Advanced State Estimation and Control
of an
Autonomous Ground Vehicle
using
a priori Knowledge of Vehicular Dynamics

Abstract:

This paper explores this use of collected data on vehicle dynamics to increase performance in dead reckoning, state estimation and control of an autonomous ground vehicle. Using the existing speed- and steering-control modules on an autonomous vehicle, 18 data runs were taken at a variety of steady-state speeds and steering angles. GPS and wheel odometry were used to collect position, speed and heading information. The collected data is first used to verify the validity of several common vehicle models and is then applied to a more complex model. It is found that the commonly used Bicycle Model is an inadequate vehicle model. Additional applications for this *a priori* vehicle knowledge are discussed, including advanced vehicle controllers, and enhanced dead reckoning using an Extended Kalman Filter.

Gordon Franken, Zachary Glass

MAE 340D
Independent Work w/Design
Professor Alain Kornhauser
Professor Robert Stengel
Spring 2007

Table of Contents

Acknowledgements

This paper focuses on a very specific subset of a large scale project that requires the efforts of many people across multiple disciplines. The authors would like to thank the following people and organizations for their support on our project:

- Our independent work advisors, Professors Kornhauser and Stengel, for their advice and guidance.
- Josh Herbach, for writing and debugging the logging service that allowed us to collect our data.
- Brendan Collins for writing the State Estimation in 2005 and filling us in on the details.
- Lindsay Gorman, for writing the latest incarnation of State Estimation and helping us apply our new models.
- Andrew Saxe, for writing, debugging and tuning many of the control systems on our current and past vehicles.
- Jon Mayer, for writing the GPS service and re-writing it after the serial port died.
- Our data collection team - Derrick Yu, David Benjamin and Andrew Saxe - who sat with us and tried keep our land legs as we did autonomous circles in the parking lot.
- Also, thanks to Derrick for writing most of our testing interface software.
- The PAVE team as a whole, for standing by us for support, encouragement and enthusiasm.
- All PAVE sponsors and especially Ford Motor Company for donating the vehicle and FRABA for giving us the high-precision encoder at a substantial discount.
- A special thanks goes out to Dean Vincent Poor and the School of Engineering and Applied Science for their continue support of PAVE and undergraduate research.
- Finally, we would like to give additional thanks Professor Kornhauser as the PAVE faculty advisor. We speak on behalf of all the students involved with PAVE when we thank him for his unwavering support of out project and his dedication to enriching our experience. We are always grateful.

1. Introduction

This paper presents a novel low-cost approach to improving the state estimation and control of an autonomous ground vehicle (AGV). Drawing on biological cues from how humans drive, we aim to provide the robotic system with an intuitive knowledge of the vehicle.

The predominant approach in controlling a robotic vehicle is responsive, rather than based on an intuitive knowledge of the vehicle. Robots fine-tune their actions as they go, relying primarily on sensory feedback to ensure that they are tracking correctly.

This approach works extremely well when using high-accuracy inertial measurement units (IMU) and GPS. However, these units can be extremely, sometimes prohibitively, expensive. The proposed intuitive-approach, however, requires few sensors in addition to those already on every new vehicle. The lack of sensor accuracy is compensated for by the quality of the vehicle model used, since this model can be fine-tuned for the specific vehicle.

Using Princeton's AGV entry into the 2007 DARPA Urban Challenge, we will collect a substantial amount of vehicle dynamics data. This data will be merged into several vehicle models. This *a priori* knowledge of how the vehicle responds under various conditions allows for more accuracy in state estimation. Even with rudimentary sensors such as wheel encoders, we believe an intuitive system can achieve dead-reckoning results on par with a basic GPS/IMU system. In addition, knowledge of how the vehicle will move given a certain input allows for faster control system response, since tracking is less dependent on feedback and relies mostly on an open-loop feed-forward component.

First, a broad introduction to the topic is given. Section 1 concludes with a more focused look at the specific problems addressed in this paper. Section 2 provides an overview of vehicular state estimation, including sensors for measurement and an in-depth analysis of three distinct vehicle models. Section 3 outlines the data collection procedures, summarizes the vehicular dynamics data and applies it to the vehicle models. Section 4 takes a step aside and examines the low-level actuation and control of the drive-by-wire systems of an autonomous vehicle. In Section 5, various techniques for

lateral and longitudinal control are discussed, including the applicability of the previously collected vehicle data. Section 6 concludes the paper with recommendations for future work.

1.1 Background

The last two decades have shown an ever-increasing trend in consumer automobile intelligence. Anti-lock braking systems have been replaced by Vehicle Stability Control systems, which can independently apply the brakes on each wheel to correct for oversteer or understeer and prevent loss of control. Simple vacuum-based cruise-control systems have been replaced by electronic throttle servos. These are now augmented by forward-looking RADAR units to create adaptive cruise control, which allows a vehicle to maintain a safe following distance behind a slower-moving vehicle. Camera-based lane-departure warning systems that are just emerging on vehicles will soon be replaced by automated steering systems that can steer a car down the highway. Once off the highway, cars can parallel-park (albeit in a spot twice as long as the car itself- good luck finding one of those in any major city!). In addition, GPS-based navigation systems are becoming increasingly popular. Drivers now know where they are, how to get to their destination and the location of the nearest Chinese Buffet. Real-time systems allow for automatic re-routing around traffic congestion.

And yet, for all their intelligence, even for all their advanced safety features such as seatbelts and airbags, cars remain extremely dangerous. Over 40,000 Americans die each year in automobile accidents – the leading cause of death for Americans ages 3-33. The majority of these fatalities are the result of human error. It seems as though the next big safety feature will be taking the driver out of the picture.

The military has an even greater demand for autonomous vehicles, primarily for the same reason of safety. Many casualties in current and recent U.S. military activities are the result of convoy ambushes or roadside bombs. Autonomous military vehicles that could perform supply and reconnaissance duties would not only save soldiers lives, they would also cost significantly less. In 2005, the Pentagon released a study that determined the average “cost of a soldier form enlistment to internment is about $4 million” [6]. Autonomous military vehicles, once mass-produced, will cost substantially less to build

and maintain. A congressional mandate in 2001 stated that "by 2015, one-third of operational ground combat vehicles [must be] unmanned."[5]

As a result of these growing trends and in response to the congressional mandate, the Defense Advanced Research Project Agency (DARPA) was given authority to issue large cash awards to those who successfully demonstrated capable autonomous ground vehicle (AGV) technology. Thus began the DARPA Grand Challenge Program.

The real challenge of the Challenge is not to invent new autonomous systems. The majority, if not all of the required systems already exist. Rather, the challenge is to integrate all of these systems properly and come up with innovative solutions to previously hard problems. DARPA has a long-standing history of ground-breaking technological innovation from the Internet to unmanned aerial vehicles. If history has any say, autonomous vehicles may be the next breakthrough.

1.1.1 DARPA Grand Challenge Program

In October of 2003, DARPA announced the creation of the Grand Challenge program; a technological competition intended to spur innovation among private and commercial engineering groups in the field of autonomous ground vehicles. The first Grand Challenge, held in May of 2004, required vehicles to navigate over 140 miles on a difficult desert course. This event turned out to be a relative failure. Although many teams were able to make it to the final race, only a few successfully made it past the starting area and the leading team, from Carnegie Mellon University, traveled just over seven miles.

Almost immediately, DARPA announced a second Grand Challenge race, to be held in October of 2005. Over 195 teams entered the 2005 DGC, Princeton among them. The Princeton University team was also accepted as one of 43 semi-finalists and one of 23 teams to compete in the final race. The 2005 DGC course was somewhat easier than in 2004; nearly a dozen miles shorter over less varied terrain. Nevertheless the complexity of full vehicle autonomy in a desert environment still posed quite a challenge. Five teams successfully completed the course; four under the 10-hour time limit, led by "Stanley" from Stanford University. Princeton University's "Prospect Eleven"

completed 9.5 miles of the course before succumbing to a memory leak, and finished 19th.

The successful completion of the 2005 Grand Challenge demonstrated that vehicular autonomy in a desert environment is possible, but this does not imply the ability to navigate in any other complex environment. One such complex environment that has immediate applicability to both the military and civilian population is the city.

Therefore, as the logical outgrowth of the previous competitions, DARPA announced the 2007 Urban Challenge in May of 2006. This competition, like its predecessors, carries substantial cash prizes for the winners. Scheduled to take place on November 3, 2007, the Urban Challenge requires vehicles to navigate within a complex urban environment for up to 60 miles and 6 hours. Vehicles will have to exhibit proper and safe driving behaviors in a of traffic conditions such as intersections, lane changing, merging traffic, parking lots and stop signs.

Princeton University is entered into the Urban Challenge. At the time of this paper's publication, the team is waiting to hear from DARPA if they have been selected for a site-visit evaluation – one of the numerous qualification steps leading up to the final race.

1.1.2 PAVE

Princeton University's team is known as PAVE, for Princeton Autonomous Vehicle Engineering. PAVE was founded in early 2006, soon after the conclusion of the 2005 DARPA Grand Challenge, as an undergraduate research group dedicated to autonomous vehicles. Initially PAVE focused on the development and commercialization of this technology for both on- and off-road applications. The announcement on the 2007 Urban Challenge reordered its priorities.

Thanks to a successful recruitment season in the fall of 2006, and the construction of a new garage space in the basement of the Carl A. Fields center in February 2007, PAVE is well-equipped to tackle the complex set of requirements defined by the Urban Challenge. PAVE currently consists of over twenty undergraduate students – many of them underclassmen, and has attracted substantial media attention for its sponsors and for the University. It remains the only undergraduate-led research group on campus.

1.1.3 Vehicles

Through generous in-kind sponsorship from major American automobile manufacturers, PAVE has received two vehicles for competition use; a GMC Canyon Crew Cab pickup truck in 2005, and a Ford Escape Hybrid SUV in 2007. The Canyon was substantially modified for computer control and named "Prospect Eleven". The Escape has also been modified for drive-by-wire operation, although it remains unnamed at the time of this paper's publication.

1.1.3.1 Prospect Eleven

The scope of the 2005 Grand Challenge required a minimum of medium-range obstacle detection. The only environmental sensor on Prospect Eleven was a Point Grey Bumblebee stereo camera with a 12cm baseline. Global localization came from a Trimble DGPS unit, which provides 1m accuracy position data at 1Hz. Local positioning, used to interpolate between GPS updates, was accomplished by dead reckoning based on ABS wheel ticks from the front wheels. A Kalman filter was used to maintain the state vector.

Prospect Eleven's navigation algorithm was a home-grown scheme that evolved from a genetically-tuned controller into a reactive approach that converted the surrounded world to polar coordinates and operated on a nearness diagram. The end result was an effective controller that exhibited a variety of desired behaviors, such as GPS waypoint-following, obstacle avoidance and rapid path convergence. The navigation algorithm outputted desired speed and desired steering angle to the respective low-level control systems.

The fundamental drive-by-wire systems required for autonomous vehicle operation include steering, brakes, throttle and transmission. In Prospect Eleven, all of these systems were interfaced with or modified. In addition, the horn, headlights, turn signals and an emergency brake were also put under computer control. Steering was accomplished by attaching a set of gears and a DC motor directly to the steering column. An optical rotary encoder provided feedback for the angle of the steering wheel. Braking was accomplished through a cable connection to the brake pedal from a linear actuator. A potentiometer and a tension sensor were both used for feedback control of the braking

system. The accelerator pedal on Prospect Eleven was electronic, so it required no mechanical interaction – only an electrical interface which simulated the sensor response of the actual pedal. All modifications were done such that the vehicle remained human-drivable. PID control loops were used to maintain desired steering angle and desired speed, as set by the navigation algorithm. An additional PID controller was used to govern the braking system's linear actuator. These controllers were tuned through a combination of system ID analysis, and manual adjustments.

Prospect Eleven used two computers for on-board data processing. Named Santiago and Prospero, both computers had AMD-64 processors running at 2GHz – about the same as an average desktop. Prospero was used exclusively for stereo vision depth map calculation and obstacle detection. Obstacles were transmitted over Ethernet to Santiago, which handled the rest of the Navigation and Control computing. Almost all of the code was written in C#; vision code was in C++.

The rear bench of the crew-cab was removed and a large wooden frame was installed. This frame housed all of the computers and electronics. Two additional batteries and an inverter/charger, located beneath the frame, handled the electrical needs of the additional systems. A wooden boom on the roof provides support for a variety of sensors and antennae, including a digital compass, GPS and a rotating beacon light.

Prospect Eleven remains both human-drivable, and fully capable of autonomous operation. It is used for testing new code and sensors, although the primary development for the Urban Challenge takes place on the current competition vehicle, the Ford Escape.

1.1.3.2 The Escape

Princeton University's competition vehicle for the 2007 Urban Challenge is a yet-to-be-named red Ford Escape Hybrid SUV (herein referred to as "The Escape"). It was donated in January 2007 by Ford Motor Company. The team immediately set to work: modifying the vehicle for computer control, and outfitting it with an array of sensors and algorithms to perceive and respond to its complex environment.

The Urban Challenge has a much greater scope than the previous Challenges in the desert and requires more variety, coverage and range from the sensing systems. PAVE has maintained its low-cost approach and selected a vision-based sensor

configuration, supplemented by long- and short- range RADAR. Three stereo cameras will provide medium and long-range obstacle detection over 180 degrees in front of the vehicle. A single forward-facing color camera is used for lane-detection. Two additional black and white cameras provide blind-spot coverage. A long-range radar on the front is used to aid in car-detection, while a short-range radar is located in the rear for collision avoidance while reversing. A Trimble Ag114 GPS provides global position information at 5Hz with sub-meter accuracy. Wheel odometry information will be available from all four wheels, in addition to the Vehicle Speed Signal (VSS), to allow for dead reckoning. A high-precision rotary encoder is mounted on the steering column for angular position feedback. An Extended Kalman Filter (EKF) updates the state vector with data from all of these sensors.

The Urban Challenge also requires a more complex navigation scheme than before. The navigation task is split into two; Global and Local. Global navigation is responsible for high-level path planning within the specified route network. An implementation of Dijkstra's algorithm using a Fibonacci Heap allows for rapid planning and re-planning of the desired course within the network. Local navigation, along with a Sensor Fusion routine, takes the raw data from environmental sensors and combines it into a unified view of the world. From this world view, a variety of behaviors, such as lane-following, car-following, lane changing and intersection precedence are evaluated and performed. The navigation routine outputs a desired path through the world for the car to follow.

[The actuation and control of The Escape are discussed in depth in sections 4 and 5, respectively. Included here is a brief overview of their implementation.]

The Escape is a Hybrid-Electric Vehicle, so many of its existing systems are electronic. No mechanical actuators were built in order to control the steering brakes or throttle; all of these interactions are accomplished through a complex set of electronic circuits and a pair of data acquisition units. A simple lever-arm powered by a small DC gearmotor is used to shift the transmission. Additional controls, such as turn signals, are accomplished through mechanical relays. A pneumatically-power emergency brake is passively attached to the brake pedal, and is controlled by an independent emergency stop system. Much vehicle data is available over the on-board CAN-bus network, although

critical sensors such as VSS and brake-pedal switches are hooked directly to the electronics. PID controllers are used to regulate the angle of the steering wheel, as well as the forward speed and/or position of the vehicle.

The number of sensors and complexity of algorithms requires a substantial amount of computer processing power. Currently, The Escape has 4 identical computers - each with an Intel Core2Duo processor at 2.6GHz, although provisions for up to 4 additional computers have been made. The majority of code is written in C#, with the exception of vision processing code, which is in C++ for performance increase. To facilitate communication across multiple code modules on multiple computers, the recently release Microsoft Robotics Studio (MSRS) development framework is utilized. MSRS provides concurrent communication by packaging code modules into 'services' that interact via 'messages.' Multithreading and event-based issues are handled inherently by the MSRS framework, as is data logging. The computers are mounted in a shock-isolated rack in the trunk of the vehicle. All computers run directly off the DC voltage from the car battery, although and inverter/charger provides auxiliary AC power and maintains the battery charge. A wooden boom is attached to the exiting roof-rack to support several cameras, the GPS and the rotating light bar. A box mounted in the engine compartment houses all of the custom electronics used for drive-by-wire control of the vehicle.

The Escape has been successfully modified for drive-by-wire operation, and has completed a substantial amount of autonomous driving relying on a combination of stereo vision for obstacle detection and color imaging for lane following. Further integration and development is required to utilize all of the sensors and test the advanced navigation routines.

1.2 Closing the Loop

Any simple computing system follows the general paradigm of input-process-output. A robotic system usually has many complex modules, each of which also follows this paradigm. The top-level Perception-Cognition-Actuation[1] cycle, a low-level

[1] Perception, Cognition and Actuation (P-C-A) are the names given to the robotics feedback cycle more commonly known as sense-plan-act, by PAVE. In addition, PAVE has recognized that these three processes are not sufficient. A fourth step, known as Environment, is included in the cycle to complete the

controller and nearly every process in between all take in an input, process on that data, and produce an output. However, one key element separates a few processes from the bunch: feedback. Almost all low-level controllers require it; they get feedback data from a variety of dedicated sensors. On a complex robot such as an autonomous ground vehicle, the high-level P-C-A cycle is pointless without additional step of Environment to complete the loop. Not only does almost any action taken by such a robot necessarily change the environment, but the environment itself changes independently. Therefore, reliable sensing for adequate feedback is essential.

This paper makes two large assumptions about the P-C-A-E cycle, in order to simplify our analysis. First; that the Perception step is able to perfectly detect the external environment and second; that the Cognition step processes this data and outputs the optimal desired path through the world. The fundamental problem this paper aims to help solve is that of reliable and robust path tracking. Path tracking is an input-process-output cycle that relies on feedback. The feedback is accomplished in the perception step. In addition to perceiving the surrounding world, an AGV must also perceive its own motion within the world. The output is usually handled by low-level controllers, which may control the steering, speed and other actuators on the vehicle. This paper suggests two improvements to the path tracking problem; one at the perception stage, the other at the actuation stage. Both rely on advanced knowledge of the specific vehicle dynamics. First, we examine perception; estimating the vehicle motion.

2. Estimation

The state of an object is described by its relationship with its surroundings at any given time. The difficulty is that in a real world scenario the relationship between an object and the rest of the world is not always fully measurable. Invariably, we have to settle for a subset of that relationship which still accurately describes the object.

---

loop. Furthermore, the first three processes require an extra entity, known as Substrate, to function. The Substrate entails all of the hardware, software and computer architecture that allow for implementation. The above description of the vehicles is broken down along the lines of these entities (Perception, Cognition, Actuation, and Substrate).

2.1 Degrees of Freedom

With the particular problem of tracking and controlling a vehicle, we are interested primarily in motion. This narrows our state from an immeasurable number of variables to just six. These variables correspond to the six degrees of freedom (6DOF) that any object has while traveling in three dimensions. The three degrees of linear motion are commonly referred to as North, East and Up, while the three degrees of angular motion can be called roll, pitch and yaw.

The 6DOF state can be further simplified with the assumption that our vehicle will be traveling on a flat plane. With regard to z-axis motion, this is a fair assumption since cars do not travel up and down that much (if our car stops touching the ground, state estimation is the least of our worries). Two dimensional motion further implies that no pitch or roll occurs. In practice, roll and pitch do occur due to the suspension, but their effect on the cars two-dimensional position is insignificant. In two-dimensional motion, the basic vehicle state can be described by 3DOF; linear motion in the x-y plane and yawing motion about the z axis.

2.2 Measurement

Estimation cannot happen without continuous measurement of conditions, as well as some knowledge of the sensor properties. Based on our 3DOF state, we know that we need to estimate the heading and position of the vehicle, as well as an arbitrary number of derivatives, such as speed and acceleration.

A variety of sensors for measurement were experimented with. In keeping with the idea of using intuitive knowledge of the vehicle, most of the sensors measure inherent data. The major exception to this was GPS. The Global Positioning System measures with respect to external satellites and has nothing to do with the car's inherent properties. This was necessary because we are also concerned with the car motion within the world, not just with respect to itself.

2.2.1 Position

We have two methods of directly measuring position. The more accurate measurement is GPS, which has sub-meter precision and 5Hz update rate. Counting wheel ticks is the other method.

Each of the four wheels on the car has a speed sensor that consists of a magnetic sensor and a notched ring. Each time one of the notches passes the sensor, a pulse is generated. Each pulse corresponds to a specific distance the wheel has moved.
On the Escape, all four wheel speed sensors are fed into the Braking module, which then outputs the vehicle speed signal. The VSS is nominally calibrated to have 31924 pulses per mile.

By counting the number of pulses on the VSS line, we can compute the forward distance traveled. The car uses the same procedure to update the odometer. However, the wheel speed sensors are prone to systematic error buildup. For example, an incline, a change in tire pressure, even a change in weight distribution can lead to significant inaccuracies in the long term.

2.2.2 Velocity

Velocity is technically a subset of position measurement, although different methods can be used to measure it. In addition to giving the global position of the car, GPS also measures velocity. Whereas GPS position is calculated by the distances to satellites, GPS velocity is calculated from the Doppler shifts of the signals coming from the satellites. This makes GPS speed measurement very precise. One caveat is that it measures speed in the direction of motion of the GPS, which is not along the longitudinal axis of the vehicle during a turn.

The second measurement of speed also comes from the VSS. The quick-and-dirty method is simply to differentiate the numbers of ticks over time. Unfortunately, the discrete values of the ticks make this estimate extremely jumpy, so it must be smoothed using an infinite impulse response weighted average. The result is still noisy, but converges to an average speed that appears consistent.

There is a direct way to measure speed from the VSS that involves frequency measurement. Since each pulse corresponds to a distance, the frequency of pulses must

correspond to a speed. By measuring the frequency or duty cycle of the pulse train, an instantaneous speed measurement is obtained that is stable and free of noise. However, below one meter per second, the duty-cycle measurement tends to blow up due to counter overflow in the data acquisition system. Therefore, differential ticks are only used at low speeds. On a level and straight road, experimental VSS data has proven to be within one percent of GPS speed. While it is not yet as accurate as GPS, VSS has the advantage of updating about ten times faster.

2.2.3 Acceleration

Acceleration is yet another subset of the position measurement. It is a common one to measure because of the popularity of inertial measurement units (IMU) that measure linear and rotational acceleration with accelerometers and gyros. These values can be integrated once to obtain velocity and again to obtain position. However, even a small amount of noise or variation in the inertial data can lead to large errors when integrated.

We did not directly measure the accelerations of our vehicle car. Accelerometers and inertial measurement units of substantial accuracy were simply too expensive. However, by dealing with steady state data and models, we avoid most accelerations altogether. This is still accurate for driving situations which do not involve quick maneuvers, because systems with smaller perturbations spend less time in a transient response. In most urban areas (the case we are most interested in), the turns are fairly gentle. Quick lane changes would therefore be our most likely source of error.

2.2.4 Heading

We have four methods of measuring heading, only two of which ended up being useful:

Prospect Eleven was outfitted with a Honeywell Digital Compass Module. We tested its performance before attaching it to the Escape for data collection. We found that while it did tend to give decent estimates for heading (+/- 1 deg), it was very sensitive to magnetic fields and surrounding metal objects, such as a car. In addition, heavy accelerations caused large fluctuations in the reading.

Another method of measure heading was suggested by PAVE faculty advisor, Professor Kornhauser. The concept is to mount two GPS receivers spaced a certain distance apart longitudinally along the center of the vehicle. By comparing the difference in position from the two receivers, a heading vector could be generated. Although sound in theory, this method turned out to be unreliable in practice. The variation in GPS measurements meant that an average heading estimate would take several minutes to converge to within a few degrees. In real time the system is accurate to within about +/- 20, enough for a rough estimate of orientation, but not accurate enough for our measurements.

GPS uses Doppler shift to measure heading, as with the speed measurement. Rather, it measures the course angle, which is the angle between the velocity of the car and the earth fixed y-axis. Heading is the angle between the longitudinal body axis of the car and the earth fixed y-axis. In reality, these angles are usually very close to each other and are the same when going in a straight line. In most cases we assume GPS is measuring heading.

An encoder on the steering wheel also allows us to calculate heading. While it does not directly measure vehicle heading, it can be easily related to yawing velocity, the derivative of heading. This relationship is model dependent.

2.3 State Filtering

The Escape's vehicle state is defined as a 5 x 1 column vector consisting of northing, easting, heading, velocity and wheel angle. The state vector is maintained by an Extended Kalman Filter, which is updated every measurement cycle. Northing, Easting and Heading are all taken for GPS, speed is taken from VSS, and wheel angle is derived from the angle of the steering wheel. A simple vehicle dynamics model is currently used by the EKF to update the measurements, although the more advanced dynamics models in this paper will soon be applied.

2.4 Modeling

A car's actual motion is extremely complex because of all the moving parts and non-uniform forces acting upon it. However, the forces involved in vehicular motion can

be simplified to yield a model which is both easy to implement and reasonably accurate. We examined three models of the car's 3DOF dynamics. The first is purely geometric; the second is born out of the observation that the dominant forces on a car are the cornering forces and the third is an attempt to empirically define the car's state with real world data.

2.4.1 The Canyon Model

The "Canyon Model" was a basic geometric model which was implemented on Prospect Eleven for the 2005 Grand Challenge, although it was superseded by a high-precision GPS/IMU system that was loaned to the team for the race. The model had two wheels, front and rear, with front wheel drive only. The model assumed that the car perfectly tracked the angle made by the front wheel without slipping. Below is a list of the overall assumptions followed by the model's equations of motion due to steering and speed inputs:

Assumptions:

- Steady State: The vehicle velocity and turning angle are held constant.
- 2-D: The vehicle motion is two dimensional.
- No lateral weight shift: This assumes that the shift of the weight from one side of the car to the other under angular accelerations (mostly due to cornering) is negligible. The two wheel model then becomes approximately the same as a four wheel one.
- No longitudinal weight shift: This is similar to lateral weight shift except that this fixes the apparent center of gravity in the model. Otherwise, it would shift during steady state due to factors like aerodynamics.
- The road is level.
- Aerodynamic effects are negligible.
- Tire property effects (due to shape, pressure, etc.) are negligible.
- Tires do not slip.

Geometry:

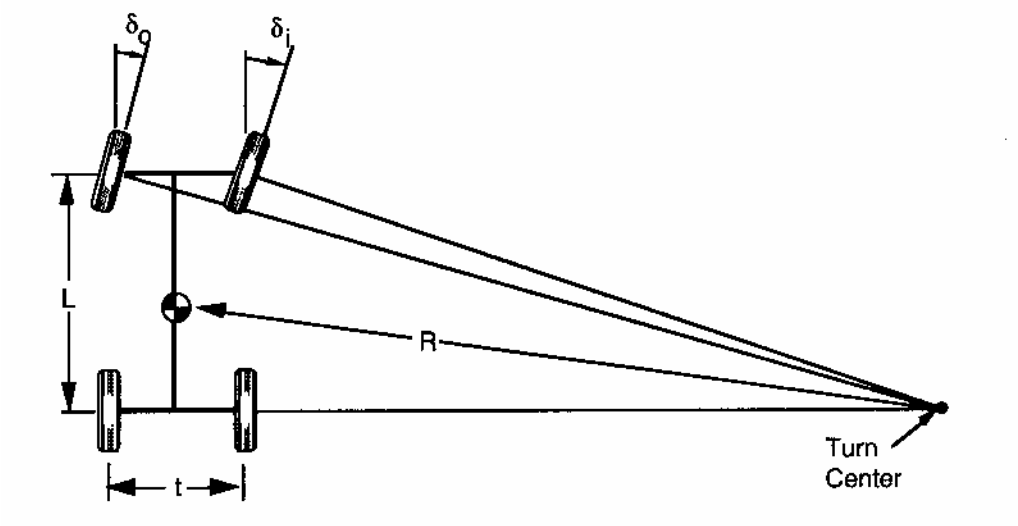

Figure 1: Ackerman Geometry ([1], 197)

The diagram above yields the following relationship:

$$\delta = avg(\delta_o, \delta_i) = \sin\left(\frac{\ell}{R}\right)$$

This can be rewritten to give tracking radius as a function of wheel angle:

$$\frac{1}{R} = \frac{1}{\ell}\sin^{-1}(\delta)$$

Using the definition of yawing velocity as angular velocity in the x-y plane, yawing velocity becomes:

$$r = \frac{V}{R} = \frac{V}{\ell}\sin^{-1}(\delta)$$

Equations of Motion:

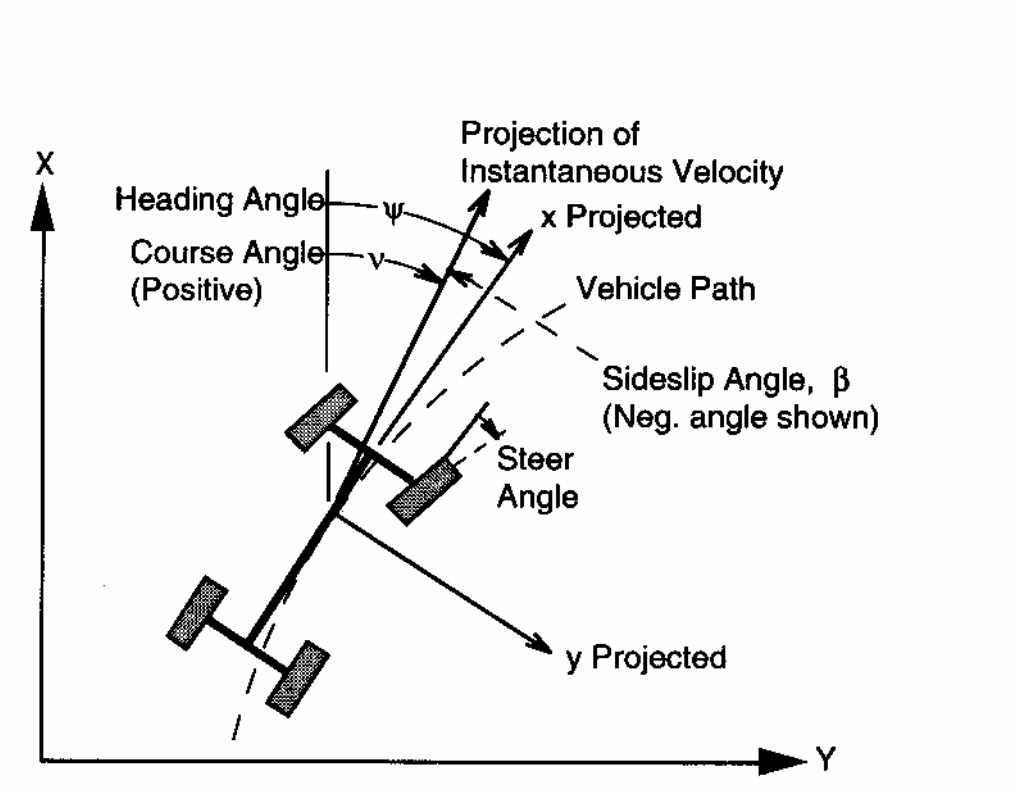

Figure 2: Earth-Fixed System ([1], 9)

The basic system of equations for 3DOF motion is:

$$X = X_0 + \dot{X}t$$
$$Y = Y_0 + \dot{Y}t$$
$$\psi = \psi_0 + \dot{\psi}t$$

The velocities in the x and y directions can be substituted in to yield:

$$X = X_0 + V\sin(\psi)t$$
$$Y = Y_0 + V\cos(\psi)t$$

The derivative of heading is yawing velocity, so the heading equation can be rewritten as:

$$\psi = \psi_0 + rt = \psi_0 + \frac{V}{\ell}\sin^{-1}(\delta)t$$

The equations above are continuous and dependent on an initial state. Often it is more useful to have equations dependent on the previous state. Additionally, real world data is discrete, not continuous. Below are the same equations rewritten to reflect this:

$$X_{j+1} = X_j + \Delta X = X_j + V\sin(\psi_j)\Delta t$$
$$Y_{j+1} = Y_j + \Delta Y = Y_j + V\cos(\psi_j)\Delta t$$
$$\psi_{j+1} = \psi_j + \Delta\psi = \psi_j + r\Delta t = \psi_j + \frac{V}{\ell}\sin^{-1}(\delta)\Delta t$$

2.4.2 The Bicycle Model

The bicycle model is a two wheel model which assumes that the dominant effect on vehicle steering is tire slip caused by angular acceleration. This means that in turning, the bicycle does not track the wheel angle exactly. It tracks a composite angle of both the wheel angle and slip angle. In other respects it is similar to the Canyon Model.

Assumptions:

- Steady State: The vehicle velocity and turning angle are held constant.
- 2-D: The vehicle motion is two dimensional.
- No lateral weight shift: This assumes that the shift of the weight from one side of the car to the other under angular accelerations (mostly due to cornering) is negligible. The two wheel model then becomes approximately the same as a four wheel one.
- No longitudinal weight shift: This is similar to lateral weight shift except that this fixes the center of gravity in the model. Otherwise, it would shift during steady state due to factors like aerodynamics.
- The road is level.
- Aerodynamic effects are negligible.
- Tire property effects (due to shape, pressure, etc.) are negligible.

[The following derivation is taken from pages 144-161 of Milliken]

Equations of Motion:

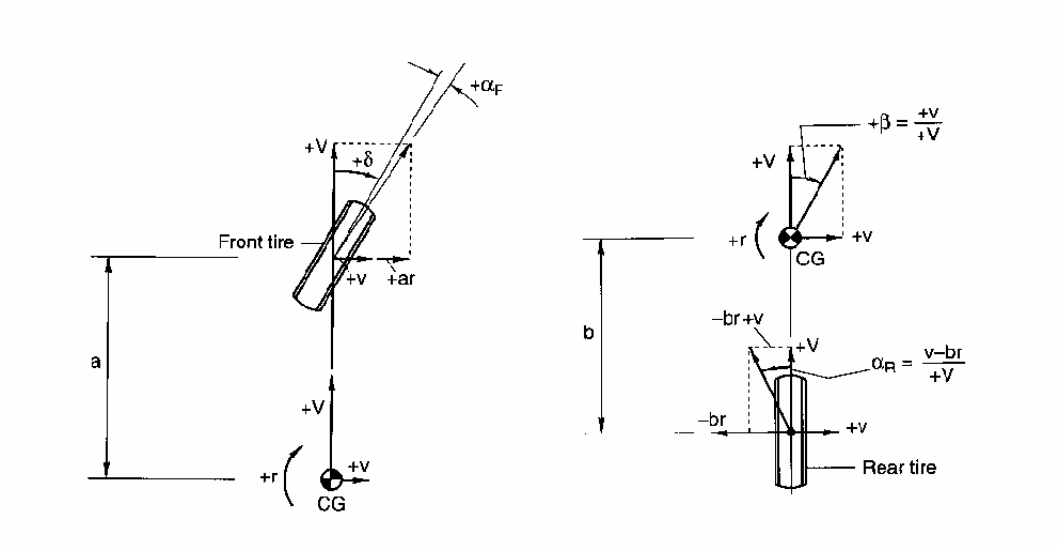

Figure 3: Tire Cornering Effects ([2], 147-148)

The basic forces and moments due to sideslip are:

$$N = I_z \frac{dr}{dt}$$

$$Y = ma_y$$

The side force is due to a combination of an angular acceleration (Vr) and a lateral acceleration ($\dot{v}$):

$$Y = ma_y = m(Vr + \dot{v}) = m(Vr + V\dot{\beta}) = mV(r + \dot{\beta})$$

Summing the individual tire forces yields the equations below:

$$Y_F = C_F(\beta + \frac{ar}{V} - \delta)$$

$$Y_R = C_R(\beta - \frac{br}{V})$$

$$Y = Y_F + Y_R = (C_F + C_R)\beta + \frac{1}{V}(aC_F - bC_R)r - C_F\delta$$

Note that the tire forces are assumed to be linearly related to the angles they make. Further assumption that sideslip angle, yawing velocity, and wheel angle are independent yields:

$$Y = f(\beta, r, \delta) = \frac{\partial Y}{\partial \beta}\beta + \frac{\partial Y}{\partial r}r + \frac{\partial Y}{\partial \delta}\delta = Y_\beta \beta + Y_r r + Y_\delta \delta$$

$$Y_\beta = (C_F + C_R)$$

$$Y_r = \frac{1}{V}(aC_F - bC_R)$$

$$Y_\delta = -C_F$$

Combining the linearization above with the original force equation yields:

$$mV(r + \dot{\beta}) = Y_\beta \beta + Y_r r + Y_\delta \delta$$

A linearization of the sum of z-axis moments is given below:

$$N = N_F + N_R = Y_F a - Y_R b = C_F(\beta + \frac{ar}{V} - \delta)a - C_R(\beta - \frac{br}{V})b$$

Applying the assumption that sideslip angle, yawing velocity, and wheel angle are independent yields:

$$N = f(\beta, r, \delta) = \frac{\partial N}{\partial \beta}\beta + \frac{\partial N}{\partial r}r + \frac{\partial N}{\partial \delta}\delta = N_\beta \beta + N_r r + N_\delta \delta$$

$$N_\beta = (aC_F - bC_R)$$

$$N_r = \frac{1}{V}(a^2 C_F + b^2 C_R)$$

$$N_\delta = -aC_F$$

Combining the linearization with the original moment equation yields:

$$I_z \dot{r} = N_\beta \beta + N_r r + N_\delta \delta$$

In steady state motion, all the derivatives of velocity are zero. The basic equations of motion become:

$$\dot{r} = 0$$

$$\dot{\beta} = 0$$

$$mVr = Y_\beta \beta + Y_r r + Y_\delta \delta$$

$$0 = N_\beta \beta + N_r r + N_\delta \delta$$

Given that $r = \frac{V}{R}$, rearranging the force equation to solve for β yields:

$$\beta = -\left(\frac{N_\delta}{N_\beta}\right)\delta - \left(\frac{N_r}{N_\beta}\right)\left(\frac{1}{R}\right)V$$

Substituting into the moment equation to remove β and replace r with $\frac{V}{R}$ yields:

$$-Y_\delta\delta = Y_\beta\left[-\left(\frac{N_\delta}{N_\beta}\right)\delta-\left(\frac{N_r}{N_\beta}\right)\left(\frac{1}{R}\right)V\right]+\left(Y_rV-mV^2\right)\left(\frac{1}{R}\right)$$

Rearranged to group δ and R:

$$\left(Y_\beta N_\delta - N_\beta Y_\delta\right)\delta = \left[N_\beta\left(VY_r-mV^2\right)-VY_\beta N_r\right]\left(\frac{1}{R}\right)$$

This yields the following steady state response equations:

$$\frac{1/R}{\delta} = \frac{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)}{\left[N_\beta\left(VY_r-mV^2\right)-VY_\beta N_r\right]}$$

$$\frac{r}{\delta} = \frac{V/R}{\delta} = \frac{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)}{\left[N_\beta\left(Y_r-mV\right)-Y_\beta N_r\right]}$$

$$\frac{a_y}{\delta} = \frac{V^2/R}{\delta} = \frac{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)V}{\left[N_\beta\left(Y_r-mV\right)-Y_\beta N_r\right]}$$

Only one main steady state response equation remains, $\frac{\beta}{\delta}$:

$$\frac{\beta}{\delta} = \frac{-\left(\frac{N_\delta}{N_\beta}\right)\delta-\left(\frac{N_r}{N_\beta}\right)\left(\frac{1}{R}\right)V}{\delta} = -\left(\frac{N_\delta}{N_\beta}\right)-\left(\frac{N_r}{N_\beta}\right)\frac{V/R}{\delta}$$

Replacing $\frac{V/R}{\delta}$ with its response equation yields:

$$\frac{\beta}{\delta} = \frac{N_rY_\delta - N_\delta\left(Y_r-mV\right)}{\left[N_\beta\left(Y_r-mV\right)-Y_\beta N_r\right]}$$

For 3DOF motion, the response of interest is $\frac{r}{\delta}$, which is based on the response of $\frac{1/R}{\delta}$.

By substituting in the actual values of coefficients, the $\frac{1/R}{\delta}$ response can be simplified:

$$\frac{V\left(N_\beta Y_r - Y_\beta N_r\right)}{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)} = \frac{V\left((aC_F-bC_R)\frac{1}{V}(aC_F-bC_R)-(C_F+C_R)\frac{1}{V}(a^2C_F+b^2C_R)\right)}{\left(-(C_F+C_R)aC_F+(aC_F-bC_R)C_F\right)} = a+b=\ell$$

$$\frac{1/R}{\delta} = \frac{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)}{\left[N_\beta\left(VY_r - mV^2\right) - VY_\beta N_r\right]} = \frac{1}{\left[\frac{V\left(N_\beta Y_r - Y_\beta N_r\right)}{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)} - \frac{N_\beta mV^2}{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)}\right]} = \frac{1}{\left[\ell - \frac{N_\beta mV^2}{\left(Y_\beta N_\delta - N_\beta Y_\delta\right)}\right]}$$

Substituting a new coefficient into the response yields:

$$K = \left(\frac{m}{\ell}\right)\frac{N_\beta}{\left(N_\beta Y_\delta - Y_\beta N_\delta\right)}$$

$$\frac{1/R}{\delta} = \frac{1/\ell}{\left(1 + KV^2\right)}$$

$$\frac{r}{\delta} = \frac{V/\ell}{\left(1 + KV^2\right)}$$

$$\delta = \frac{\ell}{R} + \frac{\ell}{R}KV^2$$

The motion of the car in the bicycle model is very similar mathematically to the Canyon model (Figure 2). The main difference is in yawing velocity, which significantly changes the value of heading:

$$X_{j+1} = X_j + \Delta X = X_j + V\sin(\psi_j)\Delta t$$

$$Y_{j+1} = Y_j + \Delta Y = Y_j + V\cos(\psi_j)\Delta t$$

$$\psi_{j+1} = \psi_j + \Delta\psi = \psi_j + r\Delta t = \psi_j + \delta\frac{V/\ell}{\left(1 + KV^2\right)}\Delta t$$

2.4.3 The Constant Conditions Model

The Constant Conditions model is a nonlinear model of a car which tries to make as few assumptions as possible. It constrains the car to two dimensional steady state motion defined by wheel angle and velocity. The way the model avoids more constraints is by specifically not creating a theoretical model of the car. Instead, the model is simply a mathematical fit of real world data. The advantage the model has over the previous two is that it is much more accurate. However, there is no way to tell why the equation works and what the contributing factors are. Below is a list of the major assumptions of the model followed by the equations of motion, which will be derived later in the experimental data section:

Assumptions:

- Steady State: The vehicle velocity and turning angle are held constant.

- 2-D: The vehicle motion is two dimensional.
- Constant vehicle properties: All properties of the car are constant. This includes geometry, tires, etc. To this end, tire pressure will be set before vehicle use. Other properties are basically unchanging.
- Constant environment: All environmental factors which effect vehicle performance in any way are constant. This includes weather, road conditions, pavement type, air properties, etc. Constant road conditions also mean that the road is assumed to be level.
- All factors which affect the vehicle performance are dependent on velocity and wheel angle.

Equations of Motion:

The equations of motion are similar to the two previous models (see figure 2), but with a different R, as derived from the empirical data in Section 3.3.2.

$$R = 2.7665\frac{1}{\delta_{tire}} - 0.023732\log(\delta_{tire})V^2 + 0.11739$$

$$r = \frac{V}{R}$$

$$X_{j+1} = X_j + \Delta X = X_j + V\sin(\psi_j)\Delta t$$

$$Y_{j+1} = Y_j + \Delta Y = Y_j + V\cos(\psi_j)\Delta t$$

$$\psi_{j+1} = \psi_j + \Delta\psi = \psi_j + r\Delta t = \psi_j + \frac{V}{R}\Delta t$$

3. Experimental Data

The last two models are theoretically sound, but theory only gets them so far. The Bicycle model uses an "understeer coefficient" K, the value of which is unknown without measurement. Likewise, the Constant Conditions model is entirely defined by measured values and cannot be derived with theory alone. To complete these models, we needed to take data and calculate K and the Constant Conditions equation.

Since all the models are for steady state, only steady state data of heading, speed and position needed to be collected. To do this, we planned to use the exiting speed and steering controllers to maintain a constant steering angle and speed while recording heading, speed and position data as accurately as possible. This data collection procedure

entailed several other projects. GPS had to be installed, calibrated and integrated with the car's code. A speed controller and a steering controller needed to be designed, implemented and tuned. (These controllers are described in the sections 4 and 5, but for reference they are both PID). Finally, a logging service which time-stamped data needed to be implemented. The side projects were to some degree outside the scope of this project, but they were essential to our data collection.

3.1 Data Collection

Once the car was ready to record steady state data, we set the car at variety of wheel angles and speeds. The following table shows the desired wheel angles we selected, and the desired speeds at each angle.

Angle (deg)	Speeds (mph)
-90	5,10,15,25
-180	5,15,25
-270	5,15,20
-360	5,10,15
-450	5,10
180	10

Table 1: Summary of data runs

The following data was collected during each run:

GPS: Latitude, Longitude, Heading, Speed, Time

Speed: Actual, Desired, Time

Steering: Actual, Desired Time

3.2 Post-Processing

The raw data was logged in an SQL server database. A command line script was written that generated .dat files of the individual logs. Each of the three logs (GPS, Speed, Steering) per run were accumulated into an Excel spreadsheet. The data was then visually inspected and trimmed to the appropriate start and end time. The most important factor in trimming the data was ensuring that both and steering were at stabilized values for the entire dataset, as is required by the steady state assumptions of our model. Using existing libraries written by PAVE members, the GPS coordinates were transformed into a local two-dimensional Cartesian plane, centered on the Forrestal campus. Once the position data was in meters, a MATLAB function was used to

computer the radius and curvature for sets of three points, by calculating their circum circle. Initially, sets of three consecutive points were used, but the variation in GPS position made these radius computations unreliable. Instead, GPS points were chosen such that they were one second apart. Average turn radius for each run based on GPS position was then calculated

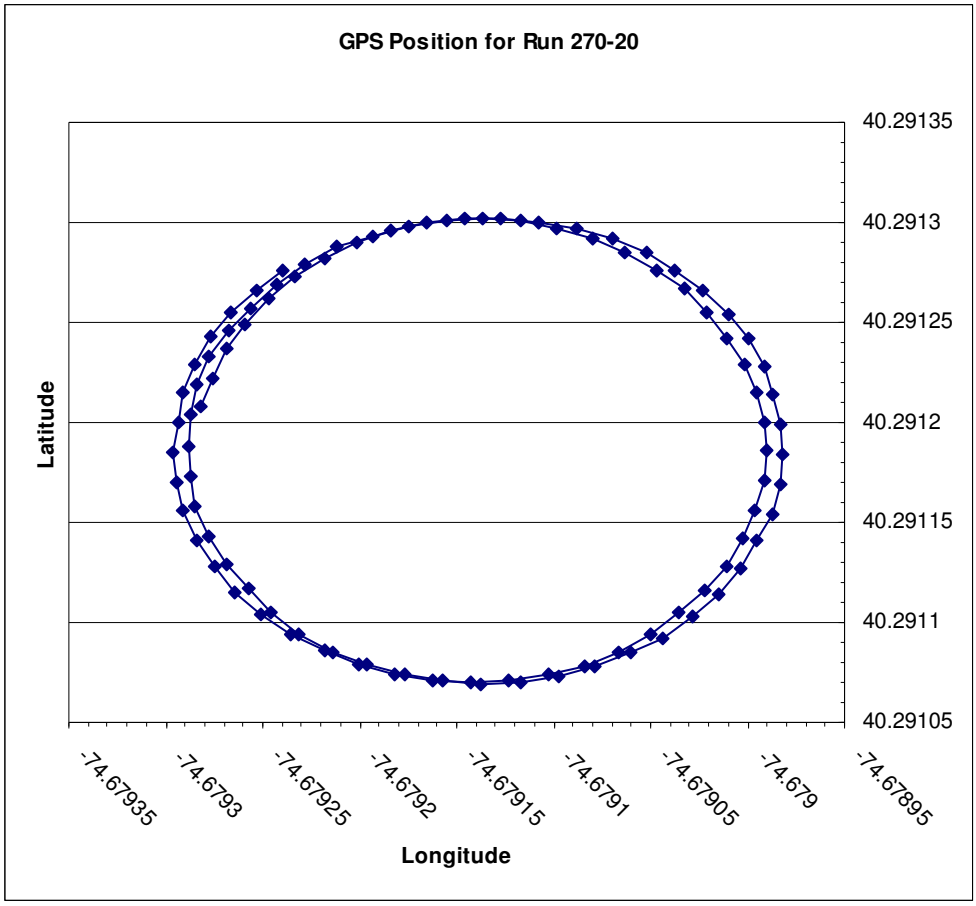

Figure 4: Example latitude/longitude plot

In addition, GPS heading was plotted against time. We determined that the slope of this graphs corresponded to yaw rate. Forward velocity divided by the yaw rate yielded another computation of the turn radius for the run. The slope was calculated using linear trendlines on the Excel graphs. The figure below is a sample plot of yaw rate, from run 180-15.

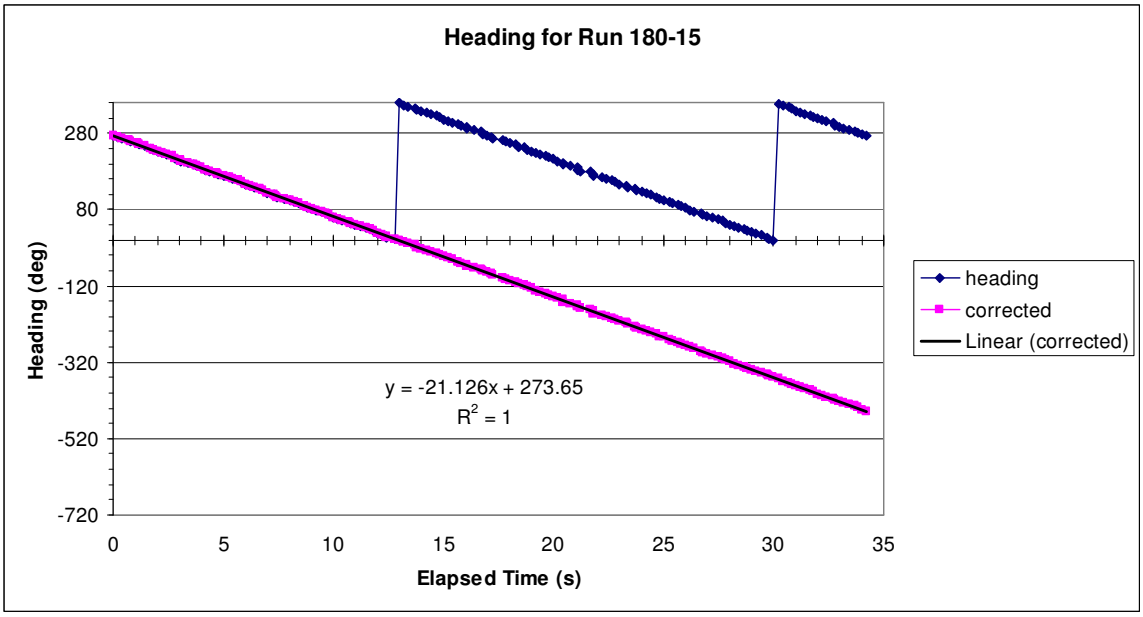

Figure 5: Sample heading and corrected heading chart, with linear fit.

The computation of turn radius from the two different methods produced extremely similar results. Further computations used the radius calculated from yaw rate, although the values were almost always within 1% or less, of one another.

Using the formulas from the bicycle model, the understeer coefficient, K was computer. Average speed from GPS data was used. The vehicle wheelbase is 2.691 meters and the steering angle was converted to wheel angle according to 17.9:1 reduction specified in the Escape manual.

Finally, the calculated and measured data from all the runs were summarized and plotted to evaluate trends.

3.3 Results

Table 2, below, summarizes the measured and computed data from all of the data collection runs. Runs at zero angle were not conducted, though the data points are used to force the correct intercept of certain functions.

Run		Measured		Computed		
Angle	Speed	Wheel Angle	Speed (m/s)	Radius (m)	Curvature (1/m)	K
0	**5**	0	2.272727273		0	
0	**10**	0	4.545454545		0	
0	**15**	0	6.818181818		0	
0	**20**	0	9.090909091		0	
0	**25**	0	11.36363636		0	
90	**5**	0.087872931	2.196803403	31.97854762	0.031270964	0.01511563
90	**10**	0.087611716	4.39923769	32.70801198	0.030573549	0.004865322

90	**15**	0.087477539	6.68215211	33.90834417	0.029491266	0.002969167
90	**25**	0.083593619	11.55389003	40.22293934	0.024861435	0.002126276
90	**25b**	0.085001249	11.42526548	40.29171488	0.024818998	0.002357133
90	**25b**	0.084248469	11.41809443	41.12450473	0.024316402	0.002476746
180	**5**	0.175133181	2.107325703	16.62387532	0.060154445	0.025140095
180	**10**	0.175124956	4.675688102	17.81942228	0.056118542	0.008760988
180	**15**	0.176166396	6.514930421	17.66912889	0.056595886	0.00444134
180	**25**	0.176202499	11.10870755	21.60055855	0.046295099	0.003672947
270	**5**	0.262852294	2.093606379	10.82820089	0.092351445	0.019793204
270	**15**	0.263716647	6.33469187	11.46221723	0.087243155	0.003841985
270	**20**	0.264493126	8.454320057	12.9872073	0.076998848	0.004359249
360	**5**	0.351353201	2.048872063	8.007074687	0.124889556	0.017673921
360	**10**	0.350870503	4.12506085	8.162053284	0.122518191	0.005493604
360	**15**	0.349515886	6.222237306	8.680706537	0.115197996	0.004093232
450	**5**	0.438776782	1.975665484	6.209055673	0.161055087	0.010309435
450	**10**	0.437991482	4.025454282	6.364280933	0.157126942	0.003970423

Table 2: Summary of Collected Data

3.3.1 Understeer Coefficient

$$\frac{r}{\delta} = \frac{V/\ell}{\left(1 + KV^2\right)}$$

The constant K in the bicycle model's yawing velocity response above is called the understeer coefficient. It describes how much the car understeers with respect to a car which steers with both front wheels perfectly parallel. Ideally, K is a constant. However, the graphs below show that this is not the case. The understeer coefficient does appear to be somewhat constant across a range of wheel angles, but not across a range of speeds. In fact, the coefficient appears to decay exponentially with speed. This indicates that the bicycle model is inherently flawed. The assumption that cornering forces are the dominant forces governing vehicle motion is not true. If they were, the understeer coefficient would be at least somewhat constant. But, the coefficient changes significantly and systematically. This also indicates that we cannot use the bicycle model with any degree of accuracy, except within a region close to the speed at which our chosen understeer coefficient matches an experimentally measured one. If pressed, we could salvage the model somewhat by modeling the coefficient as an empirical function of velocity. This would probably yield a fairly accurate model, but still an inherently flawed one.

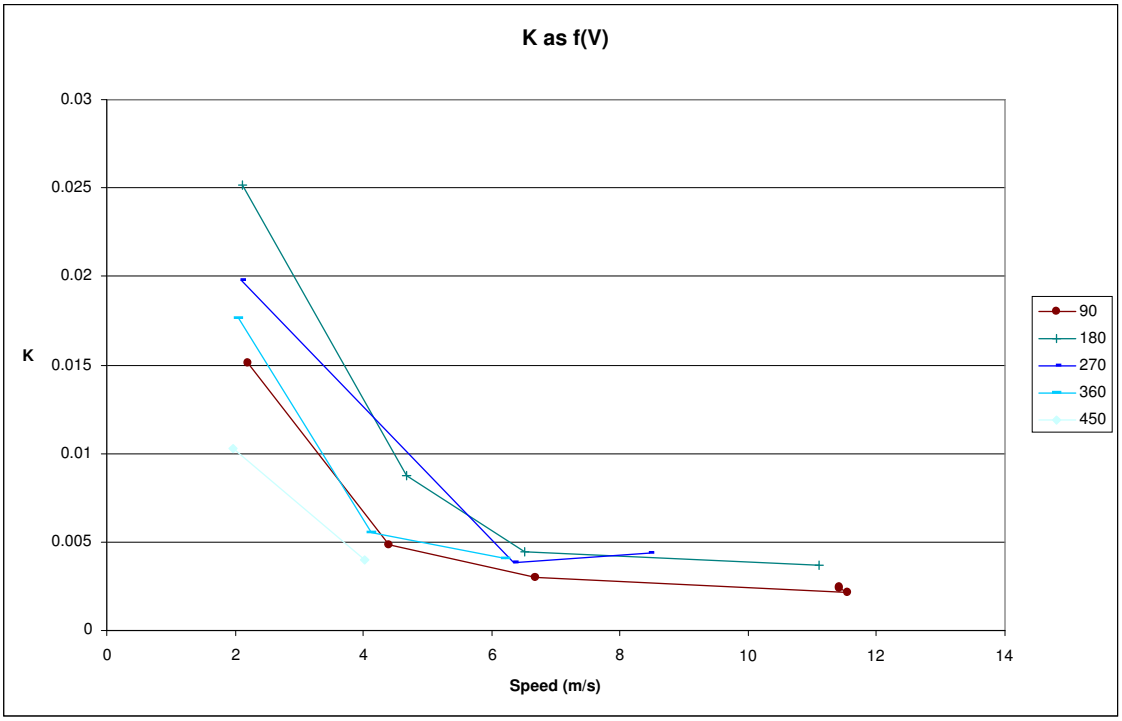

Figure 6: Understeer coefficient as a function of velocity

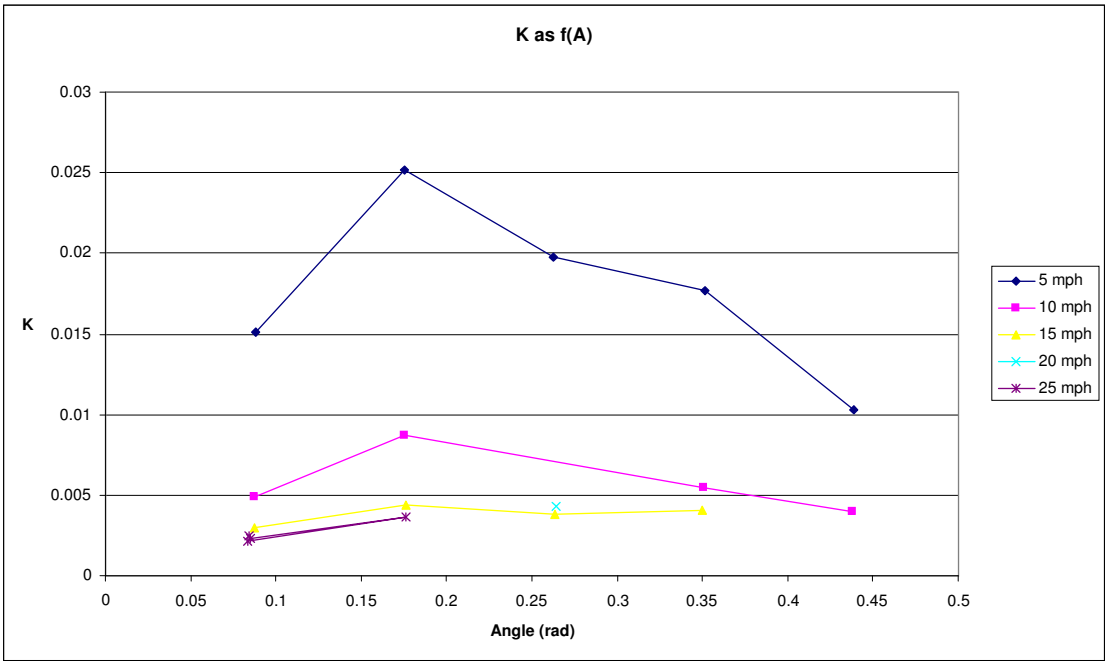

Figure 7: Understeer coefficient as a function of wheel angle

3.3.2 Curvature / Radius

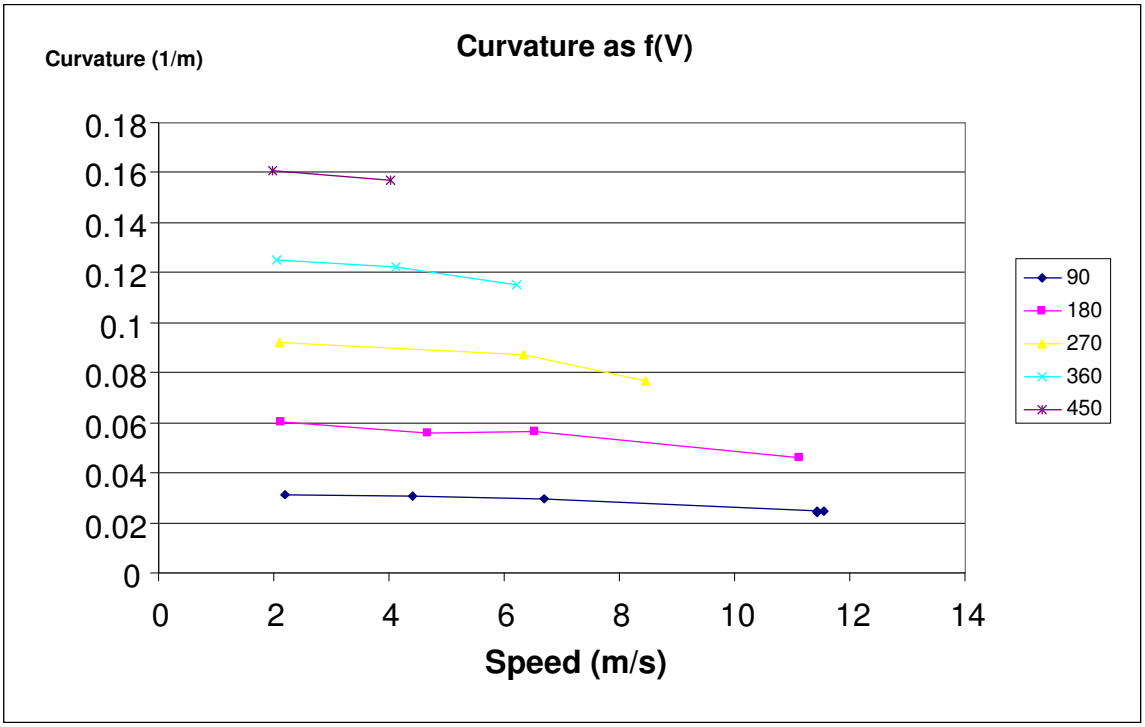

Figure 8: Curvature as a function of velocity

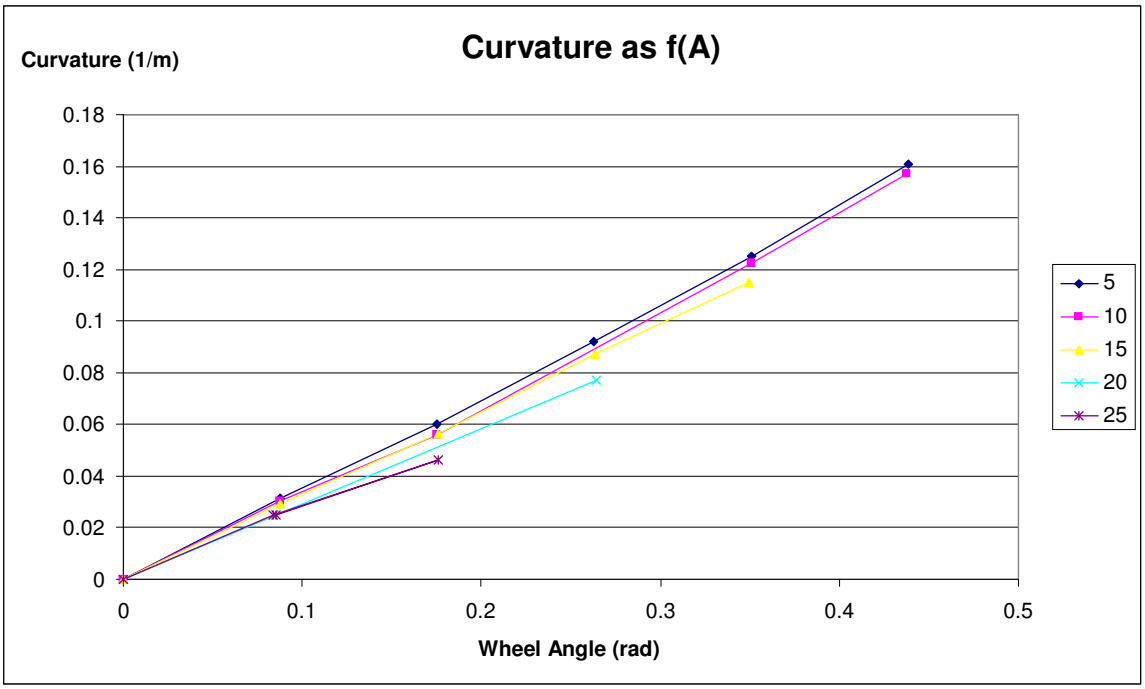

Figure 9: Curvature as a function of wheel angle

Measuring curvature (and hence radius) at a variety of speeds and wheel angles is crucial to finding a Constant Conditions equation. The trick to creating the Constant Conditions model is finding an equation which agrees (to within a certain range) with the

experimental data points. The way we did this was by taking the data points graphed above and plugging them into an equation finder [8]. The equation finder searches for a mathematical fit by brute force, comparing the data to thousands of mathematical function types. The resulting Constant Conditions equation and graph for the data above is given below:

$$R = 2.7665\frac{1}{\delta_{tire}} - 0.023732\log(\delta_{tire})V^2 + 0.11739$$

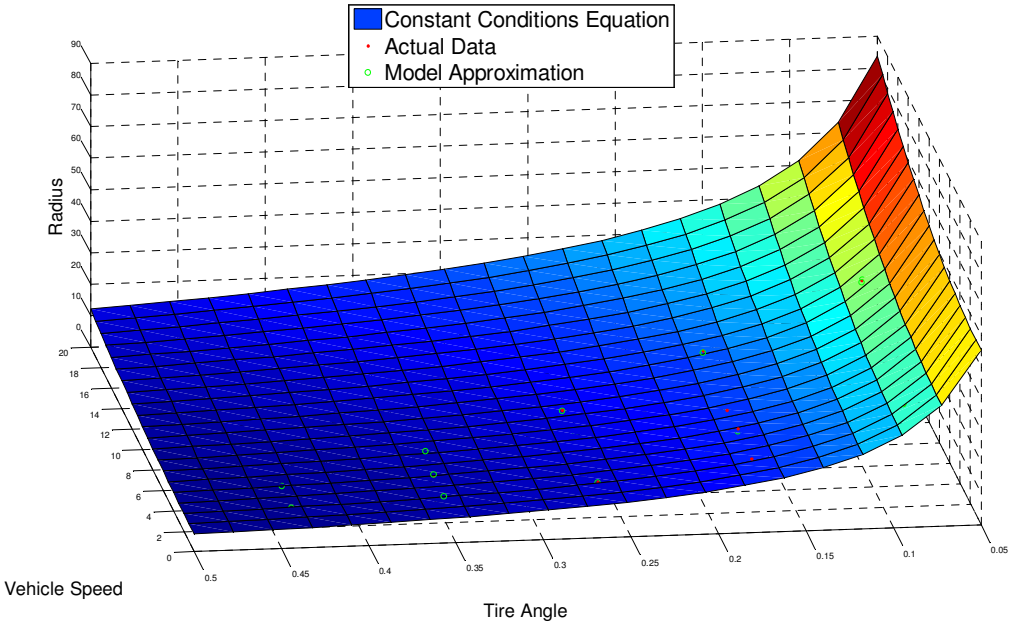

Figure 10: 3D plot of radius as a function of wheel angle and speed

As can be seen, the Constant Conditions model fits the data fairly well. It is not exact, but it is far closer than the previous two models. In all likelihood, however, this model is only accurate around the speeds and angles the data was recorded at. To prevent this from being a problem, the data we took covered the range of vehicle motion we expect to see on average roads.

4. Actuation

This section is a brief overview of the methods of actuating the car's inputs. While this is technically beyond the scope of this project, it is important to mention since the actuators and their controllers are the plants upon which our control systems operate.

4.1 Steering

To measure steering angle, a high-precision optical rotary encoder was installed on the steering column. This is the feedback sensor for the low-level steering controller. Steering wheel actuation was accomplished with some patient reverse engineering: The power steering module was tapped to turn the wheel by sending simulated signals of the power steering torque sensor. This relies heavily on data acquisition cards and custom electronics.

With both control of the wheel and position feedback, a low-level controller was designed. The steering controller is low-level since it controls steering wheel position and not vehicle motion. We settled on PI control, since we discovered that the steering wheel has high internal damping and did not need a derivative term. The proportional term is k_P =-0.25, while the integral term ia k_I =-0.002. The integral term is tiny, just enough to ensure zero steady state error without causing significant oscillations in the steady state.

The actual plant of the steering control system is not perfectly known. Until recently, we were unable to take accurate step response data. With the newly finished logging system, we now have the tools to take the data properly. With the newer data we fully expect to tune the steering controller to be even better.

4.2 Brakes and Throttle

Both the brakes and the throttle are controlled and measured electronically through the Antilock Braking System and throttle modules respectively. These again rely heavily on data acquisition cards and custom electronics. Similar to the steering system, the plant for controlling speed was not well known. We now know that brakes and throttle are not equal, nor are they entirely linear. We plan to correct for this in future control systems. However, even with the error from the nonlinear and somewhat lopsided system, we managed to create a controller which performed very well.

The speed controller is actually a higher level controller since it is directly controlling vehicle motion. As such it will be dealt with in the control section of this paper.

4.3 Transmission

Unlike the other systems, the transmission needed custom mechanical hardware to actuate it. However, it was still possible to measure the transmission position electronically. A PID control is being implemented to control the transmission state with feedback from the electronic transmission measurement. This system is not of any real interest to our project since our controllers do not operate on it.

5. Control

The previous section laid out the design for both transmission and steering controllers. However, those controllers are considered low-level, designed simply to control actuators. To follow a specified path, the car needs higher-level controllers on top of low-level ones, using state estimation for feedback.

One fundamental simplification in our path-following approach is the decoupling of the lateral and longitudinal controls. This is a distinct separation from other solutions, such as full state-based control. Not only does the non-holonomic property of an automobile make state-based planning and control substantially more difficult, the interactions between the longitudinal and lateral are substantially negligible such that the two may be safely decoupled. Decoupling these systems allows them to be tuned and tested independently, greatly simplifying and expediting the implementation of the entire robotic system.

5.1 Longitudinal

Longitudinal control is fairly straightforward. It is simply a matter of trying to control forward motion along a path. We looked into two ways of accomplishing this, speed control and position control. Fundamentally, the two controllers perform identically, but conceptually they are unique.

It should be noted that we assume brakes and throttle have identical, linear responses. In reality this is not the case, especially for brakes, and our controller performance suffers a little as a result. In the future, we plan to linearize the systems and add a gain to the brakes in order to make brakes and throttle equivalent outputs to the control.

5.1.1 Speed Control

The speed control takes in a target speed and use PID to track that reference speed. As the desired path changes (new speed limits, etc.), so does the target speed. The controller is currently implemented as PI with constants $k_I = 0.001$ and $k_P = 0.05$.

On the open road, this controller is the most intuitive to envision. The ideal target speed is the speed limit. When approaching a stop line, the target speed is zero. However, this method is not guaranteed to stop the car at the stop line without added calculations to set how the target speed changes as it approaches the stop line. The same applies to following behind another car. Setting the car's target speed to be the same as the vehicle it is following ensures that it does not drive into the vehicle. However, it does not ensure a safe following distance. Added calculations are needed to take that into account.

5.1.2 Position Control

Position control is extremely useful in many cases we expect to encounter in the Urban Challenge, such as following a fixed distance behind a car or stopping at a stop line, that are based on desired distances. Fitting step response data yields the following empirical relationship between vehicle position and throttle/brake input:

$$Y(s) = \frac{k}{s(s+b)} U(s)$$

The controller below is based on PI with an additional double integral term to make the controller have zero steady state error for ramp inputs:

$$C(s) = \frac{k_{II}}{s^2} + \frac{k_I}{s} + k_p$$

Given an input to the controller of an arbitrary distance, the car will drive that distance and stop. This ensures stopping at the stop line as well as staying a safe following distance behind other cars (given a constant distance input equal to the following distance). However, this method of control is not as useful as speed control when there are no obstacles in the path. Ramping the distance input is equivalent to setting the desired speed. The double integral term in the controller allows the car to track a ramped position input (i.e. a speed) with zero steady state error.

5.1.3 Hybrid Control

A hybrid method of longitudinal control would use speed control in open areas and position control in environments with obstacles. This takes advantage of the strong points of each system but has the major drawback of needing to implement two controllers and accurately switch between the two. In fact, since each controller can implement the other (with a little extra calculus); a hybrid system is simply a more elaborate setup for the sole purpose of being easier to understand.

5.2 Lateral

Lateral control is not as simple as longitudinal control. Whereas longitudinal control has one degree of freedom (along the longitudinal axis), lateral control has two. This is due to the fact that the car must yaw in order to move along the lateral axis. Yawing also introduces movement in the longitudinal axis, but it is negligible compared to the overall longitudinal motion. The following are three methods of control we investigated.

5.2.1 Point Following

Point following is a simple method of control, used in Prospect Eleven and many other competitors in the 2005 Grand Challenge. It is a pursuit algorithm which takes a nearby point on the path and aims the car to drive directly at that point. Technically this is open loop control since there is no feedback term – the feedback takes place in the path-planning stage. It relies on a fast update rate in order for the car's path and the ideal path to converge. The benefits to this controller are that it is easy to implement and it is accurate most of the time. However, the downside of this control method is that a following point needs to be chosen. If the point is too far away, the car will undercut a turn and possibly drive off the road. If the point is too close, sensor noise will cause the car to oscillate sharply about the path in pursuit of a point which appears to be jumping around.

This happens because the angle between the pursuit point and the front wheels of the car is computed based on the distance – the look-ahead distance acts like an inverse proportional gain. Large distances make the controller unresponsive, whereas at smaller

distances, position sensor noise is amplified and becomes substantial portion of the controller.

In addition, there is no good formula for choosing a look-ahead distance. It is usually chosen empirically based on tests of vehicle performance.

5.2.2 Radial Control

Radial control is a method we devised as a step up from point following. Point following assumes that the car can drive in straight lines. This is not the case; the car yaws as it moves laterally. In order to compensate, we decided to try a radial pursuit algorithm. It would take an arbitrarily chosen arc length along the path, and fit a circular curve between it and the car. It would then set steering angle based on the radius of the curve and vehicle speed. As a pursuit algorithm, the radial control is more accurate than point-following simply because it takes vehicle dynamics into account. However, the problem of choosing an arc length to pursue is analogous to choosing a point in the point following controller. If it is too long, it will cut corners, albeit less than point following would. If the arc length is too short, sensor noise will make the car movement jittery.

5.2.3 Cross-Track Error Controller

The cross-track error controller (insert 2 refs) avoids the problems of a pursuit algorithm by using an entirely different method. It is a nonlinear closed loop controller which tracks the lateral error of the car. This lateral distance is measured as the shortest line between the center of the front axle and the desired path. The basic control equation is below. It is designed to have a feed forward term as a rough guess of how much to turn and a term proportional to the lateral error which causes the error to converge exponentially to zero:

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{u(t)}\right)$$

δ is the wheel angle, ψ is yaw in the body frame, e is the lateral error and u is the velocity at the center of the front axle. More advance versions of the controller add terms adjusting the equation. For instance, the measured yaw could be reduced by the steady state yaw calculated from a dynamics model. This makes the feed forward term more

accurate since the car will tend to turn the steady state amount without added input. A low level steering controller could be integrated into the cross-track error controller. Stanford's implementation of the controller does just that, adding a term corresponding to a lead compensator on steering angle. In addition, Stanford also added a term damping the yawing rate to change the dynamics of the vehicle.

6. Conclusion

6.1 State Estimation Models

The data we collected indicates that only one of our three dynamics models used to estimate state, the Constant Conditions model, is accurate. Given that speed affects tracking radius, the Canyon model is obsolete. It gives radius as a function of only wheel angle. The bicycle model is also flawed since its "constant" understeer coefficient is not constant. The model can be modified by empirically finding an equation for understeer as a function of speed, but that violates the model's assumptions. As such, it is more reliable to simply fit the data empirically in the Constant Conditions model.

6.2 Measurement

In examining the vehicle speed data, we came across several interesting trends when comparing speed from GPS to speed from VSS duty cycle.

For starters, data taken while traveling about 18 m/s (40 mph) down a straight stretch of U.S. Route 1 indicated that the VSS speed estimate was about 1% lower than the GPS speed reading. This indicated to us that the VSS measurements needed to be calibrated. In fact, that code module hadn't been calibrated before we used it. The 1% offset came from plugging in the default value of 31924 ticks/mile the first time around.

Keeping in mind that VSS was about 1% lower than it should be, we then examined the data taken during our test runs. It appeared that the VSS speed was consistently *higher* that GPS measured speed, before factoring the 1% offset. (After the offset, there would have been a greater discrepancy). This offset seemed to be consistent over all of the runs.

In an effort to keep the variables in our test data to a minimum, all of our test runs were conducted with left-hand turning angles. However, we did take one data run at a right-hand turning angle, just to ensure that the vehicle performance was not turn-direction dependent.

Upon noticing the curious speed-offsets, we examined this additional data set. It turned out that the right-hand turn data had VSS speed as substantially *lower* that GPS; more than could be explained by a 1% offset. By rough estimate, after the 1% offset was applied in the correct direction to both Right- and Left-hand turns, it appeared as if each VSS speed was displaced an equal amount from the GPS speed; Right-turns was lower, left-turn was higher.

This implied that the VSS was being derived solely from wheel ticks from the right-side of the vehicle (the right-side, being outer-most in a left turn, tracks a larger radius and therefore travels faster.

To test this hypothesis, we took the faster speed measured on the VSS during a left turn and using the radius and speed calculated from GPS, predicted the radius implied by that faster speed.

Average this implied radius over the data set, we found that it was approximately .8 meters, or roughly half the track width of the car. Since the GPS is mounted along the centerline, this distance is what we would expect to see if comparing GPS speed with the right-side wheel.

We have not had a chance to contact Ford Motor Company about this interesting discovery. Nor have we been able to find any explanation in the manual. However, it is our belief that the VSS is not based on an average of all four wheels, but instead simply repeats the pulsetrain from one of the right-side wheels.

6.3 Future Work

Although a lot was accomplished in the process of completing this report, we also uncovered many more areas for improvement. What follows is a brief description of a number of futures project we anticipate undertaking as a result of this research.

- Steering Control. We would like to model the Plant of the steering system, by taking a system ID and design a better steering controller based on PI, PD, PID, or backing out some other non-linearities.
- Throttle and Brakes. We plan to linearize the brakes system, and derive a constant gain factor between throttle and brakes, so that the high-level speed control system can be linearized.
- Implement a more advanced version of the cross-track error controller, as described in [3] as opposed to [4]. We intend to use the Constant Conditions model data collected in this paper in the feed-forward term of the cross-track controller. Additionally, we want to look into a steady-state yaw offset, tweak the dynamics of the car's motion to be smoother, and try conflating with a low-level controller to decrease coupling effects.
- Kalman Filter. An immediate project will be to update the vehicle model used in by State Estimation for the EKF state updates.
- Finally, we would like to try and set up a Simulink simulation of the control system to test all the various components as well as the whole system

References

(1) Gillespie, Thomas, *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers, Warrendale, PA, 1992.

(2) Milliken, William and Milliken, Douglas, *Race Car Vehicle Dynamics*, Society of Automotive Engineers, Warrendale, PA, 1995.

(3) Thrun, et. al. *Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validationn and Racing*. To Appear in the Proceedings of the 26th American Control Conference, New York, July 2007

(4) Thrun, et.al., *Stanley: The Robot that Won the DARPA Grand Challenge,* Journal of Field Robotics 23(9), Wiley-Interscience, 2006.

(5) United States Congress, National Defense Authorization Act of 2001, Public Law 106-398, Section 220.

(6) Weiner, Tim. "Pentagon Has Sights On Robot Soldiers" *New York Times News Service*, New York, February 2005.

(7) Wong, J.Y., *Theory of Ground Vehicles*, Wiley-Interscience Publication, New York, 1993.

(8) ZunZun.com Online Data Modeling, *Fitting Results for User-Selectable Polynomial*, website: http://zunzun.com/