Cooperative Collision Avoidance at Intersections: Algorithms and Experiments

Citation: Hafner, Michael R., Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. "Cooperative Collision Avoidance at Intersections: Algorithms and Experiments." IEEE Transactions on Intelligent Transportation Systems 14, no. 3 (September 2013): 1162–1175.

As Published: http://dx.doi.org/10.1109/TITS.2013.2252901

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: http://hdl.handle.net/1721.1/97407

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Cooperative Collision Avoidance at Intersections: Algorithms and Experiments

Michael R. Hafner, Drew Cunningham, Lorenzo Caminiti and Domitilla Del Vecchio

***Abstract*—In this paper, we leverage vehicle-to-vehicle (V2V) communication technology to implement computationally efficient decentralized algorithms for two-vehicle cooperative collision avoidance at intersections. Our algorithms employ formal control theoretic methods to guarantee a collision free (safe) system, while overrides are applied only when necessary to prevent a crash. Model uncertainty and communication delays are explicitly accounted for by the model and by the state estimation algorithm. The main contribution of this work is to provide an experimental validation of our method on two instrumented vehicles engaged in an intersection collision avoidance scenario in a test-track.**

I. Introduction

In the United States, vehicular collisions kill on average 116 and injure 7,900 people per day [22]. In 2009, more than 33,800 people were killed in police-reported motor vehicle traffic crashes and about 2.2 million people were injured [2], with an estimated economic cost of $230 billion. The situation in the European Union is similar, with about 43,000 deaths and 1.8 million people injured per year, for an estimated cost of €160 billion [9]. In 2009, light vehicle crashes accounted for 68% of all U.S. motor vehicle fatalities and, of those light vehicle fatalities, 26% were from side impacts [2], suggesting crashes at intersections or on roadways close to and leading to intersections. These statistics clearly indicate that crashes at intersections have a major impact on the total number of crashes and fatalities in the United States. Furthermore, unlike other high-percentage crashes, such as road departure and rear end, for which radar and camera-based forward collision systems are now available, there is currently no established technology to address side-impact collisions at intersections.

Vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication are setting the basis for establishing this missing technology by having vehicles cooperate with each other and with the surrounding infrastructure, sharing information about the environment, and improving overall situational awareness. Therefore, intelligent transportation systems (ITS) for inter-vehicle cooperative (active) safety have been subject of intense research world-wide in government and industry consortia, such as the Crash Avoidance Metrics Partnership (CAMP) and Vehicle Infrastructure Integration Consortium (VIIC) in the U.S., the Car2Car Communications Consortium in Europe, and the Advanced Safety Vehicle project 3 (ASV3) in Japan.

Since cooperative active safety systems are life-critical, *ad hoc* algorithms for preventing collisions are not acceptable. Instead, there is a compelling need for employing methodologies that provide formal safety guarantees, such as found in the control theory and computer science literature [18, 24, 26]. Specifically, the collision avoidance problem can be addressed by computing the set of states, called backward reachable set or capture set, that lead to an unsafe configuration (a collision) independently of the input choice [26]. Then, a feedback map is computed that restricts the control inputs when necessary to prevent entrance in the capture set. While this approach is theoretically appealing because it ensures safety by construction and applies overrides only when necessary, its practical applicability is often limited by the complexity associated with the computation of the capture set [15, 27]. Researchers have been tackling computational issues by, among other approaches, focusing on restricted classes of systems [3, 11, 13, 14].

In this work, we employ the techniques of [14], which lead to linear complexity algorithms that are implementable in real-time applications. Furthermore, the results of [14], as opposed to the others, guarantee safety in the presence of imperfect state information, due, for example, to sensor noise or communication delays, and only need a coarse model of the vehicle dynamics. We focus on a two-vehicle collision avoidance scenario at intersections and develop a decentralized control algorithm that uses V2V communication to determine whether automatic control is needed to prevent a collision. We prevent a collision through automatic control by actuating only brake and throttle, but not steering, and assuming drivers follow nominal paths as established by the driving lanes. In our intersection collision avoidance (ICA) application, the drivers retain full control of the vehicle until the system configuration hits the capture set. At this point, a control action is necessary to prevent a collision, and automatic throttle or brake are applied to both vehicles in a coordinated fashion so that one vehicle enters the intersection only after the other has exited. After the crash has been prevented, the driver regains control of brake and throttle. We report on the implementation of our algorithms on two instrumented Lexus IS 250 test vehicles engaged in a collision avoidance scenario at a test intersection at the Toyota Technical Center of Ann Arbor, MI.

Related Work. The employment of formal methods in intelligent transportation has been previously applied by the California PATH project in the 90s. The objective of the automated highway systems (AHS) project was to deploy fully autonomous highway systems incorporating vehicle platoons to increase traffic throughput, safety, and fuel efficiency [4]. More recently, work that employs job scheduling techniques [8, 17] and optimal control [19] for intersection collision avoidance has appeared. Collision warning algorithms have also been proposed for general traffic scenarios [7, 28] and

for intersections [6, 12]. Although different in scope, related to our work is also research on collision mitigation through emergency braking [16]. Directly related to this paper are experimental works on full scale vehicle test-beds focusing on collision avoidance/warning at intersections, which leverage V2V communication [20, 21]. Specifically, in [20] a fuzzy controller to manage vehicles crossing an intersection is proposed. In [21], an on-board vehicle hazard detection that uses V2V is developed to warn the driver about dangerous situations. In these papers, formal safety guarantees are not provided and cooperation between vehicles is not leveraged to provide least restrictive warnings/overrides. Here, we bridge the gap between formal methods and cooperative collision avoidance systems at intersections by developing/testing an experimental cooperative collision avoidance system based on formal control theoretic techniques.

II. Problem Overview

We consider the intersection scenario depicted in Figure 1(a), in which two vehicles approach an intersection and can potentially collide in the indicated red shaded area. A collision may occur for a number of reasons, including a distracted driver not seeing the incoming vehicle, under-estimating the vehicle speed, and violating red lights or stop signs. We seek to design controllers on board of each vehicle that use V2V communication in order to negotiate the intersection and apply automatic control only when it is absolutely necessary to prevent a collision.

We assume that, after making high level route decisions, drivers follow predefined (known) paths as established by driving lanes. Under this assumption, the methodology that we propose can be applied to any paths geometry at an intersection. Here, we consider the specific intersection scenario of Figure 1(a) to be consistent with the geometry of the test intersection employed in the experiments (Figure 1(d)). Collisions between two vehicles are prevented by only controlling the longitudinal velocity and displacement of each vehicle along its path, never controlling vehicle steering. We assume each vehicle is equipped with sensors for state measurement (absolute position, heading, velocity, acceleration, brake torque, and pedal position), V2V communication, and the ability to automatically actuate the throttle and brake. We assume our collision avoidance system is active well before the vehicles approach the intersection, preventing initial vehicle configurations generating unavoidable collision. Under the above assumptions, the safety algorithms that we illustrate here guarantee that the vehicles will never collide.

A. Test vehicles and test track

The test vehicles used in this work are modified Lexus IS 250 (2007) test vehicles (Figure 1(c)). The modifications include: computer running a Linux operating system; Differential Global Positioning System (DGPS) for position, absolute time and heading measurement; Denso Wireless Safety Unit (WSU) capable of V2V and Vehicle-to-Infrastructure (V2I) Dedicated Short-Range Communications (DSRC); connection to the Controller-Area Network (CAN) bus to read information

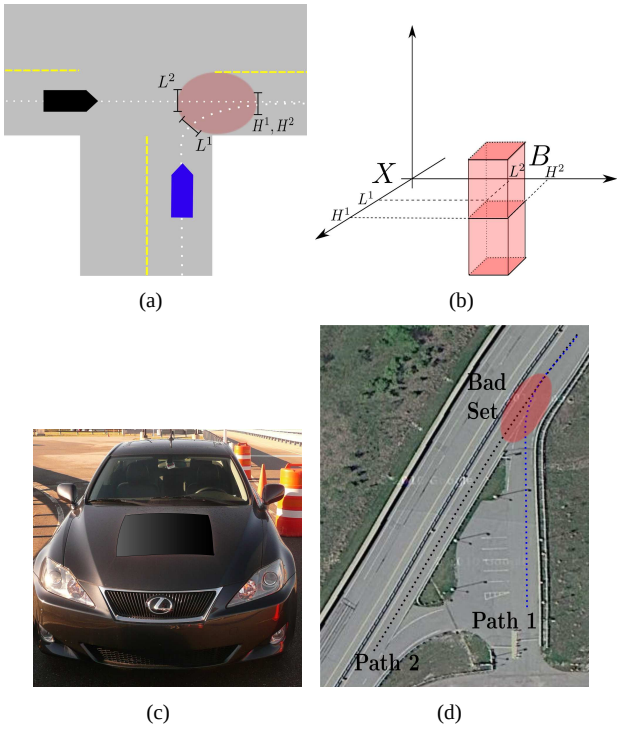


Fig. 1. (a) Intersection collision avoidance scenario with the red area denoting the bad (collision) set. Vehicle displacement is considered along the path. L^i determines the lower limit of the bad set along vehicle i path, while U^i determines the upper limit of the bad set along vehicle i path. (b) Bad set in the state space X: it is the interval $]L^1, H^1[\times]L^2, H^2[$ in the X_1 (displacement) space for every value of the speeds (vertical axis) of the two vehicles. (c) Modified Lexus IS 250 vehicles used in the experiments. (d) Top-down view of the test-track where the experiments were performed.

from vehicle sensors (velocity, acceleration, brake pedal position, transmission state, etc.); CAN bus interface with brake and throttle actuators.

The computer system is affixed inside the wheel well. The purpose of this system is to interface with all on-board vehicle sensors and actuators, in a manner that allows for rapid development, deployment and testing of software applications. The computer runs an Ubuntu Linux distribution, and consists of a Intel Core-Duo 2.0 GHz processor, 1 GB RAM, 150 GB hard drive, and a motherboard with on-board ethernet and USB ports. A USB video card is connected to the vehicle navigation display unit, and a wireless keyboard is used to control the computer from the passenger seat. The computer can read and write to the CAN bus via a USB adapter. To communicate between vehicles and interface with a DGPS unit, a Denso Wireless Safety Unit (WSU) is connected via ethernet, which is an after-market industry standard (planned) in communication and control for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) safety systems [23].

The on-board DGPS unit is capable of 0.45 m accuracy for absolute position, 1.5^o accuracy for absolute heading, and 0.1 s accuracy for absolute time. The measurement update rate is 10 Hz. Other sensors include: (i) accelerometer, based on MEMS technology, capable of 0.5 m/s^2 accuracy; (ii) speedometer, measuring average speed at the wheel, capable of 0.5 m/s accuracy; (iii) throttle pedal measurement, capable of 0.5 % accuracy; (iv) brake torque applied at wheel, capable

of 0.5 Nm accuracy. The vehicle brake controller is modified to accept brake commands from the computer via CAN bus messages. The drive-by-wire (sends ECU electric signals over CAN bus) throttle pedal, is modified to allow computer issued commands via CAN bus messages to create throttle pedal signals to the ECU. Communication is carried out by the Denso WSU unit. The message standard is the Dedicated Short-Range Communication (DSRC), which is broadcast at the 5.9 GHz band, which is dedicated to V2V and V2I communication. The WSU is connected to a top mounted antenna (Figure 1(a)). Communication is carried out with a broadcast network topology, that is, messages transmitted by a sender can be received by any listener in-range.

III. Solution Approach

The general solution approach is based on formally encoding the requirement of no-collision into a bad set of vehicle speed and position configurations to be avoided. Then, based on the vehicles dynamical model, we calculate the capture set, which is the set of all vehicle configurations that enter the bad set independently of any throttle/brake control action. Once the capture set is computed, we determine a throttle/brake control map for both vehicles that keeps the system state outside of the capture set at all times. This control map applies throttle and brake inputs only when the system configuration hits the boundary of the capture set. Otherwise, no control action is applied and the driver has full control of the vehicle.

The computations of the capture set and of the control map are usually very demanding, require an exact description of the system dynamics, and assume perfect information on the state of the system. In this section, we illustrate the approach to compute the capture set and the control map developed in [14], which exploits the specific structure of the application domain to overcome these limitations. Specifically it provides efficient algorithms, allows a coarser model obtained from suitable experiments, and is robust to imperfect state information due to sensor uncertainty and especially to communication delays.

A. System model and safety specification

We model each vehicle as a system Σ^i for $i \in \{1,2\}$, describing the longitudinal dynamics of vehicle i along its path. Each system Σ^i is an input-output system, defined by the tuple $\Sigma^i := \{X^i, \mathcal{O}^i, \mathcal{U}^i, \mathcal{D}^i, f^i, h^i\}$, where $X^i \subset \mathbb{R}^2$ is the state space describing position and speed, $\mathcal{O}^i \subset \mathbb{R}^m$ is the output measurement space, $\mathcal{U}^i := [u_L^i, u_H^i] \subset [0,1] \times [0,1]$ is the control input space representing the percentage the brake and throttle pedal are depressed, $\mathcal{D}^i := [d_L^i, d_H^i] \subset \mathbb{R}^m$ is the disturbance input space, which can be employed to account for unmodeled dynamics, $f^i : X^i \times \mathcal{U}^i \times \mathcal{D}^i \to X^i$ is the vector field modeling the dynamics of the vehicle, and $h^i : \mathcal{O}^i \rightrightarrows X^i$ is the output set-valued map that provides the set of states compatible with an output measurement. We let $x_1^i \in X_1^i$ denote the longitudinal displacement of vehicle i along its fixed path and x_2^i denote the longitudinal speed of vehicle i along its path. We denote the continuous flow of system Σ^i as $\phi^i(t, x^i, \mathbf{u}^i, \mathbf{d}^i)$, where t denotes the time, x^i denotes the initial state, $\mathbf{u}^i$ denotes the control input signal and $\mathbf{d}^i$ denotes the disturbance signal. In this paper, we will denote in bold signals, which are functions of time.

The two-vehicle system is modeled as the parallel composition of the two systems, denoted as $\Sigma = \Sigma^1 || \Sigma^2 = \{X, \mathcal{O}, \mathcal{U}, \mathcal{D}, f, h\}$, in which $X = X^1 \times X^2$, $\mathcal{O} = \mathcal{O}^1 \times \mathcal{O}^2$, $\mathcal{U} = \mathcal{U}^1 \times \mathcal{U}^2$, $\mathcal{D} = \mathcal{D}^1 \times \mathcal{D}^2$, $f = (f^1, f^2)$, and $h = (h^1, h^2)$. Accordingly, we will let $x = (x^1, x^2)$, $u = (u^1, u^2)$, and $d = (d^1, d^2)$. Furthermore, we let $x_1 = (x_1^1, x_1^2) \in X_1$ denote the pair of two-vehicle displacements. The safety specification for Σ is described in terms of a subset of the state space that needs to be avoided to prevent a collision. Specifically, we call such a set the *bad set* $\mathbf{B} \subset X$ and we will say that the system is safe if the flow never enters the bad set $\mathbf{B}$. For some initial state x_o, the system is safe if there exists some control input signal $\mathbf{u}$ such that for all disturbance input signals $\mathbf{d}$ and time t, we have that $\phi(t, x_o, \mathbf{u}, \mathbf{d}) \notin \mathbf{B}$.

From the construction of the state space and the fact that a collision between two vehicles results when they are both in the red shaded area of Figure 1 (a), $\mathbf{B} \subseteq X$ can be defined as

$$\mathbf{B} := \{x \in X \mid (x_1^1, x_1^2) \in]L^1, H^1[\times]L^2, H^2[\}, \qquad (1)$$

where $L^i < H^i$ for $i \in \{1,2\}$ (see Figure 1 (a)-(b)). We also denote $L = (L^1, L^2)$ and $H = (H^1, H^2)$.

The safe controller is based on computing a subset of the state space, called the *capture set*, denoted $\mathcal{C} \subseteq X$. The capture set is the set of all initial conditions, such that no control input can prevent a collision. The mathematical definition is given by

$$\mathcal{C} := \{x \in X \mid \forall\ \mathbf{u},\ \exists\ t,\ \exists\ \mathbf{d} \text{ s.t. } \phi(t, x, \mathbf{u}, \mathbf{d}) \in \mathbf{B}\}. \qquad (2)$$

The approach of our solution to the safety control problem is to compute the capture set, and through the application of feedback control, prevent the flow from ever entering the capture set. By the definition of the capture set, safety is guaranteed if the flow never enters the capture set.

Computing the capture set is in general a difficult problem. In the next sections, we show how exploiting the structural features of the specific system under study allows us to compute this set and handle imperfect state information.

B. Computation approach exploiting partial orders

In this section, we illustrate the main result of [14] to compute the capture set. This approach relies on (i) the state and input spaces of the system Σ^i being partially ordered and (ii) the flow of the system Σ^i being an order preserving map. Specifically, for the state space $X^i \subseteq \mathbb{R}^2$, we consider elements to be partially ordered according to component-wise ordering , that is, for $z^i, w^i \in X^i$ we have that $z^i \leq w^i$ provided $z_1^i \leq w_1^i$ and $z_2^i \leq w_2^i$. Further, we consider the partial ordering between input signals defined for signals $\mathbf{u}^i, \mathbf{v}^i$ as $\mathbf{u}^i \leq \mathbf{v}^i \Leftrightarrow \mathbf{u}^i(t) \leq \mathbf{v}^i(t)$ for all t. The inequality $\mathbf{u}^i(t) \leq \mathbf{v}^i(t)$ is defined such that $\mathbf{u}_1^i(t) \geq \mathbf{v}_1^i(t)$ and $\mathbf{u}_2^i(t) \leq \mathbf{v}_2^i(t)$. We assume that the flow of each system Σ^i is an *order preserving* map. Mathematically, this means that for initial conditions $z^i, w^i \in X^i$, inputs $\mathbf{u}^i, \mathbf{v}^i$ and disturbances

$\mathbf{d}^i, \mathbf{b}^i$, the following implication holds

$$z^i \leq w^i \ \wedge \ \mathbf{u}^i \leq \mathbf{v}^i \ \wedge \ \mathbf{d}^i \leq \mathbf{b}^i \Rightarrow \\ \phi^i(t, z^i, \mathbf{u}^i, \mathbf{d}^i) \leq \phi^i(t, w^i, \mathbf{v}^i, \mathbf{b}^i) \ \ \forall \ t. \tag{3}$$

In terms of the vehicle dynamics, this assumption implies that greater initial displacement, greater initial velocity, and greater inputs will lead to greater displacements and speeds at any time. The validity of this assumption for the vehicle dynamics is discussed in detail in Section IV, where the vehicle model is introduced. A liveliness condition is introduced by requiring that for at least one i $f_1^i(x^i, u^i, d^i) > 0$ for all x^i, u^i and d^i. From a practical point of view, this requires that vehicle i does not go in reverse and does not stop.

The order preserving property of the dynamics along with the structure of the bad set can be exploited to compute the capture set for system $\Sigma = \Sigma^1 || \Sigma^2$ with an algorithm that has linear complexity with respect to the state dimension. The algorithm is based on the *restricted* capture set, which for a fixed input signal $\mathbf{u}$, is defined as $\mathcal{C}_{\mathbf{u}} := \{x \in X \mid \exists \ t \geq 0, \ \exists \ \mathbf{d} \ \text{ s.t. } \phi(t, x, \mathbf{u}, \mathbf{d}) \in \mathbf{B}\}$. This set represents the set of initial conditions that are taken into the bad set under the *fixed* input signal $\mathbf{u}$. Define the fixed input signals $\mathbf{u}_{\mathcal{L}}, \mathbf{u}_{\mathcal{H}}$, as $\mathbf{u}_{\mathcal{L}}(t) := (u_H^1, u_L^2)$ and $\mathbf{u}_{\mathcal{H}}(t) := (u_L^1, u_H^2)$ for all t. Then, we have ([14])

$$\mathcal{C} = \mathcal{C}_{\mathbf{u}_{\mathcal{L}}} \cap \mathcal{C}_{\mathbf{u}_{\mathcal{H}}}. \tag{4}$$

The capture set can be computed by only computing the two restricted capture sets corresponding to maximum and minimum inputs. The restricted capture sets are simpler to compute, since they can be obtained by just integrating the dynamics under fixed control inputs. This is in contrast with the capture set $\mathcal{C}$, whose computation requires the solution of a differential game between the control and the disturbance.

Based on the expression of the capture set given in (4), the feedback control map is given by

$$g(x) := \begin{cases} (u_H^1, u_L^2) & \text{if } x \in \mathcal{C}_{\mathbf{u}_{\mathcal{L}}} \text{ and } x \in \partial \mathcal{C}_{\mathbf{u}_{\mathcal{H}}}, \\ (u_L^1, u_H^2) & \text{if } x \in \partial \mathcal{C}_{\mathbf{u}_{\mathcal{L}}} \text{ and } x \in \overline{\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}}, \\ \mathcal{U} & \text{otherwise,} \end{cases} \tag{5}$$

in which $\overline{\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}}$ denotes the closure of $\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}$. The controller allows the driver to chose any input until the flow hits the boundary of the capture set. The driver retains control once the flow no longer touches the boundary of the capture set. A visual interpretation of the feedback map is provided in Figure 2.

In the presence of communication delays and/or uncertain sensor readings the vehicles will not have access to the exact value of the system state but to a set of possible current system states. This can be easily incorporated in the above described control strategy [14]. Let the set of possible current system states be denoted $\hat{x} \subset X$, which can be constructed using output measurement $z \in \mathcal{O}$ as explained in Section V-A. The safety specification is now posed in terms of preventing the state uncertainty $\hat{x}$ from intersecting the bad set $\mathbf{B}$. That is, the system is safe if $\hat{x}(t) \cap \mathbf{B} = \emptyset$ for all $t \in \mathbb{R}_+$. It has been shown that this is the case if and only if $\hat{x}(t)$ never intersects both $\mathcal{C}_{\mathbf{u}_{\mathcal{L}}}$ and $\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}$ at the same time [14]. The feedback set-valued map g, as defined in (5) can still guarantee this as long as it is extended to set $\hat{x}$ as follows

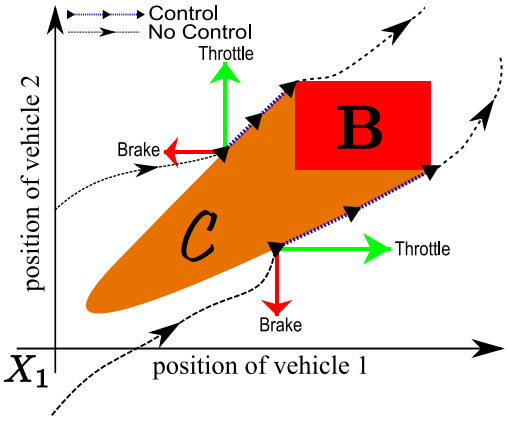


Fig. 2. Feedback map $g(x)$ shown for two separate trajectories. The orange region represents a slice of the capture set in position space corresponding to a pair of vehicles speeds. When the flow touches the upper boundary of the capture set, geometrically as $x \in \mathcal{C}_{\mathbf{u}_{\mathcal{L}}}$ and $x \in \partial \mathcal{C}_{\mathbf{u}_{\mathcal{H}}}$, the feedback controller commands the input (u_L^1, u_H^2), corresponding to vehicle 1 applying maximum brake while vehicle 2 applies maximum throttle. When the flow touches the lower boundary of the capture set, geometrically as $x \in \overline{\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}}$ and $x \in \partial \mathcal{C}_{\mathbf{u}_{\mathcal{L}}}$, the feedback controller commands the input (u_H^1, u_L^2), corresponding to vehicle 1 applying maximum throttle while vehicle 2 applies maximum brake.

$$g(\hat{x}) := \begin{cases} (u_H^1, u_L^2) & \text{if } \hat{x} \cap \mathcal{C}_{\mathbf{u}_{\mathcal{H}}} \neq \emptyset \text{ and} \\ & \hat{x} \cap \partial \mathcal{C}_{\mathbf{u}_{\mathcal{L}}} \neq \emptyset \text{ and} \\ & \hat{x} \cap \mathcal{C}_{\mathbf{u}_{\mathcal{L}}} = \emptyset, \\ (u_L^1, u_H^2) & \text{if } \hat{x} \cap \overline{\mathcal{C}_{\mathbf{u}_{\mathcal{L}}}} \neq \emptyset \text{ and} \\ & \hat{x} \cap \partial \mathcal{C}_{\mathbf{u}_{\mathcal{H}}} \neq \emptyset \text{ and} \\ & \hat{x} \cap \mathcal{C}_{\mathbf{u}_{\mathcal{H}}} = \emptyset, \\ \mathcal{U} & \text{otherwise.} \end{cases} \tag{6}$$

If the set of admissible control inputs evaluated by $g(\hat{x})$ is $\mathcal{U}$, the driver is free to apply any input. The interpretation of this feedback set-valued map is that control is applied when the state uncertainty has non-empty intersection with either $\mathcal{C}_{\mathbf{u}_{\mathcal{L}}}$ or $\mathcal{C}_{\mathbf{u}_{\mathcal{H}}}$, and simultaneously is touching the boundary of the other. We remark that by construction, feedback map g is order reversing with respect to partial order established by set inclusion, that is, $A \subset B \ \Rightarrow g(A) \supset g(B)$. This property implies that the larger the state uncertainty, the more conservative the controller will be.

C. Algorithmic Implementation

In this section, we provide a summary of the algorithms that compute the restricted capture set for the case in which the first component of the vector fields f^i do not depend on the x_1^i coordinate (displacement) [14]. This assumption is satisfied by the vehicle dynamics considered in the next section. The algorithms are implemented on-board the vehicle computer, therefore they must use a discrete-time model of the dynamics. For $n > 0$ and step size $\Delta T > 0$, the discrete-time flow of system Σ is given by $\Phi(n, x, \mathbf{u}, \mathbf{d})$ and is generated by the forward Euler approximation of the continuous time dynamics, mathematically given by $\Phi(n+1, x, \mathbf{u}, \mathbf{d}) = \Phi(n, x, \mathbf{u}, \mathbf{d}) + \Delta T f(\Phi(n, x, \mathbf{u}, \mathbf{d}), \mathbf{u}[n-1], \mathbf{d}[n-1])$, with initial condition

$\Phi(0, x, \mathbf{u}, \mathbf{d}) = x$, and sampled signals $\mathbf{u}[n] := \mathbf{u}(n\Delta T)$ and $\mathbf{d}[n] := \mathbf{d}(n\Delta T)$.

The feedback map g is implemented in discrete time, which requires an alternate definition of the capture set boundary. We will say that the set $\hat{x}[n] \subset X$ intersects the boundary and not the interior of the restricted capture set $\mathcal{C}_{\mathbf{u}}$ provided $\hat{x}[n] \cap \mathcal{C}_{\mathbf{u}} = \emptyset$ and $\hat{x}[n+1] \cap \mathcal{C}_{\mathbf{u}} \neq \emptyset$. This states that $\hat{x}[n]$ intersects the boundary and not the interior of the restricted capture set if it is currently outside of the set, but it will be inside the set at the next time step.

To compute the capture set $\mathcal{C}_{\mathbf{u}}$, we can compute a *slice* of it in the displacement space, denoted $\mathscr{C}_{\mathbf{u}} \subset X_1$, corresponding to the current two-vehicle velocity (x_2^1, x_2^2). Due to the order preserving properties of the dynamics with respect to state and input, and the structure of the bad set $\mathbf{B}$, the restricted capture set slice is computed through the back propagation of the upper and lower bounds of the bad set, i.e., $L, H \in X_1$. Specifically, define the sequences

$$\begin{aligned} L(n, x, u) &:= L + x_1 - \Phi_1(n, x, \mathbf{u}, \mathbf{d}_H), \\ H(n, x, u) &:= H + x_1 - \Phi_1(n, x, \mathbf{u}, \mathbf{d}_L), \end{aligned} \tag{7}$$

where $\mathbf{d}_L(k) := (d_L^1, d_L^2)$ and $\mathbf{d}_H(k) := (d_H^1, d_H^2)$ for all k. Given current state estimate set $\hat{x}$, the restricted capture set slice $\mathscr{C}_{\mathbf{u}}$ can be written as (Algorithm 1)

$$\mathscr{C}_{\mathbf{u}} = \bigcup_{k \in \mathbb{N}}]L(n, \sup \hat{x}, \mathbf{u}), H(n, \inf \hat{x}, \mathbf{u})[.$$

Algorithm 1 $\mathscr{C}_{\mathbf{u}}$ = CaptureSetSlice$(\hat{x}, \mathbf{u})$

Input: $(\hat{x}, \mathbf{u}) \in 2^X \times S(\mathcal{U})$

$n = 1$
loop
  if $\inf \hat{x}_1 \leq H(n, \inf \hat{x}, \mathbf{u})$ and $\inf \hat{x}_1 \notin]L(n, \sup \hat{x}, \mathbf{u}), H(n, \inf \hat{x}, \mathbf{u})[$ **then**
    $n = n + 1$
  else
    return $\mathscr{C}_{\mathbf{u}} = \bigcup_{k \leq n}]L(k, \sup \hat{x}, \mathbf{u}), H(k, \inf \hat{x}, \mathbf{u})[$.
  end if
end loop

Output: $\mathscr{C}_{\mathbf{u}} \subset X_1$.

We can determine non-empty intersection of the capture set with the state uncertainty by using the equivalence $\hat{x}_1 \cap \mathscr{C}_{\mathbf{u}} = \emptyset \Leftrightarrow \hat{x} \cap \mathcal{C}_{\mathbf{u}} = \emptyset$. The closed-loop implementation of the feedback map (6), in discrete time, is provided in Algorithm 2, where $\mathbf{u} =$ FeedbackMap$(\hat{x}[n+1], \hat{x}[n])$.

Note that for evaluating the control map, we only need to calculate the sequences $L(n, x, u)$ and $H(n, x, u)$ for two extremal constant inputs $u_{\mathcal{L}} = (u_H^1, u_L^2)$ and $u_{\mathcal{H}} = (u_L^1, u_H^2)$. Hence, we do not require the detailed model of the system Σ, we just need to know how the system responds to these two extremal constant inputs. As we will see in Section IV, this can be achieved through a series of experiments where these constant inputs are applied for a set of different initial speeds.

Algorithm 2 $u =$ FeedbackMap$(\hat{x}[n+1], \hat{x}[n])$

Input: $(\hat{x}[n+1], \hat{x}[n]) \in 2^X \times 2^X$

Construct capture set slices for state prediction.
$\mathscr{C}_{\mathbf{u}_{\mathcal{L}}} =$ CaptureSetSlice$(\hat{x}[n+1], \mathbf{u}_{\mathcal{L}})$, $\mathscr{C}_{\mathbf{u}_{\mathcal{H}}} =$ CaptureSetSlice$(\hat{x}[n+1], \mathbf{u}_{\mathcal{H}})$

Check if predicted state $\hat{x}[n+1]$ intersects both capture set slices.
if $\hat{x}[n+1] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{L}}} \neq \emptyset$ *and* $\hat{x}[n+1] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{H}}} \neq \emptyset$ **then**

  Construct capture set slices for current state.
  $\mathscr{C}_{\mathbf{u}_{\mathcal{L}}} =$ CaptureSetSlice$(\hat{x}[n], \mathbf{u}_{\mathcal{L}})$, $\mathscr{C}_{\mathbf{u}_{\mathcal{H}}} =$ CaptureSetSlice$(\hat{x}[n], \mathbf{u}_{\mathcal{H}})$

  Determine control according to equation (6).
  if $\hat{x}_1[n] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{L}}} = \emptyset$ *and* $\hat{x}_1[n] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{H}}} \neq \emptyset$ **then**
    $u = u_{\mathcal{L}}$
  else if $\hat{x}_1[n] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{L}}} \neq \emptyset$ *and* $\hat{x}_1[n] \cap \mathscr{C}_{\mathbf{u}_{\mathcal{H}}} = \emptyset$ **then**
    $u = u_{\mathcal{H}}$
  else
    $u = u_{\mathcal{L}}$
  end if

else
  No control specified.
  $u \in \mathcal{U}$
end if

Output: $u \in \mathcal{U}$.

IV. Vehicle Dynamics

The vehicle dynamics, which take throttle and brake as inputs and provide longitudinal displacement as output, is the cascade of the powertrain system and the vehicle model (Figure 3(a)). The powertrain system (Figure 3(b)) generates the wheel torque inputs in response to throttle and brake inputs. The vehicle model takes throttle and brake inputs and produces longitudinal displacement as output according to Newton's law. In this section, we describe each of the two subsystems and illustrate how the cascade of the two generates a flow that is an order preserving map when throttle inputs do not change with time. Then, we perform a system identification procedure to determine the dynamics of the cascade system only in response to maximal throttle and maximal braking, which is sufficient for the implementation of the control map as described in Section III.

A. Vehicle Model

The longitudinal displacement of the vehicle along its path is denoted by p and the longitudinal velocity is denoted by $v \in [v_{min}, v_{max}]$, where $v_{min} \geq 0$. The controlled forces that act on the vehicle are the brake input $f_b \in \mathcal{F}_b = [f_{min}, 0]$ with $f_{min} < 0$ and engine input $f_e \in \mathcal{F}_e = [0, f_{max}]$ with $f_{max} > 0$. The brake force f_b is controlled by the driver via

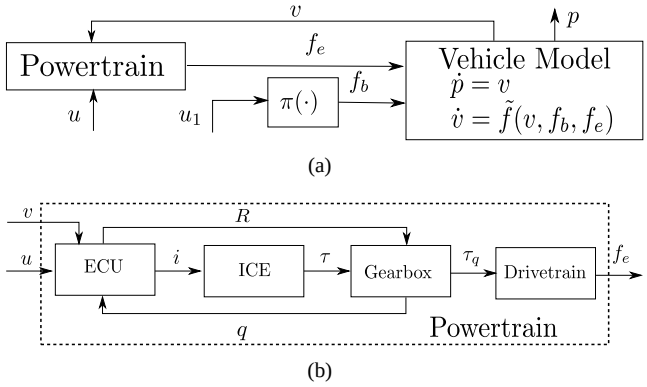


Fig. 3. (a) Block diagram representing the cascade of the powertrain model and the vehicle model. Here, p denotes longitudinal displacement and v denotes longitudinal speed. The powertrain model (b) takes the inputs u and velocity v to produce engine torque at the wheel f_e. The static map π takes the brake pedal percentage input u_1 to produce brake torque f_b. The vehicle model takes the brake force f_b and engine force f_e as inputs. (b) Powertrain system. The Engine Control Unit (ECU) is a means of controlling the fuel injection rate and the gear state q of the transmission. The output signals of the ECU are the fuel injection rate i and the gear reset R. The second block is the Internal Combustion Engine (ICE), which is where the fuel combustion takes place based on the fuel injection rate i, and produces an output torque τ at the flywheel. The next block is the transmission, which converts torque at the flywheel τ to torque at the transmission output τ_q as a function of the gear state q. The drivetrain is the last block, which transfers torque from the gearbox τ_q to force at the wheel f_e.

the surjective-monotone map $\pi : \mathcal{U}_1 \to \mathcal{F}_b$ that takes brake pedal percentage u_1 as an input, while the engine force f_e is supplied by the powertrain (Figure 3(a)). The longitudinal dynamics are given by

$$\frac{dv}{dt} = \frac{\mathcal{R}^2}{J_w + \mathcal{M}\mathcal{R}^2}(f_e + f_b - \frac{\rho_{air}}{2} C_D A_f v^2 - C_{rr}\mathcal{M}g) =: \tilde{f}(v, f_b, f_e), \tag{8}$$

where $\mathcal{R}$ is the wheel radius, $\mathcal{M}$ is the vehicle mass, ρ_{air} is the air density, C_D is the air drag coefficient, A_f is the projected vehicle cross section, and C_{rr} is the coefficient of rolling friction [29].

The longitudinal dynamics (8) generate a flow $(p(t, p_o, v_o, \mathbf{f_b}, \mathbf{f_e}), v(t, v_o, \mathbf{f_b}, \mathbf{f_e}))$ that is an order preserving map with respect to brake force input signal $\mathbf{f}_b$, engine force signal $\mathbf{f}_e$, and initial conditions (p_o, v_o). That is, larger forces f_b and f_e will result in greater displacements and speeds; larger initial conditions (p_o, v_o) will also result in larger displacements and speeds. On the input space, we use the partial order defined by by $u \leq v$ provided $u_1 \geq v_1$ and $u_2 \leq v_2$. Consequently, we have $u_L = (1, 0)$ and $u_H = (0, 1)$. Since the brake force map $\pi : \mathcal{U}_1 \to \mathcal{F}_b$ is monotone, the flow is an order preserving map also with respect to the brake input $\mathbf{u}_1$. In the next section, we illustrate the components of the powertrain.

B. Powertrain

The dynamics of the powertrain take as control inputs $u = (u_1, u_2) \in [0, 1] \times [0, 1]$, where the first component u_1 denotes the brake pedal percent input, and the second component u_2 denotes the throttle pedal percent input [5]. In our application, these inputs can be administered either by the driver or by the automatic controller. The output of the system is assumed to be the torque applied at the wheel of the vehicle f_e. An overview of the system is provided in Figure 3(b).

The first component of the powertrain is the Engine Control Unit (ECU). This sub-system determines the fuel injection rate $i \in [0, 1]$ into the Internal Combustion Engine (ICE), and the current gear $q \in \{1, 2, 3, 4, 5, 6\}$ of the gearbox. The inputs to this block consist of the current velocity of the vehicle v, the throttle pedal input u_2 and the brake pedal input u_1. The second component of the powertrain is the Internal Combustion Engine (ICE). The output of this system is the torque τ applied by the flywheel, and the input is the fuel injection rate administered by the ECU. The third component of the powertrain is the gearbox. This module consists of the transmission with a fixed gear ratio. All switching logic is determined by the ECU, which sends a reset input R to the gearbox when a gear shift has been determined. The gearbox takes torque at the flywheel τ and converts it to the torque τ_q based on the current gear. The last component of the powertrain is the drivetrain. This component transfers torque at the gearbox τ_q to force applied at the wheel f_e. This module consists of the flywheel, torque converter, variable gear ratio transformer, propeller shaft, final drive and drive shaft (details can be found, for example, in [29]).

For the powertrain model, the order preserving property of the output f_e with respect to throttle input u_2 does not hold in general. This is due to the complexity of the ECU, which controls the fuel injection rate in a manner that optimizes a set of performance metrics, such as emissions, engine thermodynamic efficiency, with transients that can be quite complex and non-monotone [5]. By design, however, this is performed in a manner that generates monotone input-output behavior at *steady-state* [10].

Therefore, the dynamics of the vehicle system that take brake u_1 and throttle u_2 commands as inputs and provide speed and displacement as output are order preserving with respect to constant throttle input at least after an initial transient. Hence, we restrict the control commands to be constant with time, so that the system dynamics generate an order preserving flow with respect to the inputs after an initial transient time ϵ. In the next section, we illustrate how to identify the vehicle dynamics for the maximal braking and throttle inputs, which is the only knowledge on the model required by our algorithm.

C. System Identification

In order to model how the powertrain responds to constant control inputs (maximal braking and maximal throttle), in principle one should model the details of all the blocks in Figure 3(b). Rather than modeling this level of detail, we exploit the fact that the approach illustrated in Section III allows for disturbance inputs, which we use here to account for unmodeled dynamics. For the input signal $\mathbf{u}$ and velocity signal $\mathbf{v}$, define the non-deterministic engine force trajectories $\mathbf{F}_e(\mathbf{u}, \mathbf{v})$ as the set of all possible output engine force trajectories applied at the wheel given an input signal and velocity signal. When the powertrain model is combined with the vehicle physics, the vehicle velocity v and engine force at

the wheel f_e are coupled through the longitudinal dynamics introduced in (8). To capture this dependency, we say a system evolution is *realizable* if the velocity trajectory $\mathbf{v}(t, v_0, \mathbf{u}_1, \mathbf{f}_e)$ and engine torque trajectory $\mathbf{f}_e([0,t])$ satisfy (8) at all time and the inclusion

$$\mathbf{f}_e([0,t]) \in \mathbf{F}_e(\mathbf{u}([0,t]), \mathbf{v}([0,t], v_0, \pi(\mathbf{u}_1), \mathbf{f}_e)). \tag{9}$$

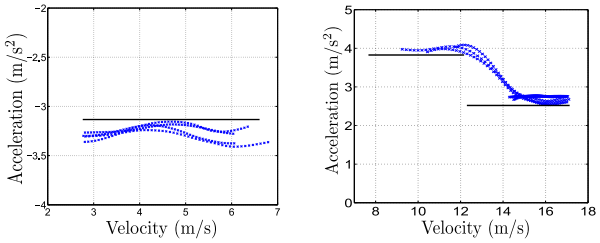
Let $\epsilon \in \mathbb{R}_+$ denote the maximum delay between initial changes in driver input u and steady state vehicle acceleration $\dot{v}$. This is the consequence of delays in: (1) software subsystems of the drive-by-wire throttle system; (2) delays in the powertrain due to chemical combustion; (3) gear shift delays; and (4) delays imposed by the Engine Control Unit (ECU) for filtering and environmental reasons. For a speed x_2, input u^*, and time-delay constant $\epsilon \geq 0$, the *permissible acceleration* set, denoted $\Upsilon(x_2, u^*, \epsilon) \subset \mathbb{R}$, is the collection of all accelerations given by

$$\begin{aligned}\Upsilon(x_2, u^*, \epsilon) := \\ &\{\tilde{f}(\mathbf{v}(t, v_0, \pi(\mathbf{u}_1^*), \mathbf{f}_e), \pi(\mathbf{u}_1^*(t)), \mathbf{f}_e(t)) \in \mathbb{R} \mid \\ &\exists\ \mathbf{f}_e([0,t]) \in \mathbf{F}_e(\mathbf{u}^*, \mathbf{v}([0,t], v_0, \pi(\mathbf{u}_1^*), \mathbf{f}_e)), \\ &\exists\ t \geq \epsilon,\ \exists\ v_0 \text{ s.t. } x_2 = \mathbf{v}(t, v_0, \pi(\mathbf{u}_1^*), \mathbf{f}_e)\},\end{aligned} \tag{10}$$

where $\mathbf{u}^*(t) = u^*$ for all t.

This is the set of all possible accelerations $\alpha = \tilde{f}(x_2, \pi(u_1^*), \mathbf{f}_e(t))$ achievable at velocity x_2 after $t \geq \epsilon$ seconds have elapsed under the constant input signal $\mathbf{u}^*$. Letting $x_1 = p$ and $x_2 = v$, we construct the vector field $f(x, u, d)$ of Section III-B for a fixed input $u = u^*$ as $f_1(x, u^*, d) := x_2$, $f_2(x, u^*, d_H) := \sup \Upsilon(x_2, u^*, \epsilon)$, $f_2(x, u^*, d_L) := \inf \Upsilon(x_2, u^*, \epsilon)$. For the case of maximum disturbance d_H (minimum disturbance d_L), the interpretation of $f_2(x, u^*, d_H)$ ($f_2(x, u^*, d_L)$) is that it represents the *greatest* acceleration (*least* acceleration) that can possibly be achieved at the velocity x_2 after the constant input u^* has been applied for *at least* $\epsilon \geq 0$ seconds. If $\Upsilon(x, u^*, \epsilon) = \emptyset$, then find the minimizer $x_2^* := \arg\min_{y_2 \in X_2} \{||y_2 - x_2|| \mid \Upsilon(y_2, u^*, \epsilon) \neq \emptyset\}$ and set $f(x, u^*, d) = f((x_1, x_2^*), u^*, d)$.

For implementing the feedback map of Section III-B, it is enough to identify experimentally $f_2(x, u_L, d_H)$ and $f_2(x, u_H, d_L)$. The identification procedure is as follows. To identify $f_2(x, u_L, d_H)$, we conducted a set of experiments called *braking trials*, in which, starting from an initial constant velocity, maximal braking $u_L = (1, 0)$ is applied and vehicle acceleration after $\epsilon = 0.7$s is recorded to provide data points for $\Upsilon(x_2, u_L, \epsilon)$ for the values of speed x_2 reached after ϵ. The value of ϵ was chosen to be enough for the vehicle to reach a steady state acceleration. Several trials for the same initial speed were performed and the infimum of these data points for every speed x_2 was computed to provide the value of $f_2(x, u_L, d_H)$. The set of initial velocities chosen is $\mathcal{V}_0 := \left\{\frac{1}{4}v_{max}, \frac{1}{2}v_{max}, \frac{3}{4}v_{max}, v_{max}\right\}$, in which $v_{max} = 8$ m/s for vehicle 1 (Blue IS 250) and $v_{max} = 17$ m/s for vehicle 2 (Grey IS 250). A brake trial consists of the following steps (1) accelerate each vehicle to a nominal constant velocity $v_0 \in \mathcal{V}_0$ on the vehicle path; (2) maintain velocity v_0 for at least 2 seconds, so transmission comes to a steady state; (3) apply brake input $u_L := (1, 0)$ via computer issued command, driver does not override command until vehicle reaches rest.



(a) (b)

Fig. 4. (a) A summary of all the experimental data for identifying $f_2^2(x_2^2, u_L^2, d_H^2)$ (black solid line) of vehicle 2. (b) A summary of all the experimental data for identifying $f_2^2(x_2^2, u_H^2, d_L^2)$ (black solid line) of vehicle 2.

Similarly, to identify $f_2(x, u_H, d_L)$, we conducted a set of experiments called *throttle trials*, in which starting from an initial constant velocity, maximal throttle $u_H = (0, 1)$ for the vehicle 1 and $u_H = (0, 0.5)$ for the vehicle 2 was applied. The set of initial velocities are given by $\mathcal{V}_0 := \left\{0, \frac{1}{4}v_{max}, \frac{1}{2}v_{max}, \frac{3}{4}v_{max}\right\}$, in which $v_{max} = 8$ m/s for vehicle 1 and $v_{max} = 17$ m/s for vehicle 2. A throttle trial consists of the following steps: (1) accelerate each vehicle to a nominal constant velocity $v_0 \in \mathcal{V}_0$ on vehicle path, if $v_0 = 0$, leave vehicle in idling state; (2) maintain velocity v_0 for at least 2 seconds, so transmission comes to steady state; (3) apply acceleration input via computer issued command, driver does not override command until vehicle reaches maximum velocity v_{max}.

For vehicle 1, which has $\mathcal{U}^1 = [0, 1] \times [0, 0.5]$ and $x_2^1 \in [0, 8.8]$ m/s, along path 1 (as shown in Figure 1(c)), we obtained $f_2^1(x_2^1, u_L^1, d_H^1) = -3.1$ and

$$f_2^1(x_2^1, u_H^1, d_L^1) = \begin{cases} 3.0 & x_2^1 \in [0, 7), \\ 1.75 & x_2^1 \in [7, \infty). \end{cases} \tag{11}$$

For vehicle 2, which has $\mathcal{U}^2 = [0, 1] \times [0, 1]$ and $x_2^2 \in [8.8, 20]$ m/s, along path 2 (as shown in Figure 1(c)), we obtained $f_2^2(x_2^2, u_L^2, d_H^2) = -3.1$ and

$$f_2^2(x_2^2, u_H^2, d_L^2) = \begin{cases} 3.9 & x_2^2 \in [0, 13), \\ 2.5 & x_2^2 \in [13, \infty). \end{cases} \tag{12}$$

Figure 4 shows the system identification results for vehicle 2. Similar plots were obtained for vehicle 1.

V. Software Implementation

The major software components of the ICA application are estimation, communication, and control (Figure 5).

A. Estimation

State estimation consists of several modules: longitudinal state measurement construction from raw measurements in UTM coordinates; calculation of the universal time; Kalman filter for local state prediction; and a full state estimator to construct the current state estimate set $\hat{x}(t) \subset X$ for the whole system. We denote with superscript "L" quantities computed on the local vehicle while with superscript "R" we denote

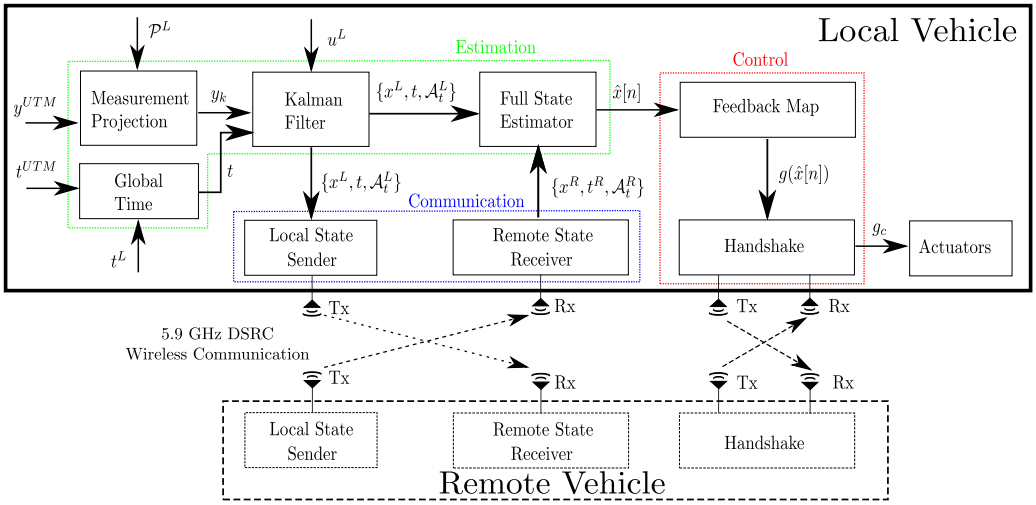


Fig. 5. Software system overview for the local vehicle. In the figure, we let the superscript L denote the local vehicle while the superscript R denotes the remote vehicle. The estimator (delimited by a green box) takes as inputs the UTM time and position information (y^{UTM} and t^{UTM}), the vehicle path information $\mathcal{P}^L$, the local vehicle time t^L, the local vehicle input u^L, and time/state information of the remote vehicle $\{x^R, t^R, \mathcal{A}_t^R\}$, and provides a set of possible position/speed configurations for the two-vehicle system $\hat{x} \subset X$. The communication system (delimited by the blue box) is a module that continuously sends to and receives information from the remote vehicle. The control system takes as input the state estimate set $\hat{x}$ computed locally and information from the control evaluation from the remote vehicle and returns the control input applied to the vehicle.

quantities of the remote vehicle that the local vehicle receives through the wireless communication. The measurement projection block is used to compute the longitudinal state measurement y_k from GPS and CAN measurements y^{UTM} (heading and position from GPS, velocity from CAN). The global time is computed by using a local time measurement t^L from the vehicle PC, and drift is removed by using the universal time t^{UTM} from the GPS system. The Kalman filter combines the longitudinal state measurement y_k and the pedal inputs u^L to compute the state estimate x^L and acceleration profile $\mathcal{A}_t^L$. This information is sent both to the communication system, and to the full state estimator. The full state estimator takes the current state estimate, time and acceleration profile $\{x^L, t^L, \mathcal{A}_t^L\}$, and combines this with the remote state information $\{x^R, t^R, \mathcal{A}_t^R\}$ to construct the full state estimate $\hat{x}[k]$ for use by the controller.

The time measurements available to each vehicle consist of the global time t^{UTM}, taken from the GPS system, and the local time t^L taken off the vehicle PC. The global time t^{UTM} is accurate, however only is received at a rate of 10 Hz, and can sometimes be unavailable due to message loss. The local time t^L is available at a higher rate of 1.5 GHz to a precision of 1 ms, however it is not accurate globally due to inherent drift in the crystal oscillator used to calculate time. To accurately compute a global time with update rate equal to 1.5 GHz, we combine the global time t^{UTM} with the local time t^L to produce the time t with using a simple moving average, where the moving average is updated every time a new global time t^{UTM} is made available.

The measurement projection block constructs a longitudinal state measurement from raw sensors on-board the vehicle. This involves projecting raw measurements onto the vehicle's path stored locally in $\mathcal{P}^L$. The source of absolute position and heading measurements is the GPS system, which provides updates at a fixed broadcast rate of 10Hz.

1) Kalman filter: For the Kalman filter, the longitudinal dynamics are assumed to be linear and hybrid, where the transmission state $q \in \{1, 2, 3, 4, 5, 6\}$ is assumed to be known at all time as obtained from the CAN bus. To model rolling friction, we add a fictitious frictional input, which takes values based on the sign of velocity, given by $u_3 = \text{sgn}(x_2)$. Since we seek to estimate also the acceleration, we add the engine torque at the wheels as a third state. Specifically, the Kalman filter state is $\hat{e} \in \mathbb{R}^3$, where the first component is longitudinal displacement, the second component is longitudinal velocity and the third component is the engine torque applied at the wheels. The output measurement is $y_k \in \mathbb{R}^3$, and incorporates longitudinal displacement, longitudinal velocity, and acceleration measured from the on-board accelerometer. The output is a discrete time signal indexed by $k \in \mathbb{N}$ with constant time-step $\Delta T > 0$, where the correspondence to time t is given by $t = k\Delta T$. The process dynamics are given by

$$\begin{aligned} \dot{\hat{e}}(t) &= A(q(t))\hat{e}(t) + B(q(t))u(t) + w(t), \\ y_k &= C_k\hat{e}(k\Delta T) + D_k u(k\Delta T) + v_k, \end{aligned}$$

where $w(t) \sim (0, Q)$ is continuous-time white noise with covariance Q, and $v_k \sim (0, R)$ is discrete-time white noise with covariance R.

Let the matrix $P(t)$ denote the estimated state error covariance, which is initialized to the identity matrix. Then, the prediction step of the filter is given by the following update equations, which represent a forward Euler approximation of the continuous time dynamics

$$\begin{aligned} \hat{e}(t) &= \hat{e}(t^-) + t_\Delta(A(q(t))\hat{e}(t^-) + B(q(t))u(t)) \\ P(t) &= P(t^-) + t_\Delta(A(q(t))P(t^-) + \\ &\quad P(t^-)A(q(t))^T + Q), \end{aligned}$$

where t^- is the time of the previous update, and $t_\Delta := t - t^-$. A prediction step is performed every time the software system

updates the current state, therefore, in general the time-step t_Δ is not constant. The correction step occurs only when a new longitudinal state measurement y is available and consists of the following update equations

$$\begin{aligned} K_k &= P(t^-)C^T(CP(t^-)C^T + R)^{-1} \\ \hat{e}(t) &= \hat{e}(t^-) + K_k(y_k - (C\hat{e}(t^-) + Du(t))) \\ P(t) &= (I - K_kC)P(t^-)(I - K_kC)^T + K_kRK_k^T. \end{aligned}$$

By nature of the fixed rate of measurements (discrete-time) and continuous-time inputs, the filter is said to be hybrid [25].

The matrices A, B, C, and D, have been identified from data for every gear q employing the system identification toolbox within MATLAB. In particular, we used a gray-box technique, where the system identification determines a vector of parameters, given a matrix structure derived from first principles. In particular, we have a second order system with rolling friction and inputs. We assume a multiplicative gear ratio from engine input to change in wheel torque. Therefore, the matrices are of the following form

$$A(q) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & a(q) \end{bmatrix}, B(q) = \begin{bmatrix} 0 & 0 & 0 \\ b_1 & 0 & b_2 \\ 0 & \alpha(q)b_3(q) & 0 \end{bmatrix},$$
$$C(q) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, D(q) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ b_1 & 0 & \alpha(q)b_3(q) \end{bmatrix}.$$

Data to preform this identification task was taken from four driving trials with varying input signals. The input signals were chosen by the driver to ensure an adequate sweep of the vehicles dynamic range under consideration. Each trial was taken on the path for which the vehicle normally drives on.

From the experimental data collected, we obtained for $q = 1$ that $a(q) = -2.5$, $b_1 = -5$, $b_2 = -0.1$, $b_3(q) = 5$, and $b_1 = 0.002$. For $q \in \{2, 3, 4, 5, 6\}$, we obtained that $a(q) = -1$, $b_1 = -5$, $b_2 = -0.1$, $b_3(q) = 5$, and $b_1 = 0.002$. The gear ratios are given by $\alpha(1) = 3.5$, $\alpha(2) = 2.0$, $\alpha(3) = 1.5$, $\alpha(4) = 1.2$, $\alpha(5) = 1$, and $\alpha(6) = 0.8$, which were taken from a technical data sheet [1]. This model was validated by comparing simulations obtained with an experimental input signal with the experimental trajectories.

To implement the Kalman filter, we chose the process and output noise covariance matrices to maximize noise rejection while still maintaining satisfactory bandwidth. We assume all noise processes are independent and identically distributed and have no mode dependency, therefore, the covariance matrices are all diagonal. The matrices are given as $R = \text{diag}(0.5, 0.3, 1)$ and $R = \text{diag}(0.5, 1, 1)$.

The Kalman filter is used to construct a state prediction. This is accomplished by computing the *acceleration profile* $\mathcal{A}_{\bar{t}}$, a set-valued signal containing all possible acceleration trajectories for future times $t \geq \bar{t}$. This allows to predict the set of possible speeds $\hat{e}_2(t)$ for $t \geq \bar{t}$. Mathematically, this is given as $\hat{e}_2(t) \in \hat{e}_2(\bar{t}) + \int_{\bar{t}}^{t} \mathcal{A}_{\bar{t}}(\tau)d\tau$. As mentioned in Section III-C, Algorithm 2 requires a two-vehicle state prediction, which has a tunable time-step Δ_p, which can be chosen by the test engineer, assumed to be less than 1.5 sec in total. With such a short time scale, it is reasonable to assume the input stays constant, that is $u(t) = u(\bar{t})$ for all $t \geq \bar{t}$. To account for the error of this assumption, we add a configurable window parametrized by the parameter $\beta \in \mathbb{R}_+$ to the resulting acceleration. As β is taken to 0, the prediction is assumed to be exact. The calculation is carried out, to obtain upper and lower bound sequences $[l_k, h_k]$, with the Hybrid Kalman filter as

$$\begin{aligned} \hat{e}_k &= \hat{e}_{k-1} + \Delta T(A(q(\bar{t}))\hat{e}_{k-1} + B(q(\bar{t}))u(\bar{t})), \\ [l_k, h_k] &= [0\ 0\ 1](C\hat{e}_k + Du(\bar{t})) + k[-\beta, \beta], \end{aligned}$$

where set addition is understood in the sense of the Minkowski sum. The acceleration profile $\mathcal{A}_{\bar{t}}(t)$ is found by taking the zero-order hold approximation of the sequence $[l_k, u_k]$.

2) Full state estimator: The Kalman filter output is the estimate of position and speed, which are the first two components of $\hat{e}$, denoted by x^L for the local vehicle and by x^R for the remote vehicle, the estimate of global time t, and the acceleration profile $\mathcal{A}_{\bar{t}}(t)$. The full state estimate is constructed by combining local state estimation from the Kalman filter with received remote vehicle state information. In accordance with feedback map $g(\hat{x})$, as defined in Algorithm 2, evaluating control involves discretizing the flow and constructing the current state estimate $\hat{x}[n]$ and a prediction $\hat{x}[n+1]$. We now define the algorithm for computing the full state estimate and prediction, with arguments local state information $(x^L, t, \mathcal{A}^L_{\bar{t}^L})$, remote state information $(x^R, t^R, \mathcal{A}^R_{\bar{t}^R})$, and prediction time-step Δ_P. The state estimate is found with $FullStateEstimate$, defined in Algorithm 3,which returns the current state estimate $\hat{x}[n]$ and state prediction estimate $\hat{x}[n+1]$.

Algorithm 3 $(\hat{x}[n], \hat{x}[n+1]) = \text{FullStateEstimate}(x^L, x^R, t, t^R, \Delta_P, \mathcal{A}^L_{\bar{t}^L}, \mathcal{A}^R_{\bar{t}^R})$

Input: $(x^L, x^R, t, t^R, \Delta_P, \mathcal{A}^L_{\bar{t}^L}, \mathcal{A}^R_{\bar{t}^R}) \in 2^{X^L} \times 2^{X^R} \times \mathbb{R}^3_+ \times S(2^{\mathbb{R}}) \times S(2^{\mathbb{R}}) \times \mathbb{R}_+$

Synchronize remote state due to transmission delay
$\hat{x}^R_1[n] = x^R_1 + (t - t^R)x^R_2$, $\hat{x}^R_2[n] = x^R_2 + (t - t^R)[\inf \mathcal{A}^R_{\bar{t}^R}(t^R - \bar{t}^R), \sup \mathcal{A}^R_{\bar{t}^R}(t^R - \bar{t}^R)]$
$\hat{x}[n] = x^L \times \hat{x}^R_1[n] \times \hat{x}^R_2[n]$

Construct prediction
$\hat{x}^L_1[n+1] = \hat{x}^L_1[n] + \Delta_p \hat{x}^L_2[n]$, $\hat{x}^L_2[n+1] = \hat{x}^L_2[n] + \Delta_p[\inf \mathcal{A}^L_{\bar{t}^L}(t - \bar{t}^L), \sup \mathcal{A}^L_{\bar{t}^L}(t - \bar{t}^L)]$
$\hat{x}^R_1[n+1] = \hat{x}^R_1[n] + \Delta_p \hat{x}^R_2[n]$, $\hat{x}^R_2[n+1] = \hat{x}^R_2[n] + \Delta_p[\inf \mathcal{A}^R_{\bar{t}^R}(t - \bar{t}^R), \sup \mathcal{A}^R_{\bar{t}^R}(t - \bar{t}^R)]$
$\hat{x}[n+1] = \hat{x}^L_1[n+1] \times \hat{x}^L_2[n+1] \times \hat{x}^R_1[n+1] \times \hat{x}^R_2[n+1]$

Output: $(\hat{x}[n+1], \hat{x}[n]) \subset 2^X \times 2^X$.

B. Communication

The state prediction performed by the estimator is necessary to account for communication delays and avoid control to be evaluated on old information. Communication delay comprises all delay experienced from the instant measurement data is

populated on-board the local vehicle until the remote vehicle uses this state information to construct a capture set for control evaluation. This can be broken down into the following major components: (1) ICA application acquisition of state information from the local state estimator; (2) construction of a remote data message as commanded by the ICA application; (3) interface with communication layer Denso WSU radio; (4) physical delay in the wireless transmission of the information; (5) reception of the message from the remote vehicle communication layer; (6) population of this state information into the ICA application for use in capture set construction and subsequent control evaluation. From experimental results, we have found that the worst case delay is 0.4 seconds. Hence the multiple predictions performed to determine $\hat{x}[n+1]$ are such that the time $\Delta_p \approx 0.4$ seconds.

C. Control

The set-valued feedback map g is computed locally on each vehicle. To accommodate delay in the system arising from communication, software and actuators (as discussed before, we evaluate the feedback controller for a set of state estimate predictions). Let the state estimate $\hat{x}[n]_i \subset X$ represent the estimate on-board vehicle i at time t. Algorithm 3 can be used recursively to construct more state estimate predictions. Define the prediction horizon count $N_p \in \mathbb{N}$, which is a configurable design parameter. We construct the state estimate predictions on-board vehicle i, given by $\hat{x}[n+j]_i$ for $1 \leq j \leq N_p$, as follows $(\hat{x}[n+j]_i, \hat{x}[n+j-1]_i) = \text{FullStateEstimate}(\hat{x}[n+j-1]_i, t+j\Delta_p, t^R+j\Delta_p, \Delta_p, \mathcal{A}^L_{t^L}, \mathcal{A}^R_{t^R})$, where the local vehicle refers to vehicle $i \in \{1,2\}$. We then use the set of predictions to evaluate the feedback map g on-board vehicle $i \in \{1,2\}$, implemented as $g(\hat{x}[n]_i) := \bigcap_{1\leq j\leq N_p} \text{FeedbackMap}(\hat{x}[n+j]_i, \hat{x}[n]_i)$.

Before applying control, the two vehicles should reach an agreement on the control commands to apply. In general, we have that $\hat{x}[n]_1 \neq \hat{x}[n]_2$. However, both sets contain the true system state x by construction. As a consequence, we have that $g(\hat{x}[n]_i) \subseteq g(x)$ given the order reversing property of the map g. As a consequence, we can take $g(\hat{x}[n]_1) \cup g(\hat{x}[n]_2)$ as the set of all possible safe control choices. In practice, we implement this with a handshake mechanism to guarantee that both vehicles choose the same actions. Specifically, the handshake module remains in the trivial initial state until a collision is predicted on-board the local vehicle. From Algorithm 2, a collision is predicted on-board vehicle i when $g(\hat{x}[n]_i) \neq \mathcal{U}$, at which point a message is sent to the remote vehicle indicating a collision has been predicted. Vehicle i then waits for a message indicating a collision has been predicted on-board the second vehicle j. If no such message is received, the application sleeps for 10 ms and then re-sends the message denoting a collision has been predicted (in case the message was not received). This process continues until a message has been received from vehicle j, or it times out. If a message is received, then a consensus control is chosen and applied to the local actuator of both vehicles.

VI. Intersection Collision Avoidance Experiments

A. Experiment Setup

Experiments were conducted at the TEMA test track in Ann Arbor, Michigan employing two modified Lexus IS 250 vehicles (Figure 1(c)). Both vehicles run ICA as they approach the intersection. The velocity of approach is not fixed, however it must be within safe limits. Each path is stored as a list of UTM co-ordinates on the respective vehicle. The speed limits for path 1 are $v_{min} = 0$ m/s and $v_{max} = 8.8$ m/s, while the speed limits for path 2 are $v_{min} = 8.8$ m/s and $v_{max} = 18$ m/s. The bad set parameters chosen are $L^1 = 55$ m, $L^2 = 75$ m, $H^1 = 65$ m and $H^2 = 85$ m. These values can be changed as they are only input parameters to the algorithm. For the specific implementation, we chose them in such a way that sufficient separation would be maintained by the vehicles when crossing the intersection. The input sets are chosen to be $\mathcal{U}^1 := [u^1_L, u^1_H] = [0, 0.3] \times [0, 0.5]$ and $\mathcal{U}^2 := [u^2_L, u^2_H] = [0, 0.3] \times [0, 1]$, which represent extremal inputs that maintain comfortable driving conditions. In general, these are design parameters that engineers have the freedom to change based on road surfaces, vehicle capabilities and general intersection dependent considerations. However, these need to remain fixed during the course of an experiment or implementation.

We consider two real-world scenarios, which we refer to as "use cases". For use case A, we assume a merging vehicle enters the intersection without properly surveying for oncoming traffic. Since the vehicle has already entered the intersection (or the speed is too high such that this is unavoidable), the only solution is for the merging vehicle to apply throttle and the straight vehicle to brake. A visualization of this is provided in Figure 6(a). For use case B, we assume a merging vehicle is approaching an intersection at high speed, and likely misjudging the speed of oncoming traffic. The solution in this case is for the merging vehicle to apply brake while the straight vehicle applies the throttle. A visualization of this is provided in Figure 6(b). We performed a total of 28 trials, 15 for use case A and 13 for use case B.

B. Experiment results

All trajectories generated by the experiments are provided in Figure 7 in the displacement plane. As it is apparent from the plots, no trajectory ever entered the bad set, hence all collisions were averted. Also, the trajectories pass fairly close to the bad set, indicating that the control algorithm is non- conservative as expected from theory. In order to better quantify the performance, we calculated the distance of the trajectory of the system from the capture set, denoted γ, and the distance of the trajectory from the bad set, denoted ζ. Table I provides the summary of the results. This table shows that the trajectory never entered the capture set nor the bad set in any trial, which follows from the non-zero values of $\wedge\zeta$ and $\wedge\gamma$. This is expected from theory as the controller guarantees that trajectories starting outside of the capture set remain outside of the capture set. Furthermore, the distances of the trajectories from the capture set are very small and can be decreased by decreasing the prediction horizon Δ_p and

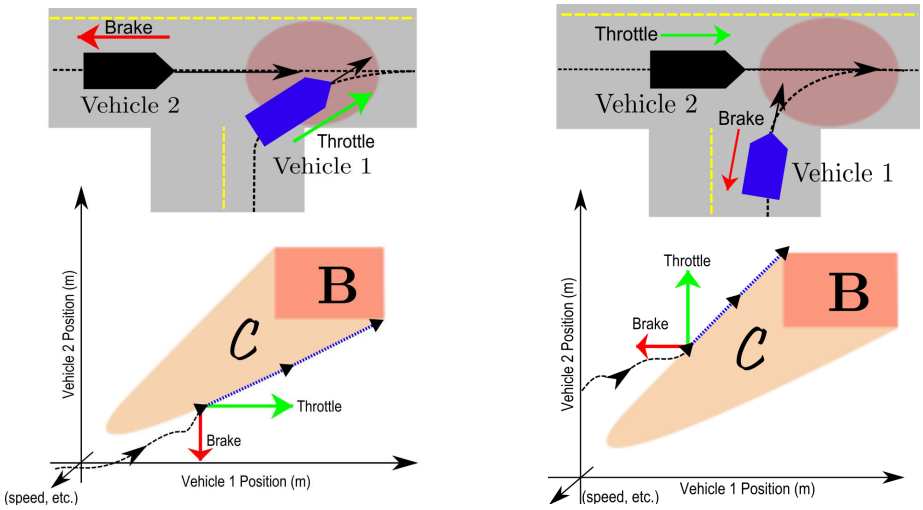


Fig. 6. (a) Use case A involves a merging vehicle entering the intersection without first checking oncoming traffic. The figure shows a top down cartoon of this scenario along with the system configuration related to the capture set in the position plane X_1 for a fixed pair of vehicle speeds. (b) Use case B involves a merging vehicle approaching the intersection while misjudging the speed of oncoming traffic. The figure shows a top down cartoon of this scenario along with the configuration of the system related to the capture set in the X_1 plane.

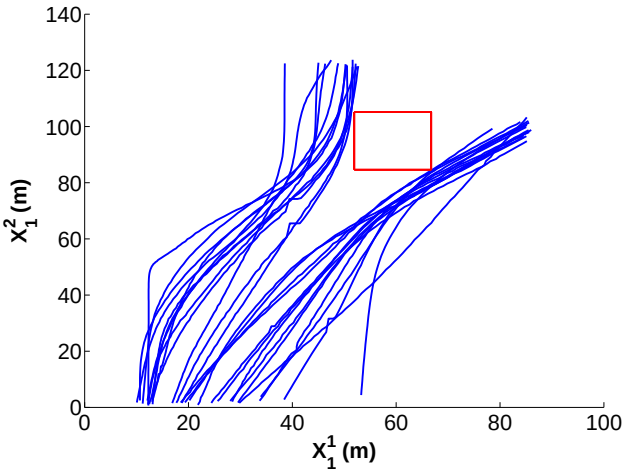


Fig. 7. All trajectories from all trials. The safety specification is maintained given that the flow of the system never entered the bad set $\mathbf{B}$ during any trial.

#	N_p	Δ_p	Info	$\zeta(\wedge,\mu)$	$\gamma(\wedge,\mu)$	(A,B)
4	3	0.4	P	0.9, 3	0.7 , 2.8	2, 2
4	4	0.2	P	0.6, 0.9	0.1, 0.6	2, 2
14	3	0.4	I	2, 5.9	2, 5.8	9, 5
6	4	0.2	I	0.7, 1.7	0.5, 1.4	2, 4

TABLE I

THE FIRST COLUMN INDICATES THE NUMBER OF TRIALS, THE SECOND COLUMN THE NUMBER OF PREDICTION STEPS N_p (SECTION V-C), Δ_p IS THE PREDICTION TIME (ALGORITHM 3),"P" DENOTES PERFECT STATE INFORMATION ($\beta = 0$ IN THE PREDICTION STEP OF SECTION V-A) AND "I" DENOTES IMPERFECT STATE INFORMATION ($\beta = 0.2$), ζ AND γ ARE THE DISTANCES OF THE TRAJECTORY FROM THE BAD SET $\mathbf{B}$ AND FROM THE CAPTURE SET $\mathcal{C}$, RESPECTIVELY, WITH $\wedge$ DENOTING THE MINIMUM VALUE AND μ DENOTING THE AVERAGE VALUE ACROSS THE TRIALS IN UNITS m.

removing the state uncertainty β. Larger prediction horizons lead the system to override sooner and as a consequence the distances from the capture set and from the bad set are larger. With no state uncertainty ($\beta = 0$), the trajectories pass closer to the capture set and to the bad set, indicating an aggressive and non-conservative controller. When uncertainty is introduced, the distances of the trajectory from the capture set and from the bad set increase because the algorithm applies control to keep an empty intersection between the predicted state uncertainty and the capture set. Our algorithms hence also provide a number of design parameters to compromise how aggressive the controller is (measured by how close to the bad set the trajectories go) with the control conservatism (the controller acts sooner than it could have). This trade off is relevant in practice because overriding the driver can be justified only if it is needed to keep the system safe.

Figure 8 shows an experimental trial with perfect state information ($\beta = 0$) and with use case A, while Figure 9 shows a trial for use case B and imperfect state information ($\beta \neq 0$). In use case A (Figure 8), the merging vehicle (vehicle 1) approached the intersection at a cruising speed of 6 m/s, while vehicle 2 approached the intersection at an accelerating speed of around 14 m/s. To avoid the collision, the drivers were overridden at time 19.7 sec when the state prediction hit the boundary of the capture set. At this time, automatic throttle was applied to vehicle 1 and automatic brake was applied to vehicle 2. This control results in vehicle 2 entering the intersection only (and immediately) after vehicle 1 has cleared the intersection. Vehicle 1 reached the speed limit v^1_{max} while applying throttle, after which time, the controller held the speed constant. The test ended after the merging vehicle exited the intersection, after which time, automatic control was deactivated and the driver retained control. While conducting this experiment, the system trajectory $\hat{x}(t)$ was at least within 0.7 m of the capture set, while never actually entering it, which

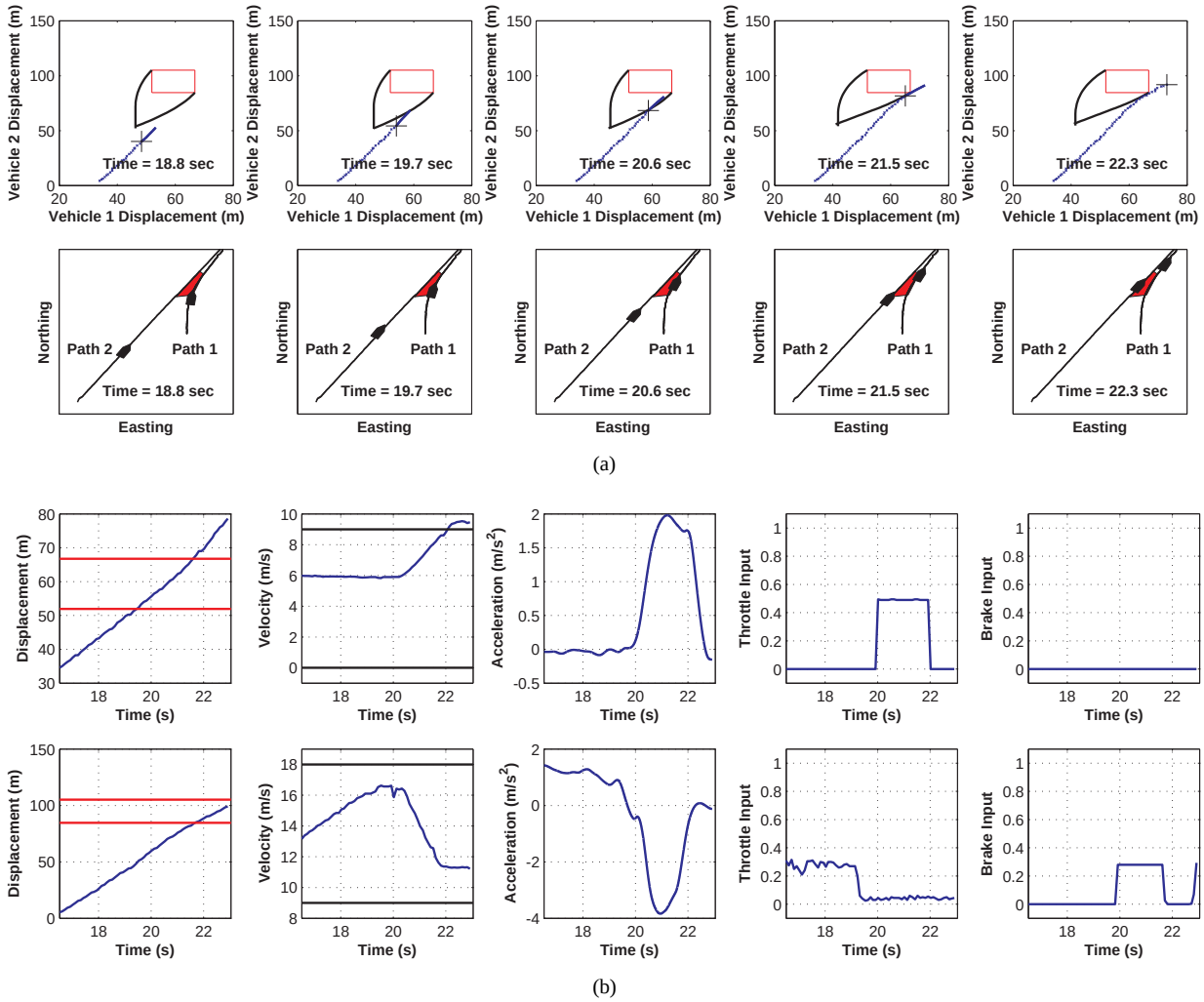


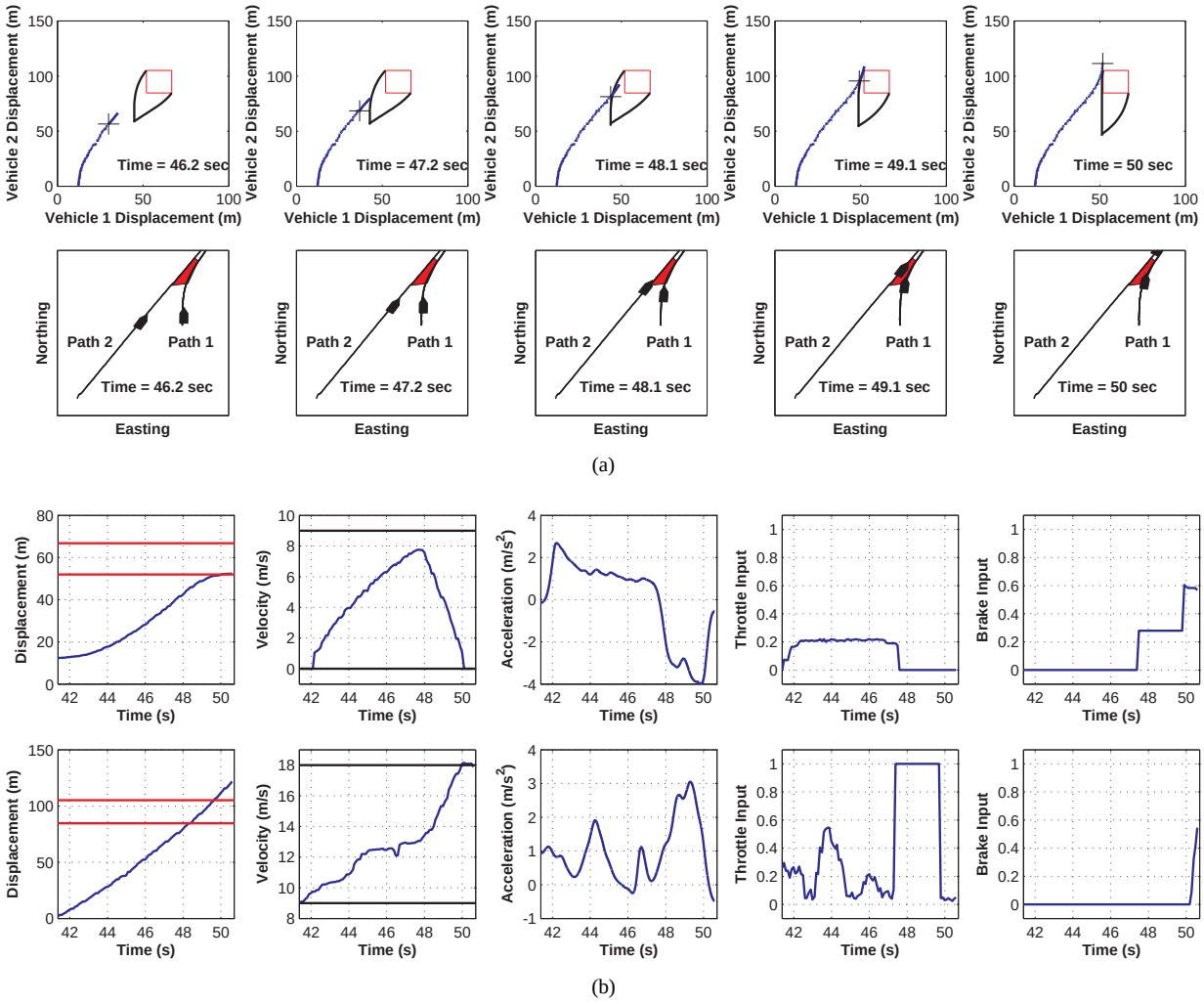
Fig. 8. An experimental trial for use case A. Here, perfect state information is assumed. (a) Snapshots showing the configuration of the vehicles at different times. The upper row shows the configuration of the vehicles (indicated by the cross) in the displacement space along with the capture set slice $\mathcal{C}$ (delimited by the black line) corresponding to the current vehicle speeds. The bad set is the red box. The solid blue line indicates the trajectory in the displacement space. The portion of this line ahead of the cross indicates the state prediction. The lower row shows the vehicle positions as they appear from a top-down view of the experiment. The red area corresponds to the bad set (red box in the upper row plots). (b) Signals for vehicle 1 are shown in the upper row, while the bottom row shows signals for vehicle 2. At time 19.7 sec, the state prediction hits the boundary of the capture set and hence vehicle 1 applies throttle and vehicle 2 applies brake.

implies safety was maintained and the control actions were not conservative.

In use case B (Figure 9), imperfect state information was considered using $\beta = 0.2$ m/s^2. In this trial, the merging vehicle (vehicle 1) started at rest, while vehicle 2 approached the intersection at an accelerating speed of around 8 m/s. Vehicle 1 attempted to violently accelerate and enter the intersection. To avoid the collision, the drivers were overridden at time 47.2 sec, when the set prediction hit the boundary of the capture set. In this case, automatic brake was applied to vehicle 1 and automatic throttle was applied to vehicle 2. This control results in vehicle 1 entering the intersection only (and immediately) after vehicle 2 has cleared the intersection. The merging vehicle reached the speed limit v^1_{min} while applying brake, after which time, the controller held the vehicle at rest. The straight vehicle reached the speed limit v^2_{max} while applying throttle, after which time, the controller held the vehicle at a constant speed. The test ended after the straight vehicle exited the intersection, after which time, automatic control was deactivated and the driver retained longitudinal control. While conducting this experiment, the system trajectory $\hat{x}(t)$ was within 0.6 m of the capture set, while never actually entering it, which implies safety was maintained and the control actions were not conservative.

VII. Conclusions

In this paper, we have presented algorithms and experimental validation on prototype vehicles for cooperative collision avoidance at intersections based on a formal control theoretic approach. Since the application considered is life-critical, algorithms for collision avoidance should have safety certificates. The proposed approach provides these certificates guaranteeing that the system stays collision free and that automatic control is not applied until absolutely necessary.



(a)

(b)

Fig. 9. An experimental trial for use case B. Imperfect state information is considered here ($\beta \neq 0$). The upper row shows the configuration of the vehicles (indicated by the cross) in the displacement space along with the capture set slice $\mathcal{C}$ (delimited by the black line) corresponding to the current vehicle speeds. The bad set is the red box. The solid blue line indicates the trajectory in the displacement space. The portion of this line ahead of the cross indicates the state prediction set. In this experiment, $N_p = 3$ and $\Delta_p = 0.4$ and the resulting uncertainty in position is very small (about 0.1 m), so it is hardly visible in the plot. However, the uncertainty on the speed is significant and it is about 0.5 m/sec. The velocity signal displays the estimate velocity x_2^L resulting from the Kalman filter. The lower row shows the vehicle positions as they appear from a top-down view of the experiment. The red area corresponds to the bad set (red box in the upper row plots). (b) Signals for vehicle 1 are shown in the upper row, while the bottom row shows signals for vehicle 2. At time 47.2 sec, the state prediction hits the boundary of the capture set and hence vehicle 2 applies throttle and vehicle 1 applies brake.

This is achieved by keeping the system state always outside the capture set, the set of all states from which a collision is unavoidable given the vehicle dynamics and the limitations on the control efforts. A number of parameters can be chosen by the designer, including the maximal and minimal brake and throttle efforts for automatic control, maximal and minimal speeds, the size of the collision set (bad set), the bounds on the modeling uncertainty, the communication delay, and the bounds on the uncertainty on the driver control actions. For example, if acceleration is not considered suitable for preventing a collision, one can set the upper and lower bounds of the throttle input to zero in the calculation of the capture set and the control map, so that evasive maneuvers will consider only braking. Of course, the control action will be more conservative in this case as the capture set will be larger. Similarly, the size of the bad set is an input parameter to the algorithm and it can be changed by the user depending on the specific intersection geometry. Experimentally, we have shown how to tune the prediction horizon and the number of prediction steps in order to adjust the conservatism, that is, how soon the controller decides that automatic control is needed to prevent an imminent collision. The later the automatic control acts, the less conservative the algorithm is, but the closer the system trajectories come to a collision (while still averting it). This trade off can be decided depending on the system specifications. The experiments finally illustrate that the (linear complexity) algorithms for evaluating the capture set and control actions are fast enough for real-time implementation, a feature that is necessary for the practical applicability of our approach. A number of future research

avenues are left to explore. These include incorporating a warning phase that gives the opportunity to the driver to react before automatic control becomes necessary. Scalability to more than two vehicles needs to be studied and initial results are promising [8]. Our approach can be applied where vehicles are on known crossing or merging paths, such as at intersections or when a vehicle merges onto a road from a parking lot or on the highway. Investigation should be carried out to extend the approach to road topologies other than intersections and merges, and to situations where intended vehicle paths and collision zones cannot be identified *a priori*.

REFERENCES

[1] http://www.vehix.com/car-reviews/2007/lexus/is-250/4dr-sport-sdn-auto-rwd/vehicle-specifications.

[2] Vehicle safety and fuel economy rulemaking and research priority plan 2011-2013. In *http://www.nhtsa.gov/staticfiles/rulemaking/pdf/2011-2013_Vehicle_Safety-Fuel_Economy_Rulemaking-Research_Priority_Plan.pdf*, 2011.

[3] M. Althoff. Reachability analysis and its application to the safety assessment of autonomous cars. In *PhD thesis, Technische Universitat Munchen*, 2010.

[4] L. Alvarez and R. Horowitz. Analysis and verification of the PATH AHS coordination-regulation layers hybrid system. In *American Control Conference*, pages 2460–2461, Albuquerque, New Mexico, 1997.

[5] A. Balluchi, L. Benvenuti, M.D. di Benedetto, C. Pinello, and A.L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: challenges and opportunities. *Proceedings of the IEEE*, 88(7):888 –912, jul 2000.

[6] F. Basma, Y. Tachwali, and H.H. Refai. Intersection collision avoidance system using infrastructure communication. In *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 422 – 427, oct. 2011.

[7] M. Braännstroöm, E. Coelingh, and J. Sjoöberg. Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Trans. on Intelligent Transportation Systems*, 11(3):658 –669, sept. 2010.

[8] A. Colombo and D. Del Vecchio. Efficient algorithms for collision avoidance at intersections. In *Hybrid Systems: Computation and Control*, 2012.

[9] European Road Safety Observatory (ERSO). Annual statistical report. 2006.

[10] Hosam K. Fathy, Rahul Ahlawat, and Jeffrey L. Stein. Proper powertrain modeling for engine-in-the-loop simulation. *ASME Conference Proceedings*, 2005(42169):1195–1201, 2005.

[11] R. Ghaemi and D. Del Vecchio. Safety control of piece-wise continuous order preserving systems. In *Proc. of IEEE Conference on Decision and Control*, 2011.

[12] D. Greene, Juan Liu, J. Reich, Y. Hirokawa, A. Shinagawa, H. Ito, and T. Mikami. An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists. *IEEE Trans. on Intelligent Transportation Systems*, 12(4):942 –953, dec. 2011.

[13] C. Le Guernic. Reachability analysis of hybrid systems with linear continuous dynamics. In *PhD thesis, Univerite Joseph Fourier*, 2009.

[14] M. R. Hafner and D. Del Vecchio. Computational tools for the safety control of a class of piecewise continuous systems with imperfect information on a partial order. *SIAM Journal of Control and Optimization*, 49:2463–2493, 2011.

[15] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 1790, B. Krogh and N. Lynch (Eds.), Springer Verlag, pages 130–144, 2000.

[16] N. Kaempchen, B. Schiele, and K. Dietmayer. Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios. *IEEE Trans. on Intelligent Transportation Systems*, 10(4):678 –687, dec. 2009.

[17] H. Kowshik, D. Caveney, and P. R. Kumar. Provable systemwide safety in intelligent intersections. *IEEE Trans. on Vehicular Technology*, pages 804 – 818, 2011.

[18] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for hybrid dynamics: the reachability problem. In *New Directions and Applications in Control Theory*, Lecture Notes in Control and Information Sciences, vol 321, W.P. Dayawansa, A. Lindquist, and Y. Zhou (Eds.), pages 193–205, 2005.

[19] J. Lee and B. Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Trans. on Intelligent Transportation Systems*, 13(1):81–90, 2012.

[20] V. Milanés, J. Pérez, E. Onieva, and C. González. Controller for urban intersections based on wireless communications and fuzzy logic. *IEEE Trans. on Intelligent Transportation Systems*, 11(1):243–248, 2010.

[21] G. K. Mitropoulos, I. S. Karanasiou, A. Hinsberger, F. Aguado-Agelet, H. Wieker, H-J Hilt, S. Mammar, and G. Noecker. Wireless local danger warning: Cooperative foresighted driving using intervehicle communication. *IEEE Trans. on Intelligent Transportation Systems*, 11(3):539–553, 2010.

[22] U.S. DOT National Highway Traffic Administration (NHTSA). Traffic safety facts. 2003.

[23] U.S. DOT National Highway Traffic Administration (NHTSA). Vehicle Safety Communications Applications vsc-a, second annual report, 2008.

[24] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[25] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

[26] C. J. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

[27] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

[28] L. Tu and C.-M. Huang. Forwards: A map-free intersection collision-warning system for all road patterns. *IEEE Trans. on Vehicular Technology*, 59(7):3233 –3248, sept. 2010.

[29] R. Verma, D. Del Vecchio, and H. Fathy. Development of a scaled vehicle with longitudinal dynamics of a HMMWV for an ITS testbed. *IEEE/ASME Trans. on Mechatronics*, 13:46–57, 2008.

Michael R. Hafner is a PhD candidate in the Systems Laboratory at the University of Michigan, Ann Arbor. His current research interests include safety control for nonlinear and hybrid systems in multi-agent settings. He received the B.S. in Electrical Engineering from the University of California Santa Barbara.

Drew Cunningham is a research engineer in the Integrated Vehicle Systems Department Toyota Motor Engineering and Manufacturing NA, Inc. His current research interests include the development and testing of advanced vehicle safety systems. He received the B.S. in computer science from Michigan State University. Contact him at drew.cunningham@tema.toyota.com.

Lorenzo Caminiti is a manager in the Integrated Vehicle Systems Department Toyota Motor Engineering and Manufacturing NA, Inc. His current research interests include the creation and validation of advanced vehicular safety applications. He received is masters in engineering from the University of Rome. Contact him at lorenzo.caminiti@tema.toyota.com.

Domitilla Del Vecchio received the Ph. D. degree in Control and Dynamical Systems from the California Institute of Technology, Pasadena, and the Laurea degree in Electrical Engineering from the University of Rome at Tor Vergata in 2005 and 1999, respectively. From 2006 to 2010, she was an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. In 2010, she joined the Department of Mechanical Engineering and the Laboratory for Information and Decision Systems (LIDS) at the Massachusetts Institute of Technology (MIT), where she is currently the W. M. Keck Career Development Associate Professor. She is a recipient of the Donald P. Eckman Award from the American Automatic Control Council (2010), the NSF Career Award (2007), the Crosby Award, University of Michigan (2007), the American Control Conference Best Student Paper Award (2004), and the Bank of Italy Fellowship (2000). Her research interests include analysis and control of nonlinear and hybrid dynamical systems with application to transportation networks and biomolecular networks.