Chapter 1

INTRODUCTION

Understanding motion control is one most important step for building an Autonomous Guided Vehicle (AGV). Control theory is a foundation for many fields including industrial automation. The topic is so broad that it can be used to study the economy, human behavior, spacecraft design, as well as the design of industrial robots and autonomous guided vehicles. Motion control systems often play a vital part in product manufacturing, assembly and distribution. Implementing a new system or upgrading an existing motion control system may require mechanical, electrical, computer and industrial engineering skills and expertise. Multiple skills are needed to understand the tradeoffs for a systems approach to the problem. Needs must be analyzed, requirements specified, sources selected for specific components and the subsystems integrated. Once a specific technology is selected, application engineers from suppliers can act as members of the design team to help ensure a successful implementation, which satisfies the production and cost requirements, quality control and safety.

Motion Control is defined [1] by the American Institute of Motion Engineers as:

> *"The broad application of various technologies to apply a controlled force to achieve useful motion in fluid or solid electromechanical systems."*

One way to consider the field of motion control is as mechatronics [1].

"Mechatronics is the synergistic combination of mechanical and electrical engineering, computer science, and information technology, which includes control systems as well as numerical methods used to design products with built-in intelligence." [1]

One application of motion control is the industrial robot [2]. Another is an autonomous guided vehicle [3, 4]. Autonomous guided vehicle is 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 autonomous guided vehicle in many fields.

Many researchers have investigated control and communication methods for autonomous guided vehicles. Problems such as coordination of multiple manipulators, motion planning and coordination of autonomous guided vehicles 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 guided vehicles was based on a single robot interacting with its environment. There is an increasing interest in multiple autonomous guided vehicles 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, using a hierarchical (central) controller may control multiple mobile robots. 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 Autonomous 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.

1.1 Objective

The objective of this study is to design the motion control for an autonomous guided vehicle (Bearcat II) built for the Association for Unmanned Vehicle Systems (AUVS) 1998 Competition, with a very interesting phenomenon, the Zero Turning Radius (ZTR) feature. The competition was held on June 1998. This thesis describes the design, development and exploratory research on the autonomous guided vehicle with zero turning radius feature. The motion control of the AGV designed has the capability of turning about the center of its drive axis, which is called the zero turning radius feature. It is gaining popularity and expanding commercially in the U.S. mowing market. They offer exceptional maneuverability and can make sharp turns possible with relatively greater ease than those without the ZTR feature. Rotating one wheel forward and the other wheel backward generally accomplishes the ZTR function. However in our design we instead vary the speeds of the left and right drive wheels while negotiating a curve. This enables the AGV to make a curved turning path parallel to the track lines.

1.2 System Design

The robot base is constructed from an 80/20 aluminum Industrial Erector Set. The AGV is driven and steered by two independent 36 volt, 12 amp motors. These motors drive the left and right drive wheel respectively through two independent gearboxes, which increase the motor torque by about twenty times. The power to each individual motor is delivered from a BDC 12 amplifier that amplifies the signal from the Galil DMC motion controller. To complete the control loops a position encoder is attached on the shaft of each of the drive motors. The encoder position signal is numerically differentiated to provide velocity feedback signal. There is a castor wheel in the rear of the vehicle, which is free to swing when the vehicle has to negotiate a turn.

The design objective was to obtain a stable control over the motion control with a good phase and gain margin and a fast unit step response. For this purpose a Galil 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 motor. The system was modeled in MATLAB using SIMULINK and the three parameters of the PID controller were selected using a simulation model to obtain the optimum response.

1.3 Background of Previous Research

An autonomous guided vehicle 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 [5], [6] 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.

The maximum speed of most AGVs is specified to be about 1 m/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. Designing a stable and smooth tracking controls becomes more difficult as the speed of the AGV increases. Consequently, it becomes important to understand the dynamic effects of various design parameters on the steering action of the autonomous guided vehicle.

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

Ozguner, et al. [7] described the steering control of a vehicle. They considered the design and stability analysis of a steering controller. The purpose 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. The ratio of the steering angle to the look-ahead offset was independent of the curve radius under reasonable approximations, during perfect lane tracking. That ratio was computed in terms of the vehicle speed and various vehicle parameters. A constant controller was designed to achieve that ratio at steady state. The controller was updated as a function of the vehicle speed. The look-ahead offset was the only information processed by the controller. It 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 processed by the controller, 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.

The mathematical block diagram of the proposed model was as shown in Figure 1.1 below.

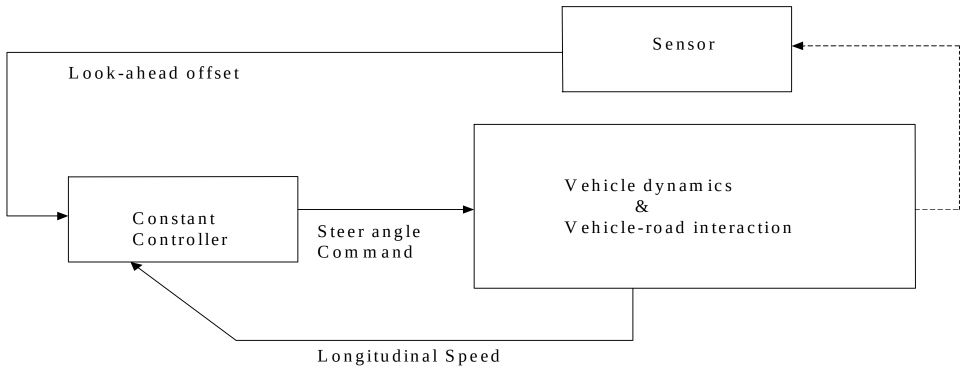


Figure 1.1: Lane following feedback structure (Ozguner, et al. [7])

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

1.4 Outline of Thesis

The remaining part of the thesis is made up as follows.

Chapter 2 deals with the concepts of motion control theory. It deals with the motion control architecture, the control problem, the description of input and output, feedback and feed forward control, linear and non-linear control, and the design process. Chapter 3 contains the introduction to zero turning radius features, its implementation and design in the Bearcat II robot, the control laws and the equations to find the speeds of the left and right drive wheels with the input of the distance error and the orientation error. The parameters for the motion controller for the system model is determined analytically and also tested manually on the real system using the Galil WSDK motion control design software kit. Chapter 4 contains a description of the Annual International Ground Robotics Competition, and the rules of the competition. Chapter 5 deals with the description of the Bearcat II robot, the structural design, system design and development, the vision guidance system, the obstacle avoidance system, motion control, safety and emergency stop features. Chapter 6 contains the results of the experimental research and Chapter 7 deals with the conclusions, some suggested modifications and recommendations.

Chapter 2

FUNDAMENTALS OF MOTION CONTROL

2.1 Introduction

This chapter presents an explanation of the motion control architecture and the basic control theory. Control systems are an integral part of modern society. It is found in almost every modern application. Control systems find widespread application in the steering of missiles, airplanes, 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 by hand so precisely would be impossible. At home we have control systems too. 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 processed as sound or picture. A control system keeps the laser beam positioned on the pits, which are cut as concentric circles on the disc. The heating system at home 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. This way the temperature at home could be regulated.

2.2 Motion Control Architectures

Subsystems and processes (or plants) in a control system are 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 2.1.

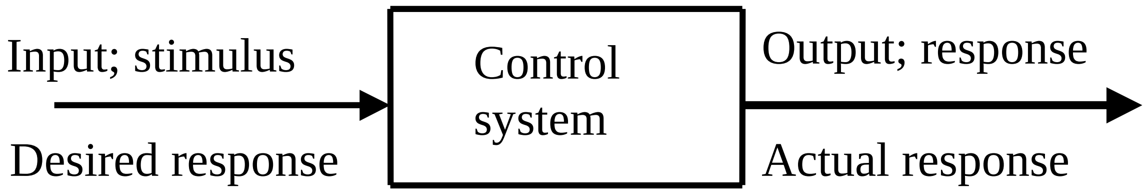


Figure 2.1: Simplified description of a control system (Nise, Norman [15])

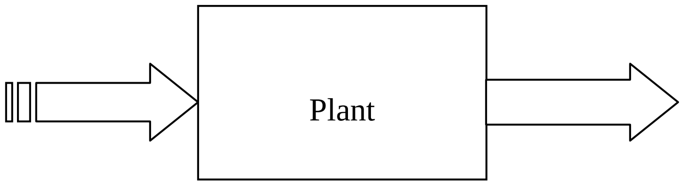
Motion control systems may operate in an open loop, closed loop non-servo, closed loop servo or a hybrid combination design as shown in Figure 2.2.

2.2.1 Open Loop Control System

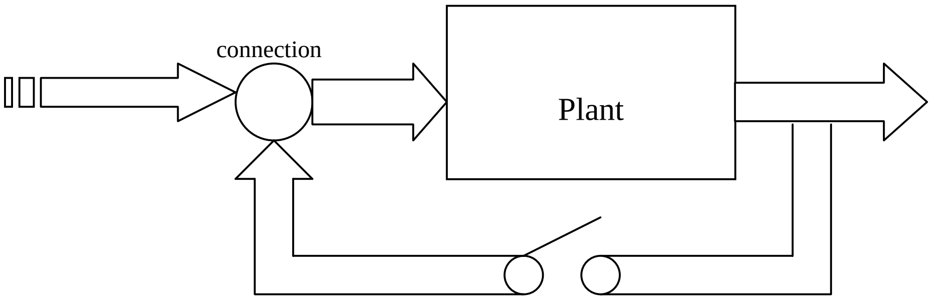
The open loop approach shown in Figure 2.2 (a) has input and an output but no measurement of the output for comparison with the desired response. Open loop systems are simply commanded by the input, as there is no measurement of the output. An example for an open loop system is a mechanical system consisting of a mass, spring, and a damper with a constant force positioning the mass. The greater the force, the greater the displacement which is the output. The system position will change with a disturbance such as an additional force, and the system will not detect or correct for the disturbance.

2.2.2 Closed Loop Control Systems

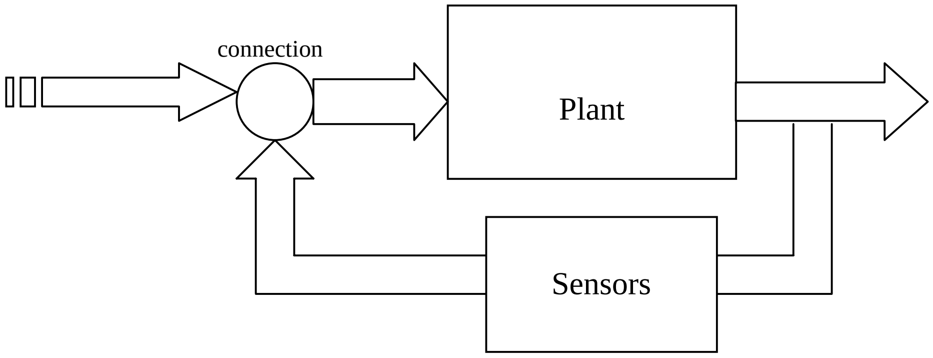
A non-servo, on-off or bang-bang control approach is shown in Figure 2.2 (b). The input signal turns the system on. When the output reaches a certain level, it closes a switch that turns the system off. A proportion or servo control approach is shown in Figure 2.2 (c). A measurement is made of the actual output signal, which is fed back and compared to the desired response. This approach will be studied in this thesis.



(a) Open loop system



(b) Non-servo system



(c) Servo system

Figure 2.2 Motion control systems may operate in several ways such as (a) open loop, (b) closed loop non-servo, or (c) closed loop servo

Closed-loop motion devices add position metrology to motorized actuation, eliminating some or all position uncertainties. Closed-loop versions of every conceivable type of electrical drive mechanism have been marketed. Fast-paced developments in motion microprocessors now present the controller designer with a wide array of affordable closed-loop positioning servo options. In the near future, cost-effective, general-purpose motion control chips will certainly begin to provide advanced capabilities presently found in only the most expensive controllers running proprietary digital-signal-processing microcode.

The components of a typical servo controlled motion control system may include an operator interface, motion control computer, control compensator, electronic drive amplifiers, actuator, sensors and transducers, and the necessary interconnections. The actuators may be powered by electro-mechanical, hydraulic or pneumatic or a combination of these power sources.

The operator interface may include a combination of switches, indicators and displays, including a computer keyboard and a monitor or display. The motion control computer generates command signals for a real time operation from a stored program. A special program in the motion control computer is the control compensator. Selecting the parameters of the compensator is often a critical element in the success of the overall system. The drive amplifiers and electronics must convert the low power level signals from the computer to the higher power signals required to drive the actuators. The sensors and transducers must make the measurement such as of position and velocity that are used for feedback to the controller. The actuators are the main drive devices which supply the force or torque required to move the load on a given motion profile. All of these subsystems must be interconnected in order to function properly.

2.2.3 Open- vs. Closed-Loop Motion Control

Open-loop systems provide electric actuation without position feedback. Most piezoelectric systems and inexpensive micrometer-replacement motorizers are of this type. Open-loop positioners are useful when remote control is desired for improved accessibility or to avoid disturbing critical components by touching them. Examples include fiber positioning, probing applications and microscopy positioning. "Open-loop" is by no means a synonym for "crude." Even inexpensive open-loop devices can achieve very fine incremental motions. Nanometer-scale incremental motions are achievable by open-loop piezo- and electrostrictive-type devices.

One can often infer the approximate position of a motion device without using an encoder. In the case of a piezo device, the applied voltage is a reasonably dependable indicator of position. The relation is imprecise due to hysteresis and non-linearities inherent in commonplace piezo materials. More recently developed electrostrictive materials operate in a similar manner with greatly reduced hysteresis.

Stepper and mini-stepper motors are examples of digital or impulse drives and move in discrete steps or rapid, quasi-continuous series of steps. The count of pulses sent to the device is a good predictor of its resulting position. However, the relationship between pulses and position can be unpredictable unless loads, accelerations and velocities are known to the designer and the motion system is set-up accordingly. In poorly engineered implementations, skipped or extra steps are frequent problems.

Advances in mini-stepping technology and incorporation of viscous motor-damping mechanisms have greatly improved the positioning dependability and vibration levels of today's highest-quality stepper devices when used open loop.

2.2.4 PID Control

The basic closed loop motion control system is illustrated in Figure 2.3. In any system, a command is input for a desired output. In this case, the desired output may be a position, velocity, or acceleration. The controller translates the command into the proper electrical signals and delivers them to the motion mechanics.

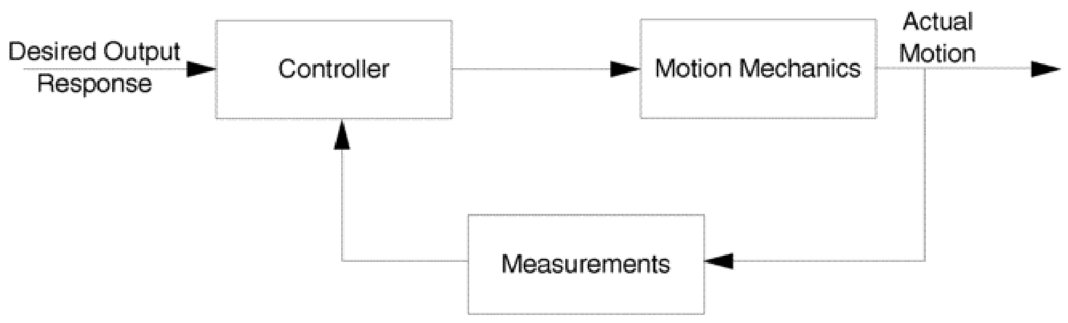


Figure 2.3. Block diagram for a basic motion control system

The mechanics, in turn respond with change in position, etc. which may or may not match the input signal to an acceptable degree. Measurements are made on the output or actual motion and fed back to the controller. The controller compares the actual to desired output and generates an error signal upon which corrective action is taken.

In the simplest systems, the error signal is amplified by a user specified gain constant and a corrective command is sent to the mechanics by the controller. This is called proportional control. Basically, the larger the gain coefficient K_p, the faster the error is corrected. Unfortunately, however, this simplicity has its limitations. As K_p is increased the mechanical system very quickly will begin to overshoot (overcorrect) and if it has low damping, it will oscillate, rendering the system unstable. Additionally, even stable values of K_p cannot completely eliminate errors, since as the error, e, approaches zero, the feedback term K_{pe} disappears. This results in some amount of steady state error whose magnitude is dependent on the friction and backlash in the system. This situation is represented in Figure 2.4.

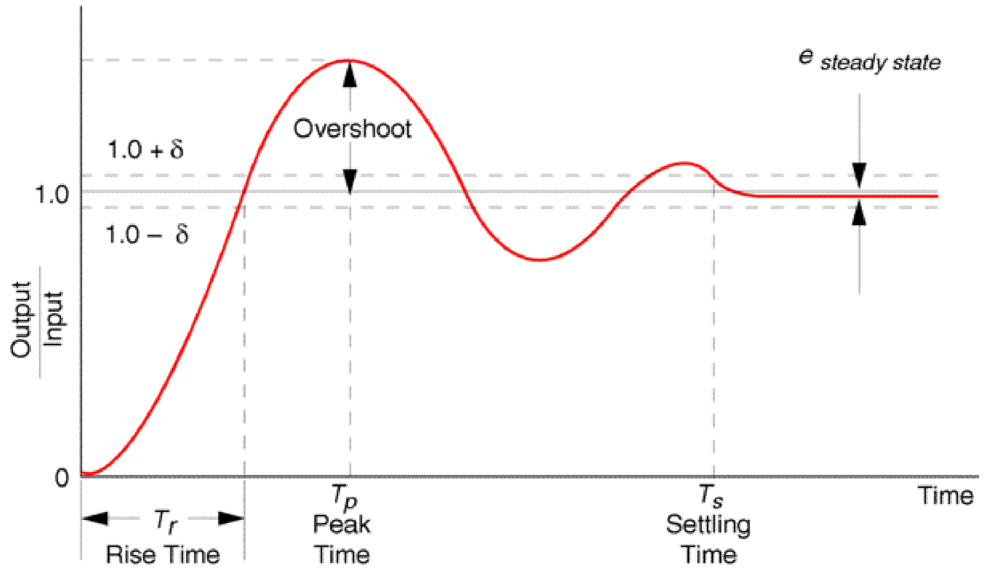


Figure 2.4. Response for a system using only proportional control leads to overshoot and non-zero steady state errors, [27].

To reduce the steady state errors in the system to an acceptable level, an additional operation must be performed on the error signal and fed back to the mechanics. Since the proportional error signal becomes ineffective as the error approaches zero, a technique, which uses the cumulative history of the error signal, was developed. By summing or integrating all past errors, a corrective signal can be generated even as the current error becomes small. By adjusting the coefficient of the integral term, K_i, an acceptable amount of steady state error can be specified.

Finally, the transient response can be made more stable by introducing an operation, which generates the derivative of the error signal. To avoid getting too deeply into control theory, one may think of this derivative term as a type of electronic damping in the system. By increasing K_d, the coefficient of the error derivative term, the transient response can be made more stable, but less fast.

The combination of Proportional plus Integral plus Derivative control is commonly called PID control. In equation form, the error compensation term generated by the controller looks like:

$$K_p e + K_i \int edt + K_d \frac{de}{dt}$$

Where e is the difference between the desired and measured output. Optimum system performance requires tuning the coefficients K_p, K_i, and K_d for the given combination of motion mechanics and payload inertias.

2.2.5 Drivetrain-Input Motion Metrology

A less expensive but often quite effective mechanical feedback implementation is to integrate a rotary encoder on a motion device's motor or leadscrew.

In many motion devices, the motor drives a leadscrew or worm via a gearbox. Typically, over 200 turns of the motor are reduced to a single turn of the leadscrew by miniaturized planetary gearboxes. This allows the use of compact and cost-effective motors. And when an encoder is mounted on a gearbox-equipped motor, the controller is provided with an enormous number of resolution-counts per turn of the leadscrew -- 20,000 encoder quadrature counts per millimeter of travel in one popular family of actuators. This yields a display resolution of 0.05 µm. However, be aware the motion capabilities of devices whose position loop is closed on the drivetrain input rather than output, fall short of what is implied by the display. This is really of concern only in critical applications where motions and repeatabilities on the order of the display's least significant digit are needed.

2.2.6 Correcting Errors Downstream from the Encoder

Drivetrain errors downstream from the encoder -- such as backlash and leadscrew pitch errors -- are non-zero in all but the most expensive devices based on non-contact direct motion metrology. While these errors, by definition, cannot be seen by the encoder, they tend to be repeatable, and some systems can be set up to estimate and correct for them, thereby improving performance. Used with a device of good mechanical quality, backlash- and leadscrew-pitch-compensation technologies can provide submicron bi-directional repeatability and incremental motion and micron-scale absolute accuracy.

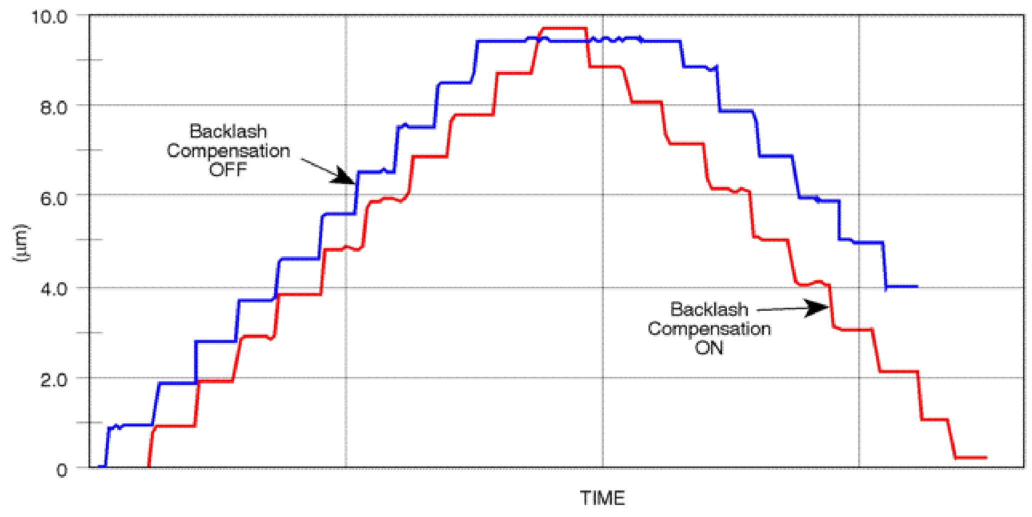
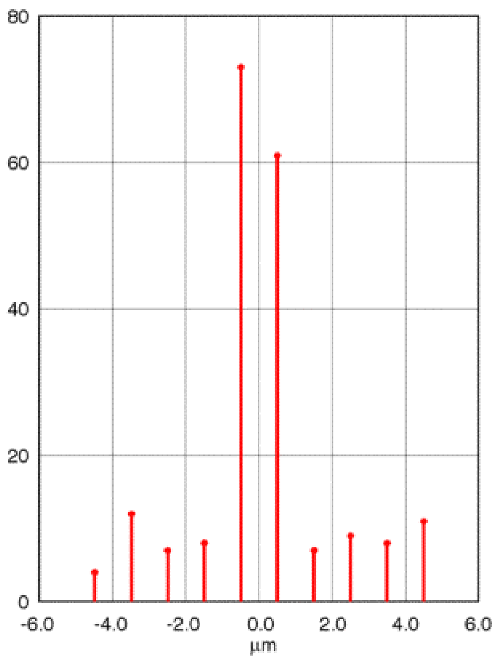


Figure 2.5. Interferometric stairstep-plots of ten successive 1 μm increments and ten 1 μm returns, with and without backlash-compensation. This capability is seen to virtually eliminate the actuator's ~4 μm backlash without causing overshoot, [27].

The best backlash compensation techniques eliminate backlash while introducing negligible over- or undershoot. The benefits of backlash compensation can easily be seen in Figure 2.5 by composing a stair-step plot using an interferometer or other independent metrology instrument: repetitively jog the device a few microns forward N times, then reverse N times. The resulting non-repeatability (undershoot) is generally due to looseness or stiction in the device. Once determined, some controllers can be programmed to eliminate most of it. Figure 2.6 represents this test over a large number of trials.



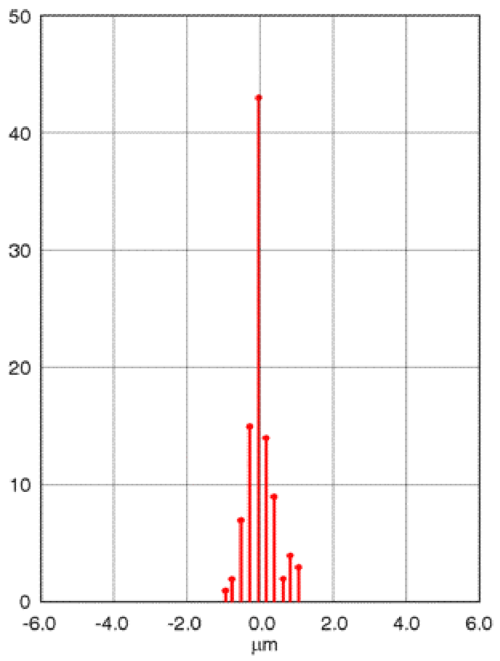


Figure 2.6. Interferometric bi-directional repeatability histograms (N=100) for random motions of a Newport 850B actuator. Left: automatic backlash compensation disabled; right, enabled, [27].

Leadscrew pitch error is generally dominated by a small (typically <1%) departure from the manufacturer's specified pitch. This "DC" pitch error can be measured without difficulty and -- along with other monotonic errors such as cosine error due to misalignment -- can be corrected by some closed-loop controllers for improved absolute accuracy.

2.2.7 Direct Output Motion Metrology

For higher performance, one needs to monitor the output position of the device, rather than its drivetrain input (e.g. motor or leadscrew position). Glass scale encoders can eliminate most of the backlash, wind-up, hysteresis, reversal error, leadscrew pitch errors, etc.

A hysteresis-free, non-contacting glass scale encoder mounted directly on the drivetrain output (linear or rotary stage platform or actuator shaft) can provide speed ranges in excess of six orders of magnitude with ±1% RMS velocity regulation. The controller continuously monitors the actual drivetrain output position, correcting it if necessary, in real time to compensate for drift, shifting loads, and other sources of error. Standard bi-directional repeatability, minimum incremental motion, position-holding and display resolution is 0.1 µm for linear stages and 1 arcsecond for rotary stages. Optional performance to 0.025 µm (linear) or 0.5 arcsecond (rotary) is available. The absolute accuracy is determined by the accuracy of the glass scale -- better than ±1 µm/100 mm for linear devices and 20 arcseconds for rotation stages. Even higher accuracies are available on special order.

2.2.8 Velocity regulation

Good speed regulation allows data acquisition or material processing to be performed on the fly. This can be critical in high-throughput work like microscopy, semiconductor inspection or micro-ablation, where stopping the stage at each point would be impracticably time-consuming. Clocking methods range from intervalometry to advanced real-time position interfacing.

Velocity servos are found in even the least expensive open- or closed-loop controllers. They can range from simple analog servos based on current or back-EMF feedback, to digital servos embodied in motion control microprocessor chips, to overlapping multi-servo implementations of motor, tachometer and encoder feedback. Each technique is suited to particular velocity regimes, and the finest multi-servo systems smoothly switch between their various feedback loops as appropriate.

Velocity is typically regulated to within 10-15% for open loop systems and 5-10% for closed-loop. Regulation is generally best at higher speeds. The finest systems offer regulation to 1% RMS or better through their speed range.

Velocity measurement is non-trivial since time and position must be measured simultaneously and to similar precision. One cannot simply poll an interferometer using a software intervalometer, since computers service their internal peripherals unpredictably.

Some interferometers, LVDTs, etc. have analog or digital real-time interfaces or built-in intervalometers that can be used to calculate velocity. A few instruments measure velocity directly -- notably laser Doppler velocimeters/displacement meters.

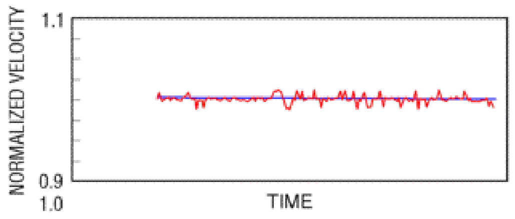


Figure 2.7a: Normalized Velocity Measurements of PM500-1A driving a 462 stage through its full travel (500 µm/sec). Superimposed is the best-fit trend-line. Histogram showed directly below., [27].

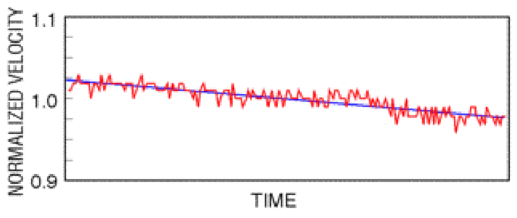
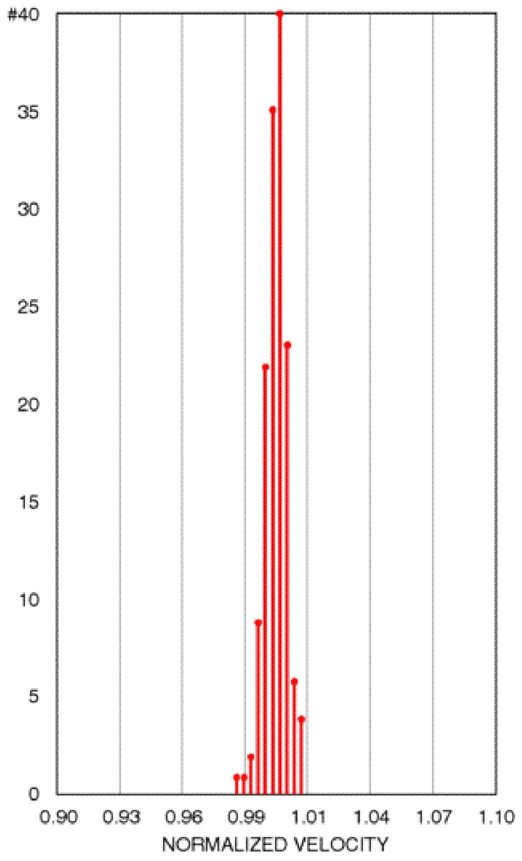
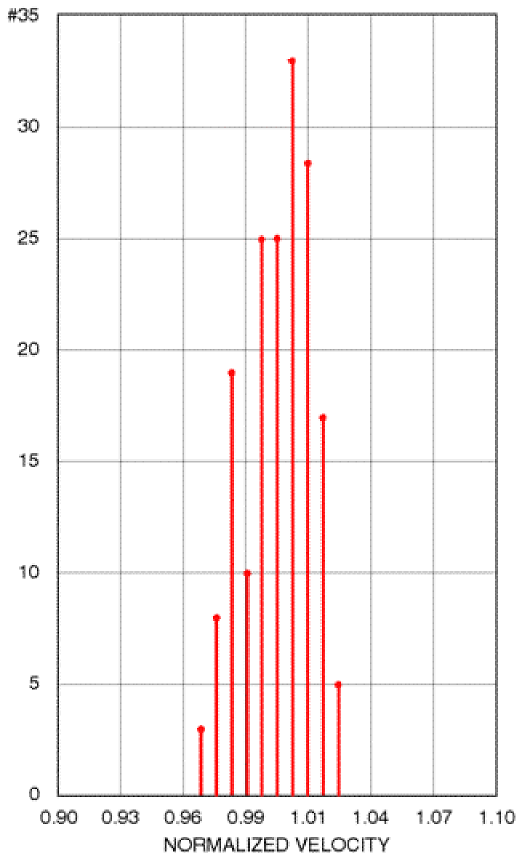


Fig 2.7b: Normalized velocity of 850B actuator. Regulation is good; some loading is seen as the 462 stage's springs are extended. (500 µm/sec). Histogram shown directly below., [27].





2.3 The Control Problem

A generic closed-loop system consists of :

- input; reference input gives the desired output
- controller
- plant, a system to be controlled by the controller
- 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 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.

2.4 Description of the Input and output

The input of a system represents a desired response and the output is the actual response. For example, when the fourth-floor button of an elevator is pushed on the ground floor of a building, the elevator rises to fourth floor with a speed and floor-leveling accuracy designed for passenger comfort. Figure 2.8 shows the input and output for the elevator system.

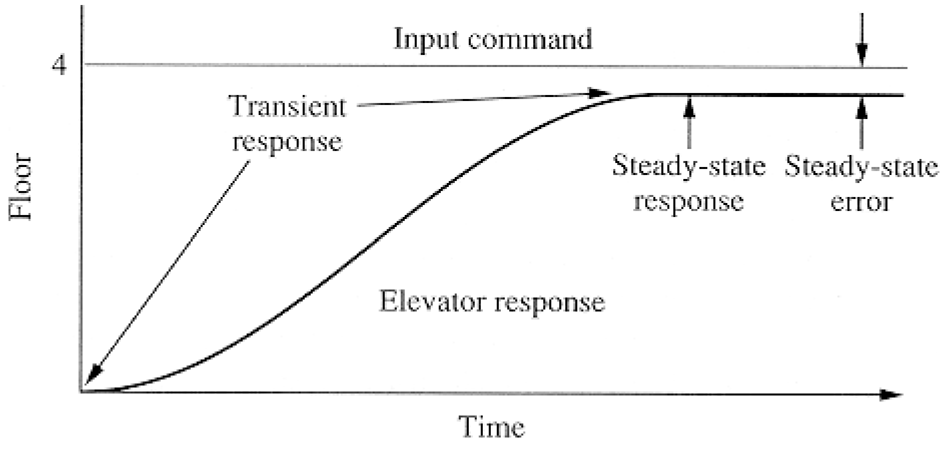


Figure 2.8: Input output for the elevator system (Nise, Norman [15])

The push of the fourth floor button of the elevator is the input and is represented by a step command. The input represents what the desired output should be after the elevator has stopped and 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 2.8 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, which 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.

2.5 Feedback and Feedforward control

Feedback Control is an error-driven strategy of a system and 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 proportional 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 for the system. For example,

$$r(t) = K_p(x_{d(t)} – x (t)) + K_v (d x(t)/dt); \qquad (2.1)$$

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 dx(t)/dt the rate of change of the state of the system.

The damping coefficient, K_p, 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 (2.1) is not quite right and 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:

$$r(t) = K_p(x_{d(t)} – x (t)) + K_v (d x(t)/dt) + K_I \int (x_{d(t)} – x (t))\, dt \qquad (2.2)$$

Where K_I is the 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 as described earlier, where PID stands for proportional, integrative, derivative.

Feedback control is an error-driven process and 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 be corrected. It should also be noted that feedback control might not be limited to the position of the system alone 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 of the system.

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 of the system. 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

$$r(t) = R^{-1}(x_d(t)) \qquad (2.3)$$

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,

$$\theta(t) = R(R^{\wedge-1}(x_d(t))) \qquad (2.4)$$

There could be several reasons for feedforward control to be unsatisfactory. It may include the following:

- 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 increasing function of time (due to non-linearities or non- stationarities in the dynamics of the system).
- Even assuming that our model is 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 advantageous to contemplate 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 (output) 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 concept 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 and 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 complex problem, as the combined system is now time-variant.

2.6 Linear control systems

In most physical systems, we have non-linear elements, but in some circumstances it may be possible to deal with them as linear elements. Then the structure 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 forms 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 remarkable.

2.7 Non-linear control systems

Non-linear systems are much more difficult to control than linear systems. This is 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 be treated on a case by case basis.

Controlling such a system is going to be difficult obviously, as we can make no claims about the response of the system to a input. Except for a few distinct 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.

Biological control systems tend to be rather small, so it seems illogical 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.

2.8 The design process

The control system design, involves a step by step process. In this section an 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 2.9 shows the diagram of the system and the Figure 2.10 shows the functional block diagram of the system. The functions are shown above the blocks, and the required hardware is indicated inside the blocks.

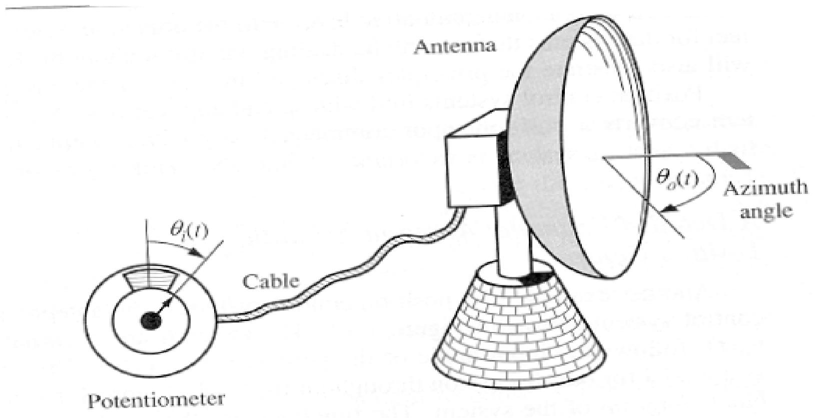


Figure 2.9: An antenna azimuth position control system (Nise, Norman [15])

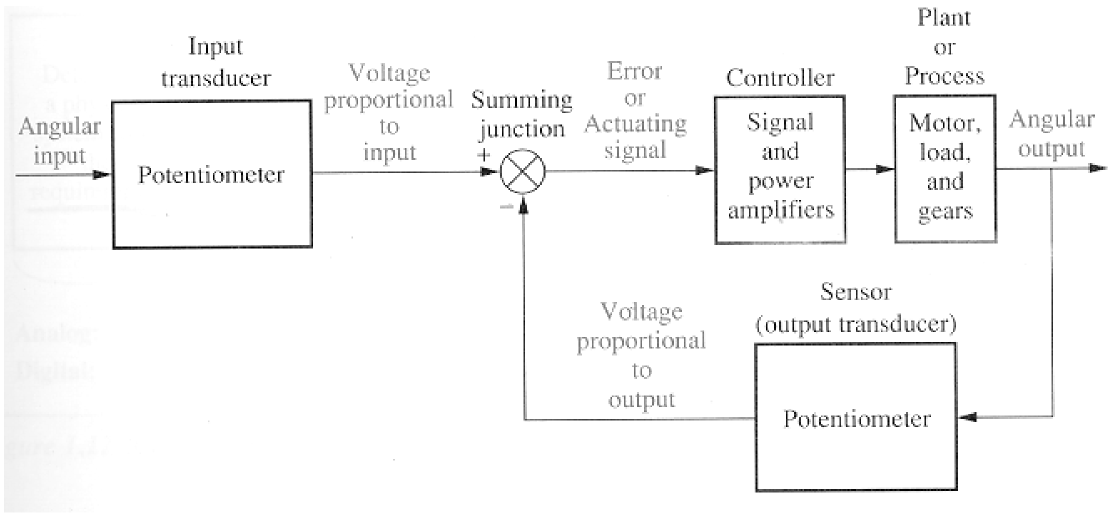

Input transducer
Angular input
Potentiometer
Voltage proportional to input
Summing junction
+
−
Error or Actuating signal
Controller
Signal and power amplifiers
Plant or Process
Motor, load, and gears
Angular output
Sensor (output transducer)
Potentiometer
Voltage proportional to output

Figure 2.10: Functional block diagram of a position control system (Nise, Norman [15])

Step 1 : Transformation of the requirements into a physical system

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

Requirements: The requirement is 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

This 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 the block diagram indicates functions as input transducer, the controller, and relevant descriptions of the amplifiers and the motors.

Step 3 : Creation of the schematic

This is the process of converting the physical system into a schematic diagram.
Relevant approximations of the system should be made and certain phenomena should be neglected to make it easier 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 abruptly 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 insignificant for the DC motor.

Load: The load consists of a 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 : Development of a mathematical model (block diagram)

From the schematic, the physical laws such as the Kirchhoff's laws and Newton's laws are used with modifying assumptions to develop a mathematical model.

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: The 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. The Laplace transform can be used for linear systems, but yields more intuitive information than the differential equations. The ability to change system parameters and rapidly sense the effect of these changes on the system response.

Model 3: State space methods: The 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 : Reduction of 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 notation that represents the system from its input to its output.

Step 6: Analysis and design

The system is analyzed 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 preferred performance.

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

Chapter 3

Theory of Zero Turning Radius and the Motion Control System Model

3.1 Introduction

The motion control of the AGV designed has the capability of turning about the center of its drive axis, which is called the zero turning radius features. It is gaining popularity and expanding commercially in the U.S. mowing market. They offer exceptional maneuverability and can make sharp turns possible with relatively greater ease than those without the ZTR Features. Rotating one wheel forward and the other wheel backward generally accomplish the ZTR Function. However in our design we instead vary the speeds of the left and right drive wheels while negotiating a curve. This enables the AGV to make a curved turning parallel to the track lines.

3.2 Design

The robot base is constructed from an 80/20 aluminum Industrial Erector Set. The AGV is driven and steered by two independent 36 volt, 12 amp motors. These motors drive the left and right drive wheel respectively through two independent gearboxes, which increase the motor torque by about twenty times. The power to each individual motor is delivered from a BDC 12 amplifier that amplifies the signal from the Galil DMC motion controller. To complete the control loops, a position encoder is mounted on each of the drive motors. There is a castor wheel in the rear of the vehicle, which is free to swing when the vehicle has to negotiate a turn. The encoder position signal is numerically differentiated to provide velocity feedback signal. Control of the AGV motion is done by the usage of differential speed drive wheel. These are drive wheels whose speeds can be varied according to the change in the direction of the track being followed. The Figure 3.1 illustrates the design of the system.

3.3 Control Law

The control of the vehicle movement in a two dimensional space can be reduced to the control of two variables: the instantaneous speed of the vehicle axis middle point and the attitude or orientation of the steering angle of the vehicle.

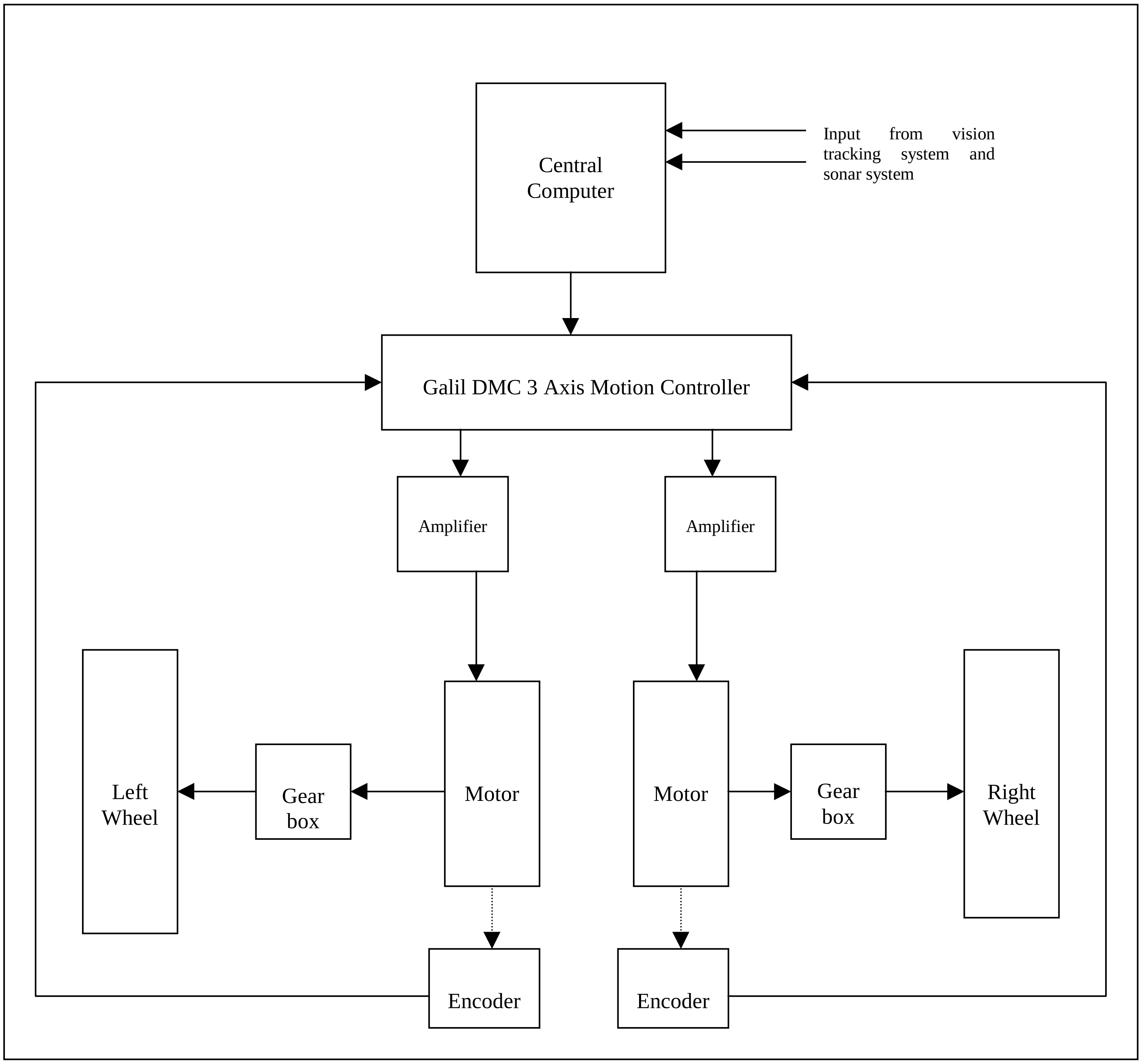


Figure 3.1: Design of the system

The instantaneous speed of the vehicle axis middle point may be described as

$$V_m = ds / dt \qquad 3.1$$

This velocity may also be expressed in terms of the velocities of the left and right wheels as:

$$V_m = (V_L + V_R) / 2 \qquad 3.2$$

Where L and R refer to the left and right drive wheel velocities, respectively. If the vehicle is going straight, these velocities will be equal to the middle point velocity. If the vehicle is turning about an instant center of rotation, these velocities will be different.
The orientation angle is also related to the left and right velocities by:

$$d\theta /dt = (V_L - V_R) / W \qquad 3.3$$

Where W is the width of the axle or distance between the wheels. Again if the vehicle is moving along a straight line the left and right velocities will be equal and the steering angle will be zero. The most common methods of turning a wheeled vehicle are the steering wheel or with differential left and right wheel drives. In the design presented we have a differential left and right wheel drive. The axis middle point is assumed to be constant and controlled by an independent control system.

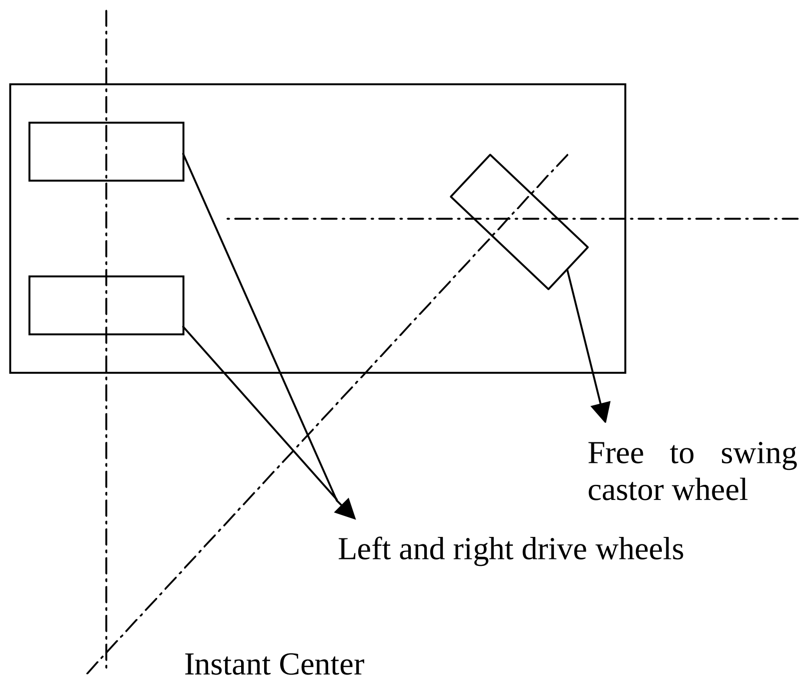


Figure 3.2: AGV with differential wheel drive

Controlling the sum and difference of the two wheel speeds does the control of the vehicle.

$$V_L + V_R = 2 \cdot V_m \qquad 3.4$$

$$V_L - V_R = W \cdot (d\theta/dt) \qquad 3.5$$

Or

$$\omega_L + \omega_R = (V_m / \pi R) \qquad 3.6$$

$$\omega_L - \omega_R = (W / 2\pi R) . (d\theta / dt) \qquad 3.7$$

Here speed V_m will be maintained constant by an independent control loop, the input of which is received from a central computer. Performances of this loop are not critical. Control acts on the sum of the voltages applied to the motors which directly controls the speeds of the drive wheels. The attitude control acts on the difference of these two voltages.

3.4 Control Problem

The inputs to the central computer are V_m and ΔV.

Where V_m is the speed for the middle point of the wheel shaft and

ΔV is the difference between the two wheel speeds.

The outputs are V_L and V_R, the speeds of the two wheels respectively.

We have

$$\omega_L + \omega_R = (V_m / \pi R) \qquad 3.8$$

$$\omega_L - \omega_R = (W / 2\pi R) . (d\theta / dt) \qquad 3.9$$

Solution:

$$\omega_L = (2V_m + \Delta V) / 2\pi R) \qquad 3.10$$

$$\omega_R = (2V_m - \Delta V) / 2\pi R) \qquad 3.11$$

Or

$$V_L = V_m + (\Delta V / 2) \qquad 3.12$$

$$V_R = V_m - (\Delta V / 2) \qquad 3.13$$

To keep a straight line tracking two error variables have been defined namely the distance error Y and the orientation error θ. The distance error is the perpendicular distance from the vehicle cenroid and the track center. The orientation error is the angle enclosed by a tangent drawn to the track line at the position of the vehicle and the direction to which the vehicle is heading or the angle enclosed between two consecutive co-ordinates picked by the vision tracking system of the vehicle.

Two different cases have been presented to find ΔV, the difference between the velocities of the left and right drive wheels of the vehicle, from which the individual velocities of the left and right drive wheels is calculated.

Case I

$$\Delta V = K_1 . Y + K_2 . \theta \qquad 3.14$$

Where Y is the distance error signal

θ is the attitude or orientation error signal and,

K_1 and K_2 are the control coefficients.

In a simple case but not optimal case

$$K_1 = 1 \qquad 3.15$$

$$K_2 = 2 . (WV_{max})^{1/2} \qquad 3.16$$

Where V_{max} is the maximum middle point velocity which can be reached by the vehicle, which was found to be 1.347 m/h at maximum voltage delivered by the amplifier to the drive motors.

Y and θ are obtained from the vision code of the vision guidance algorithm.
ΔV is calculated by substituting the values of the K_1, K_2, Y and θ in the equation. From the value of ΔV, the individual velocities of the left and right drive wheel are calculated.

Case II

In case II, a clock function is being used to find dt, the time interval, between two consecutive co-ordinates from the vision code of the vision guidance system of the vehicle. dθ is found from the two consecutive x coordinates got from the vision guidance system.

Substituting these values of dt and dθ, ΔV or ($V_L - V_R$) is calculated from:

$$d\theta / dt = (V_L - V_R) / W \qquad 3.17$$

Both these cases where tested on the Bearcat II Autonomous Guided Vehicle. Bearcat II Autonomous Guided Vehicle in the Case II followed the track lines closely than in Case I as it was for a simple case but not an optimal case.

3.5 Motion Control System Model

Selecting the right parameters for the PID controller is the most important for any motion control system. The motion control system of the AGV helps maneuver it to negotiate curves and drive around obstacles on the course. Designing a PID controller for the drive motor feedback system of Bearcat II Robot, the autonomous unmanned vehicle was therefore considered one important step for its success.

The wheels of the vehicle are driven independently by two Electrocraft brush-type DC servomotors. Encoders provide position feedback for the system. The two drive motor systems are operated in current loops in parallel using Galil MSA 12-80 amplifiers. The main controller card is the Galil DMC 1030 motion control board and is controlled through a computer.

3.5.1 System Modeling

The position-controlled system comprises a position servomotor with (Electrocraft Brush type DC motor) an Encoder, a PID controller (Galil DMC 1030 motion control board) and an amplifier (Galil MSA 12-80).

Modeling the Amplifier 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 mode of the system.

a. Voltage Loop

In this 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

$$\frac{P}{V} = \frac{K_v}{[K_t s(s\tau_m + 1)(s\tau_e + 1)]}$$

where,

$$\tau_m = \frac{RJ}{K_t^2}\ (s)\quad and \qquad \tau_e = \frac{L}{R}(s)$$

The motor parameters and the units are:

K_t : Torque constant (Nm/A)

R : Armature resistance

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

L : Armature Inductance

b. Current Loop

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

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

Where,

$K_{a-=}$ Amplifier gain

K_t and J are as defined earlier

c. 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 input. This is given by

$$\frac{\omega}{V} = \frac{\dfrac{K_a K_t}{J_s}}{1 + \dfrac{K_a K_t K_g}{J_s}} = \frac{1}{[K_g (s\tau_1 + 1)}$$

$$\textit{where,}$$

$$\tau_1 = \frac{J}{K_a K_t K_g} \qquad \textit{and therefore}$$

$$\frac{P}{V} = \frac{1}{[K_g s(s\tau_1 + 1)]}$$

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

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

The Controller

The controller in the Galil DMC 1030 board has three elements, namely the Digital-to-Analog Converter (DAC), the Digital Filter and the Zero Order Hold (ZOH).

a. Digital-to-Analog Converter (DAC)

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

For the DMC 1030, the DAC gain is given by K_d=0.0012 [V/count]

b. Digital Filter

The digital filter has a discrete system transfer function given by

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

The filter parameters are K, A and C. These are selected by commands KP, KI and KD, where KP, KI and KD are respectively the Proportional, Integral and Derivative gains of the PID controller.

The two sets of parameters for the DMC 1030 are related according to the equations,

$$K = K_p + K_d$$

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

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

c. 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,

$$H(s) = \frac{1}{\left(1 + s\frac{T}{2}\right)}$$

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

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

3.5.2 System Analysis

The system transfer functions are determined by computing transfer functions of the various components.

- Motor and the Amplifier

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

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

- Encoder

The encoder on the DC motor has a resolution of 500 lines per revolution. Since this is in quadrature, the position resolution is given by 4*500=2000 counts per revolution.

The encoder can be represented by a gain of

$$K_f = \frac{4 \times N}{2\pi} = \frac{2000}{2\pi} = 318$$

- DAC

From the Galil manual, the gain of the DAC on the DMC 1030 is represented as

K_d = 0.0012 V/count

- ZOH

The ZOH transfer function is given by

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

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

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

3.5.3 System Compensation Objective

The analytical system design is aimed at closing the loop at a crossover frequency ω. This crossover frequency is required to be greater than 200 rad/sec. An existing system is taken as a reference and the crossover frequency of that system is used since the two are similar.

The following are the parameters of the system:

1	Time Constant of the motor	K_t	2.98 lb-in/amp	0.3375N-m/amp
2	Moment of Inertia of the system	J	2.2e02 lb-in^2 (approx.)	2.54e04 Kg-m^2 (approx)
3	Motor resistance	R	0.42 ohms	
4	Amplifier Gain in current loop	K_a	1.2 amps/volt	
5	Encoder gain	K_f	318 counts/rev	

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

The block diagram of the system is shown in Figure 3.3 on page 61.

Motor:

$$M(s) = \frac{K_t}{Js^2} = \frac{0.3375}{2.54 \times 10^{-4}} = \frac{1330}{s^2}$$

- Amplifier

$$K_a = 1.2$$

- DAC

$$K_d = \frac{10}{8192} = 0.0012$$

- Encoder

$$K_f = 318$$

- ZOH

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

- Compensation filter

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

$$L(s) = M(s)K_a K_f K_d H(s)$$
$$= \frac{1.21 \times 10^6}{s^2 (s + 2000)}$$

The feed-forward transfer function of the system is given by

$$A(s) = L(s)G(s)$$

And the open loop transfer function will be

$$L(j200) = \frac{1.21\times 10^{6}}{\left(j200\right)^{2}\left(j200+2000\right)}$$

The magnitude of *L(s*) at the crossover frequency of 200 rad/sec is:

$$\left|L\left(j200\right)\right| = 0.015$$

And the phase of the open loop transfer function is given by:

$$ArgL\left(j200\right) = -180 - \tan^{-1}\left(\frac{200}{2000}\right) = -185^{\circ}$$

G(s) is selected such that A(s) has a crossover frequency of 200 rad/sec and a phase margin of 45 degrees. This requires that

$$\begin{aligned} &\left|A(s)\right| = 1 \\ &\mathit{and} \\ &Arg\left[A\left(j200\right)\right] = -135^{\circ} \end{aligned}$$

But we have A(s)=L(s)G(s)

Therefore we must have,

$$\left|G(j200)\right| = \frac{\left|A(j200)\right|}{\left|L(j200)\right|} \approx 66$$

and

$$Arg[G(j200)] = Arg[A(j200)] - Arg[L(j200)] = -135^\circ + 185^\circ = 50^\circ$$

Hence select the filter function of the form

G(s)=P+sD;

such that at crossover frequency of 200, it would have a magnitude of 66 and a phase of 50 degrees.

$$\left|G(j200)\right| = \left|P + (j200D)\right| = 66$$

and

$$Arg\left[G(j200)\right] = \tan-1\left[\frac{200D}{P}\right] = 50^\circ$$

Solving these equations, we get,

P=42;

D=0.25

The filter transfer function is given by G(s)0.25s+42.

3.5.4 System Analysis with Compensator

Now with the filter parameters known, the open loop and closed loop transfer functions are computed as follows:

$$OLTF = \frac{9.62s^3 + 2572s^2 + 1.885*10^5 s + 4.104*10^6}{s^5 + 400s^4 + 47500s^3 + 1.5*10^6 s^2}$$

The root locus and Bode plot for the system is shown in the following Figures 3.5, 3.6 and it is clear that the system is not stable in the closed loop because it has two poles at the origin. This has to be further compensated by a controller in order to stabilize the closed loop.

A controller with zeros that can cancel the poles at the origin is used. Poles are added at s=-50 and s=-150 in order to stabilize the closed loop step response.

The controller transfer function is given by

$$G(s) = \frac{s^2}{(s+50)(s+150)}$$

With the controller, the open and closed loop transfer functions are given by

$$OLTF = \frac{30.45\times10^3 s + 51.15\times10^6}{s^3 + 2200s^2 + 407500s + 15\times10^6}$$

and

$$CLTF = \frac{957.6s + 160876}{s^3 + 2200s^2 + 408457s + 15.16\times10^6}$$

The frequency step response plots of the system are shown in the figures 3.4.

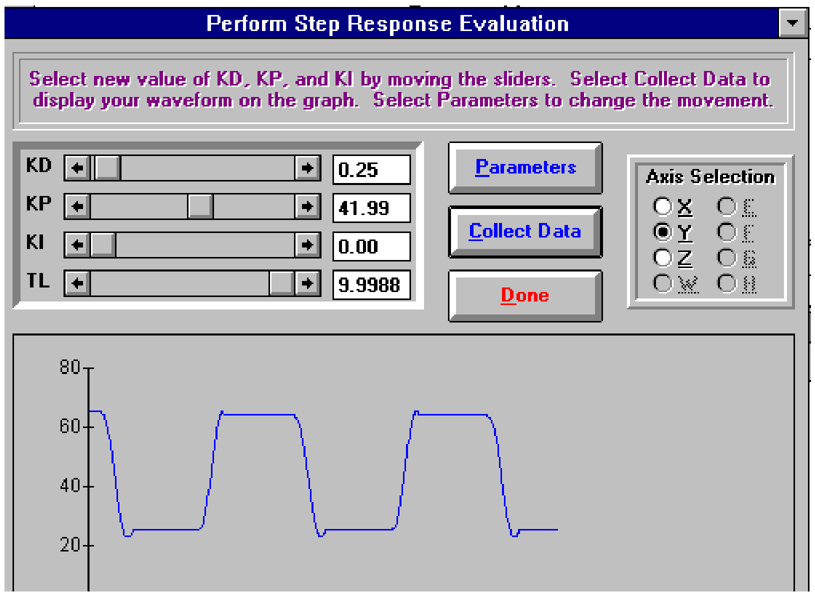
Figure 3.3. Block diagram of the matlab simulink model of the control system

Figure 3.4. Step response of the compensated system

Figure 3.5. Rlocus plot of the compensated system

Figure 3.6. Bode plots of the compensated system

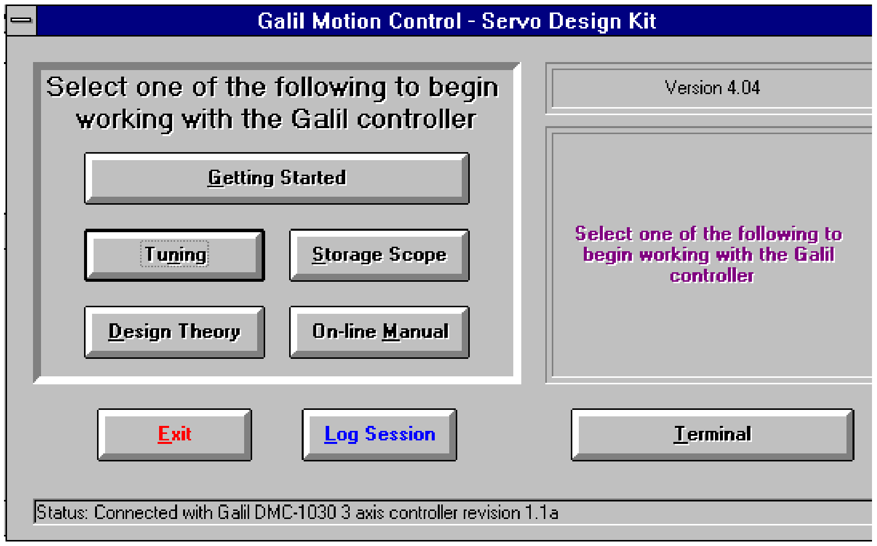
The analytical values of K_p, K_i and K_d which are the proportional, integral and derivative gains, respectively, of the PID controller, are tested for stability in the real system with the help of Galil Motion Control - Servo Design Kit Version 4.04. The step response plot is shown in Figure 3.7 below. Figures 3.8 and 3.9 shown in the following page are the start-up menu and tuning methods menu of the Galil Motion Control - Servo Design Kit Version 4.04.


Perform Step Response Evaluation

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.

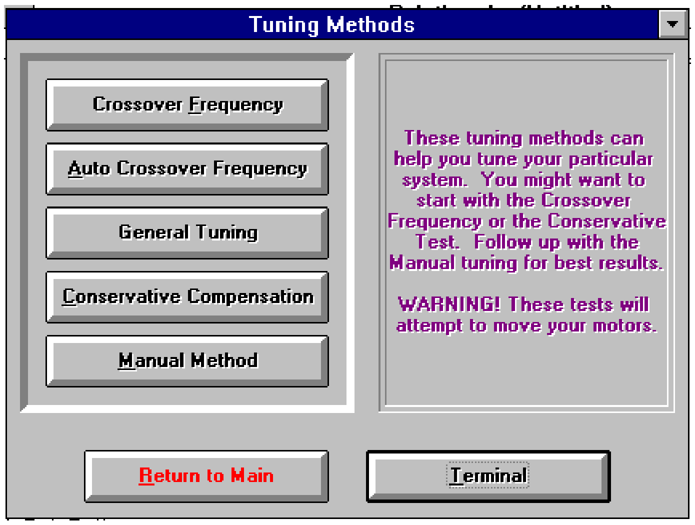
KD 0.25
KP 41.99
KI 0.00
TL 9.9988

Figure 3.7. Step Response Plot from the Galil servo design kit.


Galil Motion Control - Servo Design Kit
Select one of the following to begin working with the Galil controller
Getting Started
Tuning
Storage Scope
Design Theory
On-line Manual
Version 4.04
Select one of the following to begin working with the Galil controller
Exit
Log Session
Terminal

Status: Connected with Galil DMC-1030 3 axis controller revision 1.1a

Figure 3.8. Start-up menu of Galil motion control - servo design kit, Ver. 4.04



These tuning methods can help you tune your particular system. You might want to start with the Crossover Frequency or the Conservative Test. Follow up with the Manual tuning for best results.

WARNING! These tests will attempt to move your motors.

Figure 3.9. Tuning methods menu of the Galil motion control - servo design kit, Ver. 4.04

Chapter 4

THE AUVS COMPETITION

4.1 Introduction

This chapter reviews the Annual International Ground Robotics 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 straying 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. In 1998, most of the teams had a faster and smaller robot with better maneuverability than the previous year but no one completed the course. The AGV’s had to be faster to finish the track in a lesser time.

4.2 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, results in elimination.
4. The vehicles are not 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 safety hazard or if it violates the safety requirements, they can disqualify the vehicle. Emergency stops on the vehicle and the remote stops are tested as a measure of safety and as a qualification for the competition. The maximum speed is also tested on a special track.

Chapter 5

The Bearcat II Robot

5.1 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 II Robot. The Bearcat I 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. The Bearcat II Robot weighs approximately 450 lbs., and is 2 feet wide and 4 feet long. In the 1998 competition, the track was 10 feet wide. This meant that the space left for maneuvering was 4 feet on either side of the robot, not taking into account any obstacles. The necessity to build this new Bearcat II Robot was because of the increasing difficult competition rules. Had the organizers placed the obstacles more towards the center of the track in the 1997 competition, 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, the Bearcat II Robot was built smaller and also with the Zero Turning Radius Feature which greatly improved the maneuverability of the vehicle.

The robot base is constructed from an 80/20 aluminum Industrial Erector Set. The AGV is driven and steered by two independent 36 volt, 12 amp motors. These motors drive the left and right drive wheel respectively through two independent gearboxes, which increase the motor torque by about twenty times. The power to each individual motor is delivered from a BDC 12 amplifier that amplifies the signal from the Galil DMC motion controller. To complete the control loops, a position encoder is mounted on each of the drive motor. There is a castor wheel in the front of the vehicle, which is free to swing when the vehicle has to negotiate a turn. The encoder position signal is numerically differentiated to provide velocity feedback signal.

5.2 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 , Motion control system, Obstacle avoidance system, Speed control, Safety and Emergency stop 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.

5.3 Vision guidance system

The aim 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.5.3 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 coefficient computation model is implemented to solve the vision problem.

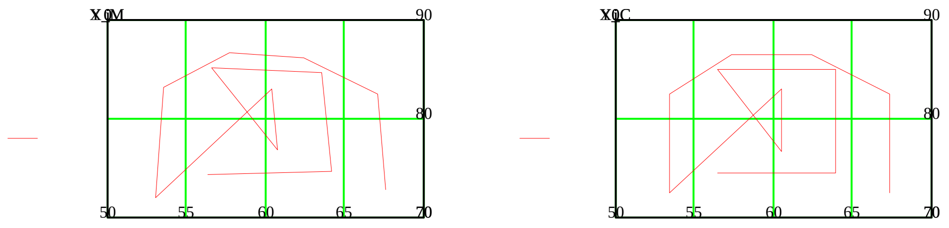


Figure 5.1 X and Y coordinates for the measured and computed vision calibration points

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 time point on a ground. Thus the vision algorithm computes the x and y coordinates 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			Image Coordinates		Computed Physical Coordinates	
	x	y	z	x	y	x	y
1	48.856	71.047	-22.533	265	341	48.464	70.745
2	56.346	71.054	-22.524	286	454	56.667	71.404
3	56.346	81.548	-22.564	342	329	55.949	81.358
4	48.831	81.583	-22.552	323	219	48.803	81.789
5	52.919	73.109	-19.057	344	377	53.33	73.475
6	52.87	79.537	-19.034	321	301	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	173	40.362	83.149

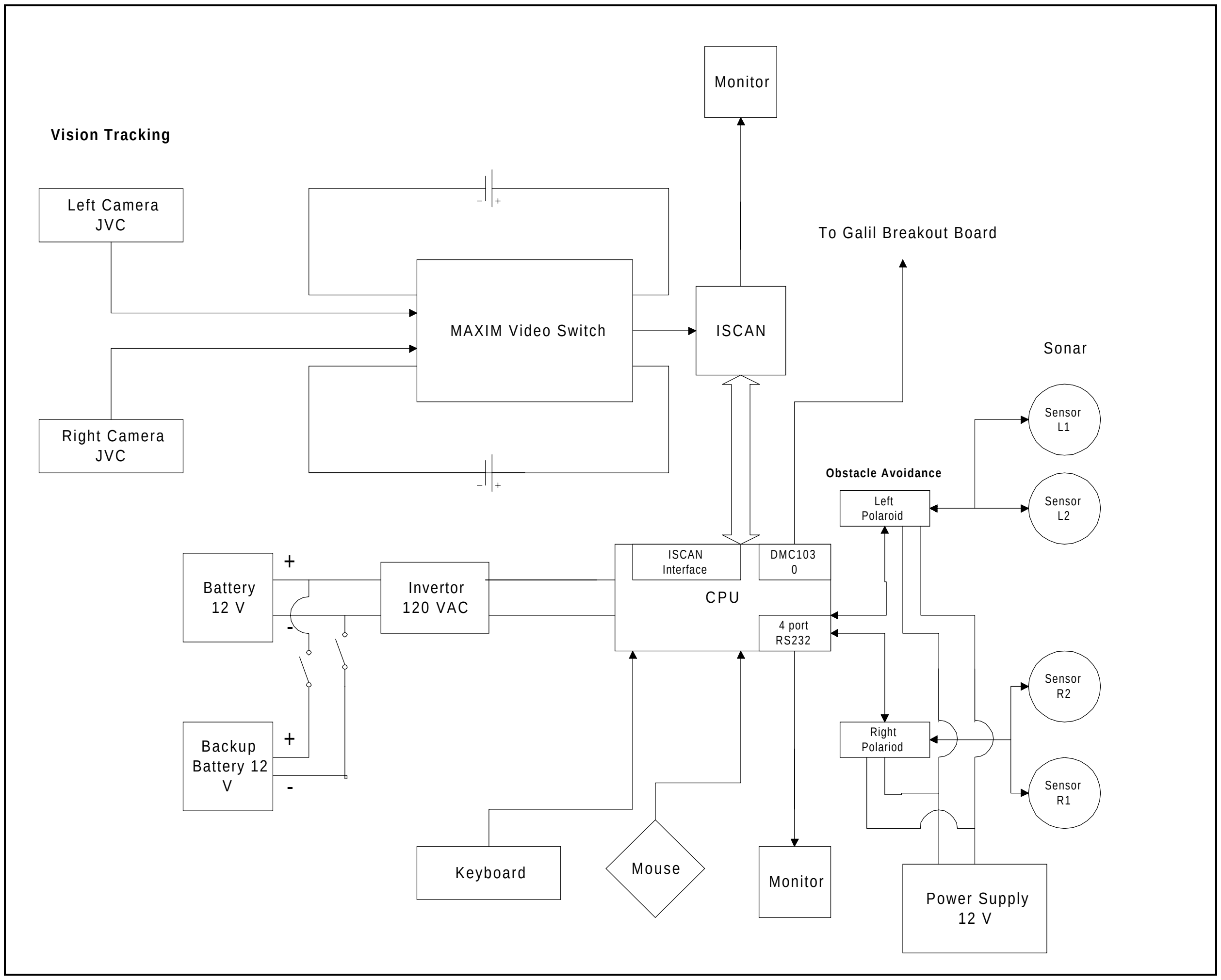


Figure 5.3: System block diagram

.

5.4 Obstacle avoidance system

The obstacle avoidance system consists of four ultrasonic transducers and a polaroid board. 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 are 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 towards 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. Since the speed of sound in air is known, 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 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.

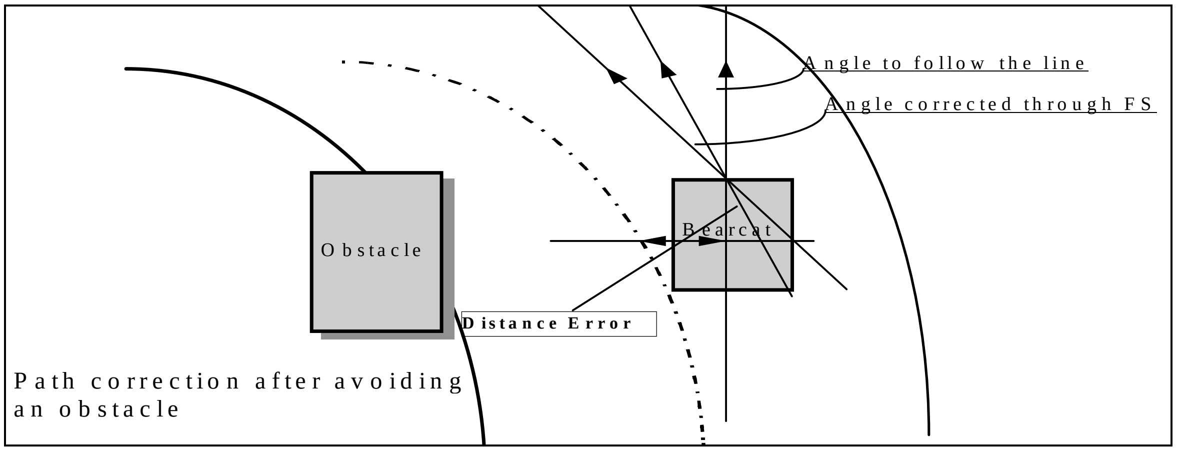

Angle to follow the line
Angle corrected through FS
Obstacle
Bearcat
Distance Error
Path correction after avoiding an obstacle

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.

5.5 Motion Control System

The motion control of the AGV designed, has the capability of Zero Turning Radius features which is gaining popularity and expanding commercially in the U.S. mowing market. Zero Turning Radius (ZTR) feature is the ability to make a turn about the center of the axis of the drive wheels. They offer exceptional maneuverability and can make sharp turns possible with relatively greater ease than those without the ZTR Features. Rotating one wheel forward and the other wheel backward generally accomplish the ZTR Function. However in our design we instead vary the speeds of the left and right drive wheels while negotiating a curve. This enables the AGV to make a curved turning parallel to the track lines.

Control of the AGV motion is done by the usage of differential speed drive wheel. These are drive wheels whose speed can be varied according to the change in the direction of the track being followed. This task can be reduced to the control of two variables:

1. The instantaneous speed of the vehicle, V_M
2. The orientation of the vehicle,θ

The velocity of the vehicle can be controlled by controlling the sum and difference of the two wheel speeds.

$V_L + V_R = 2V_M$

$V_L – V_R = W.(d\theta/dt)$

where V_L = Velocity of the left wheel

V_R = Velocity of the right wheel

and W = distance between the center of the two wheels.

The design objective was to obtain a stable control over the motion control system with a good phase and gain margin and a fast unit step response. For this purpose a Galil 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 motor. The system was modeled in MATLAB using SIMULINK and the three parameters of the PID controller were selected using a simulation model to obtain the optimum response.

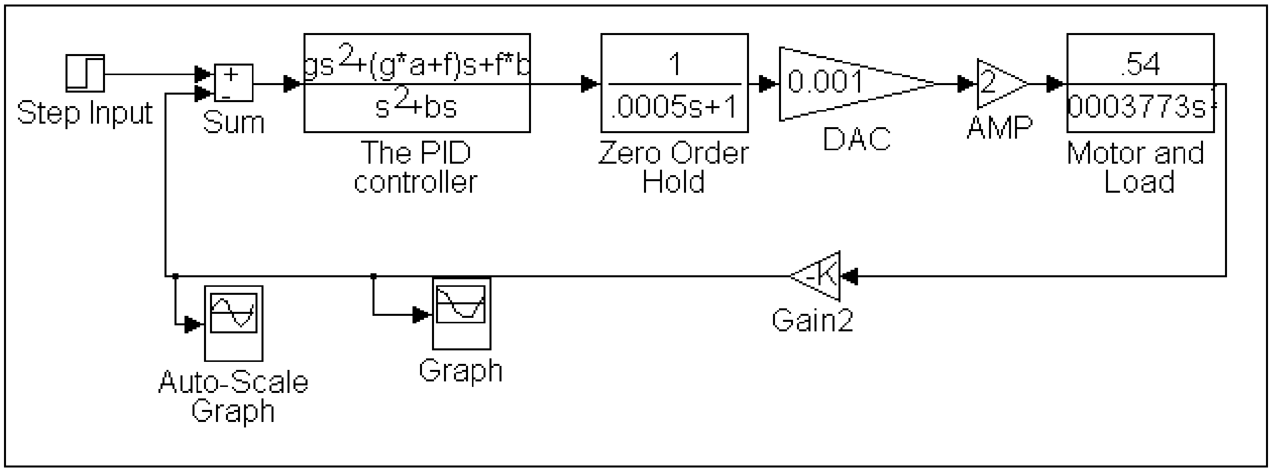


Figure 5.5 SIMULINK model

The SIMULINK model is shown in Figure 5.5 and starts with 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 equivalent digital filter used on the actual system. These analog values in the PID controller model adjust the input signal and feed it to the zero order hold. The zero order hold produces a pulse signal. The digital signal is fed to a digital to analog converter and then to an amplifier. This amplified signal is then fed to the load, which are the motors and drive wheels. The encoders detect the speeds of the left and right drive motors and gives signal that is fed back to the summation block for correction.

The unit step response was simulated in MATLAB. 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 1030. This software also allowed us to estimate the frictional losses in the gearbox, wheel bearings and alignments. A conservative tuning was performed and values 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 74 degrees and a gain margin of 86 decibels were achieved. Then, the step response was checked to minimize the overshoot and select a critically damped response. The actual tests were made in two conditions: Drive wheels off the ground, drive wheels on the ground. 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 encoder to be easily connected. The motion control system gets its input for the angle to be moved from three inputs: the distance from the obstacle and the angle of the line and distance from the robot centroid 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.

5.6 Safety and Emergency Stop System

The safety and emergency stop system consists of remote controlled emergency stops, manually operated emergency stops. The wheels of the AGV are mechanically connected to the motor through a irreversible (worm and worm wheel) gearbox. So if no power is supplied to the motors, the vehicle stops instantly. The mobile robot must is activated by a remote unit from a distance of no more than 50 feet in compliance with the rules for this contest. The remote controlled emergency stop consists of a transmitter and a receiver.

The JR Max6 transmitter uses a 9V DC and transmits FM signals at 72.470 MHz over a range of more than 50 feet. These signals are received and converted into a current that is amplified with a gain factor of 120. This amplified current cuts power to the motor.

The manual emergency stop unit consists of three manual push buttons situated on easily accessible side surfaces of the robot. When pushed, the main power is shut down through two solenoids. A solenoid acts like a relay but it is capable of handling more current.

Basically, when current flows through its coils, it closes an internal switch shutting down the main power. This serves as a safety measure against any possible runaway situation for the robot.

Chapter 6

RESULTS

This chapter presents results of the extensive laboratory tests that had been carried out to verify that the Bearcat II robot has indeed a better maneuverability over the Bearcat I robot, due to its zero turning radius features and comparatively smaller size. The Bearcat II robot was first tested for its zero turning radius capability on the June 5, 1998 before our group picture was taken with the Bearcat II robot in front of the Engineering Research Center Building at the University of Cincinnati. The observations on the performance of the individual subsystems are summarized as below.

6.1 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 were suggested to overcome them.

6.2 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.

6.3 Motion Control System

The ZTR feature of the Bearcat II robot worked satisfactorily. Implementing the PID controller variables as obtained from the SIMULINK model gave a stable response with almost zero overshoot. The system exhibits a stable but over damped closed loop behavior. The gain and phase margins are respectively 86db and 74 deg respectively. The step response shows that even in the over damped state, the system reaches steady state within half a second. In actual operation a number of other factors also contribute to the effect on these responses. Another fact to be borne in mind is that factors like inertia have been approximated since the actual inertia and damping due to friction in the system is hard to measure and account for.

6.4 Safety and Emergency Stop 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 7

CONCLUSIONS AND RECOMENDATIONS

A smaller Autonomous Guided Vehicle with better maneuverability due to its zero turning radius feature has been developed. The system embodies motion 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.

Upon testing the Bearcat II robot on the contest test track which was an uneven terrain the team members realized that a three wheeled vehicle with a marginal suspension system was not very stable. So having a good suspension system and adding the fourth wheel in future designs will improve the stability. As the castor wheel is free to swing, it seemed to get stuck in the sand and the grass at the contest test track restricting the normal functioning of the motion control system. Having a motorized castor wheel, which could also help steer the vehicle, could rectify this. The ground clearance of the vehicle was just satisfactory. A good factor of safety for the ground clearance for the future designs must be considered.

Towing the vehicle requires the disengaging of the wheels from the gearbox as the gearbox is irreversible (worm and worm wheel). In future designs a quicker way to engage and disengage the wheels from the gearbox while towing the vehicle should be incorporated.

A database can be created to store the values of where the curves are to be negotiated on the test track, and where the obstacles are located, during the first run. So the speed of the vehicle can be increased in the next runs as the vehicle has a database of the track path and the obstacles. This will help reduce the time for finishing the complete track in the next runs.

In future designs a variable speed mode could be implemented in the AGV to regulate the speed in real time according to the terrain and obstacle course on the track. This will help reduce the time to complete the competition track, which will be an important thought to be considered.

References

1. Devdas Shetty and Richard A. Kolk, **Mechatronics System Design**, PWS Publishing, Boston, MA, 1997, pp. 12-23
2. Hajime Terasaki and Tsutomu Hasegawa, "Motion Planning of Intelligent Manipulation by a Parallel Two-Fingered Gripper Equipped with a Simple Rotating Mechanism**," IEEE Transactions on Robotics and Automation**, Vol. 14, No. 2, April 1998, pp. 207-218.
3. Krzysztof Tchon and Robert Muszynski, "Singular Inverse Kinematic Problem for Robotic Manipulators: A Normal Form Approach," **IEEE Transactions on Robotics and Automation**, Vol. 14, No. 1, Feb. 1998, pp. 93-103.
4. Guy Campion, Georges Bastin and Brigitte D'Andrea-Novel, "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots**," IEEE Transactions on Robotics and Automation**, Vol. 12, No. 1, Feb. 1996, pp. 47-61.
5. M. F. Abdin, “An Investigation of the Potentials of Bi-directioanl AGV Systems,” **Proceedings of 5th International Automated Guided Vehicle Systems Conference**, Tokyo, Oct. 1987.
6. M. H. E. Larcombe, “Tracking Stability of Wire Guided Vehicles”, **Proceedings of 1st International Automated Guided Vehicle Systems Conference,** 1973, pp. 137-144.

7. Umit Ozguner, Konur A. Unyelioglu, Cem Hatipoglu, "Analytical Study of Vehicle Steering Control", **IEEE Conference on Control Applications - Proceedings 1995. IEEE**, Piscataway, NJ, USA, pp. 125-130.
8. 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 3208, **Society of Photo-Optical Instrumentation Engineers**, Bellingham, WA, USA, 1997, pp. 104-110.
9. Kalyan Kolli and E. L. Hall. "Steering Control System for a Mobile Robot", Proceedings of SPIE - The International Society for Optical Engineering. v 3208, **Society of Photo-Optical Instrumentation Engineers**, Bellingham, WA, USA, 1997, pp. 162-169.
10. Andre Kamga, Ahmed Rachid, "Speed, Steering Angle And Path Tracking Controls For A Tricycle Robot", **Proceedings of the IEEE International Symposium on Computer-Aided Control System Design**, 1996, pp. 56-61.
11. Bradley O. Matthews, Michael A. Ruthemeyer, David Perdue, Ernest L. Hall, "Development of a mobile robot for the 1995 AUVS competition", **Proceedings of SPIE - The International Society for Optical Engineering,** v 2591, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA, 1995, pp. 194-201.
12. Tayib Samu, Nikhal Kelkar, David Perdue, Michael A. Ruthemeyer, Bradley O. Matthews, Ernest L. Hall, "Line following using a two camera guidance system for a mobile robot", **Proceedings of SPIE - the International Society for Optical Engineering**, v 2904, 1996, pp. 290-297.

13. Kevin Warwick, **Control Systems: An Introduction**, Singapore, World Scientific, 2nd Edition, 1996, pp. 93-127.

14. S. Thompson, **Control systems engineering and design**, Harlow, Essex, England, Longman Scientific & Technical, New York, NY : Wiley, 1989, pp. 132-157.

15. Norman S. Nise, **Control Systems Engineering**, Redwood City, California, Benjamin/Cummings Pub. Co., 1992, pp. 7-36.

16. Naresh K. Sinha, **Control Systems**, New Delhi, John Wiley & Sons, 2nd Edition, 1995, pp. 149-181.

17. Charles L. Phillips, Royce D. Harbor, **Basic Feedback Control Systems**, Englewood Cliffs, New Jersey, Prentice Hall, 1991, pp. 147-195.

18. Mohammed S. Santina, Allen R. Stubberud, Gene H. Hostetter, **Digital Control System Design**, Fort Worth, Saunders College Pub., 1994, pp. 64-89.

19. Benjamin C. Kuo, **Digital Control Systems**, Fort Worth, Saunders College Pub., 1980, pp. 28-32.

20. Alberto Isidori, **Nonlinear Control Systems: An Introduction**, Berlin ; New York : Springer-Verlag, 1989, pp. 47-53.

21. 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.

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

23. E.L. Hall and B.C. Hall, **"**Robotics: A User-Friendly Introduction", Holt, Rinehart, and Winston, New York, NY, 1985, pp. 23-40.

24. Galil Inc, “DMC-1000 Technical Reference Guide Ver 1.1”, Sunnyvale, California 1993, pp. 10.125-10.138.

25. Reliance Electric, “Electro-Craft BDC-12 Instruction Manual”, Eden Prairie, Minnesota 1993, pp. 135-161.

26. BEI Motion Systems Company, “Model H20 Technical Specifications”, Goleta, California, 1992.

27. Newport Corporation, “Motion Control Basics”, Irvine, California, 1997.

28. S.M. Shinners, “Modern Control System Theory and Application”, John Wiley & Sons, INC., 2nd Edition, 1998, pp. 362-380.

29. M. Ehtashami, S. J. Oh, and E. L. Hall, "Omnidirectional Position Location For Mobile Robots," **Proceedings of International Conference on Intelligent Robots and Computer Vision**, Cambridge, MA, v 521, Nov. 1984, pp. 62-73

30. L. Cao, S. J. Oh, and E. L. Hall, "Dynamic Omnidirectional Vision for Mobile Robots," **Journal of Robotic Systems**, 3(1), 1986, pp. 5-17.

31. Y.Y. Huang, Z.L. Cao, S.J. Oh, E.U. Kattan and E.L. Hall, "Automatic Operation for a Robot Lawn Mower," **SPIE Conference on Mobile Robots**, Vol. 727, Cambridge, MA, Nov. 1986, pp. 344-354.

32. E.U. Kattan, S.J. Oh and E.L. Hall, "Development of an Intelligent Robot Lawn Mower," **Proceedings of VI Annual Technical Seminar of the New England Chapter of AUVS**, Bedford, MA, April 1987.

Appendix A

C++ Source code for the Motion Control

```
// car.c          5-28-98

#include "car.h"
#ifndef CAR
#define CAR
#include "sys\types.h"
#include "time.h"
extern long int spdx,spdy;
extern time_t first;

void speedCARx(long int spx){
      //This command when invoked will set the speed of bearcat II.
      //It is called with a value between 0 - 250000.

      char inx[10],sendx[15];
      gcvt(spx,6,inx);
      char *commandx="JG";
      strcpy(sendx,commandx);
      strcat(sendx,inx);
      cout<<sendx<<"\n";
if(!TEST)
   submitDMC(sendx);
   submitDMC("BGX");
}
void speedCARy(long int spy){
      //This command when invoked will set the speed of bearcat II.
      //It is called with a value between 0 - 250000.

      char iny[10],sendy[15];
      gcvt(spy,6,iny);
      char *commandy="JG ,";
      strcpy(sendy,commandy);
      strcat(sendy,iny);
      cout<<sendy<<"\n";
if(!TEST)
 submitDMC(sendy);
 submitDMC("BGY");
}



void stopCAR(){
      submitDMC("ST");
      submitDMC("MO");
      submitDMC("SH");
      }


void steerCAR(float val)
      { double dtime;
```

```
       time_t second=time(NULL);
      //This function when invoked will put the steering wheel to the
absolute
      //angle given.    This angle ranges between +-20.
      dtime=difftime(second,first);
      first=second;
      spdx+=(134.5*val)/dtime;
      spdy-=(134.5*val)/dtime;
      speedCARx(spdx);
      speedCARy(spdy);
      }


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 spdx=-.05*abs(val)+COMP_SPEED;
      speedCARx(spdx);

      //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>
/*
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              05-06-98
#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 neccasary settings for proper
operation
//of the steering and drive motors.
      submitDMC("KD2000,550");
      submitDMC("KP100,50");
      submitDMC("KI0,0");
      submitDMC("TL,9.98");
      submitDMC("FL21474836,21474836");
      submitDMC("BL,-90");
      submitDMC("IT,1");
      submitDMC("AC25600,25600");
      submitDMC("DC25600,25600");

}


#endif
```

Appendix B

C++ Source code to submit the PID Constants to the Motion Controller

```
/*    setpid.c
      Setup the contants for the PID controller
*/

void setpid()
{
 submitDMC("DP 0,0,0");
 submitDMC("SH");
 submitDMC("AC 256000,256000");
 submitDMC("DC 256000,256000");
 submitDMC("KD 1118,2564.7");
 submitDMC("KP 100.25,229");
 submitDMC("KI 0,0");
}
```