{S}'} = \emptyset$ and $Q \subseteq \bigcup_{\mathcal{S} \in \mathscr{S}} Inv_{\mathcal{S}}$. In addition, $\mathcal{A}$ satisfies:* **E1** *Every input action is enabled at every state.* **E2** *Given any trajectory υ of the input variables U, any $\mathcal{S} \in \mathscr{S}$, and $\mathbf{x} \in Inv_{\mathcal{S}}$, there exists $\tau \in trajs(\mathcal{S})$ starting from $\mathbf{x}$, such that either (a) $\tau \downarrow U = \upsilon$, or (b) $\tau \downarrow U$ is a proper prefix of υ and some action in $H \cup O$ is enabled at $\tau.\mathsf{lstate}$.*

For a set of state variables X, a state $\mathbf{x}$ is an element of $Val(X)$. We denote the valuation of a variable $y \in X$ at state $\mathbf{x}$, by the usual (.) notation $\mathbf{x}.y$. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$ or as $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ when $\mathcal{A}$ is clear from the context. An action a is said to *enabled* at $\mathbf{x}$ if there exists $\mathbf{x}'$ such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$. We denote the components of a SHIOA $\mathcal{A}$ by $X_{\mathcal{A}}, Y_{\mathcal{A}}$, etc.

An execution of $\mathcal{A}$ records the valuations of all its variables and the occurrences of all actions over a particular run. An *execution fragment* of $\mathcal{A}$ is a finite or infinite sequence $\alpha = \tau_0 a_1 \tau_1 a_2 \ldots$ such that for all i in the sequence, $a_i \in A$, $\tau \in trajs(\mathcal{S})$ for some $\mathcal{S} \in \mathscr{S}$, and $\tau_i.\mathsf{lstate} \xrightarrow{a_{i+1}} \tau_{i+1}.\mathsf{fstate}$. An execution fragment is an *execution* if $\tau_0.\mathsf{fstate} \in Q_0$. An execution is *closed* if it is finite and the last trajectory in it is closed. The first state of α, $\alpha.\mathsf{fstate}$, is $\tau_0.\mathsf{fstate}$, and for a closed α, its last state, $\alpha.\mathsf{lstate}$, is the last state of its last trajectory. The *limit time* of α, $\alpha.\mathsf{ltime}$, is defined to be $\sum_i \tau_i.\mathsf{ltime}$. The set of executions and reachable states of $\mathcal{A}$ are denoted by $\mathsf{Execs}_{\mathcal{A}}$ and $\mathsf{Reach}_{\mathcal{A}}$. A set of states $I \subseteq Q$ is said to be an *invariant* of $\mathcal{A}$ iff $\mathsf{Reach}_{\mathcal{A}} \subseteq I$.

3 Periodically Controlled Hybrid Systems

In this section, we define a subclass of SHIOAs frequently encountered in applications involving sampled control systems and embedded systems with periodic sensing and actuation. The main result of this section, Theorem 1, gives a sufficient condition for proving invariant properties of this subclass.

A *Periodically Controlled Hybrid Automaton (PCHA)* is an SHIOA with a set of (control) actions which occur roughly periodically. For the sake of simplicity, we consider the PCHAs of the form shown in Figure 1, however, Theorem 1 generalizes to PCHAs with other input, output, and internal actions.

Let $\mathcal{X} \subseteq \mathbb{R}^n$, for some $n \in \mathbb{N}$, and $\mathcal{L}, \mathcal{Z}$, and $\mathcal{U}$ be arbitrary types. Four key variables of PCHA $\mathcal{A}$ are (a) *continuous state* variable s of type $\mathcal{X}$, initialized to x_0, (b) discrete state (*location* or *mode*) variable *loc* of type $\mathcal{L}$, initialized to l_0, (c) *command* variable z of type $\mathcal{Z}$, initialized to z_0, and (d) *control* variable u of type $\mathcal{U}$, initialized to u_0. The *now* and *next* variables together trigger the $\mathsf{control}$ action periodically.

PCHA $\mathcal{A}$ has two types of actions: (a) through input action update $\mathcal{A}$ learns about new externally produced input commands such as set-points, waypoints. When an $\mathsf{update}(z')$ action occurs, z' is recorded in the command variable z. (b) The $\mathsf{control}$ action changes the control variable u. This action occurs roughly periodically starting from time 0; the time gap between two successive occurrences is within $[\Delta_1, \Delta_1 + \Delta_2]$ where $\Delta_1 > 0, \Delta_2 \geq 0$. When $\mathsf{control}$ occurs, *loc* and s are computed as a function of their current values and that of z, and u is computed as a function of the new values of *loc* and s.

For each $l \in \mathcal{L}$ the continuous state s evolves according to the trajectories specified by state model $smodel(l)$, i.e., according to the differential equation $\dot{s} = f_l(s, u)$. The timing of $\mathsf{control}$ behavior is enforced by the precondition of $\mathsf{control}$ and the stopping condition of the state models.

```
signature
internal control, input update($z' : \mathcal{Z}$)

variables
internal $s : \mathcal{X} := x_0$
internal discrete $loc : \mathcal{L} := l_0$,
  $z : \mathcal{Z} := z_0$, $u : \mathcal{U} := u_0$
internal $now : \mathbb{R}_{\geq 0} := 0$,
  $next : \mathbb{R}_{\geq 0} := -\Delta_2$

transitions
input update($z'$)
  eff $z := z'$

internal control
  pre $now \geq next$
  eff $next := now + \Delta_1$
    $\langle loc, s \rangle := h(loc, s, z)$;  $u := g(loc, s)$

trajectories
trajdef $smodel(l : \mathcal{L})$
  invariant $loc = l$
  evolve $d(now) = 1$; $d(s) = f_l(s, u)$
  stop when $now = next + \Delta_2$
```

Fig. 1. PHCA with parameters Δ_1, Δ_2, g, h, $\{f_l\}_{l \in \mathcal{L}}$. See, for example, [10] for the description of the language.

Describing and proving invariants. Given a candidate invariant set $\mathcal{I} \subseteq Q$, we are interested in verifying that $\mathsf{Reach}_A \subseteq \mathcal{I}$. For continuous dynamical systems, checking the well-known subtangential condition (see, for example [2]) provides a sufficient condition for proving invariance of a set $\mathcal{I}$ that is bounded by a closed surface. Theorem 1 provides an analogous sufficient condition for PCHAs. In general, however, invariant sets $\mathcal{I}$ for PCHAs have to be defined by a collection of functions instead of a single function. For each mode $l \in \mathcal{L}$, we assume that the invariant set $I_l \subseteq \mathcal{X}$ for the continuous state is defined by a collection of m *boundary functions* $\{F_{lk}\}_{k=1}^{m}$, where m is some natural number and each $F_{lk} : \mathcal{X} \to \mathbb{R}$ is a differentiable function[3]. Formally,

$$I_l \triangleq \{s \in \mathcal{X} \mid \forall k \in \{1, \ldots, m\}, F_{lk}(s) \geq 0\} \text{ and } \mathcal{I} \triangleq \{\mathbf{x} \in Q | \ \mathbf{x}.s \in I_{\mathbf{x}.loc}\}.$$

Note that $\mathcal{I}$ does not restrict the values of the command or the control variables. Lemma 1 modifies the standard inductive technique for proving invariance, so that it suffices to check invariance with respect to Control transitions and Control-free execution fragments. The proof appears in the full version [16].

Lemma 1. *Suppose $Q_0 \subseteq \mathcal{I}$ and the following two conditions hold:*

(a) (Control steps) For each state $\mathbf{x}, \mathbf{x}' \in Q$, if $\mathbf{x} \overset{\mathsf{control}}{\to} \mathbf{x}'$ and $\mathbf{x} \in \mathcal{I}$ then $\mathbf{x}' \in \mathcal{I}$,
(b) (Control-free fragments) For each closed execution fragment $\beta = \tau_0$ $\mathsf{update}(z_1)$ τ_1 $\mathsf{update}(z_2)$ $\ldots$ τ_n starting from a state $\mathbf{x} \in \mathcal{I}$ where each $z_i \in \mathcal{Z}$, if $\mathbf{x}.next - \mathbf{x}.now = \Delta_1$ and $\beta.\mathsf{ltime} \leq \Delta_1 + \Delta_2$, then $\beta.\mathsf{lstate} \in \mathcal{I}$.

Then $\mathsf{Reach}_{\mathcal{A}} \subseteq \mathcal{I}$.

Invariance of control steps can often be checked through case analysis which can be partially automated using a theorem prover [12]. The next key lemma provides a sufficient condition for proving invariance of control-free fragments. Since, control-free fragments do not change the valuation of the *loc* variable, for this part, we fix a value $l \in \mathcal{L}$. For each $j \in \{1, \ldots m\}$, we define the set ∂I_j to be part of the set I_l where the function F_{lj} vanishes. That is, $\partial I_j \triangleq \{s \in \mathcal{X} \mid F_{lj}(s) = 0\}$. In this paper, we call ∂I_j the j^{th} boundary of I_l even though strictly speaking, the j^{th} boundary of I_l is only a subset of ∂I_j according on the standard topological definition. Similarly, we say that the *boundary* of I_l, is $\partial I_l = \bigcup_{j \in \{1,\ldots,m\}} \partial I_j$.

Lemma 2. *Suppose that there exists a collection $\{C_j\}_{j=1}^{m}$ of subsets of I_l such that the following conditions hold:*

(a) (Subtangential) For each $s_0 \in I_l \setminus C_j$ and $s \in \partial I_j$, $\frac{\partial F_{lj}(s)}{\partial s} \cdot f_l(s, g(l, s_0)) \geq 0$.
(b) (Bounded distance) $\exists\ c_j > 0$ such that $\forall\ s_0 \in C_j, s \in \partial I_j$, $||s - s_0|| \geq c_j$.
(c) (Bounded speed) $\exists\ b_j > 0$ such that $\forall\ s_0 \in C_j, s \in I_l$, $||f_l(s, g(l, s_0))|| \leq b_j$,
(d) (Fast sampling) $\Delta_1 + \Delta_2 \leq \min_{j \in \{1,\ldots,m\}} \frac{c_j}{b_j}$.

Then, any control*-free execution fragment starting from a state in I_l where $next - now = \Delta_1$, remains within I_l.*

[3] Identical size m of the collections simplifies our notation; different number of boundary functions for different values of l can be handled by extending the theorem in an obvious way.

In Figure 2, the control and control-free fragments are shown by bullets and lines. A fragment starting in $\mathcal{I}$ and leaving $\mathcal{I}$, must cross ∂I_1. Condition (a) guarantees that if u is evaluated outside C_1, then the fragment does not leave I_l because when it reaches ∂I_1, the vector field governing its evolution points inwards with respect to ∂I_1. For a fragment starting inside C_1, condition (b) and (c) guarantee that it takes finite time before it reaches ∂I_1 and condition (d) guarantees that this finite time is at least $\Delta_1 + \Delta_2$; thus, before the trajectory crosses ∂I_1, u is evaluated again.

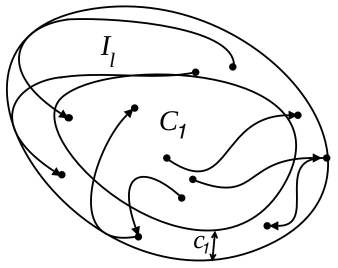


Fig. 2. An illustration for Lemma 2 with $m = 1$.

Proof. We fix a control-free execution fragment $\beta = \tau_0 \mathsf{update}(z_1)\tau_1\mathsf{update}(z_2)\ldots\tau_n$ such that at $\beta.\mathsf{fstate}$, $next - now = \Delta_1$. Without loss of generality we assume that at $\beta.\mathsf{fstate}$, $z = z_1$, $loc = l$, and $s = x_1$, where $z_1 \in \mathcal{Z}, l \in \mathcal{L}$ and $x_1 \in I_l$. We have to show that at $\beta.\mathsf{lstate}$, $s \in I_l$.

First, observe that for each $k \in \{0, \ldots, n\}$, $(\tau_k \downarrow s)$ is a solution of the differential equation(s) $d(s) = f_l(s, g(l, x_1))$. Let τ be the pasted trajectory $\tau_0 \frown \tau_1 \frown \ldots \tau_n$[4]. Let $\tau.\mathsf{ltime}$ be T. Since the update action does not change s, $\tau_k.\mathsf{lstate} \lceil s = \tau_{k+1}.\mathsf{fstate} \lceil s$ for each $k \in \{0, \ldots, n-1\}$. As the differential equations are time invariant, $(\tau \downarrow s)$ is a solution of $d(s) = f_l(s, g(l, x_1))$. We define the function $\gamma : [0, T] \to \mathcal{X}$ as $\forall\, t \in [0, T]$, $\gamma(t) \triangleq (\tau \downarrow s)(t)$. We have to show that $\gamma(T) \in I_l$. Suppose, for the sake of contradiction, that there exists $t^* \in [0, T]$, such that $\gamma(t^*) \notin I_l$. By the definition of I_l, there exists i such that $F_{li}(\gamma(0)) \geq 0$ and $F_{li}(\gamma(t^*)) < 0$. We pick one such i and fix it for the remainder of the proof. Since F_{li} and γ are continuous, from intermediate value theorem, we know that there exists a time t_1 before t^* where F_{li} vanishes and that there is some finite time $\epsilon > 0$ after t_1 when F_{li} is strictly negative. Formally, there exists $t_1 \in [0, t^*)$ and $\epsilon > 0$ such that for all $t \in [0, t_1]$, $F_{li}(\gamma(t)) \geq 0$ and $F_{li}(\gamma(t_1)) = 0$ and for all $\delta \in (0, \epsilon]$, $F_{li}(\gamma(t_1 + \delta)) < 0$.

Case 1: $x_1 \in I_l \setminus C_i$. Since $F_{li}(\gamma(t_1)) = 0$, by definition, $\gamma(t_1) \in \partial I_i$. But from the value of $F_{li}(\gamma(t))$ where t is near to t_1, we get that $\frac{\partial F_{li}}{\partial t}(t_1) = \frac{\partial F_{li}}{\partial s}(\gamma(t_1)) \cdot f_l(\gamma(t_1), g(l, x_1)) < 0$. This contradicts condition (a).

Case 2: $x_1 \in C_i$. Since for all $t \in [0, t_1]$, $F_{li}(\gamma(t)) \geq 0$ and $F_{li}(\gamma(t_1)) = 0$, we get that for all $t \in [0, t_1]$, $\gamma(t) \in I_l$ and $\gamma(t_1) \in \partial I_i$. So from condition (b) and (c), we get $c_i \leq \|\gamma(t_1) - x_1\| = \left\| \int_0^{t_1} f_l(\gamma(t), g(l, x_1)) dt \right\| \leq b_i t_1$. That is, $t_1 \geq \frac{c_i}{b_i}$. But we know that $t_1 < t^* \leq T$ and periodicity of Control actions $T \leq \Delta_1 + \Delta_2$. Combining these, we get $\Delta_1 + \Delta_2 > \frac{c_i}{b_i}$ which contradicts condition (d). ■

For PCHAs with certain properties, the following lemma provides sufficient conditions for the existence of the bounds b_j and c_j which satisfy the bounded distance and bounded speed conditions of Lemma 2.

[4] $\tau_1 \frown \tau_2$ is the trajectory obtained by concatenating τ_2 at the end of τ_1.

Lemma 3. *For a given $l \in L$, let $U_l = \{g(l, s) \mid l \in \mathcal{L}, s \in I_l\} \subseteq \mathcal{U}$ and suppose I_l is compact and f_l is continuous in $I_l \times U_l$. The bounded distance and bounded speed conditions (of Lemma 2) are satisfied if $C_j \subset I_l$ satisfies the following conditions: (a) C_j is closed, and (b) $C_j \cap \partial I_j = \emptyset$.*

Theorem 1 combines the above lemmas.

Theorem 1. *Consider a* PCHA *$\mathcal{A}$ and a set $\mathcal{I} \subseteq Q_{\mathcal{A}}$. Suppose $Q_{0\mathcal{A}} \subseteq \mathcal{I}$, $\mathcal{A}$ satisfies control invariance condition of Lemma 1, and conditions (a)-(d) of Lemma 2 for each $l \in \mathcal{L}_{\mathcal{A}}$. Then* $\mathsf{Reach}_{\mathcal{A}} \subseteq \mathcal{I}$.

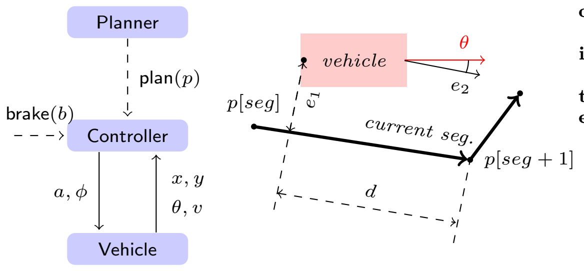
Although the PCHA of Figure 1 has one action of each type, Theorem 1 can be extended for periodically controlled hybrid systems with arbitrary number of input and internal actions. For PCHAs with polynomial vector-fields, given the semi-algebraic sets I_l and C_j, checking condition (a) and finding the c_j and b_j which satisfy conditions (b) and (c) of Lemma 2 can be formulated as a sum-of-squares optimization problem (provided that C_j and $I_l \setminus C_j$ are basic semi-algebraic sets) or proving emptiness of some certain semi-algebraic sets. We are currently exploring the possibility of automatically checking these conditions using SOSTOOLS [14] and QEPCAD [3].

4 System Model

In this section, we describe a subsystem of an autonomous ground vehicle (Alice) consisting of the physical vehicle and the controller (see, Figure 3(a)). Vehicle captures its the position, orientation, and the velocity of the vehicle on the plane. Controller receives information about the state of the vehicle and periodically computes the input steering (ϕ) and the acceleration (a). Controller also receives an infinite[5] sequence of waypoints from a Planner and its objective is to compute a and ϕ such that the vehicle (a) remains within a certain bounded distance e_{max} of the planned path, and (b) makes progress towards successive waypoints at a target speed. Property (a) together with the assumption (possibly guaranteed by Planner) that all planned paths are at least e_{max} distance away from obstacles, imply that the Vehicle does not collide with obstacles. While the Vehicle makes progress towards a certain waypoint, the subsequent waypoints may change owing to the discovery of new obstacles, short-cuts, and changes in the mission plan. Finally, the Controller may receive an externally triggered brake input, to which it must react by slowing the vehicle down.

Vehicle. The Vehicle automaton of Figure 3 specifies the dynamics of the autonomous ground vehicle with acceleration (a) and steering angle (ϕ) as inputs. It has two parameters: (a) $\phi_{max} \in (0, \frac{\pi}{2}]$ is the physical limit on the steering angle, and (b) L is the wheelbase. The main output variables of Vehicle are (a) x and y coordinates of the vehicle with respect to a global coordinate system,

[5] The verification technique can be extended in an obvious way to handle the case where the vehicle has to follow a finite sequence of waypoints and halt at the end.



```
variables                                                        1
output $x$: $\mathbb{R}$:= $x_0$; $y$: $\mathbb{R}$:= $y_0$;
  $\theta$: $\mathbb{R}$:= $\theta_0$; $v$: $\mathbb{R}$:= $v_0$                                    3
input $a$, $\phi$: $\mathbb{R}$
                                                                 5
trajectories
evolve $d(x) = v\cos(\theta)$                                    7
  $d(y) = v\sin(\theta)$
  if $|u.\phi| \leq \phi_{max}$                                  9
    then $d(\theta) = \frac{v}{L}\tan(\phi)$
    else $d(\theta) = \frac{v}{L}\tan(\frac{\phi}{|\phi|}\phi_{max})$ fi    11
  if $v > 0 \vee a \geq 0$
    then $d(v) = a$                                              13
    else $d(v) = 0$ fi
```

Fig. 3. (a) Planner-Controller system. (b) Deviation & disorientation. (c) Vehicle.

(b) orientation θ of the vehicle with respect to the positive direction of the x axis, and (c) vehicle's velocity v. These variables evolve according to the differential equations of lines 7–14. If the input steering angle ϕ is greater than the maximum limit ϕ_{max} then the maximum steering in the correct direction is applied. The acceleration can be negative only if the velocity is positive, and therefore the vehicle cannot move backwards. The controller ensures that the input acceleration is always within such a bound.

Controller. Figure 4 shows the SHIOA specification of the Controller automaton which reads the state of the Vehicle periodically and issues acceleration and steering outputs to achieve the aforementioned goals.

Controller is parameterized by: (a) the sampling period $\Delta \in \mathbb{R}_+$, (b) the target speed $v_T \in \mathbb{R}_{\geq 0}$, (c) proportional control gains $k_1, k_2 > 0$, (d) a constant $\delta > 0$ relating the maximum steering angle and the speed, and (e) maximum and braking accelerations $a_{max} > 0$ and $a_{brake} < 0$. Restricting the maximum steering angle instead of the maximum steering rate is a simplifying but conservative assumption. Given a constant relating the maximum steering rate and the speed, there exists δ as defined above which guarantees that the maximum steering rate requirement is satisfied.

A *path* is an infinite sequence of points $p_1, p_2, \ldots$ where $p_i \in \mathbb{R}^2$, for each i. The main state variables of Controller are the following: (a) *brake* and *new_path* are command variables, (b) *path* is the current path being followed by Controller, (c) *seg* is the index of the last waypoint visited in the current *path*. That is, *path*[*seg* + *1*] is the current waypoint. The straight line segment joining *path*[*seg*] and *path*[*seg* + *1*] is called the *current segment*. (d) *deviation* e_1 is the signed perpendicular distance of the vehicle to the current segment (see, Figure 3(b)). (e) *disorientation* e_2 is the difference between the current orientation of the vehicle (θ) and the angle of the current segment. (f) *waypoint-distance* d is the signed distance of the vehicle to the current waypoint measured parallel to the current segment.

```
signature
  input plan($p$:Seq$[\mathbb{R}^{]}$), brake($b$:$\{On, Off\}$)                2
  internal main
                                                                                4
variables
  input $x, y, \theta, v$: $\mathbb{R}$                                          6
  output $a, \phi$: $\mathbb{R}$ := $(0, 0)$
  internal $brake$: $\{On, Off\}$ := $Off$                                       8
    $path$: Seq$[\mathbb{R}^2]$ := $arbitrary$, $seg$: $\mathbb{N}$ := 1
    $new\_path$: Seq$[\mathbb{R}^2]$ := $path$                                   10
    $e_1, e_2, d : \mathbb{R}$ := $[e_{1,0}, e_{2,0}, d_0]$
    $now$: $\mathbb{R}$ := 0; $next$:$\mathbb{R}_{\geq 0}$ := 0                  12

transitions                                                                     14
  input plan($p$) eff $new\_path$ := $p$
                                                                                16
  input brake($b$) eff $brake$ := $b$
                                                                                18
  internal main
  pre $now = next$                                                              20
  eff $next$ := $now + \Delta$
  if $path \neq new\_path \lor d \leq 0$ then                                   22
    if $path \neq new\_path$
      then $seg$ := 1; $path$ := $new\_path$                                    24
    elseif $d \leq 0$
      then $seg$ := $seg + 1$ fi                                                26
    let $\boldsymbol{p} = \begin{bmatrix} path[seg+1].x - path[seg].x \\ path[seg+1].y - path[seg].y \end{bmatrix}$
      $\boldsymbol{q} = \begin{bmatrix} path[seg+1].y - path[seg].y \\ -(path[seg+1].x - path[seg].x) \end{bmatrix}$   28
      $\boldsymbol{r} = \begin{bmatrix} path[seg+1].x - x \\ path[seg+1].y - y) \end{bmatrix}$
      $e_1$ := $\frac{1}{\|\boldsymbol{q}\|}\boldsymbol{q} \cdot \boldsymbol{r}$;   $e_2$ := $\theta - \angle \boldsymbol{p}$   30
      $d$ := $\frac{1}{\|\boldsymbol{p}\|}\boldsymbol{p} \cdot \boldsymbol{r}$   fi
                                                                                32
  let $\phi_d = -k_1\ e_1 - k_2\ e_2$
  $\phi = \frac{\phi_d}{|\phi_d|} \min(\delta \times v, |\phi_d|)$             34

  if $brake = On$ then $a$ := $a_{brake}$                                      36
  elseif $brake = Off \land v < v_T$
    then $a$ := $a_{max}$ else $a$ := 0 fi                                      38

trajectories                                                                    40
  $d(now) = 1$;  $d(d)$ = -$v \cos(e_2)$
  $d(e_1) = v \sin(e_2)$;  $d(e_2) = \frac{v}{L}\tan(\phi)$                    42
  stop when $now = next$
```

Fig. 4. Controller with parameters $v_T \in \mathbb{R}_{\geq 0}$, $k_1, k_2, \delta, \Delta \in \mathbb{R}_+$ and $a_{brake} < 0$.

The brake(b) action is an externally controlled input action which informs the Controller about the application of an external brake ($b = On$) or the removal of the brake ($b = Off$). When brake(b) occurs, b is recorded in $brake$. The plan(p) action is controlled by the external Planner and it informs the Controller about a newly planned path p. When this action occurs, the path p is recorded in new_path. The main action occurs once every Δ time starting from time 0 and updates $e_1, e_2, d, path, seg, a$ and ϕ as follows: A. if new_path is different from $path$ then seg is set to 1 and $path$ is set to new_path. B. Otherwise, if the waypoint-distance d is less than or equal to 0, then seg is set to $seg+1$ (line 26). For both of the above cases several temporary variables are computed which are in turn used to update e_1, e_2, d as specified in Lines 30-31; otherwise these variables remain unchanged. C. The steering output to the vehicle ϕ is computed using proportional control law and it is restricted to be at most δ times the velocity of the vehicle. This constraint is enforced for the mechanical protection of the steering. The steering output ϕ is set to the minimum of $-k_1e_1 - k_2e_2$ and $v \times \delta$ (line 34). D. The acceleration output a is computed using bang bang control law. If $brake$ is On then a is set to the braking deceleration a_{brake}; otherwise, it executes a_{max} until the vehicle reaches the target speed, at which point a is set to 0.

Along a trajectory, the evolution of the variables are specified by the differential equations on lines 41-43. These differential equations are derived from the update rules described above and the differential equations governing the evolution of x, y, θ and v.

Complete System. Let $\mathcal{A}$ be the composition of the Controller and the Vehicle automata. It can be checked easily that the composed automaton $\mathcal{A}$ is a PCHA. The key variables of $\mathcal{A}$ corresponding to those of PCHA are (a) a continuous variable $\langle x, y, \theta, v, e_1, e_2, d\rangle$ of type $\mathcal{X} = \mathbb{R}^7$, (b) a discrete variable $\langle brake, path, seg\rangle$ of type $\mathcal{L} = \texttt{Tuple}[\{On, Off\}, \texttt{Seq}[\mathbb{R}^2], \mathbb{N}]$, (c) a control variable $\langle a, \phi\rangle$ of type $\mathcal{U} = \mathbb{R}^2$, and (d) two command variables $z_1 \triangleq brake$ of type $\mathcal{Z}_1 = \{On, Off\}$ and $z_2 = path$ of type $\mathcal{Z}_2 = \texttt{Seq}[\mathbb{R}^2]$. For convenience, we define a single derived variable $s \triangleq \langle x, y, \theta, v, e_1, e_2, d\rangle$ encapsulating the continuous variable of $\mathcal{A}$. The input update actions of $\mathcal{A}$ are brake(b) and plan(p). The command variables z_1 and z_2 store the values b and p, respectively, when these actions occur. An internal control action main occurs every Δ time, starting from time 0. That is, values of Δ_1 and Δ_2 as defined in a generic PCHA are $\Delta_1 = \Delta$ and $\Delta_2 = 0$. The control law function g and the state transition function h of $\mathcal{A}$ can be derived from the specification of main action in Figure 4. Let $g = \langle g_a, g_\phi\rangle$ where $g_a : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ and $g_\phi : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ represent the control law for a and ϕ, respectively, and let $h = \langle h_{s,1}, \ldots, h_{s,7}, h_{l,1}, h_{l,2}, h_{l,3}\rangle$ where $h_{s,1}, \ldots, h_{s,7} : \mathcal{L} \times \mathcal{X} \times \mathcal{Z}_1 \times \mathcal{Z}_2 \to \mathbb{R}$ describe the discrete transition of x, y, θ, v, e_1, e_2 and d components of s, and $h_{l,1} : \mathcal{L} \times \mathcal{X} \times \mathcal{Z}_1 \times \mathcal{Z}_2 \to \{On, Off\}$, $h_{l,2} : \mathcal{L} \times \mathcal{X} \times \mathcal{Z}_1 \times \mathcal{Z}_2 \to \texttt{Seq}[\mathbb{R}^2]$ and $h_{l,3} : \mathcal{L} \times \mathcal{X} \times \mathcal{Z}_1 \times \mathcal{Z}_2 \to \mathbb{N}$ describe the discrete transition of *brake*, *path* and *seg*, respectively. The definition of g and h appears in [16]. From the state models of Vehicle and Controller automata specified on line 14 of Figure 3 and lines 42-41 of Figure 4, we see that $\mathcal{A}$ only has one state model. For any value of $l \in \mathcal{L}$, the continuous state s evolves according to the differential equation $\dot{s} = f(s, u)$ where $f = \langle f_1, f_2, \ldots, f_7\rangle$ and $f_1, \ldots, f_7 : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ are associated with the evolution of the x, y, θ, v, e_1, e_2 and d components of s, respectively.

5 Analysis of the System

Overview. The informally stated goals of the system translate to the following:

A. (*safety*) At all reachable states of $\mathcal{A}$, the deviation (e_1) of the vehicle is upper-bounded by e_{max}, where e_{max} is determined in terms of system parameters.
B. (*segment progress*) There exist certain threshold values of deviation, disorientation, and waypoint-distance such that from any state $\mathbf{x}$ with greater deviation, disorientation and waypoint-distance, the vehicle reduces its deviation and disorientation with respect to the current segment, while making progress towards its current waypoint.
C. (*waypoint progress*) The vehicle reaches successive waypoints.

In Sections 5.1 and 5.2, we define a family $\{\mathcal{I}_k\}_{k\in\mathbb{N}}$ of subsets of $Q_{\mathcal{A}}$ and using Lemma 2 and Lemma 3, we conclude that they are invariant with respect to the control-free execution fragments of $\mathcal{A}$. From the specification of main action, we see that the continuous state changes only occurs if $path \neq new\_path$ or waypoint-distance $d \leq 0$. Hence, using Theorem 1, we conclude that any execution fragment starting in $\mathcal{I}_k$ remains within $\mathcal{I}_k$, provided that path and current segment do not change. In Section 5.3, we discuss the proofs for properties (B) and (C) and the derivation of geometric properties of planner paths that can be followed by $\mathcal{A}$ safely. Complete proofs appear in the full version [16].

5.1 Assumptions and Family of Invariants

We define, for each $k \in \mathbb{N}$, the set $\mathcal{I}_k$ which bounds the deviation of the vehicle e_1 to be within $[-\epsilon_k, \epsilon_k]$. This bound on deviation alone, of course, does not give us an inductive invariant. If the deviation is ϵ_k and the vehicle is highly disoriented, then it would violate $\mathcal{I}_k$. Thus, $\mathcal{I}_k$ also bounds the disorientation such that the steering angle computed based on the proportional control law is within $[-\phi_k, \phi_k]$. To prevent the vehicle from not being able to turn at low speed and to guarantee that the execution speed of the controller is fast enough with respect to the speed of the vehicle, $\mathcal{I}_k$ also bounds the speed of the vehicle. $\mathcal{I}_k$ is defined in terms of $\epsilon_k, \phi_k \geq 0$ as $\mathcal{I}_k \triangleq \{\mathbf{x} \in Q \mid \forall i \in \{1, \ldots 6\}, F_{k,i}(\mathbf{x}.s) \geq 0\}$ where $F_{k,1}, \ldots, F_{k,6} : \mathbb{R}^7 \to \mathbb{R}$ are defined as follows:

$$F_{k,1}(s) = \epsilon_k - s.e_1; \quad F_{k,2}(s) = \epsilon_k + s.e_1; \quad F_{k,3}(s) = \phi_k + k_1 s.e_1 + k_2 s.e_2;$$
$$F_{k,4}(s) = \phi_k - k_1 s.e_1 - k_2 s.e_2; \quad F_{k,5}(s) = v_{max} - s.v; \quad F_{k,6}(s) = \delta s.v - \phi_b.$$

Here $v_{max} = v_T + \Delta a_{max}$ and $\phi_b > 0$ is an arbitrary constant. As we shall see shortly, the choice of ϕ_b affects the minimum speed of the vehicle and also the requirements of a brake action. We examine a state $\mathbf{x} \in \mathcal{I}_k$, that is, $F_{k,i}(\mathbf{x}.s) \geq 0$ for any $i \in \{1, \ldots 6\}$. $F_{k,1}(s), F_{k,2}(s) \geq 0$ means $s.e_1 \in [-\epsilon_k, \epsilon_k]$. $F_{k,3}(s), F_{k,4}(s) \geq 0$ means that the steering angle computed based on the proportional control law is in the range $[-\phi_k, \phi_k]$. Further, if $\phi_k \leq \phi_{max}$, then the computed steering satisfies the physical constraint of the vehicle. If, in addition, we have $\phi_b \geq \phi_k$ and $F_{k,6}(s) \geq 0$, then the vehicle actually executes the computed steering command. $F_{k,5}(s) \geq 0$ means that the speed of the vehicle is at most v_{max}.

For each $k \in \mathbb{N}$, we define $\theta_{k,1} = \frac{k_1}{k_2}\epsilon_k - \frac{1}{k_2}\phi_k$ and $\theta_{k,2} = \frac{k_1}{k_2}\epsilon_k + \frac{1}{k_2}\phi_k$, that is, the values of e_2 at which the proportional control law yields the steering angle of ϕ_k and $-\phi_k$ respectively, given that the value of e_1 is $-\epsilon_k$. From the above definitions, we make the following observations about the boundary of the $\mathcal{I}_k$ sets: for any $k \in \mathbb{N}$ and $\mathbf{x} \in \mathcal{I}_k$, $\mathbf{x}.e_2 \in [-\theta_{k,2}, \theta_{k,2}]$, $F_{k,1}(\mathbf{x}.s) = 0$ implies $\mathbf{x}.e_2 \in [-\theta_{k,2}, -\theta_{k,1}]$, $F_{k,2}(\mathbf{x}.s) = 0$ implies $\mathbf{x}.e_2 \in [\theta_{k,1}, \theta_{k,2}]$, $F_{k,3}(\mathbf{x}.s) = 0$ implies $\mathbf{x}.e_2 \in [-\theta_{k,2}, \theta_{k,1}]$, and $F_{k,4}(\mathbf{x}.s) = 0$ implies $\mathbf{x}.e_2 \in [-\theta_{k,1}, \theta_{k,2}]$.

We assume that ϕ_b and all the $\epsilon_k' s$ and ϕ_k's satisfy the following assumptions that are derived from physical and design constraints on the controller. The region in the ϕ_k,ϵ_k plane which satisfies Assumption 1 can be found in [16].

Assumption 1. *(Vehicle and controller design) (a) $\phi_k \leq \phi_b \leq \phi_{max}$ and $\phi_k < \frac{\pi}{2}$ (b) $0 \leq \theta_{k,1} \leq \theta_{k,2} < \frac{\pi}{2}$ (c) $L \cot \phi_k \sin \theta_{k,2} < \frac{k_2}{k_1}$ (d) $\Delta \leq \frac{c}{b}$ where $c = \frac{1}{\sqrt{k_1^2 + k_2^2}}(\phi_k - \tilde{\phi})$, $b = v_{max}\sqrt{\sin^2 \theta_{k,2} + \frac{1}{L^2}\tan^2(\tilde{\phi})}$ and $\tilde{\phi} = \cot^{-1}\left(\frac{k_2}{k_1 L \sin \theta_{k,2}}\right)$.*

If the vehicle is forced to slow down too much at the boundary of an $\mathcal{I}_k$ by the brakes, then it may not be able to turn enough to remain inside $\mathcal{I}_k$. Thus, in verifying the above properties we need to restrict our attention to *good executions* in which brake inputs do not occur at low speeds and are not too persistent. This is formalized by the next definition.

Definition 2. *A* good execution *is an execution α that satisfies: if a* $\mathsf{brake}(On)$ *action occurs at time t then (a) $\alpha(t).v > \frac{\phi_b}{\delta} + \Delta|a_{brake}|$, (b)* $\mathsf{brake}(Off)$ *must occur within time $t + \frac{1}{|a_{brake}|}(\alpha(t).v - \frac{\phi_b}{\delta} - \Delta|a_{brake}|)$.*

For the remainder of this section, we only consider good executions. A state $\mathbf{x} \in Q_{\mathcal{A}}$ is reachable if there exists a good execution α with $\alpha.\mathsf{lstate} = \mathbf{x}$.

5.2 Invariance Property

We fix a $k \in \mathbb{N}$ for the remainder of the section and denote $\mathcal{I}_k, F_{k,i}$ as $\mathcal{I}$ and F_i, respectively, for $i \in \{1, \ldots, 6\}$. As in Lemma 2, we define $I = \{s \in \mathcal{X} \mid F_i(s) \geq 0\}$ and for each $i \in \{1, \ldots, 6\}$, $\partial I_i = \{s \in \mathcal{X} \mid F_i(s) = 0\}$ and let the functions $f_1, f_2, \ldots, f_7 : \mathbb{R}^7 \times \mathbb{R}^2 \to \mathbb{R}$ describe the evolution of x, y, θ, v, e_1, e_2 and d, respectively. We prove that I satisfies the $\mathsf{control}$-free invariance condition of Lemma 1 by applying Lemma 2.

First, we show that all the assumptions in Lemma 2 are satisfied. All the proof appears in the full version [16] which do not involve solving differential equations but require algebraic simplification of the expressions defining the vector fields and the boundaries $\{\partial I_i\}_{i \in \{1,\ldots 6\}}$ of the invariant set.

The next lemma shows that the subtangential, bounded distance and bounded speed conditions (of Lemma 2) are satisfied. The proof for $j = 5$ is presented here as an example. The rest of the proof is provided in [16].

Lemma 4. *For each $l \in \mathcal{L}$ and $j \in \{1, \ldots, 6\}$, the subtangential, bounded distance, and bounded speed conditions (of Lemma 2) are satisfied.*

Proof. Define $C_5 \triangleq \{s \in I \mid s.v \leq v_T\}$. We apply Lemma 3 to prove the bounded distance and the bounded speed conditions. First, note that the projection of I onto the (e_1, e_2, v) space is compact and C_5 is closed. Let $\mathcal{U}_I = \{g(l, s) \mid l \in \mathcal{L},\ s \in I\}$. From the definition of I, it can be easily checked that f is continuous in $I \times \mathcal{U}_I$. In addition, $s.v = v_{max}$ for any $s \in \partial I_5$. Since $a_{max}, \Delta > 0$, $v_{max} = v_T + \Delta a_{max} > v_T$. Therefore, $C_5 \cap \partial I_5 = \emptyset$. Hence, from Lemma 3, the bounded distance and bounded speed conditions are satisfied. To prove the subtangential condition, we pick an arbitrary $s \in \partial I_5$ and $s_0 \in I \setminus C_5$. From the definition of I and C_5, $v_T < s_0.v \leq v_{max}$. Therefore, for any $l \in \mathcal{L}$, either $f_4(s, g(s_0, l)) = 0$ or $f_4(s, g(s_0, l)) = a_{brake}$ and we can conclude that $\frac{\partial F_5}{\partial s} \cdot f(s, g(s_0, l)) = -f_4(s, g(s_0, l)) \geq 0$. ■

From the definition of each C_j, we can derive the lower bound c_j on the distance from C_j to ∂I_j and the upper bound b_j on the length of the vector field f where the control variable u is evaluated when the continuous state $s \in C_j$. Using these bounds and Assumption 1(d), we prove the sampling rate condition.

Lemma 5. *For each $l \in \mathcal{L}$, the sampling rate condition is satisfied.*

Thus, all assumptions in the hypothesis of Lemma 2 are satisfied; from Theorem 1 we obtain that good execution fragments of $\mathcal{A}$ preserve invariance of $\mathcal{I}$, provided that the path and current segment do not change over the fragment.

Theorem 2. *For any* plan*-free execution fragment β starting at a state $\mathbf{x} \in \mathcal{I}$ and ending at $\mathbf{x}' \in Q_{\mathcal{A}}$, if $\mathbf{x}.path = \mathbf{x}.new\_path$ and $\mathbf{x}.seg = \mathbf{x}'.seg$, then $\mathbf{x}' \in \mathcal{I}$.*

5.3 Identifying Safe Planner Paths: An Overview

From invariance of $\mathcal{I}_k$'s, we first show progress from $\mathcal{I}_k$ to $\mathcal{I}_{k+1}$ and then identify a class of planner paths that can be safely followed by $\mathcal{A}$. Owing to limited space, we describe the key steps in this analysis. The complete proofs appear in [16].

From Theorem 2, we know that for each $k \in \mathbb{N}$, $\mathcal{I}_k$ is an invariant of $\mathcal{A}$ with respect to execution fragments in which the *path* and the current segment do not change. First, we show that for each k, starting from any reachable state $\mathbf{x}$ in $\mathcal{I}_k$, any reachable state $\mathbf{x}'$ is in $\mathcal{I}_{k'} \subseteq \mathcal{I}_k$, where $k' = k + n$ and $n = \max(\lfloor \frac{\mathbf{x}.d - \mathbf{x}'.d}{v_{max}\Delta} \rfloor - 1, 0)$. Recall that $\mathcal{I}_k$ and $\mathcal{I}_{k'}$ are defined in terms of the deviation and the disorientation bounds ϵ_k, ϕ_k and $\epsilon_{k'}$, $\phi_{k'}$, respectively. We show that for each k, there exist nonnegative constants a_k and b_k, with $\epsilon_{k+1} = \epsilon_k - a_k$ and $\phi_{k+1} = \phi_k - b_k$, for which the above progress condition is satisfied. Furthermore, for k smaller than the threshold value k^*, we show that a_k and b_k are strictly positive, that is, $I_{k'} \subsetneq I_k$. This essentially establishes property (B), that is, upto a constant threshold, the vehicle makes progress towards reducing the deviation and disorientation with respect to its current waypoint, provided the waypoint distance is large enough. Figure 5 shows a sequence of shrinking $\mathcal{I}_k$'s visited by $\mathcal{A}$ in making progress towards a waypoint.

Next, we derive a sufficient condition on planner paths that can be safely followed with respect to a chosen set $\mathcal{I}_k$ whose parameters $\epsilon_k \in [0, e_{max}]$ and $\phi_k \in [0, \phi_{max}]$ satisfy Assumption 1. The choice of $\mathcal{I}_k$ is made such that it is the smallest invariant set containing the initial state $Q_{0\mathcal{A}}$. The key idea in the condition is: *longer path segments can be succeeded by sharper turns*. The proof is based on an invariant relationship $\mathcal{R}$ amongst the deviation, the disorientation, and waypoint distance. Following a long segment, the vehicle reduces its deviation and disorientation by the time it reaches the end, and thus, it is possible for the vehicle to turn more sharply at the end without breaking an invariance of $\mathcal{I}_k$ and the relationship $\mathcal{R}$.

Assumption 2. *(Planner paths) Let $p_0, p_1, \ldots$ be a planner path; for $i \in \{0, 1, \ldots\}$, let λ_i be the length of the segment $\overline{p_i p_{i+1}}$ and σ_i be the difference in orientation of $\overline{p_i p_{i+1}}$ and that of $\overline{p_{i+1} p_{i+2}}$. Then,*

(a) $\lambda_i \geq 2v_{max}\Delta + \epsilon_k$.
(b) Let $n = k + \lfloor \frac{\lambda_i - \epsilon_k - 2v_{max}\Delta}{v_{max}\Delta} \rfloor$. Then, λ_i and σ_i satisfy the following conditions: (a) $\epsilon_n \leq \frac{1}{|\cos\sigma_i|}(\epsilon_k - v_{max}\Delta|\sin(\sigma_i|)$ and (b) $\phi_n \leq \phi_k - k_1 v_{max}\Delta \sin|\sigma_i| - k_1\epsilon_n(1 - \cos\sigma_i) - k_2|\sigma_i|$ where, given ϵ_k and ϕ_k, ϵ_j and ϕ_j are defined recursively for any $j > k$ by $\epsilon_j = \epsilon_{j-1} - a_{j-1}$ and $\phi_j = \phi_{j-1} - b_{j-1}$.

The relationship between λ and the maximum value of σ which satisfies this assumption is shown in Figure 5. The choice of ϵ_k's and ϕ_k's affects both the requirements on a safe path (Assumption 2) and the definition of good executions. Larger ϵ_k's and ϕ_k's allow sharper turns in planned paths but forces brakes to occur only at higher speeds. This tradeoff is related to the design flaw of Alice as discussed in the introduction of the paper. Without having quantified

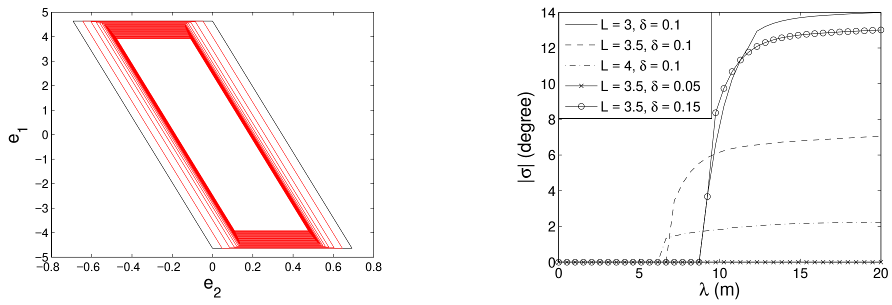


Fig. 5. *Left.* $\mathcal{I}_k$ in black, $\mathcal{I}_{k+i}$ in red for $i > 0$. *Right.* Segment length vs. maximum difference between consecutive segment orientations, for different values of L and δ.

the tradeoff, we inadvertently allowed a path to have sharp turns and also brakes at low speeds—thus violating safety.

Consider a path that satisfies Assumption 2. Further assuming that (a) new planner paths begin at the current position, (b) Vehicle is not too disoriented with respect to new paths, and (c) Vehicle speed is not too high, we establish that $\mathcal{I}_k$ is an invariant of $\mathcal{A}$. Since for any state $\mathbf{x} \in \mathcal{I}_k$, $|\mathbf{x}.e_1| \leq \epsilon_k \leq e_{max}$, invariance of $\mathcal{I}_k$ guarantees the safety property (A). For property (C), we note that for any state $\mathbf{x} \in \mathcal{I}_k$, there exists $v_{min} > 0$ such that $\mathbf{x}.v \geq v_{min} > 0$ and $|\mathbf{x}.e_2| \leq \theta_{k,2} < \frac{\pi}{2}$, that is, $\dot{d} = f_7(\mathbf{x}.s, u) \leq -v_{min} \cos \theta_{k,2} < 0$ for any $u \in \mathcal{U}$. Thus, it follows that the waypoint distance decreases and the vehicle makes progress towards its waypoint.

The simulation results are shown in Figure 6 which illustrate that the vehicle is capable of making a sharp left turn, provided that the path satisfies Assumption 2. In addition, we are able to replicate the stuttering behavior described in the Introduction when Assumption 2 is violated.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
2. N. P. Bhatia and G. P. Szegö. *Dynamical Systems: Stability Theory and Applications*, volume 35 of *Lecture notes in Mathematics*. Springer-Verlag, 1967.
3. C. W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *SIGSAM Bull.*, 37(4):97–108, 2003.
4. J. W. Burdick, N. DuToit, A. Howard, C. Looman, J. Ma, R. M. Murray, and T. Wongpiromsarn. Sensing, navigation and reasoning technologies for the DARPA Urban Challenge. Technical report, DARPA Urban Challenge Final Report, 2007.
5. N. E. DuToit, T. Wongpiromsarn, J. W. Burdick, and R. M. Murray. Situational reasoning for road driving in an urban environment. In *International Workshop on Intelligent Vehicle Control Systems (IVCS)*, 2008.
6. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *STOC*, pages 373–382, 1995.

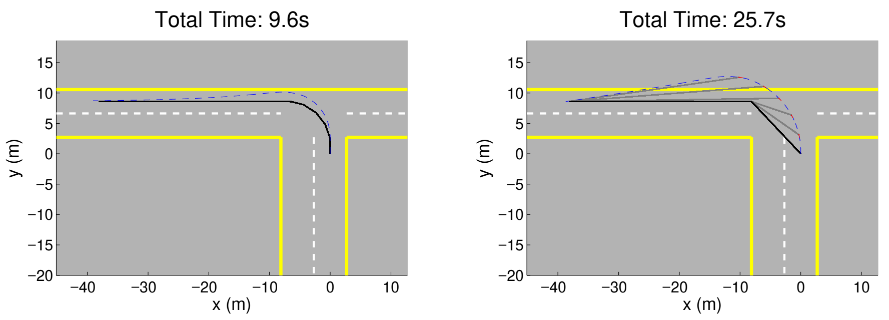


Fig. 6. The positions of the vehicle as it follows a path to execute a sharp left turn. The solid line and the dash line represent, respectively, the path and the positions of the vehicle. The initial path is drawn in black. The positions of the vehicle is plotted in blue except when *brake* is triggered in which case it is plotted in red. *Left.* The path satisfies Assumption 2. *Right.* The path does not satisfy Assumption 2 and the replan occurs due to excessive deviation. The replanned paths are drawn in grey.

7. D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool, 2005.
8. G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *HSCC'99*, pages 137–151. Springer, 1999.
9. N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, August 2003.
10. S. Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.
11. S. Mitra, Y. Wang, N. Lynch, and E. Feron. Safety verification of model helicopter controller using hybrid Input/Output automata. In *HSCC'03*, volume 2623 of *LNCS*, pages 343–358. Springer, 2003.
12. S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas. PVS: Combining specification, proof checking, and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer-Aided Verification, CAV '96*, number 1102, in LNCS, pages 411–414, New Brunswick, NJ, July/August 1996. Springer-Verlag.
13. P. Prabhakar, V. Vladimerou, M. Viswanathan, and G. E. Dullerud. A decidable class of planar linear hybrid systems. In *HSCC'08*, volume 4981 of *LNCS*, pages 401–414. Springer, 2008.
14. S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *CDC'02*, pages 741–746, 2002.
15. V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. E. Dullerud. Stormed hybrid systems. In *ICALP (2)*, *LNCS* 5126, pages 136–147. Springer, 2008.
16. —. Periodically controlled hybird systems: Verifying a controller for an autonomous vehicle. TechReport CaltechCDSTR:2008.003, California Inst. of Tech. Full version: http://resolver.caltech.edu/CaltechCDSTR:2008.003.
17. T. Wongpiromsarn and R. M. Murray. Distributed mission and contingency management for the DARPA urban challenge. In *International Workshop on Intelligent Vehicle Control Systems (IVCS)*, 2008.