Steering Control of the Autonomous Vehicle: CajunBot

John Herpin[1] and Afef Fekih[2]
University of Louisiana at Lafayette, Lafayette, LA 70504, USA

Suresh Golconda[3],
University of Louisiana at Lafayette, Lafayette, LA 70504, USA

and
Arun Lakhotia[4]
University of Louisiana at Lafayette, Lafayette, LA 70504, USA

This paper describes the steering controller of the CajunBot II, an autonomous vehicle designed for the DARPA Urban Challenge. The autonomous vehicle is a modified Jeep Wrangler Rubicon equipped with INS/GPS for localization, three sets of sensors and three single board computers. The control objective is to make the lateral error at a certain point ahead of the vehicle zero. The distance of this point from the vehicle is called the look-ahead distance. A proportional Integral controller augmented with a moving average filter to dump the yaw internal dynamics is proposed to achieve this objective. The steering controller was tested under full-scale conditions and its dynamic performances are discussed here.

Nomenclature

J	=	Vehicle moment of inertia (kg*m$^{2)}$
M	=	Vehicles mass (kg)
l_f	=	Distance from the center of gravity to the front wheel axel (meters)
l_r	=	Distance from the center of gravity to the back wheel axel (meters)
L	=	$l_f + l_r$
d_s	=	Distance from center of gravity to virtual sensor point (meters)
C_r	=	Rear cornering stiffness (N/rad)
C_f	=	Front cornering stiffness (N/rad)
v	=	Vehicle velocity (m/s)
δ	=	The steering angle
ΔI	=	Instantaneous yaw rate of the vehicle
y	=	Lateral displacement of the virtual point
μ	=	Road adhesion factor (from 1 to 0)

[1] Electrical and Computer Engineering Department, P.O. Box 43890, Lafayette, LA 70504, (e-mail: john.herpin@gmail.com)
[2] Electrical and Computer Engineering Department, P.O. Box 43890, Lafayette, LA 70504 (phone: (337) 482-5333; fax: (337) 482-6568; e-mail: afef.fekih@louisiana.edu).
[3] Center for Advanced Computer Studies, P.O. Box 44330, Lafayette, LA 70504, (e-mail: suresh_golconda@yahoo.com).
[4] Center for Advanced Computer Studies, P.O. Box 44330, Lafayette, LA 70504, (phone: (337) 482-6766; fax: (337) 482-5791; e-mail: arun@louisiana.edu).

I. Introduction

CONSIDERABLE attention has been focused recently by the National Defense Authorization for the advancement of robotic technology and machine intelligence. The goal is to achieve the fielding of unmanned, remotely controlled technology such that by 2015, one-third of the operational ground combat vehicles are unmanned. The objective is to carry out dangerous missions using a machine instead of a human protecting the war fighters and allowing valuable human resources to be used more effectively.

The DARPA Urban Challenge is an autonomous vehicle research and development program with the goal of developing technology that will keep war fighters off the battlefield and out of harm's way. The Urban Challenge features autonomous ground vehicles maneuvering in a mock city environment, executing simulated military supply missions while merging into moving traffic, navigating traffic circles, negotiating busy intersections, and avoiding obstacles,[1].

CajunBot-II is an autonomous vehicle developed for the DARPA Urban Challenge. It's a 2004 Jeep Wrangler Rubicon equipped with an INS/GPS for localization and a variety of environment sensors, stereo vision, and radars,[2]. The vehicle has the electronics and sensors needed to address all the urban challenge requirements. It has the power generation capacity necessary to run the computers and electronics for an extended period of time. Its software system has the ability to utilize multiple environment sensors, including, LIDARs, radars, and cameras. Data from individual sensors is analyzed independently, and then fused. The fusion algorithm takes into account the reliability of each sensor in a particular context. The planning component of the software system consists of three layers, developed incrementally. The foundation layer provides the ability to complete a course in the absence of any obstacles. The second layer provides the ability to navigate around static obstacles. And the final layer provides the ability to work safely in traffic. The ability to follow a lane is the primary means of traveling on the road. Thus, the vehicle does not need very accurate or dense waypoints for navigation. The control system can track a path very accurately even at high speeds. The entire system integrates well to fulfill all the requirements of the urban challenge.

The CajunBot-II steering controller is discussed in this paper. The steering controller is designed based on the look-ahead approach. Look-ahead control is a predictive control strategy where information about some of the future disturbances to the controlled system is assumed to be available,[3,4]. The steering control system operates on the sensed signals so as to maintain the vehicle in a desired state. The steering controller is designed to minimize error while following a reasonable path at a reasonable speed,[4,5]. Through the use of a lead compensator, we have been able to create a steering controller that tracks a variety of paths with reasonable performance and stability, even when these paths are such that the vehicle cannot track closely given its non-holonomic constraints. Additionally, through the use of a moving-average filer, we were able to increase the smoothness of the steering.

This paper is organized as follows. Section 2 gives an overview of the CajunBot II. The dynamic model of the vehicle is described in section 3. The steering controller is discussed in section 4.The performance of the controlled vehicle is presented in section 5. Finally, section 6 gives some concluding remarks.

II. Overview of the CajunBot II

CajunBot-II is an autonomous vehicle designed for the DARPA urban challenge. It's a 2004 Jeep Wrangler Rubicon equipped with an electronic drive-by-wire system, an INS/GPS for localization, and two high-output alternators, each with maximum current output of 250 A designed for the DARPA challenge.

The autonomous vehicle is equipped with a variety of environment sensors, to detect dynamic and static obstacles, including SICK LMS LIDAR sensors, Alasca XT LIDAR sensors from Ibeo, stereo vision cameras, and radars. The algorithm used to analyze a sensor's data to extract obstacle information is intrinsically tied to the type of sensor being used. The software system is decomposed into a collection of programs communicating via a blackboard supported by a layer of middleware. Localization from the GPS and INS is improved by a Kalman filter based algorithm using a SICK LIDAR, and vehicle commands. Environment sensor data is fused using probabilities derived from the operating context. The mission is mapped into a series of maneuvers using the mission, vehicle state, and environmental information. A maneuver is applicable only on a path with certain geometry. Execution of a maneuver takes into account the dynamic driving conditions. Human driving experiences are used to catalogue the maneuvers needed to satisfy the urban challenge requirements,[5].

CajunBot-II is equipped with three computers referred to as Main, NTP, and Logger. Each is a Single Board Computer in EPIC form factor with a 1.8 GHz Pentium M. The Main computer is used for processing SICK LIDAR data, path planning, and steering. The NTP computer provides the network time protocol service for the other computers. The Logger saves all the data acquired during a run and also has the ability to broadcast the data over a wireless network. The Main and Logger computers run Fedora Core 5 (FC5). The NTP service, although light-weight, is provided by a separate machine because the PPS patch needed for NTP is available for the Linux 2.4

kernel, and not for the Linux 2.6 kernel used in FC5. The Ibeo LIDAR system and Iteris LDW have their own computing units. All the computing devices, sensors, and other equipment communicate over Ethernet, connected through a Gigabit Managed Switch from Dell.

Commands from CajunBot's computing system are actuated through a drive-by-wire system which was designed and installed by Electronic Mobility Control (EMC). Through the drive by wire system, all of the functions normally controlled through the jeep's dashboard interface can be controlled. This includes but is not limited to, starting the engine, turn signals, windshield wipers, and headlights. Additionally, the system actuates steering, throttle and gear changes.

Figure 1 gives an overview of the vehicle with its steering actuation and navigation system.

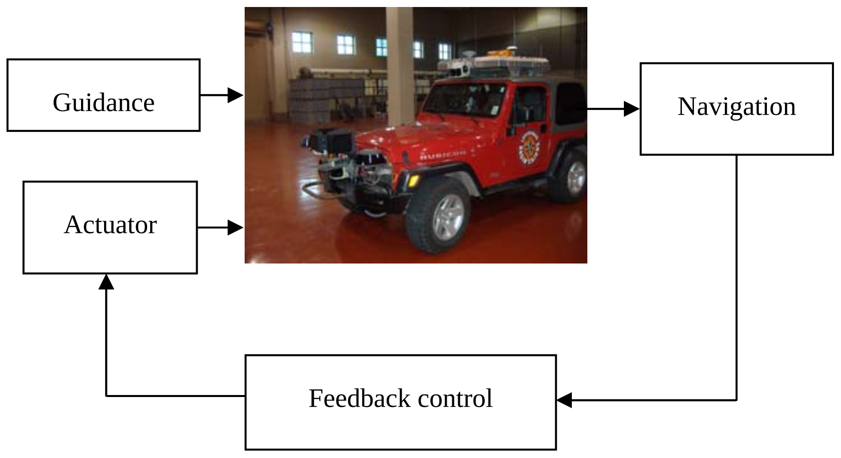

Fig 1. Feedback control of the autonomous vehicle CajunBot

A. Steering Actuation System

The steering commands from the controller are actuated via a drive by wire system which was designed and installed by EMC. The steering and throttle are each actuated by a servo motor. The position of the motor is determined by an analogue voltage input. The motor will always move at a maximum constant speed until the position is reached or a new command is implemented. The steering and throttle commands as implemented by the software are values ranging from -1.0 to +1.0, which correspond to the minimum and maximum voltages respectively. In the case of throttle, -1.0 is equivalent to a full brake while +1.0 is full throttle. In the case of steering, -1.0 is equivalent to a full left-turn while +1.0 is a full right-turn. The average ground wheel angle corresponding to a full left or right turn is 28 degrees off center. The ground wheel is assumed to change linearly with the steering wheel.

The limits of the actuation system play a significant role in the performance of the steering controller. The steering actuator introduces a delay of .24 seconds between the time when the steering command is implemented and when the actuator begins to move. Additionally, the steering actuator can only change the ground wheel angle at a rate of 15.6 degrees per second. This corresponds to 3.6 seconds to go from an extreme left to extreme right turn.

B. Navigator

The Navigator module takes as input the Trail computed by the Executive module. It is responsible for driving the vehicle along the sequence of waypoints in the Trail, keeping the vehicle as close to the given path as possible, while also maintaining the speed specified in the Trail. The Navigator uses a lead-lag compensator controller for controlling the vehicle's steering ,[3] and a PD controller for controlling its throttle and brakes.

The dynamic equations and the parameters of the vehicle are described in the following section.

III. Vehicle Dynamic Model

When the objective is to develop a steering control system for automatic lane keeping, it is useful to utilize a dynamic model in which the state variables are in terms of position and orientation error with respect to the road, [6-8]. The dynamic model of the vehicle in terms of error with respect to road is described by the following state space model,[8].

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{1}$$

Where:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{\Omega_1 + \Omega_2}{v} & \Omega_1 + \Omega_2 & \dfrac{\Omega_1 (d_s - l_f) + \Omega_2 (d_s + l_r)}{v} \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{2(C_f l_f - C_r l_f)}{Jv} & \dfrac{2(C_f l_f - C_r l_r)}{J} & -\dfrac{2(C_f (l_f^2 - l_f d_s) + C_r (l_r^2 - l_r d_s))}{Jv} \end{bmatrix} \tag{2}$$

$$B = \begin{bmatrix} 0 & 0 \\ \Omega_1 & \dfrac{\Omega_2 l_r - \Omega_1 l_f - v^2}{v} \\ 0 & 0 \\ \dfrac{2C_f l_f}{J} & -\dfrac{2\left(C_f l_f^2 + C_r l_r^2\right)}{Jv} \end{bmatrix}, \tag{3}$$

$$C = [1 \quad 0 \quad d_s \quad 0], \; D = [0 \quad 0] \tag{4}$$

$$\Omega_1 = 2C_f\left(\frac{1}{M} + \frac{l_f d_s}{J}\right), \; \Omega_2 = 2C_r\left(\frac{1}{M} + \frac{l_r d_s}{J}\right) \tag{5}$$

with:

$x = [y_s \quad \dot{y}_s \quad \omega \quad \dot{\omega}]^T$, the state vector.

$u = [\delta]^T$, the control vector

The state variables are readily available from the combination of GPS and INS. These parameters of the state space model are illustrated in figure 2 below.

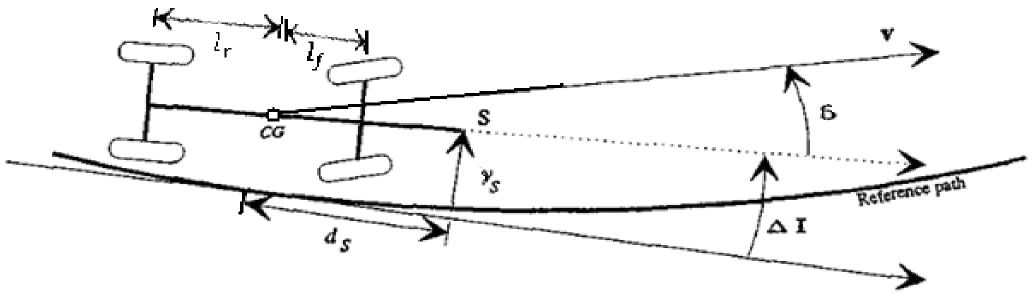

Fig 2. Lateral vehicle dynamics

These parameters influence the steering controller which projects a *steering-target point,* s, in front of the vehicle and computes the *steering position,* δ, that would minimize the *lateral displacement,* y, of the steering-target point from the desired path. The steering controller will be detailed in the next section.

IV. Steering Controller

The steering controller is designed to minimize error while following a reasonable path at a reasonable speed,[9]. Assuming that all paths and speeds fall well within the range of the vehicle's operating abilities; the steering controller is designed based on the look-ahead approach. Unlike look-down-controllers, which use the lateral deviation of one or more control points on the vehicle from the path, this method uses the lateral deviation from the path of a "virtual sensor" point at a specified distance in front of the vehicle as input and seeks to minimize this. The distance from the vehicle's center of gravity to this point is called the "look-ahead distance".

The steering controller projects a steering-target point in front of the vehicle and computes the steering position that would minimize the lateral displacement of the steering-target point from the desired path. The distance from the vehicle's center of gravity to the steering-target point is called the *steering-target or distance*. The transfer function from steering angle to lateral acceleration of the steering-target point is given by,[6,7]:

$$\frac{\ddot{y}(s)}{\delta(s)} = \frac{\mu C_f v^2\left(Ml_f d_s + I\right) s^2 + \mu^2 C_f C_r L v\left(d_s + l_r\right) s + \mu^2 C_f C_r L v^2}{\left(IMv^2\right) s^2 + \mu v\left(I\left(C_r + C_f\right) + M\left(C_f l_f^2 + C_r l_r^2\right)\right) s + \mu M v^2\left(C_r l_r - C_f l_f\right) + \mu C_f C_r L^2} \tag{6}$$

In which δ is the steering angle, y is the lateral displacement of the virtual point, μ is the road adhesion factor (from 1 to 0), ΔI is the instantaneous yaw rate of the vehicle, v is the linear velocity of the vehicle (speed), and d_s is the distance from the center of gravity to the virtual sensor (look-ahead distance)

By applying the Laplacian integrator $\frac{1}{s}$, twice, we get the transfer function from ground wheel angle to lateral displacement of the virtual sensor point. Using this transfer function and unity feedback (assuming the sensor makes little or no changes to the measurement) the control law can be derived.

Using the root locus approach, a lead-lag compensator can be designed. The lead lag compensator has the following standard form:

$$K(s) = K\left(\frac{s-a}{s-b}\right) \tag{7}$$

Where K is the gain of the system, |b|>|a| for a lead compensator and |a|>|b| for a lag compensator. The use of a lead compensator shifts the root locus of the system to the left, and thus increases the stability region and also increases the response time. The lag compensator is primarily used to improve the steady state performance of the system. That is, it helps reduce steady state error. Because the primary concern for steering is stability and good tracking performance, we will focus mainly on the lead compensator.

The root locus of the function is reported in figure 3.

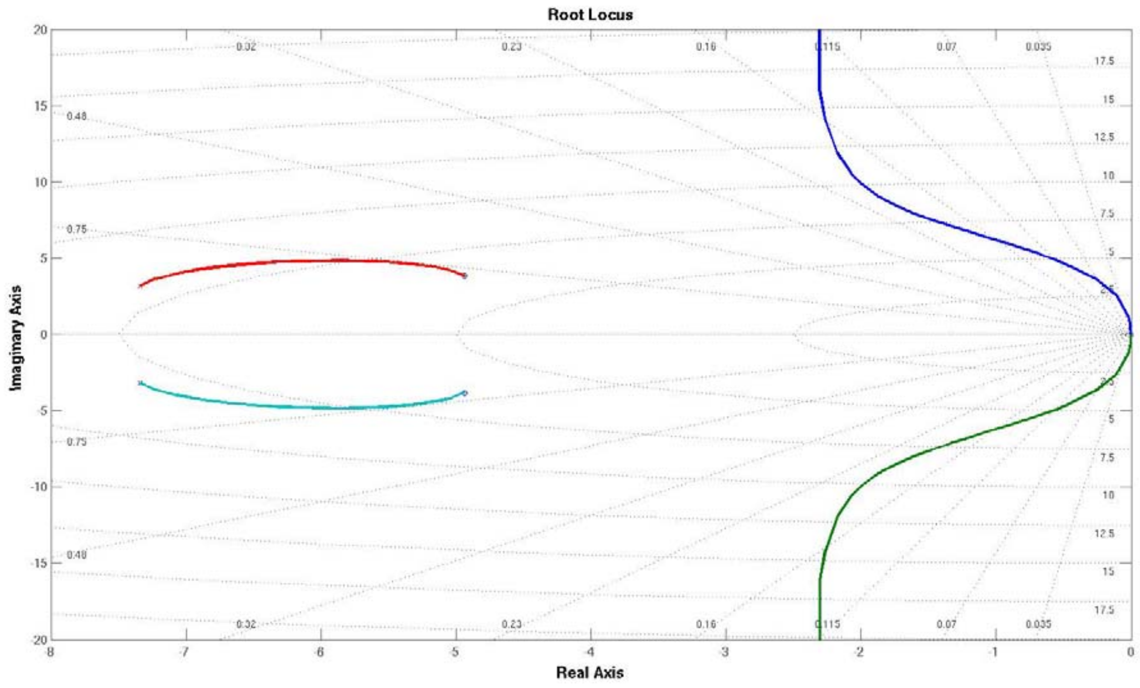

Fig 3. Root locus plot

The values of the design parameters a and b are computed in order to achieve a desired dampening ratio and natural frequency. This process is most commonly done by trial and error. Afterwards, the gain is adjusted until maximum performance is observed. For Raginbot, the following control law was used:

$$K(s) = 3\left(\frac{s+1}{s+20}\right) \tag{8}$$

The step response of the closed loop system is shown in figure 4.

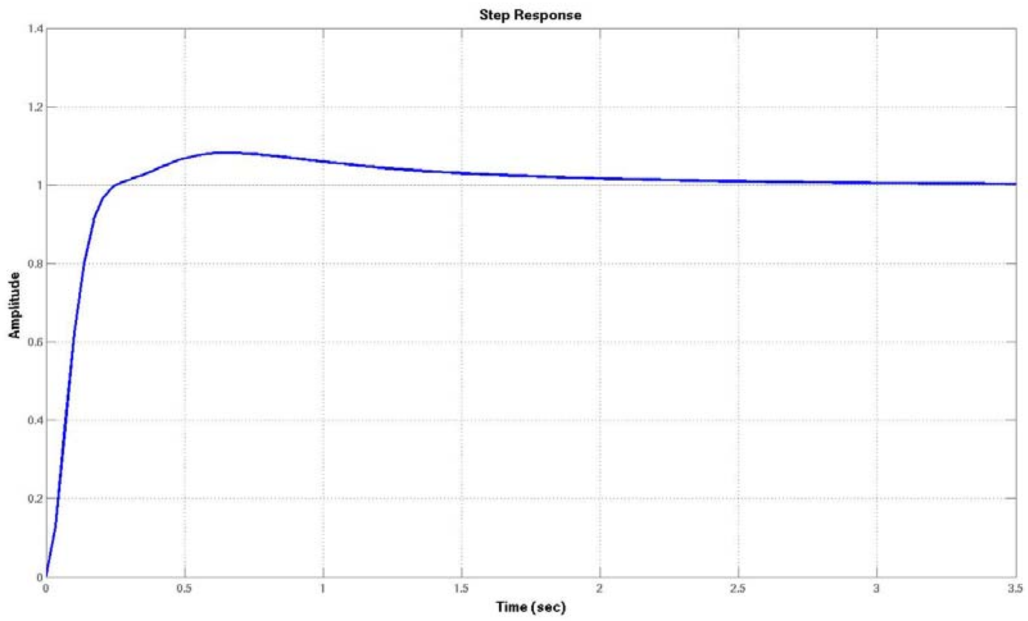

Fig 4. System step response

We can see that with the above considered design parameters, reasonable response is achieved.

The transfer function is described in the time domain as follows:

$$\delta(t) = PE(t) * e^{(-20t)} + D\frac{dE(t)}{dt} * e^{(-20t)} \tag{9}$$

where * means the convolution of the two terms and $E(t)$ is the error term. P and D are the proportional and derivative gains, respectively. The values for P and D are left as variables and used during implementation for tuning the controller.

V. Implementation

To validate the performance of the proposed look-ahead controller, the following is a series of experiments conducted for the CajunBot II, under several conditions. The known parameters of the vehicle are illustrated in Table 1.

Table 1 Nominal parameters of the CajunBot II

Description	Parameter	Value	Units
Moment of inertia	J	8470	Kg m^2
Vehicle mass	M	2000	Kg
Vehicle velocity	v	15	m/s
Vehicle width	b	1.694	m
Distance from center of gravity to front axle	l_f	1.5	m
Distance from center of gravity to rear axle	l_r	1	m

In the implementation, the gains are being automatically tuned to compensate for model mismatch, system delays and other characteristics caused by the digital implementation of the controller. Moreover, the steering-target distance is computed dynamically as a function of the vehicle's speed. What may be counter-intuitive, this distance decreases at higher speeds. A speed plan generates slower speed when the path has a greater lateral acceleration. The longer steering-target distance compensates for the steering delay of 0.24 s, thus preparing for the vehicle to turn ahead of time. On the other hand, at higher speeds a small change in the steering angle leads to larger change in the lateral displacement (for the same time interval). That is, the steering is more sensitive at higher speeds. A smaller steering-target distance at higher speeds ensures that the system generates gentle changes and provides greater stability. Presently, the look-ahead distance has a maximum value of 3.0 meters from the front bumper of the vehicle. When the vehicle reaches a speed of 3.0 m/s the look ahead distance begins to decline at a rate of .75 meters per 1.0 m/s increase in speed until it reaches a minimum of .25 meters from the front bumper.

Finally, the lateral displacement of the steering-target point y_s is computed as the distance from the steering-target point to the path, measured perpendicular to the vehicle's heading. When the vehicle is oriented near perpendicular to the path, such as at an intersection, the lateral displacement computed would be infinite, leading to an unpredictable behavior.

While the Executive module generates smooth paths through the intersection, for reliability and to achieve algorithm independence, it is important that the steering controller achieves its expected tracking performance if the Executive makes an error. One solution to this problem is to use a line perpendicular to the path segment to calculate the virtual sensor's lateral displacement. However, this too becomes troublesome as this can have multiple solutions and its computation is more complex. We use the following method for determining the lateral displacement. The steering-target point is determined by the normal method; however a second point, *vehicle track point,* is determined. This point is set by tracing the length of the steering-target distance along the vehicle's projected path. The distance between the vehicle-target point and the vehicle-track point is taken as the lateral displacement. For small heading errors, the difference in vehicle's heading and the path orientation, this computation yields values close to those from the previous method. However, when the heading error is large the new computation yields a stable solution. This method is also immune to variations in the path shape.

In order to account for small scale oscillations (or jitters) in the steering and to minimize the negative effects of GPS shifts, a moving averages filter is added to the controller,[10]. The filter is applied to both the input, virtual sensor error, and the output, steering angle. The filter is implemented by averaging any three points around a measured value and substituting the averaged value for the actual center value. Analytically, this can be viewed as forcing the values to change in a smooth manner while introducing a small delay of 0.05 seconds, or one interpolation at 20 Hz. This delay may seem counterproductive; however this delay is small when compared to the .24 second delay of the steering actuator, which is already accounted for in the look ahead distance.

Figure 5 illustrates the cross track error as the vehicles drives over a variety of paths. The cross track error is calculated as the distance from the INS to the path measured perpendicular to the vehicles heading. Typically, cross track error is calculated as the distance to the path from the center of gravity of the vehicle; however the navigation data on the CajunBot II is recorded as the location of the INS which is reasonably close to the center of the wheel base. For our purpose it serves as a reasonably close estimate.

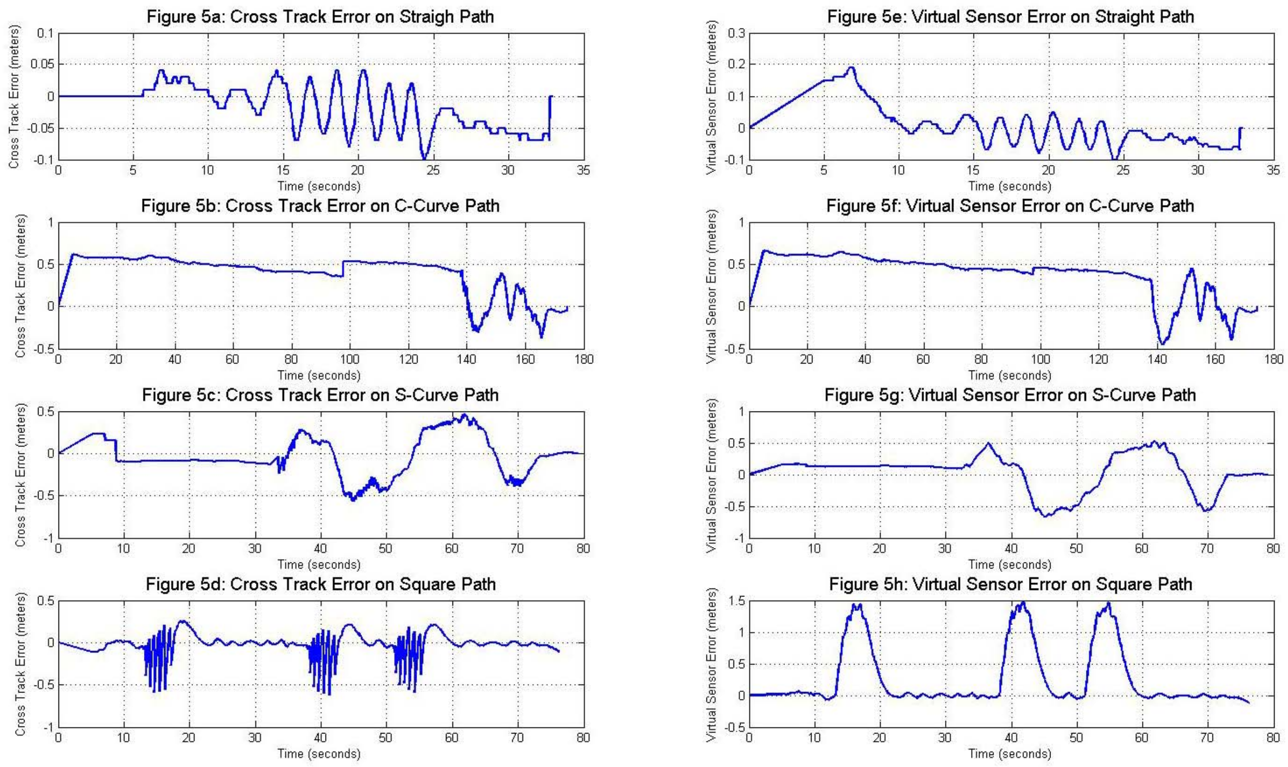

Fig 5. Cross-Track Error and Virtual Sensor Error for various paths

While testing the steering performance, the vehicle was set to travel at a maximum speed of 12 m/s. Each curved path, figure 5b and 5c, was made by driving the intended path manually and collecting the navigation data. The path is then extracted from this data in the form of 2 meter long segments that approximate the curve. The square path was made of 4 straight segments connected by arcs or a circle.

Figure 5a shows the cross track error as the vehicle drives down a long straight path. The maximum cross track error is approximately 10 cm. It is worth noting that the GPS of CajunBot is only accurate up to 10 cm. Figure 5b shows the cross track error of the vehicle as it travels a long path curved in one direction. The vehicle reaches a maximum speed of 9 m/s. The track used for figure 5c was a S-shaped curve. On this path, the vehicle reaches a maximum speed of 7 m/s. On the square path from figure 5d, the vehicle reaches a maximum speed of 12 m/s on the straight segments and takes each turn at a speed of 3 m/s.

Figure 5 shows that the vehicle reaches a maximum cross track error of .6 meters while maintaining stability. This error may be considered high by many standards, however, it is necessary in order to maintain stability and smoothness of steering on the wide variety of paths that CajunBot might encounter. For the tests shown in figure 4, the paths were made of either straight segments or made from splines collected from actually driving the vehicle. In the path planning process, many times, exact lane data in the form of splines is largely unavailable. In this case, the path is made of straight lines connecting waypoints. In this case, the angles between connecting segments can grow increasingly large, making it impossible for the vehicle to follow these paths with a reasonable cross track error while maintaining stability. In order to maintain stability in all situations and to further increase the smoothness of steering and control, the steering is tuned to not follow the path as closely as would be otherwise desirable..

VI. Conclusion

A look ahead steering controller was designed and implemented for an autonomous vehicle, with a primary aspect being to make the lateral error at a certain point ahead of the vehicle zero.
Through the use of a lead compensator, we have been able to create a steering controller that tracks a variety of paths with reasonable performance and stability, even when these paths are such that the vehicle cannot track closely given its non-holonomic constraints. Additionally, through the use of a moving-average filer, we were able to increase the smoothness of the steering.

For the current implementation of path generation, the steering controller exhibits optimal performance. In the future, if more complete lane data becomes available, the controller may be tuned to track the path more closely. However, this tuning may lead to instability if the path has any discontinuous features.

Acknowledgments

This work is partially supported by the University of Louisiana at Lafayette through the Louisiana Governor's Information Technology Initiative.

References

[1]DARPA Grand Challenge Race, [online]. Available: http://www.darpa.mil/grandchallenge

[2]Lakhotia, A., Golconda, S., Maida, A, Mejia, P., Puntambekar, A., Seetharaman, G., Wilson, S. "CajunBot: Architecture and Algorithms," *Journal of Field Robotics*, Vol.23, No.8, pp.555-578, August 2006.

[3]Hingwe, P., M. Tomizuka, "A Variable Look-Ahead Controller for Lateral Guidance of Four Wheeled Vehicles," *Proceeding of the American Control Conference*, Pennsylvania, 1998, pp. 31-35.

[4]Chen C., Tan H. S., "Steering control of High Speed Vehicle: Dynamic Look ahead and Yaw Rate Feedback, " *Proceeding of the 37th IEEE Conference on Decision and Control*, Tampa, Florida, 1998, pp. 1025-1030.

[5]Maida, A. S., Golconda, S., Mejia, P., and Lakhotia, A., "Subgoal-Based Local Navigation and Obstacle-Avoidance Using a Grid-Distance Field," *International Journal of Vehicle Autonomous Systems*, Vol. 4, No. (2-4), pp. 122-142, 2006.

[6]Ozguner, U., Unyelioglu, K. A., Hatipoglu, C., "An Analytical Study of Vehicle Steering Control," *Proceeding of the 4th IEEE Conference on Control Applications*, Albany, New York, September 1995, pp. 125-130.

[7]Rajamani, R., *Vehicle Dynamics and Control*, Springer-Verlag, New York NY, 2006

[8]Gillespies, T. D., "*Fundamentals of Vehicle Dynamics*," Society of Automotive Engineers, Inc., 1992.

[9]Guldner, J., Tan, H. S, Patwardhan, A., "Study of Design Directions for Lateral Vehicle Control," *Proceedings of the 36th Conference on Decision and Control*, San Diego, California, 1997, pp. 4732-4737.

[10]Trepagnier, G. P., Nagel J., Powell K. M., Koutsougeras, C., Dooner, M., "KAT-5: Robust Systems for Autonomous Vehicle Navigation in Challenging and Unknown Terrain," *Journal of Field Robotics*, Vol. 23, No. 8, pp. 509-526, August 2006.