Threat-aware Path Planning in Uncertain Urban Environments

Georges S. Aoude, Brandon D. Luders, Daniel S. Levine, and Jonathan P. How

***Abstract*— This paper considers the path planning problem for an autonomous vehicle in an urban environment populated with static obstacles and moving vehicles with uncertain intents. We propose a novel threat assessment module, consisting of an intention predictor and a threat assessor, which augments the host vehicle's path planner with a real-time threat value representing the risks posed by the estimated intentions of other vehicles. This new threat-aware planning approach is applied to the CL-RRT path planning framework, used by the MIT team in the 2007 DARPA Grand Challenge. The strengths of this approach are demonstrated through simulation and experiments performed in the RAVEN testbed facilities.**

I. Introduction

Whether driving on highways or navigating in the middle of a battlefield, intelligent vehicles must be able to quickly and robustly compute motion plans in very uncertain worlds. The sources of this uncertainty may be internal, i.e., incomplete or imperfect knowledge of the vehicle model, or external, i.e., incomplete or imperfect knowledge of the environment, and may be present either in sensing or in predictability [1]. This paper addresses problems involving uncertainty in predictability, and in particular the intentions of the other vehicles within our vehicle's world.

To make meaningful predictions of other vehicles' intentions, a smart vehicle should be able to gather information from the environment to build models approximating those intentions. It is not safe to assume that all drivers obey all "rules of the road," as many collisions occur when this is not the case, but classifying all other vehicles as hostile would be overly conservative. The information gathered by our vehicle might include onboard camera images, radar-based measurements of surrounding objects, or messages intercepted or shared on some communication channels.

The vehicle planner must be able to address the uncertainty in each vehicle's future state and actions, or at a higher level, the behavior / intent governing those actions. Existing path planners typically rely on *a priori* information to generate trajectories, applying reactive maneuvers or replanning as needed to correct the path online and maintain feasibility. However, uncertainty in object predictability is typically not considered explicitly in the global path planner. Instead, moving vehicles are often assumed to follow known trajectories or policies, which the planner reacts to locally. While full probabilistic representation of the environment is possible, such as with POMDPs [2], [3], the corresponding solution techniques are computationally intractible for real-time path planning problems of even modest complexity or dimension. Recently, sampling-based techniques such as rapidly-exploring random trees (RRTs) [4] have been extended to incorporate motion patterns for moving obstacles learned offline via Gaussian processes [5]. However, these patterns may not be sufficient to capture all possible unexpected behaviors, which are usually the main reason behind collisions.

The main motivation of this work is the set of challenges faced by autonomous vehicles in the 2007 DARPA Grand Challenge (DGC) [6], which involved navigating an outdoor urban environment in the presence of other vehicles while obeying all traffic regulations. One of the main challenges of this race was negotiating traffic intersections, where several vehicles were involved in collisions or near-collisions. Several explanations have been offered for these occurrences, but the one that motivated this work is the inability of the autonomous vehicles to anticipate the intent of other vehicles [7]. With some knowledge of those intentions, the motion planner could incorporate the risk posed by those vehicles when considering potential trajectories, improving safety.

This paper introduces a new framework for autonomous vehicles which enables path planning algorithms to explicitly incorporate both obstacle uncertainty and the corresponding risk posed to the vehicle. A threat assessment module, consisting of an intention predictor and a threat assessor, augments the host vehicle's path planner with a real-time threat value for each potential trajectory, reflecting the risks posed by the estimated intentions of other vehicles. The strengths of this approach are demonstrated through simulation and experiments performed in the MIT RAVEN facility [8].

II. Problem Statement

Consider the autonomous system denoted by $\mathcal{HV}$,

$$\dot{s}(t) = f(s(t), u(t), t), \tag{1}$$

where s is the state and u is the control input; we have that s is constrained to the state space $\mathcal{S}$, $s \in \mathcal{S}$, while u is contstrained to the control space $\mathcal{U}$, $u \in \mathcal{U}$. Our objective for safe motion planning is to minimize some desired cost functional over the duration of the motion,

$$L = \Psi(s(t_f), t_f) + \int_{t_0}^{t_f} \Gamma(s(t), u(t), t)dt, \tag{2}$$

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

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

subject to the vector of component-wise constraints

$$\boldsymbol{C}(s(t), u(t), t) \leq \mathbf{0}. \tag{3}$$

Of the many constraints $\boldsymbol{C}$ might include, we are interested in those necessitating collision avoidance with N other vehicles, denoted here as $\mathcal{OV}_i$, $i = 1 \dots N$. These can typically be written in the form

$$\min_{t_0 \leq t \leq t_f} \|p_s(t) - p_{r_i}(t)\| > \epsilon \quad \forall i = 1 \dots N, \tag{4}$$

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 $r_i(t)$, ϵ is the minimum allowed distance between the $\mathcal{HV}$ and the $\mathcal{OV}s$, and $t_f = t_0 + T_h$ where T_h is the time horizon of interest. 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 Ω_i,

$$\Omega_i = \{p_s(t) \mid \|p_s(t) - p_{r_i}(t)\| \leq \epsilon\}. \tag{5}$$

In uncertain and/or non-cooperative environments, $r_i(t)$ is typically not available for the $\mathcal{HV}$. For $\mathcal{HV}$ to maintain guaranteed safety, some mechanism must be able to identify the input sequence $u(t)$ which yields a feasible path for all possible realizations of $r_i(t)$, $i = 1 \dots N$. In the event that no such path exists, the same mechanism should select $u(t)$ in order to minimize some threat level. Here we define the threat level $\mathcal{T}_i$ for vehicle i as inversely proportional to t_{c_i}, the earliest possible time of collision between $\mathcal{HV}$ and $\mathcal{OV}_i$:

$$\mathcal{T}_i = \frac{1}{t_{c_i}}, \quad t_{c_i} = \inf\{t \mid (p_s(t), p_{r_i}(t)) \in \Omega_i\}. \tag{6}$$

Equation (4) thus cannot typically be guaranteed at all times, so this constraint is instead converted to a penalty in the objective function. We define the new cost functional

$$J = L + w \cdot \sup_{i \in 1 \dots N} \mathcal{T}_i, \tag{7}$$

where L is defined in (2), $\mathcal{T}_i$ is the threat level defined in (6), and w is a user-selected weighting factor representing the tradeoff between safety and optimality. The constraints (4) are then removed from (3), yielding

$$\bar{\boldsymbol{C}}(s(t), u(t), t) \leq \mathbf{0}. \tag{8}$$

Problem Definition (Threat-Aware Dynamic Motion Planning): Given a state space $\mathcal{S}$, a control space $\mathcal{U}$, an initial state $s_0 \in \mathcal{S}$ and a final state $s_f \in \mathcal{S}$, compute the control input sequence $u(t)$, $t \in [0, t_f]$, $t_f \in [0, \infty)$ that minimizes (7) subject to the constraints (1) and (8).

III. Threat-Aware Path Planning

Several global path planners, including sampling-based planners, use the philosophy that collision detection should be done in a "black box" which decouples the host vehicle's path planning from any specific geometric or kinematic obstacle models [9]. This paper proposes that threat assessment should be done in a similar "black box" framework, providing planners with a quantitative threat metric to identify safer paths during trajectory generation. This threat assessment considers the unknown intentions of moving $\mathcal{OV}s$, which may present a collision risk for the $\mathcal{HV}$.

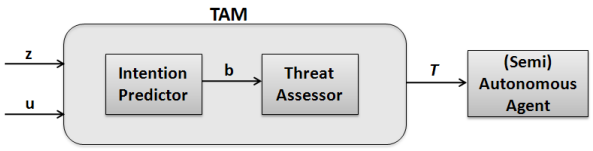


Fig. 1. High-level architecture of the TAM. The inputs of the TAM are the measurement history z of the $\mathcal{OV}s$ and the candidate control sequence u generated by the $\mathcal{HV}$ planner. The IP computes the intention vector b for each $\mathcal{OV}$, used by the TA to identify the threat level $\mathcal{T}$.

A. High-level TAM Architecture

The proposed architecture features a threat assessment module (TAM, Figure 1), consisting of an intention predictor (IP) and a threat assessor (TA). The IP uses observations taken by the $\mathcal{HV}$ of the $\mathcal{OV}s$ to make predictions of their intentions. Instead of using low-level reasoning to predict future *trajectories*, the IP uses higher-level logic and learning to provide the planner with predicted future *intentions*, key to earlier and more accurate prediction of future collisions. The TA then converts this prediction into a set of potential paths for the $\mathcal{OV}$, and returns a threat value that a candidate trajectory $u(t)$ (or equivalently, $s(t)$), $t \in [t_0, t_f]$ for $\mathcal{HV}$ incurs in the vicinity of these potential paths. As a result, the planner generates paths that are threat-aware by construction, thus improving the vehicle's reactive behavior.

Several distinct representations have been proposed to model intentions for humans and autonomous systems; here we adopt the Ecological Recogniser architecture introduced by [10]. Using the language of [10], the $\mathcal{HV}$ is the recognising agent, while each $\mathcal{OV}$ is an intending agent. We follow the abstract definition that these intentions are directly responsible for the actions executed by the intending agents. Intentions may be desired plans (*e.g.,* following a straight path) or other high level intentional states (*e.g.,* not following the rules of the road). Let the M-vector $\boldsymbol{b}_i$ denote the intention vector of $\mathcal{OV}_i$; the j^{th} entry b_i^j corresponds to the belief $\mathcal{OV}_i$ is operating under the jth intention. The cost functional (7) is then modified to embed the intention information,

$$J = L + w \cdot \sup_i \sum_j b_i^j \mathcal{T}_{ij}, \tag{9}$$

where $\mathcal{T}_{ij}$ is the threat value of the trajectories followed by $\mathcal{OV}_i$ as a realization of its j^{th} intention; note that $i \in \{1 \dots N\}$ and $j \in \{1 \dots M\}$.

B. Agent Model

Each agent is modeled using the standard bicycle model

$$\begin{aligned} \dot{x} &= v\cos(\theta), \quad & \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= \frac{v}{L}\tan(\delta), \quad & \dot{v} &= a, \end{aligned} \tag{10}$$

where (x, y) is the rear axle position, v is the forward speed, θ is the heading, L is the wheelbase, a is the forward acceleration, and δ 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 $|\delta| \leq \delta_{\max}$. Each admissible control sequence $u : [0, t_f] \mapsto \mathcal{U}$ is piecewise constant.

In the work that follows, we make several assumptions to constrain the focus on the 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 a detection radius (Section V-C), and that all sensor measurements are noise-free. Additionally, the path planner is provided with a complete map of the environment *a priori*, excluding any dynamic obstacles or agents.

IV. Developed Algorithms

This section describes the implementation of the IP and TA components of the TAM and the resulting threat-aware path planner, using closed-loop rapidly exploring random trees (CL-RRT), into which they are embedded.

A. Intention Predictor (IP)

The Ecological Recognizer architecture [10] that the IP adopts has two key components: a pattern matcher and a reasoning module. The pattern matcher is a classifier that is trained offline to recognise different possible intentions by observing agent states, and operates online by giving a continuous 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 $\boldsymbol{b}$.

Here we are interested in estimating the intentions of human vehicles in urban environments, specifically near intersections. This Ecological Recognizer implementation is based on an approach previously developed and demonstrated by the authors in simulation [11] for classifying agent intentions using support vector machines (SVM) and Bayesian filtering (BF). The human driver intention classification problem is very complex, due to the various nuances of human behaviors. SVM is suitable for the IP because it has been shown to be both robust and efficient for classification problems [12], such as lane-change detection [13].

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 $\boldsymbol{b}$. SVM-BF also includes a threshold detector, such that the final vector $\boldsymbol{b}$ is a unit basis vector specifying the most likely intention according to the threshold value [11]. Based on experimenting with different combinations of kernel functions and features, the best results were obtained using the Gaussian radial basis function and combining the following three features: 1) the *relative distance* between the $\mathcal{OV}_i$ and the entrance of the intersection it is approaching, 2) the *speed* of the $\mathcal{OV}_i$, and 3) 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 detection radius, and both vehicles are approaching the same intersection.

B. Threat Assessor (TA)

The threat-aware CL-RRT algorithm (Section IV-C) incorporates the computation of a threat value for each candidate trajectory it generates, given knowledge of the current $\mathcal{OV}$ states. We propose an approach, detailed in Algorithm 1, that combines a fast sampling-based reachability method with intention prediction information provided by the IP (Section IV-A) to efficiently estimate the threat level. The threat assessor has a finite time horizon, limiting the $\mathcal{HV}$ lookahead horizon for possible future $\mathcal{OV}$ paths, in order to focus computation on imminent threats. The choice of horizon length is domain-specific, but should allow the $\mathcal{HV}$ sufficient time to react in a dynamic environment.

Algorithm 1 Intention-based Threat Assessment

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

Algorithm 2 Compute-Threat (*Reach-Set*, $\mathcal{OV}$, e)

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

The Threat-Assessor algorithm (Algorithm 1) begins by calling the Compute-Intention-Reachability (CIR) subroutine, discussed below, 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 2 is called to compute the threat incurred by the $\mathcal{HV}$ trajectory in the region reached by the paths corresponding to e_j. Note that the earliest time to collision is converted into a threat value using (6) (line 3 of Algorithm 2). This value is weighted by the probability b_i^j provided by the IP. Finally, the threat is computed as the maximum value of all threats created by each $\mathcal{OV}_i$ (line 8 of Algorithm 1).

The CIR algorithm (detailed in [14]) is also based on the CL-RRT algorithm, but uses the resulting tree only for simulation of the different possible paths of a $\mathcal{OV}$ (which is not controlled by the operator). No best path is selected, but rather the entire tree is biased towards regions corresponding to the learned intentions of the $\mathcal{OV}s$. This biasing is achieved by devoting a portion of the tree samples to regions corresponding to the intention e_j, with remaining samples being taken uniformly throughout the environment. The algorithm also includes time-parametrization extensions (introduced in [15]) that tailor the RRT algorithm to the efficient computation of intented paths by the $\mathcal{OV}s$.

C. Threat-Aware CL-RRT Planner

In this section, we present a global threat-aware planner which builds on the CL-RRT implementation developed for MIT's Talos vehicle in the 2007 DGC competition [16]. The CL-RRT algorithm extends the rapidly-exploring random tree (RRT) algorithm [4], [9], which grows a tree of dynamically

feasible trajectories by randomly sampling points toward which the tree is extended. The CL-RRT algorithm adds a path-tracking control loop in the vehicle prediction model, such that sampling takes place in the reference input space rather than in the vehicle input space. The algorithm thus maintains the exploration bias of traditional RRT algorithms, while allowing for generation of smooth trajectories more efficiently. Furthermore, because the RRT algorithm is sampling-based, the quality of the planning tree and resulting paths are scalable with the available computational resources. Please refer to [16] for a detailed description of the CL-RRT planner, and [6] for an explanation of the planner integration in the Talos system architecture.

The proposed threat-aware planner is shown in Algorithm 3. The main addition is the threat computation in lines 6 and 10. This computation calls Algorithm 1 to efficiently calculate the threat of colliding with the possible paths of the other $\mathcal{OV}s$ (Section IV-B). Whereas the original CL-RRT heuristic identifies nearest nodes based on path duration (in the Dubins sense), we extend this heuristic to also include the threat value (line 6). In line 10, the threat along the edge to a new node is computed and stored at the new node. The best path is then chosen in accordance with the cost defined in (9) (line 15 of Algorithm 3), which includes both the total time to the goal and the threat along the trajectory. These modifications embed the threat computation in the CL-RRT trajectory generation, leading to the selection of safer paths which are threat-aware by construction.

Another related addition is the use of threat propagation logic. First, the root node is initialized to have zero threat. When adding a new node, if the path from the selected node (initially the root) to the new node incurs a non-zero threat (computed using Algorithm 1), then an intermediate node is created at the location of the earliest collision, and both the intermediate node and the new node inherit the computed threat value. Note that if a node with a non-zero threat value is chosen to be expanded, the new node inherits that value and no new threat computation is required. The rationale is that the threat value does not decrease along an edge connecting a node with non-zero threat to its child (recall (6)). A useful property of the threat heuristic is that it is admissible: it underestimates the "true" cost. This is due to the sampling-based nature of the CIR algorithm, which underapproximates the earliest time of collision, as well as the assumption that children nodes inherit the threat value of their parent if the value is non-zero. Before choosing a best path, the threat values of the tree are updated from the bottom up (*i.e.,* starting from the leafs) by reassigning each parent's threat value to be the minimum of the threat of its children, and so on, until the root is reached.

Algorithm 3 Threat-Aware CL-RRT

```
 1: repeat
 2:     Measure current vehicle states and environment
 3:     Propagate states by computation time limit
 4:     repeat
 5:         Generate sample for the input to controller
 6:         Sort nodes in tree using threat and time heuristics
 7:         for each sorted node do
 8:             Form controller input by drawing line from node to sample,
                then propagate
 9:             if propagated portion is collision free then
10:                 Compute threat of propagated portion (Algorithm 2)
11:                 Add sample to tree break
12:             end if
13:         end for
14:     until time limit is reached
15:     Choose best path and repropagate from current states
16:     if repropagated trajectory is infeasible then
17:         Remove infeasible portion from tree and go to line 15
18:     end if
19:     Send best path to controller
20: until vehicle reaches target
```

V. Experimental Results

This section presents experimental results which validate the effectiveness of the threat-aware CL-RRT algorithm, augmented with the TAM, in enabling an autonomous vehicle to successfully identify and avoid an errant human-driven vehicle under real-world uncertainty. These results focus on the challenging task of avoiding collisions at intersections, in particular with errant drivers who do not observe a stop sign and enter the intersection out of turn.

First, a brief overview of the experimental infrastructure is provided. Next, we present a subset of hardware results showing the interaction between an errant human driver and an autonomous vehicle at intersections; additional experiments can be viewed in the video attachment or at `http://acl.mit.edu/IROS10TAM.mp4`. Further experiments and infrastructure details are given in [14].

A. Hardware

Hardware demonstrations were performed in the Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) [8], a testbed which uses motion-capture cameras to provide high-fidelity vehicle state data. We have constructed a representative road network within the RAVEN testbed, including many intersection types and road signs, to emulate a realistic driving environment (Figure 2).

All vehicles in this experiment use the iRobot Create platform [17], a skid-steered vehicle, modified with a software wrapper to emulate traditional automotive steering (10). Each experiment consists of 1 or 2 autonomous vehicles and 1 human-driven vehicle. The autonomous vehicles are controlled through an off-board wrapper which receives waypoints from the threat-aware CL-RRT software (Section V-B) and steers using pure pursuit [18]. The human-driven vehicle is controlled through a wireless steering wheel [19], including acceleration and brake pedals. A human driver may steer their vehicle through simple visual inspection, or via an onboard dual-camera (Figure 2) interface which emulates the first-person driving perspective (Figure 3). This interface includes a real-time "GPS" display, which guides the driver through an intended sequence of waypoints (Section V-B).

B. Software

The planning software consists of two primary modules, discussed below. The navigator module accesses an abstract

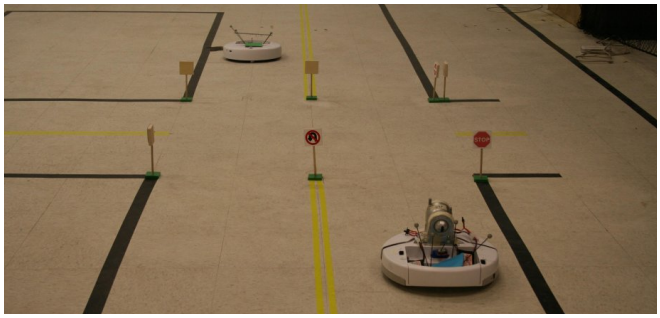


Fig. 2. The road network used for experiments. In this image, both a human-driven vehicle (front-right) and autonomous vehicle (back-left) are approaching a four-way, stop-sign intersection.

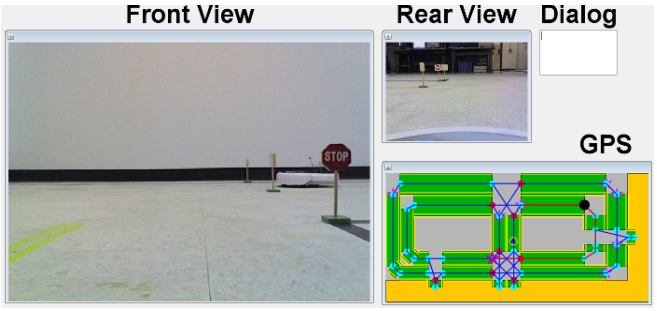


Fig. 3. Human interface for the human-driven vehicle, including real-time navigation and visual feedback.

representation of the road network and selects sequences of waypoints, yielding the current goal location for the vehicle. The CL-RRT motion planning module then uses local knowledge of the environment (e.g., obstacles, terrain, and other agents), including the TAM, to plan dynamically feasible paths to this goal.

The navigator module leverages a sparse representation of the RAVEN road network (Figure 4), constructed using the Route Network Data File (RNDF) specification from the 2007 DGC [20]. As the RNDF represents the traversability graph, 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 [6]. 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 threat-aware CL-RRT algorithm (Algorithm 3) is implemented as an offboard, multi-threaded Java application, designed to support arbitrary vehicle and environment models in both simulation and hardware. The CIR algorithm [14] is embedded in this offboard planner as a separate thread which runs asynchronously for each $\mathcal{OV}$. Over fixed time intervals (2s), the algorithm grows a new reachability tree based on the current $\mathcal{OV}$ 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 IV-B). Figure 4 shows the display representation generated by the application.

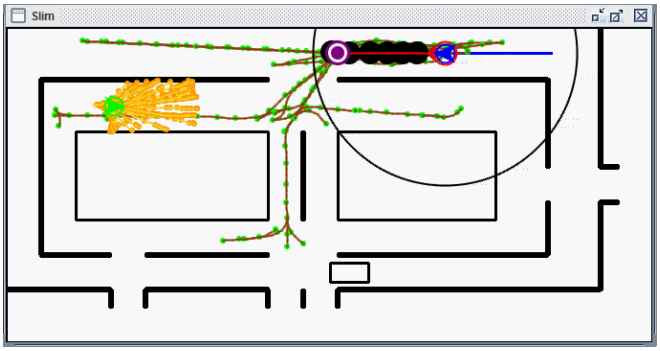


Fig. 4. Display view of the RAVEN road network for the host vehicle (blue chevron, path history in blue). The host uses the CL-RRT tree (brown edges, green nodes) to identify the best known path (red edges, black nodes) to the current waypoint (purple). This path should avoid all obstacles (black boxes), the human-driven vehicle (green circled chevron) and, if active, their reachability tree (yellow edges, orange nodes). The other autonomous vehicle is represented as the car-sized rectangle at bottom-center.

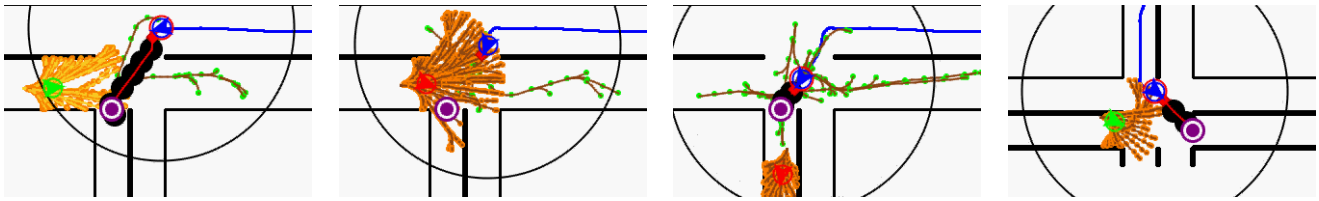
C. Results

The results below demonstrate the threat-aware closed-loop RRT algorithm in the context of stopping at intersections. In each experiment, one human-driven vehicle and one or two autonomous vehicles are navigating through the RAVEN road network simultaneously. The human driver may choose to not stop at an intersection and instead enter it out of turn, violating the rules of the road. The human driver's intent is classified as either good or errant, based on whether or not they correctly approach the intersections. The objective of the autonomous vehicle(s) is then to not only avoid all collisions, but also minimize the risk of collision with an errant human driver in intersections.

The human-driven vehicle is perceived to be a threat if three conditions are satisfied: (1) the IP module has classified the drivers' behavior as errant; (2) the errant driver is within the host vehicle's detection radius (Figure 4); and (3) both vehicles are approaching the same intersection. In this case, the autonomous vehicle is to avoid that vehicle's reachability tree constructed by the CIR algorithm; otherwise the reachability tree is not an active constraint.

Over the course of this experiment set, 20 intersection encounters occurred: 10 with one autonomous vehicle in the network, and 10 with two autonomous vehicles in the world. Of these encounters, only one resulted in a collision; in this case, the human-driven vehicle aggressively approached the autonomous vehicle, which had already stopped to avoid it.

Figure 5 demonstrates two particular scenarios in which the human-driven vehicle interacts with an autonomous vehicle at an intersection. In Scenario 1 (Figure 5(a)-(c)), both vehicles are attempting to enter the same lane, with the human-driven vehicle approaching from the left and making a right turn, and the autonomous vehicle approaching from the right and making a left turn. The autonomous vehicle arrives first and, observing (through the IP) that the human driver is decelerating, plans a path through the intersection (Figure 5(a)). After entering the intersection, however, the IP module detects the human driver accelerating, as if it will enter the intersection out of turn. The IP module classifies



(a) Scenario 1, Part 1 (b) Scenario 1, Part 2 (c) Scenario 1, Part 3 (d) Scenario 2

Fig. 5. Zoomed-in display view of two interactions between an autonomous vehicle (detection radius in black) and a human-driven vehicle in an intersection. Figures (a)-(c) show snapshots of a scenario in which the human-driven vehicle is classified as errant; in Figure (d), the human-driven vehicle is classified as good. Note that the reachability tree darkens if the driver is perceived to be a threat.

this behavior as errant, and thus the TA module seeks to minimize the corresponding threat level $\mathcal{T}$. Though the autonomous vehicle cannot avoid the reachability tree's outer reaches, it does avoid the inner nodes (where the time to collision is lower) by quickly coming to a stop (Figure 5(b)). Once the errant driver is no longer a threat, the autonomous agent finishes crossing the intersection (Figure 5(c)).

In Scenario 2 (Figure 5(d)), the autonomous driver is making a left turn, while the human-driven vehicle is crossing the intersection (from left to right in the figure). Here, the human-driven vehicle has correctly come to a stop at the intersection, and the IP module has classified it as a good driver. Unlike Figure 5(b), the autonomous driver then proceeds through the intersection, perceiving that the human driver is likely to obey the rules.

VI. Conclusion

This paper has introduced a new approach for path planning in urban environments subject to uncertainty in the intentions of other vehicles. This work combines an intention predictor (IP) and a threat assessor (TA) to create a threat assessment module (TAM), embeddable in path planning algorithms, which runs efficiently and is suitable for real-time implementation. The IP uses high-level logic based on the Ecological Recogniser, while the TA uses sampling-based techniques to efficiently compute reachable future paths for surrounding vehicles. Using TAM applied to the CL-RRT planner, an autonomous vehicle can successfully identify and avoid an errant driver in realistic intersection scenarios, as demonstrated through hardware results.

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, Kenneth Lee, Vishnu Desaraju, Alejandro Dos Reis, and Kevin Kleinguetl for their contributions to this research effort.

References

[1] S. M. Lavalle and R. Sharma, "On motion planning in changing, partially-predictable environments," *International Journal of Robotics Research*, vol. 16, pp. 775–805, 1997.

[2] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington, "Planning under continuous time and resource uncertainty: A challenge for AI," in *AIPS Workshop on Planning for Temporal Domains*. Citeseer, 2002, pp. 91–97.

[3] A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte, "Parametric POMDPs for planning in continuous state spaces," *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 887–897, 2006.

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

[5] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 1056–1062.

[6] 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.

[7] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, and F. R. Kline, "The MIT - Cornell Collision and Why it Happened," *Journal of Field Robotics*, vol. 25, no. 10, pp. 775 – 807, Oct. 2008.

[8] 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.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

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

[11] 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

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

[13] 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.

[14] G. S. Aoude, B. D. Luders, K. K. H. Lee, and J. P. How, "Threat Assessment Design for Driver Assistance System at Intersections," in *IEEE Conference on Intelligent Transportation Systems*, Madeira, Portugal, September 2010.

[15] G. S. Aoude, B. 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.

[16] 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.

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

[18] 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.

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

[20] 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