Motion Planning Algorithms for Autonomous Intersection Management

Tsz-Chiu Au
Department of Computer Science
The University of Texas at Austin
1 University Station C0500
Austin, Texas 78712-1188
chiu@cs.utexas.edu

Peter Stone
Department of Computer Science
The University of Texas at Austin
1 University Station C0500
Austin, Texas 78712-1188
pstone@cs.utexas.edu

Abstract

The impressive results of the 2007 DARPA Urban Challenge showed that fully autonomous vehicles are technologically feasible with current intelligent vehicle hardware. It is natural to ask how current transportation infrastructure can be improved when most vehicles are driven autonomously in the future. Dresner and Stone proposed a new intersection control mechanism called *Autonomous Intersection Management* (AIM) and showed in simulation that intersection control can be made more efficient than the traditional control mechanisms such as traffic signals and stop signs. In this paper, we extend the study by examining the relationship between the precision of cars' motion controllers and the efficiency of the intersection controller. We propose a planning-based motion controller that can reduce the chance that autonomous vehicles stop before intersections, and show that this controller can increase the efficiency of the intersection control mechanism.

Introduction

Recent advances in intelligent vehicle technology suggest that autonomous vehicles will become a reality in the near future (Squatriglia 2010). Today's transportation infrastructure, however, does not utilize the full capacity of autonomous driving systems. Dresner and Stone proposed a multiagent systems approach to intersection management called *Autonomous Intersection Management* (AIM), and in particular describe a *First Come, First Served* (FCFS) policy for directing vehicles through an intersection (Dresner and Stone 2008). This approach has been shown, in simulation, to yield significant improvements in intersection performance over conventional intersection control mechanisms such as traffic signals and stop signs. Despite its impressive performance, we believe that it is possible to make this intersection control mechanism more efficient by considering how best autonomous vehicles can utilize the intersection management protocol.

In this paper, we present an improved controller for autonomous vehicles to interact with intersection managers in AIM. First, we leverage Little's law in queueing theory to understand how the performance of an autonomous vehicle relates to the overall intersection throughput. Then we identify approaches to improve the motion controllers of autonomous vehicles such that they can plan ahead of time when they make reservations in the AIM system and traverse the intersection at a higher speed. We used motion planning techniques to address two problems in making and maintaining reservations: (1) how a vehicle computes the best time and velocity for arriving at the intersection such that it is less likely to stop at the intersection; and (2) how a vehicle decides whether it can arrive at the intersection at the time and velocity proposed by the intersection manager, such that it can cancel the reservation earlier if it knows it cannot make it. We predict that the use of these planning techniques can improve the throughput of intersections and reduce the traversal time of vehicles, thus providing motivation for autonomous vehicles to adopt these planning-based controllers.

Autonomous Intersection Management

Traffic signals and stop signs are very inefficient—not only do vehicles traversing intersections experience large delays, but the intersections themselves can only manage a limited traffic capacity—much less than that of the roads that feed into them. Dresner and Stone have introduced a novel approach to efficient intersection management that is a radical departure from existing traffic signal optimization schemes (Dresner and Stone 2008). The solution is based on a *reservation* paradigm, in which vehicles "call ahead" to reserve space-time in the intersection. In the approach, they assume that computer programs called *driver agents* control the vehicles, while an arbiter agent called an *intersection manager* is placed at each intersection. The driver agents attempt to reserve a block of space-time in the intersection. The intersection manager decides whether to grant or reject requested reservations according to an *intersection control policy*. In brief, the paradigm proceeds as follows.

- An approaching vehicle announces its impending arrival to the intersection manager. The vehicle indicates its size, predicted arrival time, velocity, acceleration, and arrival and departure lanes.
- The intersection manager simulates the vehicle's path through the intersection, checking for conflicts with the paths of any previously processed vehicles.

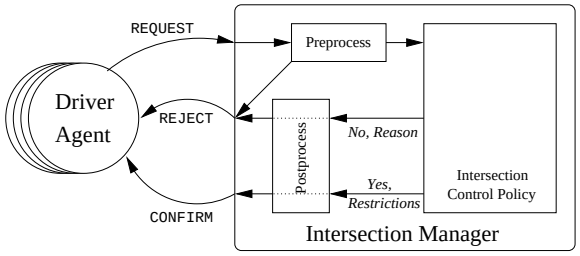
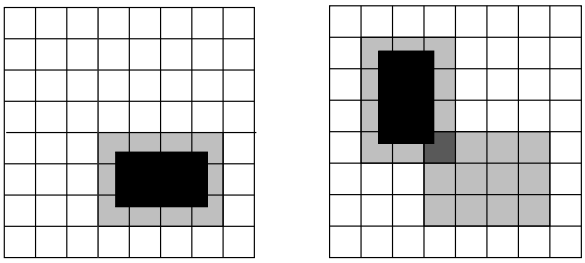


Figure 1: Diagram of the intersection system.



(a) Successful (b) Rejected

Figure 2: (a) The vehicle's space-time request has no conflicts at time t. (b) The black vehicle's request is rejected because at time t of its simulated trajectory, the vehicle requires a tile already reserved by another vehicle. The shaded area represents the *static buffer* of the vehicle.

- If there are no conflicts, the intersection manager issues a reservation. It becomes the vehicle's responsibility to arrive at, and travel through, the intersection as specified (within a range of error tolerance).
- The car may only enter the intersection once it has successfully obtained a reservation.

Figure 1 diagrams the interaction between driver agents and an intersection manager. A key feature of this paradigm is that it relies only on vehicle-to-infrastructure (V2I) communication. In particular, the vehicles need not know anything about each other beyond what is needed for local autonomous control (e.g., to avoid running into the car in front). The paradigm is also completely robust to communication disruptions: if a message is dropped, either by the intersection manager or by the vehicle, delays may increase, but safety is not compromised. Safety can also be guaranteed in mixed mode scenarios when both autonomous and manual vehicles operate at intersections. The intersection efficiency will increase with the ratio of autonomous vehicles to manual vehicles in such scenarios.

The prototype intersection control policy divides the intersection into a grid of *reservation tiles*, as shown in Figure 2. When a vehicle approaches the intersection, the intersection manager 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 each simulated time step, the policy determines which reservation tiles will be occupied by the vehicle.

If at any time during the trajectory simulation the requesting vehicle occupies a reservation tile that is already reserved by another vehicle, the policy rejects the driver's reservation request, and the intersection manager communicates this to the driver agent. Otherwise, the policy accepts the reservation and reserves the appropriate tiles. The intersection manager then sends a confirmation to the driver. If the reservation is denied, it is the vehicle's responsibility to maintain a speed such that it can stop before the intersection. Meanwhile, it can request a different reservation.

Empirical results in simulation demonstrate that the proposed reservation system can dramatically improve the intersection efficiency when compared to traditional intersection control mechanisms. To quantify efficiency, Dresner and Stone introduce *delay*, defined as the amount of travel time incurred by the vehicle as the result of passing through the intersection. According to their experiments, the reservation system performs very well, nearly matching the performance of the optimal policy which represents a lower bound on delay should there be no other cars on the road (Figure 14 in (Dresner and Stone 2008)). Overall, by allowing for much finer-grained coordination, the simulation-based reservation system can dramatically reduce per-car delay by two orders of magnitude in comparison to traffic signals and stop signs.

Little's Law

First of all, let us consider factors that affect the maximum throughput characteristics of intersections. An important result in queueing theory is Littles law (Little 1961), which states that in a queueing system the average arrival rate of customers λ is equal to the average number of customers T in the system divided by the average time W a customer spends in the system. In the context of intersection management, Little's law can be written as $L = \lambda W$, where

- L is the average number of vehicles in the intersection;
- λ is the average arrival rate of the vehicles at the intersection; and
- W is the average time a vehicle spends in the intersection.

Note that the arrival rate is equal to the throughput of the system since no vehicle stalls inside an intersection.

Little's law shows that the maximum throughput (i.e., the upper bound of λ) an intersection can sustain is equal to the upper bound of L divided by the lower bound of W, where the upper bound of L is the maximum number of vehicles that can coexist in an intersection, and the lower bound of W is the minimum time a vehicle spends in the intersection. Thus, Little's law shows that there are two ways to increase the maximum throughput: 1) increase the average number of vehicles in an intersection at any moment of time, and 2) decrease the average time a vehicle spends in an intersection.

A trivial upper bound on L is the area of the intersection divided by the average static buffer size of the vehicles. But this bound is rather loose and in practice unachievable. Nonetheless, it provides us some hints about the dependence between the maximum throughput and the average static buffer size of the vehicles. Unfortunately the size of an intersection is a hard limit and the static buffer sizes

cannot be too small—there is little an intersection manager can do to squeeze more vehicles into the intersection. Therefore, we cannot dramatically increase the average number of vehicles in an intersection at any moment of time.

Little's law shows that another way to increase the maximum throughput is to reduce the average time a vehicle takes to traverse an intersection. In other words, a vehicle should maintain a high speed during the traversal of the intersection in order to shorten its traversal time. Vehicle's velocity in the intersection depends on two factors: 1) the initial velocity when the vehicle enters the intersection, and 2) the acceleration during the traversal. In the following sections, we will present two techniques that allow vehicles to maintain a high speed during the traversal.

Optimizing Arrival Times and Velocities via Planning Techniques

One of the keys to entering an intersection at a high speed is to prevent vehicles from stopping before entering the intersection. FCFS, by itself, reduces the number of vehicles that stop at an intersection, and therefore it allows vehicles to enter an intersection at a high speed most of the time. In fact, it is one of the main reasons why FCFS is more efficient than traffic lights and stop signs (Dresner and Stone 2008). While FCFS has done a good job in this regard, there is still room for improvement on the autonomous vehicles' side such that driver agents can help by preventing themselves from stopping before an intersection.

There are two scenarios in which an autonomous vehicle has to stop before an intersection in FCFS. First, the vehicle cannot obtain a reservation from the intersection manager and is forced to stop before an intersection. This happens when the traffic level is heavy and most of the future reservation tiles have been reserved by other vehicles in the system. Second, the vehicle successfully obtains a reservation but later determines that it will not arrive at the intersection at the time and/or velocity specified in the reservation. In this scenario the vehicle has to cancel the reservations and those reservation tiles may have been wasted. The effect of a reservation cancellation is not only that the vehicle in question has to stop, but also that temporarily holding reservation tiles may have prevented another vehicle from making reservation. Both of these effects lead to a reduction in the maximum throughput of the intersection.

A poor estimation of arrival times and arrival velocities can lead to the cancellation of reservations. In previous work, the estimation of arrival times and arrival velocities is based on a heuristic we called the *optimistic/pessimistic* heuristic, that derives the arrival time and arrival velocity based on a prediction about whether the vehicle can arrive at the intersection without the intervention of other vehicles (Dresner 2009). However, this heuristic does not guarantee that the vehicle can arrive at the intersection at the estimated arrival time or the estimated arrival velocity; in fact, our experiments showed that vehicles are often unable to reach the intersection at the correct time, forcing them to cancel their reservations after holding the reservations for quite some time.

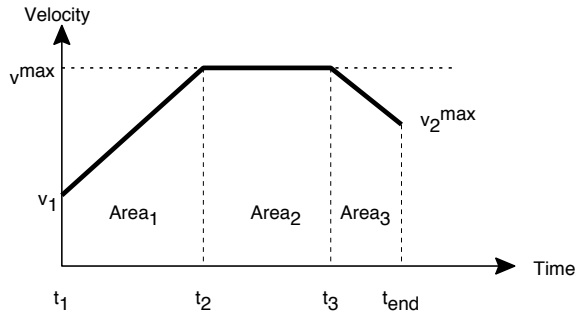
To avoid this problem we propose a new approach to estimate the arrival time and arrival velocity. In our approach when a driver agent estimates its arrival time and arrival velocity, it also generates a sequence of control signals. These control signals, if followed correctly, ensure that the vehicle will arrive at the estimated arrival time and at the estimated arrival velocity. We can formulate this estimation problem as the following multiobjective optimization problem: among all possible sequences of control signals that control the vehicle to enter an intersection, find one such that the arrival time is the smallest and the arrival velocity is the highest.

For an acceleration-based controller, the sequence of control signals is a time sequence of accelerations stating the acceleration the vehicle should take at every time step. We call a time sequence of accelerations an *acceleration schedule*. Like many multiobjective optimization problems, there is no single solution that dominates all other solutions in terms of both arrival time and arrival velocity. Here we choose arrival velocity as the primary objective, since a higher arrival velocity can allow the vehicle to enter the intersection at a higher speed. Our optimization procedure involves two steps: first, determine the highest possible arrival velocity the vehicle can achieve, and second, among all the acceleration schedules that yield the highest possible arrival velocity, find the one whose arrival time is the soonest.

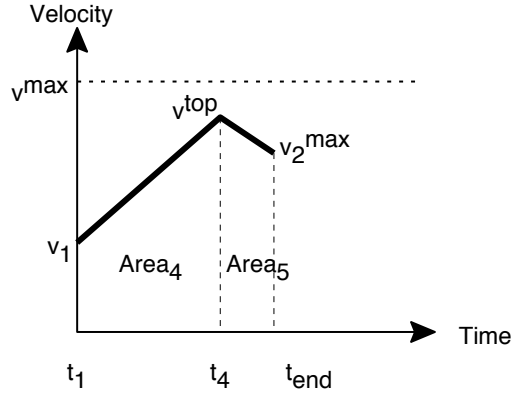
We illustrate how the estimation procedure works using a time-velocity diagram as shown in Figure 3. In this figure, v_1 is the current velocity of the vehicle, t_1 is the current time, D is the distance between the current position of the vehicle and the intersection, v^{max} is the speed limit of the road, and v_2^{max} is the speed limit at the intersection. In addition, we define a_{max} and a_{min} to be the maximum acceleration and the maximum deceleration (minimum acceleration), respectively. We can see that any function $v(\cdot)$ in the time-velocity diagram that satisfies the following five constraints is a feasible velocity schedule for velocity-based controllers.

1. $v(t_1) = v_1$;
2. $\int_{t_1}^{t_{end}} v(t)\,dt = D$, where t_{end} is the arrival time (i.e., the distance traveled must be D);
3. $v(t_{end}) \leq v_2^{max}$ (i.e., the arrival velocity cannot exceed the speed limit at the intersection);
4. $0 \leq v(t) \leq v^{max}$ for $t_1 \leq t \leq t_{end}$ (i.e., the velocity cannot exceed the speed limit of the road or be negative at any point in time); and
5. $a_{min} \leq \frac{d}{dt}v(t) \leq a_{max}$ for for $0 \leq t \leq t_{end}$ (i.e., the acceleration at any point in time must be within the limitations).

We call $v(\cdot)$ a *velocity schedule*, which can be directly used in velocity-based controllers. A velocity schedule is *feasible* if it satisfies the above constraints. Our objective is to find a feasible velocity schedule $v(\cdot)$ such that $v(t_{end})$ is as high as possible while t_{end} is as small as possible. For acceleration-based controllers, we can compute the corresponding feasible acceleration schedule by the derivative of $v(\cdot)$ (i.e., $\frac{d}{dt}v(t)$).



(a) Case 1: $\text{Area}_1 + \text{Area}_3 \leq D$



(b) Case 2: $\text{Area}_1 + \text{Area}_3 > D$

Figure 3: The time-velocity diagrams for the estimation of the arrival time and the arrival velocity.

We propose an *optimization procedure* that can find $v(\cdot)$ with the highest possible $v(t_{end})$ and smallest t_{end}. The basic idea of the procedure is as follows. First of all, compute two values Area_1 and Area_3 as shown in Figure 3(a). To compute Area_1, find a point (t_2, v^{max}) in the velocity-time diagram such that (t_2, v^{max}) is an interception of the line extending from (t_1, v_1) with slope a_{max} and the horizontal line $v = v^{max}$. Let Area_1 be the area of the trapezoid under the line segment from (t_1, v_1) to (t_2, v^{max}). Similarly, to compute Area_3, we arbitrarily choose an arrival time t'_{end} and then find an intercepting point (t'_3, v^{max}) between the line $v = v^{max}$ and the line passing through the point (t'_{end}, v_2^{max}) with slope a_{min}. Let Area_3 be the area of the trapezoid under the line segment from (t'_3, v^{max}) to (t'_{end}, v_2^{max}). Note that Area_3 does not depend on the value of t'_{end} and t'_3; we only need to know the value of v_2^{max}, v^{max} and a_{min} to compute Area_3.

If $\text{Area}_1 + \text{Area}_3 \leq D$, the vehicle can accelerate to v^{max}, maintain the speed for a certain period of time, decelerate to v_2^{max}, and finally reach the intersection (Case 1 in Figure 3(a)). Then $\text{Area}_2 = D - \text{Area}_1 - \text{Area}_3$ is non-negative. Let d be $\frac{\text{Area}_2}{v^{max}}$. Then we can determine the actual value of t'_3 and t'_{end} by $t_3 = t_2 + d$ and $t_{end} = t_3 + \frac{2 \times \text{Area}_3}{v^{max} + v_2^{max}}$. From this the optimization procedure can find a piecewise linear function for $v(\cdot)$ such that $v(\cdot)$ is a feasible velocity schedule. For acceleration-based controllers, the optimization procedure returns the acceleration schedule $\langle (t_1, a_{max}), (t_2, 0), (t_3, a_{min}) \rangle$, which succinctly represents the derivative of $v(\cdot)$.

If $\text{Area}_1 + \text{Area}_3 > D$, the vehicle cannot accelerate to v^{max} because the distance D is too small—if it accelerates to v^{max}, it does have time to decelerate and its arrival velocity will exceed v_2^{max}. But the vehicle may still be able to accelerate to a velocity v^{top} that is less than the speed limit v^{max} and then decelerate to v_2^{max} when it arrives at the intersection. To check whether it is possible to do so, the optimization procedure tries to find the intersection point (t_4, v^{top}) between (1) the line passing through (t_1, v_1) with slope a_{max} and (2) the line passing through (t_{end}, v_2^{max}) with slope a_{min} (see Figure 3(b)). Furthermore, the area under the line segments in Figure 3(b) must be equal to D (i.e., $\text{Area}_4 + \text{Area}_5 = D$). Then we got the following system of equations: (1) $t_4 - t_1 = \frac{v^{top} - v_1}{a_{max}}$; (2) $t_{end} - t_4 = \frac{v_2^{max} - v^{top}}{a_{min}}$; (3) $\text{Area}_4 = (t_4 - t_1)(v_1 + v^{top})/2$; (4) $\text{Area}_5 = (t_{end} - t_4)(v^{top} + v_2^{max})/2$; and (5) $D = \text{Area}_4 + \text{Area}_5$. With some calculations, we get $v^{top} = \sqrt{\frac{a_{max}(v_2^{max})^2 - a_{min}v_1^2 - 2a_{max}a_{min}D}{a_{max} - a_{min}}}$. It can be shown that v^{top} is real if $D \geq 0$, thus v^{top} always exists. Finally, the procedure checks to ensure that $\text{Area}_4 \geq 0$ and $\text{Area}_5 \geq 0$. It turns out that $\text{Area}_4 \geq 0$ and $\text{Area}_5 \geq 0$ if and only if $v^{top} \geq v_1$ and $v^{top} \geq v_2^{max}$. Thus, if $v^{top} \geq v_1$ and $v^{top} \geq v_2^{max}$, the acceleration schedule is $\langle (t_1, a_{max}), (t_4, a_{min}) \rangle$ as shown in Figure 3(b).

If $v^{top} < v_1$ or $v^{top} < v_2^{max}$, either $\text{Area}_4 > D$ or $\text{Area}_5 > D$. This implies that it is impossible to arrive at the intersection with the maximum arrival velocity v_2^{max} while satisfying all the constraints. In this case, the procedure will try to find an acceleration schedule that maximizes the arrival velocity, namely v_2, where $v_2 < v_2^{max}$. First, if $v_1 \leq v_2^{max}$, the vehicle can keep accelerating until it hits the intersection, and the arrival velocity will be maximized despite it is less than v_2^{max}. Thus, the procedure simply returns the acceleration schedule $\langle (t_1, a_{max}) \rangle$, which maximize the arrival velocity and minimize the arrival time. Second, if $v_1 > v_2^{max}$, the vehicle is too close to the intersection and it does not have time to decelerate to a velocity less than v_2^{max}. There is no feasible acceleration schedule for this case since the arrival velocity is larger than the speed limit at the intersection. The vehicle controller should avoid this case by avoiding making reservations too late.

The optimization procedure considers piecewise linear functions only such that slopes of the line segments can only be either a_{max}, a_{min}, or 0, because for any non-piecewise linear function that satisfies the constraints, we can always find a piecewise linear function with a smaller t_{end} and/or a larger $v(t_{end})$.

Validating Arrival Times and Velocities via Planning Techniques

In the previous section, we presented an optimization algorithm that an autonomous vehicle can use to determine its arrival time and velocity with the guarantee that the vehicle can arrive at the intersection at the arrival time and velocity if it follows the acceleration schedule (or the velocity

schedule) closely. The use of this algorithm can prevent the vehicle from making reservations whose arrival times and velocities are not achievable and avoid stopping before the intersection due to these faulty reservation requests.

However, an improved reservation request is not sufficient to ensure that the vehicle can avoid stopping before an intersection and entering an intersection at high speed; the vehicle must also take the confirmation message sent from the intersection manager into account. When an autonomous vehicle receives a confirmation message about the reservation it makes, the message instructs the vehicle to arrive at the intersection at a specific arrival time and at a specific arrival velocity. Depending on the traffic conditions and the intersection management policy, the arrival time and velocity in the confirmation message are not necessarily the same as the ones proposed by the vehicle in the reservation requests. More importantly, there is a delay between sending the reservation request and receiving the confirmation message, and during that time the vehicle's position and velocity may have changed and thus the vehicle may no longer be able to follow the acceleration schedule generated for the reservation request.

Thus it is important for vehicles to check the confirmation message to see whether it can still arrive at the intersection at the given arrival time and velocity. If the vehicle finds that the arrival time and velocity are unachievable, the vehicle should cancel the reservation as early as possible for two reasons: (1) it avoids holding the reservation tiles that the vehicles cannot use and release them as early as possible to let other vehicles to take them; and (2) the vehicle can send another reservation request as soon as possible and hopefully it will then get a feasible reservation time and velocity. In short, an early cancellation of unpromising reservation can improve the throughput of an intersection.

In order to check whether a vehicle can arrive at the intersection at the designated arrival time and velocity, we need another procedure to solve the following problem: given an arrival time t_{end} and an arrival velocity v_{end}, find a sequence of control signals such that the vehicle can arrive at the intersection at time t_{end} and velocity v_{end} while satisfying all the velocity and acceleration constraints. If the procedure *proves* that no such sequence of control signals exists, the vehicle should cancel the reservation to free the reservation tiles and make another reservation request.

Here is our problem definition. Given

- the current time t_1 and the current velocity v_1 of the vehicle;
- the arrival time t_{end} and the arrival velocity v_{end} (we assume v_{ened} is less than the speed limit at the intersection);
- the distance D between the current position of the vehicle and the intersection;
- the speed limit of the road v^{max} (we assume v^{max} is less than or equal to the maximum velocity of the vehicle);
- the maximum acceleration a_{max} and the minimum acceleration a_{min} (i.e., the maximum deceleration) of the vehicle.

The objective is to decide whether an acceleration schedule (or a velocity schedule) exists such that the vehicle can arrive at the intersection at time t_{end} at velocity v_{end} while satisfying all the constraints. If no such acceleration schedule exists, the vehicle should cancel the reservation; otherwise, the vehicle can follow the acceleration schedule in order to arrive at the intersection at the given time and velocity.

We call this problem "the validation problem", as opposed to "the optimization problem" in the previous section. As its name suggested, the validation problem has no optimization because t_{end} and v_{end} are given beforehand. Instead, it is a decision problem in which a certificate is an acceleration schedule that can be verified by a simulation of the vehicle following the acceleration schedule.

There are sampling techniques for motion planning that can effectively explore a complicated configuration space and generate a solution path (e.g., rapidly-exploring random trees (LaValle 1998; LaValle and James J. Kuffner 2000)). These sampling techniques, however, provide no guarantee of finding the solution. More importantly, these techniques cannot be used to prove the non-existence of solutions for a motion planning problem. In our intersection management problem, showing the non-existence of acceleration schedule that meets the requirements is very important. If an acceleration schedule does not exist and the algorithm cannot prove its non-existence, the vehicle cannot decide whether it should cancel the reservation until possibly it is very close to the intersection. Therefore, we are looking for an efficient algorithm that can decide whether the validation problem has solutions. In this section, we will present such an algorithm.

Once again, we rely on an analysis of the time-velocity diagram. Given t_1, t_{end}, v_1, D, v_{end}, v^{max} a_{max} and a_{min} as defined above, we draw a time-velocity diagram as shown in Figure 4. The idea is to find a function $v(\cdot)$ in the time-velocity diagram connecting the point (t_1, v_1) and (t_{end}, v_{end}) while satisfying the following constraints:

1. $v(t_1) = v_1$ and $v(t_{end}) = v_{end}$;
2. $\int_{t_1}^{t_{end}} v(t)\,dt = D$ (i.e., the distance traveled must be D);
3. $0 \leq v(t) \leq v^{max}$ for $t_1 \leq t \leq t_{end}$ (i.e., the velocity cannot exceed the speed limit or be negative at any point in time); and
4. $a_{min} \leq \frac{d}{dt}v(t) \leq a_{max}$ for $t_1 \leq t \leq t_{end}$ (i.e., the acceleration at any point in time must be within the limitations).

If such a function exists, a vehicle following the velocity schedule $v(\cdot)$ (or the acceleration schedule $\frac{d}{dt}v(t)$) can arrive at the intersection at time t_{end} at velocity v_{end}; otherwise, it is impossible for the vehicle to arrive at the intersection at t_{end} and v_{end} without violating some constraints. Hence, the key to answer the validation problem is to decide whether $v(\cdot)$ exists.

First of all, let us study the shape of $v(\cdot)$ in the time-velocity diagram if it exists. In Figure 4, we draw two lines starting at (t_1, v_1) with slope a_{max} and a_{min} respectively. Similarly, we draw two lines ending at (t_{end}, v_{end})

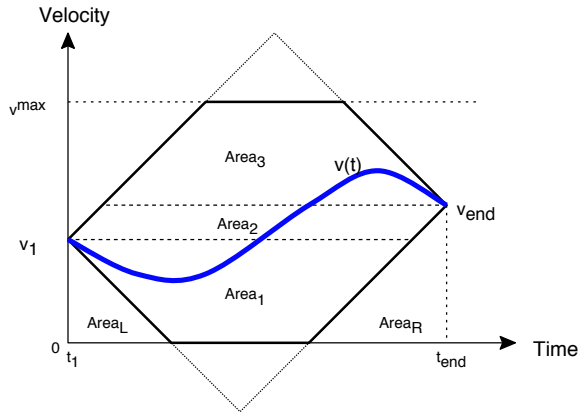


Figure 4: The time-velocity diagram for the validation of the arrival time t_{end} and the arrival velocity v_{end}.

with slope a_{max} and a_{min}. The four lines form a parallelogram, and we are certain that $v(\cdot)$ must lie inside this parallelogram; otherwise, it will violate the acceleration constraints $a_{min} \leq \frac{d}{dt}v(t) \leq a_{max}$. In addition, since $0 \leq v(t) \leq v^{max}$ for $t_1 \leq t \leq t_{end}$, we know $v(\cdot)$ must lie inside the hexagon represented by the solid lines in Figure 4. Here we assume that the lines intersect to form a hexagon; but it is trivial to extend our approach to handle degenerate cases in which the lines do not form a hexagon. Then the remaining contraint that we need to deal with is to make sure that the area under $v(\cdot)$ must be D (i.e., $\int_{t_1}^{t_{end}} v(t)\,dt = D$). We call this constraint the *travel distance* constraint.

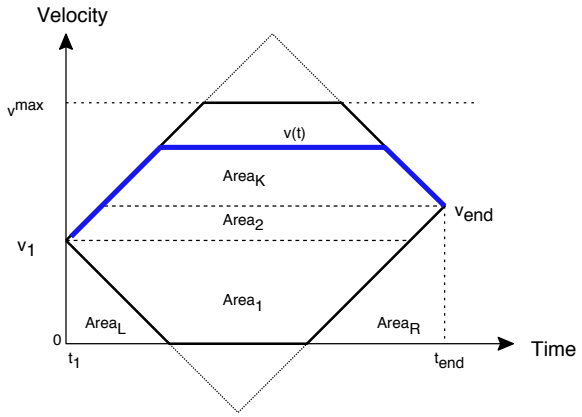
A key insight for checking whether $v(\cdot)$ satisfies the travel distance constraint is that we don't need to check all possible functions for the constraints; instead we only need to check any one of the three piecewise linear functions in Figure 5 to see which one satisfies the constraint. Intuitively, imagine the area under $v(\cdot)$ in Figure 4 is liquid and the hexagon is a container. Then the liquid inside the container will eventually level off and the shape of the liquid will be one of the piecewise linear functions in Figure 5 whose area is also D.

To select the right piecewise linear function among the functions in Figure 5, we look at the value of D. There are five possible cases:

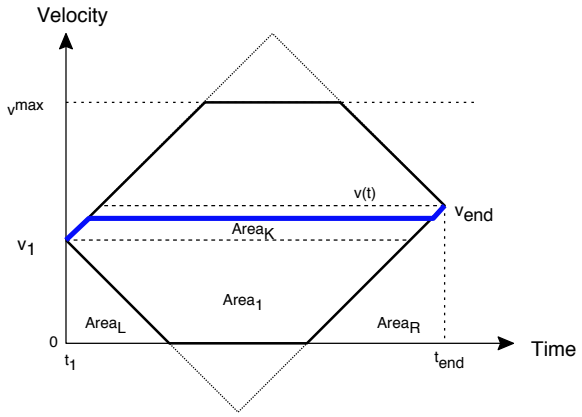
- Case 1: $\text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2 < D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2 + \text{Area}_3$,
- Case 2: $\text{Area}_L + \text{Area}_R + \text{Area}_1 < D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2$,
- Case 3: $\text{Area}_L + \text{Area}_R \leq D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1$,
- Case 4: $D < \text{Area}_L + \text{Area}_R$,
- Case 5: $\text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2 + \text{Area}_3 < D$,

where Area_1, Area_2, and Area_3 are the areas of the parallelograms inside the hexagon, and Area_L and Area_R are the areas of the triangles at the left and right corners of the graph. See Figure 4 for the location of these areas.

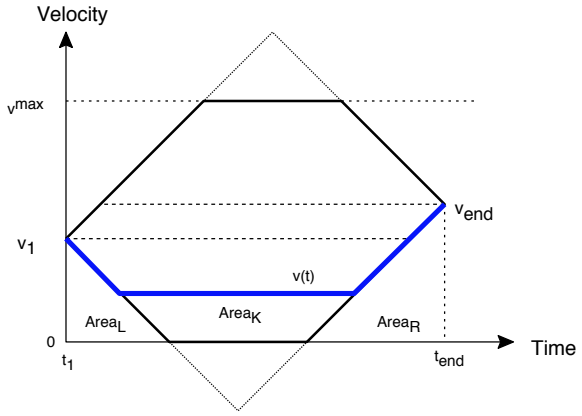
The first three cases are illustrated in Figure 5. The last two cases are infeasible cases in which no $v(\cdot)$ satisfies the



(a) Case 1: $\text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2 < D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2 + \text{Area}_3$



(b) Case 2: $\text{Area}_L + \text{Area}_R + \text{Area}_1 < D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1 + \text{Area}_2$



(c) Case 3: $\text{Area}_L + \text{Area}_R \leq D \leq \text{Area}_L + \text{Area}_R + \text{Area}_1$

Figure 5: The three piecewise linear functions for the validation of the arrival time and the arrival velocity.

travel distance constraint. In case 4, the vehicle is too close to the intersection; even if the vehicle decelerates as much as possible and then accelerate as much as possible, the vehicle cannot reach the intersection at the given arrival time and velocity. In case 5, the vehicle is too far away from the intersection; no matter how it runs it cannot reach the intersection at the given arrival time and velocity without exceeding the speed limit or acceleration limitions.

These five cases are exhaustive and mutually exclusive; thus we can identify which case it is for any given D. Based on this property, we propose a *validation procedure* that proceeds as follows: first, compute Area_1, Area_2, Area_3 Area_L, and Area_R using basic geometric calculations. Second, check which of the five cases is the case for the given constraints. Finally, if the case is one of those in Figure 5, compute the two intersections of the lines in the pairwise-linear function and return the acceleration schedule; otherwise, an infeasible case is found and the vehicle acts accordingly.

We can show that our validation procedure returns a solution *if and only* if a $v(\cdot)$ exists that satisfies all of the constraints. It is a nice property as we mentioned early, since it can determine the impossibility of arriving at the intersection at the given time and velocity as early as possible. In this case, the vehicle can reject the reservation to free up the reservation tiles and let other vehicles to reserve them. The vehicle can then issue another reservation request as soon as possible and hopefully get a better arrival time and velocity from the intersection manager.

Experimental Evaluation

We call the driver agent using the optimization procedure and the validation procedure a *planning-based* driver agent, since it uses motion planning techniques to evaluate the arrival time and velocities before reaching the intersection. To evaluate the planning-based driver agent we implemented it in the AIM simulator and conducted an experiment to compare it with the driving agent based on the optimistic/pessimistic heuristic implemented in (Dresner 2009). In this experiment, the intersection has four incoming lanes and four outgoing lanes in each of the four canonical directions. The speed limits of the lanes are set to be $25\ m/s$. The static buffer size of the vehicles are set to be $0.25m$, which is sufficient for the simulated vehicles in the simulator. Other parameters of the autonomous vehicles are: the internal time buffer is $0s$, the edge time buffer is $0.25s$, and the maximum acceleration is $4\ m/s$. Then we vary the traffic level of each lane from 0.1 vehicles per second to 0.3 vehicles per second, and at each traffic level we run the simulator for one hour (simulated time) and compute the average delay of the vehicles.

The result of the experiment is shown in Figure 6. From the figure, we can see that when the traffic level is below 0.15 vehicles per seconds, most vehicles can get through the intersection without stopping (i.e., average delay is almost 0) and there is little difference between the performance of both driver agents. However, when the traffic level is more than 0.15 vehicles per seconds, the average delay of our planning-based driver agents is much lower than the average delay of the driver agents based on the optimistic/pessimistic heuristic. When the average delay of the heuristic-based driver agents levels off at the 0.25 traffic rate (which indicates that the throughput of the intersection has been saturated), the average delay of the planning-based driver agents remains low. Thus the use of our planning-based controller can increase the maximum throughput of the intersection and reduce the average delay.

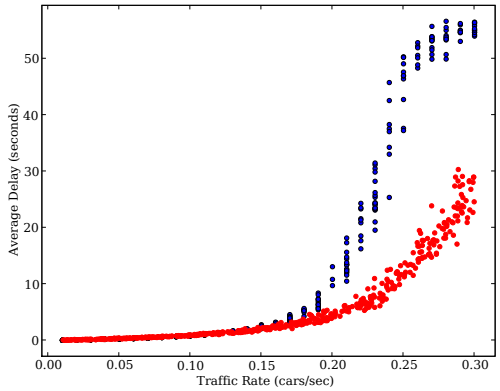


Figure 6: Comparison of the planning-based driver agent (red dots) with the driver agent based on the optimistic/pessimistic heuristic (purple dots).

Related Work

Intelligent Transportation Systems (ITS) is a multidisciplinary field concerns with advancing modern transportation systems with information technology (Bishop 2005). A noticeable research project on ITS is the Berkeley PATH project, which proposed a fully-automated highway system (Alvarez and Horowitz 1997). But most of the existing work on ITS focus on how to assist human drivers in the existing transportation infrastructure, and do not assume vehicles are driven autonomously by computer. Hence, most of the tools developed by transportation engineer (e.g., TRANSYT (Robertson 1969) and SCOOT (Hunt et al. 1981)) aim to optimize traffic signals rather than substitute them with a better mechanism. For intersection management, there are many work on the problem of intersection collision avoidance (Lindner, Kressel, and Kaelberer 2004; Naumann and Rasche 1997; Rasche et al. 1997; Naumann, Rasche, and Tacken 1998; Reynolds 1999; USDOT 2003). But none of these work concerns with autonomous vehicles. Balan and Luke presented a history-based traffic control (Balan and Luke 2006) that is potentially applicable to autonomous vehicles. Queueing theory has been widely used in traffic analysis (Mannering, Washburn, and Kilareski 2008). Our analysis emphasizes how microscopic control of autonomous vehicles (via planning techniques) could affect the throughput of an intersection. Motion planning is an important subject in robotics and control theory. When compared with existing work in motion planning (e.g, rapidly-exploring random trees (LaValle 1998;

LaValle and James J. Kuffner 2000)), our motion planning algorithms make use of the non-existence of solutions to improve the throughput in autonomous intersection management.

Conclusions and Future Work

The DARPA Urban Challenge in 2007 showed that fully autonomous vehicles are technologically feasible with current intelligent vehicle technology (DARPA 2007). Some researchers predict that within 5–20 years there will be autonomous vehicles for sale on the automobile market. Therefore the time is right to rethink our current transportation infrastructure, which is designed solely for human drivers. Dresner and Stone proposed to substitute traffic signals and stop signs for a new intersection control mechanism, namely FCFS, that takes advantages of the capability of autonomous vehicles, and demonstrated its effectiveness in simulation (Dresner and Stone 2008). In this paper we show that the efficiency of FCFS can be improved by using better vehicle controllers with motion planning techniques that takes reservation parameters into account. We explicated the relationship between the throughput of an intersection and various parameters of the intersection and vehicles via Little's law, and proposed planning-based techniques to increase the throughput of an intersection. These findings allow us to implement specific improvements to our autonomous vehicle with the goal of achieving better reservations in AIM. In the future, we intend to modify and implement the algorithms for real autonomous vehicles, and evaluate them in the real world settings.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104 and IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030).

References

Alvarez, L., and Horowitz, R. 1997. Traffic flow control in automated highway systems. Technical Report UCB-ITS-PRR-97-47, University of California, Berkeley, Berkeley, California, USA.

Balan, G., and Luke, S. 2006. History-based traffic control. In *Proceedings of the International Joint Conferenceon Autonomous Agents and Multi Agent Systems (AAMAS)*, 616–621.

Bishop, R. 2005. *Intelligent Vehicle Technology and Trends*. Artech House.

DARPA. 2007. DARPA Urban Challenge. `http://www.darpa.mil/grandchallenge/index.asp`.

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

Dresner, K. 2009. *Autonomous Intersection Management*. Ph.D. Dissertation, The University of Texas at Austin.

Hunt, P. B.; Robertson, D. I.; Bretherton, R. D.; and Winton, R. I. 1981. SCOOT - a traffic responsive method of co-ordinating signals. Technical Report TRRL-LR-1014, Transport and Road Research Laboratory.

LaValle, S. M., and James J. Kuffner, J. 2000. Rapidly-exploring random trees: progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, 293–308.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept, Iowa State University.

Lindner, F.; Kressel, U.; and Kaelberer, S. 2004. Robust recognition of traffic signals. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2004)*.

Little, J. D. C. 1961. A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research* 9(3):383–387.

Mannering, F. L.; Washburn, S. S.; and Kilareski, W. P. 2008. *Principles of Highway Engineering and Traffic Analysis*. Wiley, 4 edition.

Naumann, R., and Rasche, R. 1997. Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of the 30th ISATA - ATT/IST Conference*.

Naumann, R.; Rasche, R.; and Tacken, J. 1998. Managing autonomous vehicles at intersections. *IEEE Intelligent Systems* 13(3):82–86.

Rasche, R.; Naumann, R.; Tacken, J.; and Tahedl, C. 1997. Validation and simulation of decentralized intersection collision avoidance algorithm. In *Proceedings of IEEE Conference on Intelligent Transportation Systems (ITSC 97)*.

Reynolds, C. W. 1999. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference*, 763–782.

Robertson, D. I. 1969. TRANSYT — a traffic network study tool. Technical Report TRRL-LR-253, Transport and Road Research Laboratory.

Squatriglia, C. 2010. Audi's robotic car drives better than you do. `http://www.wired.com/autopia/2010/03/audi-autonomous-tts-pikes-peak`.

USDOT. 2003. Inside the USDOT's 'intelligent intersection' test facility. Newsletter of the ITS Cooperative Deployment Network. Accessed online 17 May 2006 at `http://www.ntoctalks.com/icdn/intell_intersection.php`.