Evasion Planning for Autonomous Vehicles at Intersections

Tsz-Chiu Au[1], Chien-Liang Fok[2], Sriram Vishwanath[2], Christine Julien[2], and Peter Stone[1]

***Abstract*— Autonomous intersection management (AIM) is a new intersection control protocol that exploits the capabilities of autonomous vehicles to control traffic at intersections in a way better than traffic signals and stop signs. A key assumption of this protocol is that vehicles can always follow their trajectories. But mechanical failures can occur in real life, causing vehicles to deviate from their trajectories. A previous approach for handling mechanical failure was to prevent vehicles from entering the intersection after the failure. However, this approach cannot prevent collisions among vehicles already in the intersection or too close to stop because (1) the lack of coordination among vehicles can cause collisions during the execution of evasive actions; and (2) the intersection may not have enough room for evasive actions. In this paper, we propose a preemptive approach that pre-computes evasion plans for several common types of mechanical failures before vehicles enter an intersection. This preemptive approach is necessary because there are situations in which vehicles cannot evade without pre-allocation of space for evasion. We present a modified AIM protocol and demonstrate the effectiveness of evasion plan execution on a miniature autonomous intersection testbed.**

I. INTRODUCTION

Recent robotic car competitions and demonstrations have convincingly shown that autonomous vehicles are feasible with current generation of hardware [1]. Looking ahead to the time when autonomous cars will be common, Dresner and Stone proposed a new intersection control protocol called *Autonomous Intersection Management* (AIM) and showed that by leveraging the control and network capabilities of autonomous vehicles it is possible to design an intersection control protocol that is much more efficient than traffic signals [2]. By removing the human factor from the control loop, autonomous vehicles, with the help of advanced sensing devices, can be safer and more reliable than human drivers. In addition, the AIM protocol exploits the fine control of autonomous vehicles to allow more vehicles to simultaneously cross an intersection, thus effectively reducing the delay of vehicles by orders of magnitude compared to traffic signals [3]. However, the assumption that vehicles can always be reliably controlled is unrealistic, since mechanical failures such as tire blowouts can happen in reality. Since a small mechanical error can result in a devastating collision, AIM is not *fail-safe*.

To remedy this problem, Dresner and Stone proposed a simple way to mitigate catastrophic failure, namely to notify vehicles when problems occur and prevent new vehicles from entering the intersection [4]. Upon receiving distress signals from vehicles or monitoring devices at an intersection, the intersection manager will promptly stop granting reservations to prevent vehicles from entering the intersection and broadcast emergency messages to all vehicles at the intersection. In their experiments, the worst cases have at most 3.23 vehicles involved in the collisions, and their conservative estimation shows that a 77% drop in the number of vehicles per accident is possible when compared with today's traffic control at intersections [2]. Hence, the number of vehicles involved in individual incidents can be drastically reduced.

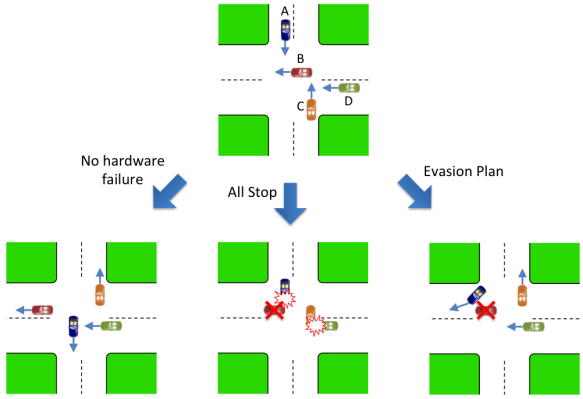


Fig. 1. All-stop strategy versus evasion planning.

This approach assumes vehicles will take ad hoc evasive actions upon receiving an emergency notification. The success of these last-second evasive actions depends on the coordination among vehicles. Suppose there are four vehicles at an intersection as shown in Fig. 1. If there is no mechanical failure, all vehicles will go through the intersection without collisions. However, if Vehicle B suddenly loses power and stops at the middle of the intersection, other vehicles cannot stop because vehicles will continue to move briefly after braking due to their momentum, resulting in collisions due to this All-Stop policy as shown in Fig. 2. Instead, vehicles should take evasion actions—Vehicle A turns right, and both Vehicles C and D continue to move without stopping. If there is a lack of coordination between Vehicles A and C, Vehicle A could think that Vehicle C is going to turn left. This may cause Vehicle A to also turn left and collide with Vehicle C.

Another drawback of taking last-second evasive actions is that there may not be enough room for evasion. Suppose there is a slow-moving Vehicle E in the front right side of Vehicle A, as shown in Fig. 2. As can be seen, Vehicle A cannot avoid collisions no matter what it does. To avoid this problem, the AIM protocol should have reserved some space for evasion actions of Vehicle A. We therefore propose the *preemptive* approach of evasion: the AIM protocol should

[1]Department of Computer Science, The University of Texas at Austin. {chiu,pstone}@cs.utexas.edu

[2]Department of Electrical and Computer Engineering, The University of Texas at Austin. {liangfok,theory,c.julien}@mail.utexas.edu

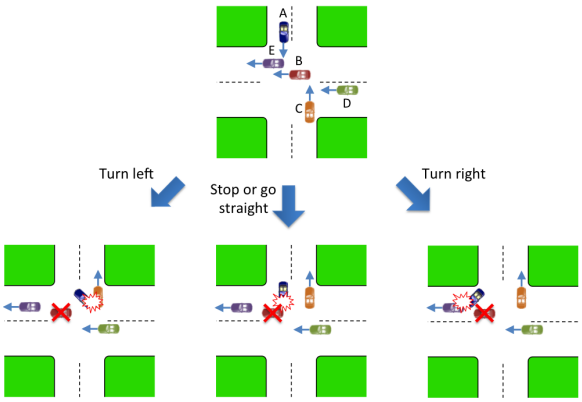


Fig. 2. A collision due to the lack of space for evasive actions.

generate evasive actions for incoming vehicles and reserve enough space for these evasive actions before vehicles enter an intersection.

In this paper, we present an implementation of this preemptive approach to make the AIM protocol fail-safe with respect to a restricted set of mechanical failures. We focus on a restricted set of mechanical failures only because there is no obvious way to prevent collisions in the general case—unless only one vehicle is allowed in the intersection at a time. Since there is no perfectly fail-safe protocol, we opt for a protocol that protects vehicles from a number of common mechanical failures and rely on the last-second ad hoc evasive actions to handle other failures.

Before we describe our protocol in detail in Section VIII, we present the related work and a summary of the original AIM protocol in Section II and Section III, respectively. We then outline the simulation of mechanical failures in Section IV, the detection of mechanical failures in Section V, the definition of evasion plans in Section VI, and the formal statements of our problem in Section VII. Finally, we discuss an implementation of evasion plan execution on a miniature autonomous intersection testbed in Section IX.

II. RELATED WORK

Safety is a major factor in the design of AIM, particularly in the incorporation of *buffers* to handle sensing and control errors of autonomous vehicles. The protocol is fail-safe in terms of communication failure—no vehicle is allowed to enter an intersection when the communication network is disrupted. More importantly, if all vehicles follow the protocol exactly, the AIM protocol guarantees no collision can occur. Dresner and Stone conducted a failure-mode safety analysis and argued that AIM is safer than traffic signals for human drivers [4]. They proposed preventing vehicles from entering an intersection after an accident but did not suggest what vehicles should do to avoid collisions if they cannot stay out of the intersection.

There is much work regarding safety properties of traffic signals. For instance, there are studies of the relationship between accidents and traffic level [5] as well as accidents in particular types of intersections [6], [7]. But these studies focus on traditional intersections and do not consider the possibility of improving intersection efficiency and safety with information technology and artificial intelligence.

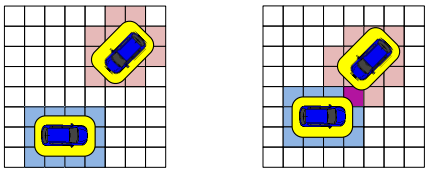
Cooperative collision avoidance systems deal with the safety of vehicles on road networks including intersections. Colombo and Vecchio studied the least restrictive controller for collision avoidance of vehicles at an intersection via computing the maximal controlled invariant set [8]. Hafner et al. presented experimental results for an intersection collision avoidance system that utilizes Dedicated Short-Range Communications to share safety critical state information and generates control commands to vehicles to prevent collisions [9]. Our work is somewhat similar to [10], which proposed a hybrid architecture for collision avoidance at intersections. Their approach is based on failsafe maneuvers, which are infinite horizon open-loop contingency plans with a safety guarantee with respect to some set of vehicles even if sensors and communication fail in the network. However, the main theorem (Theorem 4) of the paper requires that all vehicles follow their intersection-crossing algorithm exactly. Our paper relaxes this requirement by avoiding collision even if some vehicles lose control and fail to follow the protocol.

III. AUTONOMOUS INTERSECTION MANAGEMENT

The AIM protocol is based on a *reservation* paradigm, in which vehicles "call ahead" to reserve space-time in the intersection [2]. The system assumes that computer programs called *driver agents* control the vehicles, while an arbiter agent called an *intersection manager* (IM) is placed at each intersection. The driver agents attempt to reserve a block of space-time in the intersection. The IM decides whether to grant or reject requested reservations. In brief, the paradigm proceeds as follows.

- An approaching vehicle announces its impending arrival to the IM. The vehicle indicates its predicted arrival time, velocity, acceleration, and arrival and departure lanes.
- The IM simulates the vehicle's path through the intersection, checking for conflicts with the paths of any previously processed vehicles.
- If there are no conflicts, the IM issues a reservation. It then becomes the vehicle's responsibility to arrive at, and travel through, the intersection as specified.
- The car may only enter the intersection once it has successfully obtained a reservation.

The prototype intersection control policy operates by dividing the intersection into a grid of *reservation tiles*. When a vehicle approaches the intersection, the IM uses the data in the reservation request regarding the time and velocity of arrival, vehicle size, etc. to simulate the intended journey across the intersection. At this stage the IM also introduces safety buffers that further ensure a safe traversal of the intersection (Fig. 3). At each simulated time step, the policy determines which reservation tiles will be occupied by the vehicle. If the vehicle's space-time request has no conflict, the reservation is successful; otherwise, the reservation request will be rejected.



(a) Successful (b) Rejected

Fig. 3. (a) The vehicles' space-time request has no conflicts at time t. (b) The vehicle's request is rejected because at time t of its simulated trajectory, the vehicle requires a tile already reserved by another vehicle. The yellow area represents the *space buffer* of the vehicle.

Empirical results demonstrated that the proposed system can dramatically improve the intersection efficiency when compared to traditional intersection control mechanisms [2]. Overall, by allowing for much finer-grained coordination, the simulation-based reservation system can dramatically reduce per-car delay by two orders of magnitude relative to traffic signals and stop signs. This reduction of delays can translate into less traffic congestion [11], which in turn leads to better fuel efficiency and lower emissions.

IV. SIMULATION OF MECHANICAL FAILURE

Our evasion planning relies on the prediction of outcomes of mechanical failures to determine what courses of actions can prevent collisions. In this section, we describe the simulation of mechanical failures conducted by IMs.

A. Physical Model of Vehicles

Intersection managers employ a simplified physical model of vehicles' movement when computing the trajectories of vehicles and the set of tiles that are potentially occupied by the vehicles [2]. In this model, the set of differential equations for non-holonomic motion are:

$$\frac{\partial x}{\partial t} = v \cdot \cos(\phi), \quad \frac{\partial y}{\partial t} = v \cdot \sin(\phi), \quad \frac{\partial \phi}{\partial t} = v \cdot \frac{\tan \psi}{L}, \quad (1)$$

where (x,y) is the position of the center of the front of vehicle, ϕ is the direction of the vehicle, and L is the length of the vehicle's wheelbase. The position and the direction depend on the steering angle ψ and the velocity v. At each time step in the simulation, the IM solves these equations numerically by holding v and ψ as constant, while v and ψ may have different values at different time steps. The change of v is determined by an *acceleration profile* $a(t)$, which is the acceleration the vehicle should use at time t. The change of ψ is determined by a sequence of *target points* $\hat{p}(p)$, where p is the position on the trajectory. The vehicle should always aim at $\hat{p}(p)$ by adjusting its steering angle ψ.

B. Mechanical Failure

When a mechanical failure occurs, the vehicle may no longer follow the given acceleration profile and target points and start to deviate from its trajectory. The effect of mechanical failures on v and ψ also depends on the type of mechanical failures as well as the position and the velocity of the vehicle when the failure occurs. For example, if a vehicle loses power, the vehicle will start to decelerate and the deceleration will depend on the current velocity.

In our current implementation, IMs consider only a finite set F of mechanical failures whose effects can be predicted. A partial list of these failures is: (a) A tire blowout occurs; (b) The gas pedal gets stuck; and (c) Loss of power due to empty gas tank, etc. We can mathematically describe the effect of a mechanical failure $f \in F$ as a mapping from the vehicle type, the vehicle position, and the vehicle velocity to $a(\cdot)$ and $\hat{p}(\cdot)$, which are substitutes for the acceleration profile and the target points, causing the vehicle to have a different trajectory.

C. Handling Uncertainty

The simulation of trajectories is sensitive to the *precise* position and velocity of the vehicles. When an IM runs a simulation of a vehicle, the vehicle has not entered the intersection yet. Thus all the IM knows is the set of tiles in the grid that might be occupied by the vehicle in the future. Since vehicles can legally be at any position inside the buffer covered by the tiles, the IM should not assume it is always located at the center of the buffer. Similarly, the vehicle can legally arrive at the intersection at a sightly different time and velocity within the given error bounds. Therefore, the simulation should simulate a *bundle* of trajectories, which comprises all possible trajectories of the vehicles. When a mechanical failure occurs, the deviation will be applicable to all trajectories in the bundle.

To compute trajectory bundles, we use intervals to present the uncertainty of positions and velocities. In our current implementation, IMs solve the differential equations in Eq. 1 numerically; thus we estimate the trajectory bundles by solving the equations repeatedly with different values in the intervals, especially the boundary values. Fortunately, the space and time buffers are rectangles, and hence we can often simplify the calculation by assuming the numerical solutions with boundary values encompass all values in the bundle.

D. Simulation of Collisions

There is a short period of time between the occurrence of a mechanical failure and the execution of evasion plans in which the IM cannot control vehicles and collisions can happen. For example, if Vehicle A in Fig.1 is very close to Vehicle B such that the IM fails to detect that Vehicle B loses its power on time, Vehicle A will inevitably collide with Vehicle B. While there is little the IM can do to prevent this collision, the IM should try to prevent multiple-vehicle collisions by preventing other vehicles from hitting Vehicles A and B after the initial collision. One problem is that after the collision, the positions of Vehicles A and B change abruptly. If they get in the way of Vehicle C, the evasion plan should instruct Vehicle C to turn left instead. Therefore, IMs need to simulate collisions and predict the outcomes.

One difficulty of the simulations is that there is a great deal of randomness in collisions and it is hard to predict their effects. Thus we instead use the notion of *collision circle* which is a circular area centered at the position of a collision. We assume that vehicles will stop completely within the collision circle after a collision, and therefore

the evasion plan should prevent vehicles from entering the collision circle. Currently, our implementation cannot model multiple-vehicle collisions outside the initial collision circle.

V. FAILURE DETECTION AND IDENTIFICATION

Let t_0 be the time at which the mechanical failure $f \in F$ of a vehicle σ occurs. Normally if the vehicle in question is able to detect the problem immediately and the communication with the IM is uninterrupted, the vehicle can send a message to inform the IM that something wrong has occurred. Even if the vehicle failed to inform the IM about the problem, the IM can rely on traffic cameras to detect the mechanical failure. For example, Kamijo et al. proposed an algorithm, called "spatio-temporal Markov random field" that tracks vehicles at intersections to determine when an incident has occurred [12]. Let t_1 be the time at which f is detected by the IM, where $t_1 \geq t_0$. Our system has a maximum delay T_{detect} of the detection of a failure, such that $t_1 < t_0 + T_{\mathsf{detect}}$. If $t_1 \geq t_0 + T_{\mathsf{detect}}$, the IM cannot generate evasive actions for vehicles and must rely on ad-hoc evasive actions that cannot guarantee prevention of collisions.

VI. EVASION PLANS

An *evasion plan* π is a set of control instructions for vehicles inside or near the intersection. The objective of an evasion plan is to prevent vehicles that are not involved in the accident from collisions in the future. Let us define an evasion plan precisely as follows. Suppose t_2 is the time at which an evasion plan is executed. Let Σ_1 be the set of all vehicles inside the intersection at t_2. Let Σ_2 be the set of all vehicles outside the intersection at t_2 but that are past the *point of no return*—the position beyond which a vehicle cannot stop before the intersection and must enter the intersection. Moreover, let Σ_3 be the set of all vehicles involved in the accident. Then the set $\Sigma = (\Sigma_1 \cup \Sigma_2) \setminus \Sigma_3$ is the set of all vehicles controlled by an evasion plan. An *evasion plan* is a list of n pairs $\langle (b_1, \delta_1), (b_2, \delta_2), \ldots, (b_n, \delta_n) \rangle$, where $n = |\Sigma|$, b_i is a boolean value indicating whether the vehicle $\sigma_i \in \Sigma$ should stop, and δ_i is the new direction for σ_i. We say (b_i, δ_i) is the control instruction for the vehicle σ_i.

When a vehicle receives (b_i, δ_i), it will check the value of b_i to see whether it needs to stop. If b_i is true, the vehicle will immediately steer towards δ_i until the direction is δ_i, and then break as hard as possible to stop. If b_i is false, no modification of the vehicle's trajectory is needed; the vehicle will keep using the acceleration profile and the sequence of target points to leave the intersection.

While a more complicated evasion plan should offer better ways to evade vehicles and avoid collisions, we opt for simple evasion plans that can be quickly generated because the IM has only a few seconds to generate such plans.

VII. COMPLETENESS OF INCIDENT HANDLERS

We define an *incident* as a triple (σ, f, t_0), which denotes that a mechanical failure $f \in F$ of a vehicle $\sigma \in \Sigma_3$ occurred at time t_0. When an incident (σ, f, t_0) occurs, one or more collisions may occur subsequently, and the number of vehicles involved in the collisions may increase in multiple-vehicle collision scenarios. Let Σ be the set of all vehicles inside the intersection or past their points of no return, but not involved in any collisions at time t_2 after the incident. A *solution* to an incident is an evasion plan π executed at time t_2 such that all vehicles in Σ will never collide after t_2. More precisely, we denote a solution by (π, t_2), and say a solution is *valid* for an incident (σ, f, t_0) if there exists a time $t_1 \in [t_0, t_0 + T_{\mathsf{detect}})$ such that if the IM detects the incident at t_1, it can execute π at time t_2 to avoid any further collisions except those that have already occurred on or before t_2, where $t_2 \in [t_1, t_1 + T_{\mathsf{exe}})$ for some constant T_{exe}.

```
Procedure AcceptReqeuestMessage(r)
  // Input: r = (σ, t', v', l1, l2) is a request message
  // Let DB be the evasion plan database.
  Compute the trajectory τ of the vehicle σ from l1 to l2
  Find a set A1 of tiles occupied by σ on τ
  If some tiles in A1 have been assigned to other vehicles
    Send a reject message to σ
  End
  For each time step t0 during the traversal of τ
    For each mechanical failure f ∈ F
      Let I = (σ, f, t0) be an incident
      If UpdateEvasionPlanDB(DB, I) = False
        Send a reject message to σ
      End
    End
  End
  Let P be the set of plans in DB for incidents involving σ
  Let A2 be the set of all tiles used by all plans in P
  If all tiles in A2 are either occupied by σ or unoccupied
    Assign tiles in A1 to σ
    Assign tiles in A2 to corresponding vehicles in evasion plans
    Send a confirm message to σ
  Else
    Send a reject message to σ
  End
```

Fig. 4. The request handler in the modified AIM protocol.

An *incident handler* is a process that runs in parallel with an IM. Its job is to detect the occurrence of mechanical failures and schedule the execution of evasion plans for them. An incident $I = (\sigma, f, t_0)$ is t_1-*solvable* if there exists a valid solution (π, t_2) for I, where $t_2 \in [t_1, t_1 + T_{\mathsf{exe}})$ is the execution time of π. An incident is *solvable* if it is t_1-solvable for all $t_1 \in [t_0, t_0 + T_{\mathsf{detect}} + T_{\mathsf{exe}})$. In this paper, we assume that all incidents are solvable. An unsolvable incident does not mean that there is no way to mitigate the scale of collisions; but in our framework there is no guarantee that the vehicles not involved in the collisions will not collide after t_2.

An incident handler *solves* an incident I if it generates a valid solution for I. An incident handler is *complete* if it solves all solvable incidents. That is, a complete incident handler can generate valid solutions for all kinds of failures of all vehicles entering an intersection at all times. It is hard to build a complete incident handler to deal with all kinds of failures. Therefore, we will focus on building an incident handler that is F-*complete*—the completeness for a *restricted* set F of possible mechanical failures for which the incident handler can generate valid solutions. Note that IMs only allow vehicles whose incidents can be preemptively handled with some evasion plans to enter the intersection. A vehicle

```
Procedure UpdateEvasionPlanDB(I)
  // Input: I = (σ, f, t_0) is an incident
  Simulate the incident I for a period of time T_detect + T_exe
  For each time step t_2 ∈ [t_0, t_0 + T_detect + T_exe)
    Let Σ_1 be vehicles inside the intersection at t_2
    Let Σ_2 be vehicles passed the point of no return at t_2
    Find the set Σ_3 of vehicles collided on or before t_2
    Let A be the set of occupied tiles on and after t_2
    DB(I, t_2) := SearchEvasionPlan(I, (Σ_1 ∪ Σ_2) \ Σ_3, A, t_2)
  End
  // Check whether evasion plan exists for all t_1
  For each time step t_1 in [t_0, t_0 + T_detect)
    found := False
    For each time step t_2 in [t_1, t_1 + T_exe)
      If DB(I, t_2) ≠ Nil, then found := True
    End
    If found = False, then return False
  End
  Return True
```

Fig. 5. The procedure for updating the evasion plan database.

that is denied by the IM should resubmit the reservation requests until the IM successfully pre-computes all evasion plans for all possible incidents with this vehicle.

VIII. MODIFIED INTERSECTION MANAGERS

In this section, we present a modified IM that pre-computes evasion plans. Fig. 4 is the pseudo-code of the request handler of IMs, which is invoked when it receives a reservation request $r = (\sigma, t', v', l_1, l_2)$, where σ is the vehicle's ID, t' is the arrival time, v' is the arrival velocity, l_1 is the lane at which σ enters the intersection, and l_2 is the lane from which σ leaves the intersection. The modification includes (1) the incremental update of the *evasion plan database* DB and (2) the allocation of tiles for evasion plans. DB is a mapping from incidents and execution times to evasion plans that will be used by the incident handler.

After computing the trajectory τ of σ, the IM will simulate all kinds of mechanical failures in F that may happen at every time step t_0 when σ moves across the intersection along τ. For each possible incident I, the IM calls UpdateEvasionPlanDB to compute the evasion plans for all possible execution times $t_2 \in [t_0, t_0 + T_{\text{detect}} + T_{\text{exe}})$ (see Fig. 5). UpdateEvasionPlanDB stores the evasion plans in DB and checks whether there is at least one evasion plan for each detection time t_1. After that, AcceptReqeuestMessage assigns the tiles to σ and the evasion plans if the tiles have not been occupied. Unlike the original AIM implementation, each tile can be assigned to more than one vehicle because the set of tiles used by different evasion plans can overlap. It is safe for evasion plans to share some tiles since only one evasion plan will be executed at a time. Moreover, if the new trajectory of σ is different from the original trajectory of σ, the tiles on the original trajectory can be used for evasion. Finally, the IM sends a confirmation message to σ.

Fig. 6 shows the search procedure used by UpdateEvasionPlanDB to find an evasion plan at time t_2 for a particular incident I. SearchEvasionPlan is a recursive procedure that enumerates all possible evasion plans until it finds one that safely avoids collisions on and after t_2. In each iteration, it first checks whether it is safe for a vehicle σ

```
Procedure SearchEvasionPlan(I, Σ, A, t_2)
  // Input: I = (σ, f, t_0) is an incident
  // Input: Σ is a set of vehicles not collided before t_2
  // Input: A is a set of occupied tiles
  // Input: t_2 is time at which evasion plan will be executed
  If Σ is an empty set, then return an empty list ⟨⟩.
  Let σ be the first vehicle in Σ
  Let A_1 be the set of tiles on the original trajectory of σ
  If A ∩ A_1 = ∅       // σ will not collide
    π_1 := SearchEvasionPlan(I, Σ \ {σ}, A ∪ A_1, t_2)
    If π_1 = Nil, then return Nil    // no evasion plan is found
    Return ⟨(True, 0)⟩ ⊕ π_1    // ⊕ is the concatenation operator
  Else
    Let δ_0 be the direction of σ at t_2
    For each direction δ ∈ {δ_0 + k × (π/9) : −3 ≤ k ≤ 3}
      Let τ_2 be the new trajectory after executing (False, δ) at t_2
      Let A_2 be the set of tiles occupied by σ on τ_2
      If A ∩ A_2 = ∅       // σ will not collide
        π_2 := SearchEvasionPlan(I, Σ \ {σ}, A ∪ A_2, t_2)
        If π_2 ≠ Nil, then return ⟨(False, δ)⟩ ⊕ π_2
      End
    End
    Return Nil       // no solution
```

Fig. 6. The search procedure for finding an evasion plan.

```
Procedure IncidentHandler()
  Loop until an incident I = (σ, f, t_0) is detected at t_1
    Stop granting reservations to vehicles.
    For each time step t_2 in [t_1, t_1 + T_exe)
      If DB(I, t_2) ≠ Nil       // an evasion plan exists
        π := DB(I, t_2)
        Schedule π to be executed at time t_2
        Break the for-each loop
      End
    End
    If no evasion plan is scheduled for I
      secondaryIncidentHandler(I, t_1)
    End
```

Fig. 7. The procedure of the incident handler.

to move along its original trajectory without stopping using instruction $(\text{False}, 0)$. If not, it will check into which direction σ can turn and then stop. To limit the search space, the procedure considers a finite number of directions in the set $\{\delta_0 + k \times (\pi/9) : -3 \le k \le 3\}$ only, where δ_0 is the direction of σ at time t_0 and k is an integer. If it finds a direction δ that is safe for σ, the instruction for σ is (True, δ); otherwise, the procedure backtracks to consider another evasion actions for previous vehicles. Finally, the procedure will return a valid evasion plan at t_2 for I, or Nil if there is no solution. The time complexity of the procedure is $O(8^{|\Sigma|})$, where $|\Sigma|$ is usually a small number because there is a upper bound on the number of vehicles that can be in an intersection simultaneously.

The incident handler continually monitors the intersection to detect any failure (see Figure 7). Once it detects an incident, it will immediately inform the IM to stop sending out confirmation messages. It will then retrieve the evasion plan π^* with the earliest execution time from DB and schedule π^* for execution. If no evasion plan is found or the failure is unknown, the incident handler will invoke a secondary incident handler which will either notify the vehicles to take evasive actions themselves or generate an evasion plan on demand to avoid collisions as much as possible and minimize the severeness of inevitable collisions in the future.

The following theorem states that the incident handler, in collaboration with the modified intersection manager, is F-

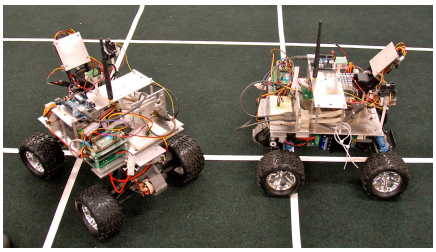


Fig. 8. The robots used in our testbed in their final orientation. The robot on the right entered the intersection first but suffered a mechanical breakdown. The robot on the left executes an evasion plan that causes it to steer to the left thereby avoiding collision with the first robot.

complete. But due to space limitation we omit the proof.

Theorem 1: The incident handler in Fig. 7 is F-complete.

IX. IMPLEMENTATION PROOF-OF-CONCEPT

We implemented evasion planning in an autonomous intersection testbed at UT Austin [13], which consists of small robotic vehicles based on the Traxxas Stampede R/C car mobile chassis as shown in Figure 8. While miniature relative to real vehicles, they move in the same non-holonomic manner where only the front two wheels are steerable. The vehicles are autonomous with on-board x86 computers and various micro-controllers for accessing the on-board sensors and actuators. White lines are placed on the ground denoting lanes that the robots follow using on-board CMUCam2 vision sensors, while overhead markers are placed at the points of approach, entry, and exit surrounding the intersection. The robots use upward facing IR range sensors to detect these markers and, combined with odometer measurements, compute their positions in and around the intersection.

Two robots are configured to approach the intersection from perpendicular directions. As they approach, they communicate with the IM that grants both vehicles entry times that ensure safe crossing assuming no mechanical failures occur. We tested this scenario to verify safe passage. To evaluate situations that arise due to mechanical failures, we modified the software to simulate a mechanical fault (i.e., after entering the intersection, the motors are intentionally disabled). Only the first robot that enters the intersection is configured to suffer this fault. We verified that the fault results in a collision by having the two robots travel through the intersection like before, and observing the first robot stop midway and the second robot colliding into it. Finally, evasion planning is enabled by having the first robot report the mechanical failure to the IM. Upon receiving this, the IM sends the second robot an evasion plan, which in this case is to turn left. The second vehicle is able to successfully avoid collision by turning left after the first robot stalls, thus demonstrating the feasibility of our approach. A video showing all three experiments accompanies this submission.

X. CONCLUSIONS AND FUTURE WORK

Autonomous intersection management can dramatically improve intersection efficiency, reduce traffic delays, and alleviate traffic congestion [3]. But concerns about the safety and questions like "what if a tire blows out?" are common. This paper addresses this concern by proposing a modified AIM protocol that is fail-safe with respect to a restricted set of mechanical failures. The fail-safe property is crucial to the deployment of AIM to the real world.

We argue that it is essential to compute evasion plans preemptively because last-second evasive actions are not always successful without pre-allocation of spaces in an intersection. Our approach guarantees that once an evasion plan is executed, there will be no collision among vehicles that have not yet collided. Note that our approach still requires vehicles to carry out ad-hoc evasive actions if the mechanical failure is not one of those considered by the IM.

In the future, we intend to (1) derive a close-form solution to Eq.1 to compute the trajectory bundle exactly; (2) include a better collision model for multiple-vehicle collisions; (3) relax the assumption that no two mechanical failures occur nearly at the same time; and (4) examine the tradeoff between the intersection efficiency and the number of mechanical failures that can be addressed preemptively.

Acknowledgments. Part of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

REFERENCES

[1] DARPA, "DARPA Urban Challenge," http://www.darpa.mil/grandchallenge/index.asp, 2007.

[2] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research (JAIR)*, March 2008.

[3] D. Fajardo, T.-C. Au, S. T. Waller, P. Stone, and C. Y. D. Yang, "Automated intersection control: Performance of a future innovation versus current traffic signal control," *TRR: Journal of the Transportation Research Board*, no. 2259, pp. 223–232, 2012.

[4] K. Dresner and P. Stone, "Mitigating catastrophic failure at intersections of autonomous vehicles," in *The Fifth Workshop Agents in Traffic and Transportation (ATT 08)*, May 2008.

[5] T. Sayed and S. Zein, "Traffic conflict standards for intersections," *Transportation Planning and Technology*, vol. 22, no. 4, pp. 309–323, August 1999.

[6] J. A. Bonneson and P. T. McCoy, "Estimation of safety at two-way stop–controlled intersections on rural highways," *Transportation Research Record*, vol. 1401, pp. 83–89, 1993.

[7] B. N. Persaud, R. A. Retting, P. E. Gardner, and D. Lord, "Safety effect of roundabout conversions in the united states: Empirical bayes observational before-after study," *Transportation Research Record*, vol. 1751, pp. 1–8, 2001.

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

[9] M. R. Hafner, D. Cunningham, L. Caminiti, and D. D. Vecchio, "Automated vehicle-to-vehicle collision avoidance at intersections," in *Proceedings of World Congress on Intelligent Transport Systems*, 2011.

[10] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, 2011.

[11] T.-C. Au and P. Stone, "Motion planning algorithms for autonomous intersection management," in *AAAI 2010 Workshop on Bridging The Gap Between Task And Motion Planning (BTAMP)*, 2010.

[12] S. Kamijo, Y. Matsushita, K. Ijeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 108–118, 2000.

[13] C.-L. Fok, M. Hanna, S. Gee, T.-C. Au, P. Stone, C. Julien, and S. Vishwanath, "A platform for evaluating autonomous intersection management policies," in *Proceedings of the International Conference on Cyber-Physical Systems (ICCPS), to appear*, 2012.