# Threat Assessment Design for Driver Assistance System at Intersections

Georges S. Aoude, Brandon D. Luders, Kenneth K. H. Lee, Daniel S. Levine, and Jonathan P. How

***Abstract*— This paper considers the decision-making problem for a human-driven vehicle crossing a road intersection in the presence of other, potentially errant, drivers. Our approach relies on a novel threat assessment module, which combines an intention predictor based on support vector machines with an efficient threat assessor using rapidly-exploring random trees. This module warns the host driver with the computed threat level and the corresponding best "escape maneuver" through the intersection, if the threat is sufficiently large. Through experimental results with small autonomous and human-driven vehicles, we demonstrate that this threat assessment module can be used in real-time to minimize the risk of collision.**

## I. INTRODUCTION

The field of road safety and safe driving has witnessed rapid advances due to improvements in sensing and computation technologies. Active safety features like anti-lock braking systems and adaptive cruise control have been widely deployed in automobiles to reduce road accidents [1]. However, the US Department of Transportation (DOT) still classifies road safety as "a serious and national public health issue." In 2008, road accidents in the US caused 37,261 fatalities and about 2.35 million injuries. A particularly challenging driving task is negotiating a traffic intersection safely; an estimated 45 percent of injury crashes and 22 percent of roadway fatalities in the US are intersection-related [2]. A main contributing factor in these accidents is the driver's inability to correctly assess and/or observe the danger involved in such situations [3].

This data suggests that driver assistance or warning systems may have an appropriate role in reducing the number of accidents, improving the safety and efficiency of human-driven ground transportation systems. Driver assistance systems can be classified into two categories: infrastructure-based systems, such as intelligent vehicle highway systems (IVHS), and vehicle-based systems, such as car-to-car (C2C) communications [4]. Vehicle-based systems are expected to become available more quickly on commercial vehicles, compared to their infrastructure-based counterparts [5]. Such systems typically augment the driver's situational awareness, and can also act as collision mitigation systems. This research is focused on threat assessment algorithms that can be applied to vehicle-based systems.

Research on intersection decision support systems has become quite active in both academia and the automotive industry. In the US, the federal DOT, in conjunction with the California, Minnesota, and Virginia DOTs and several US research universities, is sponsoring the Intersection Decision Support (IDS) project [3]. In Europe, the InterSafe project was created by the European Commission to increase safety at intersections. The partners in the InterSafe project include European vehicle manufacturers and research institutes [6]. Both projects try to explore the requirements, tradeoffs, and technologies required to create an intersection collision avoidance system, and demonstrate its applicability on selected dangerous scenarios [3], [6].

Several measures have been proposed to characterize the threat level of dynamic road situations. These approaches typically measure collision risk by time-to-collision ($TTC$) and its variants [7], such as headway time [8] or required deceleration [9]. However, these measures are 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 may approach from a variety of angles. Ref. [5] investigated a collision mitigation system for intersection-like scenarios using a time-to-react ($TTR$) measure. However, it is assumed that the system has a prediction model of the dangerous vehicle's future motion, which is unlikely to be available for a driver behaving erratically and thus unpredictably.

In this paper, we are interested in assisting human drivers with negotiating busy intersections in the presence of possibly errant drivers with uncertain intentions. We propose a novel design for a threat assessment module, which combines a learning-based intention predictor with an efficient sampling-based threat assessor to compute the threats of errant drivers in real-time. This threat data is used to evaluate the safety of several possible escape paths, which may be proposed to the human driver if evasive maneuvers are warranted. The approach is demonstrated through experimental results in the RAVEN facilities [10].

## II. PROBLEM STATEMENT

We now define the road intersection threat assessment problem that is analyzed in this paper.

*Definition 1 (**Intersection Threat Assessment Problem**):* Consider a host vehicle $\mathcal{HV}$ approaching an intersection involving one or more other possibly errant vehicles $\mathcal{OV}s$; compute the escape maneuver $u^*$ that minimizes the threat level $\mathcal{T}$ that the $\mathcal{HV}$ incurs over a fixed time horizon $T_h$.

G. S. Aoude, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, gaoude@mit.edu

B. D. Luders, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, luders@mit.edu

K. K. H. Lee, S. M. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, kh2lee@mit.edu

D. S. Levine, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, denu@mit.edu

J. P. How, Richard C. Maclaurin Professor of Aeronautics and Astronautics, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, jhow@mit.edu

In this work, we assume the region of interest is localized to a finite volume around the host vehicle, called the *active region* and represented by a detection radius. This assumption bounds the number of vehicles affecting decision making, and partially represents the limitations of the sensors and wireless communications systems likely to be installed on the vehicles (Section III-B).

We consider $\mathcal{HV}$ to be safe if it avoids colliding with all other vehicles, denoted here as $\mathcal{OV}_i$, $i = 1 \dots N$. This constraint can typically be written in the form

$$\min_{t_0 \le t \le t_f} \|p_s(t) - p_{r_i}(t)\| > \epsilon \quad \forall i = 1 \dots N, \tag{1}$$

where $r_i(t)$ is the state of vehicle $i$ at time $t$, $p_s(t)$ is the position vector corresponding to $s(t)$, $p_{r_i}(t)$ is the position vector corresponding to $r_i(t)$, $\epsilon$ is the minimum allowed distance between the $\mathcal{HV}$ and the $\mathcal{OV}s$, and $t_f = t_0 + T_h$. We say that $\mathcal{HV}$ is in collision with $\mathcal{OV}_i$ if the tuple $(p_s(t), p_{r_i}(t))$ enters the closed "collision" set $\Omega_i$, whose boundary is given by the scalar equation

$$\|p_s(t) - p_{r_i}(t)\| = \epsilon. \tag{2}$$

In typical (i.e. uncertain and/or non-cooperative) driving scenarios, the future trajectories $r_i(t),\ t \in [t_0, t_f]$ of the $\mathcal{OV}s$ are not known bythe $\mathcal{HV}$. Moreover, some realizations of $r_i(t)$ may cause (1) to become infeasible for every admissible escape maneuver. Let $u : [t_0, t_f] \mapsto U$ denote the set of all admissible control sequences. To maximize the safety of $\mathcal{HV}$, we compute the admissible escape maneuver $u^*$ that minimizes the threat level $\mathcal{T}$, defined below, for all possible realizations of $r_i(t)$, $i = 1 \dots N$.

*Definition 2 (**Threat Level**):* Define the threat level $\mathcal{T}_i(u)$ corresponding to vehicle $\mathcal{OV}_i$ and escape maneuver $u$ as inversely proportional to $t_{c_i}$, the earliest possible time of collision:

$$\begin{aligned} \mathcal{T}_i(u(.)) &= \frac{1}{t_{c_i}(u(.))}, \\ t_{c_i}(u(.)) &= \inf\{t \mid (p_s(t), p_{r_i}(t)) \in \Omega_i\}. \end{aligned} \tag{3}$$

The threat level $\mathcal{T}$ corresponding to escape maneuver $u$ is then defined as

$$\mathcal{T}(u(.)) = \max_{i \in 1 \dots N} \mathcal{T}_i(u(.)). \tag{4}$$

*Definition 3 (**Best Escape Maneuver**):* The best escape maneuver $u^*$ is defined as

$$u^* = \arg\min_{u \in U} \mathcal{T}(u(.)). \tag{5}$$

Note that in (3), $p_s(t)$ is generated by the escape path $u$.

## III. Solution Approach

The proposed threat assessment provides assistance to the host vehicle $\mathcal{HV}$ in navigating dynamic environments populated by multiple other vehicles with uncertain intentions. The approach is demonstrated in intersection scenarios, but can be generalized to arbitrary road geometries. The assessment is computed in a threat assessment module (TAM) that is designed as a black box (Figure 1), allowing it to be easily incorporated in a driver assistance system. The TAM presented in this paper is an adaptation of the architecture developed in Ref. [11] to generate safe trajectories for autonomous vehicles.

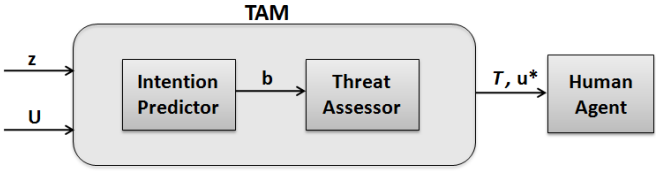



Fig. 1. High-level architecture of the Threat Assessment Module (TAM). The inputs of the TAM are the measurement history $z$ of the $\mathcal{OV}s$ and the set of control escape sequences $U$ generated for the $\mathcal{HV}$. The IP computes the intention vector $b$ for each $\mathcal{OV}$, used by the TA to identify the threat level $\mathcal{T}$and best escape maneuver $u^*$.

### A. High-level Architecture

The TAM consists of two main components, an intention predictor (IP) and a threat assessor (TA). The IP incorporates high-level reasoning in modeling the intentions of the $\mathcal{OV}s$, and uses observations to make a prediction of future behaviors. Such a predictor might, for example, classify the threat posed by other vehicles based on whether or not their modeled intentions conform to "typical" driving behavior [12], [13]. The TA converts this prediction into a set of potential paths that the $\mathcal{OV}$ is likely to follow. The threat level can then be evaluated for each candidate $\mathcal{HV}$ escape maneuver, in order to identify the best escape maneuver.

Several distinct representations have been proposed to model intentions for humans and autonomous systems; here we adopt the Ecological Recogniser architecture introduced by Ref. [14] and shown in Figure 2. Using the language of the figure, the $\mathcal{HV}$ is the recognising agent, while each $\mathcal{OV}$ is an intending agent. It is implicitly assumed that the list of intentions fully partitions the space of all possible behaviors for the intending agents. We will also follow the abstract definition that these intentions are directly responsible for the actions or activities executed by the intending agents [14]. Intentions may be desired plans (*e.g.,* following a straight path) or other high level intention states (*e.g.,* not following the rules of the road).

Let the $M$-vector $b_i$ denote the intention vector of $\mathcal{OV}_i$; the $j^{\text{th}}$ entry $b_i^j$ corresponds to the belief $\mathcal{OV}_i$ is operating under the $j$th intention. The definition of the threat level (4) is modified to embed the expectation of each vehicle's threat over all possible intentions,

$$\mathcal{T}(u(.)) = \max_{i \in 1 \dots N} \left( \sum_{j \in 1 \dots M} b_i^j \mathcal{T}_{ij}(u(.)) \right) \tag{6}$$

where $\mathcal{T}_{ij}$ is the threat value of the trajectories followed by $\mathcal{OV}_i$ as a realization of its $j^{th}$ intention.

### B. Agent Model

Each agent is modeled using the standard bicycle model

$$\begin{aligned} \dot{x} &= v\cos(\theta), \quad & \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= \frac{v}{L}\tan(\delta), \quad & \dot{v} &= a, \end{aligned} \tag{7}$$

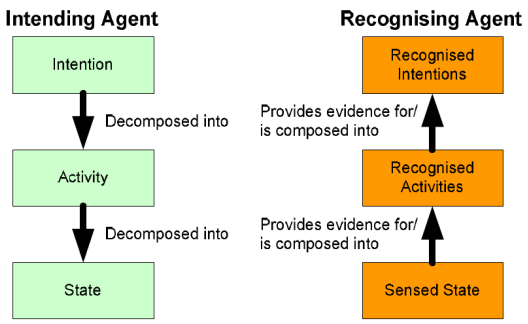



Fig. 2. The intentional behavior of the Intending Agent and the intention recognition of the Recognizing Agent [14].

where $(x, y)$ is the rear axle position, $v$ is the forward speed, $\theta$ is the heading, $L$ is the wheelbase, $a$ is the forward acceleration, and $\delta$ is the steering angle (positive counter-clockwise). The state of the vehicle is $s = (x, y, \theta, v) \in \mathcal{S}$, while the input is $u = (\delta, a) \in \mathcal{U}$, including the constraints $a_{\min} \leq a \leq a_{\max}$ and $|\delta| \leq \delta_{\max}$.

In the work that follows, we make several assumptions to constrain the focus on uncertainty in the prediction of obstacles or other agents. We assume that the current states and models of the $\mathcal{HV}$ and the $\mathcal{OV}s$ are perfectly known by the $\mathcal{HV}$ within the active region, and that all sensor measurements are noise-free. While this assumption is somewhat unrealistic, the addition of estimation filters on noisy measurements should have a minimal effect on the TAM architecture. The focus of this paper is on the planning subproblem, rather than sensing; future implementations will relax these assumptions. Additionally, the TAM is provided with a complete map of the environment *a priori*, excluding any dynamic obstacles or agents. Finally, the number of escape control functions of the $\mathcal{HV}$ is assumed to be finite. Since the $\mathcal{HV}$ typically follows the rules of the road, it is reasonable to predefine some typical escape maneuvers that a "good" host driver might follow, *e.g.,* maximum braking or an increase in throttle.

## IV. Developed Algorithms

This section describes the implementation of the different components of the TAM, tailored specifically for the road intersection problem. The IP component is based on the Ecological Recogniser architecture presented in [14]. The TA component combines knowledge of the intentions produced by the IP and domain-specific information of the environment to quickly propagate reachable paths for the moving vehicles using a reachability-based algorithm. Then, for each admissible escape maneuver, it evaluates the threat generated by these paths. It finally returns the maneuver with the lowest threat level over all vehicles. The result is an intention-reasoning TAM that runs efficiently and is suitable for real-time implementation.

### A. Intention Predictor (IP)

The IP translates observations that $\mathcal{HV}$ makes of the other $\mathcal{OV}s$ into a prediction of their intentions. Instead of using low-level reasoning to predict future *trajectories*, the IP uses higher-level logic to provide the TAM with predicted future *intentions*, which is key to more timely and accurate prediction of collisions. Each intention can then be translated to a richer set of future paths that exploit domain-specific knowledge using the TA (Section IV-B).

The chosen Ecological Recogniser architecture [14] is applicable to our problem because it is designed for an intelligent agent trying to recognize the intention of other agents with access to their state data but not actions. Additionally, this architecture can handle intentions which are not easily described by a simple set of rules, as can be the case in this problem. The Ecological Recogniser architecture has two key components, a pattern matcher and a reasoning module. The pattern matcher is a classifier that is trained offline to recognize the different intentions by observing agent trajectories, and operates online by continuously giving an estimate of the intentions over time. The reasoning module filters these estimates of intentions, along with knowledge of previous encounters with the agent, to give a final estimate of the intention vector $b$.

Since we are interested in estimating the intentions of vehicles near intersections, we use the specific implementation of the Ecological Recogniser that is based on an approach previously developed and demonstrated in simulation [13] for classifying other agent intentions using support vector machines (SVM) and Bayesian filtering. Even before considering autonomous vehicles, the human driver intention classification problem is very complex because of the various nuances of human behaviors. SVM is a suitable method for the IP because it has been shown to be a robust and efficient approach for classification problems [15], such as lane-change detection [16]. The IP design consists of an SVM combined with a Bayesian filter that uses the SVM outputs over a specific time period to compute the intention vector $b$. SVM-BF also includes a threshold detector, so the final $b$ vector is a unit basis vector specifying the most likely intention according to the threshold value. The SVM-BF was trained using human driving data collected in RAVEN [10]. For more details about the SVM-BF approach, please refer to [13]. Based on experimenting with different kernel functions and several combinations of features, the best results were obtained using the Gaussian radial basis function and combining the following three features:

- the relative distance $\Delta x$ between the $\mathcal{OV}_i$ and the entrance of the intersection it is approaching;
- the speed of the $\mathcal{OV}_i$;
- the longitudinal acceleration of the $\mathcal{OV}_i$.

Note that the SVM-BF algorithm is only activated when the distance between the $\mathcal{OV}_i$ and the $\mathcal{HV}$ is within some danger radius, and both vehicles are approaching the same intersection, which is representative of a limited computation budget and restricted knowledge of the world.

### B. Threat Assessor (TA)

The TA (Algorithm 1) computes a threat value of a candidate escape maneuver for the $\mathcal{HV}$ given a knowledge of the

**Algorithm 1** Intention-based-Threat-Assessment ($u$)

1: $\mathcal{T}_i \leftarrow 0 \quad \forall i$
2: **for** each agent $\mathcal{OV}_i$ **do**
3: *Reach-Set* $\leftarrow$ Eval-Intention-Reachability ($\boldsymbol{b}_i$)
4: **for** each intention $e_j$ with probability $b_i^j$ **do**
5: $\mathcal{T}_i \leftarrow \mathcal{T}_i + b_i^j \cdot$ Eval-Threat(*Reach-Set*,$\mathcal{OV}_i, e_j, u$)
6: **end for**
7: **end for**
8: **return** $\max_{i=1\ldots N} \mathcal{T}_i$

current states of the $\mathcal{OV}s$. It combines a fast sampling-based reachability method with intention prediction information provided by the IP (Section IV-A) to efficiently estimate the threat level. A similar version of the TA algorithm that assumes worst-case behavior for the other drivers was developed by the authors in [17]. The threat assessor has a finite time horizon, limiting the $\mathcal{HV}$ lookahead horizon of the possible future paths of the other $\mathcal{OV}s$, in order to focus calculations on imminent threats. This choice is domain-specific, but it should be long enough to allow the $\mathcal{HV}$ sufficient time to react in a dynamic environment. To obtain the best escape maneuver $u^*$, the TAM calls Algorithm 1 for each each maneuver $u$ in the set of escape maneuvers, and returns the maneuver with the minimum threat level.

Algorithm 1 first calls the Eval-Intention-Reachability subroutine (Algorithm 2) on line 3 to create the reachability set for each $\mathcal{OV}_i$. The resulting sets are biased based on the perceived intentions of the $\mathcal{OV}s$. For each intention $e_j$ of each $\mathcal{OV}_i$, Algorithm 3 is called to compute the threat incurred by the $\mathcal{HV}$ escape maneuver $u$ in the region reached by the paths that correspond to the intention $e_j$. Note that the earliest time to collision is converted into a threat value using (3) (line 3 of Algorithm 3). This value is weighted by the probability $b_i^j$ provided by the IP. Finally, the threat returned is computed as the maximum value of all threats created by each $\mathcal{OV}_i$ (line 8 of Algorithm 1). Note that, in practice, the computed reachability sets for each $\mathcal{OV}_i$ are saved between successive calls to Algorithm 1.

Algorithm 2, sometimes referred to as the RRT-Reach algorithm, extends the rapidly-exploring random tree (RRT) [18], [19] 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. [20], 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. A key advantage of the RRT algorithm is the scalability of the planning tree to use whatever computational resources are available.

Unlike traditional RRT approaches, the RRT-Reach tree is not used to identify some path that reaches a goal location. Rather, the entire tree is analyzed to find the maximum threat along each of the trajectories. The choice of samples is designed to be biased towards regions corresponding to the learned intentions of the $\mathcal{OV}s$. It is done through sampling

**Algorithm 2** Eval-Intention-Reachability ($b$)

1: Measure current vehicle state and environment
2: **repeat**
3: Take sample for input to controller using bias from intention probability vector $b$
4: **repeat**
5: Update time range in time heuristics
6: Find list of nearest neighbors using time range
7: Sort list using distance heuristics
8: **for** each sorted node **do**
9: Call propagation function
10: **if** propagated portion is collision free **then**
11: Add sample to *Tree* **break**
12: **end if**
13: **end for**
14: **until** timestamp reaches time horizon and no collision free portion was found
15: **until** time limit for growing tree is reached
16: **return** *Tree*

**Algorithm 3** Eval-Threat (*Reach-Set*, $\mathcal{OV}$, $e$, $u$)

1: **for** each path $path_k$ in *Reach-Set* that ends in a region of intention $e$ **do**
2: $t_k \leftarrow$ compute earliest time of collision of $path_k$ with $\mathcal{HV}$ escape maneuver $u$ within time horizon $T_h$
3: $\mathcal{T}_k \leftarrow \frac{1}{t_k}$
4: **end for**
5: **return** $\max_k \mathcal{T}_k$

some percent of the time in regions corresponding to the component $e_j$ of intention $e$. The remaining percent of the time is spent sampling uniformly in the environment.

Algorithm 2 includes several additional extensions to the RRT approach introduced in Ref. [20], based on time parameterization of the RRT tree. The extensions tailor the RRT algorithm to the efficient computation of intented paths by the $\mathcal{OV}s$.

First, since we are interested in both approximating the vehicle's fixed-horizon reachability set and checking for collision between moving vehicles, a "timestamp" has been explicitly added to the state of the vehicle to track the time along each generated trajectory. While propagating, if the timestamp reaches the time horizon $t_f$, where $t_f = t_0 + T_h$, the propagation is interrupted and the current portion of the trajectory is checked for feasibility. Also, when searching for nearest neighbors, the algorithm skips any node with a timestamp already equal to $t_f$.

Second, a time-based heuristic is introduced in the nearest neighbor selection; only neighbors with a timestamp lying in a specified time range are eligible to be considered in the nearest neighbor calculation. This time range is initialized to $[t_{\text{root}}, t_{\text{root}} + t_{\text{increment}}]$, and the $k$-nearest neighbors inside this time range are considered for feasibility check. If none of them leads to the creation of a feasible path, the time range is increased to $[t_{\text{root}} + t_{\text{increment}}, t_{\text{root}} + 2 \times t_{\text{increment}}]$, and so on, until a feasible path is found, or the time range reaches $t_f$, in which case the sample is ignored, and a new sample is taken. Simulation results have suggested that the use of such heuristics can result in better approximation of the natural

expected paths of the $\mathcal{OV}s$.

Finally, we 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. Future work will consider the reachability set completeness problem in further detail.

## V. Experimental Results

This section presents experimental results which validate the effectiveness of the TAM in assisting human drivers approaching intersections with errant drivers, subject to real-world uncertainty. These results verify that the TA can accurately identify drivers who do not observe a stop sign and enter the intersection out of turn, constituting a violation of the rules of the road. The results also show that the IP is capable of providing the human driver with the correct course of action, based on the intention and likely paths of the errant driver.

Including a human driver in the experiment helps to validate the TAM, since it will ultimately be used to assist human drivers. For example, the response to TAM alerts can be measured to determine the appropriate notification time. The autonomous driver can also be used to emulate a variety of human driving behaviors acquired from actual urban traffic data within the testbed.

The hardware results are presented after an overview of the experimental infrastructure, including testbed, hardware, and software. A video showing several different experimental scenarios, including the results below, is available at `http://acl.mit.edu/ITSC10TAM.mov`.

### A. Testbed

Hardware demonstrations were performed in the Real-time indoor Autonomous Vehicle test ENvironment (RAVEN), a testbed designed for the rapid prototyping of decision-making and control algorithms for unmanned aerial and ground vehicles [10]. Vehicle state data is collected using motion-capture cameras [21], which detect each vehicle via a unique pattern of attached reflective dots. Many different configurations can be tested with minimal setup effort, yielding high-fidelity state data for potentially dangerous driving scenarios without risk of injury.

We have constructed a representative road network within the RAVEN testbed, including multiple intersection types and road signs, to emulate a real-world driving environment for testing the TAM (Figure 3). The road network is $11.2 \times 5.5$ m$^2$ in size, and is capable of accommodating multiple intersection types and as many as 10 vehicles running simultaneously.

### B. Hardware Infrastructure

All vehicles in this experiment use the iRobot Create platform [22]. The vehicle is a 2-wheeled, skid-steered vehicle with a maximum speed of 0.5 m/s and near-instantaneous acceleration. A software wrapper imposes rate limits in acceleration and wheel speed differences, such that the vehicle emulates traditional automative dynamics (7).

Each experiment consists of one human-driven vehicle and one autonomous vehicle. The human-driven vehicle is controlled through a wireless steering wheel [23], including acceleration and brake pedals. The mapping of the steering and pedal inputs to the vehicle motion has also been tuned to emulate traditional control of an automobile, including turning, acceleration, and braking behaviors. The human driver may steer their vehicle through direct visual inspection of the testbed, or via a "virtual dashboard" interface which simulates the first-person driving perspective (Figure 4). Two cameras mounted onboard the human-driven vehicle provide real-time visual feedback in both the forward and rear directions (Figure 3), while a world map "GPS" guides the driver through the desired sequence of navigation waypoints. The autonomous vehicle is controlled through an off-board wrapper (Section V-C).

### C. Software Implementation

The TAM is implemented across three software modules for the host vehicle. The *Intention Predictor* classifies each driver according to the observed behaviors. The *Threat Assessor* determines the likely paths of those vehicles based on the perceived behaviors. Finally, if a vehicle poses a threat, the *Escape Path* module calculates suitable evasive trajectories for the human driver and offers the best course of action given the errant driver's paths. A vehicle is perceived to be a threat if three conditions are satisfied: (1) the IP module has classified the vehicle's behavior as errant; (2) the vehicle is sufficiently close to the host vehicle (within the black circle in Figure 5); and (3) both vehicles are approaching the same intersection.

The Intention Predictor module accesses observations of the states of each vehicle on the road network and provides these inputs to the classifier. Depending on the high-level behaviors, the classifier determines the most probable intent of each vehicle in the environment. In this paper, classification is performed on an autonomous vehicle; the same classification process is performed on a human-driven vehicle in Ref. [11].

The Threat Assessor module (Algorithm 1) predicts paths for each vehicle, based on the perceived intentions, using the Compute-Intention-Reachability module (Algorithm 2). This latter module is embedded in the external planner as a separate thread which runs asynchronously for each vehicle $\mathcal{OV}_i$ in the environment. Over fixed time intervals (1 s), the algorithm grows a new reachability tree based on the current $\mathcal{OV}$ position in the testbed. At the end of each time interval, the reachability tree is relayed to the host vehicle thread, which then calls the Escape Path algorithm (Section IV-B) if the threat conditions are met.

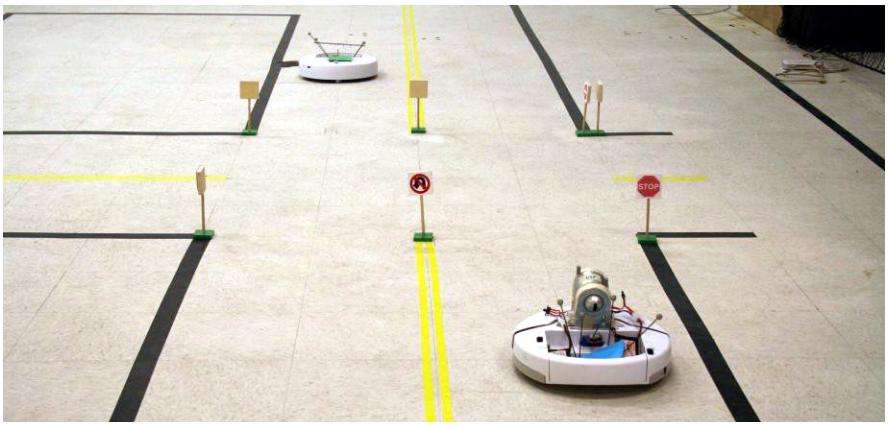


Fig. 3. The road network constructed within the RAVEN testbed to test the TAM algorithm. Here both a human-driven vehicle (front-right) and autonomous vehicle (back-left) are approaching a four-way, stop-sign intersection. The rear-mounted camera, one of two, is clearly visible on the human-driven vehicle.

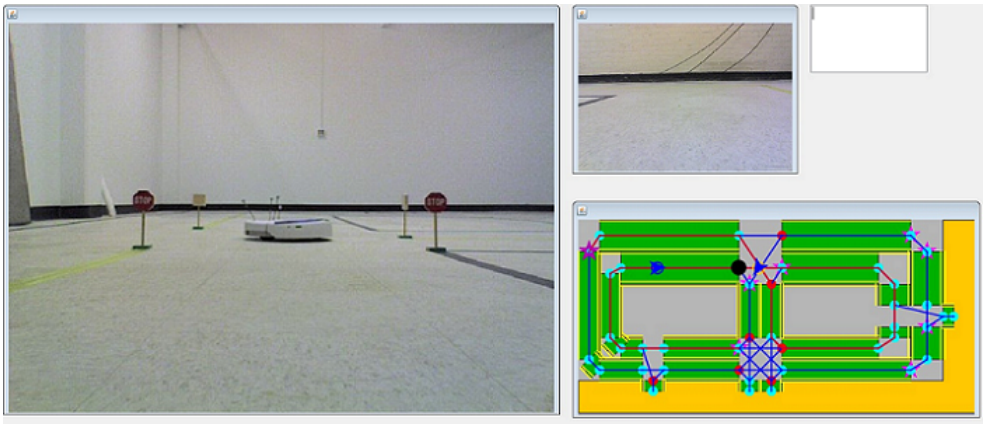


Fig. 4. Human interface for the human-driven vehicle, including real-time video feeds of the forward (left) and rear (top-center) directions of the vehicle, a dialog box (top-right), and a world map "GPS" guiding the driver to their next waypoint.

The Escape Path module (Algorithm 3) generates evasive maneuvers by simulating multiple possible actions of the human driver for sufficiently long time intervals (here 6 s), and determining whether and/or when the resulting trajectories enter the time-parameterized reachable set of the errant vehicle. In the current implementation, three possible actions are evaluated for the human driver – maintaining current speed, maximum acceleration (up to 0.4 m/s), or maximum braking. Escape paths are measured against the intention-reachability paths to determine the time to a potential collision, with the minimum-threat escape maneuver identified via (5). If the best escape maneuver requires acceleration or braking, the human driver is alerted to do so; no alert is provided if the driven can safely maintain their current speed over the prediction time horizon.

The autonomous vehicle is controlled via an offboard, multi-threaded Java implementation of the standard CL-RRT algorithm [20]. This is the same framework as used for autonomous vehicles in Ref. [11], but without the threat-aware components; only the human-driven vehicle is assessing the threat level of other vehicles. Instead, the autonomous agent simply treats other vehicles as static obstacles to avoid. The vehicle is controlled via pure pursuit steering control [24] and proportional-integral speed control. When approaching each intersection, a weighted coin flip within the autonomous vehicle logic determines whether or not the vehicle will properly decelerate and come to a stop.

A top-level navigator module is used in both vehicles' software packages to provide a sequence of geographic waypoints. The navigator module leverages a sparse representation of the RAVEN road network, constructed using the Route Network Data File (RNDF) specification from the 2007 DGC [25]. This representation includes the location, size, and connectivity of lane segments and "zones" (*e.g.,* parking lots); intersections are implicitly defined by the exit-entrance connections between lane/zone endpoints. An $A^*$ implementation is used to select the shortest-distance waypoint path in the graph between the current and desired waypoints, respecting lane directionality constraints [7]. As the vehicle (whether human or autonomous) approaches each intermediate or desired waypoint, the navigator selects the next waypoint on the list as the new target. An arbitrary list of desired waypoints is provided for each vehicle; the ultimate objective is to observe the vehicles' interactions at intersections. The diagram at bottom-right in Figure 4 shows the RNDF for the RAVEN road network, constructed from a simple .txt file. Figure 5 shows the display representation of the world used in the results below.

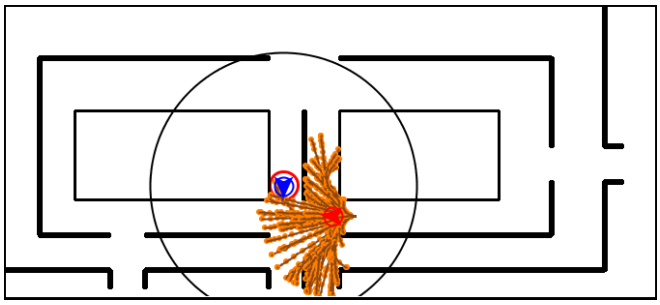


Fig. 5. Display representation of the RAVEN road network, including the host vehicle (blue chevron; detection radius in black), autonomous vehicle (red chevron), and reachability tree (brown edges, orange nodes). In this figure, the human driver has been alerted to stop to avoid an imminent collision. See Figure 6 for a full legend.

Given the geographic waypoint sequence, each escape path is constructed through forward simulation of the vehicle model connecting these waypoints, in a manner similar to the RRT algorithm.

### D. Results

The following results demonstrate the TAM providing a human driver with feedback during a simulataneous intersection approach with another driver. In this hardware demonstration, both the human-driven vehicle and autonomous vehicle are navigating through the RAVEN road network simulataneously, following a random sequence of waypoints. While the human driver is to respect all rules of the road, the autonomous vehicle may choose to not stop at an intersection and instead enter it out of turn (Section V-C). In this case, the TAM should identify the autonomous driver as errant based on observed behaior, and use the reachability tree to alert the human driver if a corrective action is necessary.

Figure 6 shows four successive timesteps, or phases, of a representative scenario where the human driver interacts with an autonomous driver at one of the RAVEN intersections. Figure 7 shows the threat levels incurred by executing either the "same speed," "accelerate," or "stop" escape maneuvers at each. In this scenario, both the human and autonomous vehicles are attempting to enter the southbound lane of the intersection: the human driver is approaching from the left and making a right turn, while the autonomous vehicle is approaching from the right and making a left turn.

The human driver reaches the intersection first, and has right-of-way to proceed into the intersection (Figure 6(a)). The autonomous driver behavior is classified by the IP as normal, and the TAM does not post any warnings because the autonomous vehicle is outside the host vehicle's detection radius. After stopping for several seconds, the human driver accelerates into the intersection, anticipating that the autonomous driver will decelerate. However, the IP observes the autonomous driver is actually accelerating into the intersection, and classifies the driver as errant (Figure 6(b)). The TA module generates the autonomous driver's reachability tree, and the escape path module determines the threat of either proceeding at the same speed, accelerating, or stopping. Since stopping minimizes the threat level (Figure 7),

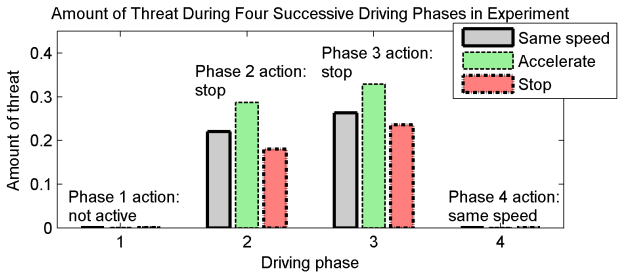



Fig. 7. Threat levels for the possible escape paths during each phase of Figure 6.

the TAM alerts the human driver to stop. (Even though the reachability tree does not reach the human-driven vehicle, the escape paths—which are not shown—do intersect with the set.) As the autonomous driver continues to move, the human driver stops within a few seconds, in response to the TAM's recommendation (Figure 6(c)). Once the errant driver clears the intersection, the TAM determines that all actions are safe, causing the alert to switch off (Figure 6(d)).

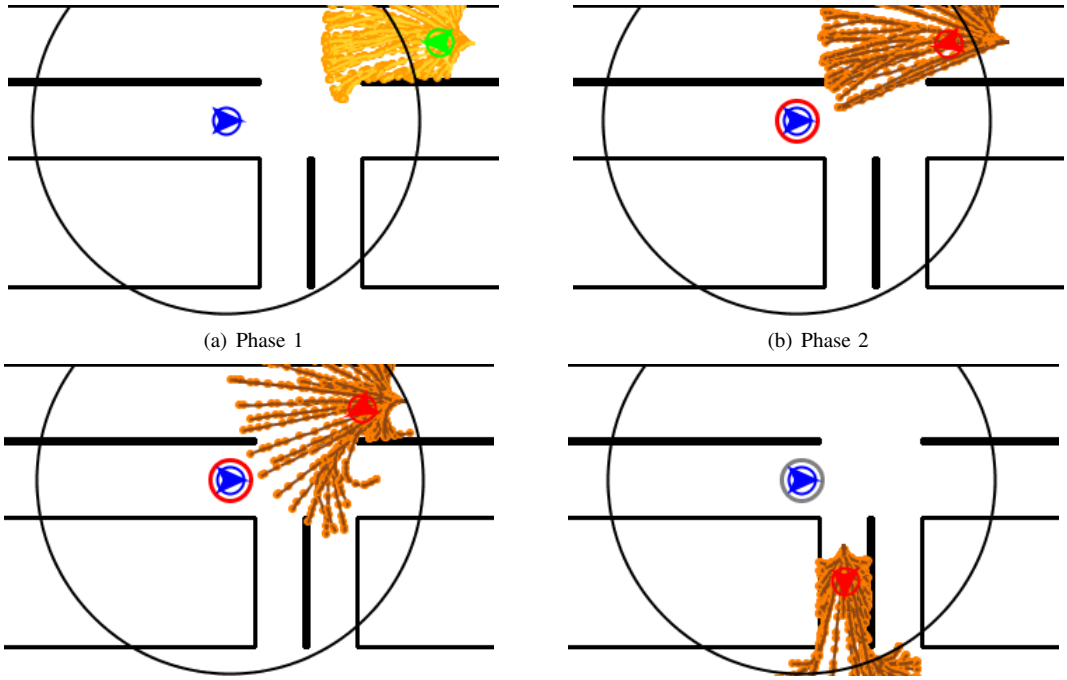
## VI. Conclusions and Future Work

This paper introduced a new approach for human driver assistance planning at road intersections. It is based on combining an intention predictor (IP), threat assessor (TA), and escape paths to create a threat assessment module (TAM) which runs efficiently and is suitable for real-time implementation. The IP performs classification on vehicle state information to determine threat levels via high-level logic, including the Ecological Recogniser, support vector machines, and Bayesian filtering. The TA performs efficient sampling-based reachability computations, via RRT, to identify possible future paths of surrounding vehicles. Finally, the escape paths are evaluated to maximize time to collision and thus minimize the threat level. The TAM has been demonstrated and validated in the RAVEN testbed, using both human-driven and autonomous vehicles.

Future experiments will increase both the variety and complexity of interactions between vehicles at intersections, including different intersection types. We will consider multi-vehicle intersection interactions, simulating multiple drivers at busy intersections. The algorithm will be tasked to simultaneously classify behaviors of multiple vehicles, build reachability trees, construct escape paths, and post alerts, all in real time. Additionally, future work will study human drivers' response times to different alerts, so that the response time can be appropriately factored into the alert generation.

## Acknowledgements

Research funded by Ford Motor Company and Le Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) Graduate Award. The authors would like to thank Dr. Tom Pilutti, Professor John Leonard, Dr. Luke Fletcher, Vishnu Desaraju, Alejandro Dos Reis, and Kevin Kleinguetl for their contributions.



(a) Phase 1 (b) Phase 2

(c) Phase 3 (d) Phase 4

Fig. 6. Zoomed-in display view of a representative interaction between the human-driven host vehicle (blue chevron) and autonomous vehicle at an intersection; each figure shows a successive snapshot of this interaction. The autonomous vehicle is classified in real-time as safe (green chevron) or errant (red chevron). If the autonomous vehicle is perceived to be a threat, the reachability tree darkens, and the outer ring around the host vehicle indicates the alert type (grey = maintain speed, red = brake, green = accelerate). Escape paths are not shown.

## REFERENCES


[1] W. D. Jones, "Keeping cars from crashing," *IEEE Spectrum*, vol. 38, no. 9, pp. 40–45, 2001. [Online]. Available: http://dx.doi.org/10.1109/6.946636

[2] National Highway Traffic Safety Administration, "Fatality analysis reporting system encyclopedia," 2010. [Online]. Available: http://www-fars.nhtsa.dot.gov/Crashes/CrashesLocation.aspx

[3] B. Bougler, D. Cody, and C. Nowakowski, "California Intersection Decision Support: A Driver-Centered Approach to Left-Turn Collision Avoidance System Design," UC Berkeley, Tech. Rep., 2008.

[4] Y. Zhao and A. House, "Vehicle Location and Navigation Systems: Intelligent Transportation Systems," *Artech House*, pp. 221–224, 1997.

[5] J. Hillenbrand and K. Kroschel, "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*, 2006, pp. 1–6.

[6] K. Fuerstenberg, "New European Approach for Intersection Safety - The EC-Project INTERSAFE," in *Proceedings of IEEE Intelligent Transportation Systems*, 2005, pp. 432–436.

[7] J. Leonard, J. How, S. Teller, *et al.*, "A Perception-Driven Autonomous Urban Vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727 – 774, 2008.

[8] A. Polychronopoulos, M. Tsogas, A. Amditis, U. Scheunert, L. Andreone, and F. Tango, "Dynamic situation and threat assessment for collision warning systems: the EUCLIDE approach," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2004, pp. 636–641.

[9] R. Karlsson, J. Jansson, and F. Gustafsson, "Model-based statistical tracking and decision making for collision avoidance application," in *Proceedings of the IEEE American Control Conference*, vol. 4, 2004.

[10] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[11] G. S. Aoude, B. D. Luders, D. S. Levine, and J. P. How, "Threat-aware Path Planning in Uncertain Urban Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010 (to appear).

[12] T. Kumagai, Y. Sakaguchi, M. Okuwa, and M. Akamatsu, "Prediction of driving behavior through probabilistic inference," in *Proceedings of the Eighth International Conference on Engineering Applications of Neural Networks*, 2003, pp. 8–10.

[13] G. S. Aoude and J. P. How, "Using Support Vector Machines and Bayesian Filtering for Classifying Agent Intentions at Road Intersections," Massachusetts Institute of Technology, Tech. Rep. ACL09-02, September 2009. [Online]. Available: http://hdl.handle.net/1721.1/46720

[14] C. Heinze, "Modelling Intention Recognition for Intelligent Agent Systems," Ph.D. dissertation, Melbourne, Victoria: Defence Science and Technology Organisation, 2004.

[15] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[16] H. M. Mandalia and D. D. Dalvucci, "Using Support Vector Machines for Lane-Change Detection," *Human Factors and Ergonomics Society Annual Meeting Proceedings*, vol. 49, pp. 1965–1969, 2005.

[17] G. S. Aoude, B. D. Luders, and J. P. How, "Sampling-Based Threat Assessment Algorithms for Intersection Collisions Involving Errant Drivers," in *IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italy, September 2010.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Ames, IA, Tech. Rep. 98-11, Oct. 1998.

[19] ——, *Planning Algorithms*. Cambridge University Press, 2006.

[20] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, September 2009.

[21] Vicon, "Motion capture systems from Vicon," 2010. [Online]. Available: http://www.vicon.com/

[22] iRobot, "iRobot: Education & research robots," online http://store.irobot.com/shop/index.jsp?categoryId=3311368.

[23] Microsoft, "Description of the Xbox 360 wireless racing wheel," 2010. [Online]. Available: http://support.microsoft.com/kb/927344?sd=xbox

[24] S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov stability of a nonlinear path-following guidance method," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1718–1728, November-December 2007.

[25] DARPA, "Urban challenge: Route network definition file (RNDF) and mission data file (MDF) formats," Defense Advanced Research Projects Agency, Tech. Rep., March 2007. [Online]. Available: http://support.microsoft.com/kb/927344?sd=xbox