# Sampling-Based Threat Assessment Algorithms for Intersection Collisions Involving Errant Drivers

**Georges S. Aoude** * **Brandon D. Luders** **
**Jonathan P. How** *** **Tom E. Pilutti** ****

* *Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, (e-mail: gaoude@mit.edu).*
** *Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, (e-mail: luders@mit.edu).*
*** *Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, (e-mail: jhow@mit.edu).*
**** *Ford Research and Advanced Engr, Dearborn, MI 48121, USA, (email: tpilutti@ford.com).*

**Abstract:** This paper considers the decision-making problem for a vehicle crossing a road intersection in the presence of other, potentially errant, drivers. This problem is considered in a game-theoretic framework, where the errant drivers are assumed to be capable of causing intentional collisions. Our approach is to simulate the possible behaviors of errant drivers using RRT-Reach, a modified application of rapidly-exploring random trees. A novelty in RRT-Reach is the use of a dual exploration-pursuit mode, which allows for efficient approximation of the errant reachability set for some fixed time horizon. Through simulation and experimental results with a small autonomous vehicle, we demonstrate that this threat assessment algorithm can be used in real-time to minimize the risk of collision.

*Keywords:* obstacle avoidance, driver assistance systems, autonomous vehicles.

## 1. INTRODUCTION

The field of road safety and safe driving has witnessed rapid advances due to improvements in sensing and computation technologies. Efforts are currently underway to use more advanced concepts like car-to-car (C2C) communications and intelligent vehicle highway systems (IVHS) for improving the safety and efficiency of human-driven ground transportation systems. Negotiating a traffic intersection safely is one of the most challenging driving tasks. An estimated 45 percent of injury crashes in the United States are intersection-related, and result in approximately 22 percent of roadway fatalities in the US. Most of them happen at intersections with stop signs or traffic signals (Fat, 2008). A main cause of these accidents is the driver's inability to correctly assess and/or observe the danger involved in such situations (Bougler et al., 2008). These facts suggest that driver assistance or warning systems may have an opportunity in reducing the number of accidents.

The Intersection Decision Support (IDS) project (in the US) and the InterSafe project (in Europe) partner with several universities to try to explore the requirements, tradeoffs, and technologies required to create an intersection collision avoidance system, and demonstrate its applicability on selected dangerous scenarios (Bougler et al., 2008; Fuerstenberg et al., 2005). Improving safety at intersections is also relevant for motion planning of autonomous vehicles. Talos, the MIT autonomous Land Rover LR3 that participated in in the 2007 DARPA Grand Challenge (DGC), faced a challenge common to most contestants: negotiating intersection traffic (Leonard et al., 2008). A threat assessment module embedded in the motion planner of autonomous systems could improve its safety by considering, during trajectory generation, the risks incurred by the different behaviors of the other vehicles.

To characterize the threat level of dynamic road situations, several measures have been proposed. These approaches typically measure collision risk by time-to-collision ($TTC$) and its variants, headway time ($t_h$), or required deceleration ($a_{req}$)(Karlsson et al., 2004). However, these measures are mainly tailored to frontal collision warning systems where the unpredictable dangerous driver is leading the host driver, and cannot typically be applied to intersection scenarios where the dangerous driver can approach from various angles. Ref. (Hillenbrand and Kroschel, 2006) investigated a collision mitigation system for intersection-like scenarios using a time-to-react ($TTR$) measure. However, it is assumed that a prediction model of the other vehicle's future motion is available, which is unlikely for a driver behaving erratically and thus unpredictably.

In this paper, we are interested in the problem of assisting human drivers or autonomous vehicles in negotiating busy intersections in the presence of possibly erring drivers. The proposed approach is based on a threat assessment algorithm that models the intersection problem as a pursuit-

⋆ Research funded by Ford Motor Company and Le Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) Graduate Award.

evasion game, and solves it on-line using a modified form of rapidly-exploring random trees (LaValle, 1998; Kuwata et al., 2009). A key advantage of the RRT algorithm is its natural bias toward exploring the reachable space. It is applied here to achieve computationally efficient, real-time construction of a vehicle's reachability set for a fixed time horizon in a dynamic and possibly complex environment involving several vehicles and obstacles, at the expense of completeness guarantees. The approach is demonstrated through both simulation results and experiments in the RAVEN facilities (How et al., 2008).

## 2. PROBLEM STATEMENT

Given a "host" vehicle approaching a road intersection involving one or more errant vehicles, the objective of this work is to evaluate the threat level that the host vehicle incurs by following each of its possible escape maneuvers. We define the threat level as the remaining time to earliest possible collision of the host vehicle with any errant vehicle.

To bound the number of vehicles affecting the decision making, the region of interest is localized to a finite volume around the intersection, called the *active region*. We assume that every vehicle in the active region has been classified as either predictable or errant by a classification algorithm (Aoude and How, 2009), part of the onboard threat assessment module. To simplify the problem, we further assume that only two agents are involved: an errant vehicle, modeled as a hostile agent which may not be following the rules of the road; and a host vehicle, assisted by the threat assessment algorithm to minimize the threat of an intersection collision.

**Vehicle Model:** Each vehicle is modeled using the standard nonlinear bicycle model,

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \frac{v}{L}\tan(\delta), \quad \dot{v} = a, \tag{1}$$

where $x$ and $y$ refer to the rear axle position, $v$ is the forward speed, $\theta$ is the vehicle heading, $L$ is the wheelbase of the vehicle, $a$ is the forward acceleration, and $\delta$ is the steering angle (positive counter-clockwise). The state of the vehicle is $(x, y, \theta, v) \in \mathcal{S}$, and the input is $(\delta, a) \in U$, subject to the constraints $a_{\min} \leq a \leq a_{\max}$ and $|\delta| \leq \delta_{\max}$. Each admissible control function $u(t)$ is a piecewise constant function $u : [0, T_h] \mapsto U$, where $T_h$ is a finite and fixed time horizon.

**Game theoretic formulation of IP:** The intersection threat assessment problem (hereafter referred to as IP) can be formulated as a perfect information zero-sum differential game $\mathcal{G}$ with state constraints and free final time (Isaacs, 1999). The formulation is similar to the one presented in Ref. (Ehtamo and Raivio, 2001). Let subscripts 1 and 2 denote the errant vehicle and host vehicle, respectively. If we let $s_i(t)$ and $u_i(t)$ refer to the state and control of vehicle $i$ at time $t$, where $i \in \{1, 2\}$, then (1) can be represented as

$$\begin{aligned} \dot{s}(t) &= \begin{bmatrix} \dot{s}_1(t) \\ \dot{s}_2(t) \end{bmatrix} = \begin{bmatrix} f(s_1(t), u_1(t)) \\ f(s_2(t), u_2(t)) \end{bmatrix}, \\ s(0) &= s_0, \quad t \in [0, \infty). \end{aligned} \tag{2}$$

The controls and states are assumed to satisfy the constraints

$$C_i(s_i(t), u_i(t)) \leq 0, \qquad i \in \{1, 2\}, \tag{3}$$

The game terminates with a collision if the state of the game $s(t)$ enters the closed target set $\Omega$, expressed as

$$\|(x_1(t_f), y_1(t_f)) - (x_2(t_f), y_2(t_f))\| \leq \epsilon, \tag{4}$$

where $\epsilon$ represents the collision distance between the two vehicles, and the free final time $t_f$ of $\mathcal{G}$ represents the time to collision

$$t_f = \inf\{t | (s(t)) \in \Omega\}. \tag{5}$$

The value $t_f = +\infty$ indicates the game terminated without a collision. The objective of the host vehicle is to maximize the final time $t_f$, while the objective of the errant vehicle is to minimize it. Thus the payoff function of $\mathcal{G}$ is simply

$$J(s(t), u(t), t) = t_f. \tag{6}$$

Suppose that $\mathcal{G}$ admits a saddle point $(\gamma_1^*, \gamma_2^*) \in \Gamma_1 \times \Gamma_2$ in feedback strategies. If both players execute their feedback saddle-point strategies, the value of the game is

$$V(s(t), t) = \min_{\gamma_1 \in \Gamma_1} \max_{\gamma_2 \in \Gamma_2} J(s(t), u(t), t) \tag{7}$$

$$= \max_{\gamma_2 \in \Gamma_2} \min_{\gamma_1 \in \Gamma_1} J(s(t), u(t), t). \tag{8}$$

Computing the feedback solutions requires solving the Hamilton-Jacobi-Isaacs equation (Isaacs, 1999). Except for some simple cases, the solution quickly becomes intractable (Isler et al., 2005). Instead, open-loop representations of the feedback saddle-point strategies are typically solved. This paper proposes an approximate open-loop solution using sampling-based algorithms, which can efficiently identify feasible solutions for complex motion planning or reachability problems (Lavalle, 2006; Bhatia and Frazzoli, 2004; Isler et al., 2005).

**Sampling-based relaxation of IP game**: To find an approximate solution of $\mathcal{G}$ using sampling-based algorithms, we introduce the following two assumptions: 1) *The number of control functions (*i.e., *policies) of the host vehicle is finite.* Since the host vehicle typically follows the rules of the road, it is reasonable to predefine some typical escape paths of a "good" host driver facing a dangerous situation; and 2) *The game $\mathcal{G}$ has a fixed time horizon $T_h$* (typically in the order of few seconds). The value of $T_h$ should be long enough to ensure that the host vehicle has enough time to execute its escape maneuver in the active region. It also limits the size of the problem by neglecting capture solutions outside the active region.

Then the sampling-based relaxation of the IP game will approximate the lower value (8) of the open loop representation of the IP by solving

$$\tilde{V}(s(t), t) = \max_{u_2 \in \tilde{\mathcal{U}}_2} \min_{u_1 \in \tilde{\mathcal{U}}_1} J(s(t), u(t), t) \tag{9}$$

$$\approx \max_{u_2 \in \mathcal{U}_2} \min_{u_1 \in \mathcal{U}_1} J(s(t), u(t), t), \tag{10}$$

where $\mathcal{U}_i$, $i \in \{1, 2\}$ represents the set of admissible control functions for each vehicle, and $\tilde{\mathcal{U}}_1$ and $\tilde{\mathcal{U}}_2$ are subsets of $\mathcal{U}_1$ and $\mathcal{U}_2$ respectively. Here $\tilde{\mathcal{U}}_1$ is computed using a sampling-based algorithm (Section 3.1) that is probabilistically complete, while $\tilde{\mathcal{U}}_2$ is a user-defined approximation of $\mathcal{U}_2$ that consists of a finite number of typical control functions of an evader at an intersection.

## 3. OVERVIEW OF THE APPROACH

This section details the threat assessment algorithm, which computes the approximate sampling-based solution to the IP game (Section 2) for the host vehicle to safely avoid an erratic driver. Central to this algorithm is the RRT-Reach subroutine (Section 3.1), which efficiently approximates the reachability set for the errant vehicle via time parameterization (Section 3.2). The tree uses biases for both exploration and pursuit of the host vehicle, representing the errant vehicle's "objective" to minimize the collision time $t_f$. The threat assessment algorithm then uses the RRT-Reach tree to evaluate the safety of each available escape path (Section 3.3).

### *3.1 RRT-Reach Algorithm*

The RRT-Reach algorithm extends the rapidly-exploring random tree (RRT) (LaValle, 1998; Lavalle, 2006) algorithm, which grows a tree by randomly sampling points toward which dynamically feasible trajectories are simulated. In particular, we use the closed-loop RRT (CL-RRT) algorithm of Ref. (Kuwata et al., 2009), which samples inputs to a controller rather than the vehicle itself. The algorithm thus maintains the exploration bias of traditional RRT algorithms, while allowing for generation of smooth trajectories more efficiently. To approximate the solution to the IP game, the RRT-Reach algorithm adds a game-theoretical component to the RRT approach. The approach is called RRT-Reach because it uses RRT to create trajectories for the pursuer to reach the evader escape paths in minimum time. We assume that both the pursuer (errant driver) and evader (host driver) have full knowledge of each other's policies. The approach utilizes this perfect information assumption through efficient, biased sampling in the RRT-algorithm.

RRT-Reach operates in two modes, exploration mode and pursuit mode. It chooses the exploration mode with probability $p_{mode}$, and the pursuit mode with probability $1 - p_{mode}$. The more the environment is constrained, the higher the value $p_{mode}$ should be set. Algorithm 1 describes the flow of the RRT algorithm.

In the exploration mode (Algorithm 2), the algorithm explores the state space by approximating the reachability set of the pursuer. As with the traditional RRT algorithm, it generates a sample (line 1), uses some heuristic to sort the nodes (line 5), attempts to propagate a trajectory from each node to the sample (line 7), and checks for constraint violation (line 8).

In the pursuit mode (Algorithm 3), the algorithm uses the knowledge of the escape paths of the evader (host vehicle) to bias the sampling towards the position of the evader. This step starts by updating the position of the evader using the escape timestamp (line 3), which increases every time the pursuit step is called. It then uses this position as a sample for the pursuer tree to grow toward. In order to increase its efficiency in capturing the evader, it checks if the pursuer has a chance to arrive "on time" to the sample by computing the unconstrained minimum time from the root to the sample using Dubins distance (line 4). If on-time arrival is not possible, it throws the sample away, and moves to the next escape path sample. Otherwise, it performs a propagation step (line 7–17) similar to the one of Algorithm 2, but with an additional condition in the neighbor selection process: neighbors must have a timestamp smaller than the current escape timestamp to be selected, as capture is otherwise impossible.

To incorporate these time-based decisions, a "timestamp" has been explicitly added to the state of the vehicle to track the time along each generated trajectory. While propagating (Algorithm 4), if the timestamp reaches $T_h$, the propagation is interrupted and the current portion of the trajectory is checked for feasibility. Also, when searching for nearest neighbours, the algorithm skips any node with a timestamp already equal to $T_h$. Unlike traditional RRT approaches, the RRT-Reach tree is not used to identify some path which reaches a goal location. Rather, the entire tree is analyzed to find the maximum threat along each of the trajectories (Section 3.3). Thus in this case the "goal" is not some location, but in fact the vehicles being in a state of collision. Finally, note that RRT-Reach does not include a completeness guarantee on the reachability set; there may be some feasible trajectories which are not included when work on constructing the reachability set is completed. However, the problem of computing the full reachability set in real-time is computationally intensive when subject to complex dynamics and complex, dynamic environments. The RRT-Reach algorithm is designed to rapidly approximate the reachability set and improve the approximation with more available time, regardless of the current problem complexity.

**Algorithm 1** RRT-Reach Algorithm

```
1: Measure current vehicle state and environment
2: repeat
3:    Sample p_mode uniformly from [0,1]
4:    if p_mode is less or equal to exploration bias then
5:       Perform an exploration step (Algorithm 2)
6:    else
7:       Perform a pursuit step (Algorithm 3)
8:    end if
9: until time limit for growing tree is reached
```

**Algorithm 2** Exploration Step

```
1: Take sample for input to controller
2: repeat
3:    Update time range in time heuristics
4:    Find list of nearest neighbors using time range
5:    Sort list using distance heuristics
6:    for each sorted node do
7:       Call propagation function using simulation of controller
8:       if propagated portion is collision free then
9:          Add sample to tree break
10:      end if
11:   end for
12: until timestamp reaches time horizon and no collision free
    portion was found
```

### *3.2 Time Parametrization*

Simulation results (Section 4) have suggested that the use of time-parametrized heuristics in the nearest node selection process (line 5 of Algorithm 2; line 10 of Algorithm 3) can result in shorter paths and a more efficient approximation of the IP minimization component. The

**Algorithm 3** Pursuit Step

```
1: Update escape timestamp
2: for each escape path do
3:    Set sample equal to escape path at current path time
4:    if minimum unconstrained time to reach sample from root is
      less than escape timestamp then
5:       continue {continue to next escape path}
6:    else
7:       repeat
8:          Update time range in time heuristics
9:          Find list of nearest neighbors using time range and
            escape timestamp
10:         Sort list using distance heuristics
11:         for each sorted node do
12:            Call propagation function using simulation of con-
               troller
13:            if propagated portion is collision free then
14:               Add sample to tree break
15:            end if
16:         end for
17:      until timestamp reaches time horizon and no collision free
         portion was found
18:   end if
19: end for
```

heuristic acts as a suboptimal strategy for identifying time-minimal paths in the tree to new samples, resulting in a more realistic reachability set for the errant vehicle.

The time heuristic consists of partitioning the nodes into incremental time ranges, where the time of each node is measured from the root, and using only one range at a time in the nearest node search. The time ranges are considered in order from shortest time from root to longest time from root (line 4 in Algorithm 2; line 9 in Algorithm 3). Using the list of nodes whose timestamp lies inside the current time range, the nearest node function then uses the Dubins distance metric to compute the $k$ nearest neighbors (line 5 in Algorithm 2; line 10 in Algorithm 3). If none of the $k$ neighbors generate a feasible trajectory, and the time range has not yet reached the time horizon $T_h$, the algorithm moves to the next-furthest time range and repeats the cycle (line 12 in Algorithm 2; line 17 in Algorithm 3). Each nearest node search is limited to a subset of the tree nodes, significantly reducing the complexity of this calculation.

### *3.3 Threat Assessment Algorithm*

This section introduces the threat assessment (TA) algorithm used to solve the sampling-based relaxation of the IP (See Algorithm 5). The inputs to the TA algorithm are the reachability tree of the errant vehicle computed by RRT-Reach (Algorithm 1), and a list of escape paths that the host driver could follow. These escape paths can be learnt from statistical traffic data. For each escape path and for each timestamp, the algorithm checks the distance between the escape path and each node of the reachability tree *with the same timestamp*. If it finds that an escape path is within a distance smaller than some safety threshold distance of the other vehicle, it flags that escape path as unsafe, and stores the time of collision.

The TA algorithm generates a list of times of collision for each escape path, and return the path(s) with the highest time. By doing so, the TA algorithm performs the maximization operation in the IP value function (10). Note that since the algorithm stops checking times beyond the time horizon, we tacitly assume that the escape paths flagged "safe" have a time of collision equal to infinity.

**Algorithm 4** Threat Assessment Algorithm

```
1: Compute reachability tree of errant vehicle using RRT-Reach
   (Algorithm 1)
2: Obtain list of escape paths of host vehicle
3: repeat
4:    for each escape path E_j flagged as safe do
5:       for each node n_k in the reachability tree with time stamp
         equal to t_i do
6:          if ||n_k − E_j(t_i)|| ≤ safety threshold then
7:             Set collision time of E_j to t_i and flag E_j as unsafe
8:             break
9:          end if
10:      end for
11:   end for
12:   Increment time t_i by Δt_i
13: until time t_i equals T_h or no more safe escape paths
14: return escape path(s) with highest collision time
```

## 4. SIMULATION RESULTS

In this section, the TA algorithm is demonstrated in simulation for a rural stop-controlled intersection scenario, known to have a high risk of collisions. Consider the problem of helping a driver on a major road (e.g. a highway) avoid a collision with an errant driver on a minor road (e.g. a rural road). As the errant driver approaches the major road, he/she misses the STOP sign or (equivalently here) loses control of the vehicle. It thus does not decelerate as expected, creating an unpredictable and dangerous behavior for other drivers. The host vehicle is equipped with a classifier (Aoude and How, 2009) which would quickly flag the other vehicle as dangerous, and then launches the TA algorithm to compute the risk of available escape maneuvers. This scenario deals with two main classes of intersection accidents, straight crossing paths (SCP) and right-turn into path (RTIP), which together account for more than 40% of light vehicle intersection accidents (Fat, 2008).

The initial position and heading of each vehicle are shown in Figure 1. The initial velocity is $v_1(0) = 1.5$m/s for the errant vehicle and $v_2(0) = 2.0$m/s for the host vehicle, whose velocity is assumed constant. The RRT-Reach algorithm uses a tree size of 2000 nodes with a time horizon of $T_h = 3$s, and relies on a pure-pursuit controller (Kuwata et al., 2008) to control the steering motion of the errant vehicle in its propagation step. To make the problem even more constrained, a small obstacle is added to the minor road around location (x,y) = (2.5, 0.5). It may represent some object (*e.g.,* tree) that is obstructing part of the errant driver's road.

Figure 1 shows the output of the reachability tree generated by RRT-Reach along with the four possible escape maneuvers for the host vehicle: 1) decelerate with acceleration $a = -2.5\text{m/s}^2$ until full stop, 2) keep moving straight with same initial velocity, 3) turn right with same initial velocity, and 4) turn left with same initial velocity. All four paths are dynamically feasible, and represent a subset of maneuvers that the host driver would typically follow to minimize the risk of collision at an intersection. Table 1

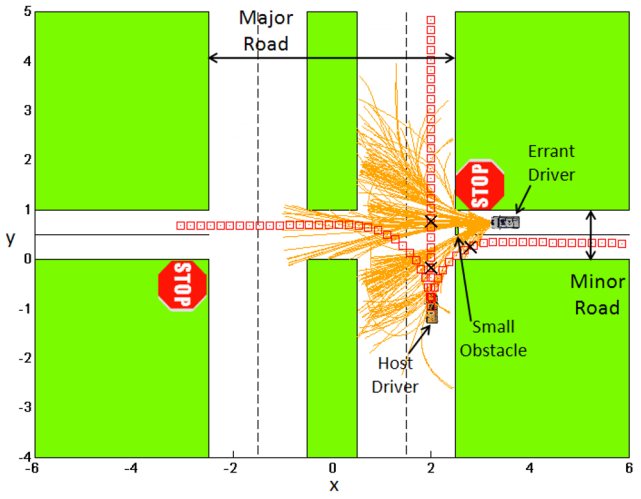



Fig. 1. TA Algorithm applied to the stop-controlled intersection. The tree generated for the errant vehicle using RRT-Reach is shown, as well as four escape maneuvers for the host vehicle. For each escape maneuver, the location of earliest possible collision is marked ×.

Table 1. Summary of TA simulation results

| Escape | Description | Time of Collision |
|---|---|---|
| 1 | Decelerate until stop | 1.14s |
| 2 | Keep going straight | 0.88s |
| 3 | Turn right | 0.78s |
| 4 | Turn left | $\infty$ |

shows the results of the TA algorithm. Since Maneuver 4 has the highest time of collision, it is chosen as the recommended maneuver, and the value of the game $\mathcal{G}$ is set to infinity.

## 5. EXPERIMENTAL RESULTS

A hardware testing platform is critical to analyzing and validating the performance of a threat assessment algorithm. Such a platform allows the incorporation of real-world uncertainty, such as uneven terrain or modelling errors, and provides meaningful data for learning algorithms. In these experimental results, an autonomous vehicle uses the TA algorithm to successfully avoid an errant, human-driven vehicle in a basic intersection scenario.

### *5.1 The RAVEN Testbed*

Hardware demonstrations were performed within the Real-time indoor Autonomous Vehicle test ENvironment (RAVEN), a testbed designed for the rapid prototyping of control algorithms for unmanned aerial and ground vehicles (How et al., 2008). Both vehicles used in the experiment are commercial-off-the-shelf remote-controlled trucks (Figure 2). The errant vehicle is driven remotely by a human operator, to emulate erratic driving behavior.

The autonomous host vehicle is controlled using an external Java planner which includes the real-time RRT-Reach algorithm. The host vehicle's waypoint path is determined using an extension of the real-time CL-RRT algorithm (Kuwata et al., 2009). This algorithm shares much of the same code and parameters as the RRT-Reach

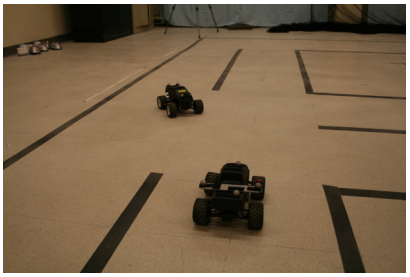


Fig. 2. Hardware setup in RAVEN.

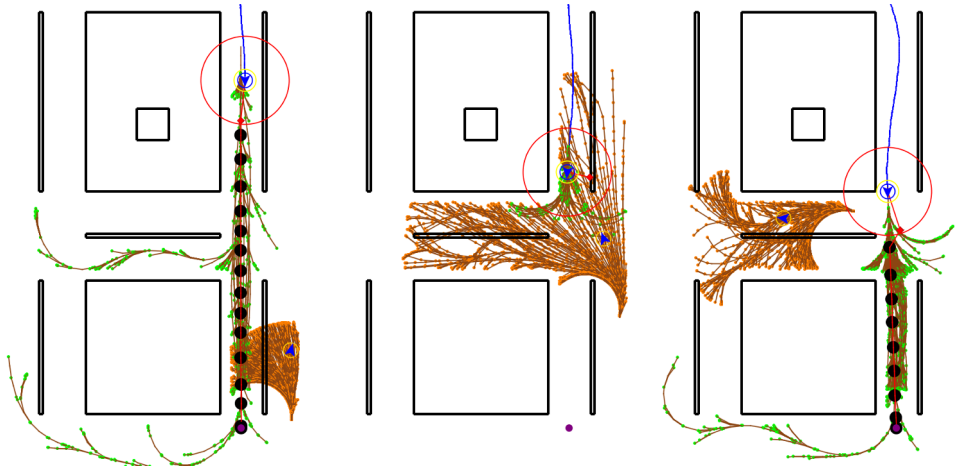
algorithm, reflecting the similarities of the two game-theoretic "players," and includes a closed-loop prediction model for the remote-controlled trucks. Unlike RRT-Reach (Section 3.1), the host vehicle *does* have a desired goal location and requires waypoint commands. Thus the external planner includes an execution loop which periodically identifies the best path in its RRT tree and sends it to the truck wrapper for execution (Kuwata et al., 2009). Finally, a safety constraint is added to each node, requiring that the vehicle can safely decelerate to a stop from each node before it is added to the tree.

The RRT-Reach algorithm is embedded in the external planner as a separate thread which runs asynchronously with the host planner thread. Over fixed time intervals (2s), the algorithm grows a new reachability tree based on the current errant vehicle position in RAVEN. At the end of each time interval, the reachability tree is relayed to the host RRT thread, which then relays it to the TA algorithm (Section 3.3). Any path in the host vehicle's tree can be an escape maneuver; even if no path is feasible, the vehicle will safely come to a stop due to the safety constraints.

### *5.2 Results*

We have tested several dangerous intersection scenarios in the RAVEN testbed; here we focus on the Left Turn Across Path/Opposite Direction (LTAP/OD) scenario, known to be one of the riskiest intersection encounters. In these experiments, the RRT-Reach algorithm uses a time horizon of $T_h = 5$s and a tree capacity of 2000 nodes, while the host vehicle RRT uses a tree capacity of 1000 nodes. The reference speed for both vehicles is $v_i = 0.5$ m/s.

In this scenario, the host vehicle is approaching an intersection when the errant driver arrives in the opposing direction and cuts off the host vehicle by turning left without stopping (Figure 2). The host vehicle's objective is to go straight through the intersection. Figure 3 shows the planner view of the host and errant vehicles, and their RRT trees, at various points in the experiment. As both vehicles approach the intersection, the host vehicle has a feasible path through the intersection (Figure 3(a)). However, as the errant vehicle enters the intersection, the TA algorithm indicates that the only safe action for the host vehicle is to stop, and it does so (Figure 3(b)). Once the errant driver cannot feasibly collide with the host, the host continues its path toward its goal (Figure 3(c)). (Because the RRT-Reach tree is initialized at the errant vehicle's present position and requires finite time to be computed, the errant vehicle is not at the current RRT-Reach tree root. This is accounted for in the TA algorithm.)



(a) $t = 6$s (b) $t = 9$s (c) $t = 12$s

Fig. 3. Results for a slow-moving errant driver (bottom chevron) turning left from an opposing direction to the host vehicle (top chevron).

## 6. CONCLUSION AND FUTURE WORK

This paper has presented a novel threat assessment algorithm which provides real-time assistance for a human or autonomous driver navigating through an intersection in the presence of potentially errant drivers. The planning problem can be formulated as a pursuit-evasion game. The centerpiece of the TA algorithm is the RRT-Reach algorithm, which applies RRTs with a dual exploration-pursuit mode to efficiently approximate the errant vehicle's reachability set. Given possible escape maneuvers, the TA algorithm can then identify the maneuver which minimizes the risk of collision. Hardware demonstrations in RAVEN have validated that this algorithm can be executed in real-time to successfully avoid errant drivers.

## 7. ACKNOWLEDGEMENTS

The authors would like to thank Daniel Levine, Vishnu Desaraju, and Amit Bhatia for their contributions to this research effort.

## REFERENCES

(2008). Fatality analysis reporting system encyclopedia. URL `http://www-fars.nhtsa.dot.gov/Crashes/CrashesLocation.aspx`.

Aoude, G.S. and How, J.P. (2009). Using Support Vector Machines and Bayesian Filtering for Classifying Agent Intentions at Road Intersections. Technical Report ACL09-02, Massachusetts Institute of Technology. URL `http://hdl.handle.net/1721.1/46720`.

Bhatia, A. and Frazzoli, E. (2004). Incremental search methods for reachability analysis of continuous and hybrid systems. In R. Alur and G.J. Pappas (eds.), *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, 142–156. Springer.

Bougler, B., Cody, D., and Nowakowski, C. (2008). California Intersection Decision Support: A Driver-Centered Approach to Left-Turn Collision Avoidance System Design. Technical report, Univ. of California, Berkeley.

Ehtamo, H. and Raivio, T. (2001). On Applied Nonlinear and Bilevel Programming or Pursuit-Evasion Games. *Journal of Optimization Theory and Applications*, 108(1), 65–96.

Fuerstenberg, K., GmbH, I., and Hamburg, G. (2005). A new European approach for intersection safety-the EC-Project INTERSAFE. In *Proceedings of IEEE Intelligent Transportation Systems*, 432–436.

Hillenbrand, J. and Kroschel, K. (2006). A Study on the Performance of Uncooperative Collision Mitigation Systems at Intersection-like Traffic Situations. In *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, 1–6.

How, J., Bethke, B., Frank, A., Dale, D., and Vian, J. (2008). Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2), 51–64.

Isaacs, R. (1999). *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Dover Publications.

Isler, V., Sun, D., and Sastry, S. (2005). Roadmap based pursuit-evasion and collision avoidance. *Proc. Robotics, Systems, & Science*.

Karlsson, R., Jansson, J., and Gustafsson, F. (2004). Model-based statistical tracking and decision making for collision avoidance application. In *Proceedings of the IEEE American Control Conference*, volume 4.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J.P. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5), 1105–1118.

Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., and How, J. (2008). Motion Planning in Complex Environments using Closed-loop Prediction. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*. Honolulu, Hawaii.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Ames, IA.

Lavalle, S.M. (2006). *Planning Algorithms*. Cambridge University Press.

Leonard, J., How, J., Teller, S., et al. (2008). A Perception-Driven Autonomous Urban Vehicle. *Journal of Field Robotics*, 25(10), 727 – 774.