

Steering Control for a Skid-Steered Autonomous Ground
Vehicle at Varying speed

Suresh Golconda

February 23, 2005

Acknowledgements

I would like to thank my advisors, Dr. Arun Lakhotia and Dr. Anthony Maida, for their valuable guidance and extraordinary support. The time they spent with me in discussing the research idea aided in the fruition of my research. I greatly appreciate their patience and encouragement all throughout my thesis. I also would like to extend my deepest gratitude to the committee member, Dr. Charles D. Cavanaugh.

Special thanks to Joshua Bridevaux, for the excellent support he provided and for the countless hours he spent with me, testing the algorithms in the field. Special thanks to Pablo Mejia for the expert guidance he provided me throughout the thesis. I want to thank Firaz Bouz, Nitin Jyoti and Adrian Aucoin for their active support. Also, I thank other members of Team CajunBot for their valuable feedback.

I want to thank Dr. Anthony Maida again, for trusting me, for providing constant support and encouragement, right from my initial days in this country and at the university.

I would like to acknowledge my pet dog, Rocky, who is now in his final stages of life, whose sweet memories gave me incredible energy during my work.

Table of Contents

1	Introduction	1
1.1	Autonomous Ground Vehicle (AGV)	1
1.2	Research Objectives	2
1.3	Research Contributions	4
1.4	Significance of the Research	4
1.5	Organization of the Thesis	5
2	Background	6
2.1	CajunBot	6
2.2	Steering Mechanisms	6
2.2.1	Advantages of Skid Steering	8
2.2.2	Disadvantages of Skid Steering	8
2.3	Control Systems	10
2.3.1	PID Controller	11
2.4	Smith Predictor	14
2.5	Issues in Steering Control	14
3	Steering Controller	17
3.1	Steering Controller	17
3.2	Vehicle Control Strategies	19
3.3	Steering Algorithm	20
3.3.1	PID Controller	22
3.3.2	Safety Filter	22
3.3.3	Prediction Filter	23

3.3.4	Final Equation	24
3.4	Skid Steering Commands	24
4	Insight into the Steering Controller	25
4.1	Tuning the Algorithm	25
4.2	Influence of the Filters	26
4.3	Comparison with a Pure PID Controller	27
5	Empirical Evaluation	31
5.1	Experimental Design	31
5.1.1	Procedures	31
5.1.2	Independent Variables	31
5.1.3	Dependent Variables	32
5.2	Data Analysis	33
6	Conclusions and Future Work	39
	Appendix	41
1	Derivation of PID controller for steering	41
2	Preliminaries	42
3	Progressive Steering Controller	43
4	Conversion of the Progressive Algorithm to PID Form	45
	References	49
	Abstract	50
	Biographical Sketch	52

List of Tables

1	Description of the graphs	34
2	Basic functions	42
3	Vehicle parameters	43
4	Instantaneous measures at time t	43
5	Comparison of PID parameters and steering controller parameters	46

List of Figures

1	Picture of CajunBot	2
2	Single-axle explicit steering [15]	7
3	Double-axle explicit steering [15]	7
4	Skid steering [15]	8
5	Turns in skid steering [15]	9
6	General architecture of a closed-loop control system	10
7	Open-loop control system	10
8	General usage of Smith Predictors in control systems	15
9	Level 0 data flow diagram of steering controller	17
10	Level 1 data flow diagram of steering controller	18
11	Illustration of the “Way-point” mode steering control strategy	20
12	Illustration of the “Follow-the-carrot” mode steering control strategy	21
13	Level 2 components of steering algorithm.	21
14	Illustration of the source of the steering commands	28
15	Suggested path for the experimental runs shown in Figure 16, 17, 18, 19	33
16	Heading error for safety off and prediction off	35
17	Heading error for safety off and prediction on	36
18	Heading error for safety on and prediction off	37
19	Heading error for safety on and prediction on	38

1 Introduction

1.1 Autonomous Ground Vehicle (AGV)

An Autonomous Ground Vehicle (AGV) is a fully automated vehicle that can travel on a specific route on the ground without any human intervention.

Autonomous ground vehicles are already in use for space exploration, for example, the NASA's Mars Rover [8, 7, 6]. The Robot Sheepdog Project (RSP) developed by Silsoe Research Institute in collaboration with the University of Bristol, University of Leeds, and Oxford University uses autonomous vehicles to gather and maneuver a flock of ducks to a predetermined location safely [22, 21, 23]. The Autonomous Crop Treatment Vehicle [18, 4] developed by Silsoe Research Institute is used for treating crops with fertilizers and other chemicals. PackBot Scout and Military R-Gator, developed by iRobot¹, are commercial military surveillance autonomous vehicles.

The research and development effort building AGVs so far has focused on developing vehicles that operate on limited terrain conditions and on applications where speed is not an important issue. These vehicles are not conducive for use in battlefields. The US Congress has chartered the Defense Advanced Research Projects Agency (DARPA²) to bridge this gap and produce AGVs that can be used in the front lines by 2015. To accelerate the research and development of AGVs, DARPA conducted the DARPA Grand Challenge-2004³ competition. The AGVs developed by the teams in this competition were expected to complete a 210 mile off-road course from Barstow, California to Primm, Nevada in a maximum time of 10 hours. To complete 210 miles within 10 hours the average speed of a vehicle should be more than 20 miles per hour, a speed at which the vehicle needs a highly

¹iRobot's Military R-Gator, <http://www.packbot.com>.

²Defense Advanced Research Projects Agency, <http://www.darpa.mil>.

³DARPA Grand Challenge-2004, <http://www.darpa.mil/grandchallenge>.

sophisticated steering control algorithm to stay on the suggested path.

This thesis came about by attempting to develop a steering control algorithm for CajunBot, a six-wheeled AGV developed by the University of Louisiana at Lafayette for the DARPA Grand Challenge 2004. CajunBot, shown in Figure 1 was developed on a MAX IV 6-wheeled All Terrain Vehicle (ATV). The MAX IV ATV has three wheels on each side and each set of these three wheels is controlled independently. The ATV is steered to the left by locking or slowing down the wheels on the left. Steering to the right is achieved similarly by locking the right set of wheels. The steering mechanism of CajunBot commonly referred to as a skid steering mechanism is the same as that of an army tank or bulldozer.



Figure 1: Picture of CajunBot

1.2 Research Objectives

The objective of this research is to develop a steering controller for a skid-steered autonomous vehicle, like CajunBot, with the following goals:

1. It should steer the vehicle to follow a path consisting of a sequence of GPS locations

with minimal overshoot and oscillations;

2. It should be able to maneuver the vehicle at varying speeds;
3. It should not attempt turns that may be detrimental to the vehicle's safety.

The challenges in achieving the above objectives can be placed in three categories:

1. challenges in developing any control system;
2. challenges in steering control of an automobile;
3. challenges specific to a skid-steered vehicle.

One common challenge in developing a control system is accounting for *dead time*. This is the time elapsed between when a controller issues a command and when it receives feedback about completion regarding the command. As the dead time increases, the control problem becomes more difficult. As the controller may not have accurate information of the execution status of its commands, it may make incorrect decisions and continue giving commands to turn when the previous commands may have been sufficient to complete the turn.

The control of an automobile at high speeds introduces more challenges. As the speed increases the effect of the dead time becomes more pronounced. The vehicle would have traveled a much greater distance during the delay, thus causing the vehicle to overshoot or oscillate.

The steering control problem becomes more difficult for a skid-steered vehicle as the speed and steering control for skid-steered vehicles are interdependent. The amount of braking to be applied in order to achieve a turn varies with the speed of the vehicle.

A skid-steered vehicle's steering behavior is more dependent on the terrain on which the vehicle is traveling because of the very fact that the skid-steered vehicle skids on the

ground surface in order to achieve a turn. Thus, a steering controller must also take into account the characteristics of the terrain on which the vehicle is traveling.

1.3 Research Contributions

In this thesis we develop a Proportional Integral Differential(PID) control system based steering algorithm for a skid-steered vehicle with the following properties:

1. it can steer the vehicle at varying speeds;
2. it ensures the vehicle safety by preventing sharp turns at high speeds;
3. it uses a prediction technique to prevent the vehicle from over steering during sharp turns or while trying to align itself onto a straight line.

The steering algorithm has been validated empirically in a simulated environment and on the real vehicle. The empirical results are presented in the thesis.

A small but significant side result from this thesis is an association of the general control PID parameters K_p and K_d in domain specific terms to steering aggressiveness, and maximum time to correct heading error. The association of parameters makes it easier to tune the control parameters and to make the parameters dependent on speed.

Safety and prediction techniques are applied as a sequence of filters over the computation from the PID equation. By separating these concerns in separate filters, the PID control system is used more effectively by concentrating on only a small range of heading errors and a high range of speeds.

1.4 Significance of the Research

There are three critical software modules of an autonomous vehicle:

1. Obstacle detection;
2. Path planning;
3. Steering control.

The obstacle detection module detects the position of any obstacles in the path. The path planning module builds a collision-free path around the obstacles. The control module is responsible for driving the vehicle on the planned path.

Of the three modules, the steering control module plays the most crucial role because in the absence of any obstacles in the path, an AGV can drive without the obstacle detection module and the path planning modules but not without the steering control module.

1.5 Organization of the Thesis

Chapter 2 provides background information on CajunBot, an overview of automobile steering mechanisms, control systems and Smith Predictors. Chapter 3 introduces the proposed steering controller. Chapter 4 discusses the implementation and results of the steering controller experiments. Chapter 5 gives the conclusions. Chapter 6 summarizes some limitations of the proposed algorithm and potential future work to address these limitations. The appendix provides the derivation of the PID controller.

2 Background

This chapter provides an overview of CajunBot on which the present steering control system is implemented. It also presents information regarding steering mechanisms and control systems. Finally, it presents discussion on difficulties in designing a control system for skid-steered vehicles in general and for CajunBot in particular.

2.1 CajunBot

CajunBot is built on a MAX IV 6-wheel all terrain vehicle with a 25 hp twin-cylinder engine. The vehicle weighs 1,200 lbs. After adding additional fuel tanks, it has a fuel capacity of 35 gallons and can travel at up to 35 mph.

The computing power of CajunBot is provided by two high-speed AMD computers, several micro-controllers, and many custom circuits. CajunBot is equipped with two SICK LMS 221 scanning laser systems, three Doppler radars, and sonar sensors. For geo-location, vehicle orientation, speed, et cetera, it uses C-Nav differential GPS and an Oxford inertial navigation sensor. A 2-kilowatt electric generator supplies the electrical power for the computers, sensors, and other electronics.

2.2 Steering Mechanisms

The steering mechanisms used for maneuvering ground vehicles may be classified in three categories: single-axle steering system, double-axle steering mechanism, and skid steering. This section introduces each of these steering mechanisms emphasizing skid steering mechanisms and highlighting the advantages and disadvantages of skid steering mechanisms.

In a single-axle steering system as shown in Figure 2, front wheels are pivoted while rear wheels are fixed to the axle. This steering mechanism is most commonly used in

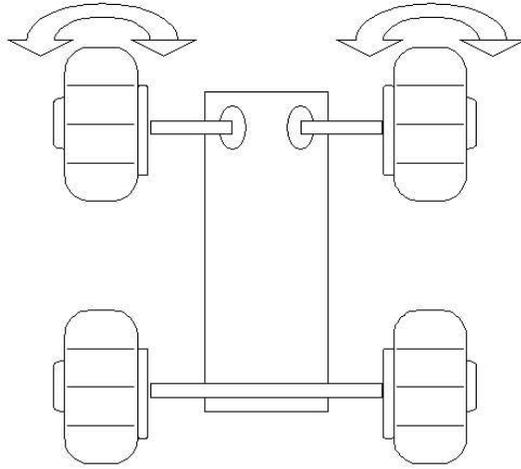


Figure 2: Single-axle explicit steering [15]

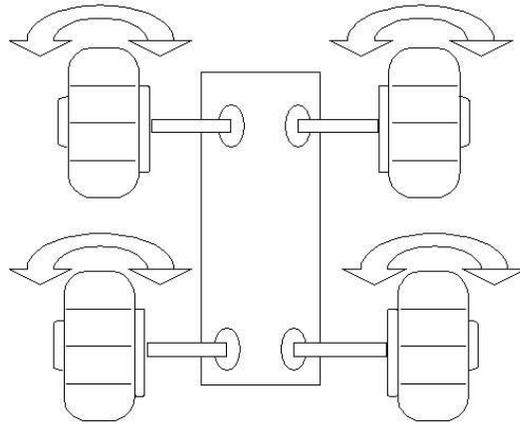


Figure 3: Double-axle explicit steering [15]

on-road vehicles. In a double-axle steering system as shown in Figure 3, both the front and rear wheels are pivoted. The vehicle can make sharper turns when compared to the vehicle with a single-axle steering system.

A skid-steered vehicle, as shown in Figure 4, does not have axles connecting the left and right wheels. Instead, all wheels on one side are connected via chains or other such mechanisms. The wheels on one side can be rotated independent of the other side. The vehicle may be turned by making the wheels on one side rotate faster than those on the other side. The difference in rotation causes the vehicle to skid and turn in the direction of the slow wheels; hence the term “skid-steer.”

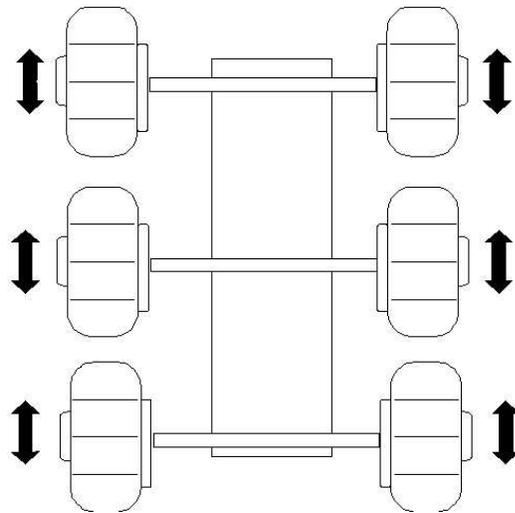


Figure 4: Skid steering [15]

2.2.1 Advantages of Skid Steering

The advantages of skid-steer mechanisms are:

1. Skid-steered vehicles can maneuver effectively and exhibit agile maneuverability;
2. Skid-steered vehicles have greater traction on the surface that they travel, hence they are best used when traveling on rough terrain.

2.2.2 Disadvantages of Skid Steering

Disadvantages of skid-steered mechanisms are as follows:

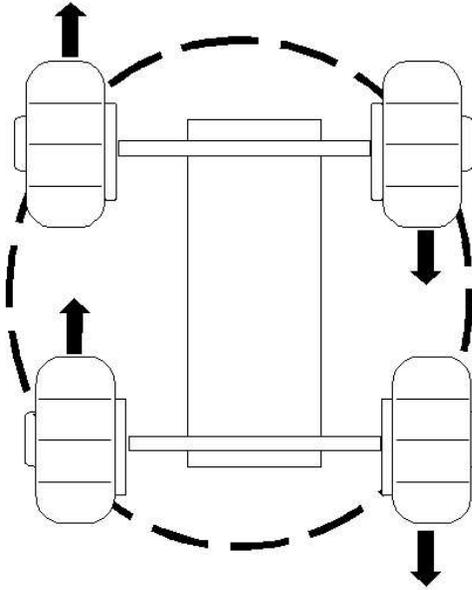


Figure 5: Turns in skid steering [15]

1. Performance of a skid-steered vehicle strongly depends on the surface of the terrain and speed of the vehicle;
2. Skid steering systems' behavior is complex to model mathematically, because they maneuver by skidding on the ground where the amount of the skid varies with the surface of the terrain and the speed of the vehicle;
3. Skid steering imposes considerable stress on parts, like axles, joints, et cetera, that are associated with the steering of the vehicle;
4. Wheels have considerable wear and tear due to skid steering. Skidding also damages the surface of the terrain;
5. Speed and steering of skid steering systems are strongly dependent on each other. Any attempt to maneuver the vehicle affects the speed of the vehicle and vice versa.

2.3 Control Systems

A system responsible for controlling another system is called a control system. There are two structures of control systems, namely, closed-loop control systems and open-loop control systems.

Closed-loop control systems, as shown in Figure 6, have a feedback mechanism, where the output from the control system is taken as input to controller. The controller uses this output from the system to generate control commands for the next cycle. As shown in Figure 7, open-loop control systems do not have feedback from the system's output. It gives the output commands as a function of its input. Open-loop controllers are most commonly used for controlling servo-motors.

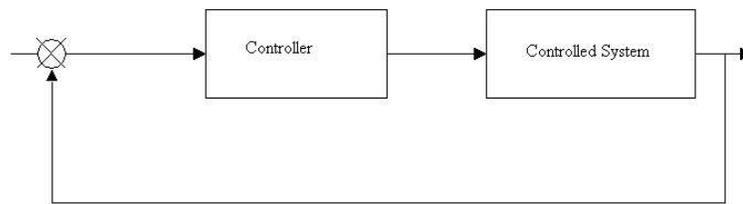


Figure 6: General architecture of a closed-loop control system

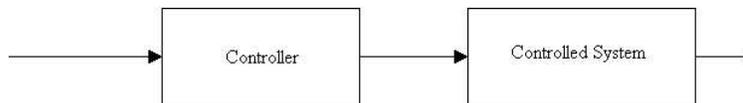


Figure 7: Open-loop control system

There are different techniques for implementing a control system [19]:

- **PID:** Proportional Integral Differential, or PID controllers, are the most common controllers used in industrial applications. They are best suited for controlling a complex system with a single control variable.

- **Neural Networks:** A neural network can be trained to control a system. The entities used for training the neural network are “input data” and “control commands,” where control commands are used to control the system behavior for a given input. Neural network controllers have the ability to learn from the controlling experience. They are best suited for environments where multiple variables have to be tuned to control a system system [1, 2, 10].
- **Expert System:** An expert system can be used to control a system. Control systems designed from expert systems have the capability to learn from the experience. Expert systems are efficient in decision making and do not have complex data processing. Expert systems are suitable for simple decision making.
- **Dynamic Matrix Control:** Dynamic Matrix Control is a fixed process model control technique designed for controlling multiple variables. These controllers involve significant data processing.

Other control techniques include fuzzy logic, statistical process control, statistical quality control and batch control.

We elaborate more on the PID technique since it forms the basis of our steering controller.

2.3.1 PID Controller

PID controllers are the most commonly used controllers for industrial applications. Two common applications of a PID controller are the thermostat and cruise control. A PID controller is a closed-loop controller whose output control variable (d_t) is based on the error (e) between the user-defined set point and the measured process variable.

The general form of a PID controller is:

$$d = K_p e + K_i \int e dt + K_d de/dt \quad (1)$$

The three terms of a PID controller give it its name, as described below:

- **Proportional Term:** The term $K_p e$ has value proportional to the error e and is referred to as the proportional term.
- **Integral Term:** The value of the term $K_i \int e dt$ is an integral of the error e . It sums up small errors over time and produces control commands that will reduce the final error.
- **Differential Term:** The value of the term $K_D de/dt$ is a function of the differential of the error e . It counteracts the K_p and K_i terms when the output changes quickly. It helps reduce overshoot.

The variables K_p , K_i , and K_d are PID control parameters. The values of these variables may be tuned to get the desired response for a given error (e) in the system.

There are two approaches for using a PID controller [14]. They are:

1. **Absolute Approach:** In the absolute approach, the controller generates an absolute value for command d_t in every iteration of the control loop as shown by the equation below.

$$d = K_p e + K_i \int e dt + K_d de/dt \quad (2)$$

An absolute PID controller needs greater care for preventing spikes in its output as a result of spikes in input which usually is due to errors.

2. **Incremental Approach:** In the incremental approach a change in the value of the

command d_t , referred as δd , is generated by the controller.

$$\delta d = K_p e + K_i \int e dt + K_d de/dt \quad (3)$$

An incremental PID controller is easier to tune for spikes in the input.

Some drawbacks of a PID controller are:

1. **No method to calculate direct values for the PID parameter:** The behavior of a PID controller depends on its control parameters which needs to be tuned through an iterative process, and it is not safe to tune these parameters on a critical system like CajunBot, as performance of PID is not guaranteed during the tuning process.
2. **No learning:** The PID control parameters have to be tuned to a specific environment. The control system does not learn. There has been some work on designing self-tuning controllers [3]. While these systems have demonstrated success, the self-tuning can take significant time.
3. **Specific to environment:** The performance of a PID controller is dependent on the environment for which it is designed. A change in the environment demands tuning the PID control parameters for new environments. For example, a PID steering controller tuned for a cemented road may not efficiently control the vehicle on grassy surfaces. One can use a look up table with a set of PID variables for different environments at the cost of increased complexity and overhead of determining PID control parameters for each possible environment.
4. **Does not handle input error:** A PID controller assumes that the input data are accurate, it does not handle input errors (fluctuating or spiking input data).

2.4 Smith Predictor

The time lag between issuing the control commands to the system and getting the feedback about the implementation of these commands is called *dead time*. Dead time is the result of accumulation of lags introduced by various elements in the control loop. Some of the contributors to dead time are measurement lags, computation lags, communication lags, and implementation lags.

Dead time is a common phenomenon found in any control system [5]. The presence of time-delay in the control loops has two major consequences. It greatly complicates the analysis and design of the feedback controller for such systems. It makes satisfactory control more difficult to achieve.

To overcome these drawbacks, Smith [16] presented a control scheme which has the potential of improving the control of loops with a time-delay. This scheme is known as the Smith Predictor or Smith dead-time compensator. Figures 6 and 8 show the general architecture of a control system with and without a Smith Predictor.

A Smith Predictor, shown in Figure 8, sits in the feedback part of the control loop. It takes the input values from the sensors and predicts what those values would be after the dead time. The prediction requires a mathematical model representing the behavior of the system. The controller then uses the predicted values to compute the control output, thereby compensating for the dead time (refer [5] and [12]). A Smith Predictor may be used in conjunction with any control algorithms [11, 13, 20, 17, 9] (refer to Figures 6-8).

2.5 Issues in Steering Control

The difficulties in developing a steering control system for skid-steered vehicles may be summarized as follows:

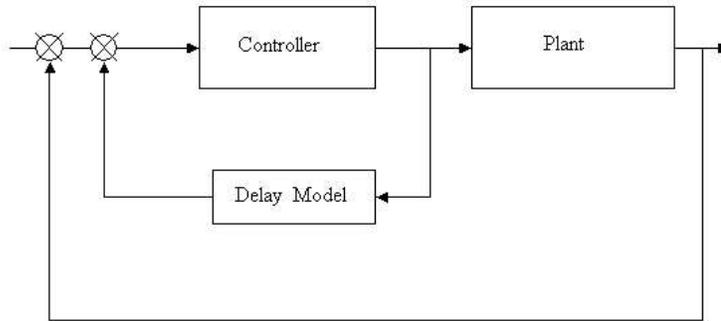


Figure 8: General usage of Smith Predictors in control systems

1. Industrial control mechanisms like PID control cannot be easily used for controlling skid-steered vehicles as the behavior of the vehicle changes with the terrain on which it is traveling and with the variation in speed of the vehicle. This demands having different PID control parameters for different environments;
2. Application of both left and right brakes simultaneously during an attempt of a turn can cause the vehicle to slow down unintentionally;
3. Speed of the vehicle and steering reactions are inter-related, which makes the steering controller harder to control the vehicle at different speeds.

In addition to the above issues, the following issues are introduced due to the design of CajunBot:

1. **Slow responding hardware:** CajunBot's steering system consists of a left brake and a right brake, each controlled using slow moving hydraulic actuators. Thus the control system has a high dead time, making it more difficult to steer without oscillations or overshoot.
2. **No feedback from the brake system:** CajunBot's steering system lacks feedback about the position of the brakes. The steering controller needs to know the extent to

which brakes are applied at any given instance in order to decide whether it should release the brake or pull it to maneuver the vehicle in the desired direction. In the absence of feedback from the brake system, the steering controller needs to estimate the extent of braking that is applied. This estimation is not error free, and degrades the performance of the steering controller.

3 Steering Controller

This section introduces the proposed steering controller and explains the different components of the steering controller.

3.1 Steering Controller

A control system designed to steer an AGV is called a steering controller. Figure 9 presents a level zero data flow diagram of a steering controller. This is the black box view of the controller. It takes as inputs the position and orientation of the vehicle along with the start and end positions of the expected straight line path. When the vehicle is not on the path or is not oriented along the path, the steering control algorithm determines the commands needed to bring the vehicle back on the path. The steering controller consists of the following three modules:

1. Vehicle Control Strategy;
2. Steering Algorithm;
3. Skid Steering Commands.



Figure 9: Level 0 data flow diagram of steering controller

There may be different strategies to steer a vehicle when it deviates from the expected path. One strategy is to bring the vehicle back on the path as soon as possible.

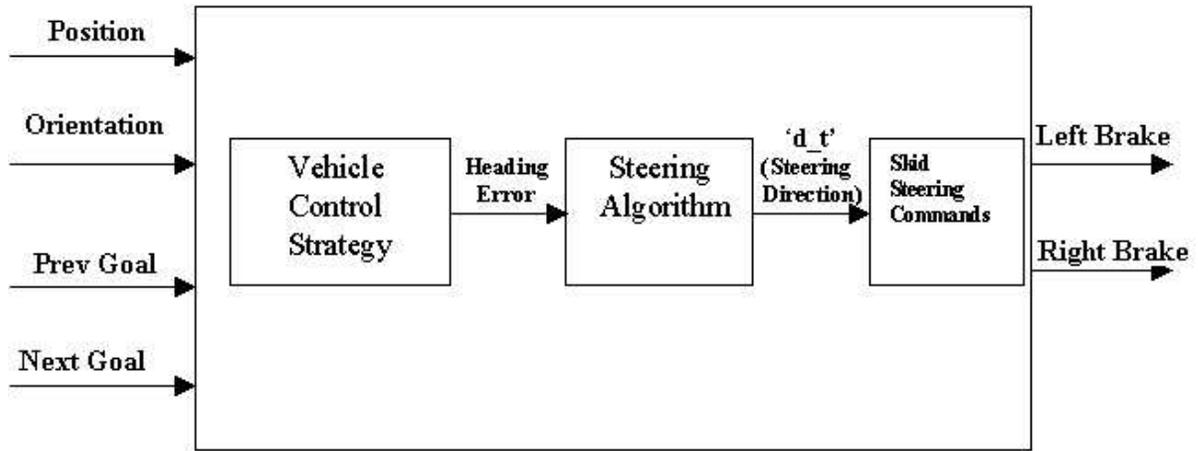


Figure 10: Level 1 data flow diagram of steering controller

Another strategy is to steer the vehicle directly to the end position of the path. The vehicle control strategy module determines the strategy for correcting the deviation from the expected path. It computes the correction in terms of heading error which is the angle the vehicle should turn to correct its deviation.

The steering algorithm takes the heading error and computes steering commands to reduce this error. Besides heading error, the steering algorithm must also consider the safety of the vehicle and dead time in the feedback system. The steering algorithm's output is independent of the steering mechanism. The skid steering command module translates the turn commands generated by the steering algorithm into left and right brake commands for a skid steering.

Generic Steering Command: The Steering Algorithm generates its generic steering commands in terms of a variable called “steering direction” (d_t), which takes values between -1 to +1. If the value of d_t is negative, the steering controller commands the vehicle to turn left by applying left brakes. Similarly for positive values, it commands the vehicle to turn right by applying right brakes. For the value of zero, no brakes are applied. When no brakes are applied, the vehicle makes no turns. The absolute value of d_t represents the magnitude

of the amount of left or right braking to be applied.

In the following section we discuss the above mentioned three modules of the steering controller. These are the vehicle control strategy, the steering algorithm, and the skid steering commands.

3.2 Vehicle Control Strategies

For the given goal position that the vehicle needs to travel, the vehicle control strategy plans the path to follow in order to reach the goal position. Two vehicle control strategies were implemented for CajunBot. These are "way-point" mode and "follow-the-carrot" mode. There are other vehicle control strategies like "pure-pursuit" and "screw theory" control on which the proposed algorithm was not tested.

1. **Way-point Mode:** This approach is based on the concept of driving the vehicle directly to the goal position. As shown in Figure 11, the angle between the present orientation of the vehicle and the line joining the vehicle position to the goal position is calculated as heading error.
2. **Follow-the-Carrot Mode:** This approach is based on the concept of driving the vehicle on a straight line path to a point (x-track point) which is some *look-ahead-distance* away from the vehicle's projected position on the line joining the previous and next goal positions.

The vehicle's position is projected onto the line joining the previous and next goal positions (see Figure 12). A point at a *look-ahead-distance* away from this projected point on the line joining the previous and the next goal position, toward the next goal position is defined as a carrot point. The heading error of the vehicle is now defined as

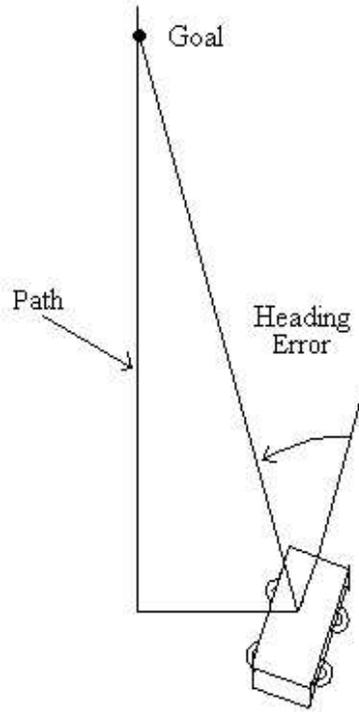


Figure 11: Illustration of the “Way-point” mode steering control strategy

the angle between the vehicle’s current orientation and the orientation of the line joining the vehicle to the carrot point.

3.3 Steering Algorithm

The steering algorithm is responsible for driving the vehicle along the path suggested by the vehicle control strategy. It achieves this by attempting to minimize the instantaneous heading error as computed by the vehicle control strategy module. As shown in Figure 13 the steering algorithm has three components, they are:

1. PID controller;
2. Safety filter;
3. Prediction filter.

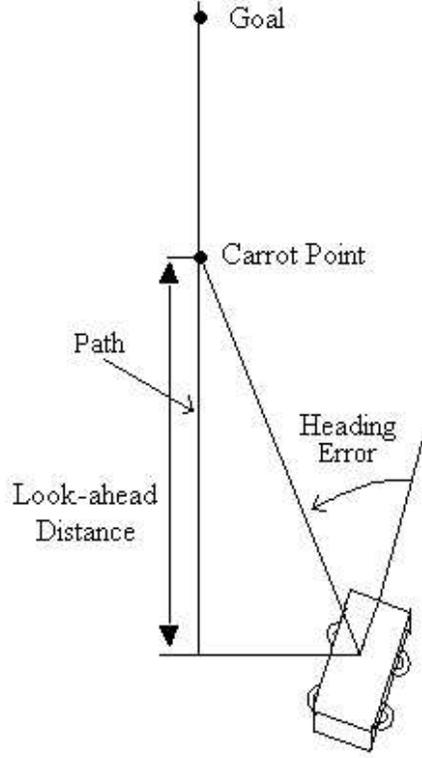


Figure 12: Illustration of the “Follow-the-carrot” mode steering control strategy

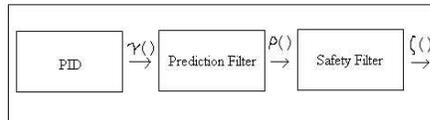


Figure 13: Level 2 components of steering algorithm.

The PID controller determines the steering commands needed to minimize the heading error of the vehicle. These control commands are bounded by the safety filter that keep the rate of change of steering commands within acceptable safety limits of the vehicle. This filter plays an important role while controlling an AGV that is traveling at high speeds.

The prediction filter addresses the issue resulting from dead time. This filter predicts the time (t_{turn}) that the vehicle would take to complete its turn and issues commands to release the steering in just enough time to account for the dead time. The prediction filter

expects that by the time the steering control implements the commands to stop turning, the vehicle would reach the desired orientation.

3.3.1 PID Controller

The PID controller uses the incremental form of PID equation. The incremental approach is very compatible with use of the safety filter. Safety is compromised when the incremental change in steering is greater than what a vehicle can safely tolerate.

The PID equation is given below.

$$\eta(he_t) = K_p he_t + K_i \int he_t dt + K_d de/dt \quad (4)$$

The equation computes δd_t , the change in steering direction at time t , as a function of heading error he_t . The terms K_p , K_d and K_i are PID control parameters. The derivation of this PID equation can be found in Appendix 1.

3.3.2 Safety Filter

When the vehicle is traveling at high speeds, it is necessary to ensure that the vehicle's orientation is changed at a safe rate. We achieve this by limiting the rate at which the term "steering direction" (d_t) is incremented or decremented by α , the maximum safe rate to turn. Where α depends on the speed of the vehicle. The function ζ , below, bounds the $\eta(he_t)$ computed by the PID equation.

$$\zeta(x) = \begin{cases} +\alpha & \text{If } x > \alpha \\ -\alpha & \text{If } x < -\alpha \\ x & \text{otherwise} \end{cases}$$

3.3.3 Prediction Filter

The steering control system of any AGV will have significant dead time typically associated with moving mechanical components for changing the direction of the vehicle.

If the steering control system waits until the heading error has become zero (i.e., the vehicle is in the desired orientation), the vehicle would overshoot leading to heading error in the opposite direction. The overshooting phenomenon will repeat at every correction, causing the vehicle to oscillate over the desired path. The prediction filter solves this problem by issuing steering commands to stop turning before the desired orientation is reached. It predicts the amount of time (T_{turn}) that the vehicle would take to reach the desired orientation and also predicts the amount of time ($T_{release}$) that steering control hardware (left or right brake) will take to release its brakes to stop turning the vehicle. If T_{turn} is smaller than or equal to $T_{release}$ it commands the steering control hardware to stop turning, that is, release appropriate left or right brake, as shown in the following equation.

$$\rho(x) = \begin{cases} +\alpha & \text{If } (T_{turn} < T_{release}) \ \&\& \ (d_t < 0) \\ -\alpha & \text{If } (T_{turn} < T_{release}) \ \&\& \ (d_t > 0) \\ x & \text{Otherwise} \end{cases}$$

where T_{turn} and $T_{release}$ are given by following equations:

$$T_{turn} = \frac{\textit{Heading error}}{\textit{Angular velocity}} \quad (5)$$

$$T_{release} = \frac{|d_t|}{\alpha} \quad (6)$$

The filter produces a positive correction ($+\alpha$) when the steering direction is negative ($d_t < 0$) and a negative correction when the steering direction is positive. The correction is

opposite of the steering direction because the filter is attempting to stop the turn.

3.3.4 Final Equation

The following equation specifies the steering incremental Δd_t . It combines the factors of safety and prediction with PID control.

$$\Delta d_t = \rho(\zeta(\eta(h e_t))) \quad (7)$$

$$\rho(\zeta(\eta(h e_t))) = \begin{cases} +\alpha & \text{If } (T_{turn} < T_{release}) \ \&\& \ (d_t < 0) \\ -\alpha & \text{If } (T_{turn} < T_{release}) \ \&\& \ (d_t > 0) \\ +\alpha & \text{If } \eta(h e_t) > \alpha \\ -\alpha & \text{If } \eta(h e_t) < -\alpha \\ \eta(h e_t) & \text{Otherwise} \end{cases}$$

3.4 Skid Steering Commands

The skid-steered vehicle accepts steering control commands that specify the amount that the left and right brakes are to be applied. These commands are generated by the steering algorithm module. They are expressed in terms of “steering direction” (d_t). These are then converted to low level steering commands. Computation of these control commands is specified in the following equations.

$$Leftbrake = \begin{cases} |d_t| & |d_t| \leq 0 \\ 0 & otherwise \end{cases}$$

$$rightbrake = \begin{cases} d_t & |d_t| > 0 \\ 0 & otherwise \end{cases}$$

4 Insight into the Steering Controller

This section provides a discussion about the parameters for tuning the proposed steering controller. It gives insight into which filter takes control during the course of steering. Finally, it compares the proposed steering controller with a pure PID controller.

4.1 Tuning the Algorithm

The behavior of the steering algorithm depends on the values of the parameters K_p , K_d , K_i and α . The first three parameters are control parameters of the PID equation. The last parameter, α , is a parameter of the safety and prediction filter.

There are mathematical models available that describe the effect of K_p , K_d , and K_i . The values of these variables affect the time it takes the system to converge to a steady state, the amount of overshoot leading to errors in the opposite directions, and the amount of oscillation.

These mathematical models give insight into the effect of these parameters but the models are for an idealized system. The model cannot directly be used to “derive” the ideal values of K_p , K_d , and K_i . As is common with control systems, the values of these parameters is determined empirically, by observing how a system behaves with different combinations of values. Tuning the values of K_p , K_d , and K_i becomes more difficult because of their interdependent effects. Their interdependence makes it difficult to associate these parameters to their effect on the steering.

When we reason about the behavior and performance of the steering, we use the following properties:

Aggressiveness(β): How aggressive is the vehicle in attempting to correct the heading error?

Time to correct(γ): What is the maximum acceptable time in which the vehicle may correct its heading error?

The PID parameters are then defined as:

$$K_p = \frac{\beta}{\gamma}$$

$$K_d = \beta$$

$$K_i = 0$$

Thus our steering controller now can be rewritten to depend on three parameters α , β , and γ all of which are defined in terms of the desired effect on the steering. These parameters are still interdependent; their influence on the vehicle is easier to predict.

4.2 Influence of the Filters

The steering commands generated by our controller may be attributed to one of the three functions:

1. the PID controller (η);
2. the safety filter (ζ);
3. the prediction filter (ρ).

It is instructive to know the source of the steering command at various stages as an AGV changes direction. Figure 14 plots the heading error over time as the vehicle travels along a simple route specified below.

Go straight 5 meters.

Turn right by 45 degrees.

Go straight 30 meters.

Turn left by 45 degrees.

Superimposed on the plot of heading error are annotations representing the function that generated the command. The sudden change in heading error at time 73 seconds happens when the vehicle traveled five meters and should make a 45 degree turn to the right. The plot shows that at that instant, the AGV determined it should turn 45 degrees. The steering commands used by the vehicle were generated by the safety filter. This indicates that the commands generated by the PID control may have been unsafe for the vehicle.

The vehicle continues to use commands from the safety filter until the heading error reaches 0.4. At this point, the source of the steering command changes to the prediction filter. This implies that the steering controller has determined it would be prudent to stop turning based on the vehicle's present speed and orientation. This effect may be seen in the change of slope of the heading error.

The vehicle overshoots and goes too far to the right. Thus it has a heading error in the opposite direction. The safety filter again overrides the PID control commands.

The vehicle oscillates between steering left and right. While this happens, the source of the commands changes from the safety filter to the prediction filter. As the oscillations dampen, the commands from the PID control function begins to take effect. This increases the rate of dampening, eventually bringing the heading error to zero.

4.3 Comparison with a Pure PID Controller

PID controllers have been used in industrial control for a long time. They have been designed to address concerns, such as overshoot, oscillations, and time to convergence. It would stand to reason that one can build a steering algorithm by using just a PID controller, without the steering and prediction filters.

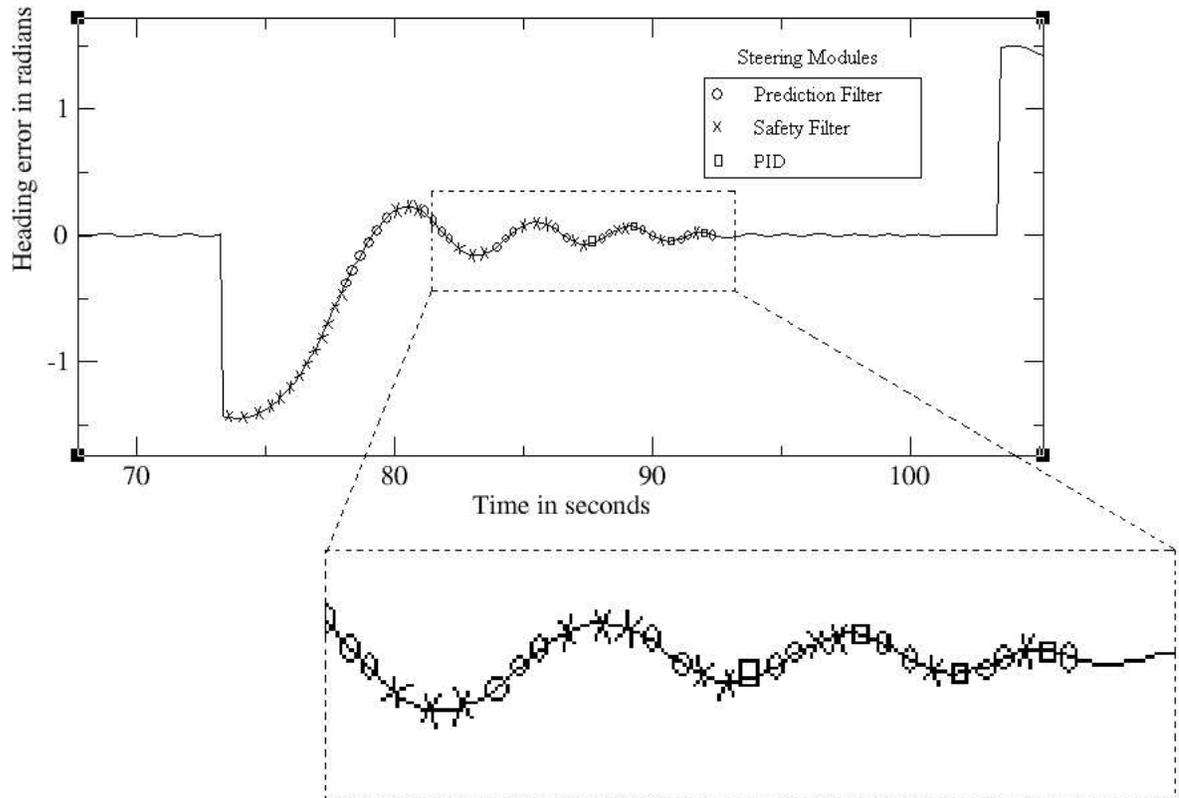


Figure 14: Illustration of the source of the steering commands

In this section we discuss the augmentation of a PID controller with safety and prediction filters.

The advantages of the safety filter are discussed first. Once an autonomous vehicle reaches a way-point, it must turn towards the next way-point. The amount of turn is indicated by heading error. This is the difference between the vehicle's heading and the orientation to the next goal. When the turn is sharp, the heading error will be high and a PID controller may attempt to compensate for this heading error by generating steering commands to make a sharp turn. If the command to turn is greater than what the vehicle can maneuver, vehicle safety may be jeopardized.

The goal is to prevent the PID controller from producing commands for sharp turns for high heading errors. This can be achieved by decreasing the values of K_p . However, such a decrease will also yield a proportional decrease in the effective steering commands for low heading errors. Even though the integral term K_i takes control for small errors, the term really gains weight after the integral sum of the errors reaches a substantial value. Thus could lead to delay in the vehicle reaching a stable state, thus endangering the vehicle when traveling at a high speed through a narrow corridor.

We now discuss the advantages of the prediction filter. The prediction filter accounts for the dead time in the steering control system. When it determines that the vehicle will reach the desired orientation with the commands already issued, it overrides the PID controller and initiates commands to stop turning.

A PID controller achieves similar functionality by using an appropriate K_p such that the command to turn reduces as the heading error reduces. However, the time it will take for the vehicle to reach a desired orientation depends on the vehicle speed.

A PID based steering controller will depend entirely on heading error, because this is what it is attempting to reduce. The PID controller may be tuned to have the same effect as

the prediction filter for a certain speed. However, when the speed is increased or decreased, the steering control will be suboptimal. This can be overcome by maintaining different PID parameters for different speeds. This has the disadvantage of increasing the complexity of the algorithm and the process of tuning the controller.

5 Empirical Evaluation

This section presents the results of empirical evaluation of the proposed steering controller. The controller was evaluated in a simulated environment.

5.1 Experimental Design

The purpose of the experiment is to study the effect of the safety and prediction filters to validate whether they achieve the objectives of safe navigation of the vehicle without oscillations at varying speeds.

5.1.1 Procedures

The experiment was performed by using the steering controller to steer a computer simulation of CajunBot. The simulated vehicle was built using ODE (Open Dynamic Engine). ODE is a physics engine used for building computer games. It is here used to simulate some of the vehicle physics of CajunBot in a virtual environment. Like the real vehicle, the simulated vehicle has two sets of wheels, where each set can be controlled independently. The simulated brake system is used to reduce the speed of wheels. When both wheels are free, a force can be applied to the vehicle in forward direction to move the vehicle forward. When one set of wheels is braked and a force is applied, it turns along the direction of the braked wheels. Thus, the virtual robot simulates a skid-steered vehicle.

5.1.2 Independent Variables

The steering algorithm has two filters. These are safety and prediction. We wish to study the effect of each filter independently. This gives us two variables “safety and prediction,” each with two values “on” and “off.” Since we wish to study the steering

controller at varying speeds, the vehicle's speed is an independent variable. While the speed can be any numeric value, we restrict its values to 1m/s, 2m/s, 3m/s 4m/s.

Our interest is in studying how the steering controller turns a vehicle. The turning of a vehicle may be described in degrees from 0 to 360, along left and right directions. We restrict the turn angles to 45° and 90° , for both left and right directions. Figure 15 shows a track with all the four turns.

The route specifies that the vehicle should travel five meters, turn right by 45 degrees, travel 30 meters, turn left by 45 degrees, travel 30 meters, turn right by 90 degrees, travel 30 meters, turn left by 90 degrees, and finally travel 30 meters.

In addition, the PID controller has three parameters K_p , K_d , and K_i . In our controller $K_i = 0$, leaving two variables K_p and K_d , each with a numeric value. The choice of the values of K_p and K_d affect the controller, so it is not appropriate to fix their values to a small set. Instead we find the combination of values of the PID parameters that gives the best results for a specific speed (1m/s) on the track of Figure 15 for each combination of the safety and prediction variables. This gives us the K_p and K_d values to use for each combination of safety and prediction at higher speeds.

5.1.3 Dependent Variables

The steering controller attempts to reduce the heading error. Thus, the heading error is an independent variable.

Our goal is to ensure vehicle safety, reduce overshoots, and reduce oscillations. Vehicle safety can be assessed by rate of change of heading. Rate of change above a certain threshold will jeopardize vehicle safety. Overshooting a turn can be measured by the change of heading error in the opposite direction of the turn. Oscillations can be measured by the number of times a vehicle's heading error changes signs before becoming zero (or close to

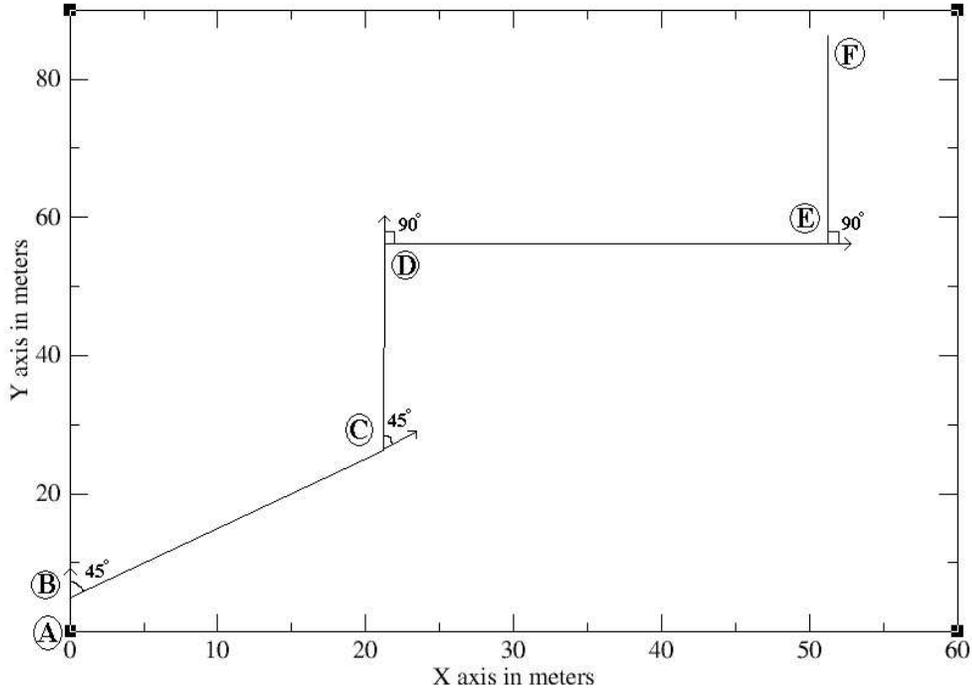


Figure 15: Suggested path for the experimental runs shown in Figure 16, 17, 18, 19

zero).

Thus, the heading error of the vehicle over time is sufficient to evaluate the performance of the algorithm for safety, overshoot and oscillations.

5.2 Data Analysis

The simulated CajunBot was run on the route shown in Figure 15 for each combination of the following values of the independent variables:

Speed: 1m/s, 2m/s, 3m/s, 4m/s;

Safety: on, off;

Prediction: on, off.

The values of K_p and K_d for which the vehicle performed best at a speed of 1m/s for a particular combination of value of safety and prediction were then used at higher speeds for

Independent Variables			Dependent Variables	
Figure Number	Safety Filter	Prediction Filter	Overshoot	Oscillations
Figure 16	off	off	yes	moderate
Figure 17	off	on	no	no
Figure 18	on	off	yes	high
Figure 19	on	on	yes	no

Table 1: Description of the graphs

the combination of safety and prediction.

Figures 16, 17, 18, 19 plot the heading error (y-axis) over time (x-axis) for the four values of safety and prediction at a speed of 4m/s. As the performance of the vehicle decreases with increasing speed, the plots for 4m/s are instructive of the change in performance at higher speeds.

In each of the Figures 16 to 19, the labels A, B, C, D, E, and F represent the position of the vehicle as indicated on the route drawn in Figure 15. The values of safety and speed used for these plots are summarized in Table 1.

Table 1 summarizes the performance of the controller for each combination of safety and prediction. The table shows that the vehicle oscillate when the prediction filter is turned off and does not oscillates when the filter is turned on. This is as expected, since the PID controller does not account for the dead-time.

The vehicle also overshoots when the prediction filter is turned off. This, again, as expected. When the prediction is turned on and safety is turned off, the vehicle does not overshoot either. The vehicle overshoots when the safety filter is turned on along with the prediction filter. At first glance this appears anomalous. When the prediction filter is turned on, one would expect it to compensate for dead time and reduce overshoot. The result, however, can be explained by the fact that the safety filter overrides the prediction filter. It indicates that when the safety filter is off, the vehicle does not overshoot, but at the expense of jeopardizing vehicle safety. When the safety filter is turned on, it restricts the rate of turn

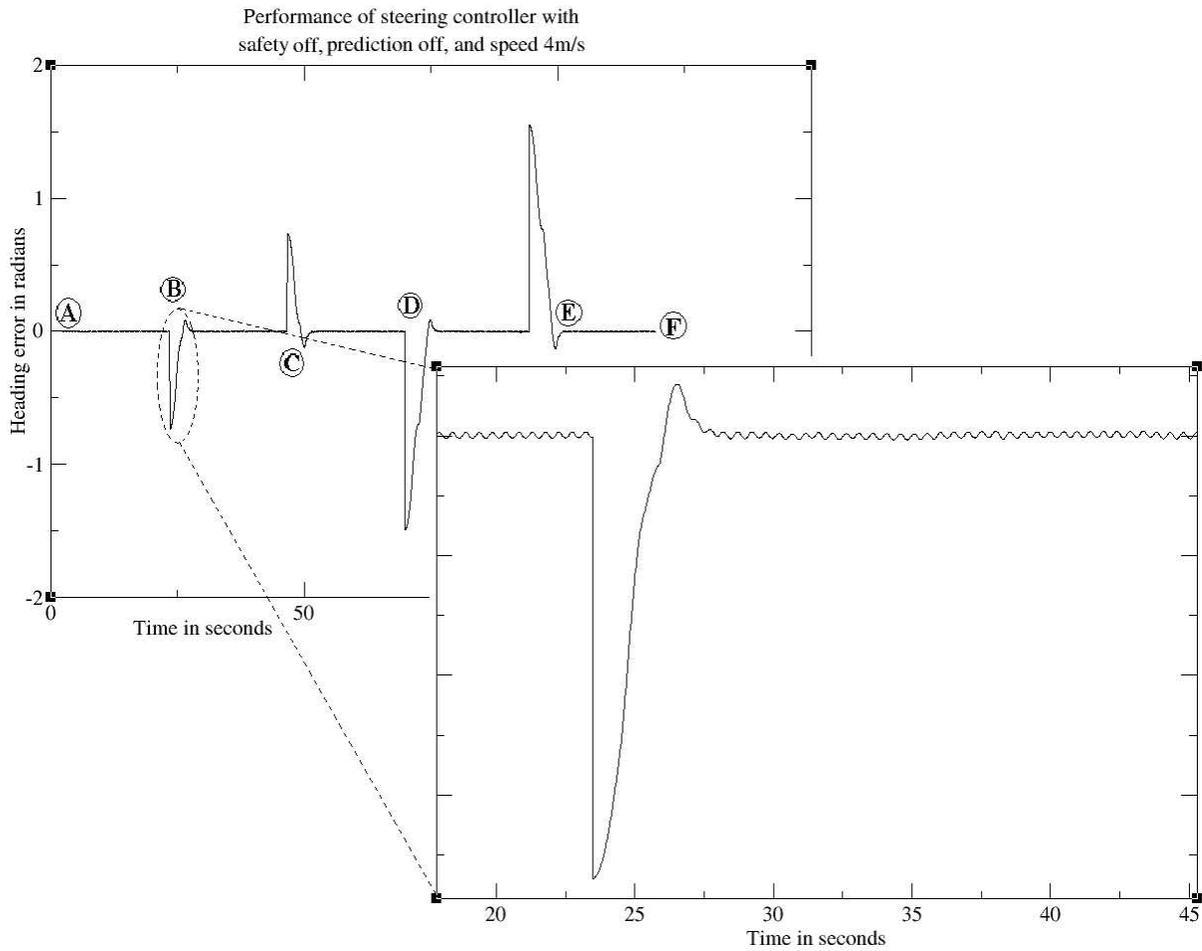


Figure 16: Heading error for safety off and prediction off

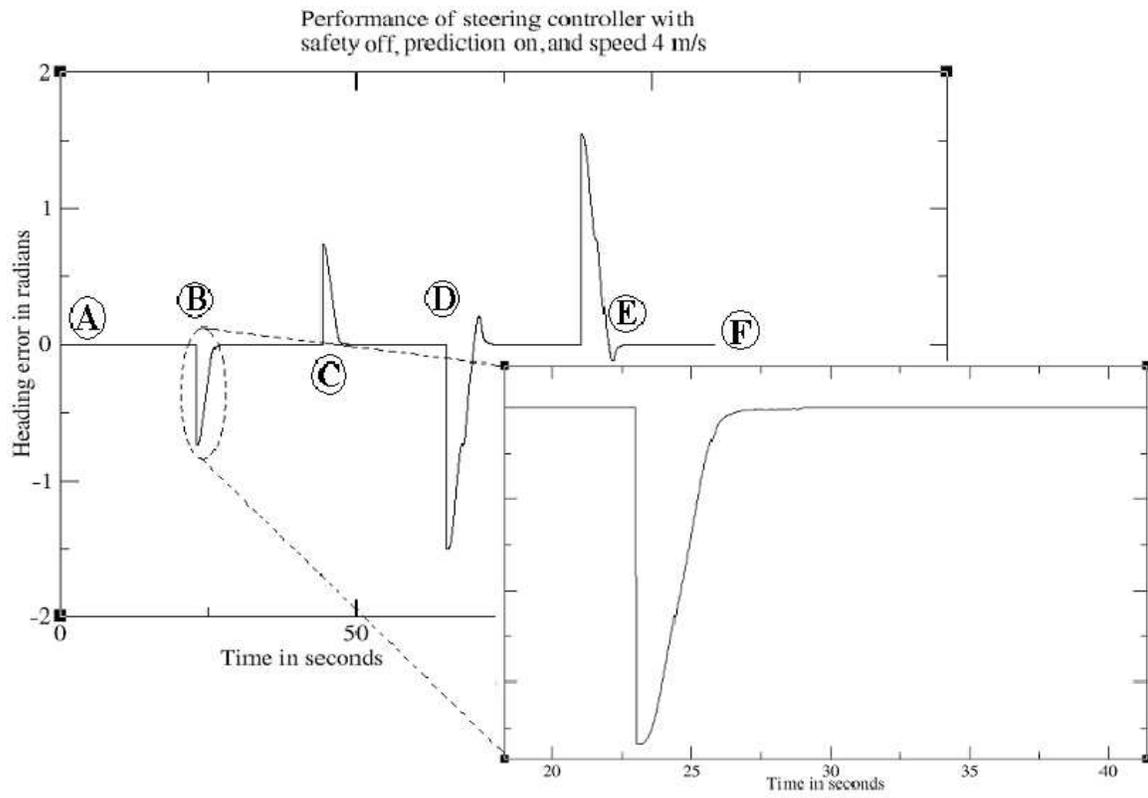


Figure 17: Heading error for safety off and prediction on

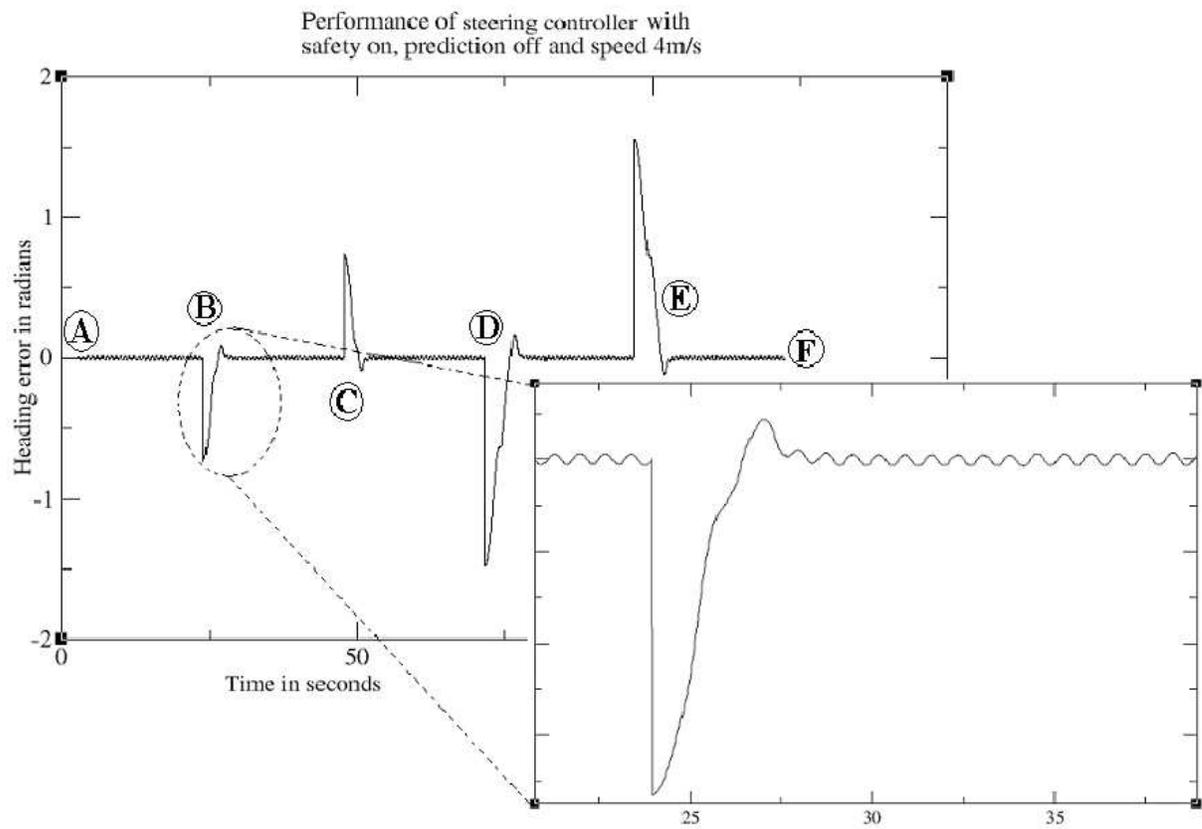


Figure 18: Heading error for safety on and prediction off

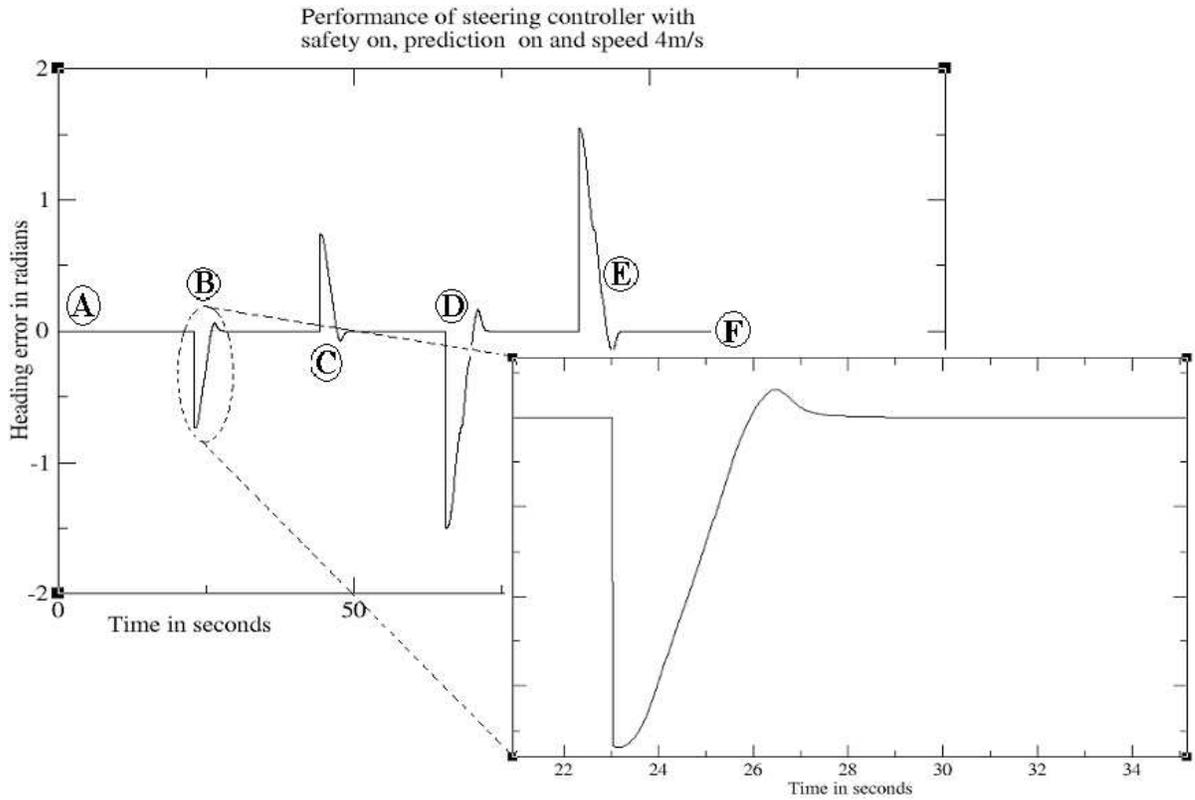


Figure 19: Heading error for safety on and prediction on

to safe limits, causing the vehicle to overshoot.

The combination of turning on safety and prediction thus gives the best results. It does overshoot, but does not have oscillations. This implies the vehicle overshoots only to ensure safety of the vehicle, and then it quickly converges to the correct orientation.

6 Conclusions and Future Work

This thesis developed a steering controller for a skid steered vehicle. The controller navigated the vehicle at varying speeds while ensuring its safety, minimizing oscillations and minimizing overshoots. The controller was developed by augmenting the classic PID controller commonly found in industrial control systems. The classic PID controller was augmented by introducing two filters. These were the safety filter and prediction filter. These filters override the commands generated by the PID controller. The prediction filter compensates for dead time in the control system. It predicts when the vehicle is expected to complete a turn and overrides the commands of the PID controller and tells the steering control to stop turning. The safety filter overrides both the PID controller and the prediction filter when it deems that the rate of turn is detrimental to the vehicle's safety.

The proposed steering algorithm was evaluated using a virtual simulated skid-steering vehicle implemented using Open Dynamics Engine. The effect of the individual filters was studied by driving the simulated vehicle with four separate experimental controllers that were obtained by separately turning each filter on and off. The vehicle was driven at four speeds, from 1m/s to 4m/s. A speed of 1m/s was used to tune the PID control variables.

The results show that the prediction filter reduces vehicle oscillations. When the filter is turned on, the vehicle does not oscillate while correcting a heading error, whether or not the safety filter is turned on. When the prediction filter is turned off, the vehicle oscillates from left to right and back attempting to orient in the desired direction.

With the prediction filter turned on the vehicle does not overshoot if the safety filter is turned off. If the safety filter is turned on the vehicle overshoots the turn, but does not oscillate. The overshoot of a turn to ensure safety of the vehicle proceeds as desired.

In our current work, the PID control parameters and the safe rate of turn has been

treated as constant. An alternative would be to make these parameters a function of speed. Tuning the parameters as a function of speed may be worthy of further investigation.

Appendix

1. Derivation of PID controller for steering

The first steering controller I implemented was based on the absolute PID control equation. I found it difficult to tune the PID control parameters, so I reverted to implementing a steering controller based on my intuitive understanding of steering. My steering controller, which I termed “progressive controller” worked very well and I had higher confidence in tuning it.

When I began writing this thesis I had to compare the progressive controller to the PID controller. To do the comparison I started tweaking the equations of the progressive controller to identify terms of the equation that matched the PID equation, and the terms that were different. To my surprise I found that I had reinvented the PID equation. The simplified equation of the progressive controller had a proportional and differential term, just like the P and D terms of the PID equation.

The experience was discouraging for the obvious reason that I had simply reinvented a work straight out of the control system textbooks. It was also enlightening in that now I had a better “gut-level” feeling of PID control equations. More importantly, now I also had a definition of the PID parameters in terms of parameters that were more intuitive in the context of steering.

In this appendix, I describe the derivation of the progressive controller and its mapping to the PID controller. This mapping may be instructive to others working on similar problems.

Table 2: Basic functions

Function		Description
Location	P	Location represented with three tuple x, y, z
Azimuth	$z(P_1, P_2)$	Angle formed by line joining P_1, P_2 w.r.t reference axis

2. Preliminaries

Steering control takes as input, position (P_t^b), heading (h_t), and speed of the vehicle (s_t), along with the previous way-point (P_t^i) and the next way-point (P_t^{i+1}) (as shown in Figure 9). It computes steering controls commands to drive the vehicle to the next way-point.

Steering Direction: The concept of using a steering direction (d_t) representation is one of the important features of this control system. It is a common representation that can be converted to low level steering commands for different steering mechanisms such as skid steering versus explicit steering.

Steering direction (d_t) (refer Table 4) takes values in interval $[-1, +1]$. Negative values represent steering control commands to turn left. Positive value represent steering controls commands to turn right. A value of zero represents the steering controls commands to make no turns.

Location (P_t) represents the position of the vehicle or position of the way-point using three tuple (P_x, P_y, P_z) (refer Table 2). This steering control algorithm has three controlling parameters α , β , and γ (refer Table 3). Parameter α represents the maximum rate at which steering controls can be altered assuring the stability of the vehicle by considering the issues like speed of the vehicle. Parameter β represents how aggressive the steering controller should be in correcting a heading error. Parameter γ represents the time in seconds within which the heading error (he_t) be should be corrected. β can be a function of γ .

Table 3: Vehicle parameters

α	Maximum safe rate to turn
β	Aggressiveness
γ	Time to correct heading

Table 4: Instantaneous measures at time t

Heading Angle	h_t
Heading Error	he_t
Position of previous goal	P_t^i
Position of vehicle	P_t^b
Position of next goal	P_t^{i+1}
Steering direction	d_t

Azimuth: Function azimuth $z(P_1, P_2)$ (refer Table 2) calculates the angle of orientation of the line joining points P_1 and P_2 with respect to the reference axis.

Heading convention: The algorithm is not bound to a particular heading convention but it would be easier to explain the algorithm using an assumed heading convention. The heading angle is zero to the east and increases in the counter clockwise direction.

3. Progressive Steering Controller

From the position of the vehicle (P_t^b) (refer to Table 4) and the position of the next way-point (P_t^{i+1}), calculate the desired heading using Equation 8.

$$\text{desired heading}(t) = z(P_t^b, P_t^{i+1}) \quad (8)$$

The difference between the desired heading and the present heading of the vehicle (h_t) gives the heading error as shown in Equation 9. We set the desired heading rate (refer to

Equation 10) to the heading error divided by time(γ) within which we expect to correct the heading error.

$$\text{heading error}(t) = z(P_t^b, P_t^{i+1}) - h_t \quad (9)$$

$$\text{desired heading rate}(t) = \frac{z(P_t^b, P_t^{i+1}) - h_t}{\gamma} \quad (10)$$

The present heading rate of the vehicle is the angular velocity at which the vehicle is turning and is calculated using the Equation 11.

$$\text{present heading rate}(t) = \frac{\Delta h_t}{\Delta t} \quad (11)$$

The difference in the desired and the present heading rates is given by Equation 12.

$$\text{difference heading rate}(t) = \frac{z(P_t^b, P_t^{i+1}) - h_t}{\gamma} - \frac{\Delta h_t}{\Delta t} \quad (12)$$

The steering direction d_t is changed by Δd . This is calculated using the Equation 13.

$$\delta d_t = \left(\frac{z(P_t^b, P_t^{i+1}) - h_t}{\gamma} - \frac{\Delta h_t}{\Delta t} \right) * \beta \quad (13)$$

4. Conversion of the Progressive Algorithm to PID Form

Let us denote the desired heading, $z(P_t^b, P_t^{i+1})$, by x_t , and rewrite Equation 13 as

$$\delta d_t = \left(\frac{x_t - h_t}{\gamma} - \frac{\Delta h_t}{\Delta t} \right) * \beta \quad (14)$$

Where $\Delta h_t = h_t - h_{t-1}$. Adding and subtracting $x_t - x_{t-1}$ to Δh_t , we have

$$\Delta h_t = (x_t - x_{t-1}) - ((x_t - h_t) - (x_{t-1} - h_{t-1})) \quad (15)$$

$$\Delta h_t = \Delta x_t - (he_t - he_{t-1}) \quad (16)$$

$$\Delta h_t = \Delta x_t - \Delta he_t \quad (17)$$

Using the definition of Δh_t from Equation 17 in Equation 14 we get

$$\delta d_t = \left(\frac{he_t}{\gamma} - \frac{\Delta x_t - \Delta he_t}{\Delta t} \right) * \beta \quad (18)$$

Re-arranging the terms yields the equation below.

$$\delta d_t = \frac{\beta}{\gamma} * he_t + \beta * \frac{\Delta he_t}{\Delta t} - \beta * \frac{\Delta x_t}{\Delta t} \quad (19)$$

The general PID control equation is given below.

$$\delta d_t = K_P * he_t + K_D * \frac{\Delta he_t}{\Delta t} + K_i * \int he_t \quad (20)$$

Let us compare Equation 18 with Equation 20. Equation 19, and 20 each have an proportional (he_t) and differential ($\frac{\Delta he_t}{\Delta t}$) terms, but Equation 19 has one additional term

Table 5: Comparison of PID parameters and steering controller parameters

$$\begin{aligned} K_P &= \frac{\beta}{\gamma} \\ K_D &= \beta \\ K_i &= 0 \end{aligned}$$

$\beta * \frac{\Delta x_t}{\Delta t}$ and the Equation 20 has the term $K_i * \int he_t$. Table 5 gives mapping that relates the terms of the two equations, leaving only the $\beta * \frac{\Delta x_t}{\Delta t}$ of Equation 19 unmapped. The term Δx_t represents the change in desired heading that increases its value when the vehicle is going off the road. On plotting the value of this term during the course of a run, it is seen that this term has almost zero value all through the run except when there is a change in goal position, when it takes a spiking large value (as desired heading changes with change in goal position). Hence we ignore this term.

References

- [1] Shumeet Baluja. Evolution of an artificial neural network based autonomous land vehicle controller. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 26(3):450–463, June 1996.
- [2] Parag Batavia, Dean Pomerleau, and Chuck Thorpe. Applying advanced learning algorithms to ALVINN. Technical Report CMU-RI-TR-96-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1996.
- [3] Peter J. Gawthrop. Self tuning PID control structures. CSC Technical Report CSC-96011, University of Glasgow, October 1996.
- [4] T. Hague, B. Southall, and N. D. Tillett. Autonomous crop treatment robot: Part II. real time implementation. *The International Journal of Robotics Research*, 21(1):75–85, January 2002.
- [5] Kang Dong Hunn. *The Robust Smith Predictor Design using Uncertainty Quantification and Application to an Industrial Manipulator*. Master’s thesis, Pusan National University, 1999.
- [6] S. Laubach. *Theory and Experiments in Autonomous Sensor-Based Motion Planning with Applications for Flight Planetary Microrovers*. Ph.D. thesis, California Institute of Technology, June 1999.
- [7] S. Laubach and J. Burdick. An autonomous path-planner for planetary microrovers. In *Sixth International Symposium on Experimental Robotics (ISER’99)*, Sydney, Australia, March 1999.

- [8] S. Laubach and J. Burdick. An autonomous sensor based path-planner for planetary microrovers. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '99)*, Detroit, May 1999.
- [9] G. Meinsma and H. Zwart. On \mathcal{H}_∞ control for dead-time systems. *IEEE Transactions on Automatic Control*, 45(2):272–285, 2000.
- [10] Nils J. Nilsson. *Artificial Intelligence*, chapter 6, pages 85–111. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1998.
- [11] J. E. Normey-Rico and E. F. Camacho. A unified approach to design dead-time compensators for stable and integrative processes with dead time. *IEEE Transactions on Automatic Control*, 47:299–305, 2002.
- [12] A. Nunez, J. Normey-Rico, C. Bordons, and E. F. Camacho. A Smith predictive based MPC in a solar air conditioning plant. *Process Control*, 15-1:1–10, April 2004.
- [13] Z. J. Palmor and M. Blau. An auto-tuner for Smith dead time compensator. *International Journal of Control*, 60:117–135, 1994.
- [14] Thana Pattaradej, Pakob Jundang, Guanrong Chen, and Pitikhate Sooraksa. A speed evaluation for conventional PID and fuzzy controllers. In *Proceedings of Intelligent Technologies*, pages 91–96, Thailand, November 2001.
- [15] Benjamin Shamah. *Experimental Comparision of skid steering vs Explicit steering for a wheeled robot* . Master’s thesis, Carnegie Mellon University, March 1999.
- [16] O. Smith. A controller to overcome dead time. *ISA Journal*, 6(2):28–33, 1959.
- [17] Pauline Sourdille and Aidan O’Dwyer. A new modified Smith predictor design. In *Proceedings of the 1st International Symposium on Information and Communication*

Technologies, ACM International Conference Proceeding Series, pages 385–390. Trinity College Dublin, 2003.

- [18] B. Southall, T. Hague, J. A. Marchant, and B. F. Buxton. Autonomous crop treatment robot: Part I A Kalman filter model for localization and crop/weed classification. *The International Journal of Robotics Research*, 21(1):61–74, January 2002.
- [19] Mark Thomson. Process control concepts in relation to online analysers. In *ICEX97 Proceedings of Instrumentation and Control Symposium*, Sydney, Australia, June 1997.
- [20] Vance J. Vandoren. The Smith Predictor: A process engineer’s crystal ball, May 1996. <http://www.manufacturing.net/ctl/article/CA188333?pubdate=5%2F1%2F1996> (last accessed November 19, 2004).
- [21] Richard Vaughan, Neil Sumpter, Andy Frost, and Stephen Cameron. Robot control of animal flocks. In *Proceedings of International Symposium on Intelligent Control, USA*, 1998.
- [22] Richard Vaughan, Neil Sumpter, Andy Frost, and Stephen Cameron. Robot sheepdog project achieves automatic flock control. In *Proceedings of the International Conference on Simulation of Adaptive Behaviour, Switzerland*, pages 489–493, 1998.
- [23] Richard Vaughan, Neil Sumpter, Andy Frost, and Stephen Cameron. Experiments in automatic flock control. *Journal of Robotics and Autonomous Systems*, 31:109–117, April 2002.

Golconda, Suresh, Bachelor of Engineering, Osmania University, Hyderabad, India, May 2002; Master of Science, University of Louisiana at Lafayette, Spring 2005
Major: Computer Science
Title of Thesis: Steering Control for a Skid Steering Autonomous Ground Vehicle at Varying Speeds
Thesis Directors: Dr. Arun Lakhotia and Dr. Anthony Maida
Pages in Thesis: 52; Words in Abstract: 305

Abstract

There are three major software components in an Autonomous Ground Vehicle (AGV). These are the obstacle detector, the path planner, and the steering controller. The obstacle detector uses sensor data to identify unnavigable areas on the terrain. The path planner computes a path around the obstacles. The steering controller generates commands for the vehicle to follow the planned path. Of the three components, the steering controller is the most crucial. Without a proper steering controller the vehicle cannot even travel a completely pre-planned path that is devoid of any obstacles.

This thesis presents the steering controller of CajunBot, a six-wheeled all terrain vehicle (ATV) that uses skid-steering for turning. The steering controller was developed with the objectives of navigating a skid-steered vehicle at varying speed with minimal overshoots, minimal oscillations, and ensuring the vehicle's safety.

The proposed steering controller consists of a PID controller with two filters, a prediction filter and a safety filter. These filters are placed in succession. The prediction filter compensates for dead-time in the control system. It overrides the commands generated by the PID controller when it deems that the commands already given are sufficient to complete the turn. The safety filter ensures that the rate of turn of the vehicle does not cross a given threshold. This filter overrides the output of the PID controller and the prediction filter.

Empirical observations in a simulated model of CajunBot show that the inclusion of the prediction and safety filter does help in achieving the objectives of the steering controller. Removing any one or both of these filters has a negative effect on the performance of the vehicle. By separating the concerns of safety and prediction in separate filters, the PID control system is used more effectively by concentrating on only a small range of heading errors and a high range of speeds.

Biographical Sketch

Suresh Golconda was born in Hyderabad, India on September 04, 1980. He graduated with a bachelor's degree in Computer Science in June 2002 from Osmania University, Hyderabad, India. He entered the master's program in Computer Science at the University of Louisiana at Lafayette in Fall 2002. Upon completion of this thesis, Suresh Golconda is set to join the Team CajunBot as well as pursue a doctoral degree in computer science.