Chapter 1

INTRODUCTION

1.1 Motivation

Mobile robots are becoming increasingly significant in industrial, commercial and scientific applications. The number of robots currently being used in industrial projects is around 70,000 and increasing. The rate at which science and industry are developing has opened the door for the use of robots in many fields.

Many researchers have investigated control and communication methods for mobile robot systems. Problems such as coordination of multiple manipulators, motion planning and coordination of mobile robot systems are generally approached with a central (hierarchical) controller in mind. There is extensive research being carried out on autonomous mobile robots. Many solutions to the problems, including path planning and obstacle avoidance, have been proposed and tested. However, most of the research on autonomous mobile robots was based on a single robot interacting with its environment. There is an increasing interest in multiple autonomous mobile robot systems due to their applicability to various tasks such as space missions, operations in hazardous environments, and military operations. Such systems present the problems of both multiple robot coordination and autonomous navigation. Again, multiple mobile robots may be controlled by using a hierarchical (central) controller. However, the tasks mentioned above obviously require many robots that are able to navigate autonomously. It is difficult to use a central controller or a hierarchical method, sometimes because of the large distances, sometimes due to robustness and versatility problems.

The problem of autonomous navigation of mobile vehicles, or Automated Guided Vehicles (AGV), involves certain complexities that are not usually encountered in other robotic research areas. For instance, the dynamic nature of the world, in indoor or outdoor environments, requires a real-time control system. Additionally, in order to achieve autonomous navigation, the real-time decision making must be based on continuous sensor information rather than off-line planning. Finally, since the vehicle continuously explores different regions, it faces a wide variety of possible world configurations, which requires a “general” and adaptable control structure.

1.2 Context

The purpose of this report is to describe the design, development and exploratory research with a modular autonomous mobile robot (Bearcat I) built for the Association for Unmanned Vehicle Systems (AUVS) 1997 competition. The mobile robot test-bed has been constructed using a golf cart base and modified to have full-speed control with guidance provided by a vision system and an obstacle avoidance system using ultrasonic sensors systems. The system design has several unique and innovative approaches. A fuzzy logic controller is used for obstacle avoidance and an innovative three-dimensional algorithm is implemented for line tracking. The system components are independently configured and are modular in design, making them easily adaptable for other vehicles. The speed and steering control systems are supervised by a personal computer through a multi-axis motion controller. The obstacle avoidance system is based on a micro-controller interfaced with several ultrasonic transducers. This micro-controller independently handles all timing and distance calculations and transmits a distance that can be used to modify the steering angle correction based on a fuzzy logic controller. Vision guidance is accomplished using two CCD cameras with zoom lenses. The vision data is processed by a high-speed tracking device, letting the computer know the X,Y coordinates of blobs along the lane markers. The system also has three emergency stop switches and a remote-controlled emergency stop switch, which can disable the traction motor and set the brake. The testing of these systems has been done in the lab as well as on an outside test track with positive results that show that at five mph the vehicle can follow a line and, at the same time, avoid obstacles. A reliability analysis has been done to anticipate faults in the system. The performance of this work must be judged at the contest. However, simply participating in the competition is a challenging, exciting and unforgettable educational experience.

The design of a mobile system is a challenging task. The specific challenge of designing an intelligent controller is in determining what information is needed, how to measure it and how to use this information in a manner that will satisfy the performance specifications of the machine. The design specifications were to build a robot that follows a line, avoids obstacles, and adapts to variations in terrain. This implied the design of separate subsystems with discrete design objectives integrated in an upper-level control logic that enables the robot to function as an integral system, meeting all the performance requirements.

At the subsystem level, the primary design considerations included the selection of equipment with the desired functional and operational features as well as reliability, commercial availability and affordability. Equally important was the compatibility of the software that controlled these units and their interfaces. Also, all the subsystem components have been chosen to be modular in design and independent in terms of configuration to increase adaptability and flexibility. This is, in fact, a unique feature of the design, since it enables replacement of existing components with more sophisticated or suitable components, as they become available. To ensure the optimal performance of the individual sub-systems, several unique approaches were tried. These include the implementation of a fuzzy logic controller for obstacle avoidance and a novel three-dimensional algorithm that out-performs the existing methods for line following. In the design and development phase of the different systems, various analytical, experimental and computational methods were utilized. These have been described in detail in chapter 5 and chapter 6.

At the upper level, the complexity of the system necessitated an automated tool to ensure system reliability. Accordingly, a computer-aided fault diagnostic system was developed to rapidly analyze and remedy system failures. Also important were the safety considerations for the robot in case of runaway situations. To this end, safety features were built into the operation of individual subsystems, in addition to three stopping devices for the entire robot. Finally, system integration has been achieved through the use of compatible interfaces and software.

1.2.1 Problem statement

To meet the specific challenge described earlier, one of the major modules was the design development of the steering and speed control systems. The flexible navigation of the automated guided vehicle in the real environment of the competition required a robust and efficient response steering mechanism, which guided the robot path. The ability of the robot to navigate successfully, which involved the detection of the 10-foot track and avoiding the obstacles on the course, depended directly on how the steering mechanism responded to the various signals that the other sub-systems fed into it. The driving signals from the vision control system, the obstacle avoidance system, the remote by-pass system, and the fuzzy logic controller were the determining factors for the steering system module. The accurate response and the actual track that the robot followed thus eventually depended on the steering control system understanding that the other modules were perfect and fed in the right signals. Each of the systems needed to be tested for accuracy of information. Then we could derive conclusions on whether or not the steering control system designed was working as per the design specifications. The speed control system also needed to be integrated successfully with the steering module to ensure the right speed of the vehicle on the track. The competition required that no vehicle exceed a speed limit of 5 mph for safety reasons. Hence, the upper limit of the design speed was initially fixed, and various factors like terrain slopes, friction, speed of reactance of the various modules, and the time taken to steer the vehicle once a signal was received were taken into consideration.

1.3 Specific goals of the research

The need for a robust and successful steering mechanism was described in the previous section. The specific aim of the proposed research can be stated as follows:

1. To develop a prototype model of the design of a steering mechanism for the automated guided vehicle for successful navigation and to meet the specific challenges of the automated unmanned vehicle systems contest in 1997.
2. To build the steering mechanism, develop a control system, and implement it by proper integration with the other system modules.
3. To develop the mathematical model and simulate the steering system response of the control system. Specifically, to develop a PID controller for the steering control system.

4. To test the steering control module in a real environment and close the feedback loop in fine-tuning the system to meet the ultimate objective of successful navigation.

1.4 Contribution to the research

The proposed model is a novel control system design for an integrated fuzzy logic controlled three-wheeled automated guided vehicle. The design development, system mathematical model and simulation can be extended to other control systems designs, but the specific nature of control, involving the multiple data input and the efficient management of the data, is still an open issue in the ongoing research of mobile robots. Considerable importance has been given to data fusion and the capability of the steering mechanism to respond in the most efficient manner to the various system modules on board BEARCAT 1.

1.5 Contribution of this work

This research work is organized into six chapters. Chapter 1 presented a discussion on the problem statement and research goals, as well as the need for the proposed research and the specific purpose for which it was aimed. Chapter 2 discusses steering control systems and existing literature in this field of work. Chapter 3 presents an introduction to control system theory, different order systems, linear and non-linear control, feed-back control systems and an example of a typical design process. Chapter 4 describes the digital control theory, PID control and the Galil DMC-1000 controller. Chapter 5 describes the mobile robot (BEARCAT 1), built at the University of Cincinnati and the Automated Unmanned Vehicles Contest. Chapter 6 deals with the kinematics of the steering mechanism and the proposed control model. Chapter 7 shows the simulation of the control system on Matlab, Simulink, a Galil inc. Software package and the test results. Chapter 8 concludes the thesis with some recommendations and areas of improvement that for the future.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

An automated guided vehicle (AGV) can be considered to be an automated mobile conveyor designed to transport materials. Most AGVs require some type of guide path, although recent developments have attempted to eliminate guide paths and achieve truly autonomous control. A state-of-art review [1], [2] showed that the steering stability of the guidance control system has not been analyzed fully. The slight "snaking" of the AGV about the track generally has been acceptable, although it hints at the instability of the steering guidance control system.

Most AGVs have a specification of maximum speed about 1 m/sec (3.28 ft/sec, 2.24 mph) although, in practice, they are usually operated at half that speed. In a fully automated manufacturing environment, there should be few personnel in the area;

therefore, the AGV should be able to run at full speed. As the speed of the AGV increases, so does the difficulty in designing stable and smooth tracking controls. Consequently, it becomes important to understand the dynamic effects of various design parameters on the steering action of the AGV.

Work has been done in the past to analyze various aspects of steering control in different situations for automobile steering or mobile robots. The kinematics and dynamics involved are dealt with by some; and the controller designs for a linear or non-linear control have been dealt with by others.

Ozguner, Et al[3] described an analytical study of vehicle steering control. They considered the design and stability analysis of a steering controller. The objective of the controller was to steer a ground vehicle along a reference line located in the middle of the lane. An arbitrary look-ahead point was located on the local longitudinal axis of the vehicle. The displacement between the look-ahead point and the reference point was called the look-ahead offset. During perfect lane tracking, the ratio of the steering angle to the look-ahead offset was independent of the curve radius under reasonable approximations. That ratio was computed in terms of the vehicle speed and various vehicle parameters. Then, a constant controller was designed to achieve that ratio at steady state. The controller was updated as a function of the vehicle speed. The only information processed by the controller was the look-ahead offset, which was measured using a radar-based or vision-based sensor. Using Routh-Hurwitz analysis, it was analytically proven that the closed loop system is stable. Given any range of longitudinal speeds, there exists a sufficiently large look-ahead distance ensuring the closed loop closed-loop stability for all speeds in that speed range.

The look-ahead information was a set of geometrical information, including road topology ahead of the vehicle center of gravity and the vehicle's orientation and position relative to the topology. In this paper the objective was to develop a new control law for automatic steering based on a specific look-ahead information.

The paper considered the vehicle model and discussed the control design and the stability analysis of the controller. A simulation example was presented.

The mathematical block diagram of the proposed model was a shown in the Figure 2.1 below.

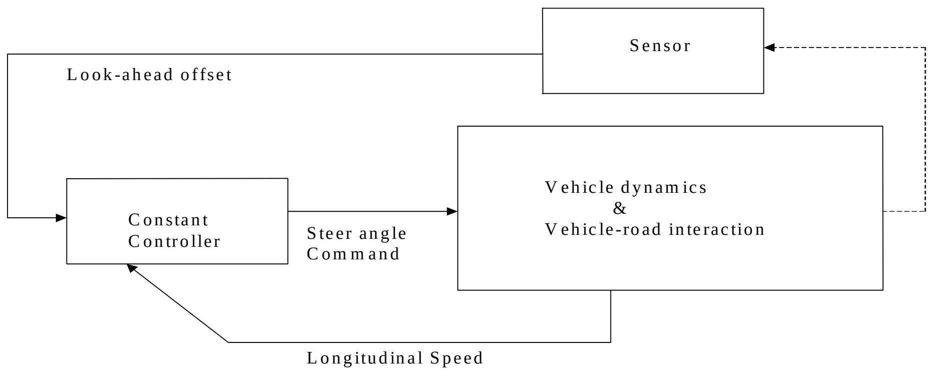


Figure 2.1: Lane following feedback structure (Ozguner, Et al. [3])

The authors conclude that the controller was successfully tested at the Transportation Research Center, Marysville, OH.

2.2 Linear and non linear controller design for robust automatic steering

Ackerman, et al[4] propose a "linear and non-linear controller design for robust steering." The research work presented by them concentrated on the automatic steering of vehicles, which they described as part of an integrated system of integrated highway system of the future. The primary task of an automated steering is to track a reference path, where the displacement from the guidepath is measured by the displacement sensor. The reference may consist of a magnetic field of an electrically supplied wire or permanent magnets in the road. The sensor is mounted in the center of the front end of the vehicle. The controller output acts on the front wheel steering angle.

The design of an automatic steering system is a robustness problem in view of the large variations in velocity and mass of the vehicle and contact between tire and road surface. In the proposed research these authors, model data and specifications for a city bus. A comparison was then made between linear and non-linear controller concepts.

For non-linear control it was investigated that the track accuracy was improved by additional feedback of the yaw rate, which could be measured by a gyro. Therefore the automatic control problem became much less dependent on the uncertain operating conditions of velocity, mass, and road-tire contact. The study showed a significant reduction in the displacement from the guideline for all Maneuvers and operating conditions. In their study, the design method used the parameter space approach, which was further exploited to explore extreme design directions. The resulting controller with fixed gains achieves good performance for a wide range of uncertainty in the operating conditions.

Additionally, in their research a non-linear controller structure was designed in an effort to further improve the performance of the automatic steering system. The non-linear controller was based on sliding mode control and included dynamic adaptation to changing operating conditions via an estimator-like observer. The advantages and drawbacks of the two approaches were contrasted in simulation studies. Finally, controller parameters of both the linear and non-linear controller were tuned automatically by optimizing a vector performance for typical maneuvers.

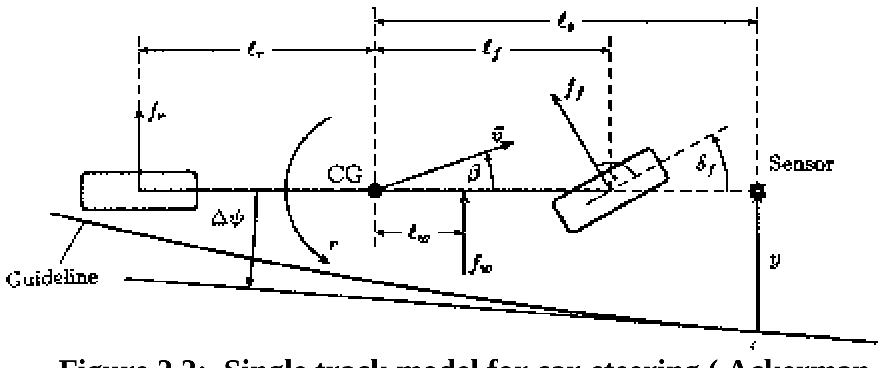


Figure 2.2: Single track model for car-steering (Ackerman, et al [4])

The Figure 2.2 shows the single-track model for the car and Figure 2.3 shows the nonlinear control structure modeled in a block diagram.

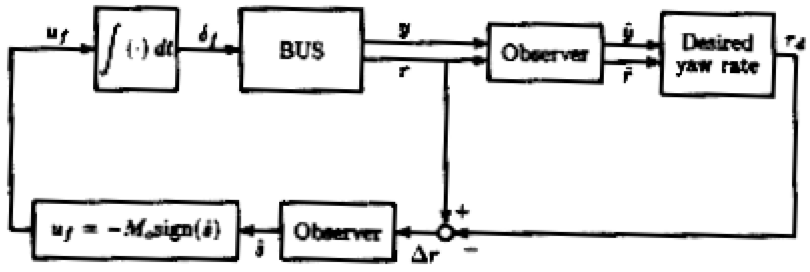


Figure 2.3: Nonlinear control structure model block diagram (Ackerman, et al [4])

2.3 Parallel Linkage Steering for an Automated Guided Vehicle.

Sung, Et al[5] have presented in their work a “parallel linkage steering for an automated guided vehicle. The paper described the computer-aided design of suitable control strategies for parallel linkage steering of an experimental automated guided vehicle. The microprocessor-based steering control system was modeled, and stability characteristics, using proportional control, were investigated via describing functions. Although the approximation is coarse, the analysis indicates the upper bound for gain and suggests derivative action for improved control. For better accuracy, a flexible and fast simulation program was developed that enabled an efficient search for “optimal” control laws for operation of the vehicle under the constant speed mode and the slow-down mode. Time-independent control laws were investigated, and the solutions are found gave tolerable tracking over 90-deg arcs of 0.5-m radius. The simulation results were shown in which the time dependent control laws were proved to be essential for improved, nonoscillatoy tracking.

The experimental AGV has a differential gear drive and parallel linkage steering as shown in Figure 2.4.

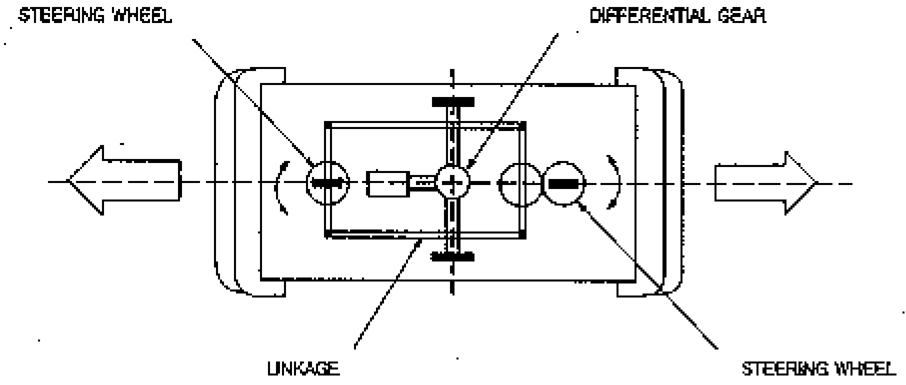


Figure 2.4: Design of experimental automated guided vehicle (Sung, Et al [5])

Six infrared sensors [light emitting diodes (LEDs)] were used on the forward and reverse guidance systems. Because of the designed configuration of the wheel positions, the AGV was symmetrical in the forward and reverse directions, which simplified bi-directional control. A DC servomotor was used for the drive, a stepper actuated the steering wheels. The steering loop was closed by a microprocessor-based-controller that computed the steering control signal to the stepper motor based on the derivation of the AGV from the track. The track contained reflective tape, and the derivation was measured by discrete number of LEDs “out of track”. This coarse sensing method was selected over more sophisticated continuous sensors for cost reasons but at the expense of the anticipated poorer tracking response.

The critical parameters affecting the steering action of the AGV were identified as follows:

1. Method of sensing the deviation from the guide path.
2. Geometrical configuration of the AGV's wheels.
3. Speed of the AGV and drive dynamics in case of variable speed operation.
4. Steering angle as a function of the deviation, i.e., the control strategy or control law.

Parameters 1 and 2 above pertained to the AGV firmware, whereas items 3 and 4 are required to be designed on the basis of firmware. It was desirable to operate the AGV at its maximum speed (constant) all the time but as was seen in the research study that was not possible when the AGV was negotiating a curve. Thus, the computer-aided design focused on finding:

1. The maximum speed possible in the constant speed mode of operation and the associated optimal control law.
2. A suitable slow-down mode of operation for the AGV at 1m/sec and the associated optimal control law.

Figure 2.5 below shows the block diagram of the steering control system used.

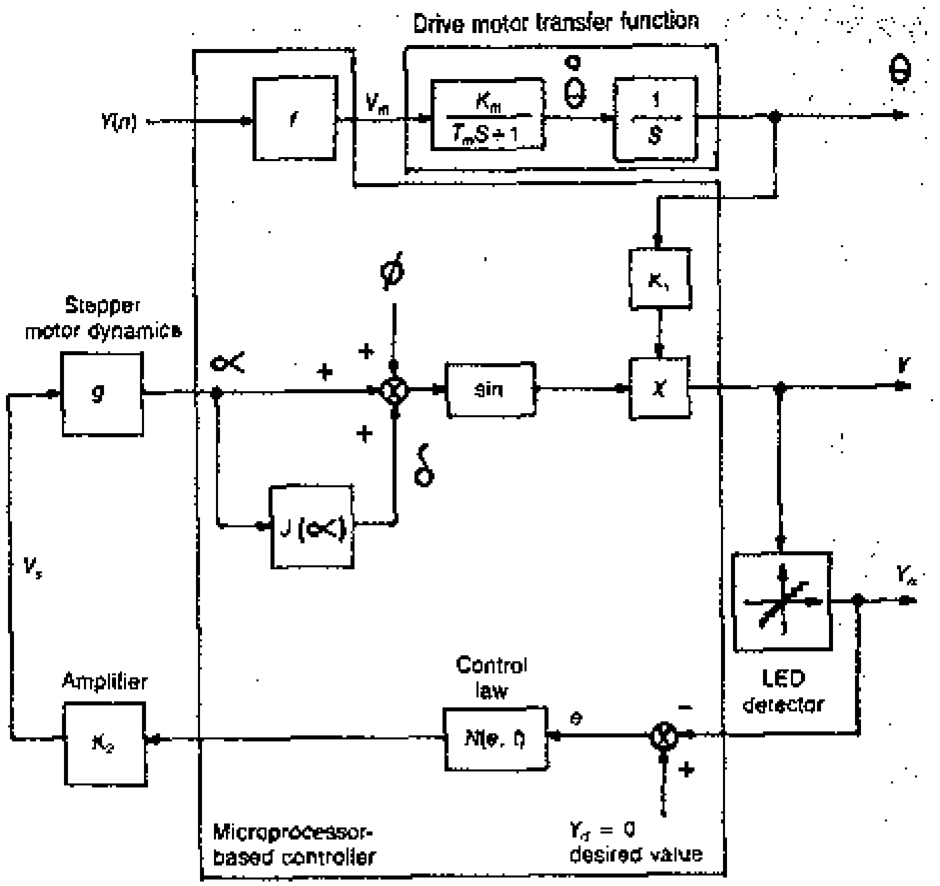


Figure 2.5: Block diagram of steering control system (Sung, Et al [5])

2.4 Modeling and Control of an Automated vehicle

Will and Zak, [6] present a vehicle model that includes the vehicle dynamics and vehicle dynamic model. The model developed was then used for conducting steering analysis of an automated vehicle. They tested the developed model on a step lane change maneuver and proposed a model-reference-based controller for the remote control of a vehicle. Stability analysis of the closed-loop system using Lyapunov approach was included.

One of the main objectives of their research was to develop a vehicle model that could be used to predict the dynamics for steering and braking maneuvers. They proposed a model that is a spring mass system acting under the influence of tire forces developed by three driver inputs. Those inputs were the front and rear wheel angles and the brake force. Another objective of their study was to develop design control strategies for automatically guided vehicles. They proposed a model-reference tracking controller design. Figure 2.6 shows the model-reference control system block diagram.

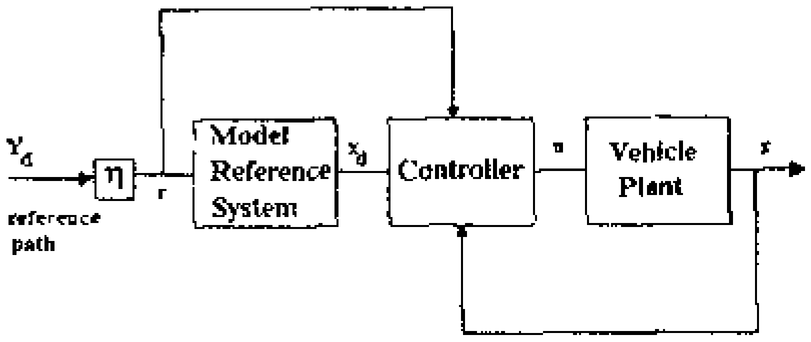


Figure 2.6: Model-reference control system block diagram with input scaling (Will and Zak [6])

Figure 2.7 below shows the responses of the truth model to the same input signal. The truth model was incapable of tracking the desired path when using an open loop control.

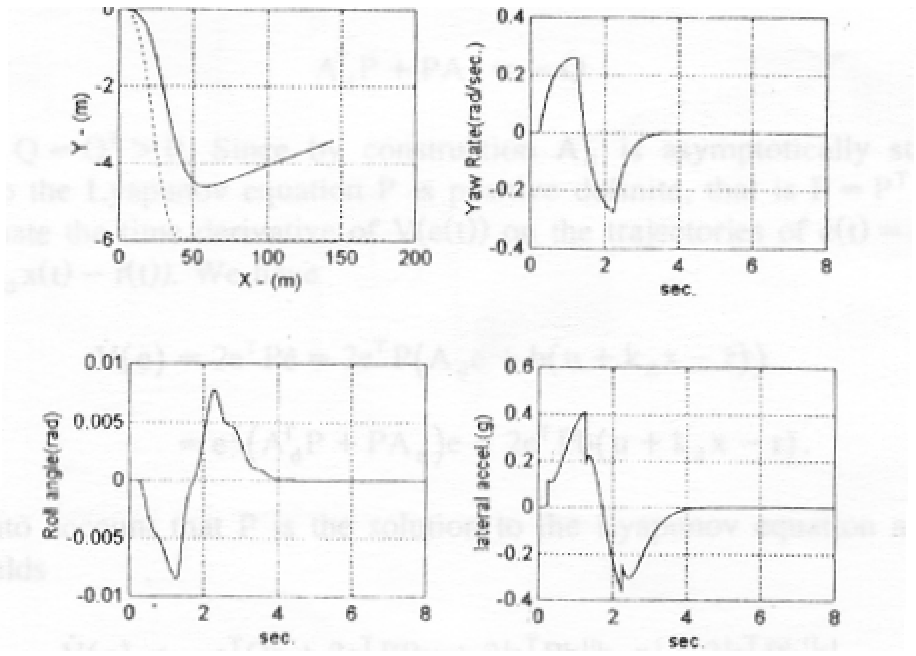


Figure 2.7: Closed-loop system performance with model-reference tracking controller (Will and Zak [6])

To improve the tracking performance, they proposed a closed-loop tracking controller.

They proposed model-reference-based controller required the availability of state variables. They proposed to use combined controller state estimator compensators that require only the input and output signals in their implementations. The model was also discontinuous in the state, which resulted in high activity, referred to as chattering. Chattering could be reduced in a continuous version of the controller proposed. The continuous controller was obtained by introducing a so-called boundary layer that smoothed out the discontinuous controller. However, instead of asymptotic stability, only a uniform ultimate boundedness can be guaranteed.

Chapter 3

AN INTRODUCTION TO CONTROL SYSTEM THEORY

3.1 Introduction

This chapter presents an explanation of basic control theory and the Galil motion controller. Control systems are an integral part of modern society. Control systems find widespread application in the steering of missiles, planes, spacecraft and ships at sea. With control systems we can move large equipment with precision that would otherwise be impossible. We can point large antennas toward the farthest reaches of the universe to pick up faint radio signals. Moving the antennas so precisely by hand would be impossible. The home is not without its own control systems. In a video disc or compact disc machine, microscopic pits representing the information are cut into a disc by a laser during the recording process. During playback, a reflected laser beam, focussed on the pits, changes intensity. The changes in light intensity are then converted into an electrical signal and processes as sound or picture. A control system keeps the laser beam positioned on the pits, which are cut as concentric circles. A home heating system is a simple control system consisting of a thermostat or bimetallic material that expands or contracts with changing temperature. This expansion or contraction moves a vial of mercury that acts as a switch, turning the heater on and off. The amount of expansion or contraction required to move the mercury switch is determined by the temperature setting.

3.2 The basic control system

A control system consists of subsystems and processes (or plants) assembled for the purposes of controlling the output of processes. For example, a furnace is a process that produces heat as a result of the flow of fuel. This process, assembled with subsystems called fuel valves and fuel-valve actuators, regulates the temperature of the room by controlling the heat output from the furnace. Other subsystems, such as thermostats, which act as sensors, measure the room temperature. In the simplest form, a control system provides an output response for a given input stimulus as shown in Figure 3.1.

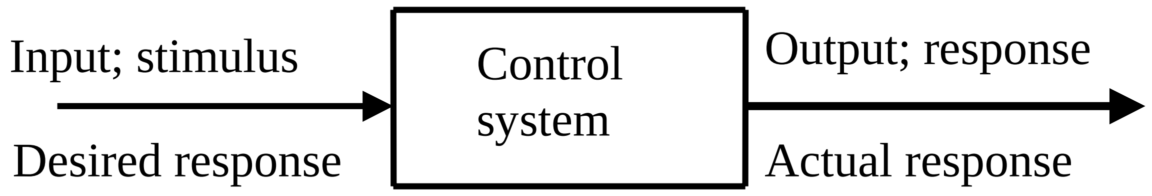


Figure 3.1: Simplified description of a control system

(Nise, Norman [29])

3.2.1 The control problem

A generic closed-loop system consists of :

- input; reference input gives the desired output (usually called a setpoint)
- controller
- plant, a system to be controlled
- measurement device; this allows the current state of the system to be assessed and generation of an appropriate error signal
- output; the controlled output actually generated by the closed loop system
- Reward function; in reinforcement problems, we do not know what the setpoints are; instead we get evaluative feedback about how effective our control law is.

There may also be a

- Plant model; having a sufficiently accurate model helps us how to build an optimal controller.

In general, the plant will be a dynamical system, that is, a system describable by a set of differential (or difference) equations. The distinction between continuous and discrete dynamical systems can be significant; in the reinforcement literature, several important results have been found for the control of discrete systems that do not apply in the continuous case. First of all, the difference between feedback and feedforward control needs to emphasized.

3.2.2 Description of the Input and output

The input represents a desired response; the output is the actual response. For example, when the fourth-floor button of an elevator is pushed on the ground floor, the elevator rises to fourth floor with a speed and floor-leveling accuracy designed for passenger comfort. Figure 3.2 shows the input and output for the elevator system.

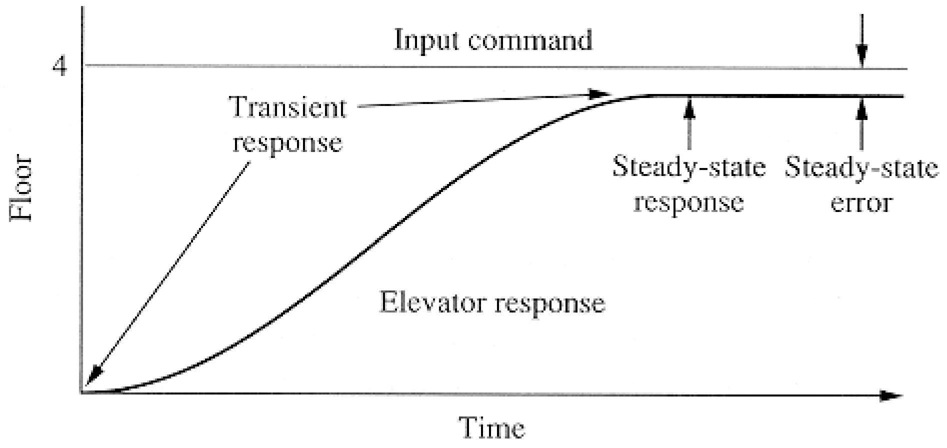


Figure 3.2: Input output for the elevator system (Nise, Norman [29])

The push of the fourth floor button is the input and is represented by a step command. The input represents what the desired output should be after the elevator has stopped; the elevator itself follows the displacement described by the curve marked *elevator response.* Two factors make the output different from the input. The first factor is the instantaneous change of the input against the gradual change of the output. The figure 3.2 shows that the physical entities cannot change their state instantaneously. The state changes through a path that is related to the physical device and the way it acquires or dissipates energy. This is called the *transient response* of the system.

After the transient response, the physical system approaches its steady-state response, which is its approximation to the commanded or desired response. The accuracy of the elevator's final leveling with the floor is a second factor that could make the output different from the input. This difference is called the *steady-state error*.

3.3 Feedback and feedforward control

Feedback Control is an error-driven strategy; corrections are made on the basis of a difference between the system's current state, and the desired state. In the simplest case of a linear feedback control, the corrections are proportion to the magnitude of the difference, or error. This may not work very well for nonlinear plants, so it is common to multiply, these control inputs by a gain matrix. The desired state acts as an attractor, and the system behaves as a simple spring. Springs are notorious for exhibiting oscillations, so it is common to include some damping terms. E.g.

$$r(t) = K_p(x_{d(t)} – x (t)) + K_v (d x(t)/dt); \qquad (3.1)$$

where r(t) is the control input, x (t) the state of the system, $x_{d(t)}$ the target state, K_p the proportional constant or the damping coefficient, K_v the derivative constant or spring constant in this case and x(t)/dt the rate of change of the state of the system.

The damping K_p coefficient can be set to give critical damping (a state in which there is just the right amount of damping to bring the system to the desired out level). Note that this is a completely model-free approach; the complex dynamical interactions of the system are considered as errors to be eliminated by feedback. As it stands, Equation (3.1) is not quite right; constant disturbances like gravity will leave a steady state offset in position. We can never get rid of this (unless we have an infinite value of K_p, so we need an extra integrative term:

$$r(t) = K_p(x_{d(t)} - x(t)) + K_v (d\, x(t)/dt) + K_I \int (x_{d(t)} - x(t))\, dt \qquad (3.2)$$

where is the K_I integral constant. So now there are three terms in the controller, it is seen that feedback loops like this are described as a three-term controller, or a PID controller where PID stands for proportional, integrative, derivative.

Feedback control is, by definition, an error-driven process, so there need to be errors for it to be doing anything. This means that it is very likely that there will be a path that lags continuously behind the desired path. Also, feedback systems usually take a finite amount of time to respond to errors, so perturbing oscillations above a certain frequency will not by corrected. Finally, it should be noted that feedback control might not be limited to the position of the system; and that feedback control can be defined at any level. Thus, it is perfectly possible to process feedback error signals at a task level, as long as we can find a reliable way of decreasing the error.

Feedforward (model-based, indirect) Control takes an alternative approach; a model of the dynamics of the system is built, and the inverse dynamics are solved for input torque. This method has the potential for very accurate control, but depends critically, on the integrity of the model. If we denote the plant dynamics by $x(t) = R(r(t), x(0))$ then

$$r(t) = R^{-1}(x_d(t)) \qquad (3.3)$$

This is the inverse dynamics of the plant; given the desired state of the system, this tells us the control inputs required to achieve that state (assuming such control input actually exists). If, however, we only have a model of the plant, say $R^{\wedge}$, then the path of the system will be described by then,

$$\theta(t) = R(R^{\wedge-1}(x_d(t))) \qquad (3.4)$$

There could be several reasons for feedforward control to be unsatisfactory:

- It may be impossible to get a sufficiently accurate model of the system, and the deviation of the produced path from the target may be a rapidly increasingly function of time (due to non-linearities or non- stationarities in the dynamics of the system).
- Even assuming that our model it perfectly accurate, we may have an error in the measurement of the initial state of the system, i.e. at $t = t_0$.
- Any external objects that cause the system to deviate from its path will not be corrected; simply brushing against a surface may introduce errors that cannot be eliminated.
- It would seem unlikely that the control system is capable of computing inverse dynamics fast enough, even given a perfect model. This may not be a problem for dedicated hardware controlling robot arms, but if were concerns about biological plausibility, then a system using purely feedforward control appears unrealistic.

It can be useful to think of control problems in a dynamical systems context. If we consider the plant and the controller as one system, then the control problem can be construed as placing demands on the dynamics of this combined system. In stable control, we require that the setpoint of the plant be a fixed point of the system. We would also usually require this attractor to be stable: small perturbations from the goal state should cause the system to converge back to the attractor. We may also be concerned with the flow field surrounding the fixed point, particularly if we want the system to collapse onto the goal state as quickly as possible. but without overshooting. The reference model mentioned above is a description of the desired behavior of the combined system. For example, in designing the control system of an aircraft, we don't have any desired states, but we will have some idea about how we wish the combined system to handle.

Most control architectures can be categorized as direct or indirect: indirect control makes use of a model of the plant in designing the controller, direct control tries to optimize the outputs of the plant directly. In the case of indirect control, a further distinction can be made between on-line and off-line optimization. Off-line optimization allows us to learn a plant model by observing the plant's behavior under different conditions (system identification), and subsequently use that model to design a controller. In on-line optimization, the plant has to be controlled all the time, while the plant model is being learned. This is obviously a much more difficult problem, as the combined system is now time-variant.

3.4 Linear control systems

Most physical systems have non-linear elements, but in some circumstances it may be possible to treat them as linear. Then the edifice of linear mathematics, which is very highly developed, can be employed to yield solutions. If a system only operates over a small range of input values, then the non-linearities can often be effectively approximated by linear functions. This may be referred to as operation about some reference point or nominal trajectory; if the non-linear equations are known, then the linearised form of these equations are often called small perturbation equations. If the non-linearities are severe enough to render the linearization approach invalid, then there is no general theory of non-linear control available at present only specific methods; methods for control in such circumstances, based on artificial neural and fuzzy logic can be found in various research works. The methods for control of linear, time-invariant systems are very well known. The only difficulty is that it requires a moderate amount of linear algebra, which can at first be intimidating for the uninitiated and often a large amount of computation that can be intimidating.

3.5 Control of non-linear systems

Non-linear systems are much more difficult to control than linear systems, mainly because the system equations are not necessarily solvable. Remembering that solving a set of differential equations means writing down a closed-form equation describing the behavior of the system under a whole variety of boundary conditions. LTI control is concerned with systems whose behavior we can completely specify; this means that the addition of carefully designed feedback produces a system that we know will behave in the desired way. When we can't solve a set of differential equations analytically, we don't know how the system will behave under a set of boundary conditions, so they need to treated on a case by case basis.

Obviously, controlling such a system is going to be difficult, as we can make no claims about the response of the system to a control input. Except for a few special cases, we will not be able to guarantee that the combined system will even be stable, let alone optimal. These two issues, stability and convergence, are much more difficult in the nonlinear case. Linear systems theory is extremely well developed, and it is often the case that convergence and stability for an adaptive controller can be proven. If we're trying to control, for example, a power plant, it may be that the consequences of the system becoming unstable are disastrous. This is why control engineers are so concerned about being able to prove stability, and it's also why they try to linearise system that are actually nonlinear. Of course, the other side to this coin is the control problem in biology. If we're interested in understanding how biological control systems work, it seems natural to borrow concepts from engineering. However, this can lead to us viewing these problems from a rather strange perspective. Biological systems routinely control highly nonlinear plants, for example, flapping wing systems, that just could not be controlled with current engineering technology. How can this be?

The point about biological control systems is that they tend to be rather small, so it seems implausible that they could possibly model the dynamics of a set of four or six flexible flapping wings. An optimization method such as evolution has no reason to be concerned with controllers that are provably stable; it's much more likely to go for the 'quick hack' approach, and let all the unsuccessful designs die. This means that it may be very difficult to get the data on biological control into some kind of unified framework. Control theory certainly gives us a way of understanding the nature of biological control problems, but understanding how the controllers actually work might mean a reinterpretation of control theory concepts.

3.6 The design process

The design of a control system involves a step by step process. The following section example of an antenna azimuth position control system is presented. The azimuth angle output of the antenna, θ_o (t), follows the input angle of the potentiometer, θ_i (t). Figure 3.3 shows the diagram of the system and the Figure 3.4 shows the functional block diagram of the system. The functions are shown above the blocks, and the required hardware is indicated inside the blocks.

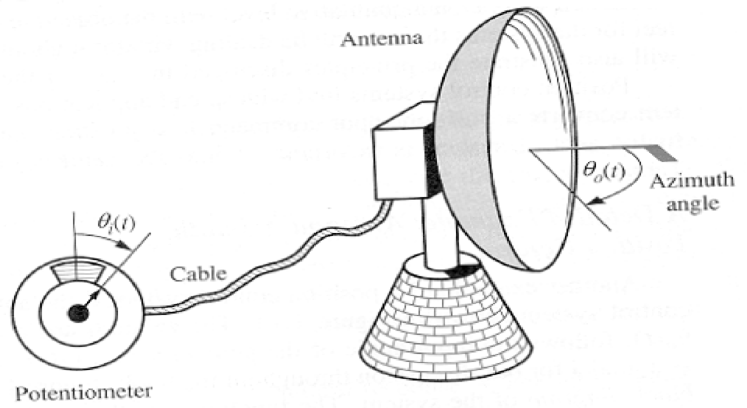


Figure 3.3: An antenna azimuth position control system (Nise, Norman [29])

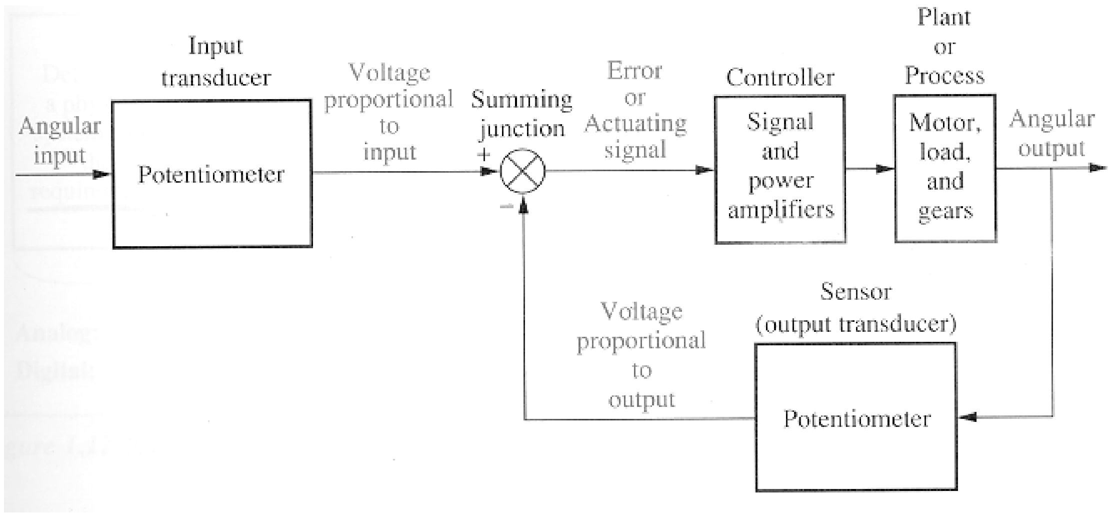

Input transducer
Angular input
Potentiometer
Voltage proportional to input
Summing junction
+
−
Error or Actuating signal
Controller
Signal and power amplifiers
Plant or Process
Motor, load, and gears
Angular output
Sensor (output transducer)
Potentiometer
Voltage proportional to output

Figure 3.4: Functional Block Diagram of a position control system (Nise, Norman [29])

Step 1 : Transform the requirements into a physical system

Antenna example: To design a control system for the position control of the azimuth of the antenna.

Requirements: To position the antenna from a remote location and describe features such as weight, and the physical dimensions.

Using this requirements the desired transient response and steady state accuracy are determined.

Step 2: Draw a functional block diagram

Involves the translation of the qualitative description of the system into a functional block diagram that describes the component parts of the system and shows their interconnections.

In the antenna example: Block diagram indicates functions as input transducer, the controller, and relevant descriptions of the amplifiers and the motors.

Step 3 : Create the schematic

Process of converting the physical system into a schematic diagram.

Relevant approximations of the system should be made and neglect certain phenomena or else making it difficult to extract information for the mathematical model.

After a single loop of design involving the analysis and interpretation of the system, decisions have to be made as to whether or not reasonable approximations were made.

If the designer feels that the system was not described fully, additional parameters are built into the design schematic.

Example : Potentiometers made like neglecting the friction or inertia, although these mechanical characteristics yield a dynamic response rather than an instantaneous response.

Implications of the assumptions are that the mechanical effects are neglected and the voltage across the potentiometer changes instantaneously as the potentiometer shaft turns.

Differential and power amplification : Assuming that the dynamics of the amplifier are rapid compared to the response time of the motors, hence we model it as pure gain K.

Dc motor : The output speed of the motor is proportional to the voltage applied to the motor’s armature. Armature consists of both the inductive and resistive effects and we assume that the inductance is negligible for the DC motor.

Load: The load consists of rotating mass and bearing friction. Hence the model consists of inertia and viscous damping whose resistive torque increases with speed like the automobile’s shock absorber or the screen door damper.

Step 4 : Develop a mathematical model (block diagram)

From the schematic, the physical laws such as the Kirchhoff’s laws and Newtons laws are used with modifying assumptions.

Kirchoff’s voltage law : The sum of voltages around a closed path equals zero.

Kirchoff’s current law: The sum of electric currents flowing from a node equals zero.

Newton’s laws : The sum of forces on a body equals zero, the sum of moments on a body equals zero.

These laws lead to mathematical models that describe the relationship between input and output of a dynamic system.

Model 1 : One such model is also a linear time invariant differential equation.

Model 2 : Transfer function is another way of modeling a system. This is obtained from linear time invariant differential equation using what is called the Laplace transform.

Laplace transform can be used for linear systems, but yields more intuitive information than the differential equations.

Ability to change system parameters and rapidly sense the effect of these changes on the system response.

Model 3 : State space methods : Advantage of modeling a system in state space is that they can be used for systems that cannot be described by differential equations.

These methods are also used to model systems for simulation the digital computer.

This representation turns an n-th order differential equation into n simultaneous first-order differential equations.

Step 5 : Reduce the Block diagram.

Subsystem models are interconnected to form a block diagram of a larger system where each block has a mathematical description.

The step involves the reduction of large number of subsystems into large system single block, with a mathematical description that represents the system form its input to its output.

Step 6: Analyze and design

Analyze to see if the response specifications and performance requirements can be met by simple adjustments of the system parameters.

If the specifications are not yet met the designer then designs additional hardware in order to effect a desired performance.

Test inputs are signals are used analytically and during testing to verify the design. Some of the standard input signals are impulses, steps, ramps, parabolas, and sinusoids.

Chapter 4

THE PID AND GALIL DMC1000 CONTROLLER

4.1 Digital control systems

The rapid development of digital technology has radically changed the boundaries of practical control system design options. It is now routinely feasible to employ very complicated, high-order digital controllers and to carry out the extensive calculations required for their design. These advantages in implementation and design capability can be achieved at low cost because of the widespread availability of inexpensive, powerful digital computers and related devices.

A digital control system uses digital hardware, usually in the form of a programmed digital computer, as the heart of the controller. In contrast, the controller, in an analog control system is composed of analog hardware, typically analog electronic, mechanical, electro-mechanical, and hydraulic devices. Digital controllers normally have analog elements at their periphery to interface with the plant; it is the internal workings of the controller that distinguish digital from analog control.

Digital control systems offer many advantages over their analog counterparts. Among these advantages are the following:

1. Low susceptibility to environment conditions, such as temperature, humidity, and component aging.
2. The cost reduction and interference rejection associated with digital signal transmission.
3. Zero “drift” of parameters.
4. High potential reliability.
5. The ability to perform highly complex tasks at low cost.
6. The potential flexibility of easily making changes in software.
7. Relatively simple interfaces with other digital systems, such as those for accounting, forecasting, and data collection.

Among the disadvantages are :

1. The introduction of errors (or noise) due to the finite precision of digital computations and the abrupt changes due to the discrete time nature of digital control.
2. The need for more sophisticated engineering in order to take advantage of higher-performance control algorithms.
3. Greater limitations on speed of operation.
4. Greater potential for catastrophic failure.

The signals used in the description of control systems are classified as continuous-time or discrete-time. Continuous-time signals are functions of continuous variable, whereas discrete-time signals are defined only for discrete values of the variable, usually with evenly spaced time steps. Discrete-time signals and their manipulators are inherently well suited to digital computation and are used in describing the digital portions of a control system. Most often, continuous-time signals are involved in describing the plant and the interfaces between a controller and the plant it controls. Signals are further classified as being of continuous amplitude or discrete amplitude. Discrete-amplitude (or quantized) signals can attain only discrete values, usually evenly spaced. The digital values are also quantized in amplitude signal shown in Figure 4.1, represented by a 3-bit binary code at evenly spaced time instants.

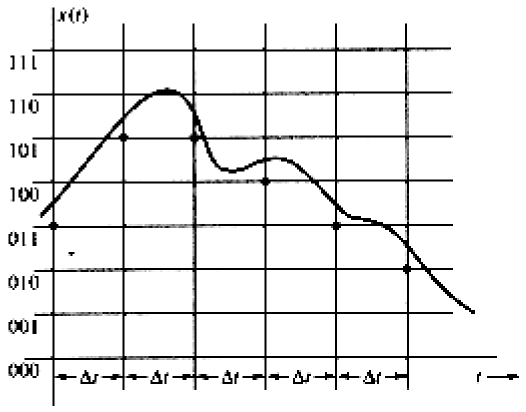


Figure 4.1: An example of a 3-bit quantized signal

(Santina, Stubberud, Hostetter [36])

In general, an n-bit binary code can represent only 2^n different values. Because of the complexity of dealing with quantized signals, digital control system design proceeds as if computer-generated signals were not of discrete amplitude. If necessary, further analysis is then done to determine if a proposed level of quantization error is acceptable.

A general control system diagram with a controller is show in Figure 4.2

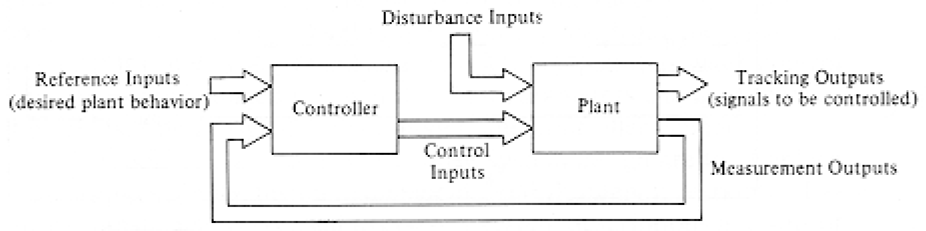

Disturbance Inputs
Reference Inputs (desired plant behavior)
Controller
Plant
Tracking Outputs (signals to be controlled)
Control Inputs
Measurement Outputs

Figure 4.2: A control system (Santina, Stubberud, Hostetter [36])

The plant is affected by input signals, some of which (the control inputs) are accessible to the controller, and some of which (the disturbance inputs) are not. Some of the plant signals (the tracking outputs) are to be controlled, and some of the plant signals (the measurement outputs) are available to the controller. A controller generates control inputs to the plant with the objective of having the tracking outputs closely approximate the reference inputs.

As described earlier, the systems and system components are classified by the nature of their mathematical model and termed continuous time or discrete time, according to the type of signals they involve. They are classified as linear if signal components in them can be superimposed. Any linear combination of signal components applied to input produces the same linear combination of corresponding output components; otherwise the system is non-linear. A continuous-time system or component is time invariant if its properties do not change with time. Any time shift of the input produces an equal time shift of every corresponding signal. If a continuous-time system is not time-invariant, then it is time-varying. On the other hand, if the properties of a discrete-time system do not change with step, then it is called step-invariant [26]. And, if the discrete-time system is not step-invariant, then it is step-varying.

Figure 4.3 shows the example of a digital control system for a continuous-time plant.

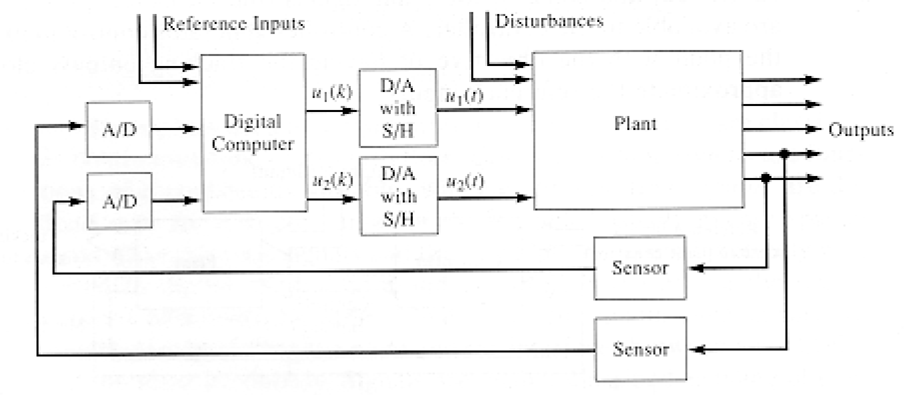


Figure 4.3: A digital control system controlling a continuous-time plant

(Santina, Stubberud, Hostetter [36])

The system has two reference inputs and five outputs, only two of which are measured by the Sensors. The analog-to-digital converters (A/D) perform sampling of the sensor signals and produce binary representations of these sensor signals. The digital controller algorithm then modifies the sensor signals and generates digital control inputs $u_1(k)$ and $u_2(k)$. These control inputs are then converted to analog signals via digital-to-analog converters (D/A). This process of transforming digital codes to analog signals begins by converting the digital codes to signal samples and then producing step reconstruction from the signal samples by transforming the binary coded digital inputs to voltages. These voltages are held constant for a sampling period T until the next samples are available. The process of holding each sample to perform step reconstruction is termed sample and hold.

The system also usually consists of a real-time clock that synchronizes the actions of the A/D and D/A and the shift registers. The analog signals inputs $u_1(k)$ and $u_2(k)$ are applied to the plant actuators or control elements to control the plant's behavior.

There are many variations on this theme, including situations in which the sampling period is not fixed, in which the A/D and D/A are not synchronized, in which the system has many controllers with different sampling periods, and in which sensor produce digital signals and the actuator accepts digital commands. Research such as Dr. Cooter's group on time warp simulation could be cited.

Two important classes of control systems are the regulator and the tracking system (or servo system). In the former, the objective is to bring the system-tracking outputs near zero in an acceptable manner, often in the face of disturbances. For example, a regulator might be used to keep a motor-driven satellite dish antenna on a moving vehicle accurately pointed in a fixed direction, even when the antenna base is moving and vibrating and the antenna itself is buffeted by winds. In a tracking system, the objective is for system outputs to track, as nearly as possible, an equal number of reference input signals. Regulation is a special case of tracking, in which the desired system-tracking output is zero.

4.2 PID controller

PID stands for Proportional, Integral, Derivative. Controllers are designed to eliminate the need for continuous operator attention. Cruise control in a car and a house thermostat are common examples of how controllers are used to automatically adjust some variable to hold the measurement (or process variable) at the set-point. The set-point is where one would like the measurement to be. Error is defined as the difference between the set-point and measurement values.

$$E = (S - M) \qquad (4.1)$$

Where, E is the error, S is the set-point and M is the measurement value.

The variable being adjusted is called the manipulated variable which usually is equal to the output of the controller. The output of PID controllers will change in response to a change in measurement or set-point. Manufacturers of PID controllers use different names to identify the three modes. These equations show the relationships:

P	Proportional Band = 100/gain	
I	Integral = 1/reset	(units of time)
D	Derivative = rate = pre-act	(units of time)

Where gain is the value of the amplification factor in the control loop, reset is the time taken for the controller to bring the output to the desired output state and the rate or pre-act is the actual rate of change of the system stability to achieve the desired state.

Depending on the manufacturer, integral or reset action is set in either time/repeat or repeat/time. One is simply, the reciprocal of the other. Note that manufacturers are not consistent and often use reset in units of time/repeat or integral in units of repeats/time. Derivative and rate are the same.

4.2.1 Proportional

With proportional band, the controller output is proportional to the error or a change in measurement (depending on the controller).

$$O = E*100/P \qquad (4.2)$$

Where O is the output of the controller. With a proportional controller, offset (deviation from set-point) is present. Increasing the controller gain will make the loop become unstable. Integral action was included in controllers to eliminate this offset.

4.2.2 Integral

With integral action, the controller output is proportional to the amount of time the error is present. Integral action eliminates offset.

$$O = \frac{1}{I} \left(\int e(t) \ \ dt \right) \qquad (4.3)$$

Notice that the offset (deviation from set-point) in the time response plots is now gone. Integral action has eliminated the offset. The response is somewhat oscillatory and can be stabilized some by adding derivative action.

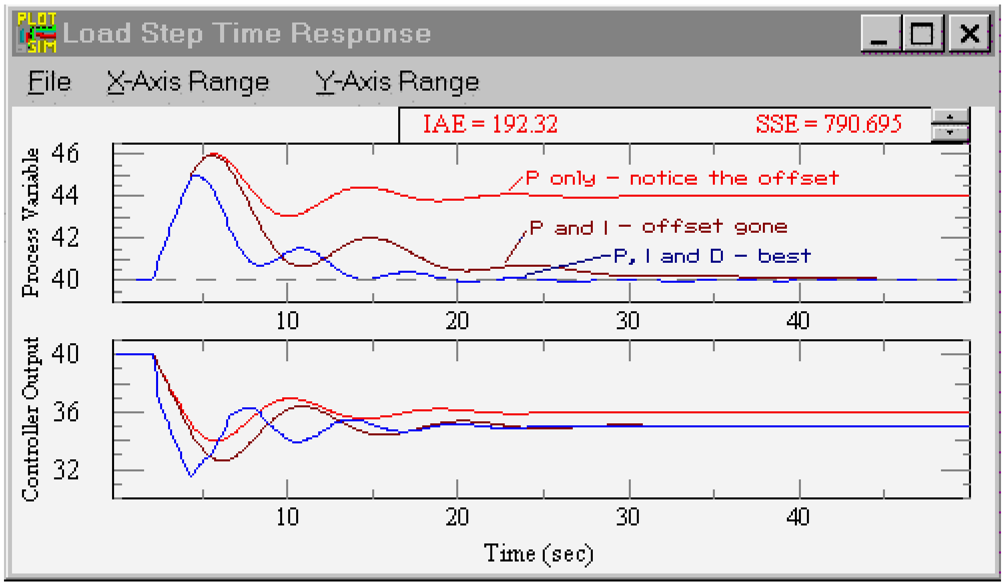


Figure 4.4: A Time Response plot showing controller action

(Graphic courtesy of ExperTune Inc. Loop Simulator)

Integral action gives the controller a large gain at low frequencies that results in eliminating offset and "beating down" load disturbances. The controller phase starts out at -90 degrees and increases to near 0 degrees at the break frequency. This additional phase lag is what you give up by adding integral action. Derivative action adds phase lead and is used to compensate for the lag introduced by integral action.

4.2.3 Derivative

With derivative action, the controller output is proportional to the rate of change of the measurement or error. The controller output is calculated by the rate of change of the measurement with time.

$$O = D\,(dm/dt) \qquad (4.4)$$

Where m is the measurement at time t.

Derivative action can compensate for a changing measurement. Thus derivative takes action to inhibit more rapid changes of the measurement than proportional action. When a load or set-point change occurs, the derivative action causes the controller gain to move the "wrong" way when the measurement gets near the set-point. Derivative is often used to avoid overshoot.

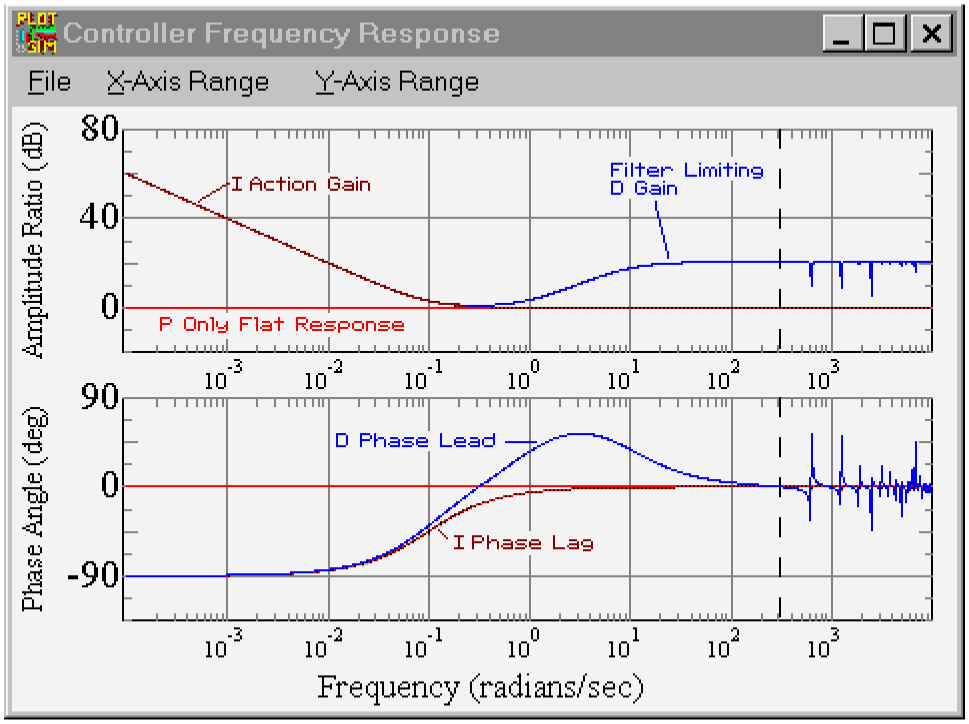


Figure 4.5: Controller frequency response plot

(Graphic courtesy of ExperTune Inc. Loop Simulator)

Derivative action can stabilize loops since it adds phase lead. Generally, if one uses derivative action, more controller gain and reset can be used.

With a PID controller the amplitude ratio now has a dip near the center of the frequency response. Integral action gives the controller high gain at low frequencies, and derivative action causes the gain to start rising after the "dip”. At higher frequencies the filter on derivative action limits the derivative action. At very high frequencies (above 314 radians/time; the Nyquist frequency) the controller phase and amplitude ratio increase and decrease quite a bit because of discrete sampling. If the controller had no filter the controller amplitude ratio would steadily increase at high frequencies up to the Nyquist frequency (1/2 the sampling frequency). The controller phase now has a hump due to the derivative lead action and filtering. (Graphic courtesy of ExperTune Inc. Loop Simulator.)

The time response is less oscillatory than with the PI controller. Derivative action has helped stabilize the loop.

4.3 Control loop tuning

It is important to keep in mind that understanding the process is fundamental to getting a well-designed control loop. For the Galil DMC-1000 controller that was used the user manual [49], describes a set of tuning procedure to be followed.

4.4 Galil DMC 1000

The DMC-1000 series motion controller is a state-of the-art motion controller that plugs into the PC bus. Extended performance capability over the previous generation of controllers include: 8 MHz encoder input frequency, 16-bit motor command output DAC, +/- 2 billion counts total travel per move, faster sample rate, bus interrupts and non-volatile memory for parameter storage. The controllers provide high performance and flexibility while maintaining ease-of-use and low cost.

Designed for maximum system flexibility, the DMC-1000 is available for one, two, three or four axes per card (add on cards are available for control of five, six, seven, or eight axes). The DMC-1000 can be interfaced to a variety of motors and drives, including stepper motors, servo motors and hydraulic actuators.

Each axis accepts feedback from a quadrature linear or rotary encoder with input frequencies up to 8 million quadrature counts per second. For dual-loop applications in which an encoder is required on both the motor and the load, auxiliary encoder inputs are included for each axis.

The DMC-1000 provides many modes of motion, including jogging, point-to-point positioning, linear and circular interpolation, electronic gearing and user-defined path following. Several motion parameters can be specified, including acceleration and deceleration rates, velocity and slew speed. The DMC-1000 also provides motion smoothing to eliminate jerk.

For synchronizing motion with external events, the DMC-1000 includes 8 optoisolated inputs, 8 programmable outputs and 7 analog inputs. I/O expansion boards provide additional inputs and outputs or interface to OPTO 22 racks. Event triggers can automatically check for the elapsed time, distance and motion to be complete.

Despite its full range of sophisticated features, the DMC-1000 is easy to program. Instruction are represented by two letter commands such as BG for begin and SP for speed. Conditional instructions, jump statements, and arithmetic functions are included for writing self-contained applications programs. An internal editor allows programs to be quickly entered and edited, and support software such as the SDK (servo design kit) allows quick system set-up and tuning.

To prevent system damage during machine operation, the DMC-1000 provides several error handling features. These include software and hardware limits, automatic shut-off on excessive error, abort input, and user-definable error and limit routines.

4.5 Microcomputer section

The main processing unit of the DMC-1000 is a specialist 32-bit Motorola 68331 series microcomputer with 64K RAM(256K available as an option), 64K EPROM and 256 bytes EEPROM. The RAM provides memory for variables, array elements and application programs. The EPROM stores the firmware of the DMC-1000. The EEPROM allows certain parameters to be saved in non-volatile memory upon power down.

4.6 Motor Interface

For each axis, a GL-1800 custom, sub-micron gate array performs quadrature decoding of the encoders at up to 8 MHz, generates a +/- 10 Volt analog signal (16 Bit D-to-A) for input to a servo amplifier, and generates step and direction signal for step motor drives.

4.7 Communication

The communication interface with the host PC uses a bi-directional FIFO (AM470) and includes PC-interrupt handling circuitry.

4.8 General I/O

The DMC-1000 provides interface circuitry for eight optoisolated inputs, eight general outputs and seven analog inputs(12-bit ADC). Controllers with 5 or more axes provide 24 inputs and 16 outputs. Controllers with 1 to 4 axes can add additional I/O with an auxiliary board, the DB-10096 or DB-10072. The DB-10096 provides 96 additional I/O. The DB-10072 provides interface to up to three OPTO 22 racks with 24 I/O modules each.

4.9 Amplifier

For each axis, the power amplifier converts a +/- 10 -Volt signal from the controller into current to drive the motor. (For stepper motors, the amplifier converts step and direction signals into current). The amplifier should be sized properly to meet the power requirements of the motor. For brushless motors, an amplifier that provides electronic computation is required. The amplifiers may either be pulse-width-modulated (PWM) or linear. They may also be configured for operation with or without a tachometer. For current amplifiers, the amplifier gain should be set such that a 10 Volt command generates the maximum required current. For example, if the motor peak current is 10 A, the amplifier gain should be 1 A/V. For velocity mode amplifiers, 10 Volts should run the motor at the maximum speed.

4.10 Encoder

An encoder translates motion into electrical pulses, which are fed back into the controller. The DMC-1000 accepts feedback from either a rotary or linear encoder. Typical encoders provide two channels in quadrature, known as CHA and CHB. This type of encoder is known as quadrature encoder. Quadrature encoders may either be single-ended (CHA and CHB) or differential (CHA, CHA-,CHB, CHB-). The DMC-1000 decodes either type into quadrature states or four times the number of cycles. Encoders may also have a third channel (or index) for synchronization.

For stepper motors, the DMC-1000 can also interface to encoders with pulse and direction signals

.

There is no limit on encoder line density; however, the input frequency to the controller must not exceed 2,000,000 full encoder cycles/second (8,000,000 quadrature counts/sec).

For example, if the encoder line density is 10000 cycles per inch, the maximum speed should be 200 inches/second.

The standard voltage level is TTL (zero to five volts); however, voltage levels up to 12 volts are acceptable. (If using differential signals, 12 volts can be input directly to DMC-1000. Single-ended 12 volt signals require a bias voltage input to the complementary inputs).

To interface with other types of position sensors, such as resolvers or absolute encoders, Galil offers the DB-10096 auxiliary card, which can be customized for a particular sensor.

Chapter 5

THE AUVS CONTEST AND THE BEARCAT DESIGN

5.1 Introduction

This chapter reviews the annual international ground robotics competition and the Bearcat robot, the steering mechanism dynamics and the design of the control system with specific emphasis on the controller design. This was the control system for the experimental vehicle taken to the competition, the simulation results and the results of which are presented in the next chapter.

5.2 The competition

Starting in 1991, the Oakland University had been organizing an Annual International Ground Robotics competition. In this competition, unmanned, automated guided robotic vehicles have to travel around an obstacle course. The robots have to stay within the track, and avoid the obstacles laid out on the course. This is a classic example of where the navigation and response of the various control systems play a very important role.

The design challenge set for the competitors by the organizers has been to make the course as difficult as possible. Until 1996, no team had been successful in their attempts to complete the course, without knocking over any of the obstacle, or staying off the track. In 1997, however, a few teams did complete the course successfully, and hence the deciding factor was the time and who would be able to complete the course in the least time.

Rules of the competition:

1. The vehicles are allowed to stray off the course. The vehicles are required to stay inside the edges of the track, and going off the track results in loss and elimination.
2. Each vehicle is allowed three runs, and the best run of the vehicle is taken into account for the final result.
3. The vehicles are not allowed to collide into obstacles. Touching the obstacle results in a loss of points, and moving or knocking the obstacles over results in elimination.
4. The vehicles are allowed to travel at a speed greater than 5 mph, and should have facilities for emergency stopping.

The vehicles are subjected to a safety inspection before they are allowed to compete in the actual competition. This inspection ensures that all the vehicles comply with the requirements. During the runs, if the judges feel the vehicle may be safe hazard or if it violates the safety requirements, they can disqualify the vehicle.

5.3 The Robot structural design

The Cincinnati Center for Robotics Research at the University of Cincinnati has been a participant in the competition, designing the Bearcat Robot.

The Bearcat Robot weighs approximately 600 lbs., and is 4 feet wide and 6 feet long. In the 1997 competition, the track was 10 feet wide. This meant that the space left for maneuvering was 3 feet on either side of the robot, not taking into account any obstacles. Had the organizers placed the obstacles more towards the center of the track, the robot would have difficulty in going around the obstacles. It would also have probably lost sight of the track, and veered off it. Hence, one of the major considerations for the future competitions should be to have a smaller robot, which can be maneuvered much more easily, and which would also be able to have a sharp turning radius ideally a Zero Tuning Radius (ZTR).

5.4 System design and development

An autonomous mobile robot is a sophisticated, computer controlled, intelligent system. The adaptive capabilities of a mobile robot depend on the fundamental analytical and architectural designs of the sensor systems used. The mobile robot provides an excellent test bed for investigations into generic vision guided robot control since it is similar to an automobile and is a multi-input, multi-output system. The major components of the robot are: Vision guidance system , Steering control system,Obstacle avoidance system, Speed control, Safety and braking system, Power unit and the Supervisor control PC.

The following is a brief description on the design and development of the main subsystems of the mobile robot.

5.5 Vision guidance system:

The purpose of the vision guidance system is to obtain information from the changing environment, the obstacle course, which is usually bounded by solid as well as dashed lines. The robot then adapts this information through its controller, which guides the robot along the obstacle course. For line tracking, two JVC CCD cameras are used for following the left and right lines. Only one line is followed at a time; however, when one camera loses the line, a video switch changes to the other camera. Image processing is accomplished with the Iscan image tracking device. This device finds the centroid of the brightest or darkest region in a computer-controlled window, and returns the X,Y coordinates of its centroid as well as size information of the blob. If no object is found, a loss of track signal is generated. This information is updated every 16 ms; however the program must wait 10 ms after moving the window to get new data. This results in a 52 ms update time for the vision system. The camera is angled down at 32 degrees and panned to the right at 30 degrees. This setup gives a 4 ft wide view of the ground. Once the data points are collected, they are entered into the algorithms. From these calculations, the angle and distance are sent to the motion control sub-system. Fig.2 gives the flowchart for the vision algorithm.

Image co-ordinates are two-dimensional while physical co-ordinates are three-dimensional. In an autonomous situation , the problem is to determine the three-dimensional coordinates of a point on the line given its image coordinates. As a solution to this problem, an innovative algorithm is developed to establish a mathematical and geometrical relationship between the physical 3-D ground coordinates of the line to follow and its corresponding 2-D digitized image coordinates. The algorithm utilizes a calibration device to determine the focal length of the cameras and the orientation of the projection system with respect to the global coordinates system. The calibration device is constructed to obtain physical co-ordinates of a point on the line with respect to the centroid of the robot within an accuracy of 0.0001". From the physical and image coordinates, the camera parameters (coefficients) are computed through a C program subroutine. Figure 5.1 compares the X and Y coordinates for the measured and computed vision calibration sample points . As a result of this reliable performance, the direct coefficients computation model is implemented to solve the vision problem.

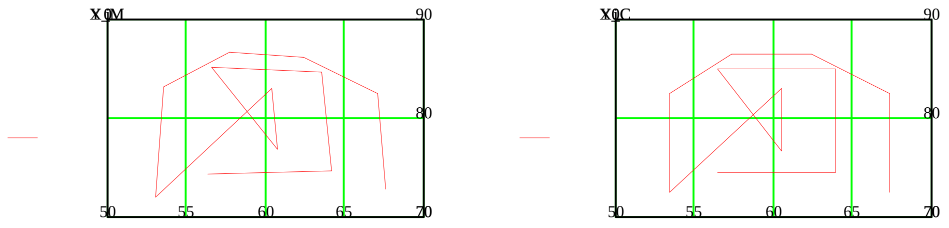


Figure 5.1 X and Y coordinates for the measured and computed vision calibration points

After the camera parameters are computed, the next stage is computing the physical coordinates, given any image coordinate. To show how the physical coordinates are computed given any image coordinate, another calibration is performed. Here, the z-coordinate for each of the points is treated as constant because in the real time implementation of this method, the z-coordinate is constrained by the ground. Table 1 shows the results of the physical coordinate computations. Correlation plots for the original and the computed x and y coordinates are computed. The linearity of the plots means that the difference between the original coordinates and the computed ones is very small. Also computed to ascertain or test the discrepancies between the two sets of coordinates is the mean square error. For each of the correlation plots, the mean square error is 0.242 for the x-axis and 0.295 for the y-axis. With a mean square error of within a tenth of an inch, the calibration process is considered to be accurate and reliable enough to compute the physical coordinates of a real life point on a ground. Thus the vision algorithm computes the x and y coordinate of a physical point with respect to the centroid of the robot and establishes a geometrical relationship between the points relative to the centroid of the robot.

Table 5.2: Set of Original and Computed Calibration Data Points

	Original physical Coordinates			Computed Physical Coordinates		Image Coordinates		
	x	y	z	x	y	x	y	
1	48.856	71.047	-22.533	265	341	220	48.464	70.745
2	56.346	71.054	-22.524	286	454	187	56.667	71.404
3	56.346	81.548	-22.564	342	329	135	55.949	81.358
4	48.831	81.583	-22.552	323	219	158	48.803	81.789
5	52.919	73.109	-19.057	344	377	153	53.33	73.475
6	52.87	79.537	-19.034	321	301	124	52.605	79.154
7	44.55	72	-24	265	245	43.97	71.762	
8	47.75	72	-24	330	229	48.414	71.789	
9	46.75	74.5	-24	265	221	46.229	74.649	
10	46	79	-24	215	195	46.417	79.236	
11	49.75	82	-24	236	168	49.908	81.871	
12	40.15	83	-24	195	40.362	83.149	81	

The system block diagram is shown in Figure 5.3.

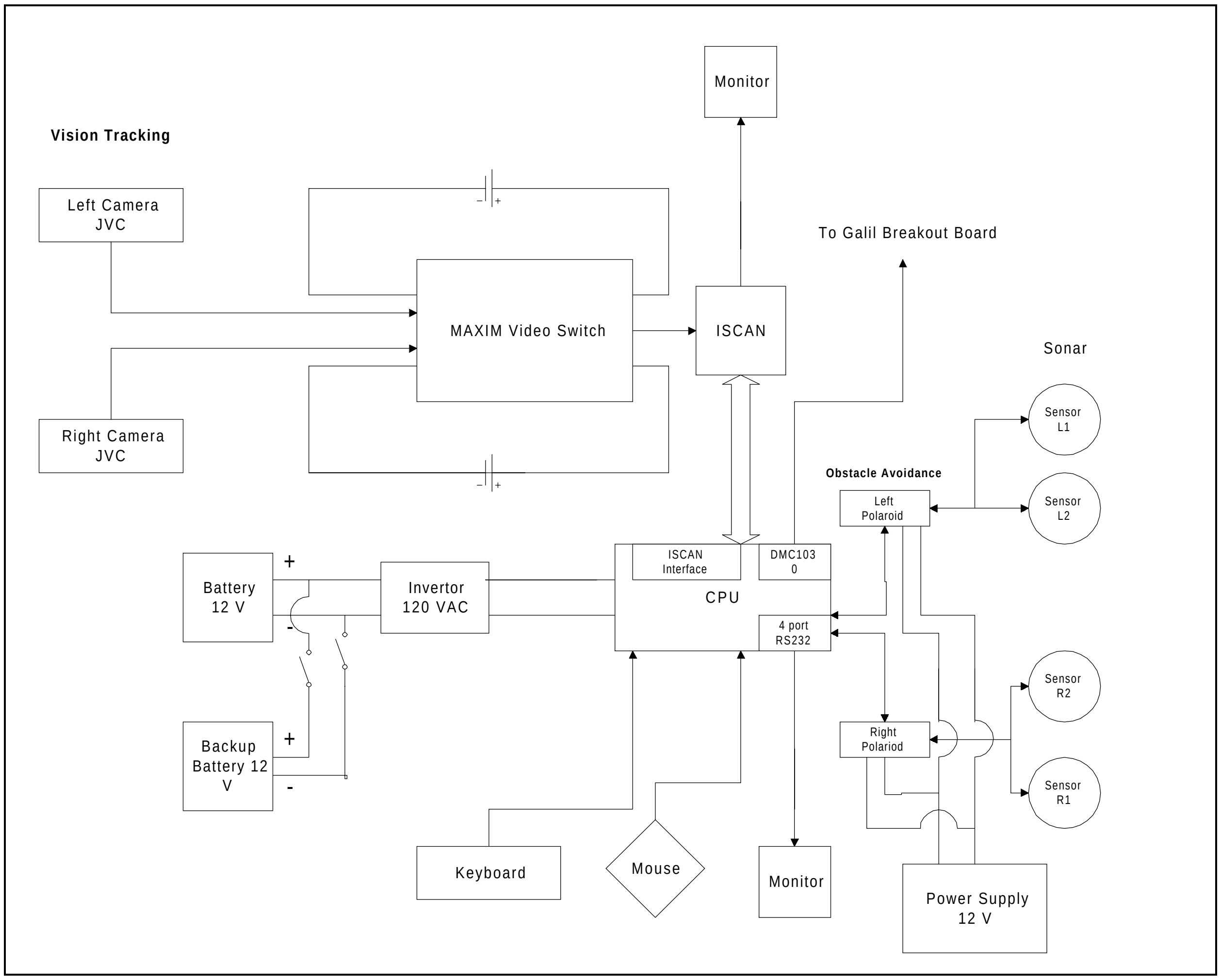


Figure 5.3: System block diagram

.

5.6 Obstacle avoidance system

The obstacle avoidance system consists of four ultrasonic transducers. A Polaroid ultrasonic ranging system is used for the purpose of calibrating the ultrasonic transducers. An Intel 80C196 microprocessor and a circuit board with a liquid crystal display is used for processing the distance calculations. The distance value is returned through a RS232 port to the control computer. The system requires an isolated power supply: 10-30 VDC, 0.5 amps. The two major components of an ultrasonic ranging system are the transducer and the drive electronics. In the operations of the system, a pulse of electronic sound is transmitted toward the target and the resulting echo is detected. The elapsed time between the start of the transit pulse and the reception of the echo pulse is measured. Knowing the speed of sound in air, the system can convert the elapsed time into a distance measurement.

The drive electronics has two major categories, digital and analog. The digital electronics generate the ultrasonic frequency. A drive frequency of 16 pluses per at 52 kHz is used in this application. All the digital functions are generated by the Intel microprocessor. The analog functionality is provided by the Polaroid integrated circuit. The operating parameters, such as the transmit frequency, pulse width, blanking time and the amplifier gain, are controlled by a developers software package provided by Polaroid.

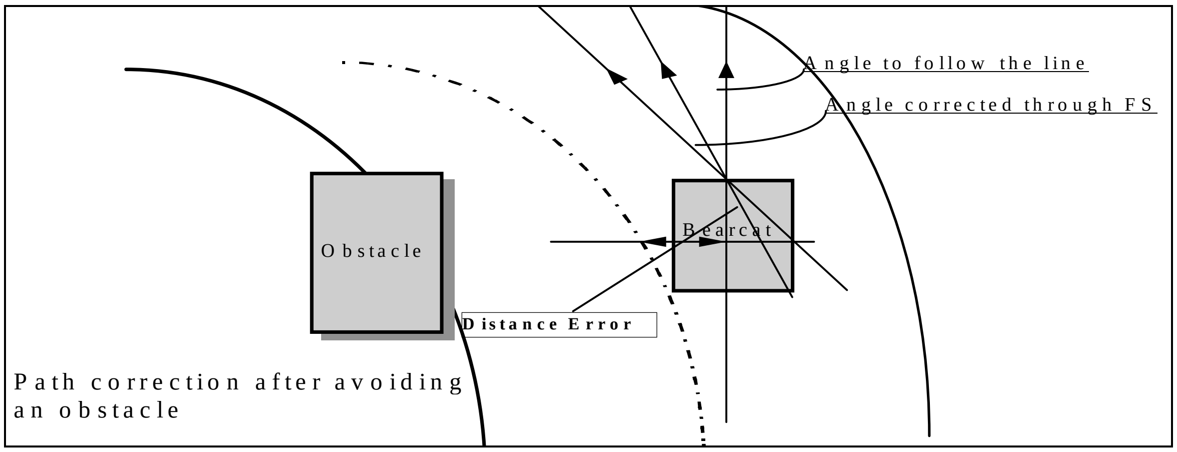

Angle to follow the line
Angle corrected through FS
Obstacle
Bearcat
Distance Error
Path correction after avoiding an obstacle

Figure 5.4: Obstacle avoidance strategy

Pseudo code for obstacle avoidance

```
Loop starts
Get input from vision input ( Scanning device )
Process and validate input.
Calculate the distance error and the angle error.
Check for obstacle on the right and left
If the obstacle is less than three feet and greater than two feet
    Change the distance error value
            If the distance error is negative and the obstacle is on the right
            Increase the distance error by 10
            If the distance error is negative and the obstacle is on the left
            Reduce the distance error by 5
            If the distance error is positive and the obstacle is on the right
            Reduce the distance error by 5
            If the distance error is positive and the obstacle is on the left
            Increase the distance error by 10
    Input the distance error and angle error to the fuzzy controller
    Steer the robot according to the output steering angle.
If the obstacle is less than two feet and greater than one feet
    If the obstacle is on the left steer the robot to right by 10 degrees
    Else steer the robot to the left by 10 degrees
If the obstacle is within one feet
```

```
        If the obstacle is on the left steer the robot to right by 20 degrees
        Else steer the robot to the left by 20 degrees
End Loop
```

The strategy for obstacle avoidance, in short, is to modify the distance error input to the fuzzy controller and use the same fuzzy controller again to compute the corrected motion direction. This way redesign of a separate controller for obstacle avoidance is avoided. The pseudo-code explains how and by what amount the distance error should be modified.

5.7 Speed control

The robot base is an E-Z-Go[i] golf cart. This cart is driven by a 36-volt, 55 amp. traction motor. Several designs were considered for controlling the large amount of power required for the traction motor, including: relays, power MOSFET's, and Insulated Gate Bi-polar Transistors (IGBTs). For the final design we choose the GE EV-1[ii] speed controller. This is a commercial controller designed for fork-lift and other industrial electric vehicles. The EV-1 is a silicon controlled rectifier (SCR), pulse width modulation (PWM) controller with a 0-10 volt control signal coming from the Galil[iii] DMC-1030 motor controller and sufficient output to drive the traction motor at full power. To complete the control loops, we have a BEI[iv] encoder mounted inside the front wheel. The encoder position signal is numerically differentiated to provide a velocity feedback signal. The electrical layout of the UC mobile robot traction system is shown in Figure 5.5

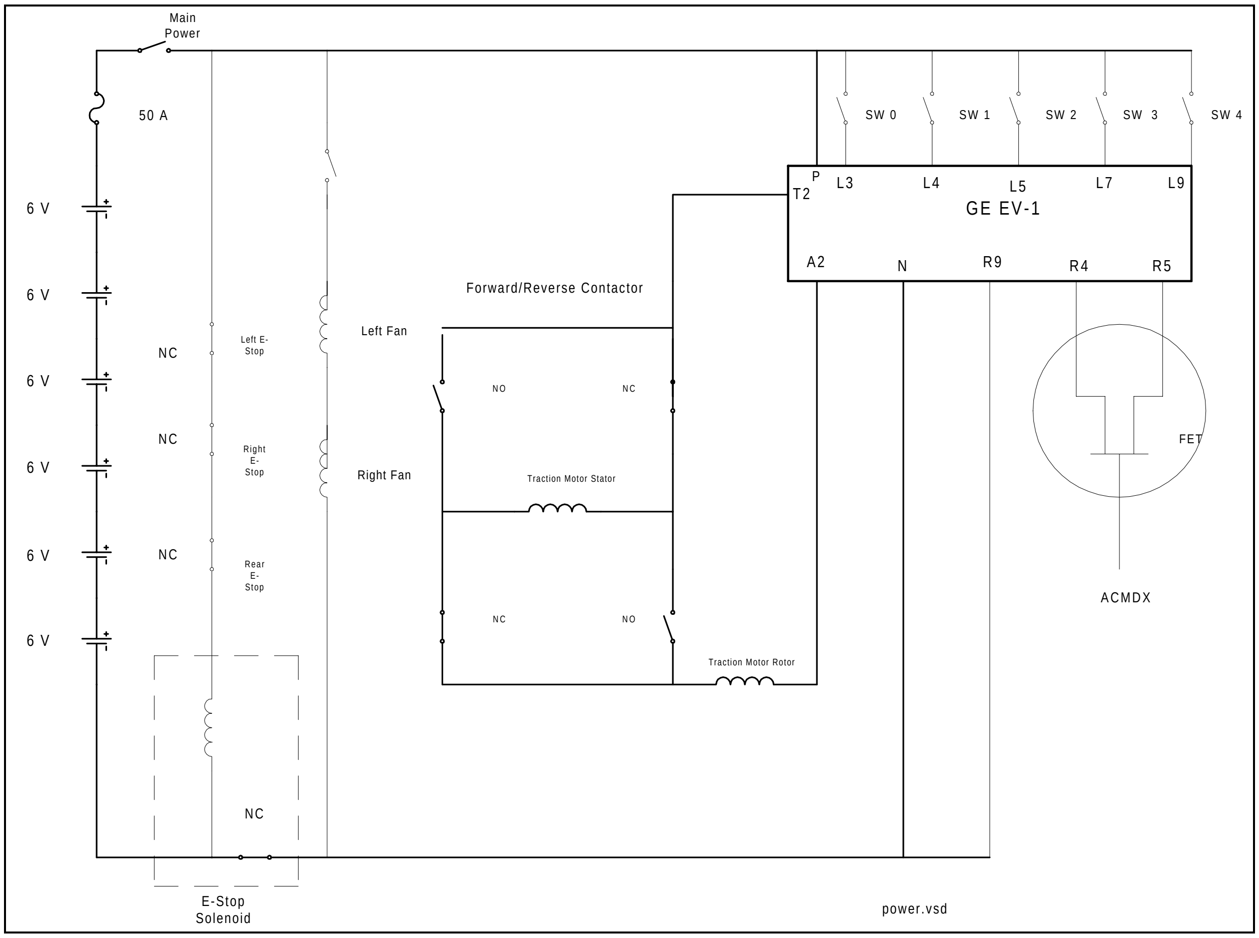


Figure 5.5: Overall traction control system

Safety is of primary concern in the system design. For safety reasons, the EV-1 has a set of three sequential switches that must be activated in order for it to run. That is, a single switch cannot turn on the machine; a sequence of three switches must be activated in a prescribed order. Also, in the main power loop, a solenoid is connected through three

E-stops, and the remote stop, as well as through the computer. This design should prevent any possible *runaway* of the vehicle since it provides a disconnect of power to all systems and application of the brake, not just breaking the low current control circuits.

The speed is controlled by the computer through the Galil motion control by varying the voltage across the R5 to R4 connections on the EV-1. A 0 voltage across these terminals will cause the motor to go at full speed. A maximum voltage of -4.5 volts will cause the motor to go at creep speed. The emergency stop circuit includes a normally closed solenoid switch in series with this power circuit. If the e-stop switch is activated, the solenoid circuit opens and cuts power to the traction motor.

Chapter 6

STEERING MECHANISM KINEMATICS AND THE PROPOSED CONTROL MODEL

6.1 Rack and pinion kinematics

The steering linkage for the mobile robot is a crank mechanism shown in Figure 6.1. The crank length is L and the connecting rod length is R. Point A is a fixed pivot point about which the steering wheel rotates.

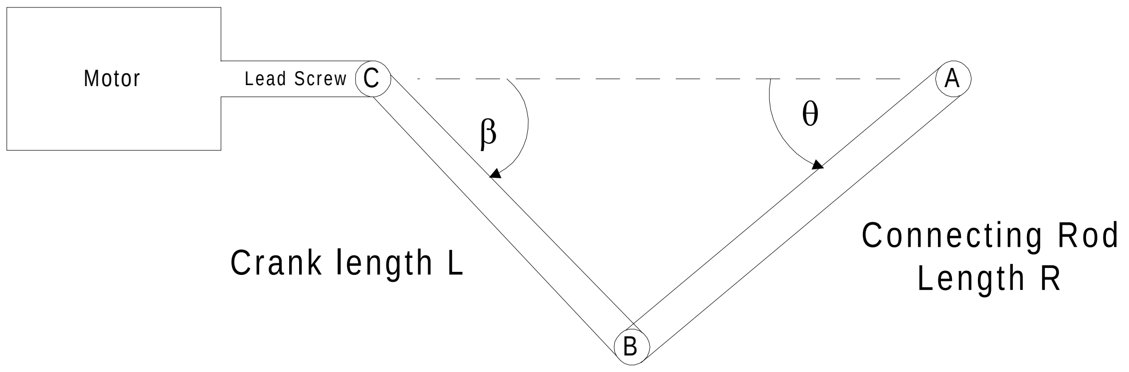


Figure 6.1: Crank mechanism

The amount of rotation of the crank AB about point A is measured by θ. The rotation of the crank about point C is measured by β. The distance points B moves is given d = R θ.

Using the law if sines, the angle β between the connecting rod BC and the horizontal can be determined by:

$$\frac{\sin(\theta)}{L} = \frac{\sin(\beta)}{R} \quad \text{or} \quad \beta = \operatorname{asin}\left(\sin(\theta)\cdot\frac{R}{L}\right) \qquad (6.1)$$

Note that this assumes that the motion of point C is horizontal. If point C moves vertically, this is a different problem. The main question is: if the lead screw advances a distance, x, what is the change of the angle, θ? Also, given a change of angle, what is the distance x?

The motion of connecting rod BC may be written in vector form as:

$$v_C = v_B + v_{CB}$$

$$v_{CB} = L\cdot\frac{d}{dt}\beta$$

$$d_{CB} = L\cdot\beta \qquad (6.2)$$

This relative velocity relationship is shown in Figure 6.2.

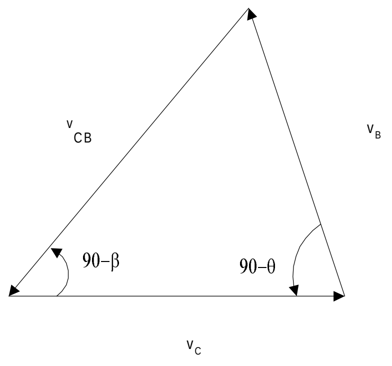


Figure 6.2 Relative velocities

One way to simplify the analysis is to consider the vertical and horizontal components of the vector velocity.

The vertical components are given by the following.

$$v_{Cy} = v_{By} + v_{CBy}$$

$$0 = R\cdot\frac{d}{dt}\theta\cdot\cos(\theta) + L\cdot\frac{d}{dt}\beta\cdot\cos(\beta)$$

$$\frac{d}{dt}\beta = \frac{(R\cdot\cos(\theta))\cdot\frac{d}{dt}\theta}{\sqrt{1-\left(R\cdot\frac{\sin(\theta)}{L}\right)^2}}$$

Using the law of cosines gives:

$$S^2 = L^2 + R^2 - 2\cdot L\cdot R\cdot\cos(\pi - \beta - \theta)$$

Solving for β gives:

$$\beta = \operatorname{acos}\left[\frac{1}{2}\cdot\frac{\left(S^2 - L^2 - R^2\right)}{(L\cdot R)}\right] - \theta$$

Solving for θ gives:

$$\theta = \operatorname{acos}\left[\frac{1}{2}\cdot\frac{\left(S^2 - L^2 - R^2\right)}{(L\cdot R)}\right] - \beta$$

The vertical component does not involve the variable x and is presented just for completeness.

An analysis of the horizontal components gives the following.

$$\frac{d}{dt}x = R\cdot\sin(\theta)\cdot\frac{d}{dt}\theta\cdot\left(1 + R\cdot\frac{\cos(\theta)}{\sqrt{L^2 - R^2\cdot\sin(\theta)^2}}\right)$$

Integrating this expression assuming x = 0 when θ = 0, gives:

$$x = R\cdot(1-\cos(\theta)) + R\cdot\left[\sqrt{\left(\frac{L}{R}\right)^2 - \sin(\theta)^2} - \frac{L}{R}\right]$$

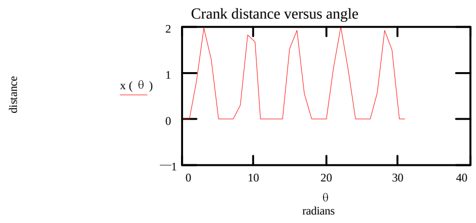
This is the forward kinematic solution, that is, given θ, we can compute x.

For example, if L=R=1, then x is shown below as a function of θ.

$\theta := 0 .. 5\cdot 6.28$

$L := 1 \qquad R := 1$

$$x(\theta) := R\cdot(1-\cos(\theta)) + R\cdot\left[\sqrt{\left(\frac{L}{R}\right)^2 - \sin(\theta)^2} - \frac{L}{R}\right]$$



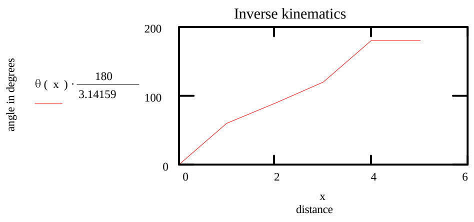
We can also determine the inverse kinematic solution, that is, given x, determine θ.

$$\cos(\theta) = \frac{\dfrac{2\cdot x\cdot(L+R) - 2\cdot L\cdot R - 2\cdot R^2 - x^2}{x - L - R}}{2\cdot R}$$

$$\theta = \pi - \mathrm{acos}\left[\frac{\left(2\cdot x\cdot L + 2\cdot x\cdot R - 2\cdot L\cdot R - 2\cdot R^2 - x^2\right)}{\left(-2\cdot x\cdot R + 2\cdot L\cdot R + 2\cdot R^2\right)}\right]$$

$$\theta(x) := \pi - \mathrm{acos}\left[\frac{\left(2\cdot x\cdot L + 2\cdot x\cdot R - 2\cdot L\cdot R - 2\cdot R^2 - x^2\right)}{\left(-2\cdot x\cdot R + 2\cdot L\cdot R + 2\cdot R^2\right)}\right]$$

$x := 0 .. 5$



Although the graph only shows one, there are two solutions for θ given a value of x. These correspond to mirrrow image positions of the crank.

6.2 Steering control system model

The steering system of the AGV helps maneuver it to negotiate curves and drive around obstacles on the course. The original steering system of the 3-wheeled cart used a rack and pinion design. This was replaced by a computer controlled steering which is a lead screw design activated by a Parker linear actuator. This linear actuator produces 110 ft-lb. of torque, creating a maximum turning speed of 20 degrees a second. This motor is controlled from the computer through the Galil DMC-1030. The DMC 1030 signal is amplified by an Electro-Craft amplifier, which provides three phase voltages to the brushless DC motor amplifier. For position feedback, a BEI encoder is directly mounted on the steering wheel giving a positive position feedback with 0.20-degree resolution.

The elements of a servo system include

1. A position servomotor with (Electrocraft brush type DC motor)
2. An Encoder
3. A PID controller (Galil DMC 1000 motion control board)
4. An amplifier (BDC 12)

6.3 Modeling the motor and the amplifier

The system can be configured in three modes namely, voltage loop, current loop and the velocity loop. The transfer function relating the input voltage V to the motor position P depends upon the configuration of the system.

The voltage loop:

In the case of the voltage loop, the amplifier acts as a voltage source to the motor. The gain of the amplifier will be K_v. And the transfer function of the motor with respect to the voltage will be

$$\frac{P}{V} = \frac{K_v}{[K_v s(sT_m + 1)(sT_e + 1)]} \tag{6.3}$$

Where,

$$T_m = \frac{RJ}{K_t^2} \tag{6.4}$$

$$T_e = \frac{L}{R}(s) \tag{6.5}$$

The motor parameters and the units are:

K_t : Torque constant (Nm/A)

R : Armature resistance

J : Combined Inertia of the motor and the load (kg-m^2)

L : Armature inductance

The current loop:

In this mode the amplifier acts as a current source for the motor. The corresponding transfer function will be as follows

$$\frac{P}{V} = \frac{K_a K_t}{Js^2} \tag{6.6}$$

Where,

K_a = Amplifier gain

K_t and J are as defined earlier.

The steering motor in the mobile robot was run on the current lop mode. The current loop allows a position control feedback and used typically for a position control system.

The velocity loop

In the velocity loop, a tachometer feedback to the amplifier is incorporated. The transfer function is now the ratio of the Laplace transform of the angular velocity to the voltage

$$\frac{\omega}{V} = \frac{\frac{K_a K_t}{J_s}}{1 + \frac{K_a K_t K_g}{J_s}} = \frac{1}{[K_g(sT_1 + 1)]} \qquad (6.7)$$

input. This is given by

Where,

$$T_1 = \frac{J}{K_a K_t K_g} \qquad (6.8) \qquad \textit{and} \qquad \textit{therefore}$$

$$\frac{P}{V} = \frac{1}{[K_g s(sT_1 + 1)]} \qquad (6.9)$$

The Encoder

The encoder is an integral part of the servomotor and has two signals A and B, which are in quadrature and 90 degrees out of phase. Due to the quadrature relationship, the resolution of the encoder is increased to 4N quadrature counts/rev. N is the number of pulses generated by the encoder per revolution.

The model of the encoder can be represented by a gain of

$$K_f = \frac{4N}{2\pi}[counts/rad] \tag{6.10}$$

The controller :

The Digital-to-Analog Converter (DAC) converts a 14-bit number to analog voltage. The input range of numbers is 16384 and the output voltage is ± 10/20V.

For the DMC-1000, the DAC gain is given by K_d = 0.0012 [v/count]

The digital filter

The digital filter has discrete system transfer function given by

$$D(z) = \frac{K(z-A)}{z + \frac{Cz}{z-1}} \tag{6.11}$$

The filter parameters are K, A and C. These are selected by commands KP, KI and KD. Where,

KP	is the proportional gain
KI	is the integral gain
KD	is the derivative gain of the PID controller.

The two sets of parameters for the DMC-1000 are related according to the equations,

$$K = K_p + K_d \tag{6.12}$$

$$A = \frac{K_d}{(K_p + K_d)} \tag{6.13}$$

$$C = \frac{K_i}{8} \tag{6.14}$$

The Zero order Hold (ZOH)

The ZOH represents the effect of the sampling process, where the motor command is updated once per sampling period. The effect of the ZOH can be modeled by the transfer function,

–

$$H(s) = \frac{1}{(1 + s\frac{T}{2})} \tag{6.15}$$

In most applications, H(s) can be approximated as 1.

Having modeled the system, we now obtain the transfer functions with the actual system parameters. This is done for the system as follows

The motor and the amplifier

The system is operated in a current loop and hence the transfer function of the motor and amplifier is given by

$$\frac{P}{V} = \frac{K_a K_t}{Js^2} \tag{6.16}$$

The encoder:

The encoder on the Dc motor has a resolution of 1000 lines per revolution. Since this is in quadrature, the position resolution is given by 4*1000 = 2000 counts per revolution.

The encoder can be represented by a gain of

$$K_f = \frac{4 \times N}{2\pi} = \frac{4000}{2\pi} = 636.9 \qquad (6.17)$$

The DAC:

From the Galil DMC-1000 specifications, the gain of the DAC is represented as

K_d = 0.0012 V/count

The ZOH:

The ZOH transfer function is given by

$$H(s) = \frac{1}{(1 + s\frac{T}{2})} \qquad (6.18)$$

Where, T is the sampling time. The sampling time in this case is 0.001s. Hence the transfer function of the ZOH is :

$$H(s) = \frac{2000}{s + 2000} \qquad (6.19)$$

6.4 System compensation objective:

The analytical system design is aimed at closing the loop at a a crossover frequency ω.

The following are the parameters of the system.

1. Time constant of the motor	K_t	0.31 N-m/amp
2.Moment of inertia of the system	J	0.035534 Kg-m^2 (approx)
3. Motor resistance	K_m	0.42 ohms
4. Amplifier Gain in current loop	K_a	2 amps/volt
5. Encoder gain	K_f	636.9 counts/rev

The design objective is set at obtaining a phase margin of 45 degrees.

- Motor :

$$M(s) = \frac{K_t}{Js^2} = \frac{0.31}{0.035534} = \frac{8.724}{s^2} \qquad (6.20)$$

- Amplifier

$$K_a = 2 \qquad (6.21)$$

- DAC

$$K_d = \frac{10}{2048} = 0.00488 \qquad (6.22)$$

- Encoder

$$K_f = 636.9 \qquad (6.23)$$

- ZOH

$$H(s) = \frac{2000}{s + 2000} \qquad (6.24)$$

- Compensation filter

  $G(s) = P + sD$ (6.25)

  $L(s) = M(s)\ K_a\ K_f\ K_d\ H(s)$ (6.26)

$$= \frac{1.08521 \times 10^5}{s^2 (s + 2000)} \tag{6.27}$$

The overall transfer function of the system is computed as follows :

$$\frac{0.471964S^2 + 4.74338S}{1.885 \times 10^{-4} S^5 + 0.004147S^4 + 7.54S^3 + 300.5938S^3 + 3021.05S} \tag{6.28}$$

Chapter 7

SIMULATION AND RESULTS

7.1 Introduction

This chapter presents the simulation of the control system and the results of the various tests. The simulation model was set up in MATLAB and the SIMULINK toolbox was used. The system PID gains were obtained for satisfactory design response. The WSDK (windows servo design kit) software was used and the system was tuned using the software. Test results are presented and the actual performance of the various subsystems are obtained. The integration of the steering system module with the other sub-systems is tested by monitoring the performance of the vehicle on a test track constructed at the University of Cincinnati.

7.2 Simulation on SIMULINK

The design objective was to obtain a stable control over the steering control system. The phase margin was required to be under 45 degrees with the percentage overshoot not exceeding 20%. For this purpose a Galil DMC 10000 motion control board was used, which has the proportional integral derivative controller (PID) digital control to provide the necessary compensation required in the control of the steering motor. The steering system without a controller is modeled as shown in Figure 7.1

Figure 7.1: Simulink model of the uncompensated system

The step responses for the uncompensated and compensated systems are shown in the appendices A and B. The simulation involved three steps

1. A Matlab source code file 1which has the model of the transfer function of the entire system shown in appendix F.
2. A second Matlab source file 2 converting the digital gains to analog gains also shown in appendix F.
3. A simulink graphics model which takes the analog values of the gains and simulates the system step response.

The SIMULINK model consists of a step input signal fed to a summation block. The values for the PID controller are set with a MATLAB file calculating the analog gains, for the various digital values tried on the actual system. These analog values are stored in the MATLAB kernel and are automatically read when the simulation file is run. These analog values in the PID controller model then adjusts the input signal and feeds it to the zero order hold. The zero order hold holds the input level until the next input is given in order to smoothen the input wave, the sampling time of which is modeled in the zero-order block. This digital signal is then fed to a digital to analog converter and then to an amplifier. The gain on the amplifier was fixed at 2 in the model but has several adjustments on the actual device. This amplified signal is then fed to the load, which the motor and the steering wheel. The actuator is a linear lead screw that drives a crank mechanism. The movement on the wheel is detected by the encoder to give a feed back signal, which is fed back to the summation block for correction.

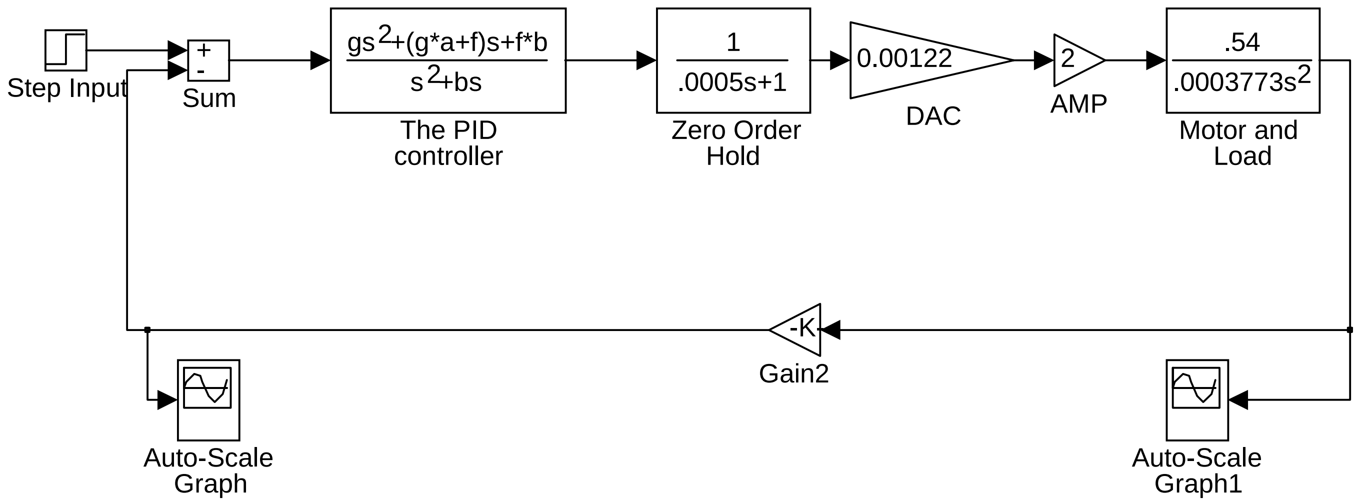


Figure 7.2: Simulink model of the compensated system

The unit step response was simulated in MATLAB and it was found that the phase margin was around 40 dB and percentage overshoot was less than 15%. The values for the PID controller were tested on the actual vehicle and were fine tuned using the software kit supplied by Galil Motion Inc.-WSDK 1000 . This software also allowed us to estimate the frictional losses in the gear mesh, the linear actuator and the rack and pinion mechanisms. A conservative tuning was performed and valued for the PID controller were identified suitable for the system. The Bode plot frequency response was measured by supplying a sinusoidal input signal to the open loop system and recording the response through the encoder. A phase margin of 40 degrees and a gain margin of 10 decibels was achieved. Then the step response was checked to minimize the overshoot and select a critically damped response. The actual tests were made in three conditions: steering wheel off the ground, steering wheel on the ground with robot moving and steering wheel on the ground with robot stationary. The torques for these conditions were measured at: 15 foot pounds, 20 foot pounds and 30 foot pounds, respectively. Tuning of the amplifier parameters especially loop gain and selection of the PID parameters were very important and required iterative adjustments.

The interface for the system was implemented using a Galil 1030 motion control computer interface board. A Galil breakout board permits the amplifier and encoded to be easily connected. The steering mechanism gets its input from the angle to be moved by two inputs : the angle from the obstacle avoidance and the angle from the vision algorithm. Feedback is provided at a low level by a position encoder and at a high level by the vision and sonar systems.

In the design process an incorrect estimation of the inertial load led to overload and burning-out of the amplifier As a solution, a torque limit was specified in the Galil motion program that limited the drive torque, current and voltage supplied by the amplifier. Other problems faced were setting of appropriate hardware control bias voltages on the amplifier and estimating turning torque. These were solved through experimental methods.

The WSDK kit is a menu driven graphical user interface that lets one tune the system. The system elements can be identified and various menu options are provided to make sure that the elements are connected properly. Then a closed loop test in done to see if the system is stable. Once it is stable the PID controller parameters are to be tuned. There are various tuning options available on the software. A first tuning test was done to get approximately close the system PID parameters. This was then checked with the MATLAB simulation and adjusted using the manual tuning option to obtain a desired satisfactory output. The system overshoot and settling times were seen in the tuning test module of the software. They are shown in the appendix C.

7.3 Computer control of the steering module

After the system was tuned the steering control module was tested through computer control. A C++ (TC++ compiler) code was written to achieve the computer control. The source code consisted of various modules (appendix E). Before the execution of the code one of the important step that needs to be done is the initialization of the Galil DMC 1000 controller. The controller has its own assembly code. The assembly code (appendix D) for the steering and speed control module is shown in the appendix. This code needs to be downloaded on the controller before the execution of the C++ code.

The number keys 4 (←) and 6 (→) on the number pad of the keyboard were designated to control the robot for a left and right turn respectively. The mobile robot showed excellent response when the source code was executed. The steering angle for one key press was adjustable and set in as a software input. From the tests conducted in the Center for Robotics research the steering angle response was excellent and was properly integrated with other subsystems.

7.4 Results of testing

After extensive laboratory testing of individual subsystems an oval outdoor test track was constructed to simulate the contest track with double lines, 4 inches wide spaced 10 feet apart with dashed segments and obstacles. The first outside test was conducted on May

10, 1997. The observations on the performance of the individual subsystems are summarized below:

7.4.1 Vision guidance system:

The vision system was able to successfully track straight lines, curves, negotiate sharp turns, as well as switch control between cameras when the line on either side disappeared. However, the external conditions did present some problems. If the weather turned sunny to cloudy or vice versa during the actual run the Iscan threshold had to be manually readjusted. Also, in certain cases the reflection of the wet grass on the track appeared brighter than the white line itself and the vision system picked those points and went off-track. Both these observations were incorporated in the Fault Diagnostic program and remedies suggested to over them.

7.4.2 Obstacle Avoidance System:

The sonar system reliably detected obstacles between 6 inches and 8 feet within an accuracy of 1 inch. The system interfaces between the Polaroid unit, Galil controller and the supervisory controller were found to be successful. The Fuzzy logic controller computed correction angles to drive the robot around the obstacles. Going down the slope on a ramp, it was observed that the sonars detected ground as an obstacle. This problem was taken care of by modifying the sonar algorithm such that if sonars on either side gave identical readings, the obstacle shall be ignored.

7.4.3 Steering control system :

The steering mechanism worked satisfactorily. Implementing the PID controller variables as obtained from the SIMULINK model gave a stable response with almost zero overshoot. The steering amplifier did smoke once when the initial position calibrated was biased in one direction and the amplifier drew excess current when made to steer in the opposite direction. To prevent repetition of this fault a safety fuse was added in the amplifier circuit.

7.4.4 Safety and Emergency Stop Braking System

This system was found to be extremely reliable and effective. The relay base was found to be drawing excess current in the initial test on the system. To protect it against the power spikes in the system, a 3-amp fuse was connected in the circuit. Also, it was found that the transmitter of the remote control unit drained its battery within 45 minutes. Though, this would not have presented any problems at the actual run at the contest, as a precautionary measure a provision was made to recharge the battery without physically removing it from the unit so as to ensure peak performance of the unit.

Chapter 8

CONCLUSION AND RECOMMENDATIONS

A stable test platform has been designed, constructed and tested. However, advanced control techniques need to be investigated. The system's modular design lends itself to a subsumption architecture, whereby any number of sensor systems could be connected and with a minimal programming effort be efficiently utilized. Also the use of a more heuristic methodology in the obstacle avoidance should be investigated. For the system to be more efficient and able to go at faster speeds, interrupt handling is a must. The program would then not have to constantly poll the obstacle avoidance or vision systems. Also, the motor control needs to have an interrupt to inform the control program when it has completed its move.

A modular intelligent robotic system has been developed for the 1997 AUVS competition. Over 2000 person hours were spent by the design team. The replacement costs of the vehicle is about $18,600. The system embodies speed control, vision line tracking and ultrasonic obstacle avoidance. The design utilizes independent sensor modules. These modules could be placed on any system to control it with minimal modifications.

References

[1] M. F. Abdin, "An investigation of the potentials of Bi-directioanl AGV Systems," proc. 5th intl. AGVS conf., Tokyo, oct. 1987

[2] M. H. E. Larcombe, "Tracking Stability of Wire Guided Vehicles, " proc. 1st Intl. AGVS cong., pp. 137-144, 1973.

[3] Ozguner, Umit. Unyelioglu, Konur A. Hatipoglu, Cem. Analytical study of vehicle steering control IEEE Conference on Control Applications - Proceedings 1995. IEEE, Piscataway, NJ, USA,95CH35764.. p 125-130

[4] Ackermann, Juergen. Guldner, Juergen. Sienel, Wolfgang. Steinhauser, Reinhold. Utkin, Vadim I. Linear and nonlinear controller design for robust automatic steering IEEE Transactions on Control Systems Technology. v 3 n 1 Mar 1995. p 132-142

[5] Sung, Eric. Loon, Ng Kok. Yin, Yee Chiang. Parallel linkage steering for an automated guided vehicle. IEEE Control Systems Magazine. v 9 n 6 Oct 1989 p 3-8

[6] Will, Anthony B. Zak, Stanislaw H. Modelling and control of an automated vehicle Vehicle System Dynamics. v 27 n3 Mar 1997. p 131-155

[7] Zadeh, A Ghazi. Fahim, A. El-Gindy, M. Neural network and fuzzy logic applications to vehicle systems: Literature survey International Journal of Vehicle Design. v 18 n 2 1997. p 132-193

[8] Kalyan Kolli and E.L. Hall. Steering Control System for a Mobile Robot

Proceedings of SPIE - The International Society for Optical Engineering. v 2591 1997. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA.

[9] Kaylan Kolli, Sreeram Mallikarjun, Krishnamohan Kola and Ernest L. Hall. Speed Control for a Mobile Robot Proceedings of SPIE - The International Society for Optical Engineering. v 2591 1997. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA.

[10] Kamga, Andre. Rachid, Ahmed. Speed, steering angle and path tracking controls for a tricycle robot Proceedings of the IEEE International Symposium on Computer-Aided Control System Design 1996.,96TH8136.. p 56-61

[11] Tagawa, Y. Ogata, H. Morita, K. Nagai, M. Mori, H. Robust active steering system taking account of nonlinear dynamics Vehicle System Dynamics. v 25 n Suppl 1996. p 668-681

[12] Matthews, Bradley O. Ruthemeyer, Michael A. Perdue, David. Hall, Ernest L. Development of a mobile robot for the 1995 AUVS competition Proceedings of SPIE - The International Society for Optical Engineering. v 2591 1995. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA. p 194-201

[13] Samu, Tayib. Kelkar, Nikhal. Perdue, David. Ruthemeyer, Michael A. Matthews, Bradley O. Hall, Ernest L. Line following using a two camera guidance system for a mobile robot Proceedings of Spie - the International Society for Optical Engineering. v 2904 1996. p 290-297

[14] Yoshimoto, Ken-ichi. Sakatoh, Masatoshi. Takeuchi, Makoto. Ogawa, Hideki. Automatic steering control algorithm using optical flow Jsae Review. v 16 n 2 Apr 1995. p 165-169

[15] BenAmar, Faiz. Steering behaviour and control of fast wheeled robots IEEE International Conference on Intelligent Robots and Systems. v 3 1997. IEEE, Piscataway, NJ, USA,97CB36108. p 1396-1401

[16] Fourier, Cornelius J. Deist, Leon A. Control of a mobile robot by means of a fuzzy algorithm

[17] Lucibello, Pasquale. State steering by learning for a class of nonlinear control systems Automatica. v 30 n 9 Sept 1994. p 1463-1468

[18] Mori, Yasuchika. Nyudo, Shin. Steering and speed control of a car by fuzzy-neural control Proceedings of the International Joint Conference on Neural Networks. Publ by IEEE, IEEE Service Center, Piscataway, NJ, USA. v 2 1993. p 1753-1756

[19] Hattori, A. Kurami, K. Yamada, K. Ooba, K. Ueki, S. Nakano, E. Control system for an autonomous driving vehicle Heavy Vehicle Systems. v 1 n 1 1993. p 99-113

[20] Hilton, J D. Hilton, D J. Design of an automatic steering and speed controller for a horticultural gantry machine. National Conference Publication - Institution of Engineers, Australia. Publ by IE Aust, Barton, Aust. n 92 pt 11. P 103-107

[21] Pears, N E. Bumby, J R. Steering control of an experimental autonomous vehicle. Transactions of the Institute of Measurement & Control. v 13 n 4 1991 p 190-200

[22] Kehtarnavaz, N. Sohn, W. Steering control of autonomous vehicles by neural networks. Proceedings of the American Control Conference. Publ by American Automatic Control Council, Green Valley, AZ, USA (IEEE cat n 91CH2939-7). v 3. p 3096-3101

[23] Sawada, T. Oguchi, Y. Satoh, C. Basic study on the influence of aging on steering control. Proceedings – Society of Automotive Engineers. Publ by SAE, Warrendale, PA, USA. p 763-769

[24] Krogh, Bruce H. guaranteed steering control. Proceedings of the American Control Conference 1985. Publ by IEEE, New York, NY, USA Available from IEEE Service Cent (Cat. n 85CH2119-6), Piscataway, NJ, USA p 950-955

[25] McMahon, C B. Tennes, B R. Burkhardt, T H. performance results: microprocessor-based steering controller using ultrasonic sensors. Paper - American Society of Agricultural Engineers Joseph, Mich, USA 83-1568,. Publ by ASAE, St. 34p

[26] santina stubberd

[27] Warwick, K. Control systems : an introduction. New York : Prentice Hall, 1989

[28] Thompson, S. Control systems engineering and design. Harlow, Essex, England : Longman Scientific & Technical ; New York, NY : Wiley, 1989

[29] Nise, Norman S. Control systems engineering / Norman S. Redwood City, Calif. : Benjamin/Cummings Pub. Co., c1992

[30] I. J. Nagrath, M. Gopal. Control systems engineering New York : Wiley, c1975

[31] Charles L. Phillips, Royce D. Harbor. Basic feedback control systems. Englewood Cliffs, N.J. : Prentice Hall, c1991

[32] Charles L. Phillips, Royce D. Harbor. Feedback control systems. Englewood Cliffs, N.J. : Prentice Hall, c1991

[33] J. C. Gille, M. J. Pelegrin [and] P. Decaulne. Feedback control systems; analysis, synthesis, and design New York, McGraw-Hill, 1959.

[34] Clarence J. Maday. Computer-aided design of feedback control systems for time response. Research Triangle Park, N.C. : Instrument Society of America, c1987.

[35] Charles L. Phillips, H. Troy Nagle. Digital control system analysis and design. Englewood Cliffs, N.J. : Prentice Hall, c1995.

[36] Mohammed S. Santina, Allen R. Stubberud, Gene H. Hostetter. Digital control system design Fort Worth : Saunders College Pub., c1994.

[37] Benjamin C. Kuo. Digital control systems. Ft. Worth : Saunders College Pub., c1992.

[38] Alberto Isidori. Nonlinear control. Berlin ; New York : Springer, c1995.

[39] Alberto Isidori Nonlinear control systems : an introduction. Berlin ; New York : Springer-Verlag, c1989.

[40] Cheng, R M H., Xiao, J W., LeQuoc, S., "Neuromorphic controller for AGV steering," Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, Piscataway, NJ, v 3. pp 2057-2062.

[41] Milacic, Vladimir R. Putnik, Goran D., " Steering rules for AGV based on primitive

function and elementary movement control**," Robotics and Computer-Integrated Manufacturing,** v 5 n 2-3 1989. pp 249-254.

[43] Cheng, R M H. Rajagopalan, Ramesh. "Kinematics of automatic guided vehicles with an inclined steering column and an offset distance, criterion for existence of inverse kinematic solutions," **Journal of Robotic Systems,** v 9 n 8 Dec 1992, pp 1059-1081.

[44] Ock Hyun Kim. Optimal Steering Control of an Auto-Guided-Vehicle with Two Motored Wheels, **Transactions of the Institute of Measurement & Control**. v 9 n 2 Apr-Jun 1987, pp 58-63.

[45] P.F. Muir and C.P. Neuman, "Kinematic Modeling of Wheeled Mobile Robots," **Journal of Robotic Systems**, 4(2), 1987, pp. 281-340.

[46] **E.L**. Hall and B.C. Hall, **Robotics: A User-Friendly Introduction**, Holt, Rinehart, and Winston, New York, NY, 1985, pp. 23.

[47] R.M.H. Cheng and R. Rajagopalan, "Kinematics of Automated Guided Vehicles with an Inclined Steering Column and an Offset Distance: Criteria for Existence of Inverse Kinematic Solution," **Journal of Robotic Systems**, 9(8), 1059-1081, Dec. 1992.

[48] General Electric, EV-1 SCR Control Manual, Charlottesville, Virginia 1986.

[49] Galil Inc, DMC-1000 Technical Reference Guide Ver 1.1, Sunnyvale, California 1993.

[50] Reliance Electric, Electro-Craft BDC-12 Instruction Manual, Eden Prairie, Minnesota 1993.

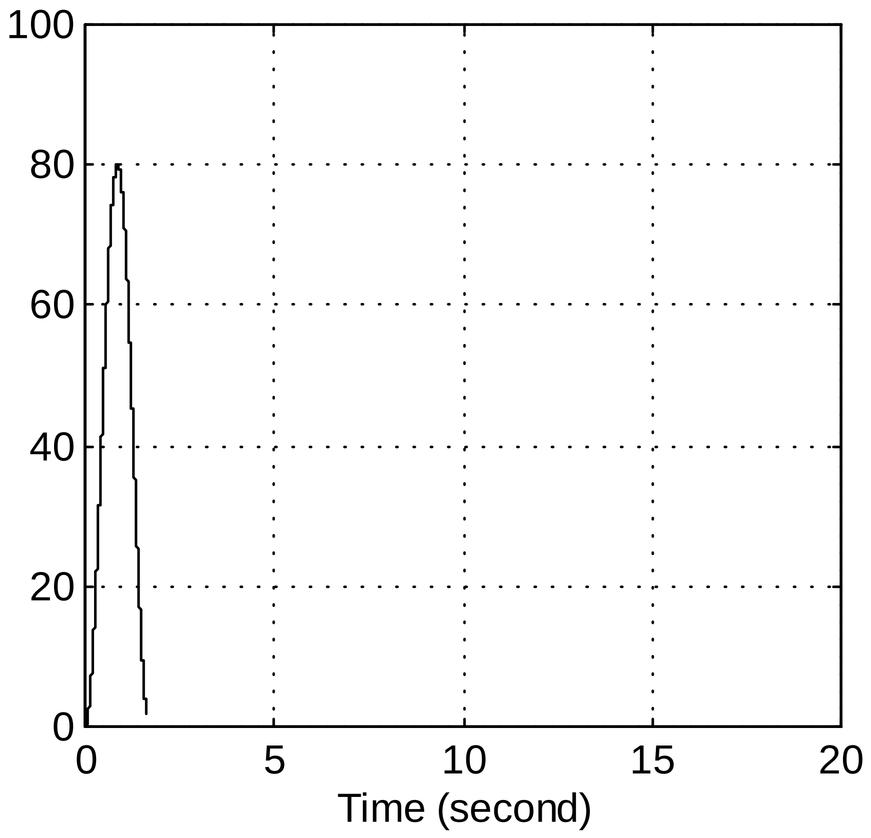
[51] Fauver Corp., Parker Linear Actuators Catalog, Cincinnati, Ohio 1993.

[52] Delco Corp., Motor Specifications, Dayton, Ohio 1993.

[53] BEI Motion Systems Company, Model H20 Technical Specifications, Goleta, California 1992

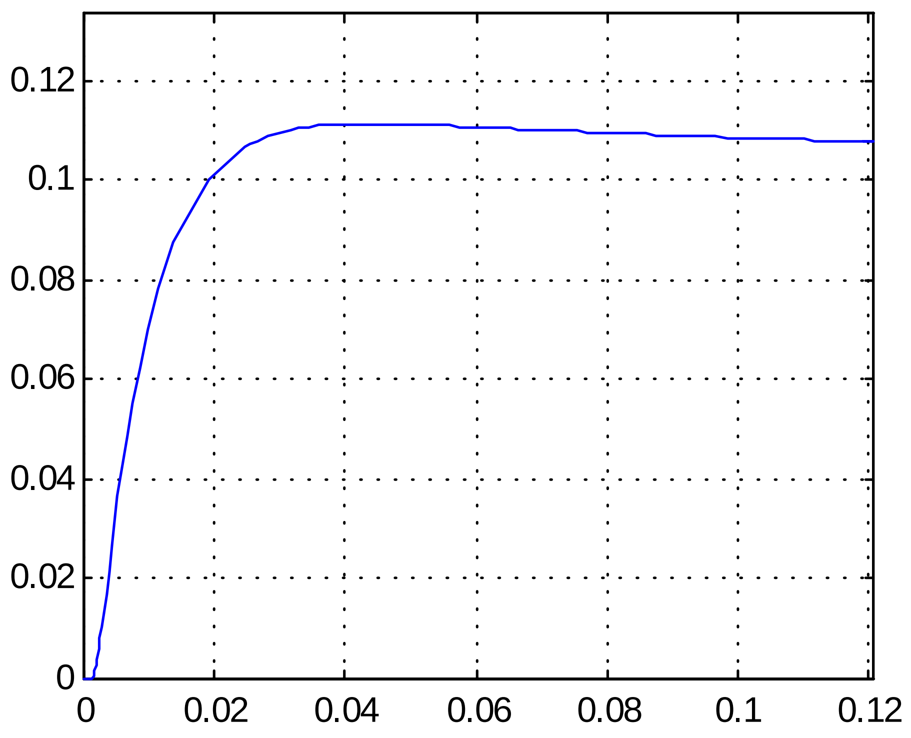
Appendix A

Step response for the Uncompensated system



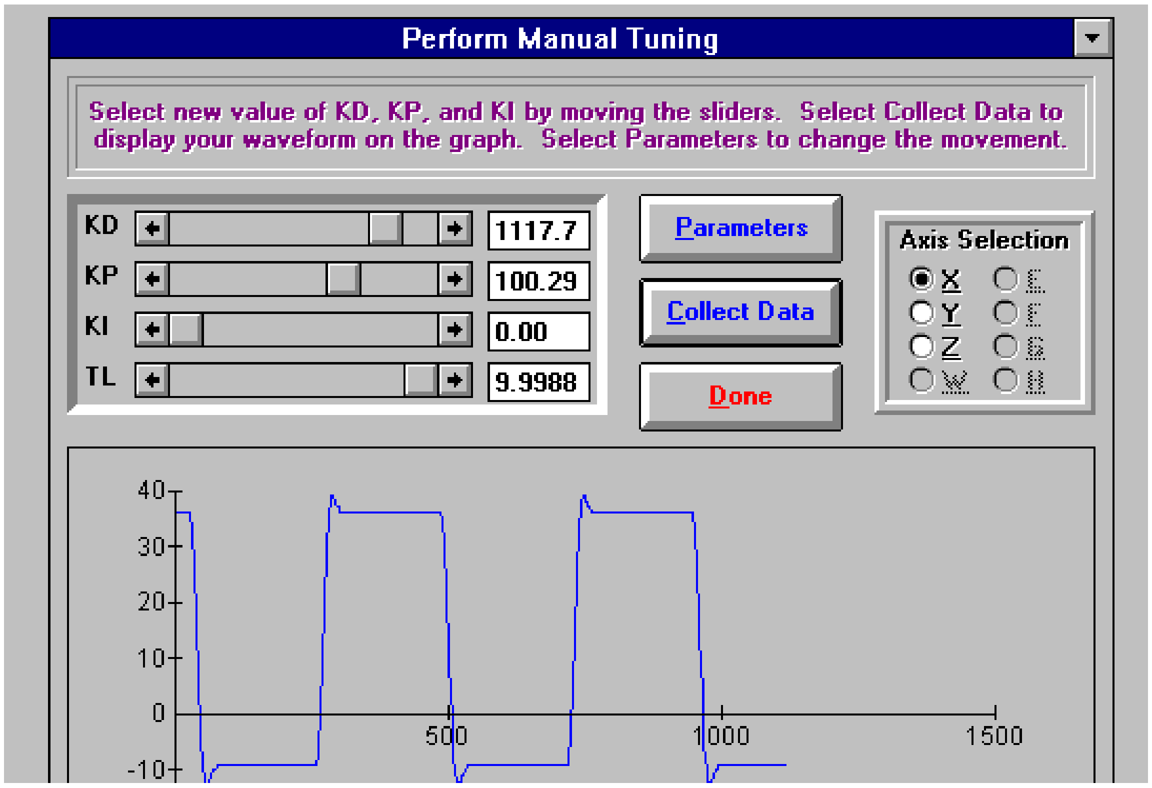
Appendix B

Step response for the Compensated system



Appendix C

WSDK software manual tuning step response


Perform Manual Tuning

Select new value of KD, KP, and KI by moving the sliders. Select Collect Data to display your waveform on the graph. Select Parameters to change the movement.

KD 1117.7

KP 100.29

KI 0.00

TL 9.9988

Axis Selection: [✓ X] Y Z W E F G H

Appendix D

Galil DMC –1000 Motion controller Assembly code

```
#INIT
OUT=0
INC=0
TWAIT=80
GAIN=.0005
GAINACC=.000
VEL=0
VELTAR=0
CB1
DIR=1
#SPEED
P1=_TPZ
T1=TIME
WT TWAIT
P2=_TPZ
VELOLD=VEL
VEL=(P2-P1)/TWAIT*1000
INC=(VELTAR-VEL)*DIR
ACCEL=(VEL-VELOLD)/TWAIT*1000*DIR
JS#REV,VELTAR<0
JS#FOR,VELTAR>0
OUT=(INC*GAIN)+(ACCEL*GAINACC)+OUT
JS#MAX,OUT>9
JS#NEG,OUT<0
OF OUT
JP#SPEED
EN
#ZERO
INC=0
EN
#MAX
OUT=9
EN
#FOR
CB1
DIR=1
EN
#REV
SB1
DIR=-1
EN
#LIMIT
JS #ZEROP,OUT<.1
INC=.2*OUT
EN
```

Appendix E

C++ Source code for the Steering control

```
// car.c          kalyan kolli 5-5-97

#include "car.h"
#ifndef CAR
#define CAR

void speedCAR(float val){
      //This command when invoked will set the speed of the golfcart.
      //It is called with a value between 7-9.9.

      char inc[10],send[15];
      gcvt(val,4,inc);
      char *command="VELTAR=";
      strcpy(send,command);
      strcat(send,inc);
      cout<<send<<"\n";
if(!TEST)
   submitDMC(send);
}
void stopCAR(){
      submitDMC("VELTAR=0");
      submitDMC("OUT=0");
      submitDMC("PA,0");
      submitDMC("BGY");
      }

void steerCAR(float val){
      //This function when invoked will put the steering wheel to the
absolute
      //angle given.    This angle ranges between +-20.

      char inc[10],send[15];
      val=val*2.85; // Changed from val*5 to val*2.85....Refer to
steering calibration.
      gcvt(val,4,inc);   // Used to convert floating point value to
string.
      char *command="PA,";
      strcpy(send,command);
      strcat(send,inc);

      if(!TEST){
            submitDMC(send);
            cout<<send<<"\n";
            submitDMC("BGY");
            submitDMC("AMY");
      }
}

void auto_steerCAR(float val){
      //This function when invoked will put the steering wheel to the
absolute
      //angle given.    This angle ranges between +-20.

      char inc[10],send[15];

      //slow car
      int spd=-.05*abs(val)+COMP_SPEED;
      speedCAR(spd);

      //steer car
      steerCAR(val);
}


void startCAR(){
      //This function when called will initialize the galil board.

      initDMC();
      set_upDMC();
       //   for(int wait= 0;wait<500;wait++);

       //   download("c:\galil\speed.dcp");
       //   submitDMC("XQ");
}
#endif
```

```
//#include <stdio.h>
/* This program when invoked gives the final angle by which the
steering module should respond.
//* developed by kalyan kolli        5-24-97
main()
{
float va=0,sondist=0,answer;
float final_angle(float, float);
while(va!=100)
{
printf("enter va and s=ondist\n");
scanf("%f%f",&va,&sondist);
answer = final_angle(va,sondist);
printf("teh answer is %f ",answer);
}
```

```
}
*/

float final_angle(float va, float sondist)
{
   int va_int,sondist_int,i=0,j=0;
   float i1_intpol,i2_intpol,answer;
   float a[10][10] = {{0,-20, -15, -10,-5, 0,5, 10, 15, 20},
            {40,-20,-16, -11,-6, -5,-3, 5, 10, 12},
            {30,-20, -17,-12,-11,-10,-5,-1,-5,-3},
            {20,-20, -18, -14,-13, -12,-10, -8, -10, -10},
            {10,-20, -20, -18,-18, -18,15, -12, -15, -15},
            {0,-20, -15, -10,-5, 0,5, 10, 15, 20},
            {-10,15, 15, 12,15, 18,18, 18, 20, 20},
            {-20,10,10,8,10,12,13,14,18,20},
            {-30,3,5,1,6,10,11,12,17,20},
            {-40,-12,-10,-5,-3,5,7,11,16,20}};
// initialize all the elements below:
  va_int=va/5;
  va_int *= 5;
//  printf(" \n %d ",va_int);
  sondist_int=sondist/10;
   sondist_int *= 10;
//   printf(" \n %d ",sondist_int);
  for(j=1;va_int != a[0][j];j++);
  for(i=1;sondist_int != a[i][0];i++);
//   printf(" \n %d, %d \n",i,j);
// interpolating between two rows, where row index is sonar distance
if((a[0][j+1]-a[0][j])!=0)
      i1_intpol=a[i][j] + (a[i][j+1]-a[i][j])*((va-a[0][j])/(a[0][j+1]-
a[0][j]));
      else
      i1_intpol=a[i][j];



if((a[0][j+1]-a[0][j])!=0)
      i2_intpol=a[i-1][j] +((a[i-1][j+1]-a[i-1][j])*((va-
a[0][j])/(a[0][j+1]-a[0][j])));
      else
      i2_intpol=a[i-1][j];

//    printf("\n %f, %f",i1_intpol,i2_intpol);

if((a[i-1][0]-a[i][0])!=0)
answer=i1_intpol+(((sondist-a[i][0])/(a[i-1][0]-a[i][0]))*(i2_intpol-
i1_intpol));
else
answer=i1_intpol;
```

```
 return(answer);
}

//galil.c               5-19-97
#include "galil.h"
#ifndef GALIL
#define GALIL


void writeDMC(char a){
      outp(DMC_Base,a);
}

int readDMC(){
      return(inp(DMC_Base));
}

void writeStatus(char a){
      outp(DMC_Status,a);
}

int readStatus(){
      return(inp(DMC_Status));
}

int galilHasInfo(){
      return((inp(DMC_Status) & 32)== 0);
}


int galilIsReady(){
      return((inp(DMC_Status)&16)==0);
}





void sendDMC(char str1[]) {
//This function when invoked will send a string to the DMC.
//Add a null character to the string
      char send[15];
      char *command=str1, *null = NULL;
      strcpy(send,command);
      strcat(send,null);


   if(galilIsReady()){
//Send character by character to DMC
```

```
      int i;
      for(i=0; send[i]; ++i) {writeDMC(send[i]);};
      writeDMC(13);     };
}

void download(char str1[])
{
//This function when download a file from the PC to the DMC buffer
   FILE *in;
      sendDMC("DL");
      cout<<"prepared to recieve\n";
      if( (in = fopen(str1, "rt")) == NULL)
   {
            fprintf(stderr, "Cannot open input file.\n");
      return;
      };

     while (!feof(in)) writeDMC(fgetc(in));
   fclose(in);
   cout<<"done";
   writeDMC('/');
   fromDMC();
   cout<<"last";
   return;
};

 double watch()
{
 // This Function when invoked will inform one full rotation of the
wheel.+
  double value;
//  sendDMC("DP,,0");
//  sleep(2);
  inDMC("TP");
//  sleep(2);

     value=inDMCreturn("TPZ");
// while (value<=1269);
// cout<<"one full rotation completed\n";
 return value;
}


char getDMC(){
      //This function when invoked will confirm data is available and
the will
      //retrieve it.

      if(galilHasInfo() ){
```

```
      return(readDMC());}
      else{
    return(0);}
}

char fromDMC(){
//This funtion when invoked will check to see if there is a character
//available and if so will return it.

      int count=0;
      while( !(galilHasInfo()) && (count<5000) ){
            count++;}
      if(count>=5000){
            cout<<"GALIL:    Time-Out error!\n";
            return(0);
            }

      else
            return(readDMC());
}


void inDMC(char str1[40]){
//This function when invoked will send a request for data, once the
command is
//recieved it will wait on the data, and display it.
      int count=0;
      sendDMC(str1);

      char str5[40];
      while(str5[count]!=':'){
      ++count;
      str5[count]=fromDMC();

      }
      str5[count-2]=NULL;

      for(int i=0;i<count;i++){
      str5[i]=str5[i+2];}

      cout<<"data recieved:"<<str5<<"DONE";
}

double inDMCreturn(char str1[40]){
//This function when invoked will send a request for data, once the
command is
//recieved it will wait on the data, and return the value as a double
int.
      int count=0; char *endptr;
      sendDMC(str1);
      char str5[40];
      while(str5[count]!=':'){
      ++count;
      str5[count]=fromDMC();

      }
      str5[count-2]=NULL;

      for(int i=0;i<count;i++){
      str5[i]=str5[i+2];}
      double  val=strtod(str5,&endptr);
      return(val);
}



void submitDMC(char str1[15]) {
//This function when invoked will send the DMC a command, and if the
command is
//legal and there are no errors it will say so.
      char error;
      sendDMC(str1);

//check for colon:
      error=fromDMC();
      if(error!=':'){
            cout<<"GALIL:    Error!!\n '"<<error<<"'";}
      else{
            cout<<"GALIL:    Command Received\n";}


};

void clearPC(){
//This function when called clears the PC buffer.

      int tries=0;
      while(tries++<2000){
            getDMC();
            };
}

void clearDMC(){
//This function when called clears the DMC buffer.

      writeStatus(0x01);
      writeStatus(0x80);
      writeStatus(0x80);
}
```

```
void initDMC(){
//This function when invoked will prepare the DMC by initalizing it to
128 FIFO

       // Set FIFO buffer to 128 bytes
       writeStatus(5);
       writeStatus(0);
       // clear dmc buffer
       clearDMC();
}

void set_upDMC(){
//This function when called sends the necessary settings for proper
operation
//of the steering and drive motors.
      submitDMC("KD,27.5");
      submitDMC("KP,739");
      submitDMC("KI,33");
      submitDMC("TL,9.98");
      submitDMC("FL,90");
      submitDMC("BL,-90");
      submitDMC("IT,1");
      submitDMC("SP,20000");
      submitDMC("AC,9216");
      submitDMC("DC,9216");

}


#endif
```

Appendix F

MATLAB code for the calculation of digital gains

Section of code which takes in digital gains and converts them into analog gains

```
GN=800
ZR=0.9
PL=0
KI=0

% Given GN...solve for K,A,B,C
K=GN;
A=ZR;
B=PL/256;
C=KI/8;

% solve for a,b,g,f

a=(2000-2000*A)/(1+A)
b=(2000-2000*B)/(1+B)
f=1000*C
g=K*(1+A)/(1+B)
```

Section of code for the compensated controller design

```
%root locus  design compensator parameters

num=[g,(g*a+f),f*b];
den=[0.0005,1.0005*5,b,0,0,0];

%solving for A,B,C,K given a,b,f,g
A=(2000-a)/(2000+a);
B=(2000-b)/(2000+b);
C=f/1000;
K=g*(1+B)/(1+A);

%galil parameters
GN=4*K;
ZR=256*A;
PL=256*B;
KI=256*C;

GN,ZR,PL,KI
grid;
RLOCUS(num,den)
title('root locus PLOT OF COMENSATED SYSTEM')
```

```
figure(2);
Bode(num,den)
margin(num,den),...
title('Bode plot OF COMPENSATED SYSTEM')
[gm,pm, wgm,wpm]=margin(num,den)
```

Section of code for the uncompensated system design

```
% Rlocus and Bode plots of uncompensated system

T=.001;
Kd=10/2048;
KA=2;
KT=.31;
Kf=2000/pi;
J=.035534;
K= KA*Kd*KT*Kf ;
num1= [J*T/2 J 0 K];
den1= [J*T/2 J 0 0];


% the rlocus plot of the uncompensated system
figure(1)
rlocus(num1,den1),...
title('RLOCUS PLOT OF UNCOMPENSATED SYSTEM'),...


% the bode plot of the uncompensated system
figure(2)
bode(num1,den1),...
margin(num1,den1),...
title('BODE PLOT OF UNCOMPENSATED SYSTEM')
```