Application and Analysis of a Robust Trajectory Tracking Controller for Under-Characterized Autonomous Vehicles

Melonee Wise and John Hsu

Abstract— **When developing path tracking controllers for autonomous vehicles the dynamic constraints of the vehicle are a critical factor. It is therefore necessary to ensure that all tracking trajectories produced by the controller are smooth and continuous. In this paper, a path tracking controller is proposed and implemented on an experimental autonomous vehicle. This tracking method decouples the low-level heading and steering control of the vehicle from the main tracking controller, therefore requiring less vehicle characterization. The results of this paper will show this method yields a RMS 0.25m cross-track error with little to no vehicle characterization.**

I. INTRODUCTION

With the recent activity in autonomous vehicle development at the DARPA Urban Challenge, many researchers ([10], [11], [12]) are focusing on developing robust path tracking controllers. These controllers either are part of the path planning or incorporate the low-level steering and speed controllers. Both approaches require extensive vehicle characterization, [2], and repeated calibration runs to ensure stability and accuracy. Additionally most of the control models for wheeled mobile robots and car-like vehicles are based around the bicycle model which does not account for tire slippage, suspension stiffness, engine throttle delay, etc.

In this paper the tracking controller is treated separately from the heading and speed controller of the vehicle. This pushes the vehicle characterization into the low level controller so that the vehicle dynamics can be modeled using the bicycle model. This allows for simplified implementation and testing of the tracking algorithm. For example, adopting the current tracking algorithm for driving the vehicle in reverse requires minimal changes to the algorithm itself; whereas careful characterization of the reverse steering characteristics are required for previously cited methods.

II. THE PATH TRACKING PROBLEM

A. Problem Description

Given a planar two dimensional trajectory composed of discrete GPS waypoints, the path follower is defined as a module which commands the vehicle to follow the specified path with some predefined tracking accuracy and passenger comfort. Given the uncertainties exhibited by the environment (uneven pavements, slippage, etc.) and nonlinear response behaviors of an under-characterized autonomous vehicle, a robust path follower must be able to track smoothly and consistently to the target trajectory. Through robust path tracking, the gap between high-level path planning and low-level hardware control of an autonomous vehicle is bridged.

1) Coordinate System: The body frame coordinates, shown in Fig.1, is a right handed coordinate system with y-axis pointing forward and z-axis pointing upwards. The rotational degrees of freedoms adheres to the right-hand rule with the exception of the yaw angle. Where yaw is left-handed with respect to the z-axis so it has the same rotational direction as the heading convention where $0°$ corresponds to north and $90°$ corresponds to east.

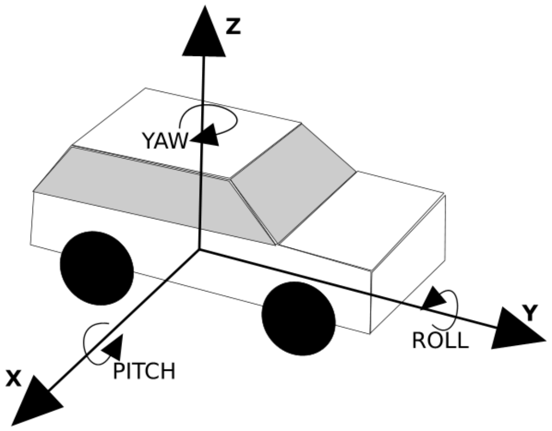


Fig. 1. Body Frame Axis System: x, y and z axis of the vehicle's body frame, with rotational degrees of freedom, pitch, roll and yaw.

III. CONTROL DESIGN

A. Governing Equations

In order to track smoothly to a given trajectory, a path must be computed from a given initial location and heading to some target point and heading on the desired trajectory, Fig. 2.

The in-line, cross-track, and heading errors, (e_x, e_y, e_θ) are given by

$$e_x = (x_t - x_c)cos(\theta_t) + (y_t - y_c)sin(\theta_t) \tag{1}$$

$$e_y = -(x_t - x_c)sin(\theta_t) + (y_t - y_c)cos(\theta_t) \tag{2}$$

$$e_\theta = \theta_t - \theta_c. \tag{3}$$

Once the path errors are determined a cubic polynomial, (4), can be used to satisfy dynamic constraints imposed by

This work was supported by Willow Garage

M. Wise is a Senior Engineer with Willow Garage, Menlo Park, CA 94025, USA mwise@willowgarage.com

J. Hsu is a Senior Engineer with Willow Garage, Menlo Park, CA 94025, USA johnhsu@willowgarage.com

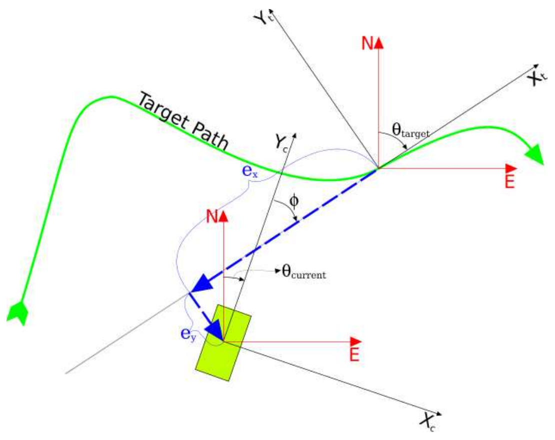


Fig. 2. Heading and error definitions.

vehicle dynamics and waypoint path planning (i.e. position, heading, and turning)[1]. The cubic polynomial is a function of the cross track error, e_y, and the constant c which dictates the steepness of the approach, as seen in Fig. 3.

$$P(x_p) = c(x_p)^3 sign(e_y) \tag{4}$$

From (4) the approach path heading angle θ_p, can be determined, where θ_t is the target heading on the desired path. Once the approach path heading angle is calculated, the approach path heading angle can be directly used to steer the vehicle via a heading controller.

$$\theta_p = \theta_t - arctan(3c\left(\frac{e_y}{c}\right)^{2/3} sign(e_y)) \tag{5}$$

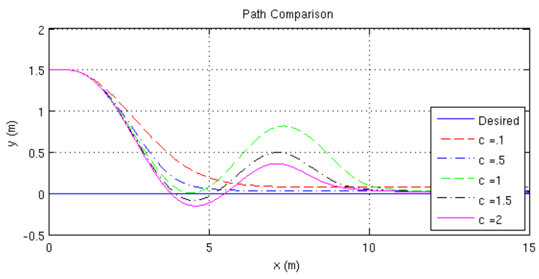


Fig. 3. The effect of the parameter c on the approach path.

Since the current approach needs to track to a specific GPS waypoint at every given time step, a controller for in-line position tracking is needed as well as for cross-track position tracking. To extend the controller for in-line error tracking, a modified bang-bang controller (7) is used. The estimated state velocity of the vehicle is given by

$$v_s = \dot{e}_x + (2a_{max}\,|e_x|)^{1/2} v_b. \tag{6}$$

where a_{max} is the maximum acceleration of the vehicle, and v_b is scalar value. By modifying the sign function of the bang-bang controller, the estimated state velocity changes smoothly satisfying the dynamic constraints of the vehicle.

$$\begin{aligned}\tfrac{dv_b}{dt} = -Av_b \quad &+ \quad (B - v_b)\max(0, e_x) \\ &- \quad (D + v_b)\max(0, -e_x)\end{aligned} \tag{7}$$

This results in an acceleration control law of the following form

$$a_{control} = \begin{cases} a_{max} & \text{if } v_s/dt > a_{max} \\ -a_{max} & \text{if } v_s/dt < -a_{max} \\ v_s/dt & \text{otherwise} \end{cases} . \tag{8}$$

The result of this control law can be used to control the speed of the vehicle and therefore close the loop around the in-line path error.

B. Implementation

The vehicle testing platform is an autonomized 2006 Ford Escape Hybrid. The basic software architecture of this vehicle is presented in Fig. 4; whereas the details of the controller hardware implementations are described in section IV-B.1.

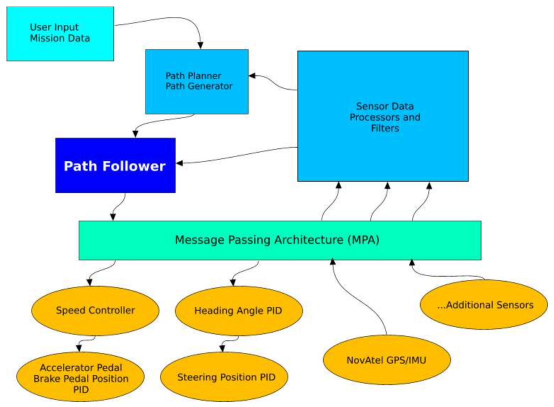


Fig. 4. Message Passing Architecture (MPA) construct.

The main software components of our autonomous vehicle system are composed of a path planner, a path follower, Message Passing Architecture (MPA), several independent sensor data processors and a few low-level control interfaces.

The backbone of the software architecture is the MPA which allows hardware and software driven processes to communicate between each other by passing messages robustly and efficiently through the use of shared memory. The basic structure of MPA is a ring-buffer queue, where all software processes can independently retrieve data chronologically or can insert new data to the end of the queue.

The input to the path follower comes from the path planner module, where a target path is specified by a list of waypoints in the global frame in the form of (9).

$$\bar{x}_i = \begin{bmatrix} x_i \\ y_i \\ v_i \\ t_i \\ \theta_i \end{bmatrix} ; i = 1, n \tag{9}$$

The velocity profile (v_i) is assigned to the list of waypoints based on hardware and environmental constraints such as available torque, path curvature, terrain roughness and posted speed limits. Given that the velocity profile has been determined, a time stamp (t_i) is assigned to each element of the waypoint list along the target trajectory.

The path planner module sends each waypoint of the target path to the path follower based on the corresponding time stamp. Since the target waypoints are a set of discrete path locations, simple two-dimensional linear interpolation based on time is used to generate each target waypoint sent to the path follower.

The in-line and cross-track tracking errors, (e_x, e_y), are defined by the distance from the vehicle's current position to the target waypoint at every time step. Given some finite tracking error, a robust path follower must be able to generate a smooth, stable and convergent landing curve to guide the vehicle back towards the intended path. Based on the tracking errors, the path follower computes the corresponding landing curve, then outputs velocity (v_c) and heading (θ_c) commands to the MPA structure.

The velocity and steering commands are picked up by the vehicle's low-level PID controllers. Separate PID loops for speed and heading control are implemented in the current vehicle platform. The steering PID controller determines the steering angle based on input heading angle. While two separate PID controllers for the accelerator and the brake works together to maintain the desired velocity. Additionally, all low-level hardware controller PID's are written in C/C++ languages with an average update rate of around $100Hz$.

IV. SIMULATIONS AND EXPERIMENTS

A. Simulation

1) Simulator Overview: The simulation environment is created using the open source Gazebo Project (http://playerstage.sourceforge.net/gazebo/gazebo.html). A snap shot of the simulator in action is shown in Fig. 5. A Gazebo vehicle model has been created with similar mass properties and acceleration/braking/steering characteristics as the actual vehicle.

2) Simulator Results: Given a test path shown in Fig. 6, the speed profile and the resulting ground tracks and cross-track errors of the simulated runs are plotted in Figs.7(a) $\sim$ 7(c). The speed profile in Fig. 7(c) has been generated by limiting the overall acceleration and multiplying the result by a weighting function proportional to the inverse of the path curvature; thus the lateral acceleration at corners are limited by predefined constants. From Fig.7(b) it is evident that the cross-track error has maximum magnitude of ~ 0.20m, while in-line tracking error spans ± 1.5m. The reason that the in-line errors are extremely large in comparison to cross-track errors is due to the fact that the vehicle was tuned in favor of passenger comfort rather than tracking accuracy. By increasing parameter a_{max} in (8), the tracking accuracy can be improved dramatically. Unfortunately, increasing in-line tracking accuracy results in noticeably more aggressive acceleration and braking behavior of the vehicle. In particular, the speed control module tends to alternate rapidly between accelerating and braking modes.

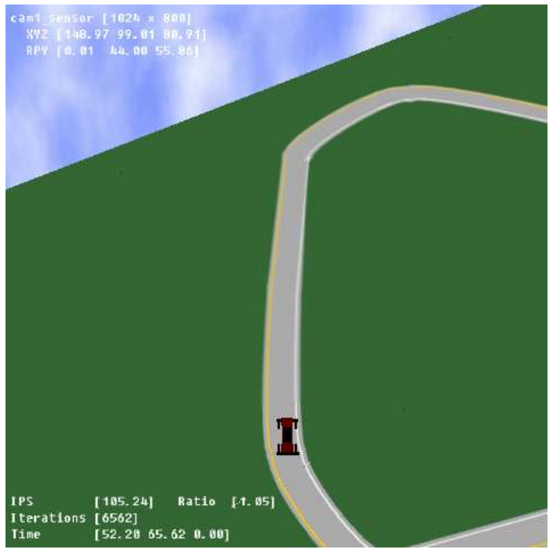


Fig. 5. A Snapshot of the Simulator

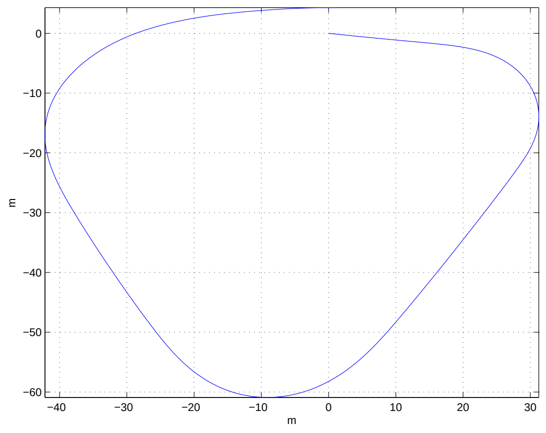
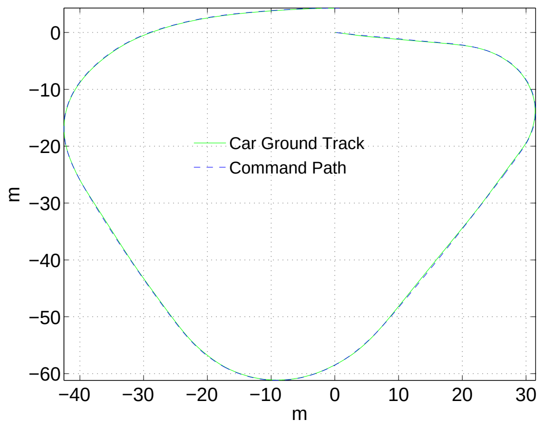


Fig. 6. Sample Test Path.

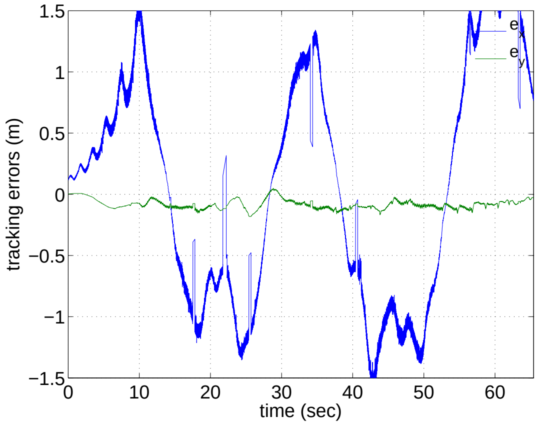
B. Experiments

1) Control Hardware Overview: The experimental platform is a Ford Escape Hybrid shown in Fig. 8, which has been reverse engineered for autonomous control. The existing core systems (steering, gear shift, accelerator, brakes, etc.) are actuated electronically which allows for easy interface with and control of these systems.

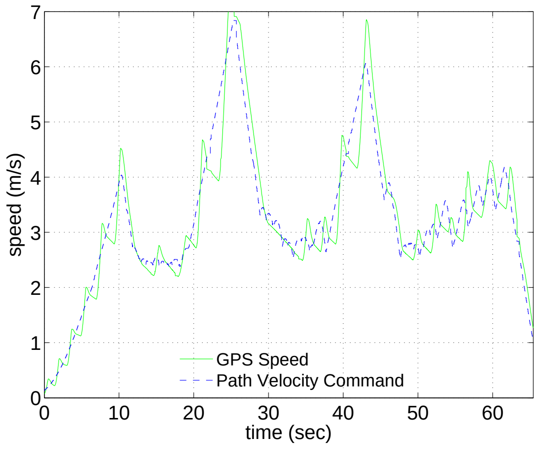
The vehicle is controlled by four custom 16Bit dsPIC Microcontroller boards, shown in Fig. 9, which interface with the existing Ford Escape computer hardware. The controllers are inserted in line, using standard Ford parts, to interface with the existing systems for easy installation, repair, or removal. Four modules are daisy chained together via a CAN bus and control the gear shift, accelerator, brakes, and steering.



(a) Simulation Ground Tracks



(b) Simulation Tracking Errors



(c) Simulation Speed Profile

Fig. 7. Simulation results tracking path in Fig. 6.

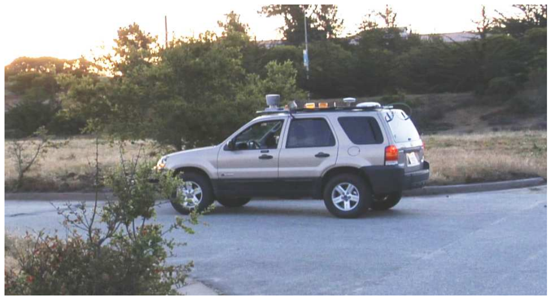


Fig. 8. Experimental Ford Escape Hybrid

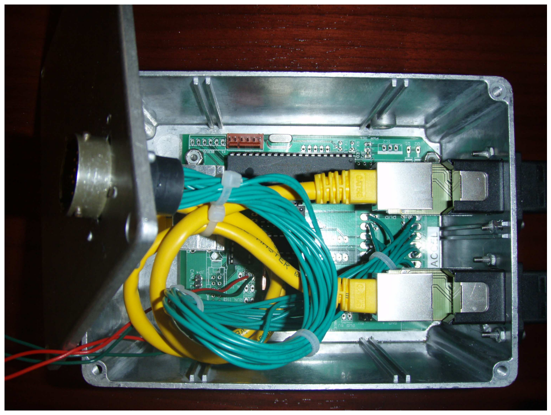


Fig. 9. Low Level Control Boards

i. Gear Shift Module: The gear shift module not only electronically selects the drive gear of the vehicle; the module also dictates whether the vehicle is in driver or autonomous mode. This is a design feature built into the system to quickly switch the vehicle from human driver mode to computer controlled mode. The gear shift module listens on the vehicle's CAN bus to determine the shifter position of the vehicle When the vehicle is in low gear the computer is able to send commands, over the computer CAN bus, controlling the gear position and other vehicle systems.
ii. Accelerator Module: The accelerator module controls the speed of the vehicle and the turn signals. A RPM sensor in the transmission determines the vehicle speed and broadcasts the speed on the vehicle CAN bus. The accelerator monitors the speed and uses a PID controller to maintain velocity set points dictated by the vehicle computer. The turn signals are turned on using a simple MOSFET switch that is activated when the module receives turn signal command from the computer.
iii. Brake Module: The brake module is responsible for sending brake control signals, and turning the brake lights on and off. The Ford Escape brakes are controlled using PWM pulses that increment and decrement an internal counter to increase and decrease the braking force. A PID controller in the brake module receives set point commands from the vehicle's computer and sends

pulses to the vehicle accordingly. The brake lights are also turned on using a simple MOSFET switch that is activated when the module receives a brake signal.

iv. Steering Module: The steering module uses the power assist motor in the Ford Escape to control the steering wheel position. A string potentiometer was added to the steering column to obtain accurate steering angles. The Ford Escape power assist motor relies on a torque sensor in the steering system to determine the amount of assist (torque) required to move the steering wheel. Similar to the brakes, a PID controller in the steering module receives set point commands from the vehicle's computer and sends torque values to the vehicle's power assist motor accordingly.

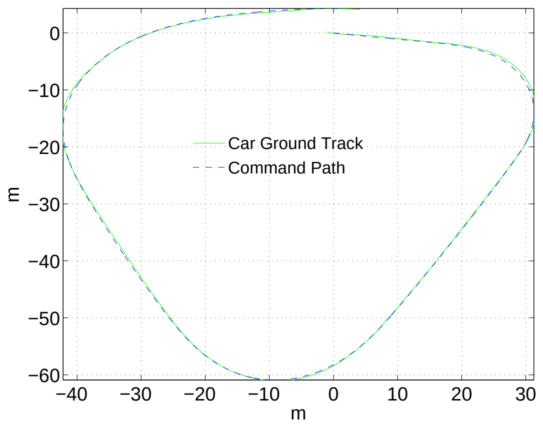
2) Sensor Hardware Overview: The vehicle is localized using an integrated senor network that utilizes the vehicle on board diagnostics and the NovAtel SPAN (Synchronized Position Attitude and Navigation) system.

i. On Board Diagnostics: The Ford Escape Hybrid comes equipped with Hall effect sensors on all four wheels and a transmission RPM sensor transmit data to the vehicle CAN bus. The on board diagnostic port on the vehicle can be used to read the CAN bus which transmits vehicle sensor data at a rate of 20Hz. This data is used for simple odometry and verifying the current position of the vehicle.

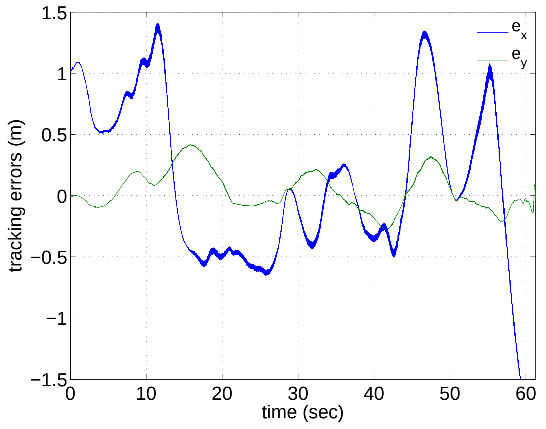
ii. NovAtel SPAN: The NovAtel SPAN system integrates a GPS (NovAtel GPS-702L) and IMU (HG1700 SPAN62). The GPS-702L receives L-Band frequencies from the OmniSTAR correction service and receives updates at a rate of 10Hz. The HG1700 SPAN62 IMU is a combined laser ring gyro and accelerometer with an update rate of 100Hz. Combining these two components the NovAtel SPAN system has a published accuracy of 0.1m and a 10 second outage accuracy of 0.39m.

3) Experimental Trials & Results: Fig.10 shows the results of tracking to the test path in Fig.6 using the current algorithm in our actual test vehicle. As a result of discrepancies between vehicle dynamics and simulator model dynamics, the maximum cross-track errors have increases from ~ 0.20m to ~ 0.40m for the actual test vehicle runs while the in-line tracking errors on the actual test vehicle have remained near the same levels as the simulation runs.

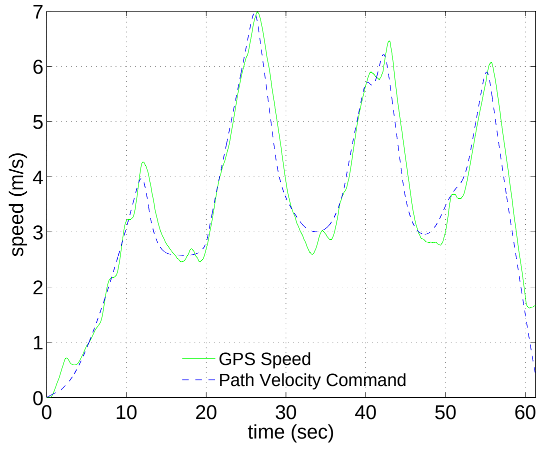
An additional test case was performed to examine the performance of the path tracking algorithm under higher lateral acceleration loads. A slalom path as shown in Fig.11 was given to the path follower to track, the results of tracking a more aggressive path are demonstrated in Fig.12. It is evident in Fig.12(b) that cross-track errors are increased while in-line tracking errors remain unchanged. The increase in cross-track error is due to the fact that the current path tracking algorithm is unable to handle lateral sliding motions caused by excessive lateral steering motions .



(a) Test Path Ground Tracks



(b) Test Path Tracking Errors



(c) Test Path Speed Profile

Fig. 10. Test Path Tracking Performance Results

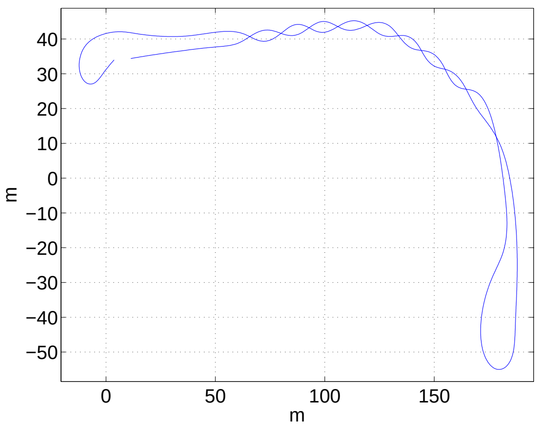


Fig. 11. Slalom Path.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The derivation, implementation and test results of a robust and stable path tracking algorithm are presented in this paper. The current path tracking scheme is well behaved and has sub-meter accuracy without the need for detailed characterization of vehicle dynamics. Even when pushed to the limits, as demonstrated in the slalom test case, the current path tracking algorithm is able to track the target path, with some sacrifice in accuracy, but without exhibiting any undesirable instabilities.

B. Future Works

As discussed in the conclusion, the in-line tracking controller had a significant negative impact on the accuracy of the trajectory controller. In light of this fact, future work will be done investigating and using other in-line tracking or velocity control methods to increase the performance of the existing path tracking controller.

VI. ACKNOWLEDGMENTS

We would like to thank our team leader, Jonathan Stark, without his long hours of hard work and dedication to this project, our research would not have been possible.

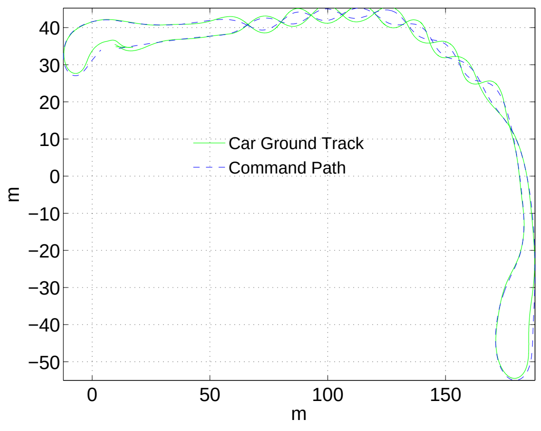
REFERENCES

[1] K. C. Koh and H. S. Cho, A Smooth Path Tracking Algorithm for Wheeled Mobile Robots with Dynamic Constraints, *Journal of Intelligent and Robotic Systems*, vol. 24, 1999, pp 367-385.

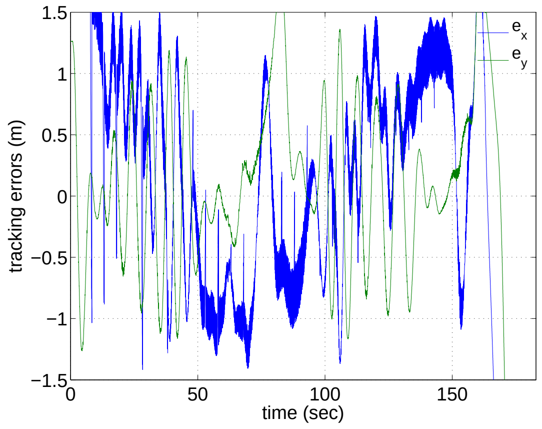
[2] Gabriel Hoffmann, Claire Tomlin, et. al., Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing, *in American Control Conference*, 2007, pp 2296-2301.

[3] Luca Consolini, Aurelio Piazzi, and Mario Tosques, Path Following of Car-Like Vehicles Using Dynamic Inversion, *International Journal of Control*, vol. 76, 2003, pp. 17241738.

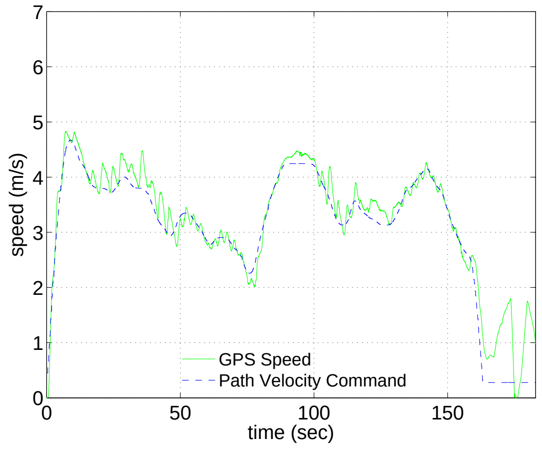
[4] ChangBoon Low and Danwei Wang, Robust Path Following of Car-Like WMR in the Presence of Skidding Effects *in International Conference on Mechatronics*, Taipei, Taiwan 2005 pp. 864-869.



(a) Slalom Run Ground Tracks



(b) Slalom Run Tracking Errors



(c) Slalom Run Speed Profile

Fig. 12. Slalom Tracking Performance Results

[5] Margan Davidson and Vikas Bahl, The Scalar ϵ-Controller: A Spatial Path tracking Approach for ODV, Ackerman, and Differentially Steered Autonomous Mobile Robots, *in International Conference on Robotics and Automation*, Seoul, Korea, 2001, pp. 175-180.

[6] F. Diaz del Rio, G. Jimenez , et al., A Generalization of Path Following for Mobile Robots, *in International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 7-12.

[7] Cripps Donald, Spatially-Robust Vehicle Path Tracking Using Normal Error Feedback, *in Proceedings of SPIE*, vol. 4364, 2001, pp. 222-238.

[8] A. M. Bloch and N. H. McClamroch, Control and Stabilization of Nonholonomic Dynamic Systems, *in IEEE Trans. Automatic Control*, vol. 37, 1992, pp. 17461757.

[9] Yutaka J. Kanayama and Fariba Fahroo, A New Continuous-Curvature Line/Path-Tracking Method for Car-Like Vehicles, *Advanced Robotics*, vol. 13, 2000, pp. 663-689.

[10] William Travis, Robert Daily, et al., SciAutonics-Auburn Engineering's Low-Cost High-Speed ATV for the 2005 DARPA Grand Challenge, *Journal of Field Robotics*, vol. 23, 2006, pp. 579-597.

[11] Isaac Miller, Sergei Lupashin, et al., Cornell University's 2005 DARPA Grand Challenge Entry, *Journal of Field Robotics*, vol. 23, 2006, pp. 625-652.

[12] Chris Urmson, Charlie Ragusa, et al., A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain, *Journal of Field Robotics*, vol. 23, 2006, pp. 467-508.