T-4227

AUTOMATIC STEERING SYSTEM FOR AN UNDERGROUND MINE HAUL TRUCK

by

James D. Lane

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-4227

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mining Engineering).

Golden, Colorado

Date 6-12-92

Signed: James D. Lane

James D. Lane

Approved: Robert H. King

Robert H. King, Ph.D.

Thesis Advisor

Golden, Colorado

Date 6/22/92

Donald W. Gentry

Donald W. Gentry, Ph.D.

Department Head, Mining

ABSTRACT

The research presented describes the process of component selection, installation and testing, perception algorithm development, and steering control algorithm development to produce a real time steering control system for an articulated underground mine haul truck. The thesis traces the evolution of an effective steering control system and describes the perception hardware and software performance. Ultrasonic transducers providing perception and a hydraulic cylinder linear potentiometer (lp) providing steering angle data comprise the sensor suite. A GESPAC system analyzes the data and controls the steering with a Vicker's electro/hydraulic proportional valve. The system of components and software presented, provides an acceptable solution for real-time steering control.

The steering algorithm presented follows a projected centerline developed from the ultrasonic transducer data defining the walls on each side of the vehicle. Several experiments test the algorithm and its ability to successfully steer the vehicle.

TABLE OF CONTENTS

FIGURES

TABLES

Abbreviated terms used in this Report

opamp- operational amplifier
lp- Linear Potentiometer
Vdc- DC Voltage
Vac- AC Voltage
Mhz- Mega Hertz
Khz- Kilo Hertz
msb- Most Significant Bit
lsb- Least Significant Bit
MPU- Micro Processing Unit
LHD- Load Haul Dump vehicle
ALV- Autonomous Land Vehicle
ADC- Analog to Digital Converter

ACKNOWLEDGEMENT

The author wishes to express his appreciation to Dr. Robert H. King, Thesis Advisor, for his support and commitment to this research and, even more so, his encouragement to pursue this research topic. Thanks are extended to Dr. Robert E. Cameron and Dr. John H. Steele, Thesis Committee Members, for all of their support and insight. This research was partially funded by the Department of Interior's Mineral Institute program and the U.S. Bureau of Mines through the Pittsburgh Research Center. The support and interest of Mr. Jeff Welsh, Mr. John McClelland and Mr. George Schnakenberg of the Pittsburgh Research Center (USBM) is deeply appreciated.

The author wishes to deeply thank his family, Phil, Diane, Jenny, Anne, and Sarah, for their many avenues of encouragement and support during his entire stay at this college.

Support from several faculty and staff members is appreciated, including: Professor Groves, Gaby Neunzert, John Jezek, Victor Tancreti, Dave Hale, and Bill Strickland.

The author wishes to express his appreciation to Julie for the loving pushes he has continually received during the past several months to complete this research sooner rather than later.

Most importantly, I thank God for the opportunities He has placed before me to work with so many generous people on such a rewarding project; it is to Him that this work is dedicated.

Chapter 1

INTRODUCTION

1.1 Problem Statement

Retro-reflective tape following with video camera and trolley or guide wire following techniques presently used provide partial solutions to autonomous navigation for underground mining equipment. These solutions require personnel to enter the mining area and install the guide mechanisms. Since autonomous equipment should remove personnel from unsafe working environments, a better solution would not require guide mechanism installation. The solution proposed herein employs ultrasonic transducers to guide a mining vehicle without guide mechanism installation.

Many advantages exist with the use of ultrasonic rangers over other automatic guidance alternatives. King, Lever, Strickland, and Lane (1991) describe these alternatives:

- -ultrasonic rangers use natural targets, so artificial targets do not have to be installed.
- -the mining machine can be guided through multiple drifts or entries; it isn't confined to those that contain wire or paint stripes.

-ultrasonic rangers can be used for detecting obstacles in vehicle path.

-ultrasonic rangers have no moving parts.

-ultrasonic rangers are robust and can withstand the mining environment.

-ultrasonic rangers have been used reliably in the mining industry for many years to measure bin levels, so their technology is not new to maintenance and operating personnel.

-ultrasonic rangers can be used for operator aids (i.e. to prevent collisions on difficult-to-see parts of the mining machine) in addition to computer guidance.

-ultrasonic rangers are relatively inexpensive and simple to replace if damaged.

-ultrasonic rangers can be packaged to operate safely in environments that contain explosive gas or dust.

-an ultrasonic ranger guidance system can operate solely with on-board power, communications and control actuators. It does not require any off-board or external systems.

1.2 Hypothesis

A guidance system can be developed that will control the steering of an articulated mining vehicle using components appropriate for the mining environment. Components include a GESPAC Motorola 68030 microprocessor, ultrasonic transducers, hydraulic cylinder linear potentiometer (lp), and an electro/hydraulic proportional valve.

The algorithm developed uses ultrasonic transducers for environment perception as the vehicle travels through the site. The ultrasonic data is transposed with respect to the steering angle and velocity, locating the vehicle relative to the mine ribs. Real time steering adjustments maintain the vehicle position within acceptable error along the target path.

1.3 Scope of Work

This thesis focuses on the development of a system to adequately steer the vehicle through a test course consisting of a straight and curved section. The brakes, shifting, and engine idle are manually controlled, since computer control of these tasks is not necessary to test the steering control system and other researchers have previously demonstrated computer control of these components.

This steering automation process is intended for pre-developed mining environments characterized by repeated cycles over a designated route. The result is a predictable environment with very little change encountered between trips.

1.4 Proposed Method

Ultrasonic transducers mounted on four corners of the vehicle and positioned at a 45 degree incidence angle produce information describing the surrounding physical environment. A linear potentiometer enclosed inside one of the steering cylinders provides information describing the vehicle steering angle. An electro/hydraulic proportional valve controls vehicle steering. The GESPAC computer provides a real time platform for data acquisition and steering control. The error experienced between the actual and target position is computed and analyzed for a series of experiments. The following list presents the tasks performed in this project.

1. Propose Algorithm ... Chapter 5.
2. Select components to test algorithm ... Chapter 3.
3. Purchase and install components ... Chapter 3.
4. Interface components ... Chapter 3.
5. Test and calibrate components ... Chapter 3.
6. Encode Algorithm ... Chapter 5.
7. Test Algorithm ... Chapter 5.
8. Analyze Data ... Chapter 5.
9. Modify algorithm, modify software, retest, reanalyze

... Chapter 5.

10. Report results ... Chapter 5, and Chapter 6.

Chapter 2

LITERATURE REVIEW

"Mining takes place in the geological environment where conditions are highly variable and unpredictable. As a result, autonomous mining equipment must have substantial cognitive abilities to recognize and deal with these unpredictable variations." [King and Steele, 1989] The autonomous vehicles developed must be easily adaptable and feasible to be considered for mining applications.

As mining equipment experiences greater levels of autonomy, eventually reaching robot status, the operation and maintenance of these devices must be kept simple. It is not feasible or practical employing several highly trained personnel to maintain and operate such autonomous vehicles. It is desirable to develop a reliable system that a mine operator can easily maintain and operate since a mine can not profitably operate when experiencing continual downtime for repairs, parts delivery, or the arrival of specially trained personnel.

The mining environment presents a high demand on

alertness from mine workers. Combining the pressure to perform with normal mining conditions of heat, humidity, dust, noise, fumes, and other hazards, a need exists to ease the demand on machine operators by removing them from environments with these conditions and placing them in a safe, protected operating environment.

The projects described herein provide solutions to automatic vehicle steering, but they may not function acceptably in a mining environment. Many solutions presented require a large budget and highly trained personnel to develop and maintain and, therefore, lack profitability. Sensitivity to dust, humidity, and continual vibration render many of these solutions unacceptable. One solution, presented by St-Amant and Vagenas, requires artificial target installation, which limits vehicle use to those targeted areas.

Hennessy and King, (1987) describe the Defense Advanced Research Projects Agency (DARPA) sponsored Autonomous Land Vehicle (ALV) research. The ALV has successfully completed autonomous navigation on black top roads and cross-country. The ALV is an eight wheel, skid steered vehicle designed for on and off road. Controllable top speed of the vehicle is

limited to 25 kph on road and 10 kph off road.The ALV utilizes color video, a scanning laser ranger, and a gyrocompass to perceive the environment and locate itself. These sensors, along with others, require a tremendous amount of computing power including three Sun processors.

McTamaney (1987) describes the FMC mobile robot test bed developed for DARPA's Advanced Ground Vehicle Technology (AGVT). DARPA worked to develop a modified M113 armored personnel carrier that is operated in remote or on-board modes. This system uses a suite of perception sensors to develop a local map which is fused with a global map to guide the vehicle from its starting point to its goal.

St-Amant (1991) describes a system designed for underground vehicle guidance using optical line and video cameras developed by the *Canadian Centre for Automation and Robotics in Mining* (CCARM) and the Noranda Technology Center (NTC). The software developed recognizes branchings, end points, and milestones using a reflective tape targeting system. This system employs two cameras, one for forward travel and one for reverse travel. The reflective tape is used to steer the vehicle and to indicate up coming events

such as corners, intersections, or crosscuts. The system operated successfully on an underground articulated Load Haul Dump (LHD) mining vehicle.

Vagenas (1991) describes a combination of teleoperation and automatic navigation of an underground LHD using a paint stripe following method. The paint stripe is applied to the ceiling of the mine, providing a target for the camera to follow. Presently, the loading and dumping tasks are carried out by remote control and teleoperation with a mine operator at the controls. A computer controls the tramming process using the paint stripe following method.

Ruff (1991) describes the guidance of a compact loader/trammer to be used in underground narrow-veined mines. The vehicle is a six-wheeled, skid-steered LHD. The LHD is equipped with ultrasonic sensors and a microprocessor to provide automatic guidance during tramming. It is also equipped with radio remote control. The LHD operates in a narrow-vein stope simulated site and successfully trams at full speed (3 mph) in forward and reverse directions. The stope width is six feet and the LHD width is 4.5 feet, providing 1.5 feet for vehicle wandering.

Denning Mobile Robotics, Inc. produces an omnidirectional Modular Research Vehicle (MRV) to perform a variety of user desired tasks. A Denning Ring assembly guides the MRV's. Common uses of the MRV's include intelligent security robot systems and mobile robot platforms for delivery tasks.

Oppenheim (1985) describes research conducted to develop a mine map with a vehicle known as the TERREGATOR. The TERREGATOR is a six-wheel, skid-steered vehicle. A "sonar ring" developed by Denning Mobile Robotics, Inc. perceives the mine ribs. Results suggest that the Denning Ring provides adequate perception capabilities to map and navigate the TERREGATOR in an unstructured environment.

Elfes (1986) describes research conducted by CMU using ultrasonics to detect and map objects and navigate vehicles in both laboratory and outdoor environments. Both cases, use a global map combined with ultrasonics to detect local features when travelling from one designated point to another as instructed according to the global map information.

Everett (1989) describes research conducted to develop a security robot utilizing ultrasonics and a large suite of

accompanying sensors. The ultrasonics are set up in a special array along the front of Robart II and a ring of ultrasonic transducers are included around the top of the vehicle. These sensors provide information valuable to environment perception and vehicle path-planning.

Steele (1988) describes research conducted on a mobile robot called Remolina, built by the Intelligent Machines Principles Division at Sandia National Laboratories in Albuquerque New Mexico. Remolina uses ultrasonics, proximity sensors, structured lighting, and a moving obstacle sensor to successfully navigate through an area laden with obstacles. Remolina plans paths around static objects and avoids moving objects while traversing an indoor lab environment. Steele developed a robust robot navigational system that uses geometric path planning and real-time avoidance of obstacles encountered while traversing the preplanned path.

The steering systems discussed present acceptable solutions for steering control in the context of their developmental environment. Most projects utilizing ultrasonic transducers combines that information with global maps to successfully guide the vehicle. The remaining ultrasonic

systems do not present a robust solution designed to steer articulated underground mine vehicles. None of the ultrasonic systems are applied to an actual underground mine vehicle with articulated steering. The ALV project presents an autonomous rugged solution, but costs are not profitable for an underground mine application and the developing and servicing personnel must be highly trained. The work described in these projects does not provide a feasible and easily installed and maintained solution for underground mine haul vehicles.

Chapter 3

SYSTEM COMPONENTS

3.1 Test Vehicle

The test vehicle, an underground mine haul truck seen in Figure 1, has articulated steering and a six cylinder air cooled Duetz diesel engine. This truck, manufactured by ELMAC, has very similar controls and steering concepts to many mining vehicles used in today's underground mines. Consequently, the steering system produced will transfer to other types of mining vehicles. This vehicle provides a real world application example rather than a small-scale model. Prototypes do not always behave like the target object, therefore it is desirable to skip this step and develop a full scale prototype.

3.2 Computer

A user friendly on-board computer system provides a desirable environment to develop a set of real-time steering commands. The GESPAC system provides many options including compact, inexpensive, low power consumption, robust, and

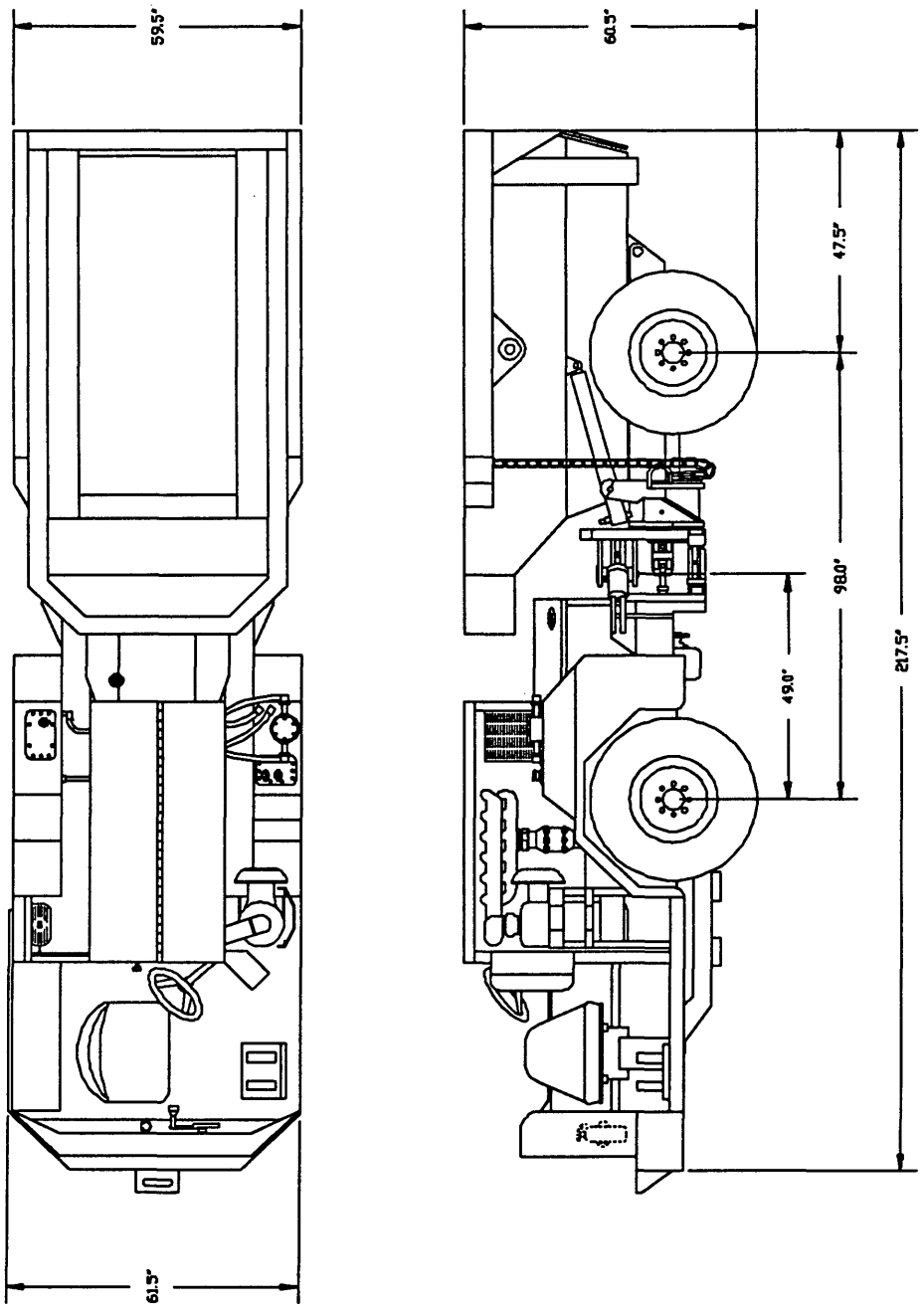

Figure 1. The Elmac dump truck test vehicle.

provides sufficient processing speed facilitating real time data acquisition and control. The system interfaces to data acquisition and machine control modules and communicates efficiently with other computers. This computer operated successfully in an underground coal mine for over a year to date. [Lever, 1991]

The GESPAC system consists of the following modules:

The GESMPU-30HF, a single Eurocard board, contains a 25 MHz Motorola 68030 32 bit microprocessor, an arithmetic coprocessor and four EPROM sockets providing 512Kbytes of program memory. The board contains 2Mbytes of CMOS Dynamic RAM and has low power requirements through the use of a CMOS processor and memory devices.

The GESMFI-1, a universal interface module on a single Eurocard, provides two RS 232-C compatible serial ports, 20 TTL I/O lines, a triple programmable 16-bit timer and a real time clock module.

The GESADC-2C, a data acquisition module on a single Eurocard, provides 16 single ended or 8 differential channels

with 12 bit resolution and an ADC conversion time of 30 microseconds. The board also contains linear instrumentation amplifiers and software programmable amplifiers.

The GESRAC-12/6 card cage holds the boards and they communicate along a G-64/G-96 bus. The rack is an industrial grade eight slot back plane with a power supply.

This hardware provides the following capabilities:

-Sustained processing rate over 4 MIPS.

-Low power consumption and battery operation capabilities.

-16 channel, 12 bit, 2000 s/s data acquisition with programmable amplifiers.

-Compact and rugged.

-Output control signals using 20 TTL I/O lines.

-External communication using 2 RS 232 serial lines.

The OS-9 operating system and application development environment completes the computing requirements. OS-9 is a high performance multitasking operating system for the 68000/30 microprocessor family. It provides multiuser capabilities, has a device independent input/output system,

hierarchical file structure, "Shell" user interface, resident assemblers, linkers and debuggers, and job control. The OS-9 kernel requires only the modules needed for the user application, thus providing real time performance characteristics. In addition, the operating system and applications programs can be merged into a single file and placed in EPROM as a diskless imbedded application. An IBM compatible 286 is presently utilized for software development and display. The PC communicates with the Gespac via RS-232 serial ports and a software package called PC bridge. Programs developed for the Gespac are loaded via PC bridge and activated from the PC through existing software functions.

3.3 Ultrasonic Transducers

Four Milltronics ST25C4 ultrasonic transducers are mounted on the corners of the truck. Their range is 2 - 14 feet and they have a field of view of 12 degrees. These transducers are robust and have been designed to operate in environments where particle suspension or humidity are present.

Each ranger is driven by its own circuit board which requires 120 volts AC input power. The boards link through a synchronizing connection which causes the rangers to pulse simultaneously. This cuts down on echo noise that is experienced between rangers when pulsing at different times. The transducers pulse at 11 Hz updating the previous reading with the newest one. The operator's manual describes these procedures in detail.

Each transducer produces a linear DC voltage output (0 - 10Vdc) proportional to the distance measured. The ADC converter receives sixteen single-ended inputs and a ground wire runs from each transducer board to the ADC external ground.

Ultrasonic transducers transmit short bursts of high frequency sound energy. The transmitted sound waves reflect off an object, and travel back to the sensor. The round trip time (time of flight) experienced is proportional to the distance between the ranger and the object and the medium through which the wave moves.

3.4 Steering Angle Transducer

Two steering cylinders equipped with linear potentiometers provide reliable steering angle data. Replacing both cylinders eliminates any differences that may exist between the original and special ordered sets. The data presented in Table 1 describes the cylinder characteristics.

Table 1. Characteristics of steering cylinder sensor.

	Digital value	Angle (Deg)	cylinder stroke(in)
full left turn	3055	41.5	6.09
full right turn	276	39.7	0
straight	1440	0	2.75

3.5 Proportional Valve

A proportional directional control valve, single-stage design, with a feedback transducer, manufactured by Vickers, provides steering control. An appropriate driver card controls the proportional valve. An interfacing circuit isolates the GESPAC parallel port from the driver card. This circuit, provides a variable +/-10Vdc voltage range.

Two hydraulic cylinders and a hydraulic steering gear box make up the manual steering. The flow rate required to steer the truck is determined by computing the volume displacement of the two steering cylinders and recording the time taken to steer from full right to full left. The computed flow rate is 2.92gpm. The proportional valve selected provides 3.43gpm. The driver card selected provides feed back control using external sensing devices and receives power from a 24Vdc, four-amp power supply.

3.6 Power Invertor

A 12Vdc to 120Vac power invertor provides 120Vac to the GESPAC, 24Vdc power supply, opamp circuit, and the ultrasonic rangers. Fuses and a circuit breaker protect this Tripplite 1000 watt invertor. It does not contain a voltage regulator on either the 12Vdc input side nor the 120Vac output side, so the actual voltage output is ten times the input dc Voltage. Therefore, since the vehicle alternator maintains a 15Vdc charge level, the actual invertor output is 150Vac. This is not acceptable and presents a need to regulate either the input or output of the invertor since the equipment operating range is between 105-135Vac. An umbilical cord connecting the

equipment to remote power presently provides the necessary power to operate the equipment.

3.7 Component Locations

The sensor locations, shown in Figure 3, provide optimum readings and easy access for maintenance or adjustments. Several mounting brackets, constructed from 0.125 inch X 2.5 inch plate steel, fasten the sensors and actuator to the truck. These brackets, shown in Figure 2, accommodate any transducer incidence angle desired to obtain optimum readings. The transducer locations provide protection during test runs. Determining optimum transducer position and location facilitates a safe, permanent installation.

Previous studies show that incidence angles as low as 30 degrees are achievable in the mine environment where walls are rough. [King, Lever, Strickland, and Lane, 1991] This research uses 90 and 45 degree incidence angles.

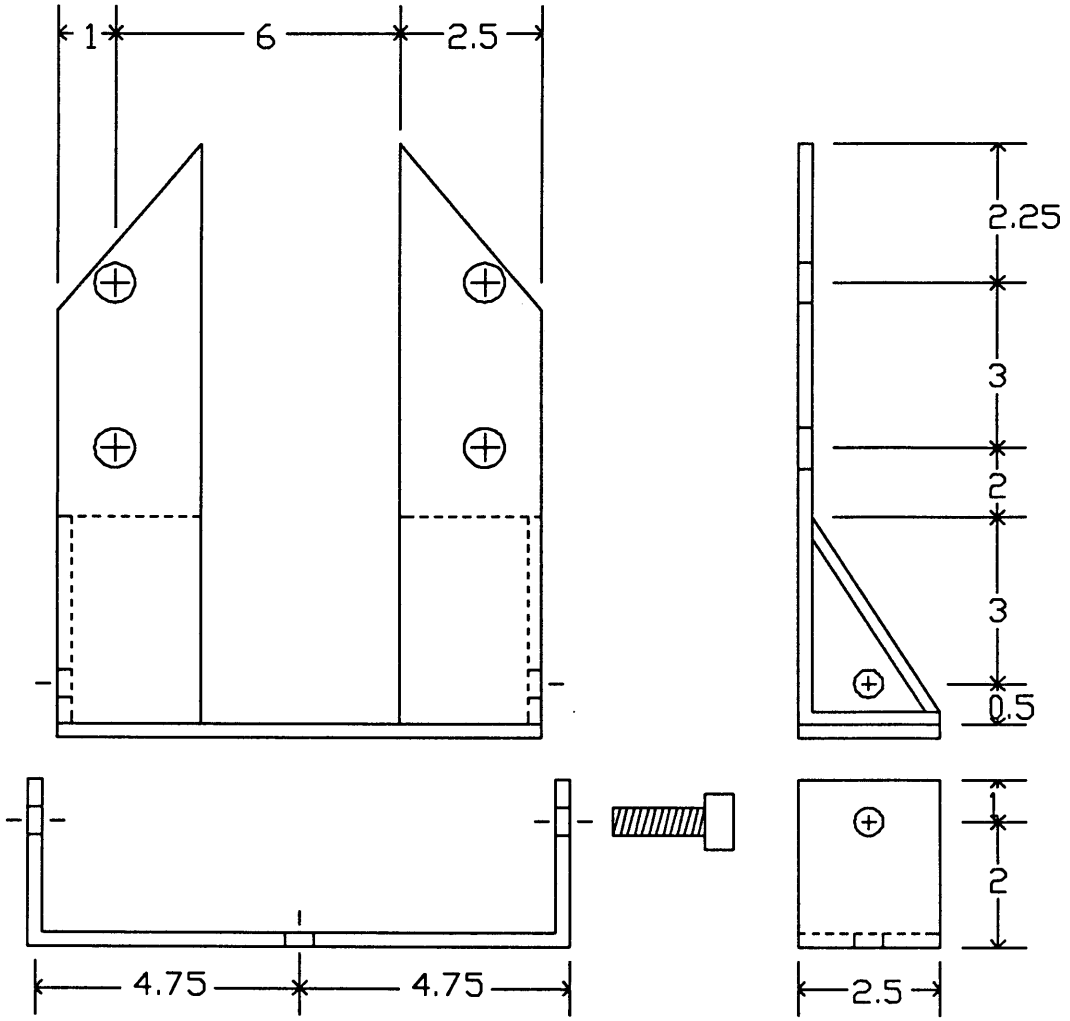

Figure 2. Ultrasonic transducer mounting bracket.

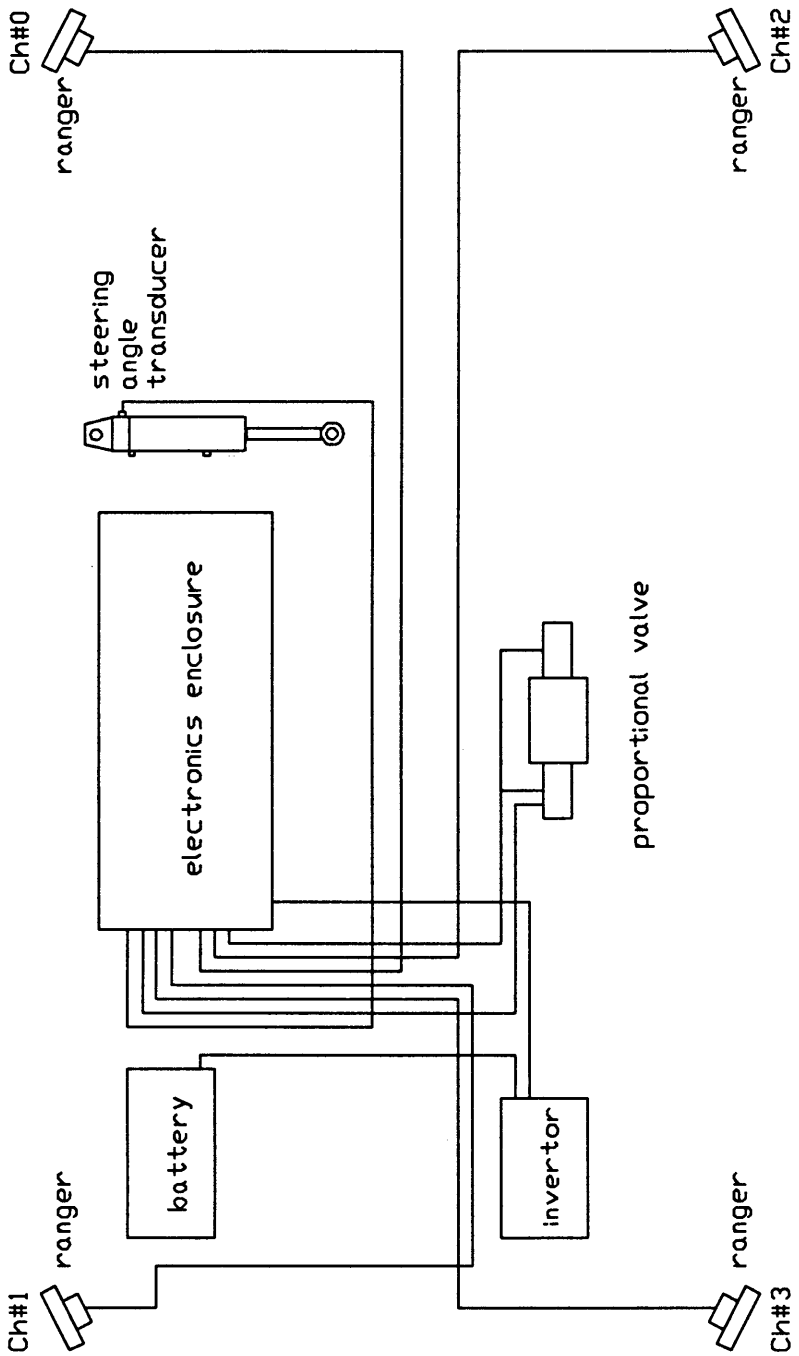

Figure 3. Component locations on vehicle.

3.8 Steering Modifications

Installing additional hoses, fittings, and a switch over valve provides switching between manual and computer controlled steering. This modification, shown in Figure 4, required adding eight hydraulic hoses, two T-fittings, and four base-plate fittings.

Mounting brackets connect the switch over valve and the proportional valve to the truck. The computer/manual steering switch over valve is located near the operator for easy access.

A second valve, seen in Figure 4 positioned between the manual steering gear box and the tank, prevents system bleeding through the manual steering when the computer controlled proportional valve is operating. This valve is closed during computer controlled steering tests and opened during manual steering operation. Without this modification, the proportional valve system looses operating pressure and fluid flow through bleed-off experienced in the manual steering gear box. The cause of this bleeding effect is uncertain, but is alleviated by including the valve.

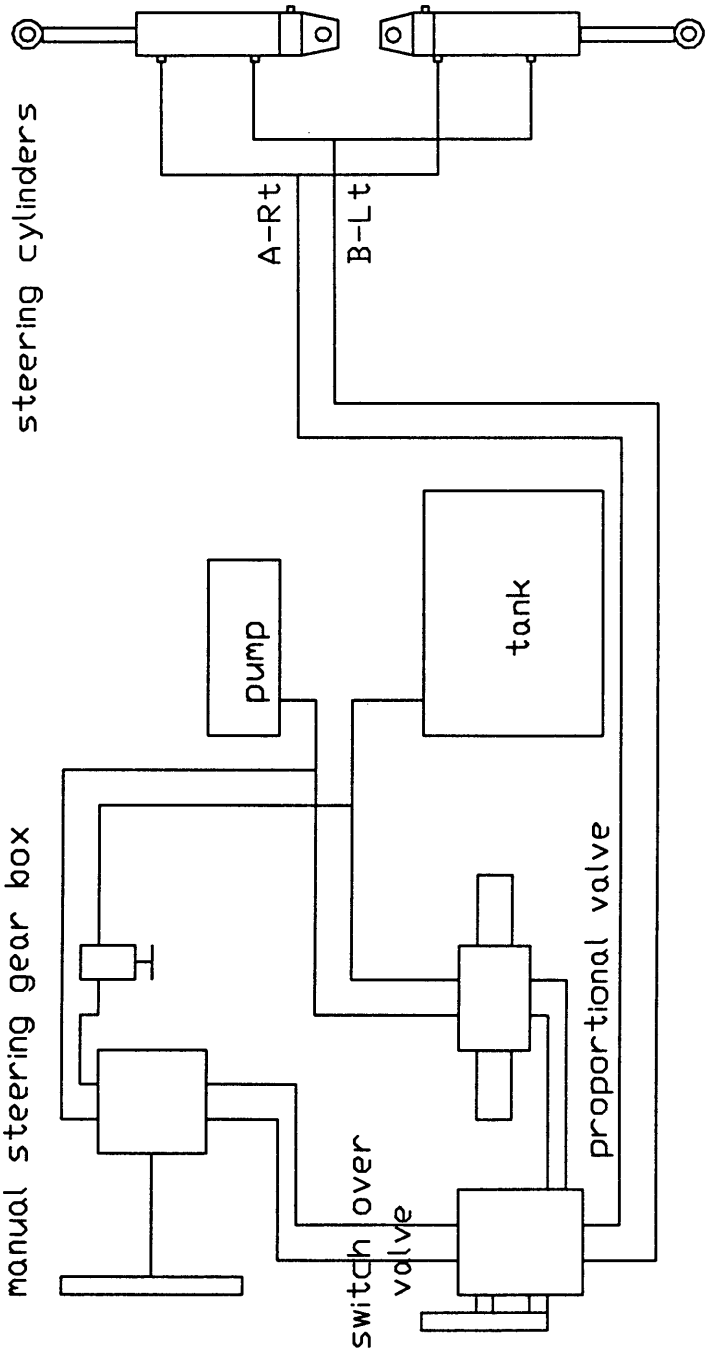

Figure 4. Hydraulic system modifications.

3.9 Component Expenses

Table 2 lists the components and corresponding costs to develop this steering system.

Table 2. Component expenses.

Component	Part Number	Cost (US$)
Steering cylinders	S24007018LLAZ	1662.00 /pair
Proportional Valve	KFDG4V-3-2C13N-Z-M-U1-H720	795.00
Driver Card	EEA-PAM-523-B-30	1056.00
Ultrasonic Transducer	ST25C4	535.00 X4
Transducer amplifier card	ST25C4	460.00 X4
Master capacitor kit	MCK-001	79.95
Assorted Terminals	975-3670	23.02
Resistor kit	525-1100	39.95
Switch over valve	DS-75	79.44
Hydraulic Hoses and fittings		200.00
DC-AC Power Invertor	PV1000FC	433.95
GESPAC	GESMPU-30 series	6291.00
TOTAL EXPENSES		14640.31

3.10 Hardware Interfacing

Figure 5, a block diagram, shows the computer interfacing with the components included in the steering control system. A driver card, shown in Figure 6, controls the proportional valve while receiving position spool feed back signals. Several connections must be made to activate the driver card and proportional valve. Figure 7 shows the ultrasonic transducer connections to the power, synchronization, and the ADC. Figure 8 shows the power connections to the ranger boards, driver card, GESPAC, and 24Vdc power supply. The valve driver card receives its power from the 24Vdc power supply. Figure 9 shows the component locations in an enclosure installed on the test vehicle.

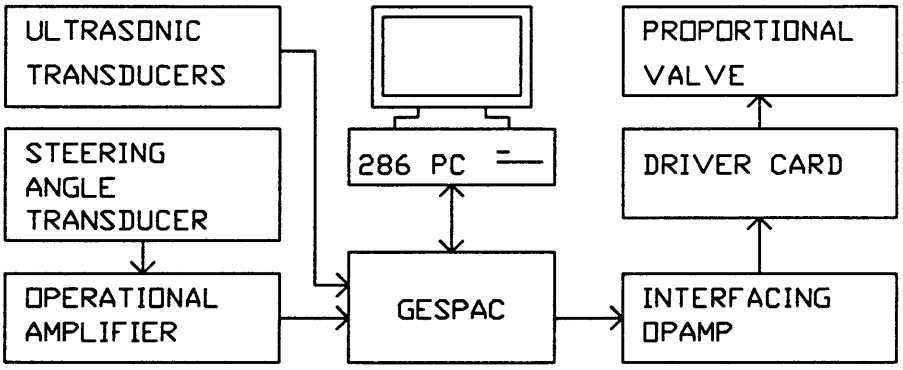

Figure 5. Block diagram of computer interfacing.

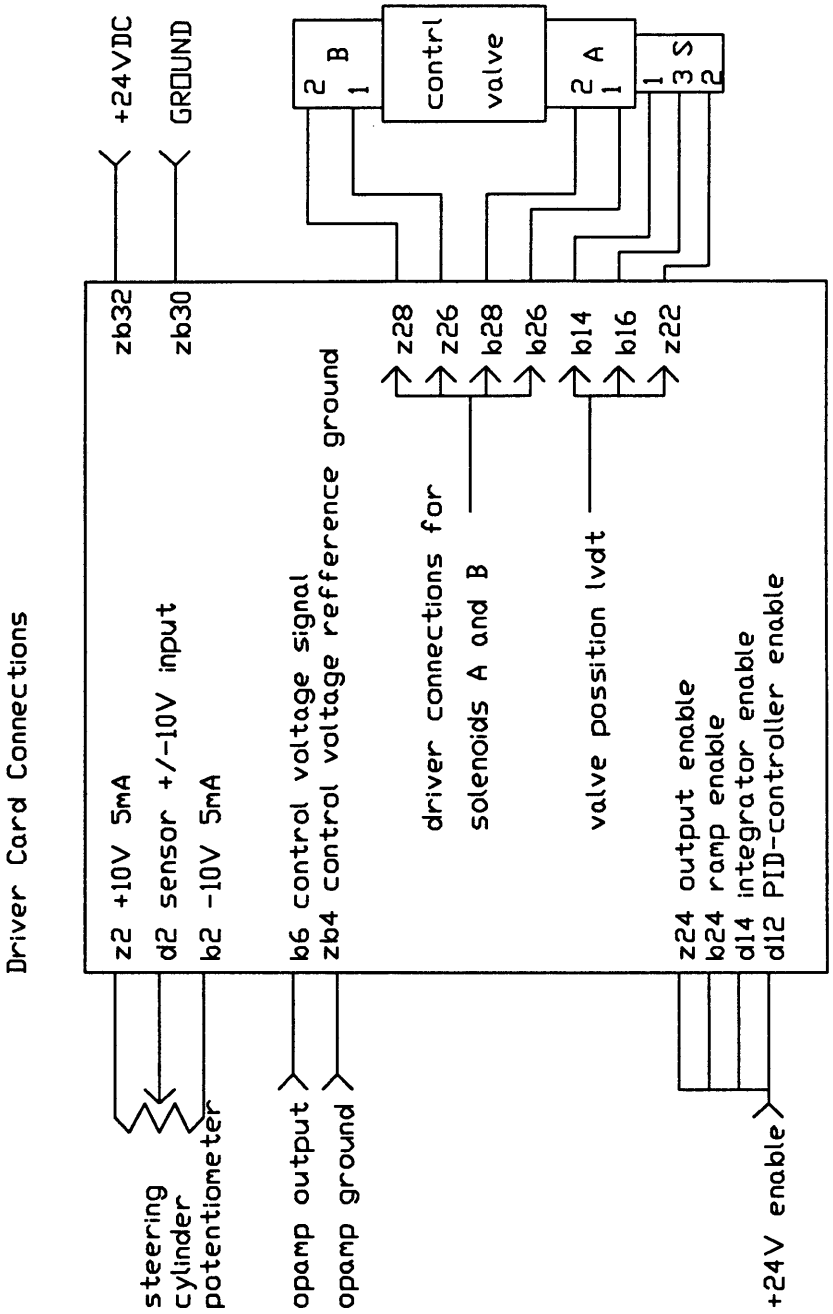

Figure 6. Valve driver card connections.

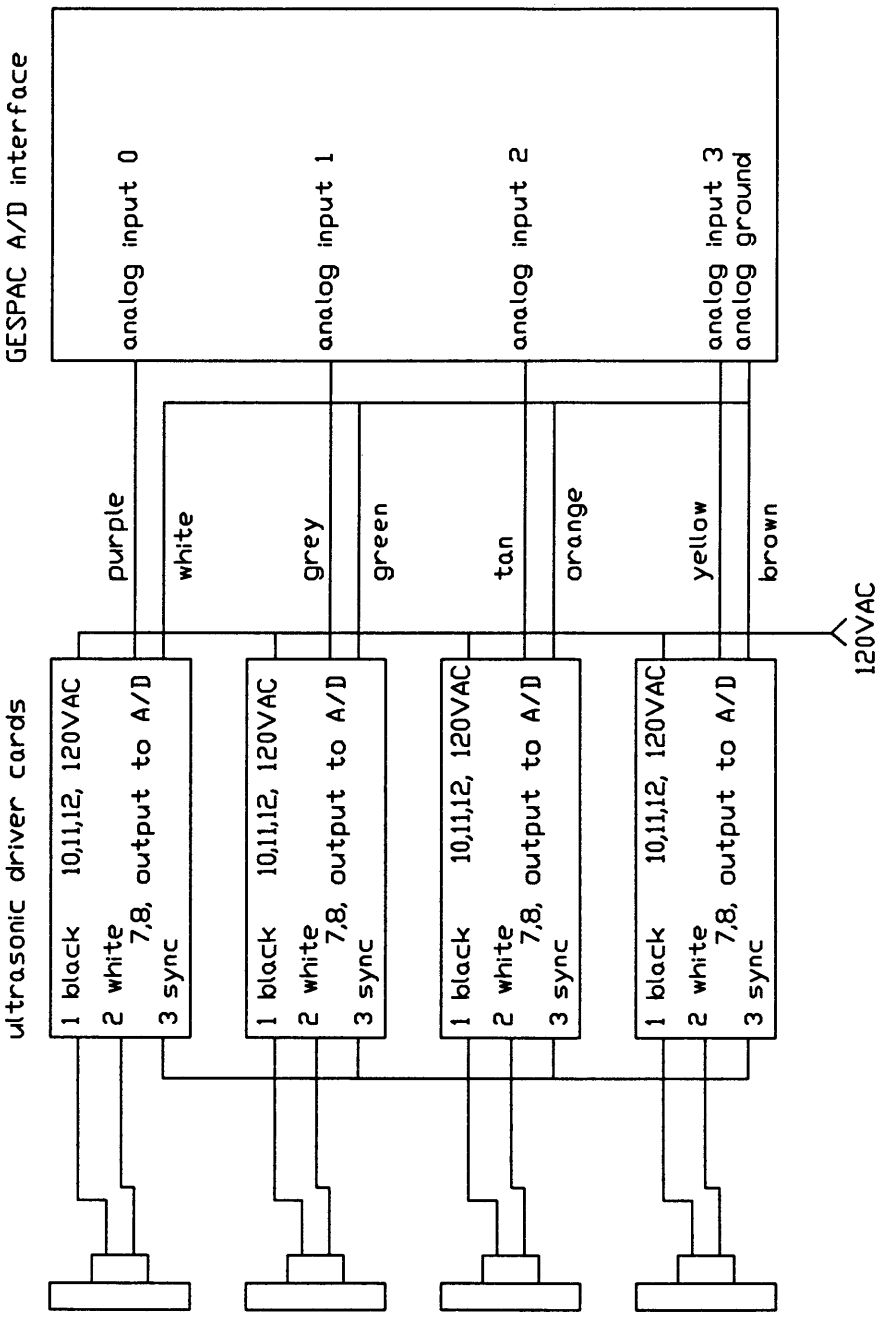

Figure 7. Ultrasonic transducer interfacing.

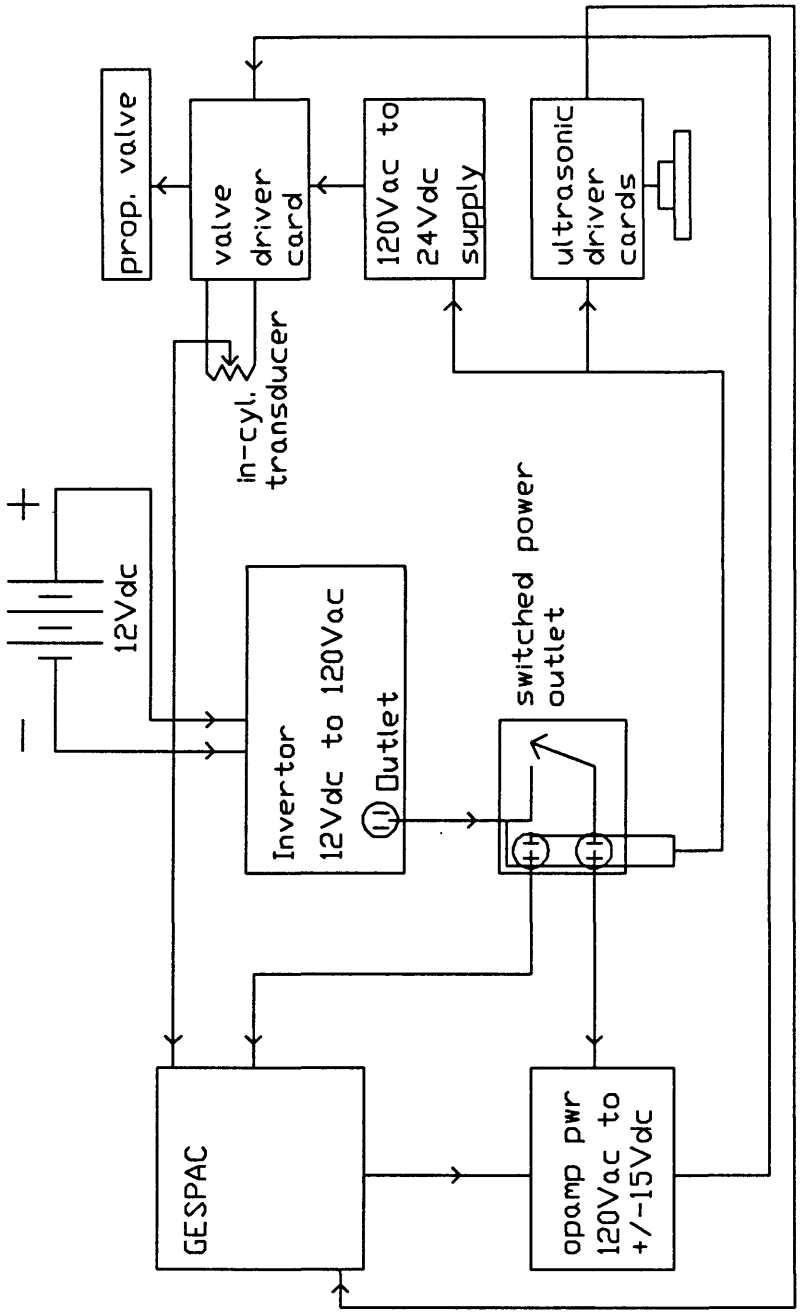

Figure 8. Power connections.

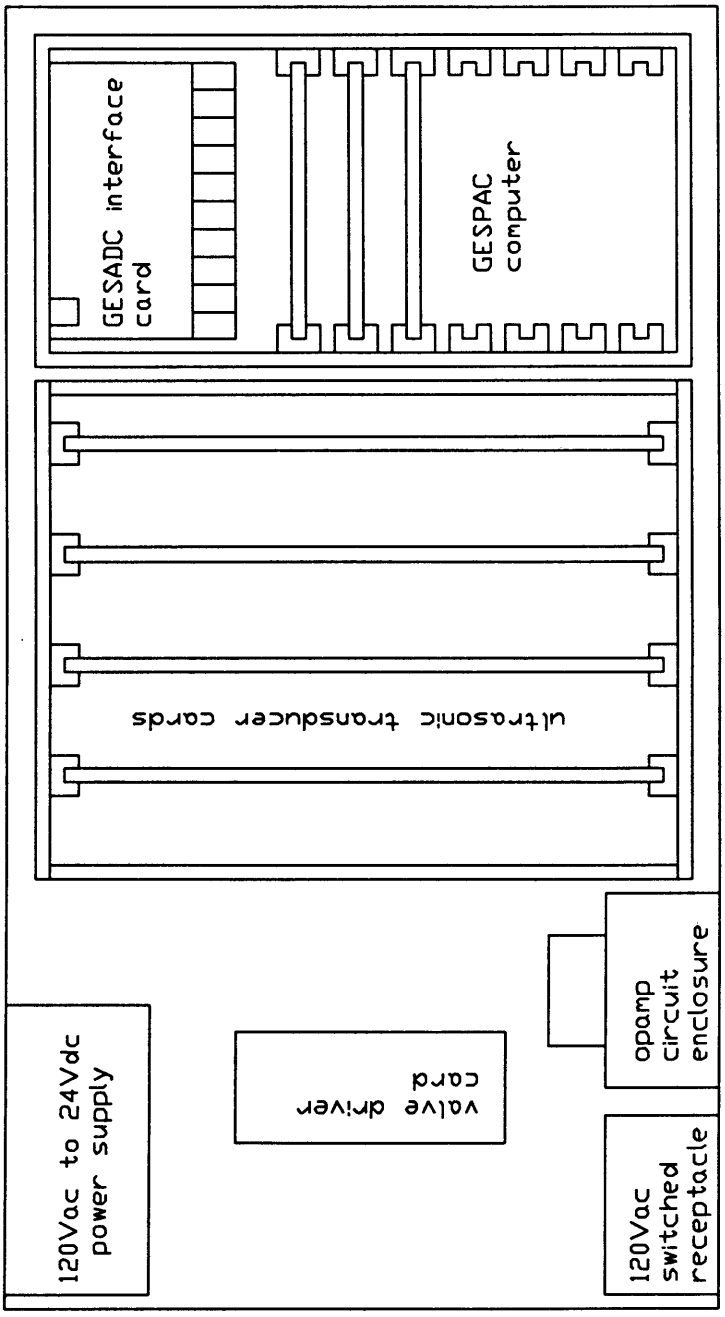

Figure 9. Enclosure components and their locations.

3.11 Proportional Valve Control

GESPAC GESMFI-1 parallel port A outputs control commands to the proportional valve driver card. The eight data bits are divided equally using the four Most Significant Bits (msb) to control the right steering corrections and the four Least Significant Bits (lsb) to control left steering corrections.

A circuit containing three opamps, shown in Figure 10, provides 0 to +10Vdc (00 to F0 hexadecimal values used in program code) for right steering corrections and 0 to -10Vdc (00 to 0F) for left steering corrections.

The four msb's of the GESPAC parallel port connect to one opamp and the four lsb's connect to one of the two remaining opamps. These two opamps output to a summing opamp circuit. The output of the summing opamp connects to the driver card control voltage input.

Table 3 shows the relationship between the input hexadecimal value in the programming code and the resulting output voltage from the opamp.

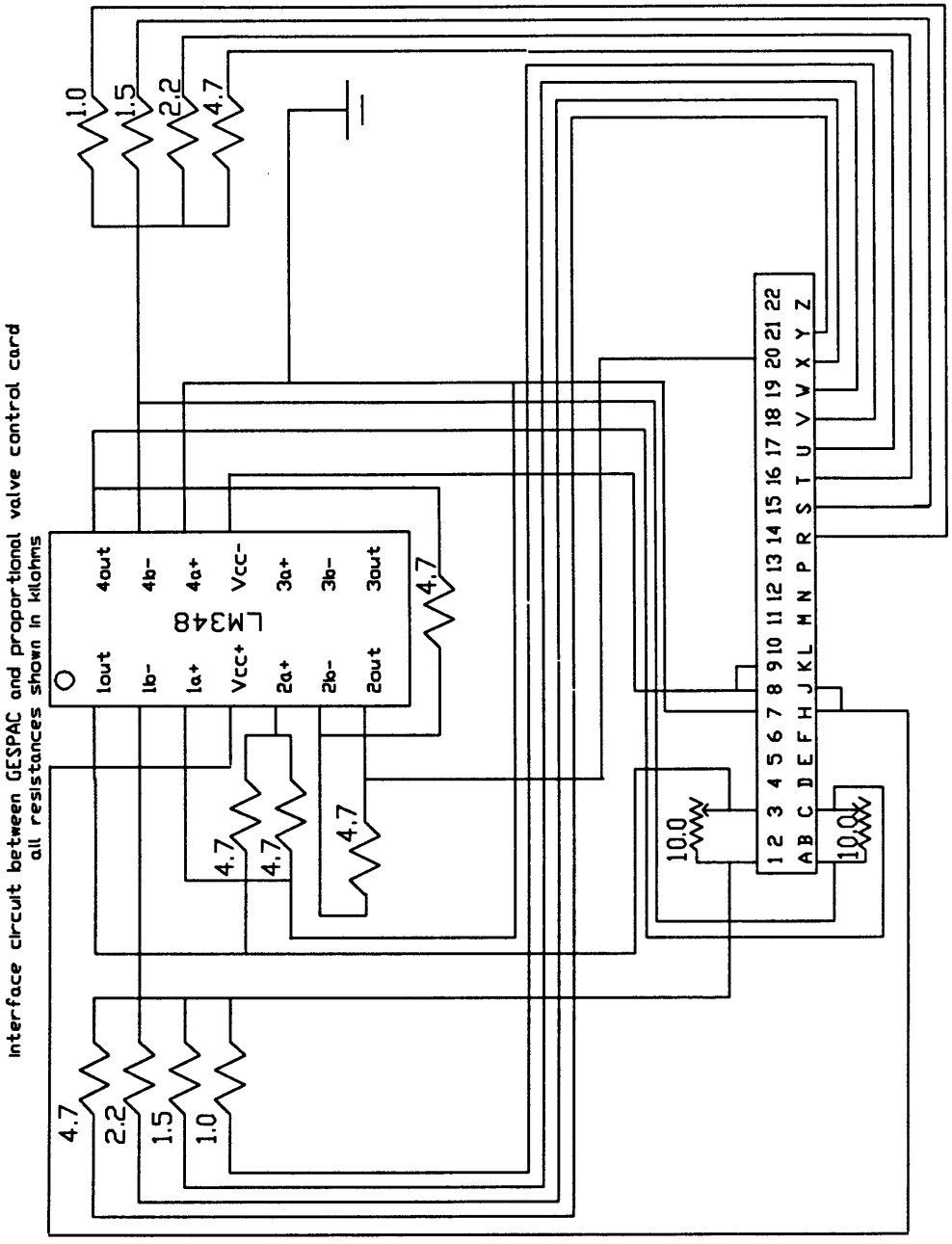

Figure 10. Opamp GESPAC/driver card interface circuit.

Table 3. Shows relationship between digital computer control values, corresponding analog opamp output values, and the resultant vehicle position.

Hex Values	Feed back position	Control Voltage (Vdc)
F0	full-right	-11.15
30	half-right	-5.12
10	center	-2.54
00	half-left	-0.65
0F	full-left	+12.33

When the four lsb's are zero and the four msb's are one the valve adjusts the truck's steering to the right. When all eight bits are zero, the valve is turned off. When the four msb's are zero and the four lsb's are one, the valve adjusts the truck's steering to the left. This is the open loop valve behavior. Varying the four msb's combination while keeping the four lsb's zero varies the right steering rate adjustment. Therefore, the steering correction rate increases as the four msb's combined value increases. The four lsb's and left turn steering corrections respond the same.

Two closed-loop options exist. The first option uses the steering cylinder linear potentiometer as the feedback sensor and the valve driver card as the feedback controller. The

linear potentiometer located in the hydraulic steering cylinder generates the closed loop signal. The four msb's control the right vehicle steering corrections. The four lsb's control the left vehicle steering corrections. Five positions exist in the closed-loop system developed for this research. These positions include, full-left-turn, half-left-turn, straight, half-right-turn, and full-right-turn. The second feed back control option uses the ultrasonic transducers as the feedback sensor and the GESPAC MPU as the controller. This case provides a large degree of accuracy and unlimited steering angle settings.

The second feed back option allows vehicle positioning at specified distances from the ribs. Using the GESPAC as the feed back controller in conjunction with the ultrasonic transducer is an acceptable solution and requires minimal computing time. This option is employed in the experiments described in chapter 5.

3.12 Test Course

The test site, shown in Figure 11, located on the west side, basement level of the Brown Building, Colorado School of

Mines Campus, provides an acceptable environment to test the steering control algorithms.

The characteristics represented in this test site will provide an adequately difficult test pad for the steering control algorithms since it is not as predictable as a typical mine. The west wall is poured concrete and the east wall is brick. The road is asphalt providing positive wheel traction. The course contains both straight and curved walls.

Most mines are developed and laid out in such a way as to make the major design characteristics fairly predictable. This mine predictability simplifies the steering control algorithms. All intersections, crosscuts, and other common features are well surveyed and mapped.

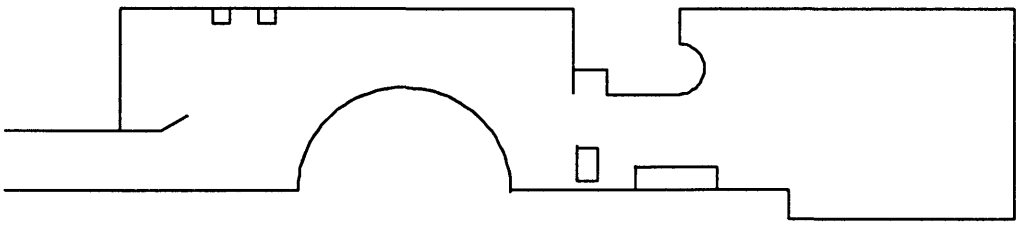

Figure 11. Map of test site.

Chapter 4

PERCEPTION

4.1 Data Interpretation

The ultrasonic transducer board generates an analog output voltage inversely proportional to the distance measured. The Analog to Digital Converter (ADC) board converts this to a proportional digital linear value ranging from 0 (14 feet) to 4096 (2 feet) representing the transducer measurements. The transducer data is transposed with respect to the side of the truck it was obtained and adjusted with respect to an X-Y reference frame. Forward truck motion along the X-axis is considered positive and truck motion to the left along the Y-axis is considered positive. Perception tests performed used a 90 degree incidence angle.

4.2 Perception Algorithm

A perception program, plt50.c, performs the necessary transformations to produce a graphical representation of the ultrasonic transducer data. A discussion of each function included and its purpose follows.

main

The main function in plt50.c consists of a loop that acquires new data file names from the user until they are finished, transposes the data files, and presents the results as a graphical map.

fix_theta

The first function accessed determines the present, local, steering angle and updates the overall global vehicle angle using the steering angle sensor data. Fix_theta scans the data for any out-of-bound readings and adjusts it so right turns generate a negative value and left turns generate a positive value. Fix_theta adds the resulting number to the global value, converts it to radian measure and sets it between the limits of zero and two pi.

dig_to_feet

Dig_to_feet adjusts the ultrasonic transducer readings according to the local and global positioning of the vehicle. Dig_to_feet adds the vehicle width to the readings

transposing them to the vehicle centerline. Dig_to_feet scales the final value obtained to represent an equivalent foot unit of measure.

running_ave

A weighted running average removes any inconsistent noise from the sensor readings, which may result from echo, or weak signal response from surfaces with low reflectability. This weighted average removes single ghost readings without effecting true readings of existing objects encountered. The running average window size is a constant adjusted by the user. The optimum window size may change with varying surface conditions experienced in different mines. Running_ave includes each new successive reading and removes each successive oldest reading in the window. Running_ave adjusts the data with a weighting value that is greatest for the newest reading and decreases as the data increases in age. Running_ave uses linearly varied weighting, so for a window size of ten, running_ave multiplies the newest reading by a weighting factor of ten. Running_ave multiplies the next older sensor reading, previously obtained, by a weighting factor of nine. This continues in a similar fashion,

incrementing to the next oldest reading and decrementing the weighting factor by one until running_ave adjusts all the data within the window. Other weighting methods may be more effective, but this method provides acceptable results for the purpose of this program.

actual_path

The algorithm assumes a constant velocity over the entire test run. Given this constant velocity, and a sampling rate of 10 Hz, the algorithm computes an incremental vector. The incremental vector is a constant distance that the vehicle travels between each line of data collected. Every tenth of a second, the ADC card obtains a reading containing sensor data including, the four short range ultrasonics on the corners of the vehicle, and the steering angle obtained from the in-cylinder linear potentiometer. From this data, actual_path determines global X-Y coordinates of the vehicle as it travels through the test site. Actual_path adds or subtracts X and Y coordinates from the global position of the vehicle depending upon its present steering angle and direction.

out_of_bounds

Out_of_bounds scans ultrasonic readings to detect voids. A void is referred to as out-of-bounds in the software code. Thirteen feet provides the cut off range for this research, since fourteen feet is the maximum distance that the ranger can detect objects. When voids occur in the data, an offset multiplier distinguishes them from those indicating objects.

make_coord

Make_coord transposes the ranger data according to the articulation angle of the vehicle, and the global coordinates of the vehicle within the map frame reference. Thus, make_coord converts the resulting data to X-Y map coordinates for display on the computer monitor.

local_map

Local_map presents the results on the monitor upon completion of all transposing. This function plots the actual centerline the vehicle travelled and the two front ranger readings depicting sensed objects during the test run.

4.3 Mapping results

Figure 12 shows the results of the data transposing and mapping program. The results show a representative map produced with the ultrasonics that compares well with the surveyed map. These results support the use of ultrasonic rangers to describe the test environment. The steering control algorithms discussed in Chapter 5 do not use this transposing, perception algorithm to control the vehicle.

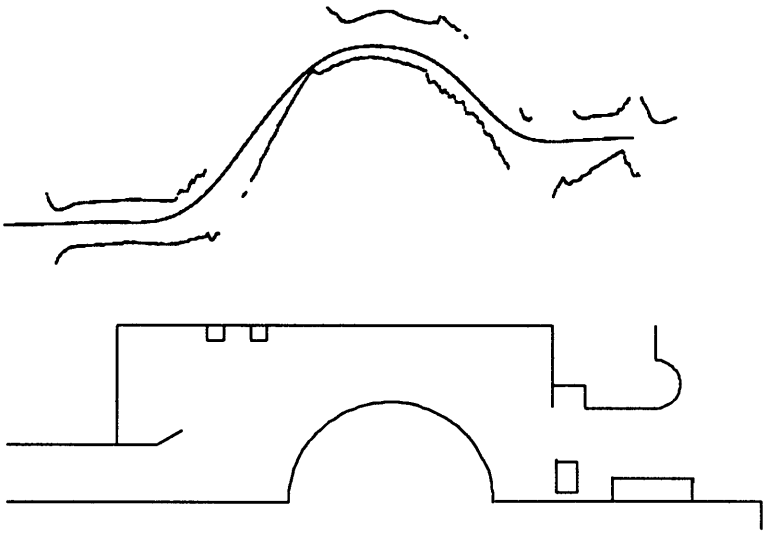

Figure 12. Surveyed map -VS- computer generated map.

Chapter 5

STEERING CONTROL ALGORITHMS

5.1 Observations

Experimental observation revealed that the rear half of the vehicle follows the front half along curved and straight paths. Therefore the steering algorithms can guide the front half while collision avoidance algorithms monitor the rear position.

The steering algorithm can either follow one wall or the centerline between two walls. The first algorithm maintains a safe maneuvering distance between the truck and the wall. The second algorithm steers the truck within a desired zone between two walls. The method presently applied combines wall following and centering to produce a smoother and more versatile solution. Steering correction activates when the truck is:

-too close to right wall

-too close to left wall

-too far away from left wall

-off center

A center-zone defined by limits produces a smoother trajectory with fewer computations. By maximizing the safe wall following distance the algorithm defines a center zone. This effectively pushes the vehicle to the center of the test course as it attempts to stay away from the walls. The control algorithm allows wall following when appropriate. The test site contains voids (out-of-range ranger readings) much like a mine drift intersection, but the west wall has fewer voids. Therefore, the truck follows the west wall when it detects voids on either side.

5.2.1 Control Program Evolution

The transducer incidence angle is 15 degrees during the first phase of testing. A program developed, follow.c, analyzes transducer data and controls the proportional valve to maintain a set distance between the transducer and a moving object. The object is a person moving toward and away from the sensor. The results verified the proper hook-up and operation of the equipment involved.

This is a simple closed-loop behavior in which the computer sends steering correction signals to the proportional

valve to maintain a set distance away from the individual moving toward or away from the sensor. A two foot safe following zone prevents continual steering corrections since the computer attempts exact settings. The logical break down of the first steering control program is shown below.

```
if (front left > MAX)
     steer toward left.
if (front left < MIN)
     steer toward right.
```

Front left is the front left ranger reading, MAX is the maximum safe wall following distance, and MIN is the minimum safe wall following distance.

Test runs conducted using this program to maintain a safe wall-following distance along the west wall as the vehicle progressed forward produced poor results. The vehicle oscillated in a manner that prevented it from successfully following the wall. Figure 13 shows the safe wall following zone between the wall and the test vehicle.

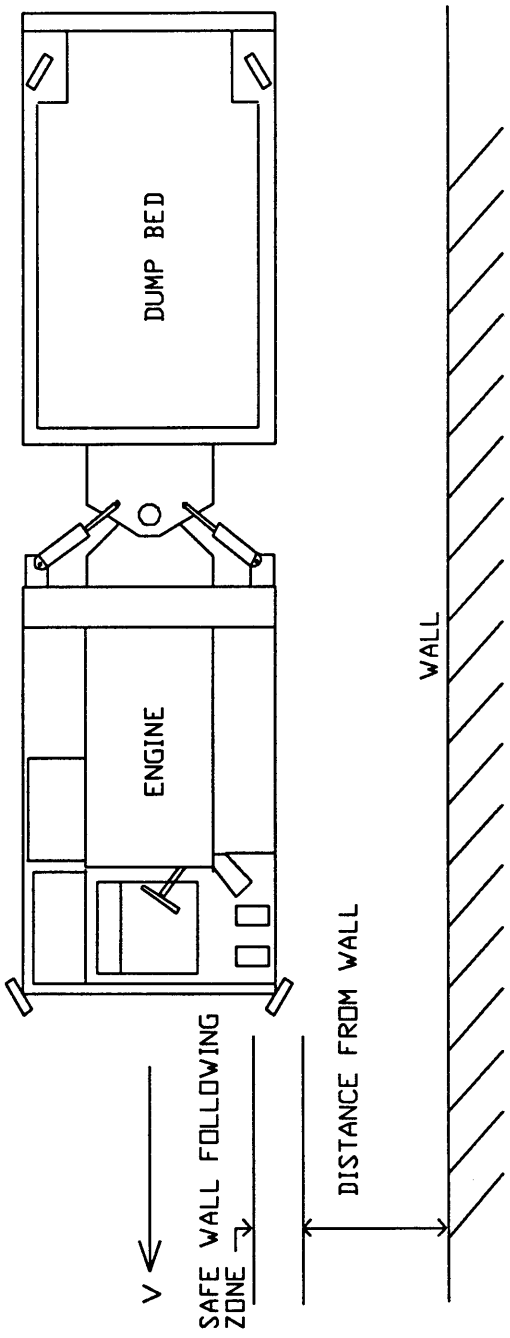

Figure 13. Safe wall following zone relationship between test vehicle and wall.

5.2.2 Experiment Series I

During this set of tests, the incidence angle is set to 35 degrees and the safety zone is 1.62 feet. The algorithm observes the front left sensor performing a single wall following routine.

Table 4. Experiment Series I results.

Experiment	Error (ft.)	Total Points	Points in Error	Velocity (ft./sec.)	% Points in Error
FE1	2.44	1528	994	0.76	65.05%
SE1	2.20	381	361	1.63	94.75%
SE2	2.31	387	339	1.59	87.60%
SE3	2.67	433	409	1.44	94.46%
SE2a	0.95	638	216	0.96	33.86%
CE2a	0.60	1008	500	1.01	49.60%
CE2	0.53	764	394	1.32	51.57%

The results of experiment series I are presented in Table 4. FE1, shown in Figure 14, is a test along the full length of the test course including both the curved portion and the straight portion. The SE(n) series represent tests run along the straight section of the course and the CE(n) series

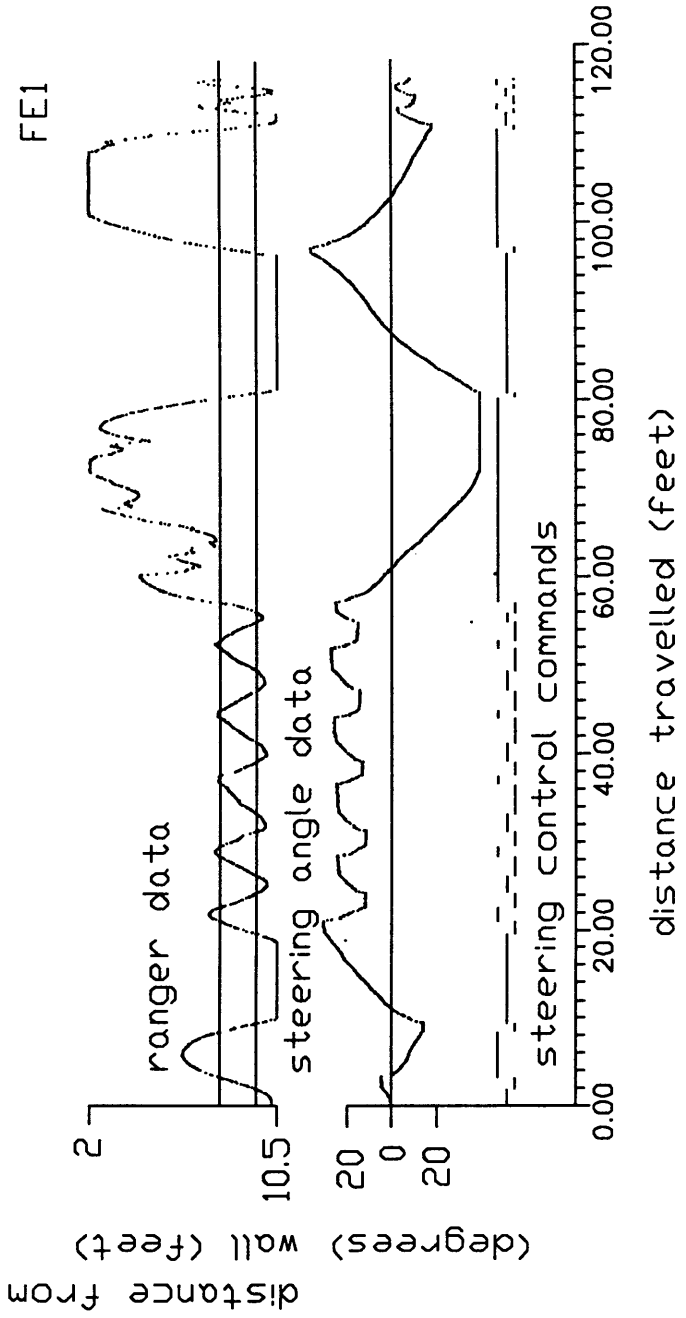

Figure 14. Full length test run (FE1).

represent tests run along the curved section of the course. This data suggests that slower velocities during the experiments produce better steering behavior results. The top signal represents the steering angle of the vehicle during a test run. The middle signal represents the ultrasonic transducer data. Two horizontal lines have been added to this signal to represent the safe wall following zone. The bottom signal represents the steering commands induced by the computer. Including all three signals in one figure shows the effect the ranger readings have on the steering commands which correct the steering angle until the vehicle returns to the safe wall following zone.

An interesting feature in this figure is the low ranger values experienced between 80 and 120 feet along the distance travelled axis. The first low reading seen occurs when the vehicle is too close to the wall and the steering is adjusted toward the right to compensate. The vehicle experiences such a large over compensation in steering adjustment to avoid collision with the wall that it goes beyond the maximum safe wall following distance. This caused the vehicle to steer back toward the left again over compensating and resulting in an under damped system with little or no stability.

Figure 15 represents data acquired during a test run along a straight wall. The initial steering angle data indicates the vehicle was turned sharply to the right (right turns correspond to negative degrees) when the experiment commenced. Shortly after the vehicle begins to progress forward, the wall following distance jumps sharply out of the safety zone and steering corrections are immediately initiated. During the next ten feet of travel, the vehicle is redirected within the safety zone and steering corrections are discontinued until the vehicle again veers off course.

The safe wall following zone has a band with of 1.62 feet and the error of the vehicle falling outside of that zone is +/-1.67 feet. The average velocity during experiment series I is 1.24 feet per second. The steering response deteriorated as the vehicle warmed up due mainly to the age of the hydraulic oil. Figure 16 represents typical results experienced when the vehicle followed the curved section of the test site.

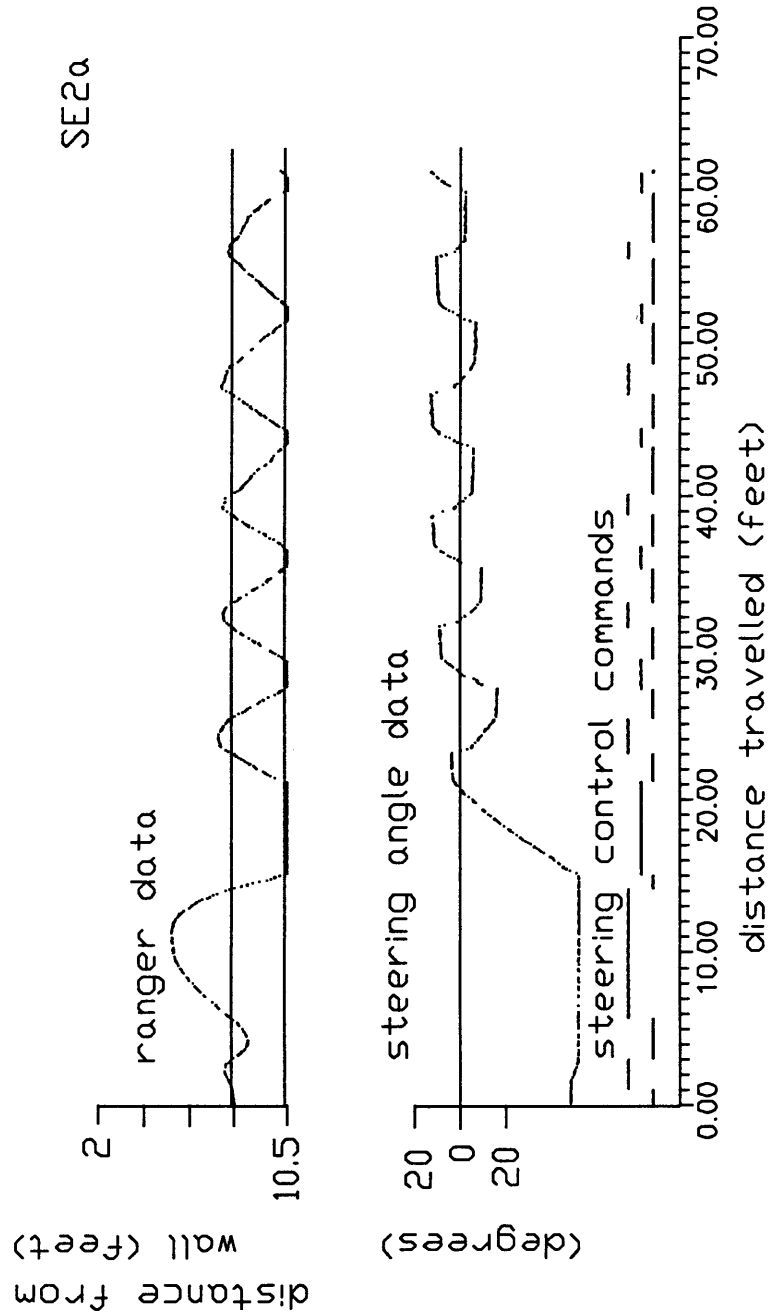

Figure 15. Straight section test run (SE2a).

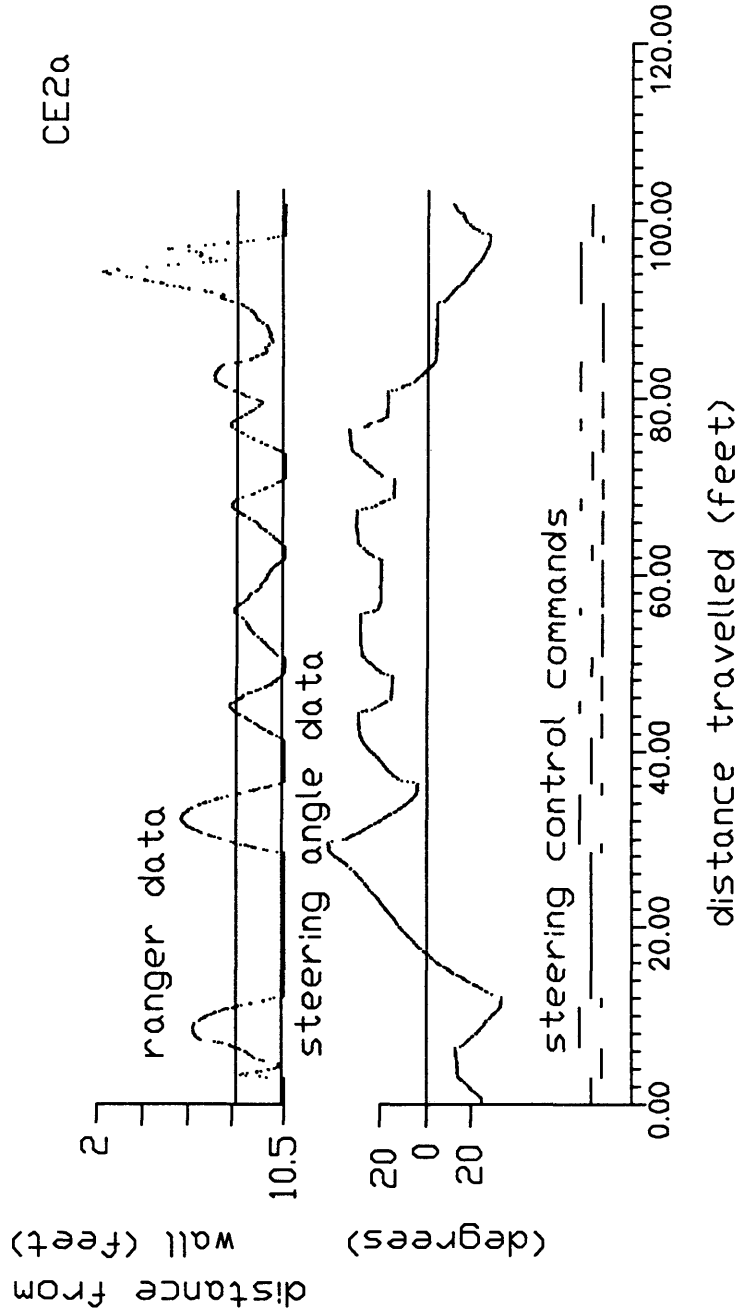

Figure 16. Curve section test run (CE2a).

5.2.3 Experiment series II

The results experienced in experiment series I led to the following observations:

- -need to develop a method to dampen the steering corrections and minimize the amplitude of the vehicle oscillations.
- -need to develop an algorithm to follow both walls.
- -the algorithm will favor west side wall over the east side wall when only one wall is detected.
- -the left wall will never be allowed to go beyond a detectable distance.
- -developing an algorithm that forces the vehicle to travel between both walls will not only maintain safe wall following distance, but will also force the vehicle to travel in the middle of the test course.

A new algorithm, follow2.c was developed from the above stated observations. The algorithm takes into consideration data from both front ultrasonic transducers. It always favors the front left transducer, so if the course is too wide for both rangers to detect objects, it will stay closer to the left ranger (west side wall). The new algorithm is outlined below.

```
if (front left or front right < MIN) or (front left > MAX)
     if vehicle is turned to right.
```

```
     and front right < front left
          readjust steering toward left.

     and front right > front left
     and front right < front right last
          maintain present steering angle.

     and front right > front left
     and front right > front right last
          readjust steering toward right turn.

if vehicle is turned to left

     and front left < front right
          readjust steering toward right.

     and front left > front right
     and front left < front left last
          maintain present steering angle.

     and front left > front right
     and front left > front left last
          readjust steering toward left.
```

Follow2 successfully completed four test runs. Two full length tests, one straight, and one curve section test are included in experiment series II. This new algorithm produced better results than the wall following algorithm tested in experiment series I, but as the vehicle reached normal operating temperature, the steering response became very sluggish. The hydraulic steering fluid in the test vehicle was twelve years old and did not maintain good viscosity as it reached operating temperature. Table 5 and Figure 17 present the results of this experiment series.

Table 5. Experiment Series II Results.

Experiment	Error (ft.)	Total Points	Points in Error	Velocity (ft./sec.)	% Points in error
FE3	1.17	1530	892	0.97	58.30
FE4	1.74	1343	957	1.11	71.26
CE5a	1.48	899	619	0.74	68.85
SE5a	1.32	665	416	0.93	62.56
CE9	1.07	780	486	0.86	62.31

The full length test run represented in Figure 17 shows noticeable improvement when compared with the full length test run presented in experiment series I. Changes introduced in the steering control algorithm dampened the steering response and produced a smoother run along the straight section of the test course. A smaller magnitude of error resulted between 60 and 100 feet when comparing both the ranger data and the steering angle data between test run FE3 (Figure 17) and FE1 (Figure 14). An improvement in the percent of points in error is seen in test run FE3 as compared to FE1 when comparing the values presented in Tables 4 and 5. Test run FE3 maintained an average velocity of 0.97 feet per second, a 22 percent increase in velocity over that experienced in test run FE1.

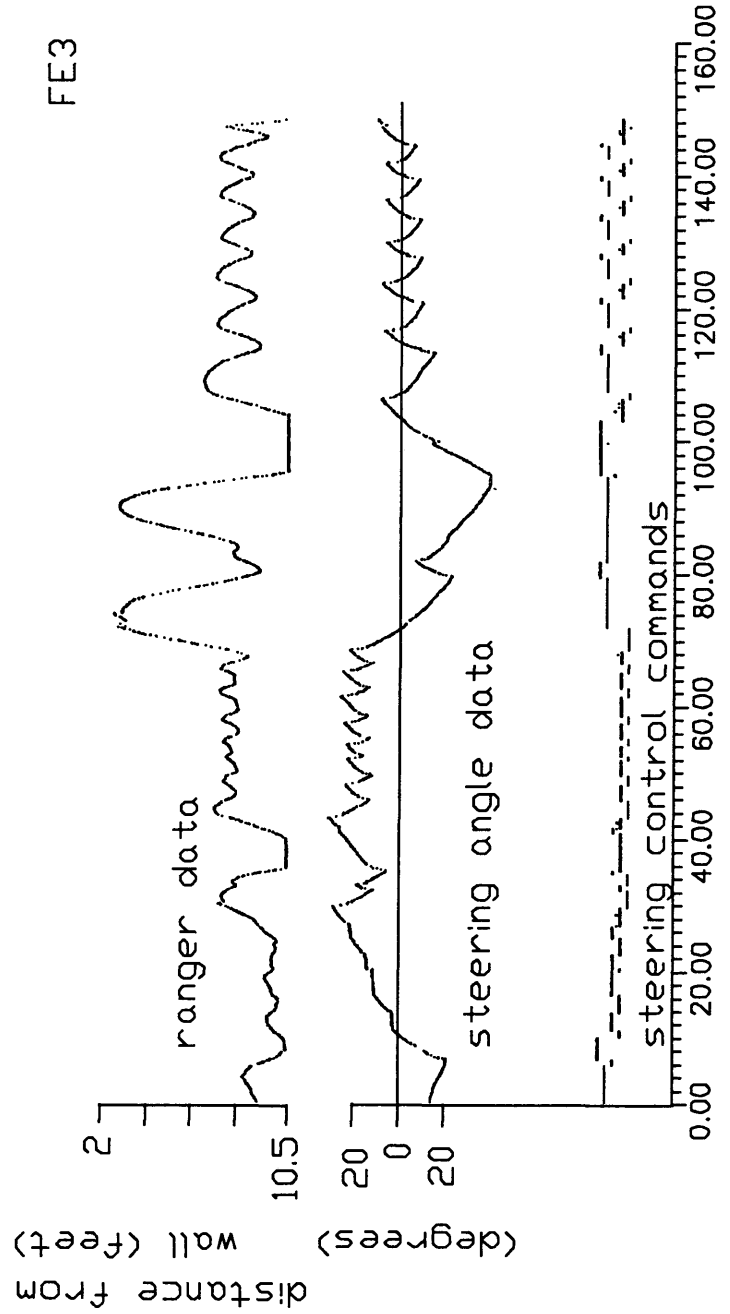

Figure 17. Full length test run (FE3).

5.2.4 Experiment Series III

The algorithm described in section 5.2.3 was tested again after changing the hydraulic fluid in the vehicle. The new oil has a positive effect. Follow2.c successfully complete five full length test runs. When the operating temperature of the hydraulic oil was reached, the vehicle still successfully steered through the test course. Table 6 shows the results of the test runs. The average error for the full length test runs presented in Table 6 is lower than the error presented in Tables 4 and 5. This is especially apparent when observing experiments FE6, FE8, and FE10.

Table 6. Experiment Series III Results.

Experiment	Error	Total Data Points	Data Points in Error	Velocity (ft./sec.)	% Points in Error
FE6	0.58	1540	537	0.97	43.87%
FE7	0.91	1498	1086	0.99	75.89%
FE8	0.78	1431	398	1.04	27.81%
FE9	1.05	931	733	1.60	78.73%
FE10	0.58	1361	464	1.10	34.09%

The percent of steering correction is noticeably lower in

experiment series III in comparison to experiment series II.

FE7, seen in Figure 18, has a higher percent of error because the vehicle was placed a greater distance from the wall, outside the safe wall following distance, before beginning the test run. FE9 shows a similar amount of error, which resulted due to a higher velocity experienced during the test.

Figure 19 shows the results of full length test run FE9 with the truck in second gear and idle rpm. Note that the magnitude of the oscillations is greater at higher speeds. This may be due to the fact that smaller steering corrections need to be initiated for higher velocities than are normally used for lower velocities. The algorithm assumes a constant velocity, and overcompensates as the velocity increases.

The amplitude of the oscillations represented in the steering angle signal increases as the vehicle travels along the curved section and again after it stabilizes on the straight section of wall. The transition between the curved wall and the straight wall occurs between 60 and 110 feet.

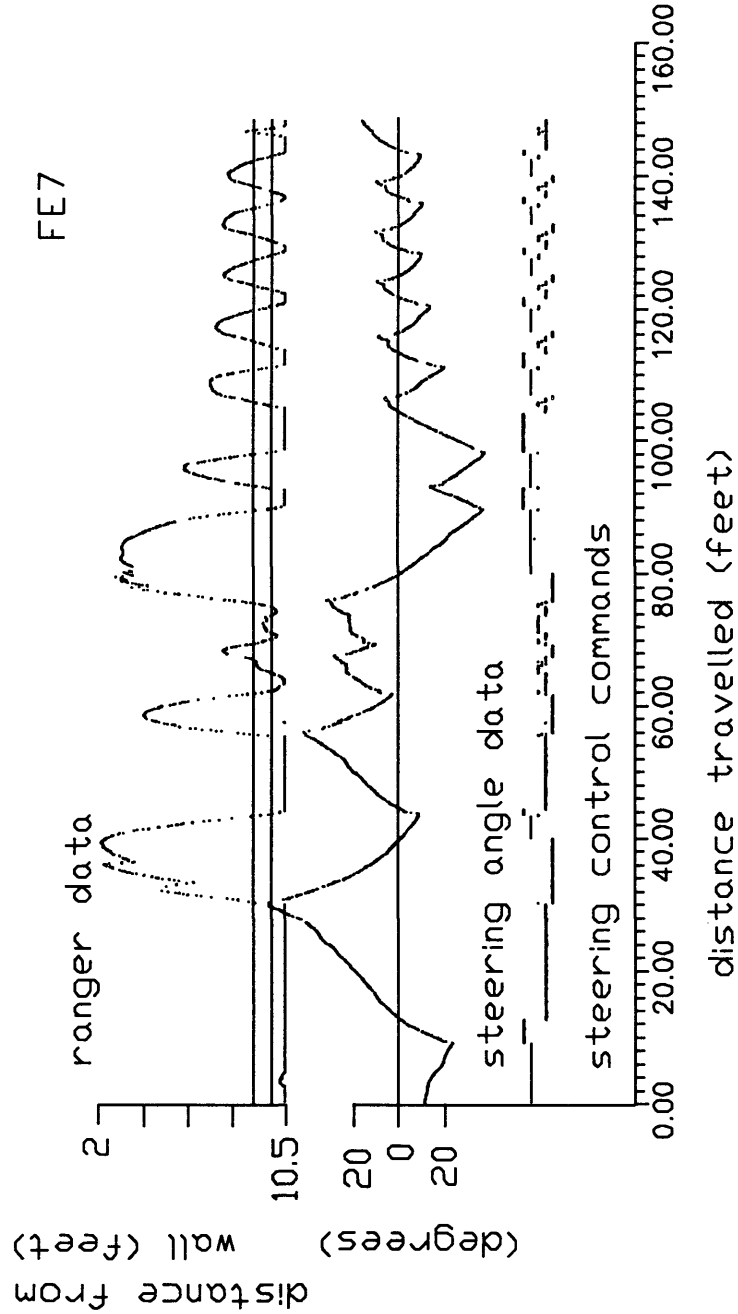

Figure 18. Full length test run (FE7).

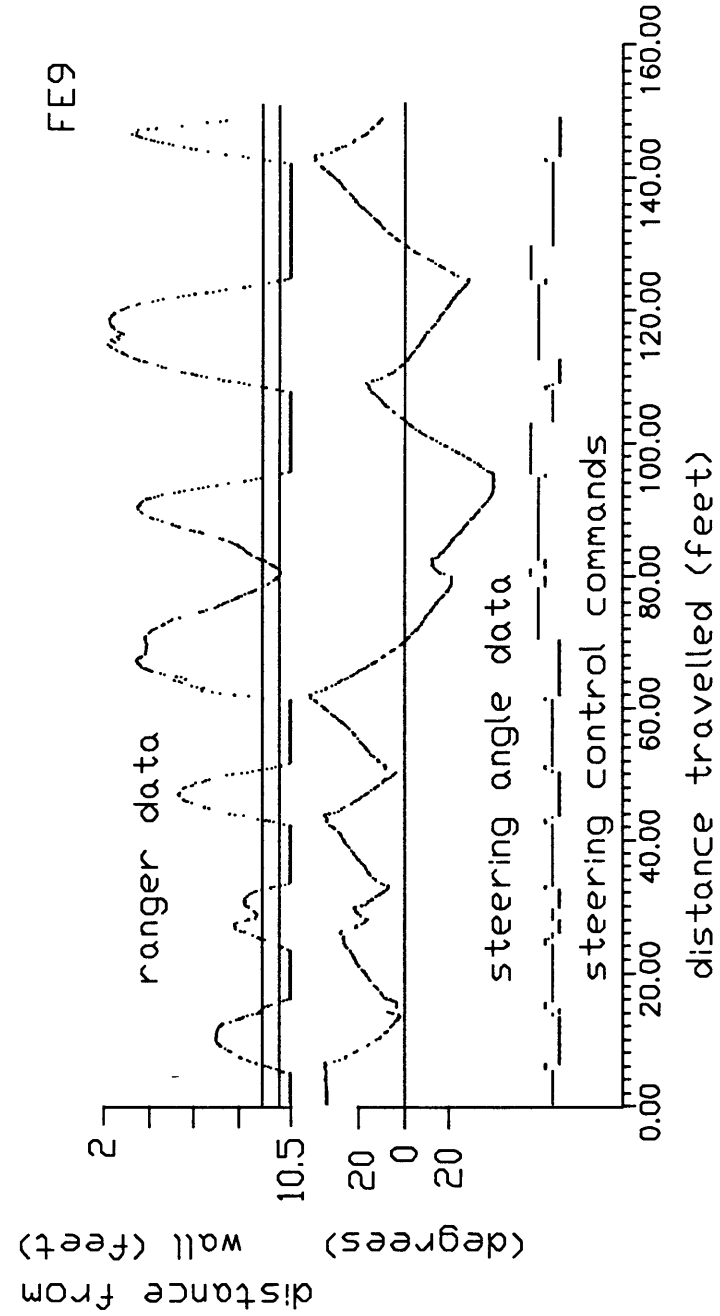

Figure 19. Full length test run (FE9).

The steering control program tested in experiment series II and III provides adequate control of the vehicle along both a curved and straight section of wall. The average velocity maintained during these tests is 1.03 feet per second. The safe wall following zone has a band width of 0.69 feet. The average error outside the safe zone for experiment series III, excluding FE7 and FE9, is +/-1.37 feet. The incidence angle was maintained at 45 degrees for experiment series II and III providing a valuable "look ahead" capability that enables steering adjustments to be made while considering the environment ahead of the vehicle.

The steering compensation is effectively dampened in experiment series III as compared to series I experiment FE1 where the vehicle's oscillation amplitude was much greater as it pulled out of a sharp right hand turn onto a straight section of wall. The oscillation frequency decreased when the vehicle velocity increased. Test run FE9 produced 60 percent fewer oscillations than that experienced in tests FE6, FE8, and FE10.

5.3 Conclusions

The system steered the vehicle through the test site during several experiments without collisions at an average speed of 1.03 feet per second. In experiment series II and III, the truck followed the original 1.62 foot wide center zone with an average error of +/-0.71 feet. The transducers operated successfully at a 45 degree incidence angle which provided a valuable "look ahead" capability providing better steering response. The distance between the wall and safe zone is three feet +/-0.71 feet. Test run, FE8, kept the vehicle in the safety zone 72% of the time. This research supports the proposed steering control system for the articulated underground mine haul test vehicle.

Figure 20, seen on the following page, is an enlarged section of Figure 15. Five areas are represented in Figure 20 by the letters A, B, C, D, and E. These areas represent different steering correction states the vehicle experiences while controlled by the steering algorithm. The vertical lines in this figure separating the different areas represent the beginning and ending points of the steering corrections described herein.

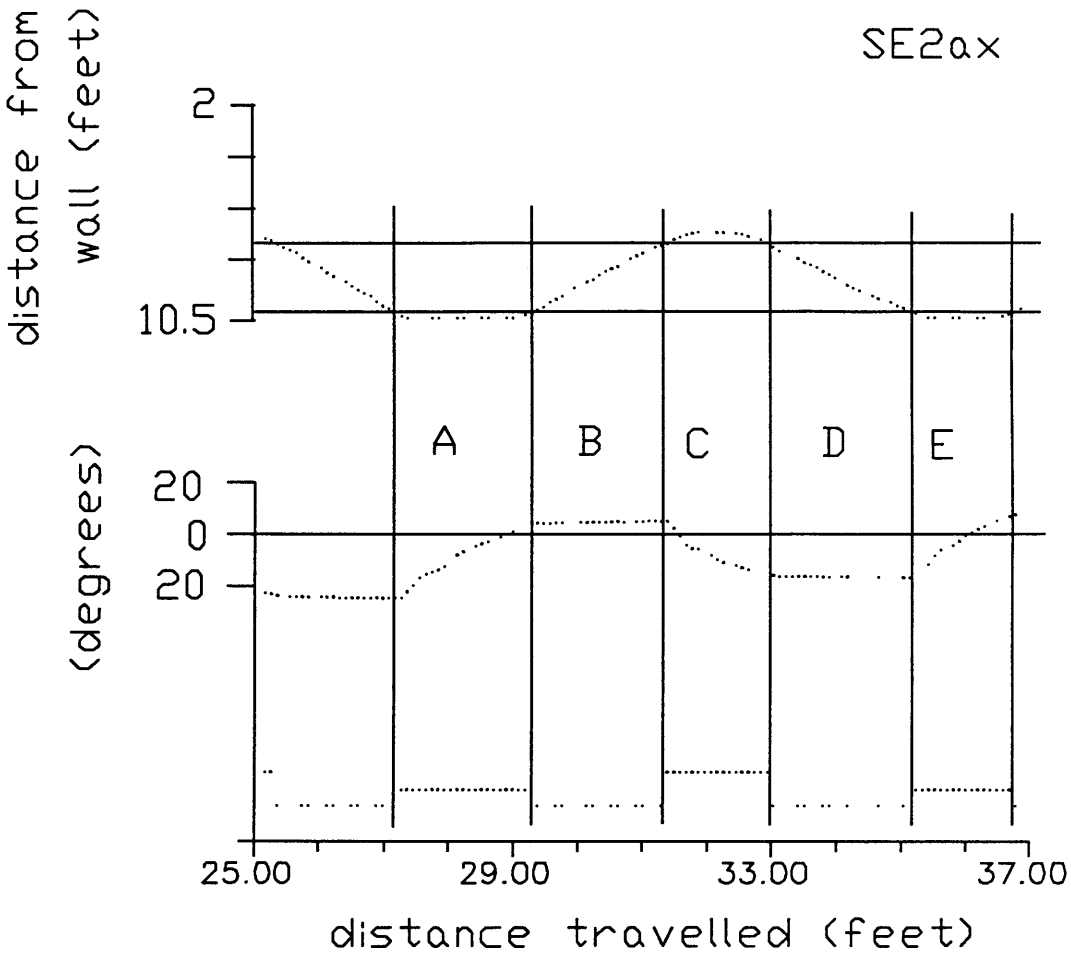

Figure 20. Enlargement of test run (SE2a) segment.

Area A represents a state when the vehicle is too far away from the wall and a void is detected. At the beginning of state A the detected wall becomes a void and the steering algorithm adjusts with a left turn to direct the vehicle back toward the wall. At the end of state A or beginning of state B the vehicle has relocated within the safe wall following zone, consequently, the wall can again be detected. During state B, no steering correction is imposed, represented by a constant steering angle, since the vehicle is within the safe wall following zone. At the beginning of state C the vehicle has travelled outside the safety zone, this time approaching too close to the wall. At this time a right steering command is imposed to correct the problem. In state D the vehicle again returns to the safe wall following zone and the steering correction is discontinued.

This control scheme produces a result that is prone to oscillate. Including a safe wall following zone contributes to minimizing continual corrections which would be experienced if a dead reckoning routine were used. A dead reckoning routine would enable the vehicle to follow a tighter path which may be more desirable. Dead reckoning would reduce the amplitude of the oscillations, but would increase the

frequency. A combination of a safety zone and dead reckoning may produce a system capable of lower frequency and oscillation amplitude. This approach will put a priority on maintaining the vehicle in the safety zone and while that condition exists attempt to maintain a constant wall following distance. This affect may be accomplished by determining an actual and desired direction vector for the vehicle and adjusting the course by a fraction of the difference between those two vectors. This solution was tested in experiments performed by St-Amant, of *Ecole Polytechnique de Montreal*, Montreal, Canada. Future experiments may prove this approach to be more effective in eliminating the frequency and amplitude of the oscillations.

Accessing an on board map while updating physical location with local sensors can also aid in minimizing oscillation frequency and amplitude. This application requires more descriptive information and computing power to incorporate. It is desirable to avoid such a solution, if possible, in order to develop a more versatile system that requires minimal user information to perform its task. However, the behavior of the vehicle might be drastically improved with a map and local sensor steering system.

An effective steering system for the described test vehicle was developed and tested and the results have been presented. This system may be transferable to similar mining vehicles presently used in underground mines. More testing and development is necessary to improve the performance of the present system before testing it in an actual mine. This research supports the steering system presented as a valid system worthy of continued research. Chapter 6 includes a thorough evaluation of the advantages and disadvantages of each of the components used in this system.

Chapter 6

RESULTS AND RECOMMENDATIONS

6.1 Component Results

Vehicle

The test vehicle chosen for this research provided a representative prototype for developing an automatic steering system, however, it should be noted that vehicle behavior in the real world is non-ideal. Several factors limit the control of this vehicle. Proper steering response requires the consideration of:

1. Hydraulic oil temperature, age, and density
2. Engine idle rpm
3. Vehicle velocity
4. Operating gear
5. Steering correction necessary

The steering response time increased as the vehicle warmed to the operating temperature which would require a

control system able to adapt to these changing conditions. The steering response time increases and decreases inversely with engine rpm. This requires a dynamic level of control when operating at a constant velocity in 1st gear at a high operating rpm or 2nd gear at a low rpm. In these two cases, the steering response time is noticeably different and requires consideration in the control algorithms.

In underground mines many conditions exist that contribute to wheel slippage, altering the predicted behavior of the vehicle under desirable, ideal conditions such as: loose dirt or rock and muddy or wet conditions. These conditions require a system that is not limited to controlling the vehicle under ideal conditions, but can easily adapt to changing conditions. The most important conclusion to draw from the test vehicle selected and almost all mining equipment is the unpredictability and poor repeatability of their task behavior when placed in a dynamic and continually changing mining environment. It is not feasible to rely upon the vehicle to behave the same way when traversing through the same test site multiple times. Consequently vehicle control must be dynamic and adaptable.

Computer

The GESPAC provides real-time analysis and control necessary for the dynamic requirements already described. It is not, however, possible to accomplish real-time debugging when refining the software on the GESPAC system. The completed code must be downloaded onto an EPROM chip in order to avoid reloading that code each time the system is rebooted. This process was not undertaken during this research and the procedure necessary to accomplish this task can not be presented. A host PC communicates with the GESPAC to provide the software development and communications capabilities with the system. Including a monitor, keyboard and disk drive in the GESPAC hardware set-up would provide a more user friendly environment in which to develop this system.

The GESPAC system is a Motorola 68030 series microprocessor, consequently, software development was relatively simple and followed the common ANSI standard. The computer is packaged in an industrial enclosure and it is robust and very reliable. The processing speed is acceptable for this application, and will accommodate for more data acquisition and component control with out difficulty. This

system is recommended for similar project applications requiring high processing speed robust design, a small enclosure size, and reliable performance. Consulting with GESPAC representatives was accomplished over the phone and proved to be invaluable.

Ultrasonic transducers

The Milltronics ST25C4 ultrasonic transducer provides reliable data for environment perception. the transducers are packed in a strong plastic enclosure to protect them from harsh environments similar to underground mines. Each transducer has a separate driver card which produces a pulse rate of 11Hz. This slow pulse rate limits the maximum vehicle velocity. It is recommended that the boards be altered to provide a higher pulse rate and possibly including all 4 transducer's circuitry on one computer card to minimize space requirements. Transducer calibration proved to be time consuming and sometimes confusing. The literature describing calibration procedures is brief. Tools needed during this procedure were the object to detect, a multimeter, screw-driver, tape measurer (15 feet), and two persons. The area in which the transducers are calibrated should provide a clear,

straight and wide path (10 feet) at least 18 feet long. Once calibration is completed it is highly advised that the system be retested at minimum and maximum distance settings (high/low distance calibration) for repeatability. During calibration procedures each setting (high/low distance) appeared to effect the other. Consequently, it was necessary to "tweak" the high and low transducer adjustments to obtain the minimum possible reading setting and the maximum possible setting. The literature provided for this sensor required technical background to accomplish hook-up and calibration, but the transducers provided reliable and valuable data and are recommended as an acceptable component for similar projects.

Proportional valve

The proportional valve selected provided reliable and adequate control of the test vehicle steering. The literature accompanying this device was extremely brief and did not provide adequate information to complete the necessary connections to control the valve. Technical background and logical deduction was necessary to properly install this valve. Though limited in control information, the valve is acceptable for this project application.

Linear potentiometer

The cylinders equipped with a linear potentiometer provided reliable and repeatable information describing the vehicle steering angle.

Op-Amp circuit

The opamp circuit built to interface the GESPAC to the valve driver card performs well. The design, included in this thesis, is simple, low in cost, and reliable. The micro chips and other necessary components are available at several electronics supplies stores.

6.2 Event Locating

Continued research on this project should focus on developing event recognizing and location updating algorithms. The events being crosscuts, intersections, corners, etc. By combining predetermined data acquired while travelling through that course it is possible to maintain and update the location of the vehicle in the mine. This location technique can be utilized to effectively guide the vehicle from one point in

the mine to another while experiencing multiple crosscuts and intersections.

6.3 Radio Control

Wireless remote control of vehicle functions including: steering, shifting, reverse/forward, throttling, braking, and engine shut-off, should be incorporated into the test vehicle. This is recommended for two reasons, it would be a good safety precaution to have off-board control of the vehicle when it is initially being tested with no personnel on-board in case any components on-board fail. Also, when a vehicle of this type is functioning in an underground mine it may experience difficulties that require human control to correct. Some of these difficulties may be ones correctable via remote radio controls from a safe operating location.

6.4 Safety Pull Rope

A safety shut-off pull rope should be installed around the perimeter of the vehicle so it can be easily accessed and deactivated from any location around the vehicle.

6.5 Wheel Encoders

Wheel encoders should be installed on all four wheels of the vehicle and a data fusion algorithm developed to combine data from all four encoders should be included to develop an acceptable vehicle velocity and location sensor. By combining recognized events and wheel encoder information, the location of the vehicle in the mine should be accurately determined.

6.6 Doppler Radar

A Magnavox doppler radar should be employed to detect truck velocity and position at any time in the test site. The radar is reliable for the velocity range of the tests performed in this project, however, much difficulty has been experienced in developing a reliable frequency to voltage converter. The raw data received from the radar is a frequency signal. This must be converted to an equivalent voltage representation to be accepted by the GESPAC ADC. It has been possible to acquire raw data from the transducers and truck steering angle sensor while assuming a constant velocity and still produce representative maps of the test site.

6.7 Long Range Transducer

A long range transducer, manufactured by Milltronics, measuring distances up to 180 feet has been employed to monitor the area in front to the truck to detect obstacles or corners. This will enable corrections to be made before the truck encounters the changing environment so as to adapt its course of travel in a more smooth and desirable manner. This transducer has a 6.5 degree field of view, and will provide the capability to detect obstacles ahead of the truck at a distance that will provide sufficient avoidance or stopping time.

6.8 Actuators

Actuators to control shifting, throttling, braking, and dumping should be specified and installed to complete the necessary tasks to render the vehicle autonomous. These functions would be necessary before incorporating Radio Control.

6.9 Machine Health

An MPU board dedicated exclusively to machine health should be included in the system and an algorithm to analyze that data should be developed. With the GESPAC system this board can be labelled with an interrupt address so when the health of the machine is questionable the main MPU will give its attention to the secondary MPU and determine if the problem existing requires immediate shut down or just a minor adjustment, otherwise, the main MPU will dedicate itself to steering control.

6.10 Final Remarks

The technology developed as a result of this project can be applied to various types of mining equipment besides haul type vehicles. It has been suggested that this technology be applied to LHD's and possibly combined with teleoperation to be used in open-stope mining environments.

Combining ultrasonic guidance with teleoperation for use in open stope ore extraction is a very foreseeable future development worthy of research consideration. This statement

is made with consideration of open-stope mining methods gaining favor internationally due to lower operating costs. The one problem this method does introduce, however, is the danger related with open-stoping and the inability to include personnel in this environment. Teleoperation can provide a remote operator to control the vehicle in the stope and ultrasonics can be employed to safely guide the vehicle in drifts while hauling the ore from the stope to the draw point. This control scheme would enable one person to operate a fleet of LHD's from a safe control environment and could possibly revolutionize the future of open stope and other mining methods.

Chapter 7

REFERENCES

Denning Mobile Robotics, Inc.
21 Concord St., Wilmington, Massachusetts

Elfes, A., December 1986, "Sonar-Based Real-World Mapping and Navigation," IEEE Journal of Robotics and Automation, vol. RA-3, No. 3, June 1987.

Everett, H.R., Gilbreath, G.A., January 1989, "Robart II, A Robotic Security Test bed," Technical Document 1450, Naval Ocean Systems Center, San Deigo, CA.

Hennessy, S.J. and King, R.H., June 1987, "Future Mining Technology Spin-Offs from ALV Program," paper presented at the IEEE/IAS, Proceedings of Mining Industry Applications Annual Meeting, and accepted for publication in the IAS Transactions.

King, R., Lever, P. J. A., Strickland, B., and Lane, J., 1991, "Ultrasonic Rangers For Underground Mine Equipment Navigation", Proceedings of The International Symposium on Mine Mechanization and Automation, Golden, CO.

King, R., Steele, J. P., "Automated LHD", Research Proposal, Golden, CO.

Lever, Paul J. A., 1991, "Learning Algorithms for a Sensor Base Intelligent Decision Support System," Dissertation, Colorado School of Mines, Golden, CO.

McTamaney, Louis S., Winter 1987, "Mobile Robots Real-Time Intelligent Control," IEEE Expert.

Oppenheim, I.J., Whittaker, W.L., "Demonstration of Robotic Mapping of Mine Spaces," final report, CMU.

Ruff, T.M., June 1991, "Ultrasonic Guidance and Remote Control of a Compact Loader/Trammer," Proceedings of The International Symposium on Mine Mechanization and Automation, Golden, CO.

St-Amant, M., Laperriere, Y., Hurteau, R., and Chevrette, G., June 1991, "A Simple Robust Vision System for Underground Vehicle Guidance," Proceedings of The International Symposium on Mine Mechanization and Automation, Golden, CO.

Steele, John, P. H., 1988, "Path planning and control with real-time response in dynamic environments," Dissertation, The University of New Mexico, Albuquerque, NM.

Vagenas, N., Sjoberg, H., Wikstrom, S., June 1991, "Application of Remote-Controlled/Automatic Load-Haul-Dump System in Zinkgruvan, Sweden," Proceedings of The International Symposium on Mine Mechanization and Automation, Golden, CO.

APPENDIX A

PERCEPTION AND CONTROL SOFTWARE

DATA ACQUISITION AND PERCEPTION SOFTWARE

```
/************************************************
*    Program to get data values from ultrasonic *
*    transducers.  This program is run on the   *
*    GESPAC system.                             *
************************************************/

#include <stdio.h>
#include <strings.h>
#include <math.h>
#include <modes.h>
#include <errno.h>
#include <module.h>
#include <sgstat.h>
#include <gesiodev.h>

main()
{
int counter,junk,k;
int path,time,date,tick,gain;
int val0,val1,val2,val3,val4,val5,val6,val7,compath;
int chan0,chan1,chan2,chan3,chan4,chan5,chan6,chan7;
int old_time, old_tick, diff,count_tick,numbt=0;
char output[50],rdbuff[90];
short day;
/*
   channel assignments
   0-rear right
   1-front right
   2-rear left
   3-front left
   4-left steering sensor
*/
     counter = 0;
     chan0 = 0;
     chan1 = 1;
     chan2 = 2;
     chan3 = 3;
     chan4 = 4;
     gain = 1;
     val0 = 0;
     val1 = 0;
     val2 = 0;
     val3 = 0;
```

```
val4 = 0;
count_tick = 0;
old_tick = 0;
/* set communications to serial port t1 */
compath = open("/t1",3);

/* connect adc board to mpu */
attach("adc12",3);

if((path = open("/adc12",S_IREAD+S_IWRITE)) == -1)
{
   printf("couldn't open adc12 device \n");
   exit(0);
}

/* get date-time data from system */
_sysdate(3,&time,&date,&day,&tick);
old_time = time;

/* disable trigger, so we can grab data at any time
   that we desire */
adc_trig_off(path);

while(counter != 1){
     /* (2147483665) 20 samples per second */
     /* (2147483678) 10 samples per second */
     tsleep(2147483678);

     /* read adc channels 0-7 */
     val0 = read_adc(path, chan0,gain);
     val1 = read_adc(path, chan1,gain);
     val2 = read_adc(path, chan2,gain);
     val3 = read_adc(path, chan3,gain);
     val4 = read_adc(path, chan4,gain);

     /* the following lines of code generate a data
     acquisition rate of XX samples per second */
     _sysdate(3,&time,&date,&day,&tick);
     diff = abs(tick - old_tick);

     if(diff >= 50)
          diff = abs(old_tick - (tick + 100));
     old_tick = tick;

     if(time == old_time)
          count_tick = count_tick + diff;
```

```
          else
          {
               count_tick = diff;
               old_time = time;
          }

          /* store adc values in a string called output */
          sprintf(output,"%u %u %u %u %u %u %ld %u\n",
            val0,val1,val2,val3,val4,val5,time,count_tick);

          writeln(compath,output,50);

          if((numbt = _gs_rdy(compath)) != -1)
{
               if(numbt > 80)
                    read(compath,rdbuff,80);

          }
     }
     close(path);
}
```

```
/*************************************************************
*  plt60.c                                                   *
*  This is a simple program to read data from the GESPAC     *
*   acquisition system.  It uses the Quinn-Curtis's async    *
*   interrupt I/O drives for communication.  This program    *
*   is run on the host computer and communicates with        *
*   the GESPAC computer to obtain the data acquisition       *
*   values.                                                  *
*************************************************************/
#include "dataq60.h"

main()
{
     time_t          bintime;
     struct          tm *curtime;
     int             ic,ip,bufans,chan,mcounter=1,err=0;

     open_com(1,19200,1,1,8,&err);

     if(err)
     {
          printf("Error on opening gespac line com1\n");
```

```
        close_com();
        exit(0);
    }
    my_initialize();
    mpl_write("getdata");
    for (ic=0; ic<10; ic++) get_data();
    while (myanswer != 'x')
    {
        if(_bios_keybrd(_KEYBRD_READY))
        {
            myanswer=_bios_keybrd(_KEYBRD_READ) & 0xff;
        }
        switch (myanswer)
        {
            case 's':
            {
                _clearscreen(0);
                store_flag = 1;
                strcpy(filetype, "r");
                make_name();
                sopen_file();
                myanswer = 0;
                break;
            }
            case 'q':
            {
                fclose(sfp);
                store_flag = -1;
                myanswer = 0;
                break;
            }
            case 'h':
            {
                my_help();
                break;
            }

            case 'x':
            {
                _settextposition(22,1);
                printf("exiting real-time mapper");
                break;
            }
        }
        get_data();
    }
```

```
     mpl_write("\3");
     close_com();
     exit(0);
}

/*************************************
* runs through initializing processes *
*************************************/
void my_initialize(void)
{
     yes = 'y';
     _clearscreen(0);
     my_help();
     while(yes == 'y')
     {
          get_date();
          printf("day: %d  month:%s\n",daynum,
                                        mymonth[monthnum]);
          printf("do you wish to change anything?\n");
          yes = getch();
     }
}

/*************************************
* get_data() routine takes data out *
* of the communications buffer.     *
*************************************/
get_data()
{
     float tick_rat;
     char temp_s[20];
     int i,j;

     res_err = mpl_response();
     if(res_err == 8)
     {
          printf("res_err = 8\n");
          exit(0);
     }

     if(res_err == 0)
     {
          sscanf(line,"%u %u %u %u %u %ld %u\n",
                    &ges_data[0],&ges_data[1],
                    &ges_data[2],&ges_data[3],
                    &ges_data[4],&mytime,&mycount);
```

```
          if (store_flag == 1)
               fprintf(sfp,"%u    %u    %u    %u    %u    %ld
                       %u\n",ges_data[0],ges_data[1],
                             ges_data[2],ges_data[3],
                             ges_data[4],mytime,mycount);

          tick_rat = mycount/100.00;
          dtime = mytime + tick_rat;
          if(oldtm == mytime)
          {
               myc = myc + 1;
          }
          else
          {
               diff=tick_rat;
               myc = 1;
               oldtm = mytime;
          }
     }
}

/*****************************
* determines gespac response *
* to command sent            *
*****************************/
int mpl_response()
{
     /*  Read a line of input from the GESPAC system */

     int err=0;
     line[0]=0;

     readln_com(line, &err);
     if(err == 6)
          printf("err = 6\n");
     else if(err)
          printf("Error reading GESPAC err = %d\n", err);
     return(err);
}

/***********************
* sends information or *
* commands to gespac   *
***********************/
mpl_write(char *str)
{
```

```
     char trans_char;
     char cb = 2;         /* CTRL B */
     int err= 0;

     if(verbose)
          printf("mdl_string = %s\n", str);

     while(trans_char = *str++)
     {
          send_com(trans_char,&err);    /* Send out String
                                           passed */
          if(err)
          {
               printf("Char not trans2 err=%d\n",err);
               close_com();
               exit(0);
          }
     }
     trans_char = '\r';
     send_com(trans_char,&err); /* End transmission with CR */
     mpl_response();
}

/*************************
* command character list *
*************************/
void my_help(void)
{
     _settextposition(3,1);
     printf("ELMAC initial data collection system\n");
     printf("options include:\n\n");
     printf("   s..............begin data collection\n");
     printf("   q..............quit data collection\n");
     printf("   r..............reset real time map\n");
     printf("   h..............display command list\n");
     printf("   x..............exit program\n");
     printf("HIT RETURN TO CONTINUE\n");
     getch();
}

/*********************************
* makes filename for data storage *
*********************************/
void make_name(void)
{
     char timesuffix[5];
```

```
      char daysuffix[2];
      strcpy(outfile,"");
      strcat(outfile,mymonth[monthnum]);
      ltoa(mytime,timesuffix,10);
      strcat(outfile,timesuffix);
      strcat(outfile,".");
      itoa((int)daynum,daysuffix,10);
      strcat(outfile,daysuffix);
      strcat(outfile,filetype);
      _settextposition(1,1);
      printf("filename is: %s",outfile);
}

/************************************
* gets date from user, this is used *
* to make the filename              *
************************************/
void get_date(void)
{
      daynum = 0;
      _clearscreen(0);
      _settextposition(3,7);
      printf("enter day code as shown below\n\n");
      printf("   1st.............01\n");
      printf("   2nd.............02\n");
      printf("   3rd.............03\n");
      printf("   10th............10\n");
      printf("   etc...\n");
      scanf("%d",&daynum);
      _clearscreen(0);
      _settextposition(3,7);
      printf("   enter date code as shown below\n\n");
      printf("   january.........1\n");
      printf("   february........2\n");
      printf("   march...........3\n");
      printf("   april...........4\n");
      printf("   may.............5\n");
      printf("   june............6\n");
      printf("   july............7\n");
      printf("   august..........8\n");
      printf("   september.......9\n");
      printf("   october.........10\n");
      printf("   november........11\n");
      printf("   december........12\n");
      scanf("%d",&monthnum);
}
```

```
/**************
* open_file() *
**************/
void sopen_file()
{
     if((sfp = fopen(outfile,"w")) == NULL)
     {
               printf("\n!!! can not open: %s !!!\n",outfile);
               exit(0);
     }
}
```

```
/*****************************************************
* July 19, 1991  Colorado School of Mines            *
* by James D. Lane                                   *
* plt50.c                                            *
* program adjusts data from ultrasonic transducers   *
* on ELMAC to develop a map of test course           *
* also plots computer determined vehicle center      *
* line of travel as compared to actual center        *
* line of travel.   This program runs on host        *
* computer using previously stored data files        *
* generated during test runs.  A detailed            *
*  description of this program is found in chapter 4*
*****************************************************/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "kmaps50.h"
#include "map50.h"
char input[15]="";
char *re="r";
FILE *fpr,*open_file();

main()
{
     information(); /*message to user*/
     while(getch()  !=  'x')  /*continue  program  until  x
                                          entered*/
     {
          my_initial(); /*initialize variables*/
          while(!(feof(fpr)))  /*continue  loop  until  eof
                                       encountered*/
          {
```

```
                fscanf(fpr,"%f %f %f %f %f %ld %d\n",
                     &ges_dat[0],&ges_dat[1],&ges_dat[2],
                     &ges_dat[3],&new_theta,&mytime,&mycount);

                fix_theta();  /*convert  angle  readings  to
                                                radians*/
                dig_to_feet(); /*convert ranger readings to
                                                feet*/
                running_ave();   /*compute   ranger   weighted
                                          running avg*/
                actual_path(); /*determine vehicle path of
                                                travel*/
                desired_path(); /*determine optimum path of
                                                travel*/
                out_of_bounds();   /*determine   ranger   void
                                         readings*/
                make_coord();  /*make  map  coordinates  with
                                                data*/

                local_map(xoff/GSCALE,(yoff+NONNEG)/GSCALE,
                     flxo/GSCALE,(flyo+NONNEG)/GSCALE,
                     frxo/GSCALE,(fryo+NONNEG)/GSCALE,
                     desiredx/GSCALE,(desiredy+NONNEG)/GSCALE);
                /*call mapping routine*/

                old_theta = new_theta; /*update old_theta*/
           }
           fclose(fpr); /*close data file*/
      }
}

/************************
* file opening routines *
************************/
void open_files()
{
     fpr = open_file(fpr,input,re);
}

FILE *open_file(f, file, what)
FILE *f;
char *file, *what;
{
    if((f=fopen(file,what))==NULL)
    {
          printf("\n\nCouln't open %s to
```

```
%s.\nEXIT\n",file,what);
          exit(-1);
     }
     else
          return(f);
}

/*****************************************
* function creates and updates running ave *
*****************************************/
void running_ave()
{
     int sinc;
     float rundiv=0.0;
     sumfl = 0.0;
     sumfr = 0.0;

     /* set x(n+1) to x(n), updates running average */
     for(sinc=0; sinc<(RUN_AVG-1);sinc++) /*this loop replaces
                                                  each aray*/
     {          /*value with the next newest value*/
          runfr[sinc+1] = runfr[sinc];
          runfl[sinc+1] = runfl[sinc];
     }

     /* assign new sensor values */
     runfr[ZERO] = front_right; /*add new ranger readings onto
                                                  beginning*/
     runfl[ZERO] = front_left;  /*of array*/
     rundiv  =  (RUN_AVG*RUN_AVG  +  RUN_AVG)/2.0;  /*compute
                                          running avg*/
                                          /*divisor*/

     /* compute weighted average for each sensor reading */
     for (sinc=0; sinc < RUN_AVG; sinc++)
     {
          sumfl  += runfl[sinc]*(RUN_AVG - sinc); /*multiply
                                          newest reading*/
          sumfr += runfr[sinc]*(RUN_AVG - sinc); /*by RUN_AVG
                                          window size*/
     }   /*and each succeeding value by RUN_AVG-sinc*/

     /* set sensor readings to weighted running average */
     front_left  =  sumfl/rundiv; /*compute  average  value  of
                                                  window*/
     front_right = sumfr/rundiv;   }
```

```
/*******************************************
*  converts steering angle to radians where *
*  0 < theta_rad < TWO_PI                   *
*******************************************/
void fix_theta()
{    /* eliminate out of bounds readings */
     if ((new_theta < THETA_MIN) && (old_theta > THETA_MIN))
          new_theta = old_theta;
     else if((new_theta < THETA_MIN)&&(old_theta < THETA_MIN))
          new_theta = THETA_MIDDLE;

     /* set up theta so it is negative for right-hand
        turns and positive for left-hand turns and then
        convert it to radians */
     theta = (THETA_MIDDLE - new_theta)/THETA_SCALE;
     theta_rad += DEG_TO_RAD*theta;

     /* set up theta_rad to fall between the values of
        zero and two pi */
     if (theta_rad <= 0)
          theta_rad += TWO_PI;

     else if (theta_rad > TWO_PI)
          theta_rad -= TWO_PI;
}

/*****************************************************
* initialize variables and get input output filenames *
*****************************************************/
void my_initial()
{
     _settextposition(20,1);
     printf("\nEnter name of input file: ");
     scanf("%s",input);
     open_files();

     theta = 0.0; /*digital angle value*/
     old_theta = 0.0; /*digital angle value*/
     theta_rad = 0.0; /*angle value in radians*/
     xoff = 0.0; /*forward progress variable*/
     yoff = 0.0; /*sideways progress variable*/
     sumfr = 0.0; /*sum of front right ranger readings*/
     sumfl = 0.0; /*sum of front left ranger readings*/
     setup_screen(1,1);
     _setviewport(1,1,598,249);
     _setcolor(12);}
```

```
/*******************************************************
* convert digital ultrasound readings to units of feet *
*******************************************************/
void dig_to_feet()
{
     /* take digital values from ultrasound sensors and
        convert them to units of feet.  Also set up values
        so they can be easily plotted with disp43.c program.
        values must be adjusted accordingly depending on which
        quadrant the vehicle is operating in.  This quadrant
        is setup with respect to the position angle of the
        vehicle overall in the test site.  It is referenced
        back to the starting point of the test site. */

     if (xoff_inc < 0)
     {
          /* 2nd and 3rd quadrants */
          front_left    =    (DATA_MAX    -    ges_dat[3]    +
                                   HWFRONT)/USCALE;
          front_right    =    (ges_dat[1]    -    DATA_MAX    -
                                   HWFRONT)/USCALE;
          sign = 1.0;
/*
          rear_right    =    (ges_dat[0]    -    DATA_MAX    -
                                   HWREAR)/USCALE;
          rear_left = (DATA_MAX - ges_dat[2] +
                                   HWREAR)/USCALE;
*/
     }

     else
     {    /* 1st and 4th quadrants */
          front_left    =    (ges_dat[3]    -    DATA_MAX    -
                                   HWFRONT)/USCALE;
          front_right    =    (DATA_MAX    -    ges_dat[1]    +
                                   HWFRONT)/USCALE;
          sign = -1.0;
/*
          rear_right    =    (DATA_MAX    -    ges_dat[0]    +
                                   HWREAR)/USCALE;
          rear_left = (ges_dat[2] - DATA_MAX -
                                   HWREAR)/USCALE;
*/
     }
}
```

```
/*****************************************************
* compute actual path travelled via string pot data *
*****************************************************/
void actual_path()
{
     yoff += VECTOR_MAG*sin(theta_rad)/YOFFSCALE;
     xoff_inc = VECTOR_MAG*cos(theta_rad);
     xoff += xoff_inc/XOFFSCALE;
}

/*********************************
* compute desired path of travel *
*********************************/
void desired_path()
{
     correction = abs(front_left) - abs(front_right);

     if (abs(front_left) < MIN_LEN)
     {
          _settextposition(21,1);
          printf("warning right:  vehicle too close to wall
                                   please steer right\n");
          _settextposition(23,1);
          printf("\n");
          desiredy = yoff + cos(theta_rad+theta*DEG_TO_RAD/2)*
                                (MIN_LEN - abs(front_left));
     }

     else if (abs(front_right) < MIN_LEN)
     {
          _settextposition(23,1);
          printf("warning left:  vehicle too close to wall ");
          printf("please steer left\n");
          _settextposition(21,1);
          printf("\n");
          desiredy = yoff + cos(theta_rad+theta*DEG_TO_RAD/2)*
                                (MIN_LEN - abs(front_left));
     }

     else if (abs(correction) > OFF_CENTER)
     {
          desiredy = yoff - cos(theta_rad+theta*DEG_TO_RAD/2)*
                                                  correction;

          if (correction > 0)
          {
```

```
                _settextposition(23,1);
                printf("warning left:  vehicle is off center
                                                    line ");
                printf("please steer left\n");
                _settextposition(21,1);
                printf("\n");       }
           else
           {
                _settextposition(21,1);
                printf("warning right:  vehicle is off center
                                                    line ");
                printf("please steer right\n");
                _settextposition(23,1);
                printf("\n");
           }
      }

      else
           desiredy = yoff;

      desiredx = xoff;
}

/***********************************
* determine out-of-bounds readings *
***********************************/
void out_of_bounds()
{
      /* locate out of bound readings and mark them */
      if (abs(front_right) > ((MAX_CUT+HWFRONT)/USCALE))
           front_right *= MULT_CUT;

      if (abs(front_left) > ((MAX_CUT+HWFRONT)/USCALE))
           front_left *= MULT_CUT;
}

/*****************************************
* translate ultrasound readings to global *
* x and y coordinate system               *
*****************************************/
void make_coord()
{
      double my_sin,my_cos,my_cos_sin;
      /* transpose ultrasound readings to global test site
         coordinates with respect to the starting point of
         the vehicle. */
```

```
     my_sin = sin(theta_rad - theta*DEG_TO_RAD/2);
     my_cos = cos(theta_rad - theta*DEG_TO_RAD/2);
     my_cos_sin = abs(sin(theta_rad + theta*DEG_TO_RAD/2));

     flxo    =    xoff    -    my_cos_sin*front_left    -
                          my_cos*FOFFX*sign/XTRUCK_SCALE;
     flyo    =    yoff    +    my_cos*front_left    -
                          my_sin*FOFFX*sign/YTRUCK_SCALE;
     frxo    =    xoff    +    my_cos_sin*front_right    -
                          my_cos*FOFFX*sign/XTRUCK_SCALE;
     fryo    =    yoff    +    my_cos*front_right    -
                          my_sin*FOFFX*sign/YTRUCK_SCALE;
/*
     blxo=xoff-sin(theta_rad-theta*DEG_TO_RAD/2)*rear_left;
     blyo=yoff+cos(theta_rad-theta*DEG_TO_RAD/2)*rear_left;
     brxo=xoff-sin(theta_rad-theta*DEG_TO_RAD/2)*rear_right;
     bryo=yoff+cos(theta_rad-theta*DEG_TO_RAD/2)*rear_right;
*/
}


/*************************
* print out user message *
*************************/
void information()
{
     _clearscreen(0);

printf("*****************************************************
                                        ********\n");

printf("*  This  program  determines  the  ELMAC  centerline
                                      travelled   *\n");

printf("*  during  a  test  run,  and  adjusts  the  Ultras
                                      accordingly.  *\n");

printf("*****************************************************
                                        ********\n");

printf("enter x to cancel or any other key to continue");
}
```

```
/*************************************************
*  ASYNPEC.C                                     *
*  asynchronous com routine for Microsoft C 5.1  *
*  Copyright 1989, Quinn-curtis                  *
*  This program does the actual communicating    *
*  between the GESPAC and the host computer      *
*************************************************/
/* asynchronous com routine for Microsoft C 5.1 */
/* Copyright 1989, Quinn-curtis */

#include <bios.h>
#include <dos.h>
#include <string.h>
#include "asynpec.h"

#define timeout  10000
#define max_buffer 10000
#define near_full 9900
#define near_empty 100

int com_flag = 0;

int overrun_flag;
char com_buffer[max_buffer];
int com_port;
int intlev;
int bf_hndshk;

int thr,rbr,ier,lcr,mcr,lsr,msr;

void far *oldfunc;


void delay(int d)
/* delay d in milliseconds at 10MHz */
{
int i;
int j;
int k;
j = 0;
for (k=0;k<d;k++)
     for (i=1;i<200;i++)
     j += 1;
}
void interrupt far com_isr()
{
```

```
if(com_flag == 1){
     /* get character and store in circular buffer */
     com_buffer[buffer_in] = inp(rbr);

     /* Increment buffer_in pointer */
     buffer_in += 1;

     /* Wrap around to zero if at end */
     if(buffer_in == max_buffer) buffer_in = 0;

     /*Increment no of characters in buffer */
     buffer_length += 1;
     if(buffer_length > max_buffer){
          buffer_length = max_buffer;
          overrun_flag = 1;
          }
     /*  Disable  RTS  if  buffer_length  exceeds  near_full
constant */
     if(buffer_length>near_full){
          outp(mcr,9);
          bf_hndshk = 1;
          }
     }
     outp(0x20,0x20);
}

void reset_buffer()
/* Reset the com buffer */
{
bf_hndshk=0;
buffer_in=0;
buffer_out=0;
buffer_length=0;
overrun_flag=0;
}

void open_com(int Cport,
     int baud,
     int parity,
     int stopbits,
     int numbits,
     int *error_code)

/*   This routine does not seem to set things up entirely
    correctly for some reason.  As a result I have hard coded
    the setup stuff I need for the MDL-16   REC-1989   */
```

```
{
int comdata;
int ptemp;
     *error_code = 0;
     com_port = Cport;
     comdata=0;
     if((numbits == 7)||(numbits == 8))
          comdata=comdata|(numbits-5);
     else *error_code = 5;

     if((stopbits == 2)||(stopbits == 1))
          comdata=comdata|(stopbits-1)<<2;
     else *error_code = 4;

     if((parity == 1)||(parity == 3)||(parity == 0))
          comdata=comdata|(parity<<3);
     else *error_code = 3;

     switch(baud){
          case 110:comdata = comdata|0x00;
          break;
          case 150:comdata = comdata|0x20;
          break;
          case 300:comdata = comdata|0x40;
          break;
          case 600:comdata = comdata|0x60;
          break;
          case 1200:comdata = comdata|0x80;
          break;
          case 2400:comdata = comdata|0xA0;
          break;
          case 4800:comdata = comdata|0xC0;
          break;
          case 9600:comdata = comdata|0xe0;
          break;
          default:*error_code = 2;
          break;
     }

     if((Cport<0)||(Cport>1))
          *error_code = 1;
/*  For some reason the above code does not seem to set the
    port up correctly.  Hence the following is a hard coded
    setup of comdata for what I need for the MDL-16.
    REC-1989                          */
     comdata = (_COM_CHR8 | _COM_STOP1);
     comdata = comdata | _COM_NOPARITY;
```

```
comdata = comdata | _COM_9600;

if(*error_code == 0)
     _bios_serialcom(0,Cport,comdata);

if(Cport == 0){
     thr=0x3f8;
     rbr=0x3f8;
     ier=0x3f9;
     lcr=0x3fb;
     mcr=0x3fc;
     lsr=0x3fd;
     msr=0x3fe;
     }else{
     thr=0x2f8;
     rbr=0x2f8;
     ier=0x2f9;
     lcr=0x2fb;
     mcr=0x2fc;
     lsr=0x2fd;
     msr=0x2fe;
     }
ptemp = inp(0x3fb);
outp(0x3fb, 128);
outp(0x3f8,6);
outp(0x3f9,0);
outp(0x3fb,ptemp);

intlev =0xC - Cport;
oldfunc = _dos_getvect(intlev);
_dos_setvect(intlev,com_isr);
_disable();
ptemp = inp(lcr)&0x7f;
outp(lcr,ptemp);
ptemp = inp(lsr);
ptemp = inp(rbr);
if(Cport == 0){
     ptemp = inp(0x21)&0xef;
     outp(0x21,ptemp);
     }
else{
     ptemp = inp(0x21)&0xf7;
     outp(0x21,ptemp);
     }
outp(ier,1);
ptemp=inp(mcr)|0xb;
```

```
      outp(mcr,ptemp);
      _enable();
      *error_code=0;
      com_flag=1;
      reset_buffer();
}

void close_com()
/* Disable com port interrupt */
{int ptemp;
      if(com_flag==1){
            _disable();
            ptemp=inp(0x21)|0x18;
            outp(0x21,ptemp);
            ptemp=inp(lcr)|0x7f;
            outp(lcr,ptemp);
            outp(ier,0);
            outp(mcr,0);
            _dos_setvect(intlev,oldfunc);
            _enable();
            com_flag=0;
            }
}

void check_com(char *c, int *error_code)
{
      if(com_flag == 0)
            *error_code = 10;
      else{
            if(buffer_length == 0)
                  *error_code = 6;
            else{
                  if(overrun_flag == 1)
                        *error_code = 7;
                  else
                        *error_code = 0;
                  *c = (com_buffer[buffer_out]);
                  buffer_out += 1;
                  if(buffer_out == max_buffer)
                        buffer_out = 0;
                  buffer_length -= 1;

                  if(bf_hndshk && (buffer_length < near_empty)){
                        outp(mcr,0xb);
                        bf_hndshk = 0;
                  }
```

```
          }
     }
}

void send_com(char c, int *error_code)
{
int handshake;
int counter;

     if(com_flag == 0)
          *error_code = 10;
     else{
          counter = 0;
          handshake = 0x0;
          do{
               counter += 1;
               delay(1);
          }
          while
               ((((inp(msr)&handshake)!= handshake) ||
                 ((inp(lsr)&0x20) != 0x20)) &&
                  (counter<timeout));
          if(counter==timeout)
               *error_code = 8;
          else{
               _disable();
               outp(thr,c);
               _enable();
               *error_code=0;
          }
     }
}

void writeln_com(char *str, int *error_code)
{int length;
int i;

     length=strlen(str);

     for(i=0;i<length;i++){
          send_com(str[i],error_code);
     }
     send_com(13,error_code);
}
void readln_com(char *str, int *error_code)
{int i=0;
```

```
char c;
int counter = 0;
     do {
          check_com(&c,error_code);
          if(*error_code == 0)
               {
               str[i]=c;
               i += 1;
               }
               else{
               delay(1);
               counter += 1;
          }
     }while((i<50000)&&(c != 13) && (counter < timeout));
     if(counter == timeout) *error_code = 8;
     str[i]=0;
}

int buffer_empty(void)
{
     if(buffer_length == 0)
          return(0);
     else
          return(1);
}
```

```
/* Prototype file for asynpec.c */
void delay(int d);
void open_com(int Cport, int baud, int parity, int stopbits,
int numbits,
          int *err_code);
void close_com();
void reset_buffer();
void send_com(char c, int *err_code);
void writeln_com(char *str, int *err_code);
void readln_com(char *str, int *err_code);
int buffer_empty(void);
int buffer_in;
int buffer_out;
int buffer_length;
```

STEERING CONTROL SOFTWARE

```
/****************************************
*  program used to setup communications *
*  with parallel ports A and B on the   *
*  GESPAC computer GESMFI-1 board.      *
*  This program runs on GESPAC          *
*  By: Tom Yost, GESPAC, 1-800-772-4451 *
*  Appended by: James D. Lane           *
*  February 5, 1992, CSM                *
****************************************/
#include <stdio.h>
#define CPU16_BASE 0x800700   /* MFI-1 base address for MPU4B
                                                    & SBS6 */
#define  CPU32_BASE  0xfff00700  /*  MFI-1  base  address  for
                                                    MPU-20/30 */
#define TRUE 1

typedef struct
{
     unsigned short  pra_ddra,
                     cra,
                     prb_ddrb,
                     crb;
} mfipia;

mfipia *ptr_base = CPU32_BASE;

main()
{
     unsigned int data;

     /*initialize the PIA to output data */
     ptr_base -> cra = 00;/*select data direction register*/
     ptr_base -> pra_ddra = 0xff;  /*set all bits to outputs*/
     ptr_base -> cra = 04; /*select data register*/
     ptr_base -> crb = 00;/*select data direction register*/
     ptr_base -> prb_ddrb = 0xff;  /*set all bits to outputs*/
     ptr_base -> crb = 04; /*select data register*/

     /* get a value and output it*/
     while(TRUE)
     {   printf("\n\nEnter an 8-bit Hex value to output to
                                          Port A\n");
```

```
          scanf("%x", &data);
          printf("Outputting %02x on port A\n", data);
          ptr_base -> pra_ddra = (short)data; /*output data to
                                                port A*/
/*        printf("\n\nEnter an 8-bit Hex value to output to
                                                Port B\n");
          scanf("%x", &data);
          printf("Outputting %02x on port B\n", data);
          ptr_base -> prb_ddrb = (short)data;
*/
     }
}         while(getch() != 'x') /*continue program until x
entered*/
```

```
/************************************************
*    February 26, 1992                          *
*    Program to get data values from ultras,    *
*    and play follow the leader with steering.  *
*    Program runs on GESPAC                     *
************************************************/

#include <stdio.h>
#include <strings.h>
#include <math.h>
#include <modes.h>
#include <errno.h>
#include <module.h>
#include <sgstat.h>
#include <gesiodev.h>

#define  CPU32_BASE  0xfff00700  /*  MFI-1  base  address  for
MPU-20/30 */
#define TRUE 1
#define DATA_MAX 3550
#define DIG_TO_FEET 350

typedef struct
{
     unsigned short  pra_ddra,
                         cra;
} mfipia;

mfipia *ptr_base = CPU32_BASE;
```

```
main()
{
     int junk,k;
     int path,time,date,tick,gain,val;
     int val0,val1,val2,val3;
     int chan0,chan1,chan2,chan3;
     int old_time, old_tick, diff,count_tick;
     short day;
     unsigned int data;

     chan0 = 0; /*rear right*/
     chan1 = 1; /*front right*/
     chan2 = 2; /*rear left*/
     chan3 = 3; /*front left*/
     gain = 1;
     val0 = 0;
     val1 = 0;
     val2 = 0;
     val3 = 0;
     count_tick = 0;
     old_tick = 0;

     /* connect adc board to mpu */
     attach("adc12",3);

     if((path = open("/adc12",S_IREAD+S_IWRITE)) == -1){
        printf("couldn't open adc12 device \n");
        exit(0);
     }

     /* get date-time data from system */
     _sysdate(3,&time,&date,&day,&tick);
     old_time = time;

     /* disable trigger, so we can grab data at any time
        that we desire */
     adc_trig_off(path);

     /*initialize the PIA to output data */
     ptr_base -> cra = 0x00;/*select data direction register*/
     ptr_base -> pra_ddra = 0xff; /*set all bits to outputs*/
     ptr_base -> cra = 0x04; /*select data register*/

     while(TRUE)
     {
          /* (2147483665) 20 samples per second */
```

```
  /* (2147483678) 10 samples per second */
  tsleep(2147483678);
  /* read adc channels 0-7 */
  val0 = read_adc(path, chan0,gain);
  val1 = read_adc(path, chan1,gain);
  val2 = read_adc(path, chan2,gain);
  val3 = read_adc(path, chan3,gain);
  /* the following lines of code generate a data
     acquisition rate of XX samples per second */
  _sysdate(3,&time,&date,&day,&tick);
  diff = abs(tick - old_tick);

if(diff >= 50)
  diff = abs(old_tick - (tick + 100));

old_tick = tick;
if(time == old_time)
  count_tick = count_tick + diff;

  else
  {
       count_tick = diff;
       old_time = time;
  }

  /* get a value and output it*/
  data = (DATA_MAX - val1)/DIG_TO_FEET;

  if (data < 3)
  {
       printf("too close! %i\n",data);
       data = 0x0f;
  }

  else if(data > 4)
  {
       printf("too far! %i\n",data);
       data = 0xf0;
  }

  else
  {
       printf("just right! %i\n",data);
       data = 0x00;
  }
```

```
          ptr_base -> pra_ddra = (short)data; /*output data to
                                                port A*/
     }
     close(path);
}


/************************************************
*    1992 newest steering control program       *
*    Program to get data values from ultras,    *
*    and guide vehicle along center line.       *
*                                               *
************************************************/
#include <stdio.h>
#include <strings.h>
#include <math.h>
#include <modes.h>
#include <errno.h>
#include <module.h>
#include <sgstat.h>
#include <gesiodev.h>
#define  CPU32_BASE  0xfff00700  /*  MFI-1  base  address  for
                                    MPU-20/30 */
#define TRUE 1
#define DATA_MAX 3670
#define DIG_TO_FEET 300
#define THETA_MIDDLE 1400
#define RUN_AVG 10
#define ZERO 0

typedef struct
{
     unsigned short  pra_ddra,
                         cra;
} mfipia;

mfipia *ptr_base = CPU32_BASE;
int  sumfr,sumfl,runfr[50],runfl[50];
void running_ave();
int front_right,front_right_last=0;
int front_left,front_left_last=0;

main()
{
     int junk,k;
```

```
int path,time,date,tick,gain,val,compath;
int val0,val1,val2,val3,val4,direction,numbt=0;
int chan0,chan1,chan2,chan3,chan4;
int old_time, old_tick, diff,count_tick;
short day;
unsigned int data;
char output[50],rdbuff[90],message[50];
chan0 = 0; /*rear right*/
chan1 = 1; /*front right*/
chan2 = 2; /*rear left*/
chan3 = 3; /*front left*/
chan4 = 4; /*right steering sensor*/
gain = 1;
val0 = 0;
val1 = 0;
val2 = 0;
val3 = 0;
val4 = 0;
count_tick = 0;
old_tick = 0;
direction = 0;

/* set communications to serial port t1 */
compath = open("/t1",3);

/* connect adc board to mpu */
attach("adc12",3);

if((path = open("/adc12",S_IREAD+S_IWRITE)) == -1){
   printf("couldn't open adc12 device \n");
   exit(0);
}

/* get date-time data from system */
_sysdate(3,&time,&date,&day,&tick);
old_time = time;

/* disable trigger, so we can grab data at any time
   that we desire */
adc_trig_off(path);

/*initialize the PIA to output data */
ptr_base -> cra = 0x00;/*select data direction register*/
ptr_base -> pra_ddra = 0xff;/*set bits to outputs*/
ptr_base -> cra = 0x04;  /*select data register*/
```

```
while(TRUE)
{
     /* (2147483665) 20 samples per second */
     /* (2147483678) 10 samples per second */
     tsleep(2147483678);

     /* read adc channels 0-7 */
     val1 = read_adc(path, chan1,gain);
     val3 = read_adc(path, chan3,gain);
     val4 = read_adc(path, chan4,gain);

     /* the following lines of code generate a data
        acquisition rate of XX samples per second */
     _sysdate(3,&time,&date,&day,&tick);
     diff = abs(tick - old_tick);

   if(diff >= 50)
     diff = abs(old_tick - (tick + 100));

   old_tick = tick;

   if(time == old_time)
     count_tick = count_tick + diff;

     else
     {
          count_tick = diff;
          old_time = time;
     }

     /* get a value and output it*/
     front_left = (DATA_MAX - val3)/DIG_TO_FEET;
     front_right = (DATA_MAX - val1)/DIG_TO_FEET;
     running_ave();

     if  ((front_left  <  10)  ||  (front_right  <  9)  ||
     (front_left > 10))
     {
          if  (val4  <  1400)  /*vehicle  turned  toward
                                          right*/
          {
               if (front_right < front_left)
               {
               /*readjust steering to left*/
               data = 0x0f;
               direction = 200;
```

```
                    }
                    else if (front_right < front_right_last)
                    {
                         /*maintain present steering angle*/
                         data = 0x00;
                         direction = 50;
                    }
                    else
                    {
                         /*readjust steering to right*/
                         data = 0xf0;
                         direction = 100;
                    }
               }

               if (val4 > 1400) /*vehicle turned toward left*/
               {
                    if (front_right > front_left)
                    {
                    /*readjust steering to right*/
                    data = 0xf0;
                    direction = -200;
                    }
                    else if (front_left < front_left_last)
                    {
                         /*maintain present steering angle*/
                         data = 0x00;
                         direction = -50;
                    }
                    else
                    {
                         /*readjust steering to right*/
                         data = 0x0f;
                         direction = -100;
                    }
               }
          }
          else
          {
               /*maintain present steering angle*/
               data = 0x00;
               direction = 0;
          }
          front_right_last = front_right;
          front_left_last = front_left;
          ptr_base -> pra_ddra = (short)data; /*output data to
```

```
                                                        port A*/
          /* store adc values in a string called output */
          sprintf(output,"%u %u %u %ld %u %d\n",
                    front_right,front_left,val4,
                    time,count_tick,direction);

     writeln(compath,output,80);
          if((numbt = _gs_rdy(compath)) != -1)
          {
             if(numbt > 80)
             {
           read(compath,rdbuff,80);
             }
     }
    }
    close(path);
}
```

APPENDIX B

TEST RUN FIGURES

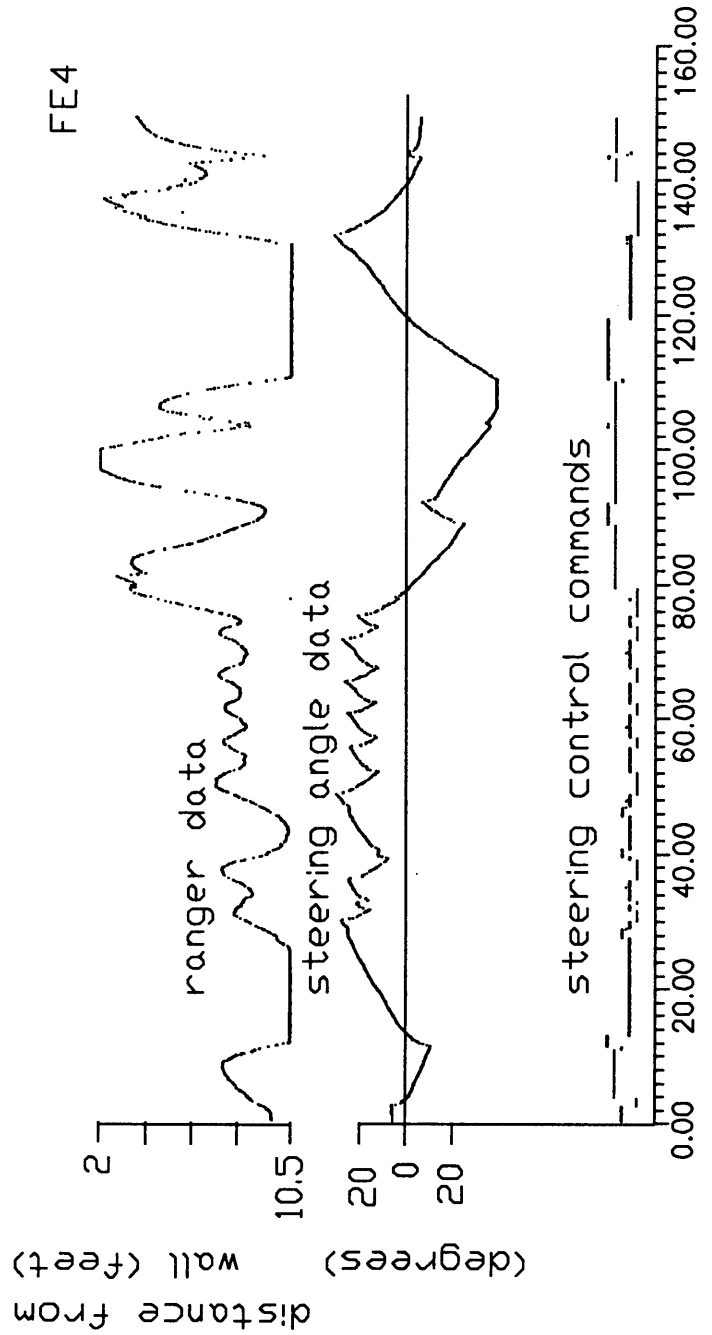

Figure B-21. Full length test (FE4).

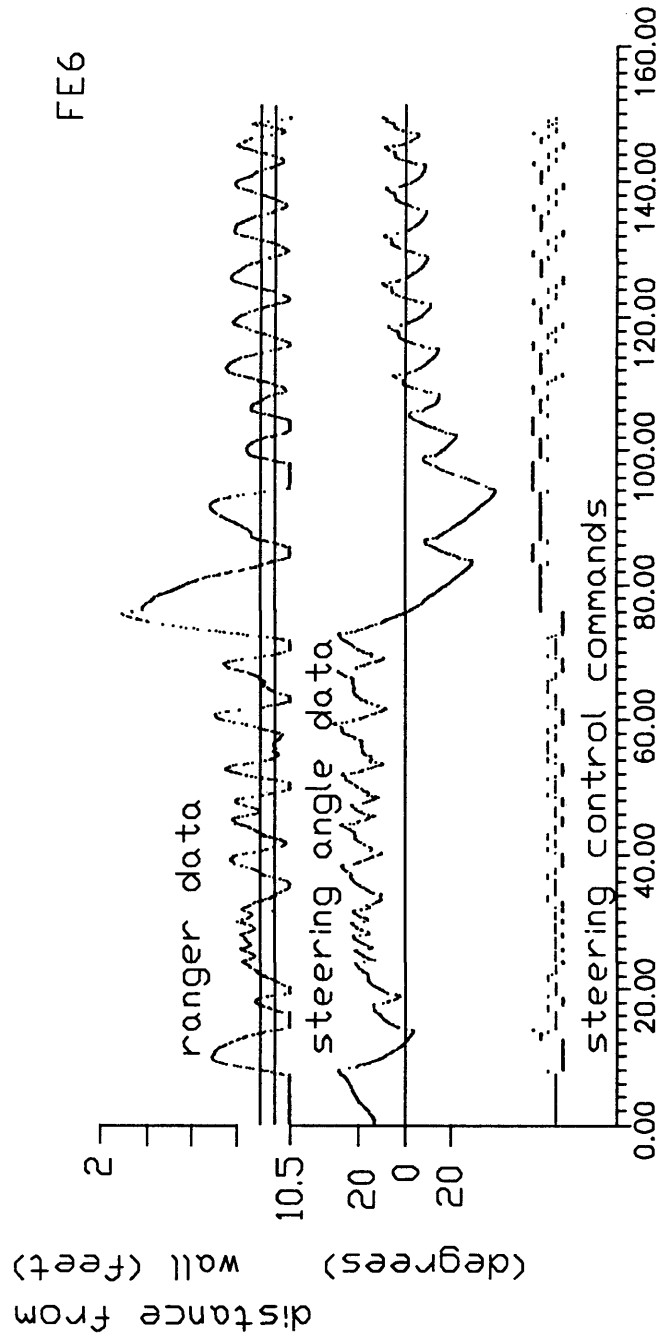

Figure B-22. Full length test (FE6).

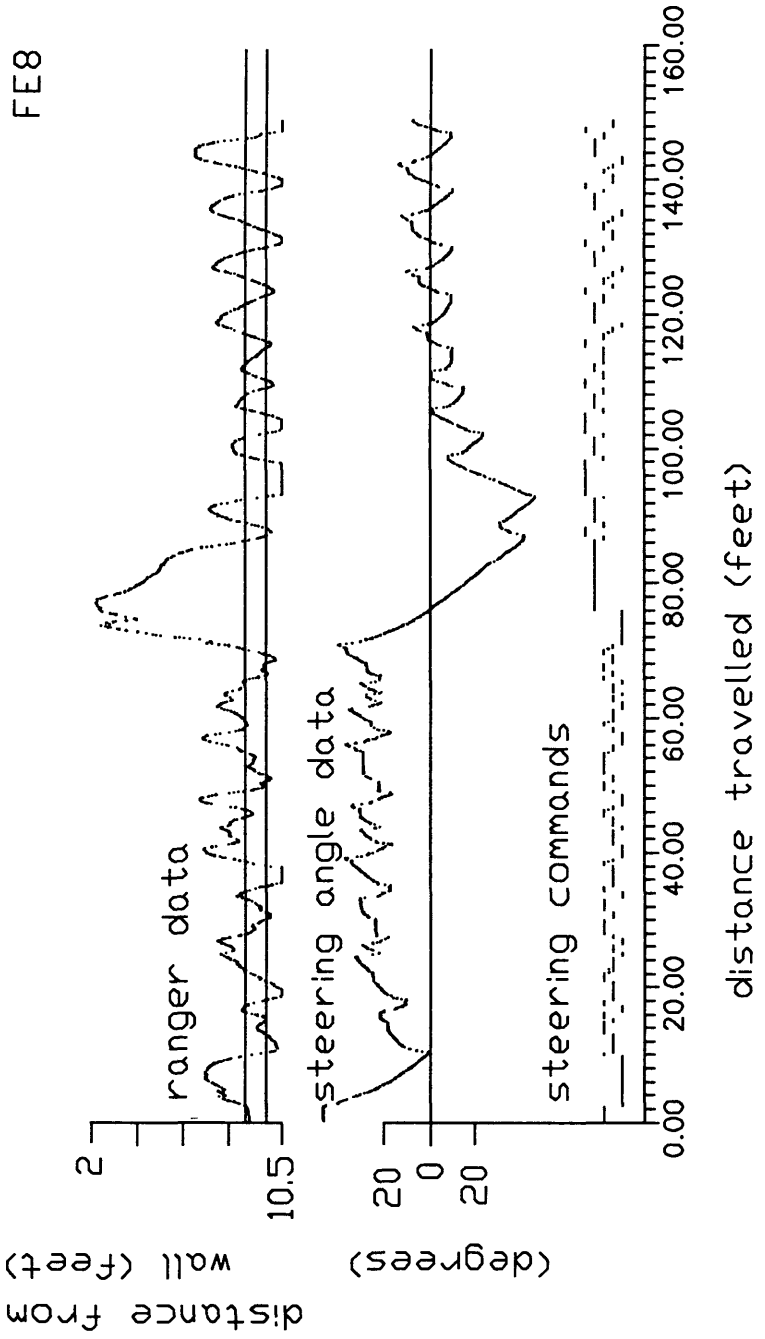

Figure B-23. Full length test (FE8).

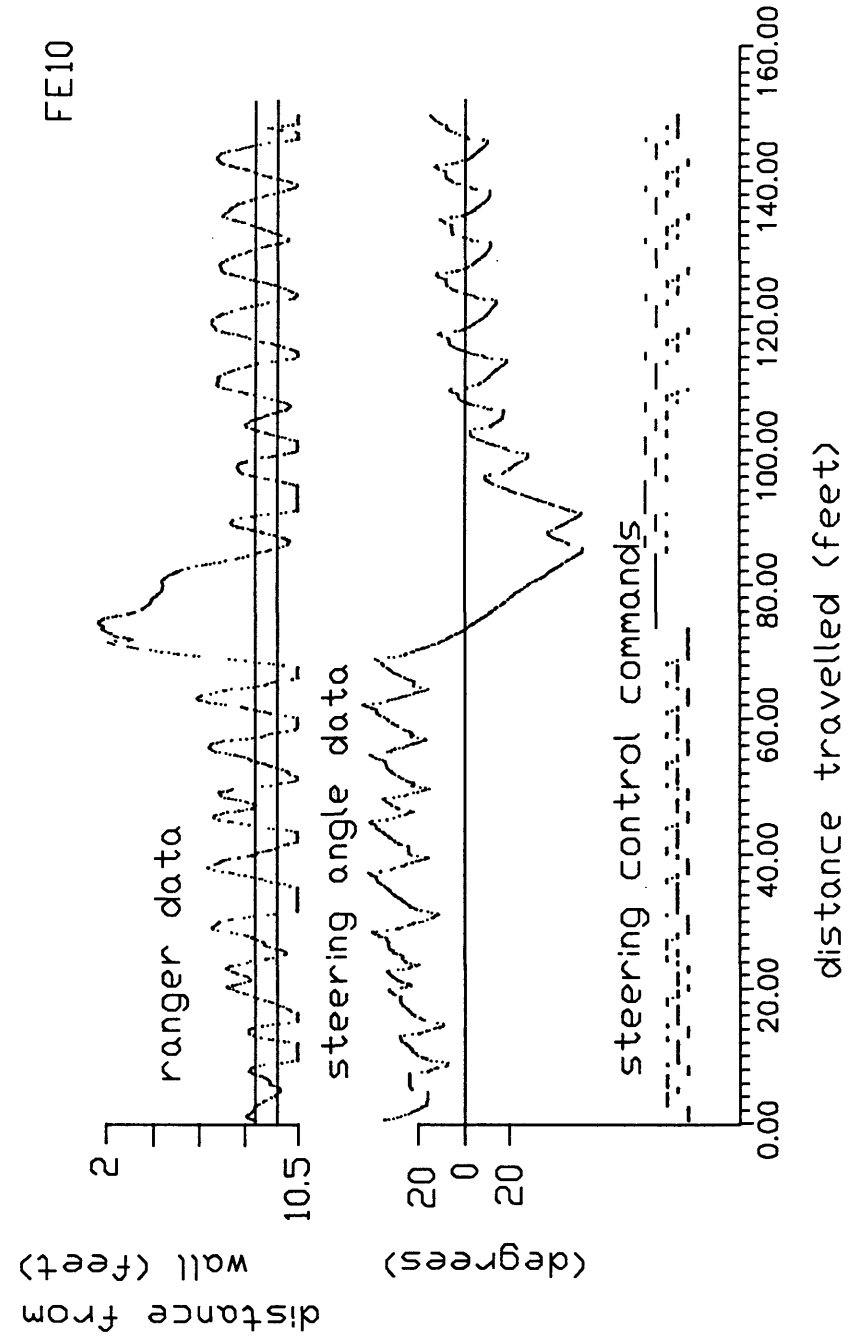

Figure B-24. Full length test (FE10).

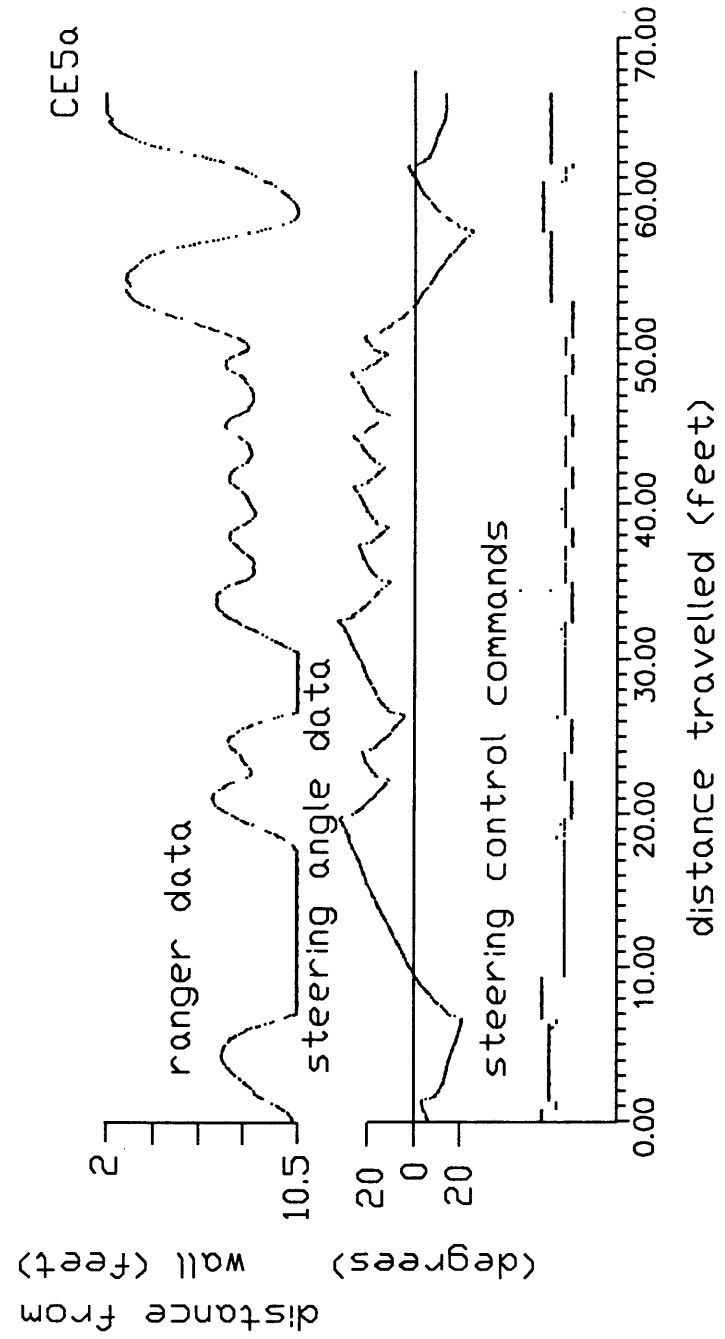

Figure B-25. Curve section test (CE5a).

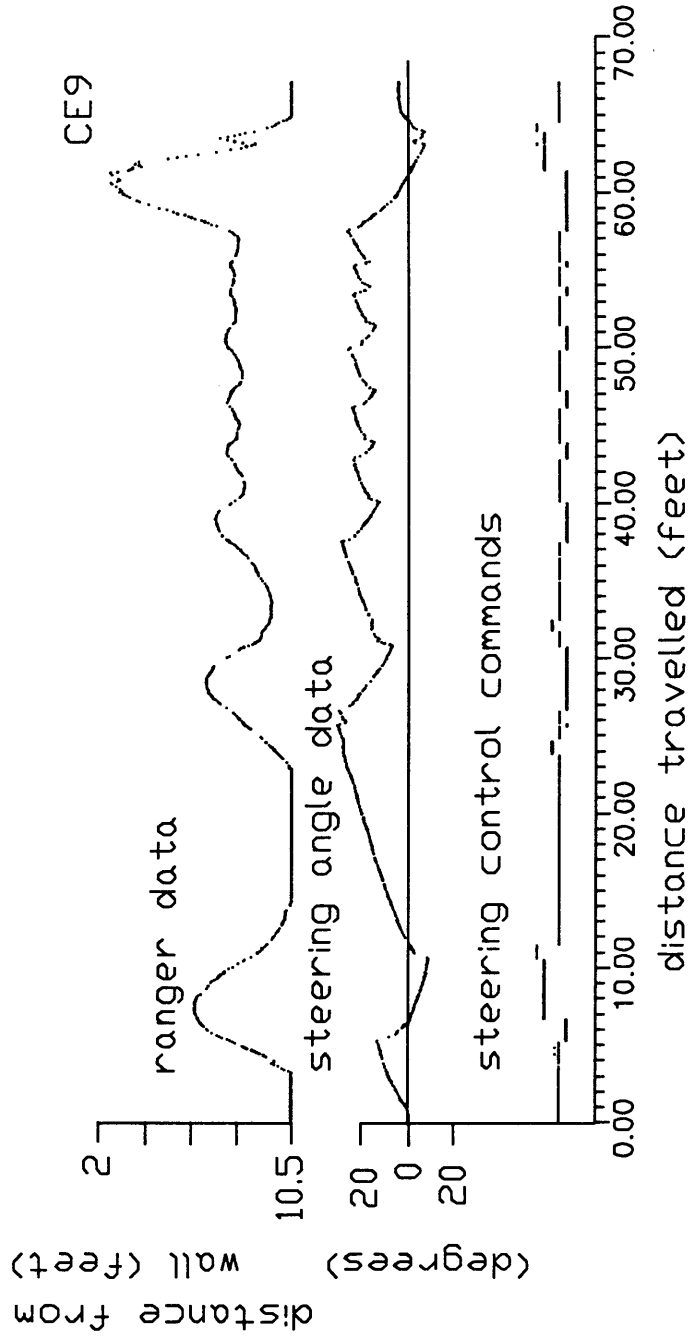

Figure B-26. Curve section test (CE9).

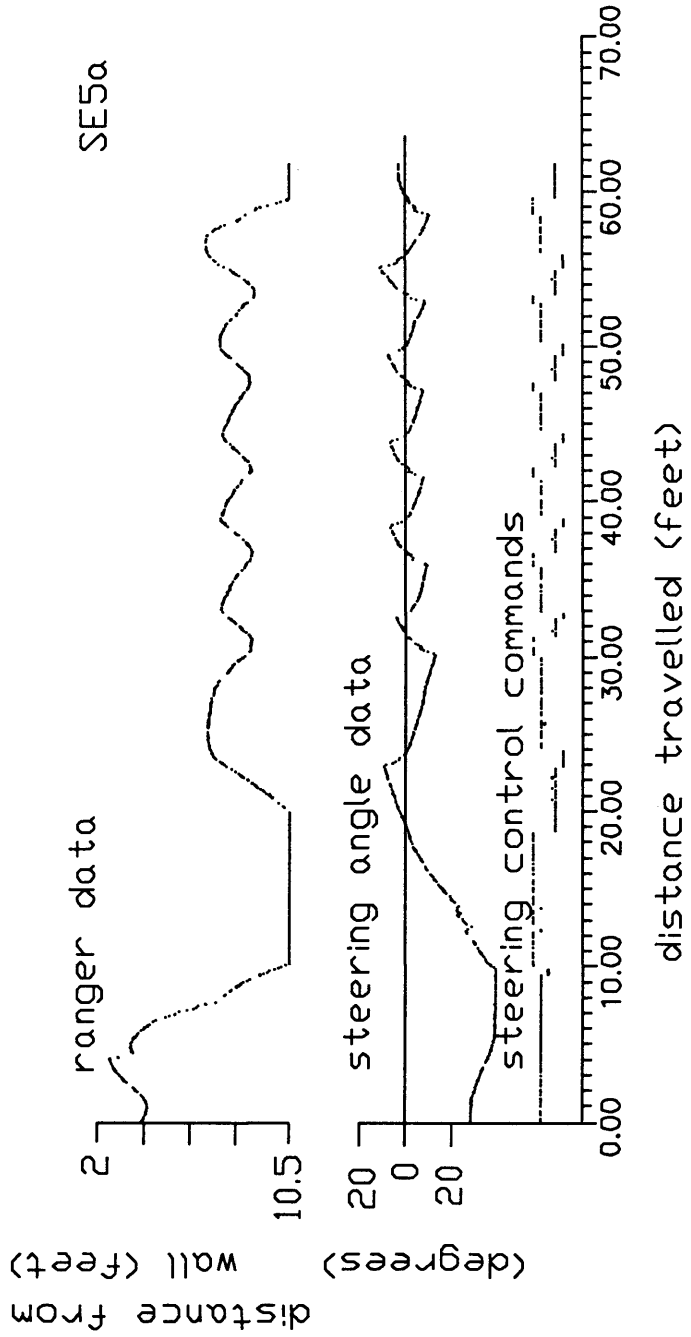

Figure B-27. Straight section test (SE5a).