Accepted for publication in the 2012 IEEE International
Conference on Robotics and Automation (ICRA 2012)

Setpoint Scheduling for Autonomous Vehicle Controllers

Tsz-Chiu Au, Michael Quinlan, and Peter Stone

***Abstract*— This paper considers the problem of controlling an autonomous vehicle to arrive at a specific position on a road at a given time and velocity. This ability is particularly useful for a recently introduced autonomous intersection management protocol, called AIM, which has been shown to lead to lower delays than traffic signals and stop signs. Specifically, we introduce a setpoint scheduling algorithm for generating setpoints for the PID controllers for the brake and throttle actuators of an autonomous vehicle. The algorithm constructs a feasible setpoint schedule such that the vehicle arrives at the position at the correct time and velocity. Our experimental results show that the algorithm outperforms a heuristic-based setpoint scheduler that does not provide any guarantee about the arrival time and velocity.**

I. INTRODUCTION

Recent developments in robotic vehicles lead us to believe that fully autonomous vehicles will be widely adopted in the future. Looking ahead to the time when such 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 capacities of such autonomous vehicles we can devise a reservation-based intersection control protocol that is much more efficient than traffic signals and stop signs [1]. The protocol, however, relies on the assumption that autonomous vehicles can always arrive at the intersection at a specific time and a specific velocity—if a vehicle enters the intersection at a different time and velocity, collisions may occur.

To compensate for the sensing and control errors that could cause a vehicle to fail to meet that requirement, the intersection manager allows each vehicle to have a *buffer*—an area around the vehicle that no other vehicle can enter at any time in the intersection, such that even if the vehicles deviate from their given arrival times and arrival velocities, there is enough room to avoid collisions. The problem is that the buffer cannot be too large; if it is, the vehicle will need a lot of space in the intersection and will prevent other vehicles from using the space, causing a tremendous decrease in the efficiency of the protocol, as demonstrated by the mixed reality simulation conducted by Quinlan, et al. [2].

Au et al. developed a motion planning algorithm for autonomous vehicles to arrive at the intersection at a specific time and velocity [3], thus allowing a much smaller buffer size. The motion planning algorithm, however, is based on a mathematical model of vehicle control that is too simplistic when compared with the control of a real vehicle. For example, the algorithm assumes vehicles can maintain a linear acceleration until arriving at a given velocity, but the acceleration can be far from linear if the vehicles are controlled by PID controllers, especially when starting from a stationary position or decelerating after a sharp brake. Therefore, the algorithm does not achieve its intended effects on a real autonomous vehicle.

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

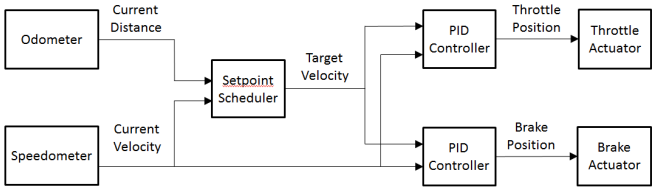


Fig. 1. The architecture of the controller for the brake and throttle actuators.

In this paper, we present a new motion planning algorithm called a *setpoint scheduler* that is based on a more realistic model of vehicle control. The model is built via empirical performance profiling of the PID controllers for the brake and throttle actuators of a vehicle. In addition, we develop a *smoothing* technique for computing a sequence of setpoints that allows the vehicle to slow down gracefully without hitting the brake too hard. The setpoint scheduler searches for a feasible trajectory for the vehicle based on a descriptive model of the PID controllers' performance. We implemented the setpoint scheduler on an autonomous vehicle, and experimentally compare it with the vehicle's default PID-based reactive controller that heuristically computes the setpoints based on the given arrival time and velocity.

II. MODELING VEHICLE PERFORMANCE

Our setpoint scheduler is designed for controlling the Austin Robot Technology vehicle, an Isuzu VehiCross that has been upgraded to run autonomously [4]. Fig. 1 shows the architecture of the controller of the brake and throttle actuators. The setpoint scheduler takes the measures from the odometer and the speedometer and computes the target velocity as the setpoint for the PID controllers which control the positions of the brake and throttle.

Our setpoint scheduler needs to know the effect of setting a setpoint in order to predict the movement of the vehicle. Given the current velocity v and a target velocity $\hat{v}$, the scheduler needs to know how long the PID controllers will take to stabilize the velocity of the vehicle at $\hat{v}$ after setting the setpoint to $\hat{v}$, and how much the vehicle will move before its velocity is stabilized. Thus our approach relies on the estimation of two functions T_{stable} and D_{stable}, where $T_{\mathsf{stable}}(v,\hat{v})$ is the *longest* time the vehicle takes to stabilize at $\hat{v}$ and $D_{\mathsf{stable}}(v,\hat{v})$ is the *average* distance the vehicle travels after setting the setpoint to $\hat{v}$ for a period of $T_{\mathsf{stable}}(v,\hat{v})$. We

call $T_{\text{stable}}(v,\hat{v})$ and $D_{\text{stable}}(v,\hat{v})$ the *stable time* and the *stable distance*, respectively. The *performance model* of the vehicle is the pair $(T_{\text{stable}}, D_{\text{stable}})$.

We construct the performance model as follows. First, the values of the initial velocity v and the target velocity $\hat{v}$ are tested between 0 m/s and 10 m/s, at increments of 0.5 m/s. For each pair of v and $\hat{v}$, we accelerate the vehicle to v and then set the setpoint to $\hat{v}$ at time t. Then we measure the time the vehicle takes to get to $\hat{v}$ and stabilize at $\hat{v}$. The criteria for stabilization is as follows: we record the time t' at which the error in velocity is less than 0.2 m/s (i.e., $|v'-\hat{v}|<0.2$ where v' is the current velocity). Then we check to see whether the velocity error still continues to be less than 0.2 m/s in the next 4 seconds. If it is the case, the stable time is $t'-t$; otherwise, we wait until the velocity error is less than 0.2 m/s again and stabilizes for 4 seconds.

For each pair of v and $\hat{v}$, we repeat the measurement five times, and choose the *maximum* value to be the stable time (denoted by $T_{\text{stable}}(v,\hat{v})$) of the vehicle when changing the velocity from v to $\hat{v}$. Then we use $T_{\text{stable}}(v,\hat{v})$ to measure the *stable distance* as follows. Once again, we control the vehicle to make it run at velocity v, and then set the setpoint to $\hat{v}$ at time t. Then we measured the distance the vehicle travels between t and $t+T_{\text{stable}}(v,\hat{v})$. The measurement was repeated five times, and we call the *average* of the measured distances the stable distance $D_{\text{stable}}(v,\hat{v})$.

Since we measured the stable time and distance at certain velocity points only, we use bilinear interpolation to estimate the values between the measured values. The results are shown in Fig. 3(a) and Fig. 3(b). Note that this particular model is specific to our vehicle, but the method of generating it is fully general.

III. SMOOTHING

Fig. 3(a) shows that the vehicle takes a long time to stabilize when the vehicle decelerates from a high speed. For example, if the vehicle decelerates from 9 m/s to 2 m/s, it will take 4.7 s to stabilize and the stable distance is 19.3 m. This problem is due to overshoot and characteristics of vehicle dynamics, as shown in the dashed line in Fig. 2. While the stable time depends on the tuning of the brake PID controller and the physical properties of the actuators, there is a way to optimize the stable time without retuning the PID controller.

The basic idea is to generate a sequence of intermediate setpoints to avoid making abrupt decreases of velocities. First, use T_{stable} to create a graph (N,E) in which the set of nodes N are the set of velocities (we choose the velocity between 0 m/s and 10 m/s at an increment of 0.1 m/s) and the edge $(v_1,v_2)\in E$ represents the stable time $T_{\text{stable}}(v_1,v_2)$. Second, the nodes on the *shortest path* between two velocities on the graph is exactly the sequence of setpoints that would minimize the stable time and provide a *smooth* transition of the velocities. Thus we used the Floyd-Warshall algorithm to compute the *all pairs shortest path* and store the shortest path structure in the vehicle's memory for repeated use. We denote the new stable time after smoothing by $\bar{T}_{\text{stable}}(v_1,v_2)$ and the new stable distance

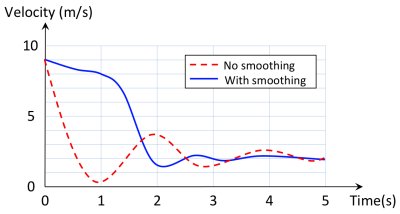


Fig. 2. Reduce the magnitude of overshoot by a sequence of intermediate setpoints generated by our smoothing procedure.

by $\bar{D}_{\text{stable}}(v_1,v_2)$. Instead of setting the setpoint to v_2 directly, the vehicle should use a setpoint schedule $\Pi(t_1,v_1,v_2)=\langle(t_1,v'_1),(t_2,v'_2),\ldots,(t_n,v'_n)\rangle$, where t_1 is the current time, $t_i=t_{i-1}+T_{\text{stable}}(v'_{i-2},v'_{i-1})$ for $1<i\le n$, and $\langle v'_0,v'_1,\ldots,v'_n\rangle$ is the nodes of the shortest path, where $v'_0=v_1$ and $v'_n=v_2$.

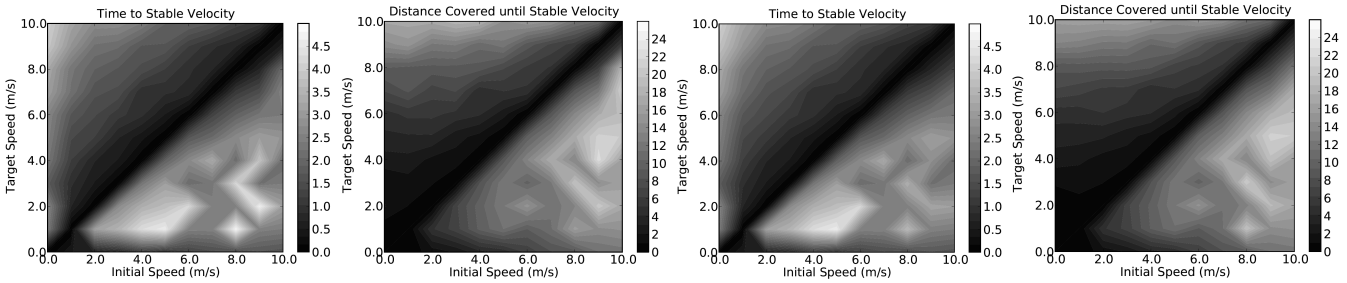
The results of the smoothing are shown in Fig. 3(c) and Fig. 3(d). After smoothing, the stable times become smaller in most cases, but the stable distances remain nearly the same. When a vehicle decelerates from 9 m/s to 2 m/s with smoothing, the smoothing procedure generates a sequence of setpoints: $\langle 8.3, 8.0, 2.0\rangle$. If the vehicle follows this setpoint sequence, the stable time is 3.2 s (a 32% decrease) and the stable distance is 19.0 m. As can be seen in Fig. 2, gradual decrease of the setpoints can reduce the duration of errors and help stabilize the vehicle at the target velocity earlier.

IV. SETPOINT SCHEDULING PROBLEMS

The goal of modeling vehicle performance is to enable long-term planning of vehicle's movement *without* knowing the details of vehicle dynamics and controls. This separation of high-level planning issues from the concerns of lower-level vehicle controls enables our planning procedures, called *setpoint schedulers*, to work with a wide variety of vehicle hardwares with different underlying control mechanisms.

Here we define the setpoint scheduling problems for AIM. We consider the case in which a vehicle is on a straight road and is moving towards an intersection that is D meters away from the position of the vehicle. We say the position of the vehicle is the *initial* position and the entrance of the intersection from the road is the *target* position. Let t_0 and v_0 be the *initial time* and the *initial velocity* of the vehicle at the initial position, respectively. At the initial position, according to the AIM protocol, the vehicle proposes an arrival time t_{end} and an arrival velocity v_{end} in the reservation request sent to the intersection manager (IM), the server located at the intersection which handles the reservation requests. The IM will then check whether the trajectory of the vehicle in the intersection will collide with other vehicles' trajectories. The IM can either reject the request if collisions may occur, or grant the reservation with which the vehicle is permitted to enter the intersection at the proposed arrival time and velocity. See [1] for more details about AIM.

Upon receiving a successful reservation, the vehicle *must* arrive at the intersection at time in $[t_{end}-t_{buf}, t_{end}+t_{buf}]$ and at velocity in $[v_{end}-v_{buf}, v_{end}+v_{buf}]$, where t_{buf} is the *time buffer* and v_{buf} is the *velocity buffer*, that are used to account for control and sensing errors of the vehicle. During the traversal towards the intersection, the vehicle constantly



(a) Stable Time(no smoothing) (b) Stable Distance (no smoothing) (c) Stable Time (with smoothing) (d) Stable Distance (with smoothing)

Fig. 3. Stable time and stable distance. The light color means longer time and distance, as indicated in the bars beside the graphs.

checks to see whether it can still arrive at the intersection at t_{end} and v_{end}, because the vehicle may deviate from the expected trajectory due to the imperfect road surface and accumulated sensing and control errors. If the vehicle finds that it can no longer arrive at t_{end} and v_{end}, it *cancels* the reservation and resends another reservation request, ensuring that it has time to stop before the intersection in the event that it cannot get a new reservation.

From a high level perspective, the vehicle deals with two problems: 1) proposing an arrival time and an arrival velocity such that the vehicle's arrival time is the soonest while the arrival velocity is the highest; and 2) deciding whether the arrival time and velocity remains *feasible* during traversal—whether the vehicle can still arrive at the given arrival time and velocity. We call the former problem the *optimization* problem while the latter the *validation* problem. The solutions to these problems subject to the speed limit v^{max} of the road and the speed limit v_{end}^{max} of the trajectory for the vehicle to safely traverse the intersection. Note that v_{end}^{max} depends on the turn direction of the vehicle at the intersection; if the vehicle makes a right turn at the intersection, v_{end}^{max} has to be very small in order to avoid strong centrifugal forces acting on the passengers inside the vehicle. If the vehicle makes a left turn, v_{end}^{max} can be larger since the turn is not as sharp as the right turn. If the vehicle goes straight through the intersection (no turn), v_{end}^{max} can be as high as v^{max}.

The solutions to both problems depend on the notion of *control signals*—the signals that the controller generates to control the vehicle. In our blackbox approach, the control signals are the setpoints (target velocity in our autonomous vehicle) that the PID controllers of the brake and throttle actuators need. We denote a *setpoint schedule* by τ. If τ is a step function, τ can be represented by a list of pairs $\langle(t_0,\hat{v}_0),(t_1,\hat{v}_1),\ldots(t_n,\hat{v}_n)\rangle$. where $\hat{v}(t)=\hat{v}_i$ for (1) $t_i \le t < t_{i+1}$ for $0 \le i < n$ and (2) $t_i \le t$ for $i=n$. In this paper, all setpoint schedules are step functions.

We formulate this optimization problem as a multiobjective optimization problem: among all possible setpoint schedules 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, subject to the speed limit constraints. We choose arrival velocity as the primary objective, because traveling at high velocity through the intersection consumes less of the space-time resource of the intersection [2], [3]. Thus, 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.

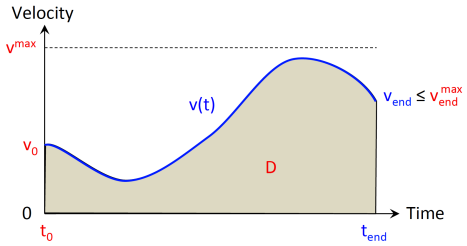


Fig. 4. The time-velocity diagram.

To visualize what we are trying to achieve for the optimization problem, see the time-velocity diagram in Figure 4. In this diagram, a function denotes the velocity of the vehicle over time, and is called a *velocity function*. A velocity function $v(\cdot)$ is *constructible* if there exists a setpoint schedule $\tau(\cdot)$ such that if the vehicle follows $\tau(\cdot)$, the velocity of the vehicle is $v(\cdot)$. A velocity function $v(\cdot)$ is *feasible* if it satisfies the following constraints:

1) $v(t_0)=v_0$;
2) $v(t_{end}) \le v_{end}^{max}$ (i.e., the arrival velocity cannot exceed the speed limit of the trajectory);
3) $0 \le v(t) \le v^{max}$ for $t_0 \le t \le t_{end}$ (i.e., the velocity cannot exceed the speed limit of the road or be negative at any point in time);
4) $\int_{t_0}^{t_{end}} v(t)\,dt = D$, where t_{end} is the arrival time (i.e., the distance traveled must be D); and
5) $v(\cdot)$ is constructible.

A setpoint schedule $\tau(\cdot)$ is *feasible* if the velocity function constructed by $\hat{v}(\cdot)$ is feasible.

The objective of the optimization problem is to find a feasible setpoint schedule $\tau(\cdot)$ such that $v(t_{end})$ is as high as possible while t_{end} is as small as possible. The validation problem, however, does not need to estimate the arrival time and velocity, as they are given. The objective of the validation problem is to decide whether a feasible setpoint schedule $\tau(\cdot)$ exists such that $v(t_{end})=v_{end}$.

V. PLAN-BASED SETPOINT SCHEDULER

Our setpoint scheduler works as follows. At the initial position, the setpoint scheduler calls *the optimization procedure* to generate a setpoint schedule for the vehicle to arrive at the intersection at the highest possible velocity (usually v_{end}^{max})

while the arrival time is as small as possible. The setpoint schedule is stored in memory and is used whenever it is time to set the setpoint to a new target velocity. The scheduler revises the setpoint schedule from time to time by using *the validation procedure*, which checks to see whether a setpoint schedule still exists to allow the vehicle to arrive at the target position at t_{end} and v_{end}. If the validation procedure finds a new setpoint schedule, it replaces the old one in memory with the new one; otherwise, the driving agent cancels the reservation and send a new reservation request.

A. The Optimization Procedure

The optimization procedure takes t_0, v_0, v^{max}, v^{max}_{end} and D as inputs, and generates an optimal setpoint schedule in terms of arrival time and velocity. There are two different cases to consider. In Case 1, the vehicle *can* accelerate to v^{max} and then decelerate to v^{max}_{end} right before the target positon. The condition for Case 1 is $\bar{D}_{\mathsf{stable}}(v_0, v^{max}) + \bar{D}_{\mathsf{stable}}(v^{max}, v^{max}_{end}) \leq D$, where $\bar{D}_{\mathsf{stable}}$ is the stable distance after smoothing, meaning that the vehicle is far enough from the target position to accelerate to v^{max} and then decelerate to v^{max}_{end}. If this condition holds, the driver agent requests a reservation with arrival time t_{end} and arrival velocity $v_{end} = v^{max}_{end}$, and puts the following setpoint schedule in the memory: $\tau_{opt} = \Pi(t_0, v_0, v^{max}) \oplus \Pi(t_1, v^{max}, v^{max}_{end})$, where Π is the setpoint schedule after smoothing, $\oplus$ is the concatenation operator of setpoint schedules, $t_1 = t_0 + \bar{T}_{\mathsf{stable}}(v_0, v^{max}) + D'/v^{max}$, $D' = D - \bar{D}_{\mathsf{stable}}(v_0, v^{max}) - \bar{D}_{\mathsf{stable}}(v^{max}, v^{max}_{end})$, and $t_{end} = t_1 + \bar{T}_{\mathsf{stable}}(v^{max}, v^{max}_{end})$.

Obviously the arrival velocity is optimal. But it is not clear whether the arrival time is the smallest, because there may exist $t'_{end} < t_{end}$ such that the validation procedure can generate a setpoint schedule. In general, it is hard to check whether such t'_{end} exists since $\bar{D}_{\mathsf{stable}}$ can be any function. But if the following equation is a strictly increasing function for any given t_{end} and v_{end}, we can show that t_{end} is optimal:

$$\mathscr{D}(\hat{v}) = \bar{D}_{\mathsf{stable}}(v_0, \hat{v}) + D_0 + \bar{D}_{\mathsf{stable}}(\hat{v}, v_{end}), \qquad (1)$$

where $D_0 = \hat{v} \times ((t_{end} - t_0) - \bar{T}_{\mathsf{stable}}(v_0, \hat{v}) - \bar{T}_{\mathsf{stable}}(\hat{v}, v_{end}))$. Due to space limitations we omit the proof of the optimality under this condition. But the idea of the proof is that $\mathscr{D}(\hat{v}_m)$ is the distance covered by the vehicle if it accelerates to the *traversal velocity* $\hat{v}$, maintains the velocity for a while, and then decelerate to v_{end} right before reaching the target position. If $\mathscr{D}(\cdot)$ is strictly increasing, the only way to arrive at the target position at v_{end} sooner is to increase the traversal velocity. But in our case, the traversal velocity is v^{max}, and the vehicle cannot go above the speed limit of the road. In practice, even if $\bar{D}_{\mathsf{stable}}$ does not satisfy this condition, t_{end} is often very close to optimal.

In Case 2, the condition $\bar{D}_{\mathsf{stable}}(v_0, v^{max}) + \bar{D}_{\mathsf{stable}}(v^{max}, v^{max}_{end}) \leq D$ fails because the vehicle is too close the intersection. In this case, the vehicle cannot accelerate to v^{max}, but must either (1) accelerate to certain traversal velocity less than v^{max} and then decelerate immediately to v^{max}_{end}, or (2) reach the target position at a velocity below v^{max}_{end}. Currently, our scheduler handles these cases in an ad-hoc way: consider each velocity $\hat{v}$ in the range $[0, v^{max}]$ at an increment of 0.5 m/s in decreasing order, test to see if the vehicle can accelerate to $\hat{v}$, and then decelerate v^{max}_{end}. If no schedule is found, call the optimization procedure repeatedly with a smaller and smaller v^{max}_{end}, until it returns a schedule. If no schedule can be found, the driver agent declares no reservation can be made.

B. The Validation Procedure

Apart from t_0, v_0, v^{max}, v^{max}_{end}, the validation procedure is also given t_{end}, v_{end}, and the current distance d ($0 \leq d \leq D$) from the target position as inputs. Like the optimization procedure, the validation procedure searches for a setpoint schedule to reach the target position. But the difference is that there is no optimization since t_{end} and v_{end} are given; all it does is to show that a *feasible* setpoint schedule exists.

Unfortunately, it is hard to search for a feasible setpoint schedule in the space of all possible setpoint schedules, since setpoints are real numbers and the structure of $\bar{D}_{\mathsf{stable}}$ can be quite complicated. Thus, our validation procedure only considers a class of *simple* setpoint schedules, each of which has the form $\tau_{simple}(\hat{v}) = \Pi(t_0, v_0, \hat{v}) \oplus \Pi(t_1, \hat{v}, v_{end})$, where $t_1 = t_{end} - \bar{T}_{\mathsf{stable}}(\hat{v}, v_{end})$ and $\hat{v}$ is the traversal velocity. Given a simple setpoint schedule, a vehicle will first accelerate to $\hat{v}$, maintain its speed at $\hat{v}$, and then decelerate at t_1 to reach the target position at t_{end} and v_{end}. We claim that it is often sufficient to consider simple setpoint schedules, because according to a simplified vehicle model in [3], if a feasible velocity function exists for a given t_{end} and v_{end}, there also exists a trapezoidal velocity function with which the vehicle can arrive at t_{end} and v_{end}. Even though the velocity function constructed by a simple setpoint schedule is not exactly a trapezoidal function (since vehicles do not accelerate linearly with PID controllers), the shape is often a close approximation of a trapezoidal velocity function.

A simple setpoint schedule τ_{simple} satisfies all constraints except the distance constraint: the traversal distance is exactly d. Therefore, the validation procedure uses the *bisection method* to search for a *feasible* traversal velocity $\hat{v}^*$ such that $\mathscr{D}(\hat{v}^*) = d$, where $\mathscr{D}(\hat{v})$, defined in Eq. 1, is the distance covered by the vehicle if the traversal velocity is $\hat{v}$. Suppose $\hat{v}_a = 0$ and $\hat{v}_b = v^{max}$ such that $\mathscr{D}(\hat{v}_a) \leq d$ and $\mathscr{D}(\hat{v}_b) \geq d$. The procedure computes $\hat{v}_c = (\hat{v}_a + \hat{v}_b)/2$ and checks whether $\mathscr{D}(\hat{v}_c)$ is greater than or equal to d. If it is true, set $\hat{v}_b$ to $\hat{v}_c$; otherwise, set $\hat{v}_a$ to $\hat{v}_c$. Then repeat these steps until the width of the interval $[\hat{v}_a,\ \hat{v}_b]$ is smaller than a small threshold, such that $\hat{v}_a \approx \hat{v}_b$. Then choose $\hat{v}_a$ as $\hat{v}^*$ and return $\tau_{simple}(\hat{v}^*)$.

The bisection method converges to a solution if (1) $\mathscr{D}(0) \leq d$, (2) $d \leq \mathscr{D}(v^{max})$, and (3) $\mathscr{D}(\cdot)$ is a continuous function. If $\mathscr{D}(0) > d$, the vehicle is too close to the target position. If $\mathscr{D}(v^{max}) < d$, the vehicle is too far away from the target position. $\mathscr{D}(\cdot)$ is often a continuous function because a small increase of the traversal velocity usually does not lead to a dramatic change of the traversal distance. Even if $\mathscr{D}(\cdot)$ is not a continuous function, the bisection method often operates in the domain of $\mathscr{D}(\cdot)$ in which $\mathscr{D}(\cdot)$ is continuous. In practice, the procedure returns a solution with no perceptible delay.

VI. EMPIRICAL EVALUATION

The experiments are designed to compare the plan-based setpoint scheduler to the naïve PID-based setpoint controller implemented on our autonomous vehicle.

A. Naïve Setpoint Controller

The naïve setpoint controller is a PID-based reactive controller for controlling the setpoint of the underlying PID controllers of the brake and throttle. This controller offers no guarantee that the car will arrive at either the correct time or correct velocity. However, in practice the naïve setpoint controller offers good performance when supplied with a feasible arrival time, in particular when seeded with the output of our optimization procedure.

The naïve setpoint controller is given an arrival velocity (v_{end}) and an arrival time (t_{end}), and at each time step it receives both the current distance to the intersection (d) and the current time (t). The controller continually outputs a velocity (v_{out}), which is used as the setpoint by the PID controllers of the brake and throttle. The objective of the controller is to minimize the difference between v_{out} and v_{end} such that when the vehicle arrives at the intersection at time t_{end}, the velocity is exactly v_{end}. At each time step, the controller calculates the velocity $v_{time} = d/(t_{end} - t)$ which will result in the car exactly matching the arrival time. Then the term $e = v_{end} - v_{time}$ is used as the error signal of the difference between v_{out} and v_{end}. Then the controller computes $u(e) = PD(e)$, where PD is a PD controller with parameters $k_p = 1.8$ and $k_d = 0.05$. The setpoint for the PID controllers of the brake and throttle is $v_{out} = v_{end} + u(e)$.

B. Experimental Setup

Our experiments were conducted in Stage using ROS [5] packages developed for our autonomous vehicle. Stage was modified to more accurately model the dynamics of the robot. We added calculations that approximate the drag and rolling resistance of the vehicle and also model the brake and throttle actuator lag. In our experiments the road has a speed limit of 10 m/s (v^{max}), which matches the speed limit of the test campus used for our autonomous vehicle. The vehicle requests a reservation at 100 m (D) from the intersection.

We ran the experiments at three different starting velocities (v_0) and at three different intersection arrival velocities (v_{end}^{max}). The velocities correspond to the three possibilities of the car traversing an intersection, straight ahead at 9.0 m/s, turning left at 6.0 m/s and turning right at 3.0 m/s. Each combination is run 30 times, with the error in arrival velocity and arrival time being measured. A negative velocity error indicates that the vehicle was going too slowly when entering the intersection. A negative time error indicates that the vehicle entered the intersection too early.

C. Experimental Results

Both the plan-based setpoint scheduler and the naïve setpoint controller use the optimization procedure presented in Section V-A to calculate an optimal arrival time (t_{end}). When the optimization procedure is called we supply a slightly lower speed limit of the road: $v^{max} - C$. This buffer C gives an opportunity for the vehicle to react to noise and control errors. Without the buffer, if the vehicle got even a millisecond behind the schedule it would not be able to catch up, as that would require the vehicle to go faster then v^{max}. This arrival time produced by the optimization procedure is guaranteed to be reachable so both controllers should be capable of arriving within some error range.

TABLE I

PERFORMANCE OF THE NAÏVE SETPOINT CONTROLLER (VELOCITY). OVERALL ERROR MAGNITUDE 0.182 ± 0.013 (M/S)

v_0 \ v_{end}	3	6	9
3	0.367 ± 0.059	-0.097 ± 0.034	-0.139 ± 0.005
6	0.300 ± 0.055	-0.108 ± 0.032	-0.142 ± 0.004
9	0.325 ± 0.049	-0.095 ± 0.035	-0.140 ± 0.006

TABLE II

PERFORMANCE OF THE NAÏVE SETPOINT CONTROLLER (TIME). OVERALL ERROR MAGNITUDE 0.172 ± 0.015 SECONDS

v_0 \ v_{end}	3	6	9
3	-0.386 ± 0.020	0.040 ± 0.034	0.098 ± 0.017
6	-0.359 ± 0.023	0.020 ± 0.019	0.113 ± 0.019
9	-0.378 ± 0.020	0.044 ± 0.021	0.080 ± 0.013

TABLE III

PERFORMANCE OF THE PLAN-BASED SETPOINT SCHEDULER (VELOCITY). OVERALL ERROR MAGNITUDE 0.078 ± 0.008 (M/S)

v_0 \ v_{end}	3	6	9
3	-0.003 ± 0.043	-0.058 ± 0.006	-0.139 ± 0.005
6	-0.051 ± 0.050	-0.058 ± 0.003	-0.140 ± 0.004
9	-0.019 ± 0.008	-0.059 ± 0.003	-0.080 ± 0.003

TABLE IV

PERFORMANCE OF THE PLAN-BASED SETPOINT SCHEDULER (TIME). OVERALL ERROR MAGNITUDE 0.181 ± 0.015 SECONDS

v_0 \ v_{end}	3	6	9
3	-0.005 ± 0.093	0.130 ± 0.033	0.072 ± 0.022
6	0.039 ± 0.093	0.131 ± 0.033	0.066 ± 0.022
9	0.439 ± 0.060	0.306 ± 0.032	0.157 ± 0.016

Tables I and II present the results for the naïve setpoint controller. It can be seen that on average the vehicle arrived at the intersection within 0.182 ± 0.013 m/s (95% confidence interval) of the intended velocity and within 0.172 ± 0.015 seconds of the estimated arrival time. The results for the plan-based setpoint controller are shown in Tables III and IV. The vehicle arrived at the intersection with a mean velocity error of 0.078 ± 0.008 m/s (95% confidence interval) and a mean arrival time error of 0.181 ± 0.015 seconds. Both approaches arrive with sufficiently small errors in time and velocity. Although the differences in the arrival time errors are insignificant, the plan-based setpoint scheduler performs significantly better in matching the arrival velocity, and reduces the velocity error by approximately 50%.

Further analysis of the results indicates that the noise present in the robot and the environment impact each approach differently. The plan-based setpoint scheduler relies on accurate $\bar{T}_{\text{stable}}$ and $\bar{D}_{\text{stable}}$ tables and on accurate velocity measurement. In theory the tables for $\bar{T}_{\text{stable}}$ and $\bar{D}_{\text{stable}}$ were

calculated to include standard errors, however there exist some hidden sources of error. For example, the setpoint schedule may request a change in velocity at 0.332 seconds, but the actuator runs at 10 Hz and therefore the effects may not take place until 0.4 seconds have passed. At 10 m/s that represents an error of 0.68 m. In addition the plan-based setpoint scheduler assumes that in between setpoints the vehicle is driving at exactly the correct velocity, but in reality the velocity is slightly different. But most of these errors can be fixed by the validation procedure. The errors indicated in Tables I–IV are mainly caused by the vehicle reaching a point sufficiently close to the intersection in which the validation procedure can no longer adapt to the errors. These errors are most likely to occur during plans that require large accelerations or decelerations, such as the drop from 9 m/s to 3 m/s (as can be seen in Table IV).

VII. RELATED WORK

Our controllers can be considered as a type of longitudinal control of autonomous/semi-autonomous vehicles, which has been widely studied since the 1960's, in particular in platooning in automated highway systems [8], [9], [10]. Studies in the 80s and 90s mainly focus on car following in a platoon [11], but our approach is more suitable for point following [12]. Finding optimal arrival times and velocities is an important issue in AIM but not in platooning.

There has been work on motion planning for autonomous vehicle (e.g., [6]), but most of it has treated the arrival time and velocity requirements as secondary. While sampling-based motion planning algorithms such as RRT [13] and its variants [14] can deal with both arrival time and velocity, they cannot determine the infeasibility of a given arrival time and velocity before the vehicle actually arrives at the target position. The detection of such infeasibility, however, can be exploited to enhance intersection efficiency under AIM [3]. Our original validation procedure can prove non-existence of motion plans under a simplified model of motions [3]. The validation procedure presented here cannot guarantee the non-existence of solutions but is good enough for AIM.

Our approach treats the low-level controls as a blackbox and focusses on the planning issues such as how to meet the arrival requirements and optimize arrival time/velocity. This allows our approach to work with a wide variety of vehicle hardware and underlying low-level controller, e.g., a nonlinear H_∞ controller [15] or the adaptive algorithms for slip control were introduced to deal with unknown interactions between tires and the road surface [16]. The use of descriptive performance models can also circumvent the difficulty of modeling vehicle dynamics and powertrains.

Our work is similar to multivariable PID controllers (e.g., [7]) but PID controllers generally do not provide any optimality and arrival guarantees. The robustness of our approach relies on 1) rescheduling initiated by the validation procedure, 2) conservative estimation of the performance model which takes the sensing and control errors into account, and 3) the time and velocity buffer provided by the intersection manager.

VIII. CONCLUSIONS AND FUTURE WORK

The plan-based setpoint scheduler introduced in this paper can control an autonomous vehicle to optimally and accurately arrive at a specific point on the road in terms of both arrival time and arrival velocity. We also introduced a smoothing technique to reduce the time a vehicle takes to stabilize at certain velocity. The ability for vehicles to meet tight time and velocity requirements can be exploited to greatly enhance the throughput of intersections [2], [3]. We have shown that a reactive setpoint controller, for example our naïve setpoint controller, can provide comparable results. However such controller does not come with any guarantees, and therefore is not ideal for systems where an unexpected error in arrival time or arrival velocity may result in a collision. In the future, we intend to deal with the variance in typical driving performance due to different road conditions.

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

[2] M. Quinlan, T.-C. Au, J. Zhu, N. Stiurca, and P. Stone, "Bringing simulation to life: A mixed reality autonomous intersection," in *IEEE/RSJ International Conf. on Intelligent Robots and Systems*, 2010.

[3] 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.

[4] P. Beeson, J. O'Quin, B. Gillan, T. Nimmagadda, M. Ristroph, D. Li, and P. Stone, "Multiagent interactions in urban driving," *Journal of Physical Agents*, vol. 2, no. 1, pp. 15–30, 2008.

[5] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. of the International Conference onf Advanced Robotics (ICAR)*, 2003.

[6] E. Frazzoli, "Robust hybrid control for autonomous vehicle motion planning," Ph.D. dissertation, Massachusetts Institute of Tech., 2001.

[7] J. Bao, J. F. Forbes, and P. J. McLellan, "Robust multiloop pid controller design: a successive semidefinite programming approach," *Industrial & Engineering Chemistry Research*, vol. 9, no. 38, 1999.

[8] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, 1993.

[9] D. Godbole and J. Lygeros, "Longitudinal control of the lead car of a platoon," *IEEE Tran. on Vehicular Technology*, vol. 43, no. 4, 1994.

[10] S. Shladover, C. Desoer, J. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the path program," *IEEE Tran. on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[11] S. Sheikholeslam and C. A. Desoer, "Longitudinal control of a platoon of vehicles," in *American Control Conference*, 1990, pp. 291–296.

[12] F. Broqua, "Impact of automatic and semi-automatic vehicle longitudinal control on motorway traffic," in *Proceedings of the Intelligent Vehicles '92 Symposium*, 1992, pp. 144–147.

[13] S. M. LaValle and J. James J. Kuffner, "Rapidly-exploring random trees: progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[14] J. hwan Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proceedings of IEEE Conference on Decision and Control*, 2011.

[15] L. Ganzelmeier and E. Schnieder, "Engineering aspects of nonlinear H_∞ control for longitudinal vehicle dynamics," in *10th IFAC Symposium on Control in Transportation Systems*, 2003.

[16] H. Lee and M. Tomizuka, "Adaptive vehicle traction force control for intelligent vehicle highway systems (IVHSs)," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 1, pp. 37–47, 2003.