Guidance Control of an Automated Vehicle Using CCD Camera Vision and Parallel Digital Signal Processing (DSP) Controller

Fabio Perelli

A Thesis

in

The Department

of

Mechanical Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Sciences at
Concordia University
Montréal, Québec, Canada

December 1995

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-612-10885-6

Canada

Guidance Control of an Automated Vehicle Using CCD Camera Vision and Parallel Digital Signal Processing (DSP) Controller

Fabio Perelli

ABSTRACT

This thesis presents the design and implementation of a high performance vehicle controller and CCD camera vision to provide guidance to an automated vehicle The developed technology of the guidance can be applied to achieve the development of automated transit vehicles for IVHS (Intelligent Vehicle and Highway Systems) In such applications, vehicle guidance is achieved by focusing a CCD camera onto a road segment, containing the lane and road, and extracting the centre of the lane by processing the image. The image data from the CCD camera is processed by an array of parallel DSP processors The DSP based controller in a PC environment carries out the task of high level control while the low level servo control is assigned to dedicated motion controllers communicating with the DSP based controller.

In this thesis, the CCD camera is focused onto a road segment containing a line that has suitable contrast with the road surface as in CONCIC-2 AGV. Captured video images of the road ahead are filtered for noise and enhanced for better contrast of the line on the road using parallel algorithms. The coordinates of the line are extracted from the images and a line fit is carried out to extract the guidance control parameters of the vehicle. The guidance control parameters of the vehicle are the instantaneous position and orientation of the vehicle with respect to the road. A variable gain controller uses the computed position and orientation information to provide guidance to the vehicle. Dynamically varying gains are necessary to allow for autonomous driving and steering of the vehicle on straight or curved roads while increasing or decreasing the vehicle's speed. Experiments carried out with a prototype automated vehicle indicate encouraging results.

ACKNOWLEDGEMENT

The author wishes to gratefully thank Dr. Ramesh Rajagopalan for his insightful supervisory role, his undivided and prompt help, and his financial support during the course of this thesis work. Sincere thanks are also extended to Dr. R.M.H. Cheng for his criticism and interesting discussions

Many thanks are to Mr. Gilles Huard for his invaluable work on the design of all of the electronics on the experimental vehicle Thanks to the technical staff at the machine shop for their efforts in building the mechanical structure of the experimental vehicle and Mr. Pharand for designing the mechanical structure. Mr. Venkateswara Rao Ayyadevara provided the author with many hints and helped with Parallel C programming. Thanks to Mr. Andrew Szyngiel, Mr Serge Ghattas, and Mr. Peter Valiquet for donating their precious time during the long testing stages These testing stages could not have been possible without the cooperation of Concordia Security.

The author acknowledges the Faculty Research Development Program (FPDP) of the Concordia University for providing funding towards the project in the form of a grant (grant # RO20I891) and the Natural Science and Engineering Research Council of Canada (NSERC) for a research grant both awarded to Dr. Ramesh Rajagopalan (grant # RA20N445).

Finally, thanks to my parents for their moral and financial support all throughout my schooling.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATIONS

γ	sum of the residual squared used to approximate the curvature of a turn
ε_0	orientation offset
ε_d	position offset
θ	desired angular position in degrees
θ_s	steering command
θ_{s1}	left wheel steering angle
θ_{s2}	right wheel steering angle
ψ	angular velocity of vehicle
ω_{x1}	left wheel angular velocity
ω_{x2}	right wheel angular velocity
δt	sampling time for the LM628 motion controller
$\frac{\delta Q}{\delta c}$	partial derivative of the sum of deviations with respect to the offset
$\frac{\delta Q}{\delta m}$	partial derivative of the sum of deviations with respect to the slope
c	offset of line (y=mx+c) representing the track in the image view
d_s	derivative sampling time for LM628 PID filter
e_i	deviations of expected values ($y=mx_i+c$) from actual values (y_i)
f_{clk}	LM628 clock frequency
m	slope of line (y=mx+c) representing the track in the image view
n	sample time

n	steering motor gear ratio
r	wheel radius
s	steering motor encoder resolution times a factor of four
x_i	horizontal location of centre line in image view
y_i	vertical location of centre line in image view
A	linear acceleration
A_{LM}	acceleration in LM628 format
F	conversion factor (65536) for LM628 velocity/acceleration conversions
G_1	position offset gain in control loop
G_2	orientation offset gain in control loop
I_l	integral sum limit for LM628
K_d	derivative gain for LM628 PID filter
K_i	integral gain for LM628 PID filter
K_p	proportional gain for LM628 PID filter
L	wheel span - distance between contact points of front wheels
L_{ac}	effective wheelbase between contact points of front and rear wheels
N	driving motor gear ratio
P_c	position counts in LM628 format for displacement
Q	sum of deviations for least squares method
R	turning radius
S	driving motor encoder resolution times a factor of four
V	vehicle's linear velocity
V_{Fl}	left wheel linear velocity

V_{F2}	right wheel linear velocity
V_{LM}	velocity in LM628 format

CHAPTER 1

INTRODUCTION

1.1 Introduction

As the 21st century comes around, more and more efforts are employed to implement efficient driverless transportation systems. The vehicles of the next century will all contain powerful electronic systems that will provide self guidance Automated Transit Vehicles (ATVs) will slowly replace the role of the driver. Presently, the driver has a key role in driving a car. With the development of ATVs the driver will have a supervisory position as the vehicle takes him/her and his/her passengers from place to place. Many efforts are being made towards the creation of self guided vehicles. Many technologies are being experimented to achieve the goal of creating a fully automatic car. It is argued that the car of the future is not far from being realized. The stumbling block is the present state of the road system because it cannot support automated transport vehicles [1.1]. Once the road infrastructure will be adapted for automated vehicles, more and more types of ATVs will surface.

One of the main driving forces of ATVs is the increased highway safety that can be achieved with autonomous guidance of cars. Automation can prevent many traffic accidents from happening. For example, of the many factors contributing to accidents, driver's errors and fatigue can be eliminated. Also human behaviour is at times unpredictable and unpredictability is the cause of many accidents. Indeed, automation cannot be realized without taking into account many of the legal issues that need to be solved [1.1]. For example if there is a collision between two ATVs who is liable for the damages? The manufacturer of the guidance control system or the owner of the vehicle? These non technical issues will be resolved by the legislature once ATVs start to appear on the road.

As with any new technology it will be at first feared and then largely accepted. An analogy can be made to when automatic pilots for airplanes were introduced [1.1]. Pilots did not like the idea of not being in control of the airplane, and many pilots questioned the safety aspects of the system. Today in-flight systems control most airplanes with increased safety and comfort, and decreased costs.

1.2 Uses of ATVs

There are many uses for ATVs. Two main groups can be defined: transportation of people and transportation of goods. Automated vehicles could initially help drivers to drive more safely. Vehicles with guidance systems on board could provide the driver with additional information. For example in bad weather infrared cameras can locate pedestrians walking along side the road or crossing the road. Most drivers driving in a heavy snowfall or thick fog would not be able to see pedestrians, or even cars in front or beside them, until the very last second with possible catastrophic consequences. Many more sensors can be utilized on cars to tackle these situations. Changing road conditions will be detected and the driver warned or the vehicle will automatically take action reducing the speed or redistributing power to the wheels to prevent skidding.

ATVs could make drivers respect more the driving laws. For example, laser and sonar range finders mounted in the front of the car can measure the distance between the car and the one travelling just in front. If a minimum safety distance is not respected then the computer on-board the vehicle can give a warning sign to the driver or automatically slow down the car. Speed limits could be better respected since the on board computer system will keep the car at the maximum speed limit. On the other hand speed limits could also be increased since cars will be easier to drive, safer and less driver dependent.

To have a vehicle that is fully autonomous in operation, many technologies have to be merged together. GPS (Geosynchronous Positioning System) satellite information on the vehicle is necessary to locate the vehicle. CD-ROM maps of the roads and highways that can be used are required. Once the present location of the vehicle is identified then the CD-ROM map can be cross-referenced to determine the best path to reach the destination. Then the automated guidance system takes over and physically drives the car on the selected sequence of roads. The road and highway infrastructure would be ready for ATVs so that road information can be passed on to the vehicles For example if sections of the highway have more traffic the vehicle is informed and alternate routes can be selected. This could be accomplished by placing beacons transmitting data every so often, so that the pertinent information for that section of road is received only.

ATVs can also be used in transporting goods from one factory to another. ATVs could eventually replace the railroad system. The main drawback of the railroad is that it cannot provide service everywhere. Usually train tracks are found only between main locations. With ATVs goods can be shipped from any location where there is a road to any other location. The computer system would select the best route and then drive the vehicle to its destination. The implementation of the guidance system could vary from vehicle to vehicle, but the purpose would remain the same. A more immediate application of ATVs would be to ship goods from a warehouse to the factory floor. This repetitive work could be done better by a machine than people.

Costs are another important aspect when discussing possible ATVs uses. So far the technology to implement some of the automated systems is very high. This is due to the fact that most systems are still being developed or are in the testing stages. Once the automated vehicles become more of a main stream item then costs will drop by applying economies of scale while producing the systems. The highest costs would incur in the modification or creation of highway/road infrastructure to handle ATV traffic. However some systems are being conceived like the IVHS (Intelligent Vehicle Highway Systems) which recently has been renamed as ITS (Intelligent Transport Systems). These systems take into account the level of on-board automation of vehicles so that ATVs can fully benefit from all of their features. Also, vehicle manufacturers would have to get together and work on some standards for ATVs. The predecessors of ATVs are the Automatic Guided Vehicles (AGVs).

1.3 History of AGVs

These vehicles are mostly used within factory floors to transport materials from workstation to workstation. General Motors uses extensively AGVs in their light truck assembly plant in Oshawa, Canada. AGVs work at low speeds, up to 1 m/s, with a good degree of accuracy when moving from location to location. AGVs can also be found delivering mail in office buildings, cleaning floors, or used for surveillance purposes.

As computers became more powerful, more integrated, smaller, and more affordable better automated vehicles could be built reaching higher speeds safely. Vehicles travelling at higher speeds can reach further destinations in reasonable time. The idea of ATVs is still based on the AGVs with the new goal of reaching higher speeds and travelling in less structured environments. AGVs need a structured environment. Depending on the guidance control scheme used electromagnetic guidance wires have to be placed, beacons have to be located, lines have to be placed so that the AGV can obtain the necessary guidance information. ATVs can navigate themselves using the standard road information. In some cases special roads have been built to improve the ATV's performance.

It should be said that most ATVs of today are still in the developmental stage. Many experimental prototype vehicles are being tested and the results are encouraging, but no commercial units are available yet. Most research in this field is done by universities and car manufacturers. One of the main problems to overcome is the high cost of development associated with the computing systems, sensors, software development, and safety features.

1.3.1 AGV Development at the Centre for Industrial Control, Concordia University

Since 1982, the Centre for Industrial Control (CIC) of the Department of Mechanical Engineering, Concordia University has been carrying out extensive analytical and practical research and development in the areas of modelling and guidance control of Automated Guided Vehicles (AGVs). The Centre has developed several prototypes to implement a variety of guidance control schemes. The research and development carried out for this thesis makes use of those R&D activities of the Centre to provide improvements in road following capabilities of these vehicles. This section describes briefly the system architecture and features of these prototypes.

During 1983-86, Cheng et al [1.2] developed the first prototype vehicle CONCIC-1. The vehicle has a tricycle wheelbase configuration, with the front wheel both steered and driving. Vehicle guidance is achieved using an optic ram based binary digitizing camera and the track to be followed is prepared by taping or painting the floor black. Image data from the camera are analyzed and the location (position and orientation) of the track with respect to the longitudinal axis of the vehicle are extracted. Thiь information is used to steer the vehicle towards the track The vehicle controller is made up of three Z80 microprocessors, each responsible for a specific function: the Central Computer coordinates and delegates actions to the subordinate subsystems; the Vision Subsystem is dedicated to digitizing the images and extracting the path information; the Vehicle Motion Controller is responsible for the drive and steering motors. CONCIC-I can operate at speeds of up to 2 m/s.

With the support of a local AGV manufacturer (the TOR Group), in 1987, the Centre was successful in obtaining a strategic grant from NSERC to develop a modular AGV, CONCIC-2 by Rajagopalan [1.3], with a variety of features. This vehicle architecture (mechanical, electrical, electronics, and software) has become the basis for the future designs of AGVs at the Centre. In addition to modularity, the architecture of the vehicle was designed to eliminate some of the limitations of CONCIC-1. Instead of designing the drive and steer train mechanisms as in CONCIC-1, CONCIC-2 made use of motor-in-wheel-drive units from Schabmuller Corporation [1.4] as the drive/steer unit. This simplified the mechanical structure of the vehicle. As a result, the wheelbase of the vehicle is modifiable as the base is divided into 9 compartments of same size and the wheel units can be located in any of the 9 compartments. The original optic ram based camera had several limitations, some of which are, (1) smaller window size 128x128; (2) pixel transfer is contiguous and cannot be accessed randomly; (3) transfer rate is slow, 70 ms for 64x128; (4) more than one camera cannot be interfaced. Hence a DMA (direct memory access) based IDETIX [1.5] binary digitizing camera from Micron Eye Corporation is used. This choice has eliminated all the limitations of the original camera used in CONCIC-1 and provided a window of 128x256. The hardware platform is changed to Intel 80286-12 and motion control of servos were dedicated to LM628 motion controller chips from National Semiconductors [1.6]. The vehicle features an interface card with the ability to control 4 drive and 4 steer motors simultaneously. The 286 performs the high level control and image acquisition and processing while the low level motion control is carried out by the interface using the LM628 chips This master-slave configuration though it is similar to that of CONCIC-1, it simplified the vehicle hardware to a great extent and use of off-the-shelf chips led to modularity and provided the ability to change things with changes in technology. Off-the-shelf amplifiers from Galil Motion Control [1.7] are used. The control software is designed in such a way to provide many options (keyboard mode, camera mode, turning modes, remote link, data acquisition and testing) to operate the vehicle. The software shells are in modular form thereby allowing addition of new control schemes in the future to the existing control software of the vehicle. A variety of control schemes have been developed and implemented. The user can modify many of the operating parameters of the vehicle without having to recompile the codes.

Though the guidance approach is similar to the one of CONCIC-1 (line following by focussing the camera on a track prepared by taping the floor black), the vehicle has many control options, variable gains based dual camera control scheme for parking, a neural controller scheme wherein the weights of the controller are adjusted in real-time to negotiate tracks with varying curvature, a predictor-corrector scheme for high speed travel, ability to identify road junctions and turn. The top speed of the vehicle is limited to 1 m/s due to the limitations on the top speed of the motor sets chosen for the vehicle.

A dynamic model based simulation package was developed by Huang [1.8] to design a variety of control schemes and to optimize the various control gains for CONCIC-2.

CONCIC-3, by Mehrabi [1.9], is the next prototype vehicle designed and built at the Centre for Industrial Control. The structure and the motor drives are the same as the ones of CONCIC-2. The main purpose of this vehicle is to test experimentally the suitability of a dynamic model based optimal control scheme for dead-reckoning control.

The need and applicability of a Computed Torque Control (CTC) scheme for AGVs is studied by Barakat [1.10]. Modelling, simulation and experimental work have been carried out for differentially driven AGVs like CONCIC-2.

1.4 Components of an ATV

The components that make up an ATV are very similar to the ones for an AGV [1.3]. There are three main component groups: the computer system, the sensors, and the actuators. The computer system is the brain of the vehicle. The sensors provide the road information to the computer system and the actuators physically perform the actual driving of the vehicle. A chassis hosts all of the components.

The computer system can be made up of different types of hardware: single processor, multiprocessors, and distributed processor. They all have the same goal, to provide good guidance control to the vehicle. Computer systems have become extremely advanced, compact, and fast, which are all requirements for ATVs. The amount of information to be processed is high and therefore powerful computer systems have to be employed. The software that runs on the computer system has to be compact, efficient and has to be able to take full advantage of the computational speed of the processor(s). The computer system has to be reliable since a computer failure could be disastrous to the ATV. In all cases manual overrides are necessary, just in case that something goes wrong, the vehicle could still be controlled.

The computer system needs information about the road to provide guidance to the vehicle. Many types of sensors can be used. A video camera is one of the more versatile sensors because a single device can provide many types of information about the road. For example, from the camera images the centre of the lane can be located. Obstacles can be detected. The geometrv of the road ahead of the vehicle can be stored and used either for improved road following or for identifying the location of the vehicle from a stored mapped. Infrared cameras can be used when the visibility is reduced like at night or during a storm or in fog. Sonars and radars can detect the presence of obstacles around the vehicle so that the computer system can either avoid the obstacles or stop the vehicle. This is useful during a lane change manoeuvre. Laser range finders are used to detect the distance of the vehicle immediately ahead so that it is not followed too closely.

Actuators are necessary to provide the mechanical action (driving and steering), based on the commands given by the host computer. Different types are available but they all share the same function: to translate an electronic signal into mechanical work. Actuators that steer the vehicle are usually made up of electric servo motors. To increase the speed of the vehicle a throttle control unit can be used or in the case of an electric driven vehicle servo amplifiers are used. Braking the vehicle can be accomplished by mechanical means or for an electric driven vehicle the servo amplifiers can provide a counter current to stop the motors. Other auxiliary sub-system can be present on an ATV depending on the purpose and complexity of the vehicle.

1.5 Summary

This chapter provides an introduction to the possibilities for automated or driverless transportation systems. Many uses for automated transportation vehicles can be envisioned and many more will appear in the future. Automated systems can increase road safety by providing the driver with additional help and information so that less accidents will occur. Goods can be shipped using the automated transportation vehicles thereby reducing driver's fatigue and errors. ATVs have developed from AGVs which are bound to more structured environments like inside a factory. AGV also work at much lower speeds. Most ATV systems are in the development stages and a few prototypes are available. Automated transportation vehicles are made up of three main components all mounted on a common chassis. A computer system which is the brain, sensors that read the environment, and actuators that translate the computer's electronic commands into mechanical commands to guide the vehicle.

CHAPTER 2

LITERATURE SURVEY AND PROBLEM DEFINITION

2.1 Introduction

Automated Vehicles can be made in different shapes and forms, however, beneath the outside looks, they all share some common elements. The most important one is a vehicle controller made up of a computer system. This system is the 'brain' of the vehicle because it receives information about the environment from sensors, processes this data, and through actuators it performs guidance control of the vehicle. There are several types of guidance systems mainly defined by the kind of sensors used to detect the environment. The propulsion system can range from gasoline engines to electric motors.

This chapter discusses the literature available on Automated Guided Vehicles (AGVs) focusing on the vehicle controller systems and guidance schemes utilized by some prototype automated vehicles. The chapter is divided into three main sections. The first one summarizes the literature on AGV controllers and guidance schemes. The second section defines the objective and scope of the research presented in this thesis. The last section briefly describes the organization of the thesis.

2.2 Present Automated Guided Vehicles (AGVs)

This section describes the existing prototypes of AGVs. Two of their main components are discussed: the vehicle controller and the guidance system. The vehicle controllers can be implemented in many ways using different types of processor(s) and hardware. However they all share the same purpose, to receive and process the information from the environment and to provide guidance to the vehicle. The guidance system is determined by the type of sensors utilized to detect the road or the surrounding environment. Also the AGV's application determines the kind of sensor(s) that is (are) going to be implemented on the vehicle.

2.2.1 Vehicle Controllers

All vehicle controllers involve some kind of computer system. The system can be based on a single processor, or on multiple processors in a distributed system, or on multiple processors working in parallel, or on various processing systems interacting with one another. The following three sub-sections describe the various vehicle controllers developed by researchers. Section 2.2.1.1 discusses some of the earlier system utilizing a single processor. To overcome the limitation of using a single processor, research moved in the direction of multiple processors, each in charge of a specific task. Section 2.2.1.2 describes the distributed processing vehicle controllers. To meet the further demands for increased computing power in vehicle controllers, controllers, made up of various processing systems including parallel processors, have been developed. Section 2.2.1.3 introduces the applications of parallel processing in the development of vehicle controllers.

2.2.1.1 Single Processor Vehicle Controller

Most of the earlier vehicle controllers consist of a single processor performing most tasks [2.1, 2.2, 2.3, 2.4, 2.5]. All of these systems lack computing power and due to the sequential nature of the earlier processors many restrictions have applied when developing the software code. All operations have to be executed sequentially: receiving the data from the sensors, processing the data to provide guidance control. All of the following vehicles have been designed for low speed operation and all successfully achieved their goals.

The Intel 80x86 processors were mostly used. Ishikawa et al [2.2] implemented a vehicle controller using an 8086 microcomputer at 5 MHz. A 80286 at 12 MHz with a 80287 math coprocessor was the hardware platform used to implement CONCIC-2 by Rajagopalan [2.4]. As faster processors were developed they were implemented in vehicle controllers. Cho [2.1] used a 80386 for a mobile robot with stereo vision. All of the above mentioned system had vision as a means of guidance. A Motorola MC68020 with an MC68881 math coprocessor computer is the basis for the Blanche autonomous vehicle from AT&T Bell Laboratories by Cox [2.5]. A SUN-3 workstation is used by Sharma & Davis [2.6] just to perform some of the processing of the vision images and in the worst case it takes 34 seconds to complete. They suggest that parallel processing is necessary to achieve real-time control using their vision algorithms. The advantage of using the above mentioned computers is that they are available commercially, software development tools like compilers, assemblers, and operating systems, are also readily available. The disadvantages are slow processing speeds especially for vision guidance and the sequential nature of operation of these processors.

2.2.1.2 Distributed Processing Vehicle Controller

In order to increase the computing power of the overall vehicle controller and to overcome the sequential processing limitations, distributed processing is implemented in many vehicle controllers. These systems use many processors, each dedicated to perform a certain function of the system. The individual processors do not necessarily have to be much faster than the ones described in Section 2.2.1.1 on single processor vehicle controllers. Each processor performs a single task and then passes on the results to the master processor. Data from multiple sensors can be received by a dedicated processor for each sensor. Then the analysis can be done by other processors in charge of just analyzing the received information. Other processors perform the output commands to the actuators.

The prototype vehicle ALVIN by Turk et al [2.7] takes advantage of multiprocessor distributed computing. The vehicle controller is made up of a VICOM image processor that is dedicated to the digitizing of the camera images and the processing of such images. A laser scanner processor interfaces the laser range scanner unit and interprets the data. The master processor is an Intel 80286/80287 and it is responsible for running the overall system. Its bus is shared by three other processors: a navigation processor based on a 80816, a vehicle control processor based on a 8086, and a multichannel controller using a 8089 that selectively receives the already processed sensory data from the vision and laser systems.

HILARE 1 and HILARE 1.5 by Noreils and Chatila [2.8] also use several processors to acquire all of the sensory data, to manipulate the data, and to guide the vehicle. The sensory inputs of stereovision, laser range-finder, ultrasonic array, and odometry from the wheels are handled by five 80286s and 8086s all sharing a MULTIBUS I bus. A radio link at 9600 baud communicates back to a SUN workstation where the stereo camera images are displayed. The stereo images are manipulated and processed by dedicated DATACUBE boards.

Navlab of Thorpe et al [2.9] at Carnegie Mellon University also uses a distributed system. The colour vision is processed by an experimental processor called Warp, the laser range finder is handled by another processing system, a SUN 3/160 workstation displays the images, monitors the vehicle, and guides it. The computer systems and the control electronics fills a minivan.

Even one of the earliest prototypes of mobile robots the Carnegie Mellon University Rover by Moravec [2.10] used a distributed architecture. Fourteen on-board processors, a remote link to a VAX 11/780. and an ST-100 array processor for image digitizing and processing. Of the 14 processors, six Motorola 6805s were dedicated to motor control. Four other 6805 were used for moving the tilt/pan/slide motors for the camera, the sonar data, the infrared proximity detectors, and the proximity switches. Four Motorola 68000s controlled the communication to the VAX system, performed some local computing, synchronized the motion with the 6805s. A serial link was used to communicate among all processors on board the vehicle.

Another early test vehicle was the Melboy by Tani et al [2.11] using a 80286/80287 combination on board the mobile robot to monitor the hardware status, controlling actuators, sampling environment information, using external commands giveı. by the host computer and to log data. The host computer, a 80386/80387, performs the overall control of the system and interfaces with the operator. Communication between host and on-board computers is through a wireless modem.

A similar system to the above is the Spot-Mark vehicle by Takeda et al [2.12]. This system has all of the computing power c n-board. A vision system (SV) processor analyzes the video data and a vehicle control system (VCS) performs all of the guidance control. A more complex distributed system using a LAN netwcrk as a means of communications between processors is the vehicle controller for GSR (Ground Surveillance Robot) by Harmon [2.13]. Transputer systems have been implemented as well. An inertial navigation system by Barshan & Durrant-Whyte [2.14] uses the INMOS T805 transputers in a network to process the information from three gyroscopes, a triaxial accelerometer, and two tilt sensors. The transputers individually process the data from the sensors and then recombine all the results for navigation status.

The advantages of using distributed processors is that more overall computing power can be achieved by adding processors. Each processor is dedicated to a specific task, usually interfacing a device, or computing some parameters. These dedicated processors free the main processor to have to interface with all the sensors, actuators, and sub-systems of the vehicle. However, monitoring from a master/main computer is usually necessary. Complex custom-made hardware and software has to be designed. The overall system's performance is still dependent upon the weakest link of the system, usually the vision system. The overall guidance control still has to wait for the image processing and analysis to take place before it can provide vehicle guidance.

2.2.1.3 Parallel Processing Vehicle Controller

To overcome the computational burden of image processing, parallel processing has been implemented by [2.15, 2.16, 2.17, 2.18]. All these vehicle controllers add parallel processing to their distributed computer systems. The necessity of parallel processing arises when dealing with the large data from a camera or other types of vision systems. As for camera vision, the images can be divided in sections and each individual section can be processed at the same time as the others by different processing units. Since the algorithms executing in each processing unit are usually identical and they execute at the same time it is called parallel processing. By adding more processing units, the overall execution times can be reduced.

Jochem & Baluja [2.15] from Carnegie Meilon University have implemented a parallel colour image processing system that has been installed on the test vehicle NAVLAB I. Their design uses massively parallel processing, using the MasPar MP-1 a massively parallel computer. The MasPar is a Single Instruction Multiple Data (SIMD) machine. It is made up of an array of 4096 processing elements (PE) and an array control unit (ACU). The array control unit provides the processing elements with the same instruction at once. The processing units that are active execute the instruction. This massively parallel system is used for clustering the colour images into road and non-road. Each pixel is handled by a processing element. The rest of the guidance is accomplished by a SUN 3/160 workstation and other computers all interconnected.

The COMODE by Rygol et al [2.16] uses a complex distributed system involving a parallel processing unit for the three dimensional stereo vision. The parallel processing unit called MARVIN is made up of an array of 25 INMOS T800 transputers. Specially developed parallel vision algorithms have been implemented, in Parallel C, to execute on the transputer array. High level control responsible for maintaining the goals of the system and providing a user/graphical interface is done by a SUN 4 workstation. The vision data is processed by the transputer array and sent to a SUN 3/110 workstation through a VME bus for further processing. The vehicle control is overseen by a SUN 3/60 workstation. The SUN workstations interact using client/server applications. The client workstation requests information to the server. The stereo images are grabber simultaneously by two Datacube Digimax frame grabbers and sent to the transputer array. The cameras can pan left and right under the supervision of another T800 transputer and Z180 processors. The drive wheels are controlled by motion controllers.

TOYOTA also uses a parallel processor for image processing in a distributed computer system in its automated highway vehicle system [2.17]. Laser and radar systems are handled by a electronic control unit (ECU). Steering, throttle, and brake systems all used separate ECUs. The vehicle control computer hosts all these systems.

Parallel processing can be used for other applications other than video processing. Blais et al [2.18] use parallel processing in their BIRIS laser range finder for navigation of mobile robots. The Motorola 68020 processor and MaxVideo image processing boards make up the BIRIS system. Another application of a large array of T800 transputers, that could be useful in automated vehicles as the vision engine, is the Kodak parallel image processor [2.19]. This system consists of an array of 120 T800 transputers programmable in C++ to perform any kind of image processing. So far it has found application in image editing workstations, image recognition, and photo analysis, just to name a few. Sharma & Davis [2.6] in their work on road boundary detection suggest the use of parallel processing as the only way to analyze three dimensional range data in real-time.

Parallel processing has led to many advances in applications of real-time image processing in automated vehicles. The complexity in vehicle controllers is increasing due to the necessary interaction of multiple computer systems to achieve real-time guidance control of the vehicles. Most of the parallel architectures are designed for specific applications. The results are encouraging especially with the introduction of Digital Signal Processors made to handle simple operations very fast and capable of multi-tasking and parallel processing using multiple processors.

2.2.2 Guidance System

The guidance system is the only mean for the vehicle to know about the surrounding environment. Some kind of guidance system is necessary for any type of automated vehicle. The application of the automated vehicle in many cases determines the type of guidance system that could be used. Camera vision is by enlarge the most common type of guidance system utilized by automated vehicles. Camera vision can provide a variety of information about the environment depending on how the camera images are treated, processed, and analyzed. Multiple sensors can be implemented as well. Infrared sensors can be added to a camera vision system to provide information during night driving, for example, that could complement the standard camera vision. The more sensors are used, the more computing power may be required to process all the information gathered by the sensors. Also the various kind of information have to be combined to provide guidance to the vehicle. The following sections describe some of the guidance schemes utilized in prototypes of automated vehicles. Section 2.2.2.1 describes the various types of camera vision systems: colour vision, grey-scale vision, stereo vision, etc. Other types of sensors are discussed in Sections 2 2.2.2 and 2.2.2.3 along with some systems using multiple sensors.

2.2.2.1 Camera Vision Systems

These systems all share one common element, the visual information of the surroundings, which is captured through a camera or cameras. How this visual information is analyzed and processed differs from system to system. Also the purpose of camera vision varies; it could be used to detect a white line on the road [2.2, 2.20, 2.21], it could be used for scene analysis to determine the road from the scenery [2.6, 2 9, 2.15, 2.16, 2.22, 2.23], or scene analysis could be used to build a map of the environment [2.1, 2.24]. All of these systems employ camera vision as the only means of vehicle guidance.

2.2.2.1.1 White Line Detection

One of the earlier system that uses the detection of a white line on the road for vehicle navigation is a system by Ishikawa et al [2.2]. It uses a 256x256 pixel camera window at 256 grey-scale levels. Edge detection techniques are used to detect the location of the line. Once the line is located it is matched by a field pattern recognition algorithm to determine the shape of the line. The overall process of image recognition and navigation takes 300 ms. No travelling speeds are mentioned.

PVS (Personal Vehicle System) and AHVS (Automated Highway Vehicle system) by Tsugawa [2.21] are two systems that use the white lines that delimit the road. The earlier PVS has two dedicated vision systems: one to detect the lanes on the edge of the road and another that uses stereo vision to avoid obstacles. PVS has been tested at speeds of around 10-30 km/h. AHVS uses a single camera to view the road. An edge detection algorithm detects the location of the road edges. The system has been driven at speeds of 50 km/h on a road with large curvature.

The HMMWV (army High Mobility Multipurpose Wheeled Vehicle) by Schneiderman & Nashman [2.20] has been tested at high speeds of up to 100 km/h while following lane markers on the edge of the road. It uses a single camera vision system. A Sobel operator is applied to the images to detect the edges of the lane markers. Then the images are binarized using a thresholding technique. The guidance is given using the geometry of the lane markers by a relative weighting of images technique. The newer images hold more importance towards the control. If however an image cannot be utilized because there is too much noise, then the information from the previous images is used for guidance. The drawback of this system is that since the camera views really far, low speed guidance, or sharp turns are difficult to achieve.

Khalfallah et al [2.25] implemented a mobile robot that follows a special track with camera vision. A CCD camera provides 512x512 images with 256 grey-levels. These images are binarized using a thresholding technique and the specially encoded symbols on the track are recovered and used for position recovery of the vehicle.

2.2.2.1.2 Scene Analysis

This type of camera vision requires more computing power and as a result most of the systems are not implemented in real-time or the vehicles travel at lower speeds The entire camera view is analyzed for identification of a possible vehicle path or for objects which have to be avoided. The ALV (Autonomous Land Vehicle) by Waxman et al [2.23] is one of the earlier vehicle that has used scene analysis for guidance. The images 256x256 pixels with a 256 grey-scale are captured from a single camera. The image processing involves: smoothing, gradient, extraction of dominant directions, Hough transformation to find the linear feature in the images. The system uses a bootstrap mode and a feedforward mode for guidance. The bootstrap mode is used initially when the vehicle is placed in an unknown environment. The image is analyzed in full to determine the location of the road. Then the feedforward mode is employed whereby only smaller windows in the image are used to identify the direction of the road. The windows' locations are estimated from the previous image. A visual knowledge base is also cross-referenced to identify landmarks or particular road types: curved roads, straight road, etc. Travelling speeds of 3 m/s have been achieved with this vehicle.

A modified vision system for the above ALV has been developed by Sharma & Davis [2.6]. It still uses the bootstrap and feedforward modes but the images are analyzed in three dimensions. Road boundaries are detected in three dimensions by matching a road model from the knowledge base. The images are extrapolated in three dimensions because the view is from a single camera. This vision system cannot be used in real-time because in the worst case it takes the algorithm 34 seconds to execute. Parallel processing and faster computers are possible solutions.

The COMODE by Rygol et al [2.16] uses stereo vision from two video cameras. Images are 512x512 pixels at 256 grey-scale. Stereo triangulation is experimented with to determine the exact location of objects within the camera views. The three dimensional structure of a scene is recovered using a set of geometrical primitives that correspond to real physical features. Operating vehicle speeds are not provided. The execution time to locate objects in a scene is around 5 seconds for 256x256 images and 10 seconds for 512x512 images. Using some feedforward techniques a stereo tracking algorithm has been developed to reduce the processing time to 200 ms. It has been reported that more computing power is expected to be achieved with a new forthcoming processor.

Scene exploration using a single camera is also employed by Boudihir et al [2.22]. The 512x512 256 grey-scale images are processed to reduce noise and enhance contrast. A Sobel operator is used to detect the edges, and thresholding is carried out to obtain binary images of the edges only. After a Hough transform to correlate the possible curves in the images, road edge equations are selected to represent such curves. The image is also divided into three zones. The closest zones are given more importance when selecting equations for the road edges. No computation times or vehicle speeds are reported.

Colour scene analysis is performed by two vision systems developed for Navlab at Carnegie Mellon University by Thorpe et al [2.9] and Jochem & Baluja [2.15]. Both systems use a single colour video camera. The colour pixels are classified or clustered in road and non-road pixels. Then from this classification a road fit is performed. New colour statistics are gathered for the new image colour classification to improve the algorithm especially in changing lighting conditions. Thorpe et al [2.9] have been able to drive the Navlab at speeds up to 50 cm/s. Jochem & Baluja [2.15] use a massively parallel processor to perform this clustering routine along with neural networks. This extra computing power allows Navlab to navigate at 10 mph.

2.2.2.1.3 Building Maps of the Environment

INRIA by Zhang & Faugeras [2.24] uses three video cameras to collect data about the environment to build a global map of the surroundings so that a navigational path can be constructed. Three dimensional line segments are reconstructed from the stereo vision. Also motion can be estimated from consecutive three dimensional frames. One of the difficulties encountered is the fusion of the three dimensionai data of possibly the same scene taken at different times. The navigational map is constantly updated as new information is acquired. The vehicle is then guided through this map. No practical test results are reported.

Y. Cho & H. Cho [2.1] use two CCD cameras to acquire stereo images of the surrounding to build a pseudo map of where the obstacles to be avoided are. A geometrical transformation is used to relate the stereo view to a three dimensional map of the environment. Edge detection is implemented to extract the objects' positions. This system cannot be used in real-time because it takes 15 seconds to process a series of images. To reduce the computational burden, DSPs (Digital Signal Processors) are suggested.

2.2.2.2 Multiple Sensors Including Camera Vision Systems

Other guidance systems utilize a multi-sensory base for retrieving information about the environment. The camera vision is complemented by other sensors usually to detect obstacles or to measure distances. The GSR (Ground Surveillance Robot) by Harmon [2.13] has many sensors to navigate itself in unknown terrain. A colour vision system provides the vehicle with information about distant terrain and the navigational goals. A laser range finder is used to determine distances. To obtain information about the local terrain a grey-scale vision system is used. This system provides the goals for local navigation. Ultrasonic proximity sensors determine the location of obstacles near the vehicle. Navigation and vehicle attitude sensors provide information about the relative vehicle position, roll, pitch and heading angle, forward speed, and rotational speed. No test results are mentioned.

Alvin by Turk et al [2.7] uses camera vision for road recognition and a laser range finder to detect distances of obstacles. Colour images are used along with colour clustering to determine what is road and not-road. After boundaries are found then guidance is provided. 480x512 colour images are processed. The amount of time between digitizing an image and producing a symbolic description of the road is 2 seconds. Alvin achieved speeds of up to 20 km/h on a straight road, free of obstacles.

Morgenthaler et al [2.26] developed a technique to combine data from a colour CCD camera and a laser range scanner to obtain guidance of a vehicle. Colour segmentation techniques are implemented to produce an image model of road and road edge points. The laser range scanner detects objects on the path and characterizes the terrain for offroad navigation. Range data from the laser system and video data are fused to provide navigational information to the guidance system. Early tests demonstrated speeds up to 20 km/h.

CONCIC-II by Rajagopalan et al [2.3, 2.4] uses binary vision to accurately follow a track and sonars to detect moving objects. Two binary cameras can be used to view the ground. A camera in mounted in the front and another in the back. The rear camera can be used for parking the vehicle or for better path following. Frame rate of up 25 frames per second are achieved and speeds of up to 1 m/s can be reached. A similar system has been developed by Ghayalod et al [2.27] that also follows a line on the ground and uses ultrasonic sensors to avoid obstacles.

The CARMEL vehicle by Zhao et al [2.28] could be a first step in the direction of Intelligent Vehicle Highway Systems (IVHS) since it combines camera vision with stored maps of the local environment. The section of map pertaining to the local position of the vehicle is stored in the computer's memory the rest of the map can be stored else where (CD-ROM for example). The vision system provides the necessary information to locate the vehicle. Preliminary tests have been conducted yielding speeds of 0.78 m/s with a sampling time of 0.03 ms.

One of the most complete systems is the TOYOTA AHVS (Automated Highway Vehicle System) [2.17]. It uses a grey-scale image from a CCD camera to detect the lanes of the road and to provide lane guidance control. A laser and radar system monitor for other vehicles in front or for objects to avoid. Not much technical information is provided on how the systems work. The guidance system has been installed in a modified Toyota car and driven autonomously up to 60 km/h. The white line position is computed by a parallel image processing unit every 100 ms.

Other car manufacturers have prototype vehicles developed [2.17]. For example, Mazda has MOVER-2 a vehicle that detects the white line with vision. Nissan uses PVS to test its automated vehicle system that also uses white line detection with vision. Volkswagen has a prototype system that can travel at 100 km/h on a special road using side wall detection using lasers. Not much information is available on those systems.

2.2.2.3 Other Sensors Systems

Camera vision is not used in the following systems that utilizes other sensors to detect the road and the guidance path [2.11, 2.14, 2.18, 2.29-2.31]. A custom built laser range finder BIRIS has been developed by Blais et al [2.18] to acquire range information for navigation of mobile robots. A CCD camera is used to pick up the reflections from a laser beam. Triangulation is also used to determine the location of possible obstacles and the location of the vehicle. No vehicle guidance tests are reported, however the accuracy of the BIRIS system is encouraging.

Laser range finder along with ultrasonic sensors are used in Melboy by Tani et al [2.11]. The laser beam is picked up by a CCD camera and using triangulation the objects ahead can be located. Ultrasonic sensors detect the distance to a facing wall with an accuracy of 3 mm. The mobile robot is also given an approximate map of the surrounding environment.

Ultrasonic sensors are used by P. van Turennout et al [2.30] in their wall-following autonomous mobile robot. The sensors measure the distance from the wall and the orientation of the mobile robot with respect to the wall. Dead-reckoning is implemented during short instances when the wall is not present. A speed of 0.4 m/s has been achieved with a distance error from the wall of a few millimetres. This system could be ported and adapted to follow the guard-rail on a highway at must higher speed

Cobart designed by Flynn [2.29] from MIT combines sonar sensors and infrared sensors to provide navigational guidance. Sonars can measure distances accurately, but not the shape of obstacles. Infrared sensors can detect objects much better, but cannot provide accurate distance measurements. The combined data is used to produce a refined map of the available workspace.

An inertial guidance system can also be used to navigate a mobile robot. Barshan & Whyte [2.14] have developed INS (Inertial Navigation System) that uses data from a gyroscope, a linear triaxial accelerometer, and tilt sensors to locate the movements of the vehicle. An initial estimate of position can be calculated in 1-2 seconds on a T805 transputer network.

An interesting guidance system has been developed by Deveza et al [2.31] using odour sensing. This project investigates the use of short-lived navigational markers made up of olfactory chemicals to guide autonomous mobile robots. Applications of this type of guidance system could consist of detecting gas leaks in pipelines by automated vehicles.

2.3 Problem Definition and Scope of Thesis

From the discussion presented so far, it can be inferred that the computing power is one of the major limiting factors in the development of automated vehicles for high speed transportation. This is particularly so for vehicles using a vision system as the guidance/navigation system. The other limiting factors for vision guided vehicles deals with the quality of the image obtained, the image processing speed and the accuracy with which the road information (curve fitting) can be deduced from the image data. The third area of improvement is in the development of a suitable control scheme (neural networks, fuzzy logic, adaptive, etc.) to achieve proper guidance.

This thesis deals with the following research and development issues pertaining to the development of high speed automated vehicles:

- Development of a High Performance Vehicle Controller.
- Development of a Guidance/Navigation technique using a CCD Camera.
- Development of Parallel Processing based Hardware and operating Software Interfaces for guidance and control of a Prototype CONCIC-4 Automated Transit Vehicle (ATV) available at CIC.

It should be mentioned here that the analytical approach and the guidance technique developed by Rajagopalan [2.4] for binary camera vision based automated guided vehicles are adopted in this thesis. A prototype vehicle, CONCIC-4, similar to that of CONCIC-2 developed by Rajagopalan [2.4] has been developed by the research staff of the Centre for Industrial Control (CIC) for implementation. Though this work makes use of the previously developed techniques, the essential difference lies in the way the vehicle controller is developed and in the way the images are processed.

The vehicle controller is based on the recent advancements in the area of parallel processing technology. An array of TMS320C40 parallel processing modules are used to develop the vehicle controller and Parallel-C software environment is used to develop the required control and the operating system software of the vehicle. Further, the image window of the CCD camera used in this work is much wider (512x512 pixels) compared to the binary camera (128x256 pixels) used in CONCIC-2 by Rajagopalan [2.4]. Hence, a large amount of data need to be processed. Though the prototype vehicle has not been tested at very high speeds due to limitations on the top speed of the driving motors and also for safety reasons, the high frame rate combined with a large window (though CONCIC-2 provides higher frame rate (25 frames/s) compared to 15 frames/s of this vehicle, its image window is 8 times smaller), it can be said that the vehicle can be operated at higher speeds thereby meeting the requirements of Intelligent Vehicle Highway Systems. The research and development presented in this work is a stepping stone to the development of high speed automated transit vehicles. The Centre for Industrial Control (CIC) has been very active and has contributed significantly to the development of prototype Automated Guided Vehicles (AGVs), use of parallel processing for real-time control, motion control of AGVs and robots and kinematic and dynamic modelling. This research supplements that knowledge by using DSP for real-time control and image processing and using CCD camera for vehicle guidance.

2.3.1 Development of a High Performance Vehicle Controller

From the literature review it has been observed that one of the main limiting factors of most automated vehicles rests on the available computing power Most systems employ camera vision for guidance purposes. In some cases other sensors are used in combination with camera vision. The amount of information that has to be processed can overwhelm many processors. Solutions so far involved distributed processing, massively parallel processors, dedicated processors, and minicomputers. In most cases, these systems use specially designed processors, lacking standard interfacing, and as a result proprietary interface cards have to be built. An alternative in solving this problem, as recommended in reference [2.1], is to take advantage of parallel processing using Digital Signal Processors (DSPs).

A high performance vehicle controller can be built around the Texas Instruments TMS320C40 DSP or C40 for short. The C40 is a multitasking parallel processor capable of 275 MIPS (million instructions per second) and 50 MFLOPS (million floating point operations per second). Many processors can communicate with one another due to the dedicated hardware links. There is large hardware and software support for this fast processor which make it perfect for real-time control. Processing modules, digitizing boards, display boards are available with C40 DSPs. Compilers and programming tools are readily available as well. The C40 can be programmed using Parallel C, which is a version of C for parallel computing. Transputers could also have been selected for designing a vehicle controller since they are also programmed in Parallel C, however their computational speed is not as high as a C40 DSP. For example, a T800-20MHz transputer provides 1.5 MFLOPS which is 33 times slower than a C40-40MHz DSP. The Intel's 80x86 family of processors cannot be used as vehicle controllers because they lack parallel processing capabilities even though their computational speed is high. Furthermore, using C40 DSPs and the PC as the host computer is a favourable choice as the C40s have access to the PC bus therefore they can communicate to motion controllers, like the LM628, directly. The transputers hosted in the PC could not directly access the PC bus and special interfaces have to be built to communicate to the motion controllers

Using C40 DSPs, a network can be built to perform all of the above tasks necessary to implement a vehicle controller. The controller has to digitize the CCD camera images, process them, retrieve the road information, perform the guidance control of the vehicle for road following, and display the processed images along with other vehicle parameters. This C40 network also has to provide information to the dedicated motion controllers to control the electric motors that drive and steer the vehicle. The necessary degree of parallelism has to be investigated to determine the minimum size of the C40 network. Also this network is scalable meaning that if more processing is required, more C40 DSPs can be added. If a higher frame rate is needed then more C40s can be used to process images in parallel.

2.3.2 Guidance System

Most autonomous vehicles utilize camera vision as a means to acquire information for guidance. Other sensors can be used in conjunction with camera vision. In most cases the intended application of the vehicle determines the necessary sensors to use. For this research the automated transit vehicle follows a road using the line on the road that has suitable contrast with the road surface. A common CCD camera can be used to look at the road ahead. The images do not have to be in colour because to identify a line on the road there is no need for colour clustering [2.15], colour classification [2.9, 2.23, 2.26] or any other colour processing technique. Grey scale images are easier to manipulate since the amount of data is reduced by a factor of three. This is due to the fact that to represent a colour pixel 24 bits are necessary (8 bits per primary colour, RGB, red, green, blue). For grey-scale pixels only 8 bits are used, therefore one third the data. The road and the line provide enough contrast to be represented by a grey scale.

The visual data is digitized by a C40 based frame grabber and it is shared between processors, working in parallel to filter out background noise and to enhance the contrast of the line on the road with respect to the background. The road information is extracted from the processed images by scanning the rows looking for centre coordinates of the line. Once the centre line coordinates are identified then a line fitting algorithm computes and equation representing the line. From this equation of the line the position and orientation of the vehicle with respect to the road are calculated. The controller uses the position and orientation data to provide for vehicle guidance. A variable gain controller is proposed in this thesis to allow the vehicle to drive autonomously in straight and curved line roads at any speed up to the mechanical limit of the motors (2.4 m/s).

2.4 Layout of Thesis

The thesis is organized in ten chapters including this one and the introduction. Chapter 3 and 4 describe the computing hardware used and the hardware and software implementation of the operating system for the ATV. Chapters 5 and 6 explain the image processing routines along with the road centre line coordinate identification algorithm. The guidance control scheme is discussed in Chapter 7. An overview of the components that make up the experimental vehicle used in this thesis can be found in Chapter 8. The experimental results are discussed in Chapter 9, followed by a chapter that concludes the achievements and points in future directions.

A description of how processors work and a summary of test results of their performance is found in Chapter 3. This chapter discusses three processors that were available for testing. Their advantages and disadvantages are elaborated along with their performance. The qualities that are looked for in a processor suitable for image processing are speed and parallel processing capabilities.

The design and implementation of the operating system for the ATV is described in Chapter 4. The chapter is divided into two sections: one dedicated to the hardware implementation and another to the software implementation. The hardware section discusses three aspects of the hardware implementation: the user interface, the motor interface, and the camera vision guidance control unit. The user interface allows the user to set up and control the vehicle. The motor interface controls the motors which receive commands from the control loop. Motion controllers are used to provide the low level motor control so that the host computer can become free to execute other tasks. The camera vision guidance unit is based on a network of C40 DSPs that digitize the CCD camera images, process them, extract the road information, and provide the necessary data to the guidance control loop of the ATV to follow a track. The software implementation section describes the six functional software blocks. The main block is the one that organizes all of the functions of the operating system. The grabber module captures the images from the CCD camera and sends them to the process blocks where images are processed to reduce the noise level and enhance the road information. A display module receives the processed images and displays them for the user. Operating parameters of the vehicle can be viewed and modified from a parameter module

Chapter 5 describes the image processing routines utilized to enhance the road and to filter the background noise. Each routine is explained in some detail and the performance of the routines is decided in terms of quality of results and processing speed Once the quality of the images is obtained then an implementation of these routines is tested on the C40 network to achieve 15 frames/s.

To retrieve the road profile information the images are analyzed so that the road centre coordinates can be identified. This algorithm is discussed in Chapter 6. When calculating the centre line coordinates corrections have to be made since the camera viewing angle is not perpendicular to the ground. The camera views less road closer to the vehicle and more on the farther side. Then, a straight line fit is computed from the centre line coordinates. A straight line fit is used for simplicity in later identification of the vehicle's position and orientation with respect to the road.

The guidance control scheme is described in detail in Chapter 7. Two guidance parameters, namely the position and orientation offset are defined. The computation of the steering angle using a proportional controller is discussed. The steering angle is calculated using both the position offset and the orientation offset. The experimental vehicle has two front drive and steering wheels and two driven wheels in the back. In order to feed the wheels with the correct velocities and steering angles geometric properties have to be used. Once each wheel's velocity and steering angle is computed these values are passed on to the low level motion controller that executes the required motions. Through extensive testing of the guidance control it is found that dynamic position offset and orientation offset gains for the control loop are necessary to achieve good track following performance in various track profiles and speeds.

Chapter 8 provides an overview of the seven main component groups that make up the vehicle. The mechanical structure is discussed along with the control components including the servo amplifiers and the motors. The guidance system components like the CCD camera and camera's view are explained. The power system of the vehicle is discussed. Electric power for the vehicle comes from six batteries and is used to drive the motors, to run the computers and the CCD camera. A data acquisition system records useful information about the vehicle's behaviour so that its performance can be evaluated during the testing stage. There are also some safety features that stop the vehicle in case of malfunction.

Experimental results are shown and discussed in Chapter 9. Calibration of the PID filters for the motion controllers is described. The results for straight track following are presented in this chapter. A discussion on the tuning of the control loop through the following of straight track with a sudden offset is included in the chapter. Results for following a curved track and an S-shaped track are shown. Lastly the results of using dynamic coefficients for the position and orientation offset gains in the control loop are discussed.

Chapter 10 focuses on conclusions and recommendations for future work. Two appendices are included: Appendix A describes the software layout so that the user can be familiar with running the vehicle and Appendix B explains in more detail the LM628 precision motion controller used for servo control of driving and steering motors.

CHAPTER 3

HARDWARE PLATFORM

3.1 Introduction

Speed and power are the two most important factors when dealing with image processing. In practice one needs a combination of both to obtain useful results in the field of image processing. Another key issue when implementing real-world systems is parallel processing. Virtually all systems have some degree of concurrency and therefore to best represent them parallel computers can be used [3.1]. Parallel computers capture all of the three attributes described above: speed, power, and parallel processing. Parallel computers achieve their speed and power by dividing up events and working on them simultaneously as they occur. One processor can handle several tasks by switching rapidly among them, or many processors can work on parallel tasks distributed among them [3.1].

In this chapter the general architecture of RISCs (Reduced Instruction Set Computer), CISCs (Complex Instruction Set Computer), transputers, and DSPs (Digital Signal Processors) is discussed in section 3.2. The T800 transputer is explained in section 3.2.1. The Intel's 80x86 family is discussed in section 3.2.2. Section 3.2.3 gives insights about the Texas Instruments TMS320C40 DSP. The results of some performance tests are discussed in section 3.3.

3.2 Processors Architecture and Performance

Conventional processors, both CISC (Intel 80x86 family, Motorola 68000 family) and RISC (Motorola 88100, Intel 80860, SPARC, AMD 29000), have sequential architectures and therefoie are not equipped to fully exploit the parallel nature of many applications. This results in tasks being handled one at a time, even thought they might actually occur together and in an interrelated fashion. In these applications, conventional processors have already reached their computational limits and new architectures need to be investigated Transputers and Digital Signal Processors (DSP) are the next logical step. However, before describing the advantages of these processors a quick revision of the CISC and RISC architectures might help in understanding why transputers and DSPs are better choices for parallel applications

A clear explanation of the main architectural differences between CISC and RISC processors can be found in Graham and King, The Transputer Handbook [3.1]. CISC stands for Complex Instruction Set Computer. These machines were designed with the concept that to increase performance the number of times a code is retrieved from main memory needed to be minimized. Therefore, individual instructions were built to be very powerful. A single instruction might initiate the execution of a long microcode sequence. It is ideal to have a different microcode sequence for every instruction available, so that less memory accesses would be required. The drawback to this scheme is that complex microcode decoding circuitry had to be used. When simple instructions had to be executed they still had to go through the complex decoding circuitry and system performance suffered. It was later found that only 20% of the instruction set commands perform 80% of all operations in most computers. Also most complex instructions were rarely being used Therefore, the idea behind RISC, Reduced Instruction Set Computer, took form Processors with simple instructions that would be executed in few clock cycles were designed. Simple instruction decode circuitry was implemented to facilitate rapid program execution. The second principle was that RISC machines would have a load/save architecture able to reduce the number of external memory accesses. Only load and store instructions should access memory, and all computational operations should manipulate data placed in registers within the processing unit. The problem with RISC machines is that at times to execute a specific task many simple instructions need to be used, whereas in a CISC machine the same task can be executed by one single instruction Therefore it can be said that RISC architectures are optimal for some applications but might lag in others. In terms of costs, RISC chips are cheaper since they are much easier to design and produce than CISC processors. They provide a higher performance versus cost than CISC chips [3.1].

Transputers and DSPs are extremely fast microprocessors that outperform RISC and CISC processors by a wide margin in many respects. Transputers and DSPs combine many advantages of both RISC and CISC processors, with DSPs being a further evolution of transputers. The word 'transputer' originated from the joining of the words transistors and computers. They perform instruction prefetch and pipelining so that the CPU is kept busy by preventing wait states due to instruction fetching. This decouples the instruction execution time from memory speed. Transputers and DSPs have a reduced instruction set which however exceeds RISC by using microcoded instructions to execute useful complex operations Also parallel processing requires dexterity in message sending and task switching. These instructions were also implemented in microcode as to minimize the memory accessing time. Also DSPs and transputers implement a 'hardware stack' architecture which greatly differs from the RISC register intensive 'load and save' architecture. RISC have simple instructions and often many of them have to be executed to obtain useful results. Therefore a large number of registers is required to store the intermediate results of several operations. As many as 100 registers might be used in RISC. This could be a drawback when multitasking because over 100 registers have to be saved to memory and subsequently restored. This operation is time intensive and programs tend to be longer since registers have to be saved at all times. DSPs and transputers use a stack to store intermediate results and work when executing long operations or multi-tasking. Data is pushed and popped from a stack and there is no need for register addressing and shorter code can be written. Also workspaces are assigned to each process and the processors are very agile in switching states since information is constantly being saved [3.1].

DSPs and transputers distance other types of processors by a great margin when combined together to form networks. They can provide concurrent processing as a single computer or as a node within multicomputer networks When executing parallel processes DSPs and transputers use a 'Distributed Resources System' approach as opposed to the 'Shared Resources System' approach of conventional CPUs. In Shared systems, sequential processors (RISC & CISC) are connected to a common bus or path from which all can access the common memory and resources. The common bus creates a major bottleneck since only one processor at a time can access it and complex bus arbitration needs to be implemented. Therefore the performance increase of these systems by adding additional processors may significantly decrease when more than four processors are used. The only way to improve performance is by using fast (but expensive) memory that is much faster than the CPUs. On the contrary DSPs and transputers use a 'Distributed Resources System' model Each CPU has its own memory, program, and communication capabilities. Once each unit has finished some tasks, data is exchanged among CPUs through high-speed links provided between CPUs. These links also allow for many network topologies such as rings, stars, arrays, and pipelines. Since these links are independent DMA channels, data can be sent and received while performing other activities [3.1].

The three classes of processors (DSPs, transputers, 80x86) are put to the test to rate them in terms of computational speed. Each processor will be recursively executing the same mathematical function in order to establish the time required per operation. In some cases the processor has to execute the operation one million times to be able to get accurate timings, since the individual operation can be solved in microseconds. Other performance indexes will be shown but not really used, since many factors affect the performance of a processor: its internal structure, its environment, and its dexterity with floating point calculations. All of these factors can not be incorporated in the published benchmarks. MIPS (Million Instructions Per Second), MFLOP (Million of Floating-Point Operations per Second), Whetstone (floating-point performance index) are some of the benchmarks used to 'sell' processors by boasting exceptional performances [3.6]. In the following sections each processor is generally described and analyzed.

3.2.1 INMOS T800 Transputer

The T800 transputer is a 32 bit microcomputer containing a 64 bit floating point mathematical coprocessor, 4Kbytes of on-chip RAM, a configurable memory interface and four standard communication links. Figure 3.1 shows the blocks that make up the architecture of the T800. The transputer is characterized by a small number of registers, organized as a stack. There are limited instructions for accessing memory and no sophisticated memory-addressing modes. However, multiple processor architectures and inter-processor communications are designed into the chip and its instruction set. Of particular interest is the FPU (Floating Point Unit) since most operations of interest will

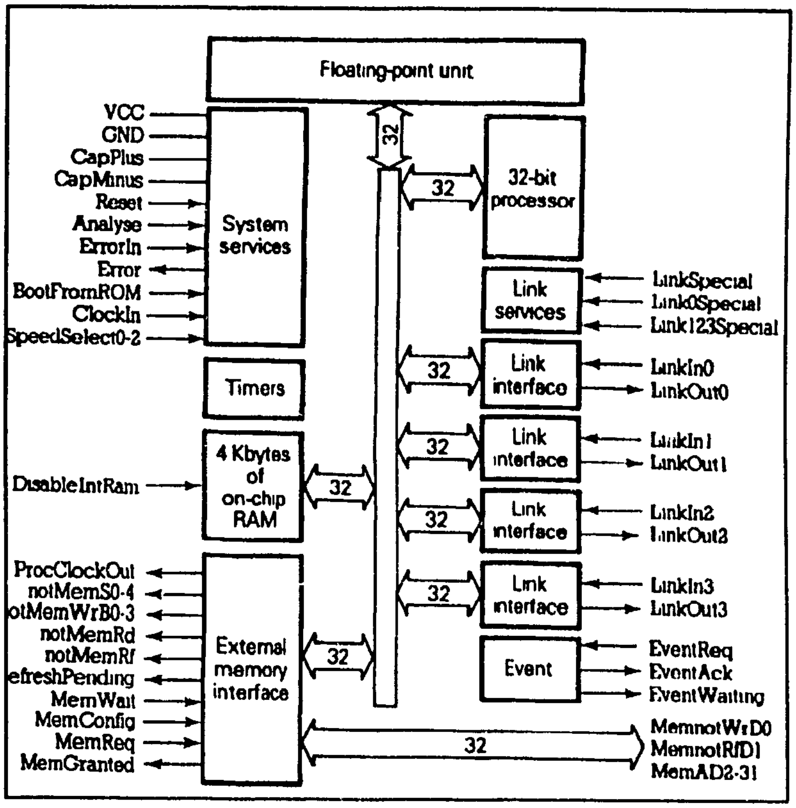


Figure 3.1 INMOS T800 Architecture [3.3]

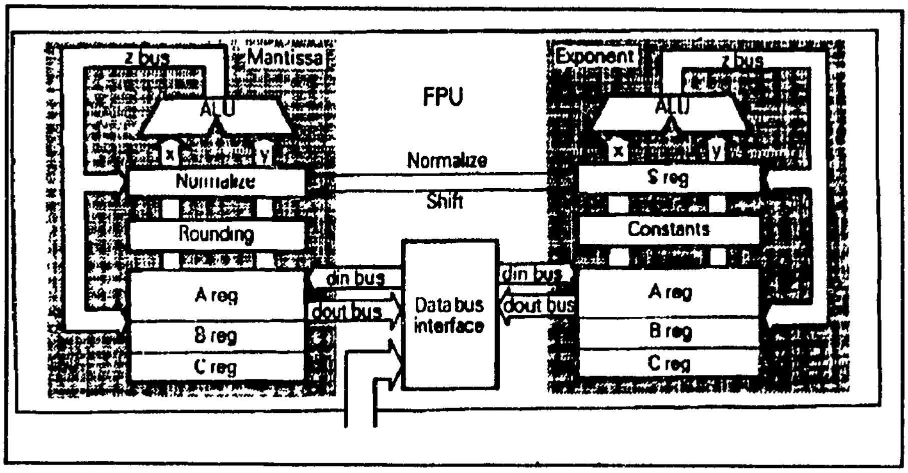


Figure 3.2 T800 Floating Point Unit Architecture [3.3]

involve calculations executed by the FPU. Figure 3.2 illustrates the architecture of the FPU. It should be noted that the CPU and the FPU operate concurrently. This means that it is possible to execute an address calculation in the CPU (since it performs all integer arithmetic) while the FPU is calculating the result of a floating point operation. This concurrency permits, for example, to calculate a floating point multiplication which takes several cycles to complete along which the CPU can continue executing the next section of code that computes the address where to store the result of the multiplication. Therefore the FPU never has to wait for the CPU to calculate addresses. As a result overall performance is increased [3.4].

Sincr transputers are designed for multitasking and multiprocessors operations, an interesting application of this philosophy can be obrerved in the implementation of the square root function. Since this operation is somewhat unusual it is implemented as a sequence of instructions rather than a single opcode. The reason behind this is that since the square root function takes a long time to be calculated the designers did not what to extend the potential interrupt latency period. Therefore with a sequence of instructions, interrupts can be serviced in between instructions [3.4].

In order to test the speed of the T800 executing mathematical instructions a simple program was run on a transputer board installed in a host PC computer. Even though the board has several transputers on it only one TRAM (T800 at 20MHz) module with 2 Mbytes of RAM was used to perform the test. The program is written in Parallel C and then compiled using Parallel C. The results of the test are organized in Table 3.1. It can be seen that all integer operations are much faster that floating point. This could be explained by the fact that the CPU (or the integer processor) has to route the floating point numbers to the

Mathematical Operation	Time no CFG (μs)	Time with CFG (μs)	format
Add/Subtract	0.2	0.91	integer
	1.1	1.24	float
Multiplications	0.8	1.44	integer
	1.7	2.24	float
Divisions	2.2	2.91	integer
	2.4	2.82	float
SQRT	18.6	22.44	float
SIN	51.2	75.64	float
COS	43.1	61.84	float
ATAN2	60.9	92.74	float

Table 3.1 T800 Timing Summary

FPU and retrieve them. Next, trigonometric functions require larger amounts of time to execute. Since there is no hardware provision for divisions, they have to be implemented in software and they take longer to be performed than multiplications.

3.2.2 Intel's 80x86 Processors Family

The early predecessors of the 80x86 family were the 4004 which was a 4 bit processor, the 8008 which was an 8 bit version of the 4004 and the 8080 which had a similar architecture as the 8008 but was more of a general purpose processor [3.7]. The 8086 (iAPX86) was the first 16 bit processor which became the standard for the rest of the 80x86 family. In fact, all subsequent processors kept compatibility with the 8086 instruction set. Before the 80286 release other less knows processors where released. The 8088 (iAPX88) was a step back from the 8086 since it was an 8 bit version the 8086. Then the 80186 (iAPX186) and the 80188 (iAPX188) were introduced for embedded control since both chips contained most of the electronics to work as a single chip computer. The 80286 (iAPX286) processor offered sophisticated mechanisms for implementing a stable operating system that supported data protection, multitasking, memory management, and direct addressing of up to 16 megabytes as well as virtual storage. However it took far too long before some of these features were implemented in an operating system [3.7].

With the 80386DX, Intel entered the 32 bit processor race. It was still compatible with the 8086 and the 80286 and provided support for 32 bit operations and data types, memory paging, and an expanded instruction set. Some other features included built-in debugging support, a virtual 8086 mode, direct addressing of up to 4 gigabytes of physical memory, and a linear address mode. A 80386SX processor was also released that supported

32 bit operations and 16 bit external data bus. Since its introduction the 80486DX became the standard of personal computing. The 486 had most features of the 386 plus it had a built-in floating point processor and an 8kbyte cache This processor will be described in more detail in the next paragraphs. The latest member of the Intel family is the Pentium processor

The 80486DX processor is able to achieve high processing speeds due to the fact that the CPU, the FPU, cache memory are all internal to the chip and signals are able to travel at high speeds over very short distances Internally, as it can be seen from Figure 3 3 the processor is divided up into nine processing units. Each unit is capable of operating in parallel, which allows for pipelined operation so that most instructions can be executed in one clock cycle.

Of interest is the Integer Unit which contains eight 32-bit general registers, the arithmetic and logic unit, and a 64-bit barrel shifter able to perform multiple bit shifts in one clock cycle. Also load, store, addition, subtraction, and logic instructions are executed in one clock cycle. It should be noted that the two 32 bit data buses are used simultaneously to transfer 64 bit operands in a single operation. All of these feature allow for fast implementation of integer computations.

The Floating-Point Unit is another key part of the CPU in order to achieve high mathematical performance. Figure 3.4 shows the architecture of the FPU built into the 486 processor. The FPU is usually called the coprocessor and as the name implies it works side by side with the processor. The processor fetches and decodes instructions and calculates addresses of memory operands while the coprocessor works in parallel executing special coded floating-point instructions. The coprocessor performs arithmetic and comparison operations on floating-point numbers and provides single instructions for many trigonometric

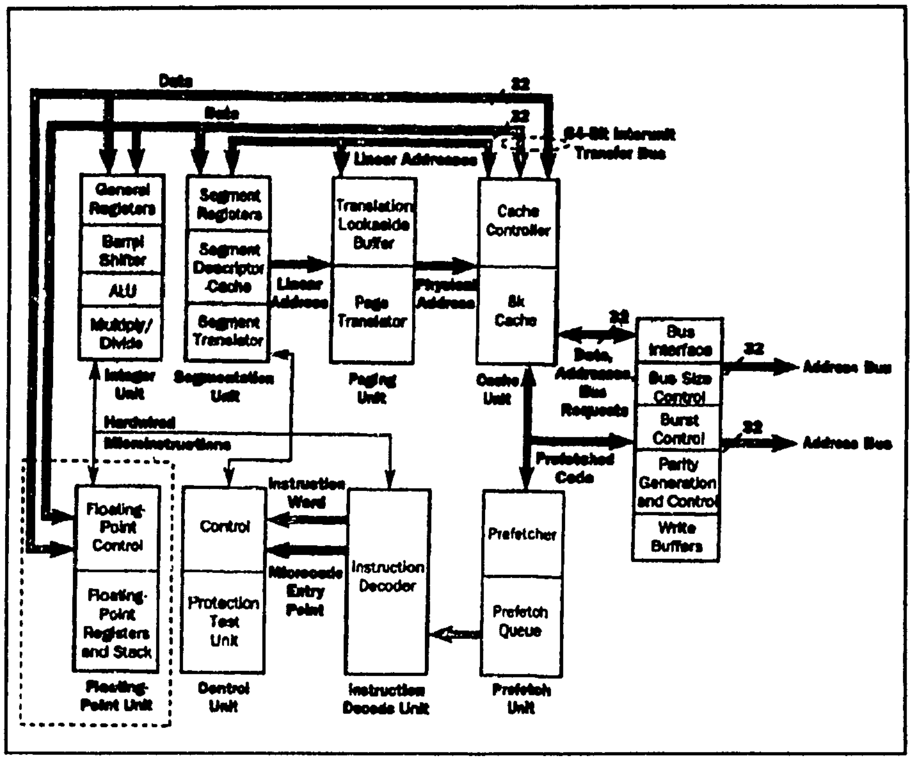


Figure 3.3 80486 Architecture [3.7]

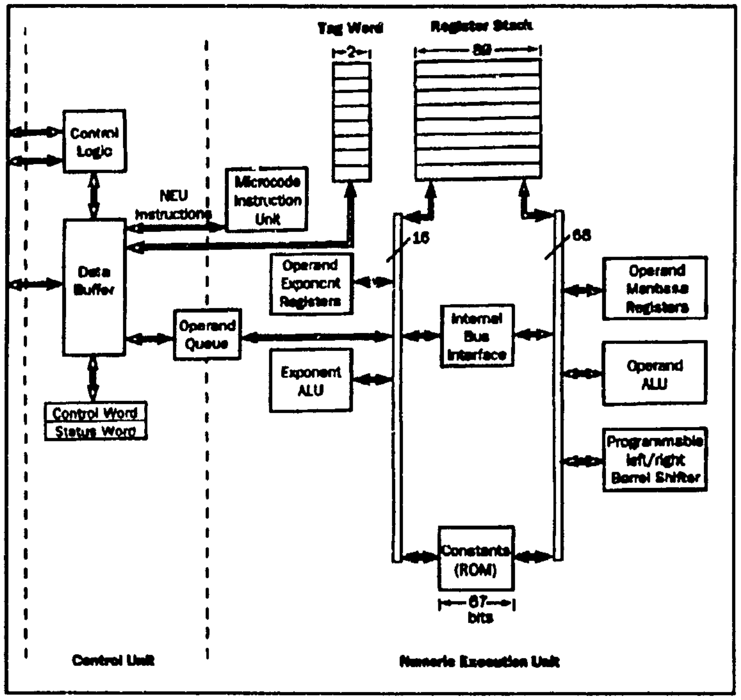


Figure 3.4 Math Coprocessor Architecture [3.7]

functions and transcendental functions. For example, along with the four basic mathematical operations (add/sub/mult/div), the coprocessor provides instructions to perform square root, scaling, remaindering, modulo, rounding, absolute value, tangent, logarithms, and exponentiation [3.7]. Some specialized hardware is used to perform specific functions For example, there are dedicated mantissa and exponent adders, and a programmable left/right barrel shifter, which significantly increase performance [3.7].

Table 3.2 shows some performance values for some mathematical operations. The test program used to obtain those values was written in C. The table also includes a column with the results for a 386SX processor. It should be pointed out that a SX processor has no mathematical coprocessing unit and therefore all floating point operations are executed in software emulating a coprocessor. The executing times are several orders of magnitude larger than for a 486DX. All operations were executed with floating point values.

3.2.3 Texas Instruments TMS320C40 Digital Signal Processor

One of the key features of this processor is parallel processing. Internally the chip is able to perform many different operations in a parallel fashion, which result in outstanding performance. The CPU is rated at 275 Million Operations Per Second (MOPS), 50 Million Floating Point Operations (MFLOP), and 320 Mbytes/sec of data throughput [3.3]. Other primary features of this processor are: six communication ports for high speed interprocessor communications, six channel Direct Memory Access (DMA) coprocessor for concurrent I/O and CPU operation, two identical external data and address buses supporting shared memory systems and high data rate along with single cycle transfers, on-chip program cache and dual-access/single-cycle RAM for increased memory access performance, and separate internal

Mathematical Operation (floating point)	486 66MHz time (μs)	386SX 40MHZ time (μs)
Add/Subtract	0.361	39.56
Multiplications	0.361	39.56
Divisions	0.361	39.64
SQRT	3.406	153.4
SIN	6.153	375.5
COS	6.923	383.5
TAN	8.626	626.9
ATAN2	16.208	-
LOG	14.505	564.1
EXP	18.681	632.9
COSH	20.714	693.4
ACOS	10.824	-

Table 3.2 80x86 Timing Summary

program , data, and DMA coprocessor buses for support of massive concurrent I/O of program and data throughput. The internal architecture of the C40 can be seen in Figure 3.5. The three main units are the Central Processing Unit, the communicᴦᵗion ports, and the DMA coprocessor [3.3].

The C40 has a register based CPU architecture. The CPU is made up of the following components: a floating-point/integer multiplier, an Arithmetic Logic Unit (ALU), a 32-bit barrel shifter, internal buses (CPU1/CPU2 and REG1/REG2), auxiliary register arithmetic units (ARAUs), and CPU register file. The multiplier performs 40 ns single-cycle multiplications on 32 bit integer and 40 bit floating-point values. The ALU executes single-cycle operations on 32 bit integer, 32 bit logical and 40-bit floating-point data. The ALU includes a barrel shifter capable of shifting left or right 32 bits in a single cycle. Using the internal buses, CPU1/CPU2 and REG1/REG2, two operands can be carried from memory and two more can be carried from the register file to execute parallel multiplies and adds/subtracts on four integer or floating-point operands in a single cycle. The auxiliary register arithmetic units are used to generate two addresses in a single cycle and since these units work in parallel with the ALU and the multiplier, increased performance can be achieved. The primary register file is made up of 32 registers. These registers can be operated on by the ALU, the multiplier, and they can be used as general purpose registers [3.3].

Six high-speed bidirectional communication ports are used for interprocessor communications. These ports work independently from the CPU, therefore increasing the total throughput. Each port is capable of supporting communication at 20 Mbytes/s since each port has 8 data bits and 4 control lines. Data which are always 32 bits wide are transferred over these ports as 4 sequential 8 bit bytes. Automatic arbitration provides

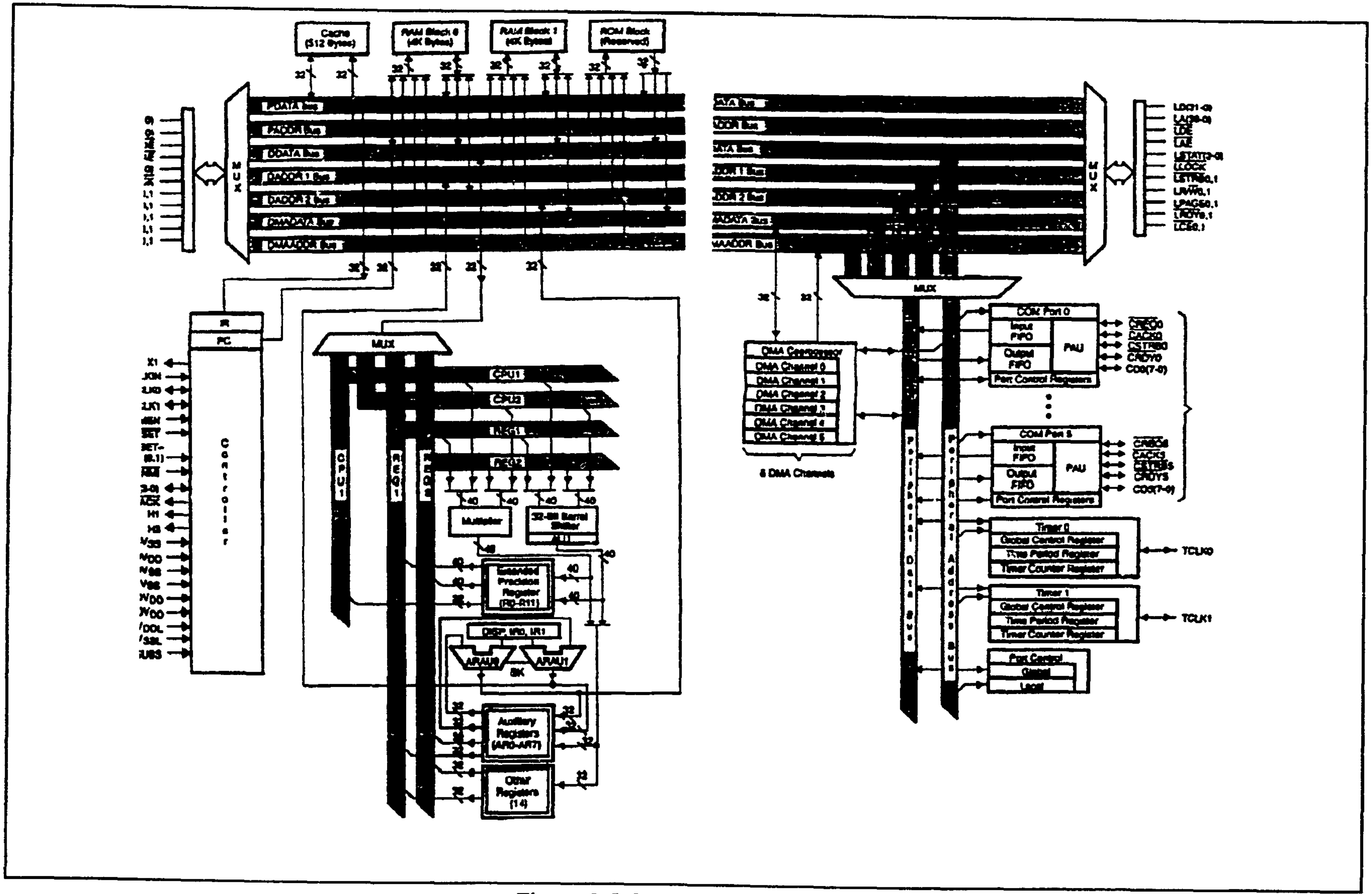


Figure 3.5 C40 Architecture [3.3]

reliable communication synchronization [3.3].

The DMA coprocessor can read or write to any memory location without interfering with the operation of the CPU. The DMA coprocessor has its own address generators, source and destination registers, and transfer counter. Since separate DMA address and data buses are provided, there is minimal conflict with the CPU. This is the basis for parallel processing since the CPU can pass on results of operations while still performing other tasks [3.3].

To summarize, there are seven 32 bit buses: separate program buses (PDATA, PADDR), data buses (DDATA, DADDR1, DADDR2), and DMA buses (DMADATA, DMAADDR). For example, the CPU can access two data values in one RAM block and perform an external program fetch in parallel with the DMA coprocessor loading another RAM block, all within a single cycle. Other features used to increase performance are the 4Kbyte RAM block built into the chip and the 128 word instruction cache. The on-chip RAM has sub-zero wait state access. The instruction cache can hold 128 instructions. Therefore the CPU can execute code without the need to use the external buses therefore freeing them for use for the DMA coprocessor. In peak performance the processor can execute up to 11 instructions per cycle [3.3].

Another interesting feature of the C40 is its repeat mode. If the program needs to execute an inner loop many times then the repeat mode is used so that all instructions of the inner loop are kept in the instruction cache and they are re-executed without any overhead. Performance is greatly improved for such cases. No external memory fetches are required since all the codes (up to 128 instructions) are stored within the CPU [3.3].

To test the mathematical performance of the C40 a test program was executed on one of the processors in the C40 network. The program was written and compiled in Parallel C. The results can be found in Table 3.3. The results obtained are interesting especially for the execution times for adds/subtract/multiply/divide which are all the same and in the picoseconds range. This could be due to the pipelining capabilities of the C40, the repeat mode feature, and the other performance enhancing features. It was found that by using a special math library called '*stdmthsr.lib*' better performance can be obtained for more complex mathematical functions. This library is a third party library that includes hand coded machine codes libraries to improve the software algorithms used to perform mathematical operations. Also, since floating point numbers and integers are manipulated in the same fashion, operations on both require the same amount of execution time. The test C40 module has 4 Mbytes of local DRAM, 1 Mbyte of global DRAM and the 4 Kbytes of on-chip RAM and it is hosted in a 486 based system. It is important to note however that the host system has no interaction with the C40 system, unless it is for user interfacing, i.e. data display and data input.

3.3 Performance Comparison

A summary of the benchmarks and performances for the processors tested can be found in Table 3.4 The TMS320C40 processor outperformed all others and in many cases with large margins. For the more complex mathematical functions the differences between the 486 66MHz and the C40 are not as large, but the C40 still has the edge over the 486. **The most important point to make is that the C40 and the T800 have the possibility of parallel processing which could improve performance more or less proportionally to the number of processors used.** A 486 cannot easily implement a parallel architecture and

Mathematical Operation (floating point)	math.lib time (µs)	stdmthsr.lib time (µs)
Add/Subtracts	5.023 ps	-
Multiplications	5.023 ps	-
Divisions	5.023 ps	-
SQRT	1.807	1.262
SIN	10.603	2.244
COS	11.024	2.191
TAN	11.989	2.480
ATAN2	15.530	3.953
LOG	12.538	2.607
EXP	13.088	2.237
COSH	14.844	2.992
ACOS	13.528	2.683

Table 3.3 TMS320C40 Timing Summary

Processors Comparative Chart

Mathematical Operation	TMS320C40 - 40 MHz 50 MFLOP * 275 MPS * -		Transputer T800-20 1.5 MFLOP * 10 MPS * 4000k Whetstones		486 66MHz 2.605 MFLOP - 11270 Whetstones	486 50 MHz 2.279 MFLOP - 8309 Whetstones	386SX 40MHz 0.016 MFLOP - 131 4 Whetstones	format
	math.lib	stdmthsr.lib	no CFG	with CFG				
Add/Subtractions	5.023 ps 5.023 ps	- -	1.1 us 0.2 us	1.24 us 0.91 us	0.361 us	-	39.56 us	float integer
Multiplications	5.023 ps 5.023 ps	- -	1.7 us 0.8 us	2.24 us 1.44 us	0.361 us	-	39.56 us	float integer
Divisions	5.023 ps 5.023 ps	- -	2.4 us 2.2 us	2.82 us 2.91 us	0.361 us	-	39.84 us	float integer
Sqrt function	1.807 us	1.262 us	18.6 us	22.44 us	3.406 us	-	153.4 us	float
Sine function	10.603 us	2.244 us	51.2 us	75.64 us	6.153 us	8.186 us	375.5 us	float
Cosine function	11.024 us	2.191 us	43.1 us	61.84 us	6.923 us	-	383.5 us	float
Tangent function	11.989 us	2.480 us	-	-	8.626 us	-	626.9 us	float
ATAN2 function	15.530 us	3.953 us	60.9 us	92.74 us	16.208 us	-	-	float
Log function	12.538 us	2.607 us	-	-	14.505 us	-	564.1 us	float
Exp function	13.088 us	2.237 us	-	-	18.681 us	-	632.9 us	float
Cosh function	14.844 us	2.992 us	-	-	20.714 us	-	693.4 us	float
Acos function	13.528 us	2.683 us	-	-	10.824 us	-	-	float

* Rated Value

Table 3.4 Processors' Performance Comparative Summary

therefore if more than one 486 processors are used together, the final performance degrades as more processors are networked together due to bus contention and memory accessing problems. On the other hand when several C40s or T800s are networked it is up to the programmer to take advantage of the parallel architecture of these processors and write code so that each processor can work more or less in parallel with the others to increase the overall computing power.

A puzzling result comes from a test executed on the C40. In order to calculate the time required to compute a mathematical function, a loop is used to make the processor compute the same function many times so that final execution time is divided by the number of times the function was executed. It turns out that in 5.023 seconds the C40 is capable of performing one trillion or 10^{12} additions or subtractions or multiplications or divisions. In other words the execution time of an addition is 5.023 picoseconds. This is nowhere near the 0.361 microseconds it takes a 486 to execute the same operation.

Figure 3.6 gives a comparative chart for the execution times needed to compute various arithmetic operations. As it can be seen, there is a large difference between the fastest (C40) and the slowest (386SX). The C40 is about 7.8 million times faster than a 386SX. Whereas in comparison with a 486 the C40 is about 71,000 times faster. The T800 is about 246,000 times slower than a C40. Also, as shown the T800 performs better without a configuration and on integer values rather than floating point values. Also the T800 is the only processor that has different execution times for add/subtractions, multiplications, and divisions, whereas the C40 and the 486 perform these operations in the same amount of time

The next comparative chart (Figure 3.7) shows the execution times required by each processor to carry out trigonometric functions. The C40 with the high performance

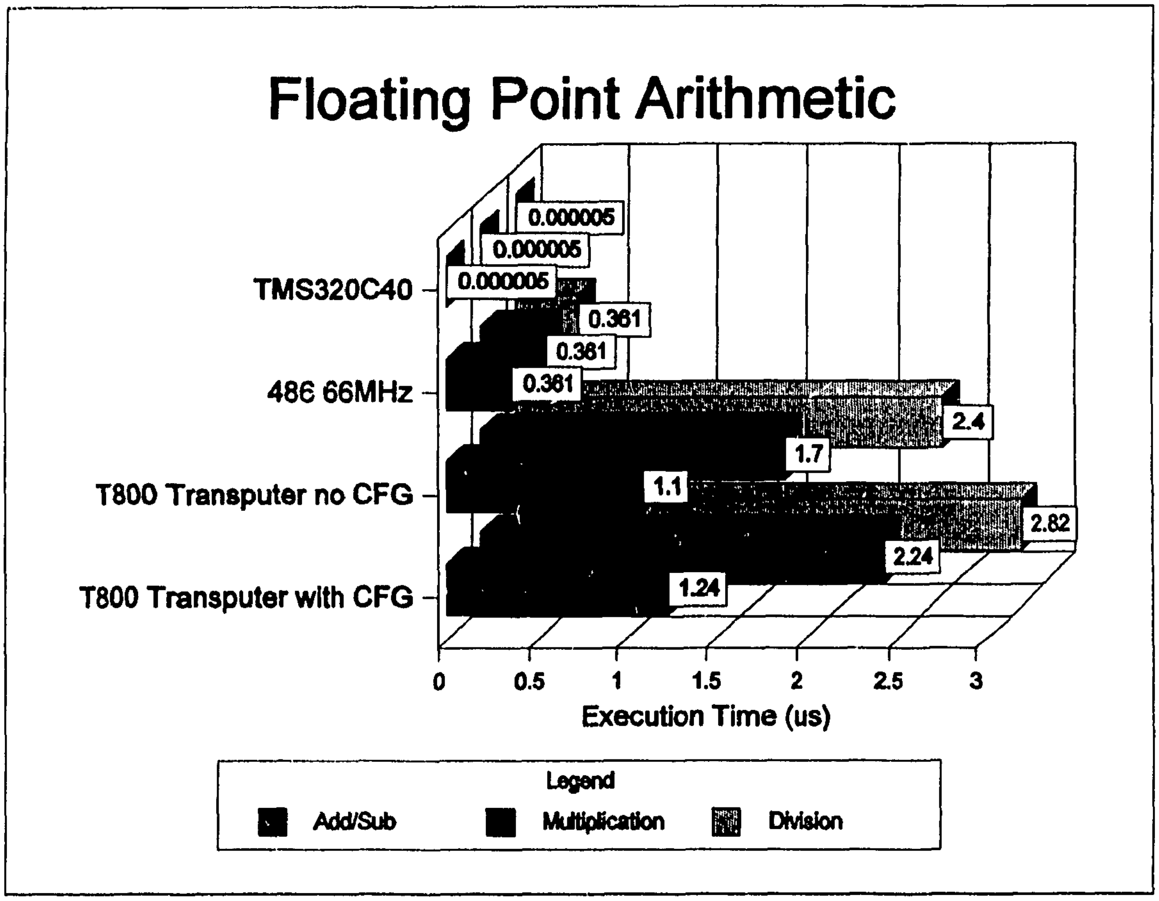


Figure 3.6 Floating Point Arithmetic Comparative Chart

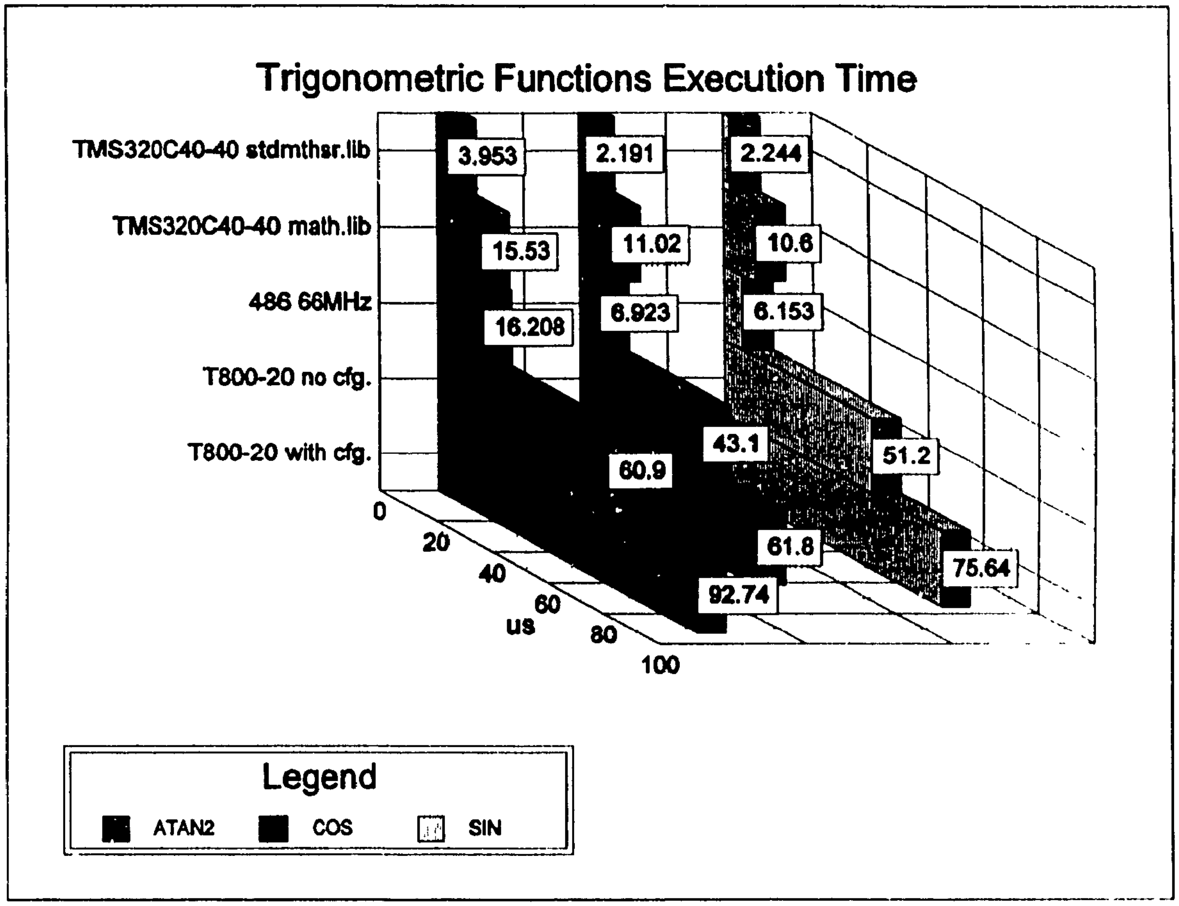


Figure 3.7 Trigonometric Functions Comparative Chart

mathematical libraries has the lowest execution times for trigonometric functions. The 486 is second, being about 3 times slower than the C40. The worst case is for the T800, its execution times being about 30 times slower than the C40 and 10 times worst than a 486. The T800 clocks at 20 Mhz, whereas the 486 at 66 Mhz, which is 3 times faster. As a note, the 486 performs better than the C40 when the C40's program is compiled using the standard libraries of Parallel C. When the high performance libraries are used then the opposite is true. This is explained by the fact that the 486 has 'hardware' opcodes to perform sine and cosine functions, whereas the C40 has to implement them using software because there are no special opcodes The results for logarithmic functions can be seen in Figure 3.8. Again the high performance libraries for the C40 help to speed up the execution times. In comparison a 486 executes the same functions about 7 times slower than the C40 with the high speed mathematical libraries. Even with the standard libraries the C40 is faster by a couple of seconds.

3.4 Summary

When selecting a processor, MIPS and MFLOPS are not the only means to judge the performance of a processor. In many cases, the overall system performance is based on how the main processor is able to interact with its environment and other processors [3.2].

The programming tools are also very important when selecting a processor [3.2]. In this case the C40 can be programmed in Parallel C which is very similar to C programming. C40 Parallel C is a compiler that embeds the Texas Instruments compiler and libraries plus it adds other useful features. Another reason for the selection of the combination of C40 and Parallel C is the fact that the C40 network can be used in conjunction with a transputer T800

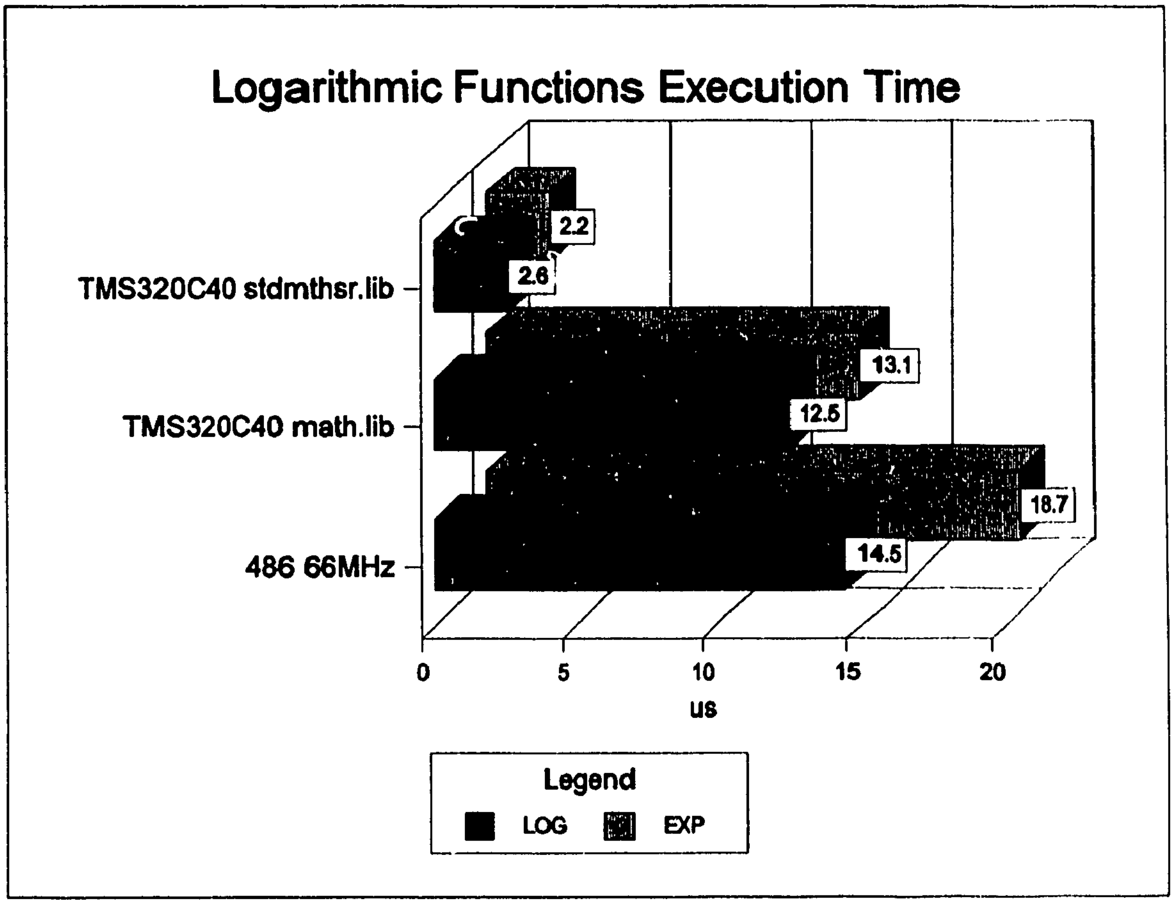


Figure 3.8 Logarithmic Functions Timing Summary

network, as the research group at the Centre for Industrial Control has been using extensively T800 transputers for real-time control. Parallel C is able to compile code for both of these networks and route all the programs to the right processors in both the C40 and the transputer network [3.5].

Furthermore, processing modules, image grabbers, and display modules have been built with C40 processors. Since all of these modules will have to be used conjointly to perform image processing, compatibility was another issue. Since all modules are built around the C40, data and programs can be exchanged easily between modules. This facilitates and speeds up the programming and execution of the code necessary to accomplish the task. Specialized libraries for image capture, processing, and display are also available, which should further help in the programming stages. A special mathematical library is also available that speeds up some mathematical functions by a factor of five, as compared to the math library provided with Parallel C. Therefore it can be concluded that there is good support for the C40 processor, both in terms of compilers and development tools.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF OPERATING SYSTEM FOR ATV

4.1 Introduction

This chapter describes how the operating system for the ATV was designed and implemented. The operating system is necessary to combine all sub-systems of the vehicle such as vision, motor control, user interface, etc. so that they all function and interact in harmony. Two distinct but interdependent parts of the operating system can be identified: the hardware implementation and the software implementation. The software is designed given the hardware but the hardware can be modified to improve the performance of the software and as a result the overall system's performance is enhanced [4.1]. The hardware is used to interface electronically all separate components: CCD camera, video displays, motion controllers, various processors, user interfaces, etc. The software is necessary to make sense and order to all of the information coming and going to these systems. The following sections explain in more detail the intricacies of the operating system.

In this chapter the hardware implementation is described first in section 4.2. Then the various hardware sub-systems are discussed, including the user interface, the motor interface, and the camera vision guidance system. Section 4.3 and several sub-sections concentrate on the software implementation of the operating system. The software is made up of several programs executing simultaneously on different processors Parallel processing is explained in section 4.4. Some advantages of parallel processing and some of its complexities are discussed as well.

4.2 Hardware Implementation

Figure 4.1 shows the various hardware components involved in the development of CONCIC-4 ATV. The hardware can be divided into three main sub-systems: the user interface, the motor interface, and the camera vision guidance control unit. The first two sub-systems are governed by an Intel 486 66MHZ processor and motherboard through an ISA 16 bit bus. The camera vision guidance control unit is based on a network of 32 bit Texas Instruments TMS320C40 Digital Signal Processors (DSPs). The DSPs and the 486 processor communicate through a shared 16k memory buffer in the PC's memory. Each sub-system will now be explained.

4.2.1 User Interface Sub-system

This system is based on the PC. The PC reads the keyboard, writes and reads from the hard disk, and it displays menus on the text display. This system enables the user to have complete control over the functioning of the vehicle.

It should be noted that most of the information being displayed and entered by the user is not processed by the PC. The PC acts as a host for the C40 DSP network. All information entered by the user is relayed to the DSPs for processing. Also the DSPs send all of the data to be displayed to the PC. All accesses to the hard disk executed by the PC are on the behalf of the DSP network.

4.2.2 Motor Interface

There are several stages in the motor interfacing. The low level motor control is executed by several motion controllers. The National Semiconductors LM628 motion control

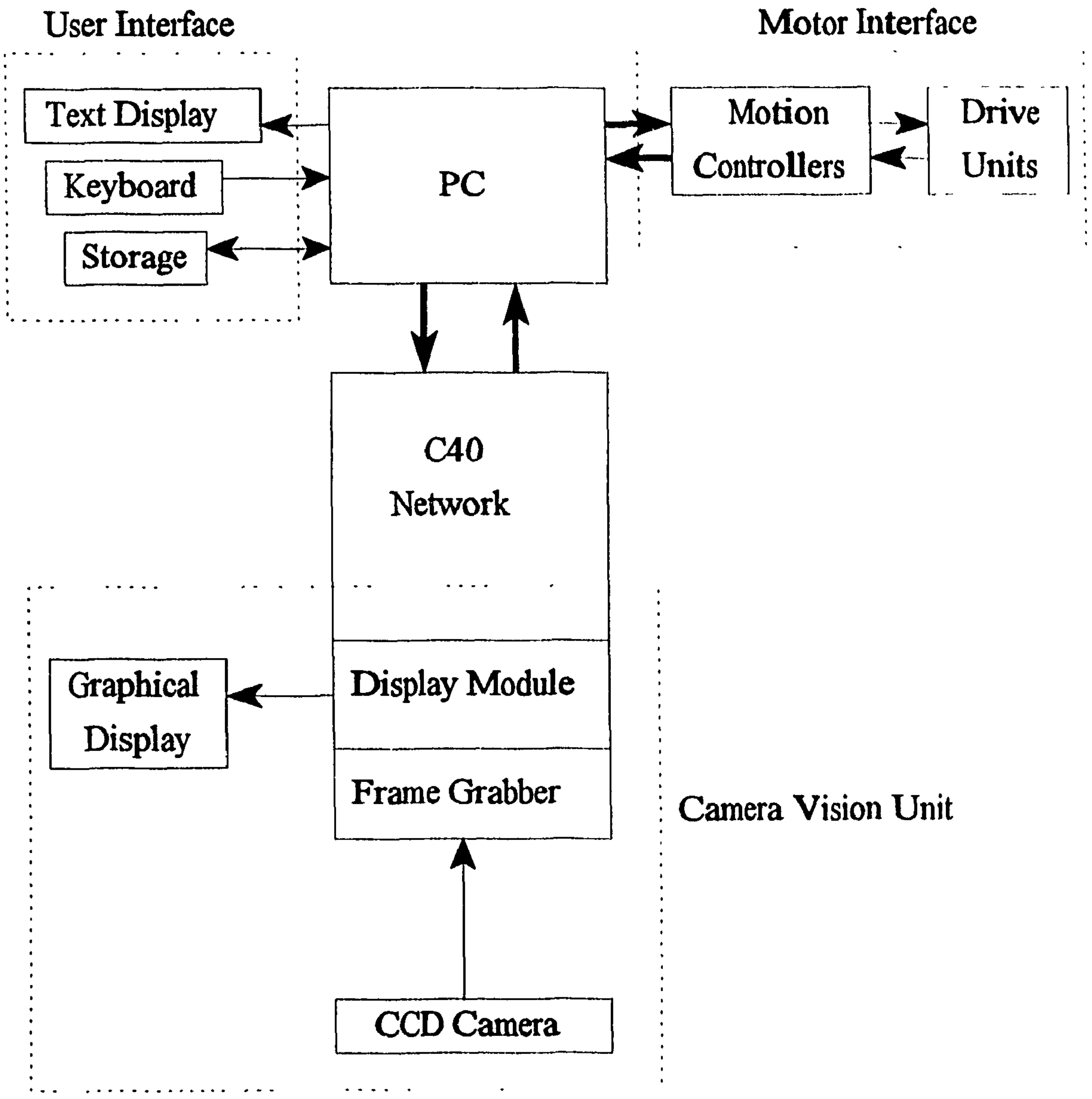


Figure 4.1 Schematic of CONCIC-4 Hardware La[illegible] [illegible]ram

processors are used in this case. These dedicated processors can operate in velocity or position mode. They monitor the motor's response through pulses received from an encoder and take action accordingly. These motion controllers are interfaced to the host computer through a set of high level commands.

The motion controllers are interfaced through an other circuitry to the PC ISA bus (Figure 4.2 shows in detail the electronics used to interface the LM628s and the PC bus). It is important to note that the PC acts as a host again because it relays data to the LM628s that was generated by the DSP network. Also the PC returns data read from the LM628s to the DSP network. The PC does not do any computation or data manipulation, it relays data to the right location or device. The exact time it takes the PC to relay the data to the LM628s from the C40 network is not know, however the delay is minimal because it is not noticed when the system is in operation.

4.2.3 Camera Vision Guidance Control Unit

This system is mostly based on the C40 DSPs. The images from the CCD camera are received and digitized by a C40 based frame grabber. These digital images are processed by two other C40 DSPs and sent to another C40 based display module where they are displayed. A fifth C40 DSP receives the information extracted from the images and computes the guidance control algorithm. This same DSP also interfaces with the PC using a shared memory location in the PC's memory. This shared buffer is 16kbytes long and it can be considered as a bidirectional queue.

The five C40 DSPs are linked to one another through the comports internal to each processor. There are some hardware restrictions on how the DSPs can be interconnected.

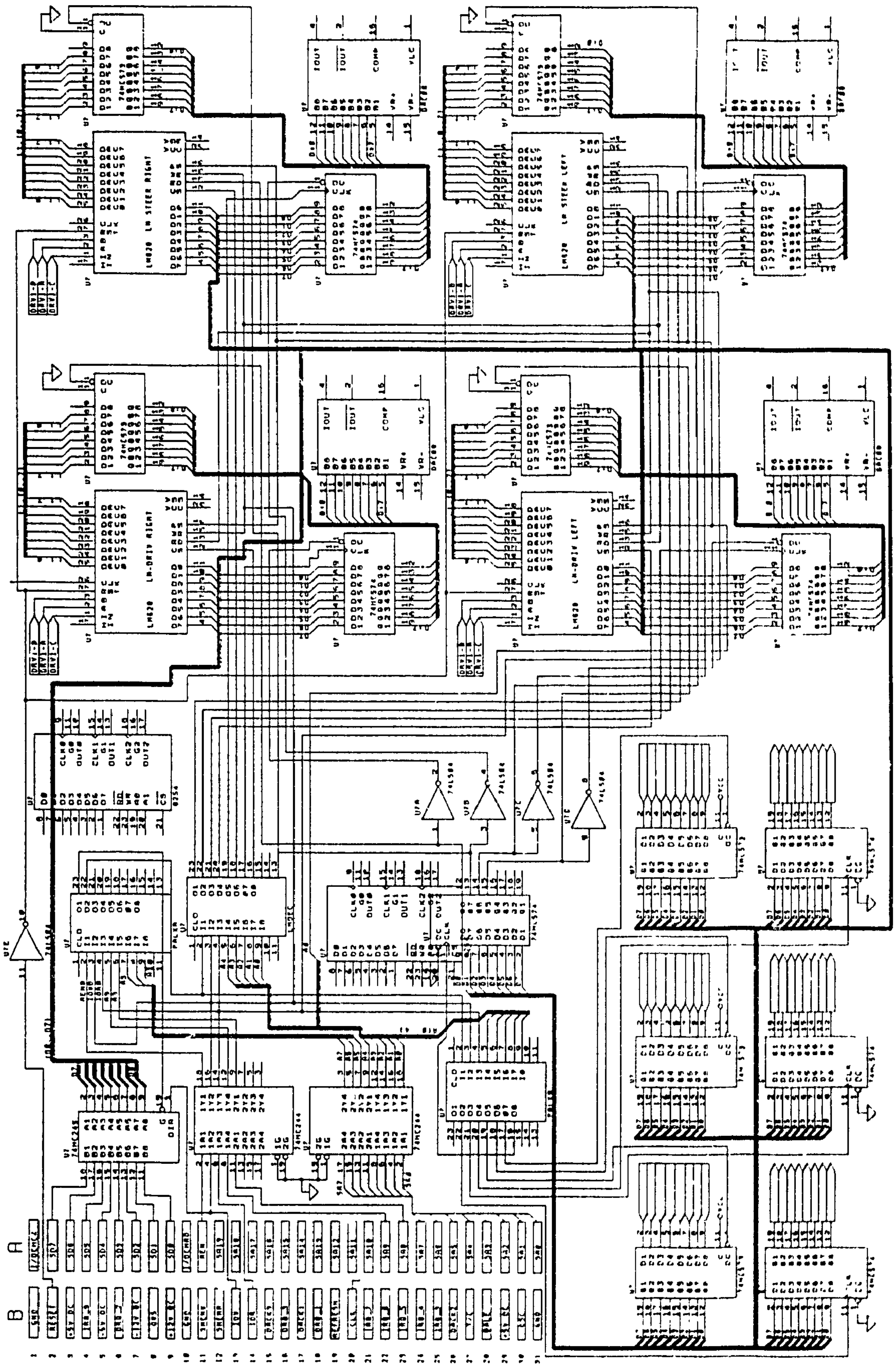
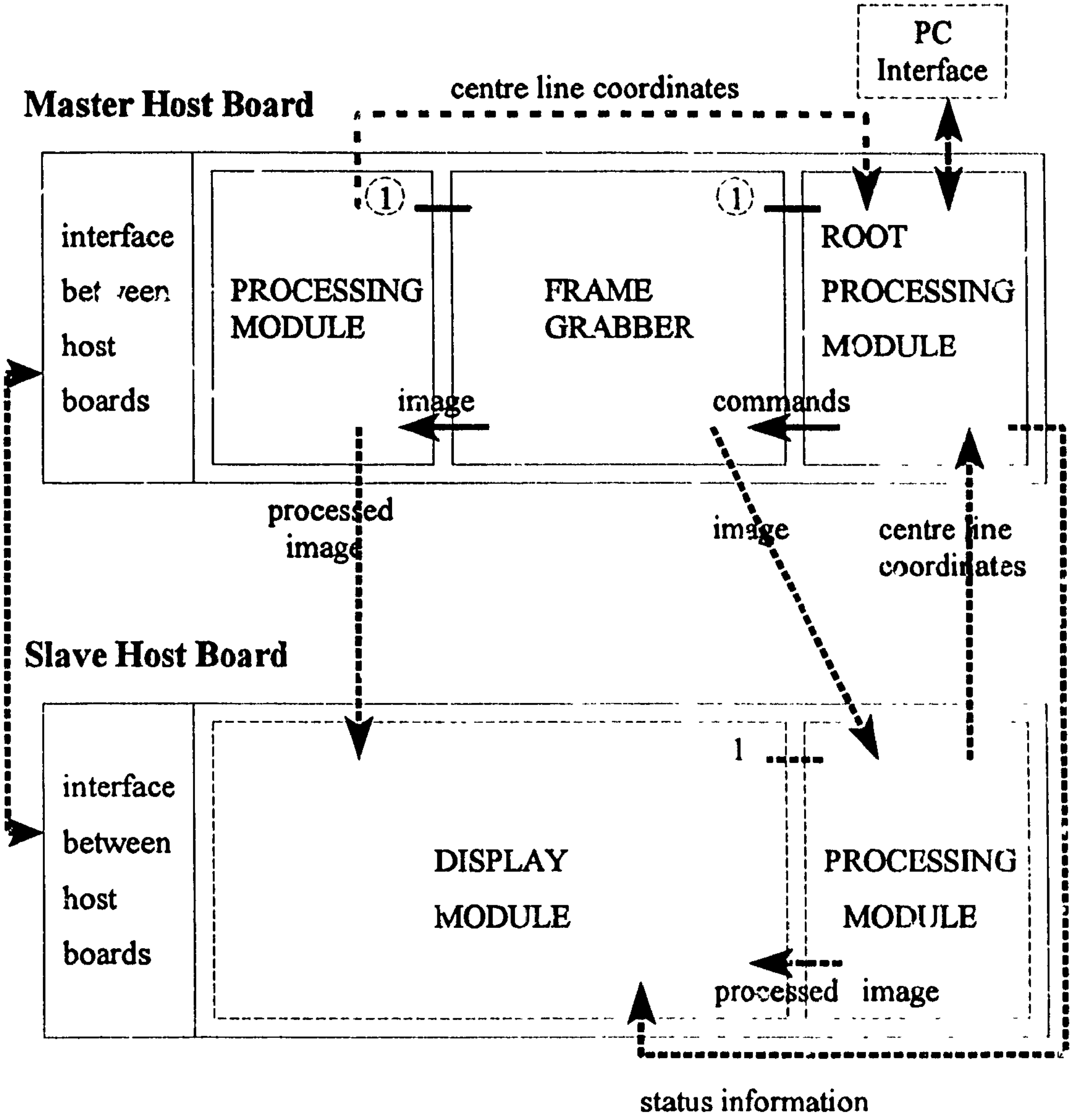


Figure 4.2 PC to LM628 Motion Controllers Interface Circuitry

Due to the design of the host boards, each DSP module is physicall,/ connecte^d with two comports to the modules adjacent to it. Therefore of the six comports available, only two are free for interconnecting DSP modules that are not adjacent to one another.

The C40 network can be described as pictured in Figure 4.3. There are two main host boards that plug in the PC's ISA bus. Each one of these host boards has room for four C40 DSP modules. It should be noted that the frame grabber module occupies two slots and the display module takes up three slots. Another important point that should be highlighted is the fact that only one of the host boards can interface with the PC. Therefore, one host board is the master and the other one becomes the slave. Also only the C40 module plugged in slot one of the master board can interface with the PC. All other modules send messages to this processor in slot one and it will relay the data to the PC. For reading PC inputs the same applies.

The reason why the network is built this way is due in part to the software requirements. The physical location of the various modules was however dictated by the allowable interconnections between modules. A C40 DSP running at 40 MHz is the 'root' processor. it communicates with the PC and it organizes the activities of the other DSPs. The frame grabber module needs to receive from the 'root' processor the correct timing for digitizing new images. Also, it sends out digital images to two processing modules, one located in the slot next to it and another located on the slave host board. These modules are C40 DSPs running at 50 MHz and they send the centre line information back to the 'root' DSP for guidance control purposes. Since these processing modules execute the bulk of the image processing, they had to be as fast as possible and that is why they were chosen to be 50 Mhz modules with enhanced DRAM memory. Finally, the display module receives the processed



(1) physical link not being used

Figure 4.3 Physical Topology of C40 Modules

images from the processing module next to it and from the other located on the master host board.

These interconnections between processors are called comports. The data transfer over these links is up to 20 MBytes/second. The link is eight bits wide and therefore four bytes are sent to complete a 32 bit word. The C40s are 32 bit processors and the smallest addressable memory is 32 bits wide. Also, since each processor has six comports built in, each comport can communicate independently and simultaneously [4.2].

4.3 Software Implementation

The overall operating system software can be divided into six functional blocks: MAIN, CAPTURE, PROCESS1, PROCESS2, DISPLAY, PARAMETERS. These blocks interact with one another as shown in Figure 4.4. Each one of these blocks represents one or more programs running in a separate processor. All blocks with the exception of PARAMETERS execute in a separate C40 DSP. The code is developed and compiled in Parallel C for the C40 network and in Borland Turbo C for the code running in the PC.

4.3.1 Description of PARAMETERS Software Block

PARAMETERS contains three programs that are called by the C40 network, but that are executed by the PC. These programs enable the user to modify or view operating parameters and data acquisition options. Both operating parameters and data acquisition options are saved on the PC's hard disk as text files. This is because text files can be read by both the PC and the C40 network (always through the PC). These three programs could have been written to execute from the C40s, however in order to save memory on the C40 module,

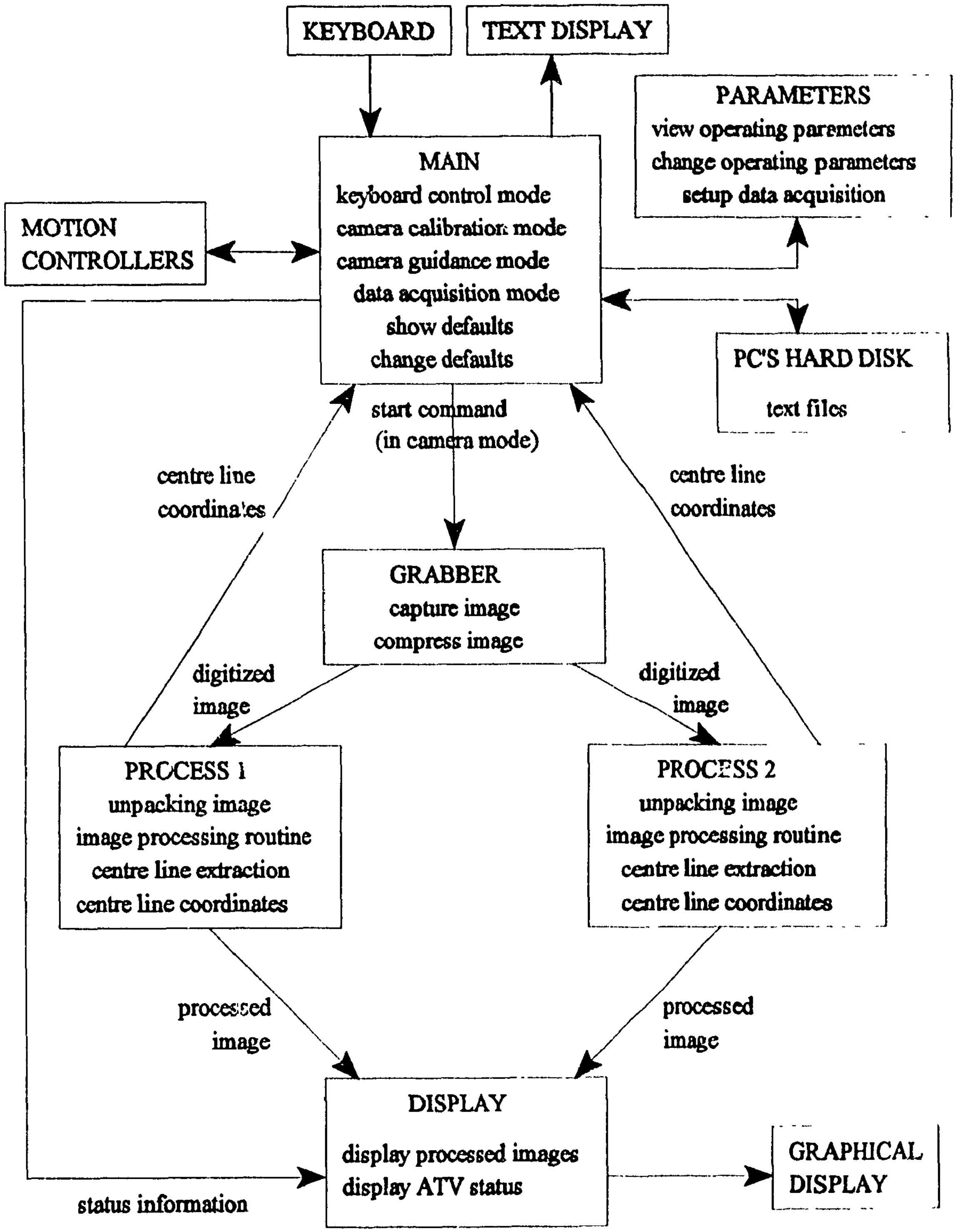


Figure 4.4 ATV's Operating System Software Layout

they are called by the C40 and executed on the PC. In fact ASCII characters which are 8 bits wide are stored in the C40 memory as 32 bit entities, therefore wasting 24 bits. Since these three programs display mostly text information, they can be running in the PC environment. As these programs are executed off-line, high speed computing is not required. The C40 processors during the setup time of the system parameters are idle waiting for the user to start the system. Once the data is saved in a text file, the C40 modules can retrieve it and use it.

4.3.2 Description of MAIN Software Block

MAIN, as the word says, is the overall control block. All decisions are made there, the user can scroll through the menus and the various modes of operation of the ATV can be selected. This program provides timing to the other blocks. It is executed by the C40 module that interfaces with the PC because all of the user information has to flow back and forth from the C40s and the PC.

This program gives the user a choice on how to operate the ATV. Keyboard mode is used to move and place the vehicle using the keyboard arrow keys. In camera calibration mode the image processing routines are executed but no control action is given to the motors. In this mode, the quality of the camera image can be evaluated and improved if necessary. The camera guidance mode allows for self-guidance using the information extracted from the CCD camera images. A data acquisition mode is also provided so that the user can record vehicle performance for analysis. Vehicle operating parameters can be viewed and modified as well.

The keyboard control mode monitors the keyboard and then sends the appropriate commands to either steering or driving motors. The vision routines are not started in this mode. Data acquisition can be started and stopped at any moment using the F6 and F7 keys respectively.

In camera guidance mode the MAIN module sends a start command to the GRABBER module and images begin to be processed. The MAIN module then receives the centre line coordinates from the PROCESS1 and PROCESS2 modules. These coordinates are used as the feedback element in the control loop. A new set of steering and commands are then given to the motion controllers. The keyboard is also being mo in case the user would like to increase or decrease the travelling speed (using the up and arrows respectively) or stop the vehicle using the 'enter' key. Data acquisition can be started and stopped at any time using the F6 and F7 keys. The main control loop executes 15 times per second or every 66 ms for real-time control.

4.3.3 Description of GRABBER Software Block

This program initializes the frame grabber hardware and then waits for a start signal from the MAIN block. Once the signal is received, an image is captured. The image has a size of 512x512 pixels and it is monochrome with a resolution of 256 shades of grey or 8 bits. The pixels are stored in a compressed format so that four pixels are placed in a 32 bit memory location. Also, the eight MSB contain the right most pixel and the eight LSB bits contain the left most pixel, therefore some bit manipulation will have to be used later.

Once an image is captured, it is sent to PROCESS1 and as soon as the next image is captured it is sent to PROCESS2 and the 3rd image goes to PROCESS1, etc. The GRABBER module is stopped by the MAIN module when the user exits the camera mode or the camera calibration mode

4.3.4 Description of PROCESS1 and PROCESS2 Software Blocks

These two b'ocks are identical and are executed simultaneously. One full 512x512 pixel image with 256 grey shades is received by the block. First, the image is decompressed and reduced to a 256x256 image. Then using a threshold value the image is binarized so that each pixel is either 'on' or 'off'. In the case of a black guiding line, the line is an 'off' pixel or a black pixel. Then two noise reduction algorithms are implemented to eliminate background noise.

A noise free image is sent to the centre line extraction algorithm. This algorithm has also the ability to remove any minor background noise left over and it scans the image horizontally for the beginning and ending pixel positions of the line. Once these positions are located, an average is taken to determine the coordinates of the centre of the line. The number of image lines scanned for determining the centre line coo.dinates can be changed by the user. Once the centre line coordinates are identified, these are returned to the MAIN block for guidance control purposes.

4.3.5 Description of DISPLAY Software Block

DISPLAY enables the user to view the processed images and some important information about the status of the vehicle This program initializes the display hardware and then receives the processed images in sequence from PROCESS1 and PROCESS2. Also, the vehicle status information is received from MAIN and displayed. Since this display board had no hardware fonts and Parallel C does not provide any graphics character libraries, they have been generated in software. A bit mapped alphabet, along with numbers and other symbols has been designed.

4.4 Parallel Processing

Parallel processing involves the execution of several tasks at the same time The ATV takes full advantage of parallel processing, and in fact parallel processing was the only solution to achieve real-time control and higher operating speeds. Most of the computing power is required to perform image processing and centre line extraction. Two C40 DSPs are in charge of those tasks. In the mean time, the frame grabber digitizes a new image, the display module displays the previous processed image, and the main guidance control loop is executed in another C40 DSP. This system allows for a control loop cycle time of 66 ms

4.4.1 Advantages of Parallel Processing

The most notable advantage of parallel processing is the ability of executing many tasks simultaneously and accomplishing more work in less time. In terms of image processing, where large sets of data need to be analyzed and manipulated, parallel processing becomes a necessity. The strategy used to save time involves starting processing of the following image by another C40 DSP meanwhile the previous one is still being treated by a different C40 processor. With this strategy it is possible to achieve a frame rate of 15 frames per second even though it takes over 300 ms for a full frame to be processed

With parallel processing, a single processor can be dedicated to each specific task This allows for tasks to be more independent of one another, as opposed to a time sharing system or a purely sequential system. Tasks are still allowed to communicate with one another by passing tokens or sharing data. Also parallel processing reflects more the parallel and multitasking nature of most natural processes For example a person takes full advantage of his/her abilities to look at the road, make decisions, adjust the speed suitably, turn the steering wheel, possibly speak to other people or listen to the radio.

Parallel processing gives the flexibility of designing specific hardware, for example, the frame grabber all it has to do is to digitize images. The display module receives already processed images and displays them.

In terms of programming, each task can be coded separately, which allows for shorter programs. Also debugging can be started by executing small sections of code in each processor. There are less conflicts of memory since each DSP has its own memory space. However there are also many disadvantages and complexities that stem from parallel processing.

4.4.2 Complexities and Restrictions of Parallel Processing

With parallel processing many tasks can be executed at once. However, these tasks in many cases are somehow linked together or they need to be executed in a specific sequence. Therefore much more attention has to be dedicated to time all applications correctly so that some operations wait until a specific task has finished and the results passed on. Debugging the whole system could become very complicated because now any task could fail and crash the whole system. In many instances the software bugs are not easily identifiable. Also since all communication between processors is done through the comports, some restrictions are imposed by the Parallel C environment. Comports can only transmit data in sequence. What this means is, that even though the C40 DSP is capable of simultaneously transmit data over several comports, the Parallel C language limits the programmer to sequential comport use. Therefore, one comport transmission has to terminate before the next one can be initiated. Also, the C40 network has no shared memory among C40 DSP and this too is a great limitation because the same data cannot be accessed by several processors, on the other hand it has to be passed to every processor that will use it. Passing small packets of data over the comports can waste time, therefore all data should be combined as much as possible and sent in one shot.

Synchronizing all tasks is the most difficult and important issue with parallel processing and great care should be taken in order to have a successfully working program.

4.5 Summary

This chapter presents some of the issues involved with the design and the implementation of the operating system for CONCIC-4 ATV. The hardware implementation of the three main hardware sub-systems is described in some detail. Furthermore, the software designed to run the ATV is explained. For a user guide on how to use the operating software please refer to Appendix A. Various programs execute simultaneously in different processors. Results are passed back and forth among the processors at the appropriate times Some reasons why parallel processing is used to speed up the system are also given along with the advantages and disadvantages of parallel processing. Parallel processing introduces complexities when programming the software. Some restrictions on parallel processing are discussed as well.

CHAPTER 5

IMAGE PROCESSING ALGORITHMS

5.1 Introduction

Since the AGV's guidance system relies upon camera vision, it is important that the images obtained from the CCD camera are processed in such a way as to filter out unwanted information or background noise. The AGV follows a line on the ground. To obtain better image processing results this line should have a definite contrast with the ground colour. A black line could be used on a white floor or a white line could be followed on black pavement. Tests have been conducted where a black line is used on a grey cement floor and the guidance system has still worked perfectly. Image processing plays a key role in allowing the recognition of the line even in less than ideal conditions.

Images captured from the CCD camera contain many bits of information which are not all important or useful in guiding the AGV. Therefore, the role of image processing is to enhance the pertinent information, while minimizing the dominance of all others. In other words, two important concepts in image processing need to be applied: contrast enhancement and noise filtering [5.1]. Contrast enhancement and noise filtering are necessary because not all images are perfect. The lighting conditions may not be ideal all the time, the contrast of the line with the road pavement might be poor, or the pavement is not of a uniform colour.

This chapter describes in section 5.2 a couple of image processing routines that either enhance or filter an image. A binarizing routine is discussed, edge detecting and the Sobel operator is introduced, some filtering techniques are explained like the median filter and the local average filter. Test results of the above mentioned routines are found in section 5.3.

Section 5.4 describes which routines are utilized by CONCIC-4 along with an explanation of the implementation of such routines.

5.2 Description of Image Processing Routines

In principle, contrast enhancement and noise filtering are the two image processing concepts which have to be applied in order to obtain valid images to provide accurate AGV guidance. However, there are many methods applicable to achieve contrast enhancement and noise filtering. Every technique has its advantages and disadvantages. In this context the disadvantages would be the complexity of the algorithm and hence the time consuming nature of the algorithm. For real time control, time is the essence and milliseconds are precious. The advantages of a technique would include its effectiveness in obtaining the required results and its low computational requirements.

An optimized set of image processing libraries are available for the C40 DSP from Sinectonalysis [5.2]. These libraries are used to carry out the image processing and image enhancement in this work. Over 300 algorithms, covering a wide spectrum of image processing techniques, have been studied and many tested with various results. Eventually four types of algorithms are regarded as possible candidates for the current application: binarizing and edge detecting algorithms, median filters, and local average filters. A description of each technique can be found in the following sections.

Colour processing is not carried out because it would have been necessary to process three times more data, since a pixel in colour is represented by 24 bits as opposed to a grey scale pixel which is only 8 bits. Also the road is usually made up of dark pavement and a white line, and therefore there is no need to represent these entities in colour when they are already in black and white.

5.2.1 Binarizing Algorithm

The images received from a CCD camera should contain two distinct entities: a black component, the guidance line, and a white or off white background. Therefore it is logical to try and create the largest possible contrast between these two items. A binarization algorithm would indeed create the largest possible contrast because a pixel is either black or white. Therefore, using some threshold value pixels are set to black if the original pixel value is below the threshold and if the original pixel value if above the threshold the pixel is set to white [5.2]. In mathematical form the following is applied:

$$B_{ij} = \{\ 0 \ \text{if} \ X_{ij} > \alpha \ ; \ 255 \ \text{if} \ X_{ij} \leq \alpha \ \} \qquad (5.1)$$

where B_{ij} is the pixel located at i^{th} row and j^{th} column, X_{ij} is the original pixel value, and α is the threshold value. Figures 5.1 and 5.2 show the original grey scale image from the AGV's CCD camera and the result of the binarization algorithm respectively. It can be seen that the line is clearly defined and most of the background has been eliminated. There is still noise in the picture that manifests itself as small blotches at random locations. These spots still need to be removed from the image and therefore binarization alone is not enough. It should be mentioned that this image was taken at a test site with good lighting but with a floor that is made up of white tiles with random black blotches. Small black lines can also be identified in between the tiles. There may have been some dirt on the floor as well.

In the binarization algorithm each individual pixel is processed once. This is also called a point process algorithm [5.3] because it modifies a pixel's value solely based upon a relation of the pixel's original value. No other pixel values are involved in the transformation. Also, the threshold value can be changed to allow for conditions of low lighting or bright

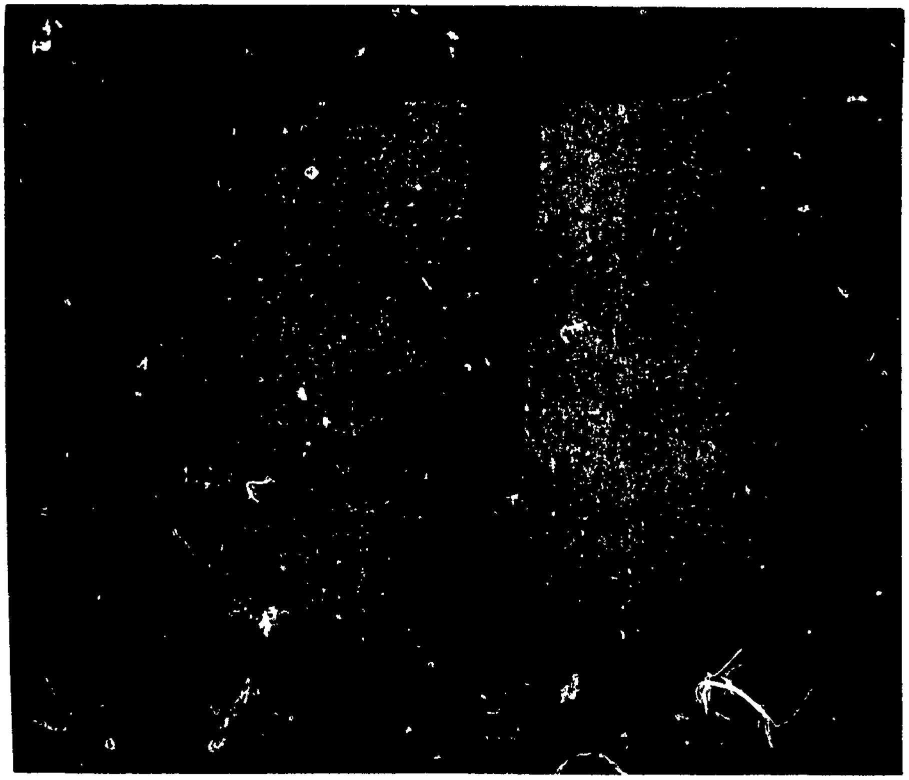


Figure 5.1 Original Image

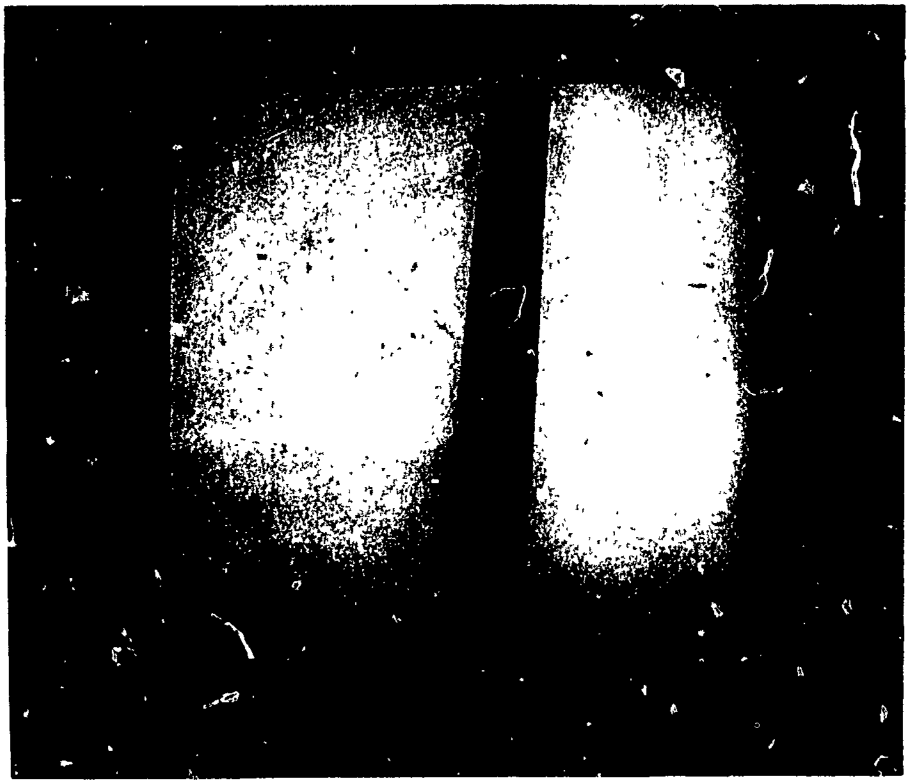


Figure 5.2 Binarized Image

light. In most cases the threshold is fixed at an average value and the aperture of the camera is opened or closed to adjust for lighting conditions. It is concluded that this algorithm by itself is not enough to obtain a clean image from which to extract the correct centre line coordinates. Therefore, this algorithm in combination with others will have to be used

5.2.2 Edge Detection and Sobel Operator

In order to extract features from an image, edge enhancement is needed. Edges are defined as sharp brightness changes in an image [5.1] and therefore by enhancing these changes special features on an image can be found. In the line following guidance system, the location of the black line has to be identified from the rest of the image. By detecting the right most and the left most edges of the line, the coordinates for the centre of the line can be computed. To determine the regions of sharp brightness changes the gradient of an image can be taken [5.4]. The gradient of an image f(x,y) at location (x,y) is defined as the two-dimensional vector

$$G[f(x,y)] = [G_x \quad G_y]^T = [\partial f/\partial x \quad \partial f/\partial y]^T \qquad (5.2)$$

The vector **G** points in the direction of maximum rate of change of f at location (x,y). Since for edge detection only the magnitude of this vector is necessary, the following results:

$$G[f(x,y)] = [G_x^2 + G_y^2]^{1/2} \qquad (5.3)$$

This quantity is the maximum rate of increase of f(x,y) per unit distance in the direction of **G**. From equation 5.2 the computation of the gradient is based on obtaining the partial derivatives ∂f/∂x and ∂f/∂y at every pixel location. From Figure 5.3 the gradient vectors for any pixel located at (x,y) can be obtained as follows

x_1	x_2	x_3
x_4	pixel	x_5
x_6	x_7	x_8

3x3 Image Region

-1	-2	-1
0	0	0
1	2	1

Gx Mask

-1	0	1
-2	0	2
-1	0	1

Gy Mask

Figure 5.3 Sobel Operator Masks

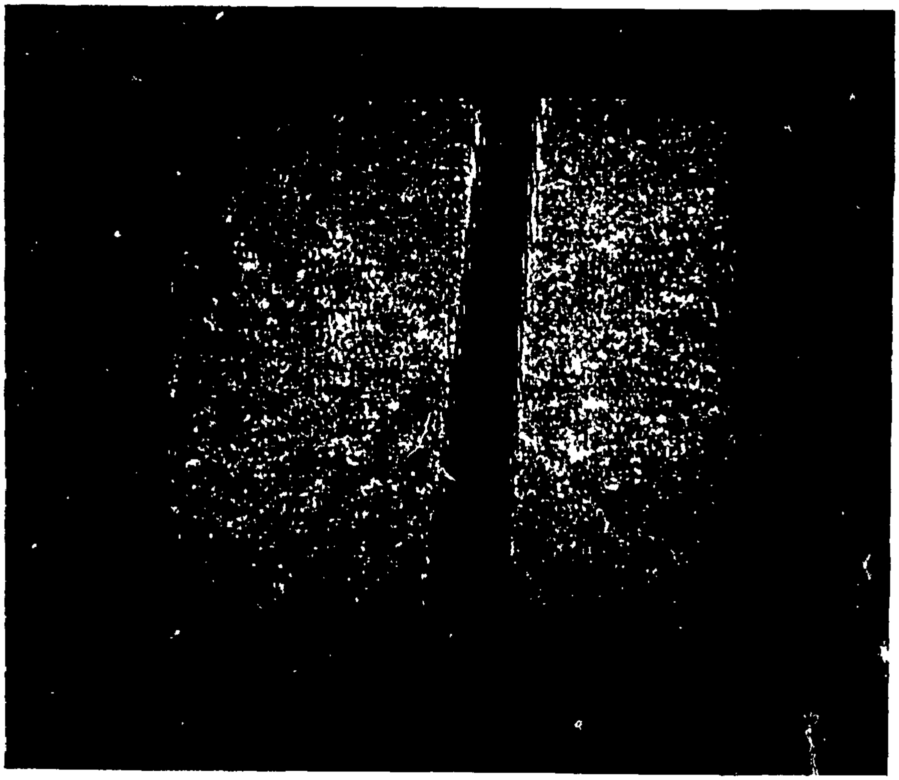


Figure 5.4 Sobel Operators

$$G_y = (x_3 + 2x_5 + x_8) - (x_1 + 2x_4 + x_6) \quad (5.5)$$

G_x and G_y are referred to as Sobel operators Using this type of Sobel operators emphasizes vertical and horizontal edges rather than diagonal edges. As with all types of differential operators the Sobel operators suffer from the inability of detecting edges in high noise environments. Figure 5.4 shows the result of applying the Sobel operators to the same original grey scale image (Figure 5.1). It should be noted that due to the noisy nature of the original image, the results of the Sobel operators are not usable because clear edges cannot be identified. In fact, Sobel operators are used in conjunction with thresholding techniques. Also this algorithm is computationally intensive and time consuming.

5.2.3 Median Filter Algorithm

The primary use of the median filter is for noise removal. Median filteing forces points with very distinct intensities to be more like their neighbours, thus reducing or eliminating intensity spikes that appear isolated in the area of the filter mask. The advantage of using this filter is that it is effective in reducing spike like noise, while preserving the edge sharpness [5.4].

Median filtering is considered more of a point process than an area process [5.3] because it uses the values of the pixels contained in the pixel neighbourhood to determine the new value for the pixel. It does not use any mathematical algorithm to calculate the new pixel value. Instead, the values of the neighbouring pixels are sorted into an ascending order and the median value is used as the new pixel value [5.1]. There are several variations of the filter since a 3x3 mask can be used or a 1x3 or a 3x1 or any *nxm* mask. Also, this filter can be used repeatedly more than once to further reduce noise. Figure 5.5 shows the results of applying a 3x3 filter to the original image (Figure 5.1).

5.2.4 Local Average Filter Algorithm

By using a local average filter image smoothing is achieved. Spurious noise effects can be diminished The main drawback encountered with this method is that edges are blurred and sharp details are lost. A slightly modified version of the local average filter, called local selective average filter, can be used that will preserve edges [5.2]. This technique is still based upon taking an average of the eight pixels surrounding a pixel plus the pixel itself. However the neighbouring pixels are then checked if they differ from the pixel in question more than the average that is computed. If that is the case, then those neighbouring pixels are not included in the computation of a new average value that will be the new value of the pixel in question [5.2]. Usually a 3x3 local selective average filter is used, even though any size can be utilized. The fact that the neighbouring pixels that are more different than the average value are discarded in the new computation of the average, allows for preserving edges in the image. In other words, surrounding pixels which are too distinct from the centre do not contribute to the result, thus preserving the edges and sharp details [5.2]. Figure 5.6 shows the results of applying the local selective average filter to the original image (Figure 5.1). It should be noted that the algorithm is slow since many computations and comparisons have to be performed.

5.3 Performance and Results of Routines

In real time control of a vehicle using camera vision, two important factors are used in evaluating image processing algorithms: quality of processed images and computational

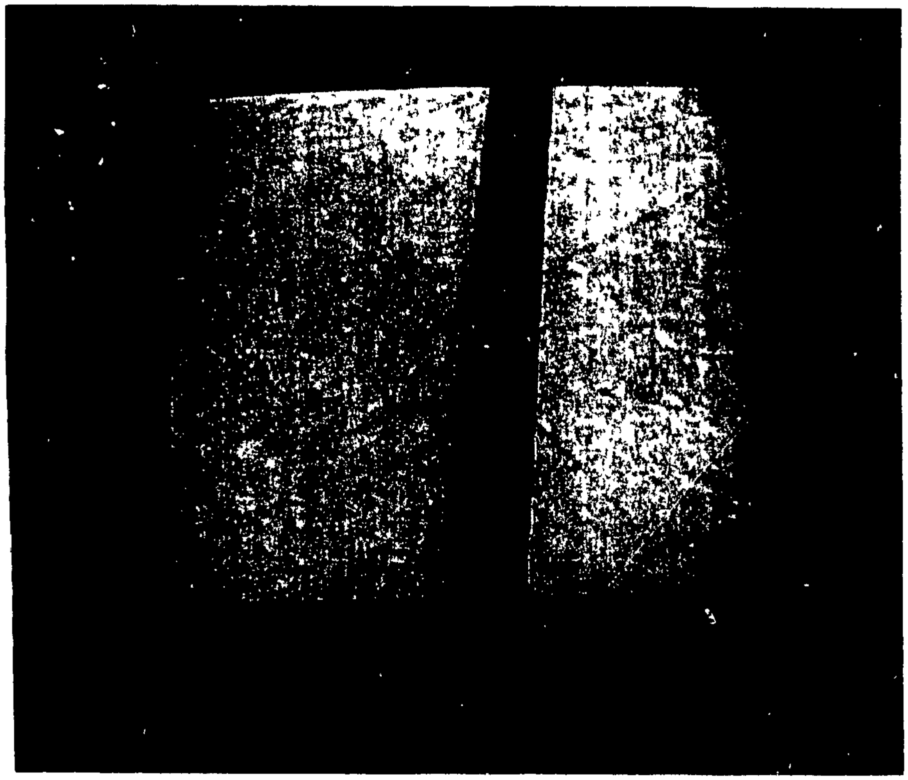


Figure 5.5 3x3 Median Filter

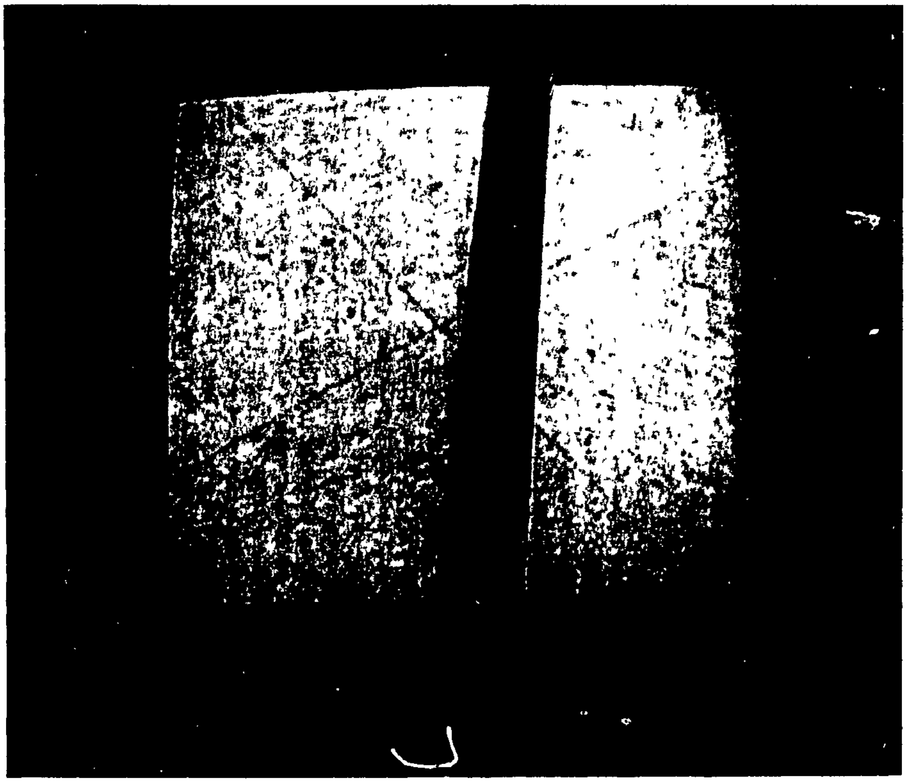


Figure 5.6 Local Selective Average Filter

speed Time is extremely precious in real time control and therefore the algorithm used has to be simple and efficient. A goal of 15 frames/s is desired, which means that each image has to be processed in less then 67 ms Parallel processing will help in achieving this frame rate.

Each of the above algorithms has been tested for execution speed and effectiveness. Table 5.1 summarizes the results of these tests It should be noted that the algorithms were performed on images of 512x512 pixels and 265 grey scale To binarize such an image it takes 134 ms and the results are very good because the resulting image clearly contains a black line and a white background with some noise. The amount of noise changes with the lighting conditions

Applying Sobel operators to an image requires 558 ms. However the resulting image does not contain clear cut edges especially when the line is not perfectly vertical. Also Sobel operators by themselves cannot be used. A thresholding or binarizing operation has to follow. This can be seen in Figure 5.7 and it should be compared with Figure 5.4 where only Sobel operators are used.

Performing median filtering on an image takes 973 ms and the resulting image has less sharp contrast noise. For best results, the median filter should be used along with a binarizing operation as it can be seen in Figure 5.8. The resulting image is almost perfect: the black line is the only entity in the image and almost all background noise is removed and these results hold for a variety of lighting conditions. The algorithm is fairly time consuming since a 3x3 mask was used. In order to reduce the execution time, different size masks were used. A 1x3 followed by a 3x1 mask yields the same results as the 3x3 mask with an execution time of 714 ms. This can be seen in Figure 5.9.

The local selective average algorithm is extremely time consuming. To process one

Algorithm	Execution Time (in ms) *	Processed Image Quality **
Binarizing	134	Very good Improves contrast Leaves backfround noise
Sobel Operators	558	Poor when applied by itself Binarizing algorithm required after
Median (3x3)	973	Good when applied by itself Excellent when applied after binarizing
Median (3x1)	357	Good horizontally in combination with binarizing algorithm
Median (1x3)	357	Good vertically in combination with binarizing algorithm
Local Average	5287	Good following a binarizing algorithm

* Processed Images 512x512 at 256 grey levels
** Only one processor used

Table 5 1 Image Processing Algorithms Results

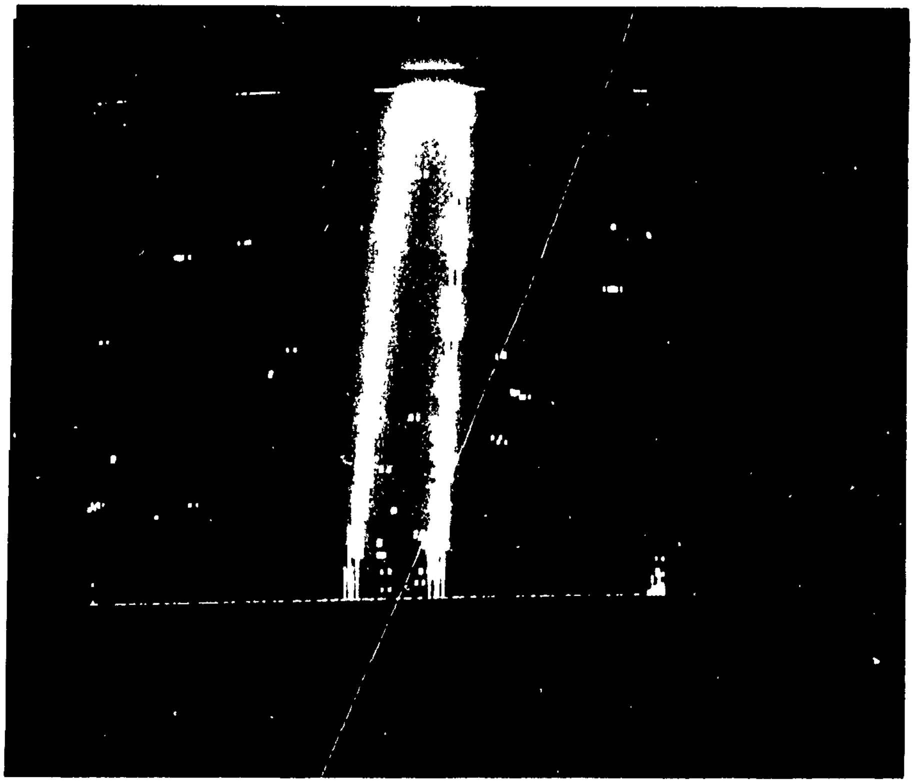


Figure 5 7 Binarized and Sobel Operators

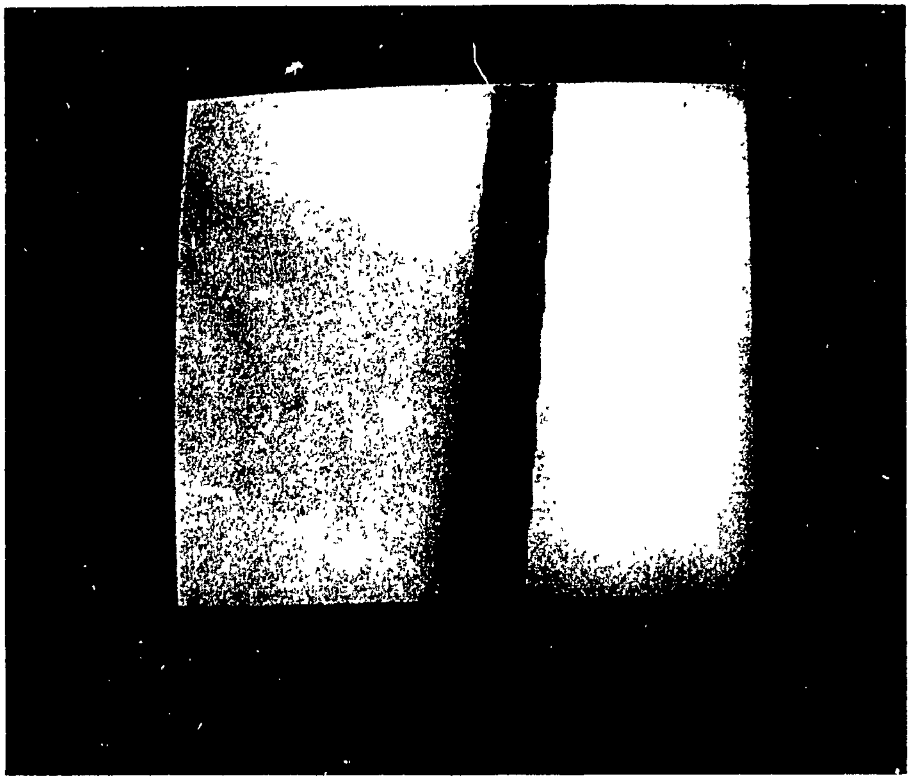


Figure 5.8 Binarized and Median Filter

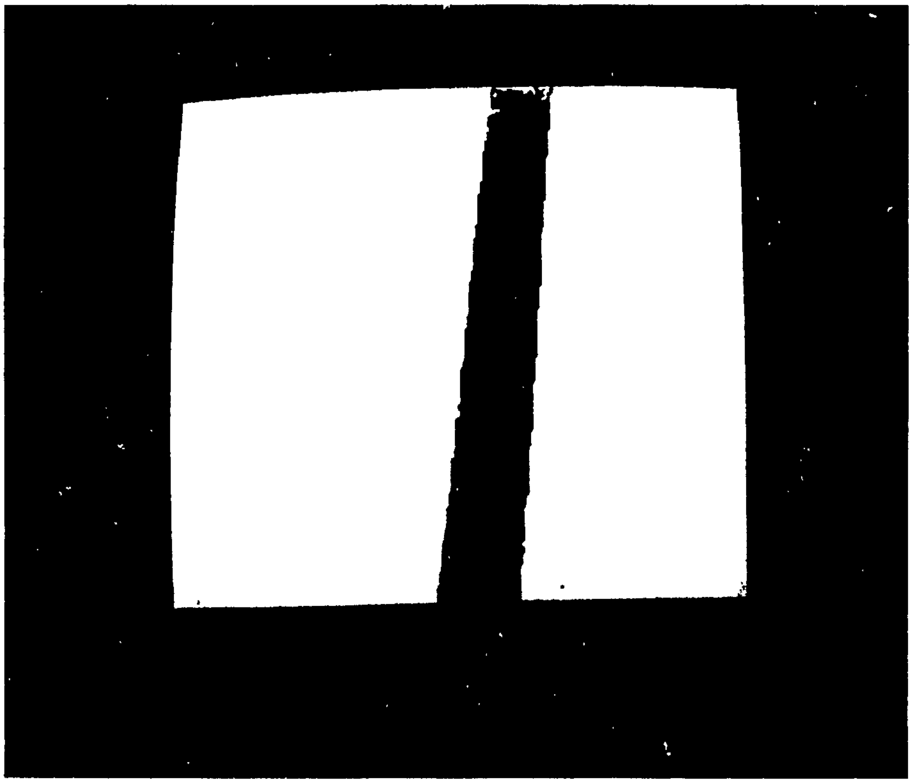


Figure 5.9 Binarized, 3x1, 1x3 Median Filters

image it takes 5287 ms or over 5 seconds. Even though it yields good results when used in combination with the binarizing algorithm as it is shown in Figure 5 10, the strict time requirements are not met at all.

5.4 Image Processing Routines of CONCIC-4

The goal of the image processing routines is to provide to the line extraction algorithm a clear image free from background noise in the least possible amount of time. Since none of the above tested algorithms, from Sinectonalysis [5 2], by itself is capable of providing an image free from background noise, a combination of algorithms has been used. After extensive tests involving several algorithms the following combination yielded excellent results both quality wise and speed wise.

The original image is first binarized using a fixed threshold level Then a 1x3 median mask is used to remove noise in the horizontal direction and finally a 3x1 median mask eliminates the remaining noise in the vertical direction. The end result is an image virtually noise free with a white background and a black line. This whole procedure requires 849 ms when executed in a single processor for a 512x512 monochrome image The order in which the image processing algorithms are executed is also important since the binarizing algorithm has to be performed first to obtain the desired results The median masks can be applied in any order and it should be pointed out that the two masks were chosen over the single 3x3 one because of speed considerations.

In order to further reduce the computational burden, the size of the image was reduced to 256x256 pixels. This was done without big losses in resolution and size of the camera window. However the number of pixels to process is reduced by one quarter. Every

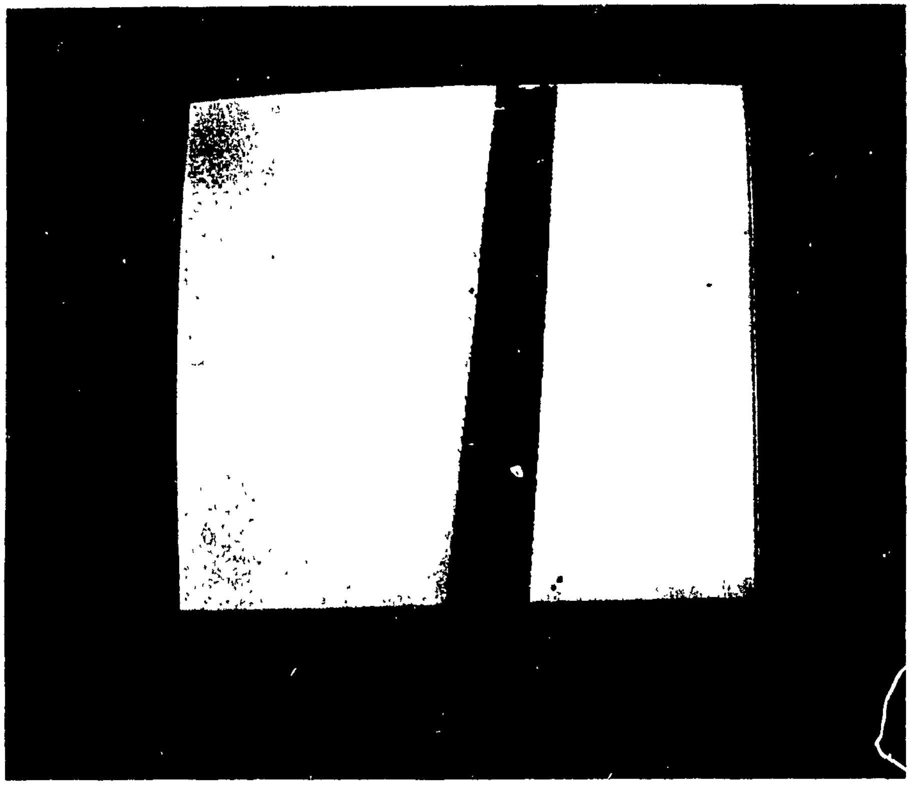


Figure 5.10 Binarized, Local Selective Average Filter

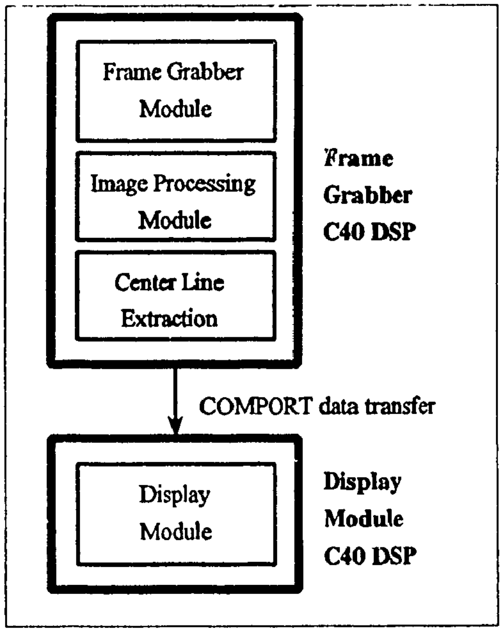


Figure 5.11 Minimal Configuration

second column and every second row is used to reduce the size of the image. The time required for processing reduces from 847 ms to 300 ms which is not quite one quarter of the time since there is still some overhead involved. With this image processing time, a frame rate of only 3.3 frames/s can be achieved which is too slow; therefore some parallel processing will have to be implemented to increase the frame rate.

5.4.1 Implementation of Routines

When testing the image processing algorithms a single processor was used for simplicity purposes. Figure 5.11 shows the minimal configuration required to capture images, process and display them. As it can be seen, the C40 processor on board the frame grabber is used to perform the image processing algorithms along with the line extraction routine The centre line extraction routine will be explained in the next chapter.

Once the required image processing algorithms are defined, the level of performance required can be increased by modifying the C40 network [5.5]. Several configurations are tested. A four C40 network is tried with two C40s processing half of the image each A five C40 network with three C40s processing one third of the image each has been tested. Then six C40s are used with four of them processing one quarter of the image each. To reduce the communication overhead the six C40 network is tested differently: the four C40s each receive a full image in sequence.

Initially, the network is modified to use some degree of parallelism by splitting the image processing between two processors as shown in Figure 5.12. The frame grabber module captures the image, unpacks it, and then passes the top half to one processor and the bottom half to the other processor for image processing. Then the processed image is

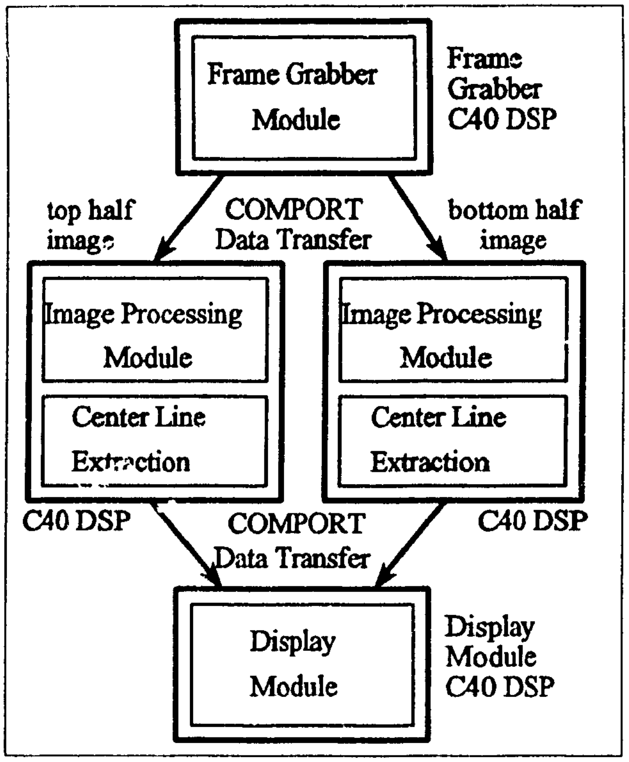


Figure 5.12 4 DSP Configuration

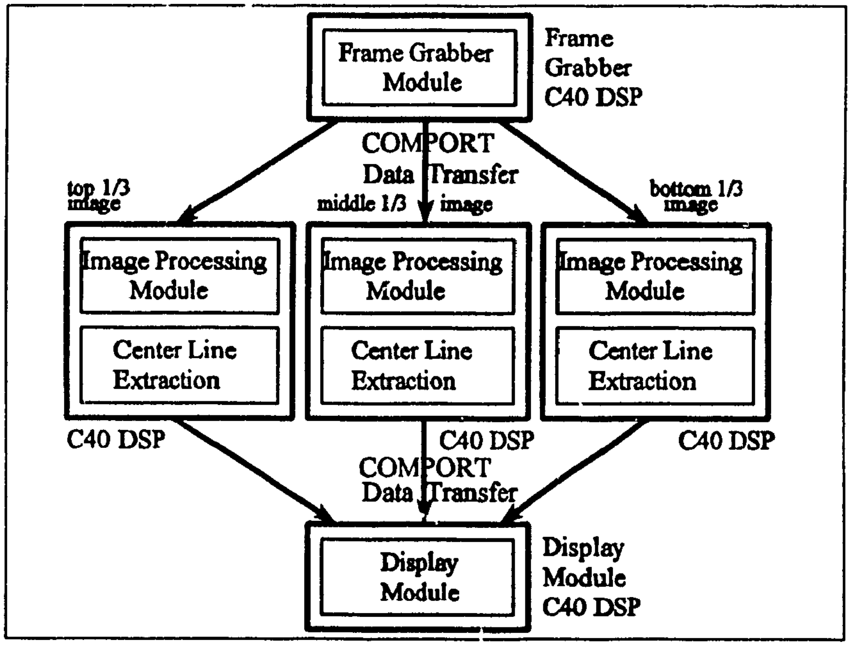


Figure 5.13 5 DSP Configuration

recombined in the display module and the centre line points are passed to the control module (not shown). Each processing C40 receives slightly more than half of the image There are about 6 lines which are being processed by both processors as an overlap, just to make sure there are no discontinuities in the processed images once recombined With this method a doubling of the frame rate from the initial setup can be obtained However 6 6 frames/s is still too slow for AGV control.

The next step to achieve higher frame rates consists in minimizing the data transfers between processors. The network topology remains the same, however the image data is kept compressed, as 4 pixels per 32 bit word, and sent to the two processing C40s Then each processor uncompresses the part of the image that it has received. The image processing algorithms remain the same. The other difference is that the processed image data is recompressed within the processing C40s before being sent to the display module By using these compression schemes the data transferred between processors is reduced by a factor of four. Some time is lost in the compression/decompression process, however the overall results are encouraging since frame rate of 9.93 frames/s are achieved With an optimized image array structure 12 frames/s are obtained. Still some more improvements are sought.

Figure 5.13 shows a 5 DSP arrangement, where the image is processed in three sections in each processing C40 DSP. Again, the image is captured by the frame grabber, kept compressed and one third of the image is sent to each processing C40 DSP. The image is divided in three horizontal slices, however some overlap exists again to keep consistency once the image is recombined. After the image processing routines and line extraction algorithm, the three sets of centre line coordinates are sent sequentially to the control block (not shown) for control purposes. The three sections of the processed image are then sent in compressed format to the display module. With this arrangement 13.7 frames/s can be obtained. To further improve the frame rate, some tests were conducted where the processed image is displayed once every *n* frames. This means that the road information is extracted from every frame but the frames are not displayed all the time. The following results, tabulated in Table 5.2, are obtained: if the processed image is displayeo once every 2 frames, the frame rate is 14.20 frames/s; if the image is displayed once every 3 frames, the frame rate is 14.36 and once every 4 frames it is 14.43. However as more frames are skipped the viewing of the images degrades and the user cannot get a realistic view of what the camera is seeing. Also the gain in frame rate does not justify the negative affects on the viewing of the processed images.

The next network topology tested consists of a six DSP arrangement, where the image is now divided into four sections each processed by a separate C40 DSP. The frame rate obtained is 14 86. There is not much improvement over the previous five DSP configuration and this is due to the extensive use of the communication links for sending smaller packs of information across processors. Since sending data from one processor to another requires some overhead, the time lost in overhead starts to play more of a role when small sets of data are sent at the time. By displaying the processed image every *n* frames, yields the following results (which are also tabulated in Table 5.2): every 2 frames gives 15.7 frames/s, every 3 frames increases to 16.1 frames/s, and every 4 frames decreases to 15.9 frames/s.

In order to reduce the speed losses due to the communication overhead a different strategy is implemented, refer to Figure 5.14. This time, instead of dividing the image into four sections, each whole image is sent to a C40 processor. Therefore, the first C40 DSP (processor-1) receives frame number *n*. It performs the image processing, the centre line

	Performance (in frames/second)	
Displaying	5 DSP Configuration	6 DSP Configuration
every image	13.70	14.9
every 2nd image	14.20	15.7
every 3rd image	14.36	16.1
every 4th image	14.43	15.9

Table 5.2 Displaying of Processed Images and Its Effects on Performance

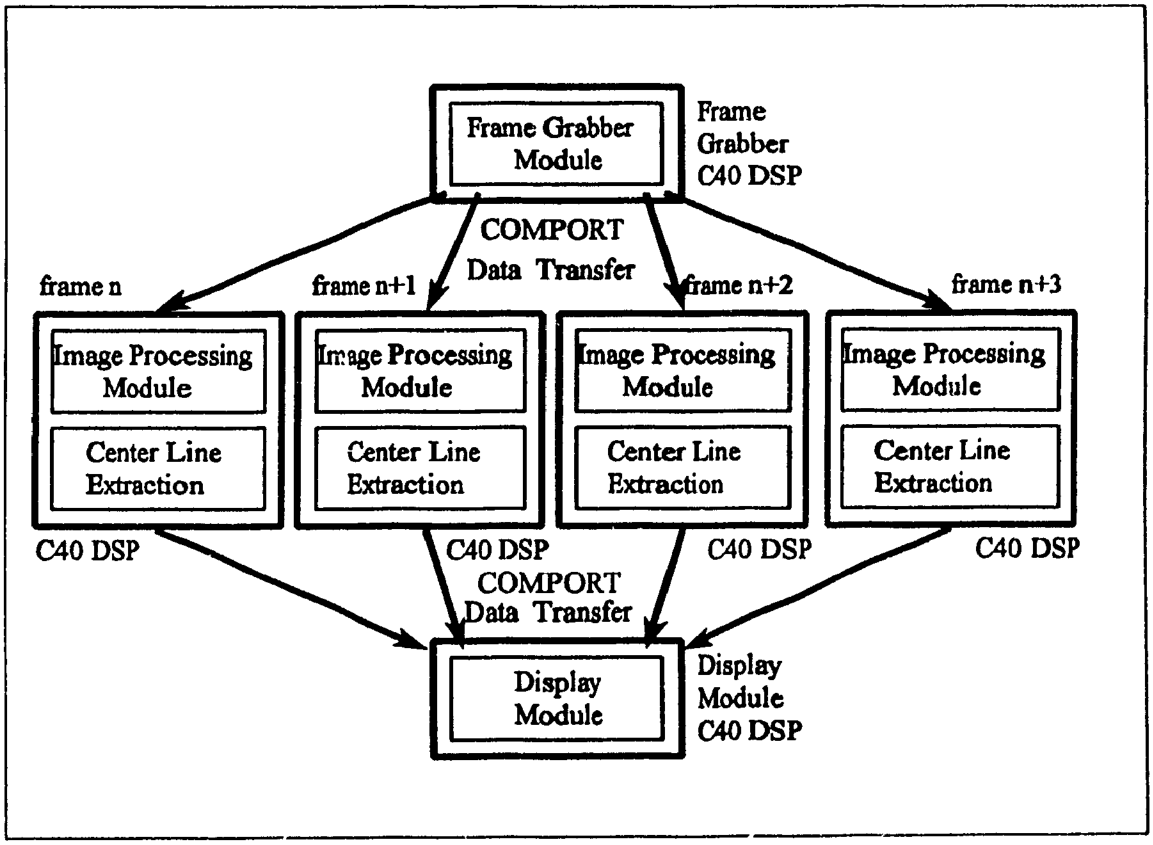


Figure 5.14 6 DSP Configuration

extraction routines on the whole image and then sends the centre line coordinates to the control loop (not shown) and the processed image to the display module. During the time processor-1 is processing image *n*, the second C40 DSP (processor-2) receives frame number *n+1* and performs the same operations as the other processor. The third C40 DSP (processor-3) works on frame *n+2* and the fourth C40 DSP (processor-4) processes frame *n+3*. Then processor-1 receives frame *n+4* and the sequence continues. With this scheme, a frame rate of 16.3 frames/s can be achieved, while displaying the processed image every frame. Another advantage of this scheme is that the road centre coordinates are available all at once from one processor. The drawback of this scheme is that the road coordinates are sent to the control loop with a delay which is longer than the delay inherent with the schemes where the image is divided into sections and processed in parallel. However, this is not a problem because this time delay can be compensated for in the control loop.

Due to some hardware limitations, the final C40 network implemented in CONCIC-4 is a five C40 DSP network, consisting of two C40s performing image processing and line extraction routines, one C40 executing the guidance control loop, one frame grabber and one display module. Each image is processed as a whole by a C40. The topology can be seen in Figure 5.15, noting that the guidance control C40 is not shown. The frame rate obtained with this configuration is 15.0 frames/s which is adequate for AGV control.

5.5 Summary

This chapter presents the image processing routines implemented to obtain images which are noise free and to some extent immune from changing lighting conditions. The algorithms chosen are effective and fast; combined with parallel processing a frame rate of 15

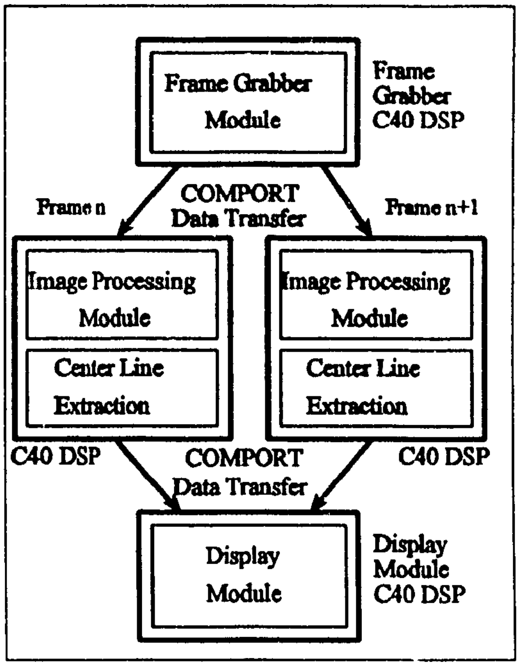


Figure 5.15 CONCIC-4 Image Processing Configuration

frames/s can be achieved. This is quite remarkable since over 7.8 Mbits of data are processed per second just in the image processing routines.

CHAPTER 6

IMAGE ANALYSIS ALGORITHM

6.1 Introduction

The image processing routines are necessary in order to provide a noise free image containing just one entity, the track, to the image analysis routine. It is this routine that searches and extracts the centre point coordinates of the track. Once these coordinates are found then a curve fitting algorithm is used to provide to the guidance control loop the necessary formulation. There are some assumptions made regarding the type of line that can be used for guidance: the line has to be of a certain width, it cannot intersect with other lines, and it has to be continuous in the direction of motion of the AGV; lateral shifts are allowed. The image analysis algorithm has also the feature of further rejecting small amounts of background noise. While analyzing the image some perspective corrections have to be implemented to compensate for the deformation of the camera view, since the camera is mounted at an angle looking down at the ground.

This chapter describes in section 6.2 how the road information is extracted from the images. In section 6.3 the relation between pixels and mm is discussed along with the necessary perspective correction. Then, the formulation for the straight line fit to represent the coordinates of the centre line is presented in section 6.4.

6.2 Road Information Extraction

After the image processing algorithms described in Chapter 5, an image similar to the one in Figure 6.1 is sent to the centre line extraction algorithm. This algorithm scans the

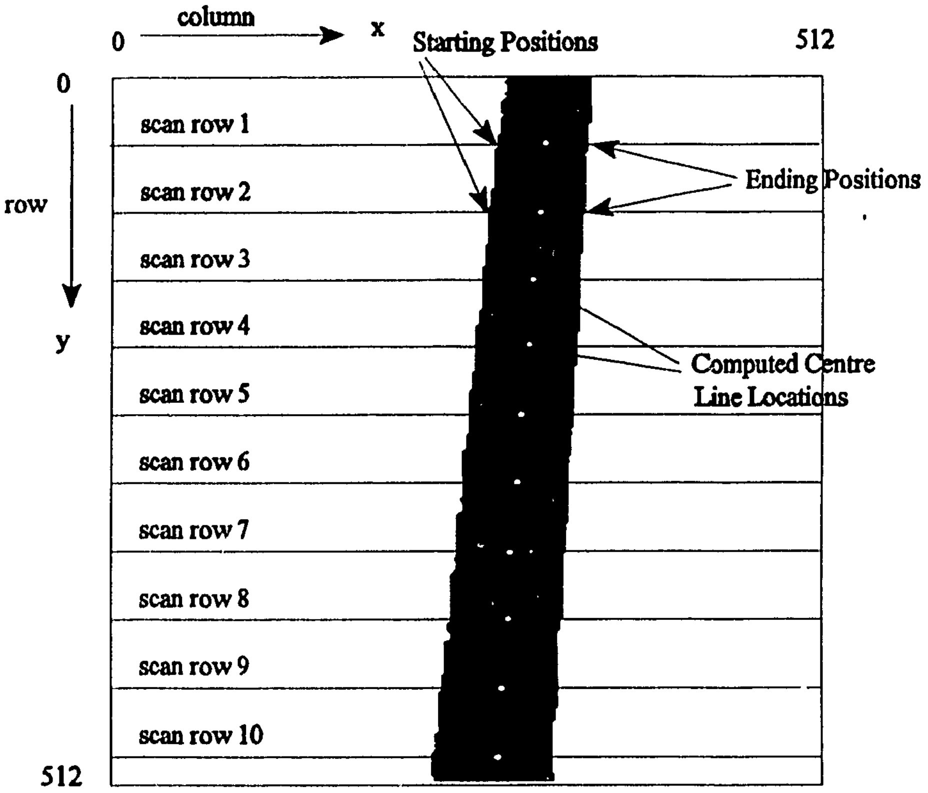


Figure 6.1 Image Horizontally Scanned by Image Analysis Algorithm

image horizontally for the presence of a black line. To speed up the detection process not every row is scanned. As shown in Figure 6 1, ten equally spaced rows are selected to locate the horizontal position of the line in the image. Once these coordinates are found, a line fitting algorithm is used to compute the equation of the resulting line The number of horizontal rows that will be used to identify the guidance line can be changed in the initial setup by the user. After extensive testing it was found that scanning ten rows yields good results. If too few rows are scanned then the resulting line fit would not be accurate enough, especially for curved tracks. If too many rows are scanned then the processing time increases.

The algorithm is shown in the flowchart of Figure 6 2 and works as follows Starting with scan row number one, each pixel on that row is checked to see whether it is white or black. If the pixel is white then the following one is checked If on the other hand the pixel is black then a tentative starting position for the line is assigned to that pixel location in the horizontal axis. The vertical position is known already since each scan row is chosen at a specific vertical row location. Once a tentative starting position has been assigned and the following four pixels are black then the starting position is made official. If the following pixels are white then the tentative starting position is reset since the black pixel encountered was just a noise pixel. The line to be recognized has to be at least four pixels wide. On average that corresponds to approximately 1.2 cm since a pixel in the middle of the image corresponds to 3 mm. Further discussion of pixel mapping can be found in section 6.3 of this chapter.

Once an official starting position has been found then the scan row is scanned for the next white pixel which corresponds to the end position of the line This position is recorded as the official ending coordinate for the line at that scan line. Once the starting and ending

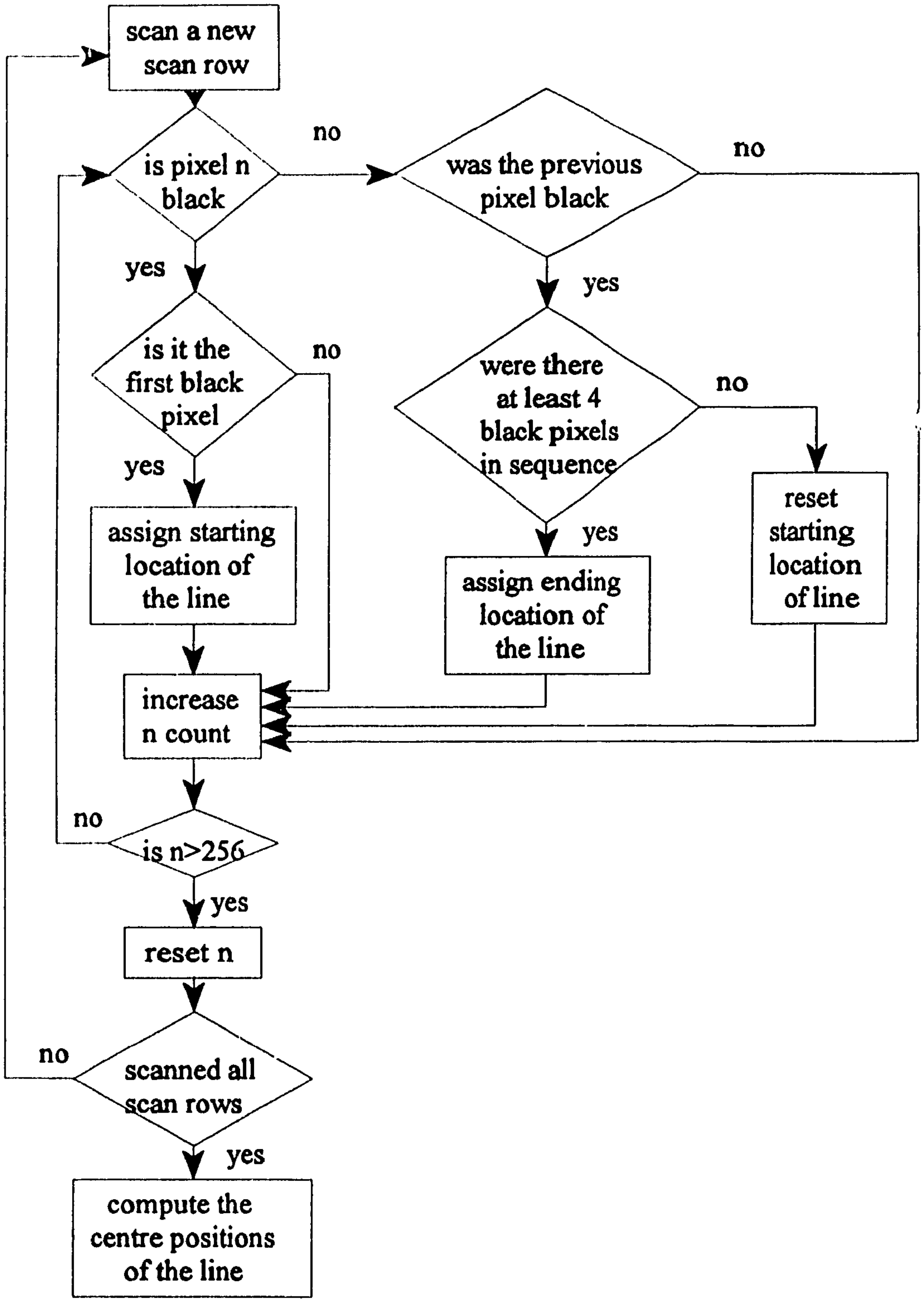

scan a new scan row
is pixel n black
no
was the previous pixel black
no
yes
yes
is it the first black pixel
no
were there at least 4 black pixels in sequence
no
yes
assign starting location of the line
yes
reset starting location of line
assign ending location of the line
increase n count
no
is n>256
yes
reset n
no
scanned all scan rows
yes
compute the centre positions of the line

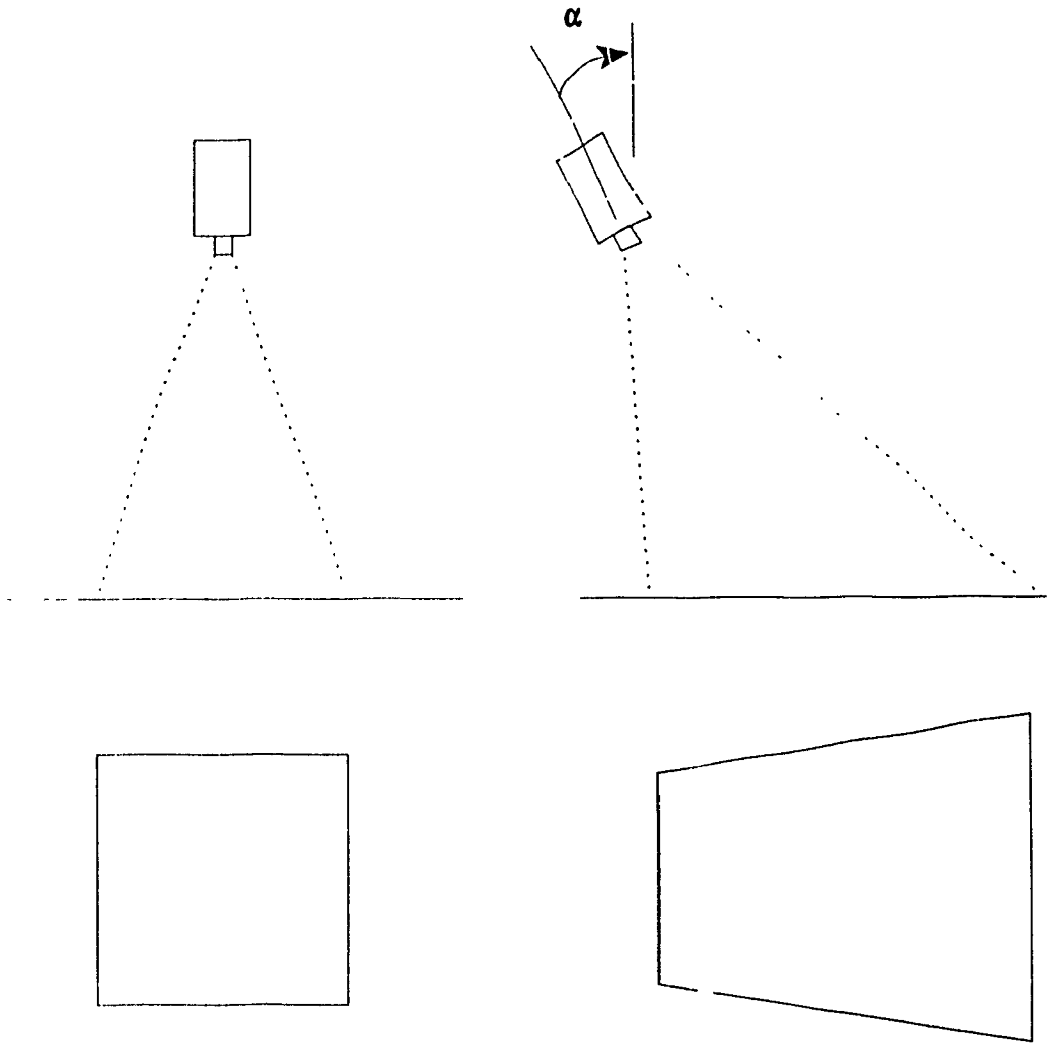
Figure 6.2 Flowchart of Image Analysis Algorithm

coordinates are found then a simple average is taken to find the centre of the line. This procedure is repeated for the total number of scan rows selected. The end result of the algorithm is a two dimensional array containing the x and y pixel coordinates of the line. The upper left hand corner of the image is the origin, that is where pixel with coordinates (0,0) is found. The pixels representing the found centres of the track are superimposed over the original image so that the user can visually see where the computed centre of the track is located. Also the array containing the coordinates of the centre of the line is sent to the main control loop where it is used for guidance control. If more than one object is found then a bad image flag is set to reject the image. This means that the guidance control loop does not receive any new feedback for this time period. The previous feedback information is used again. Up to four images in a row can be rejected before the vehicle is halted. This approach is similar to that used by Rajagopalan [6.1] in CONCIC-2.

As it was mentioned earlier, the algorithm is capable of rejecting background noise as long as the noise is not larger than four pixels. It should be pointed out that if a blob is present but not on one of the scanned rows then, it will not affect the centre line extraction algorithm because it will not be seen.

6.3 Determination of Pixel to mm Mapping

An important step in the image analysis is the identification of the relation existing between image pixels and real life objects If the camera was pointing straight down, i.e. perpendicular to the ground then every pixel would represent the same amount of an object. However since the camera is tilted at an angle with the horizon, then prospective deformation plays a role in the camera view of objects. Figure 6.3 graphically shows what the camera sees



Camera View from
Camera Perpendicular
to the Ground

Camera View from
Camera at an Angle
to the Ground

Figure 6.3 Relation between Camera Location and Camera View

in both cases. More specifically to CONCIC-4, Figure 6.4 shows how camera view pixels are related to real distances. As it can be observed in Figure 6 4 the camera sees in the top of its viewing window 90 cm of the ground whereas in the bottom part only 62 cm can be seen Since this view is mapped in a square image of 256x256 pixels, it can be said that not all pixels have the same resolution. Each pixel at the top of the image represents 3.5 mm of ground, whereas each pixel at the bottom of the image represents only 2.4 mm. Therefore, an equation representing the pixel resolution at any row n location can be derived as follows: assuming a linear change of resolution from row 0, at the top, to row 256, at the bottom

$$\text{pixel resolution at row } n \text{ (in mm/pixel)} = -0.004297\ n + 3\ 50 \qquad (6.1)$$

This explains why in figure 6.3 the camera view has a constant width straight line whereas the pixel mapping shows a straight line reducing in width. This is due to the fact that, at the bottom to represent the 5 cm in width line 20 pixels are necessary, whereas at the top only 14 pixels are required. This correction for pixel resolution is necessary when computing the actual centre line coordinates in terms of cm from the pixel values previously found

6.4 Formulation for Straight Line Fit

Once the centre line coordinates are identified, they can be used to describe the position and orientation of the AGV with respect to the guidance line. However, these points need to be combined to form the equation of a line so that this equation can be used by the guidance control algorithm. For simplicity, a straight line fit can be used to represent the actual guidance line on the ground. Also, a straight line fit would simplify the way by which the position and orientation of the AGV are computed. Section 7.2 describes how the AGV's position and orientation with respect to the guidance line are calculated. The method adopted

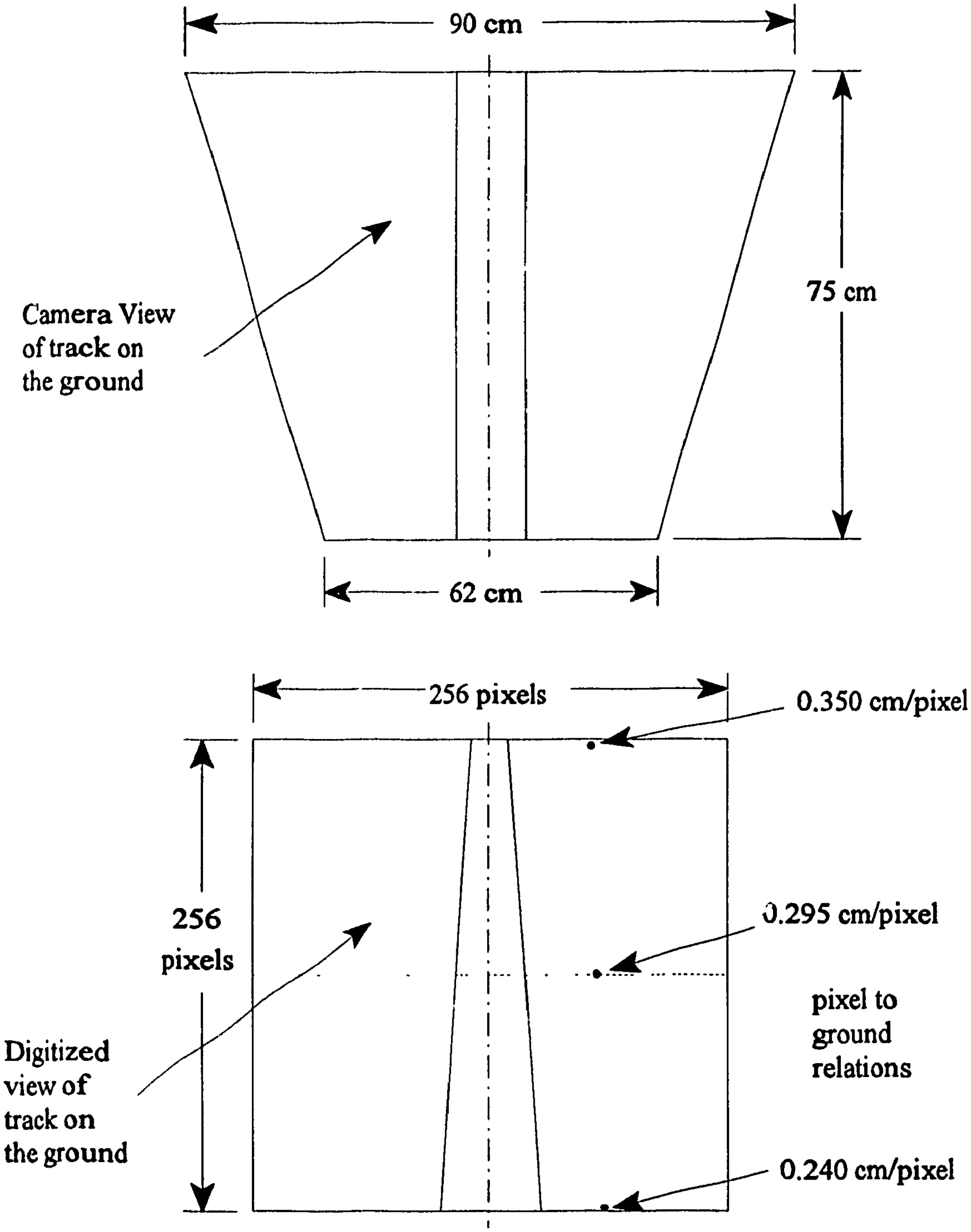

90 cm
75 cm
Camera View of track on the ground
62 cm
256 pixels
0.350 cm/pixel
256 pixels
0.295 cm/pixel
pixel to ground relations
Digitized view of track on the ground
0.240 cm/pixel

Figure 6.4 Relation between Camera View of Track and Digitized View of Track

here is similar to the scheme developed by Rajagopalan in [6.1]. This section is provided for the sake of completeness.

Given the centre line coordinates, a straight line can be found that is closest to the points by minimizing the squared distance from each point to the straight line. This method of curve fitting is called least squares method and it is used to find a regression line that best fits the data points as presented by Etter [6.2]. The equation of a straight line has the form

$$y = mx + c \tag{6.2}$$

where m is the slope of the line and c is the y-intercept of the line. With this method, for each sample observation (x_i, y_i - where y_i represents the row number and x_i the corresponding column location of the centre line) the deviation of y_i from its expected value (e_i) is calculated:

$$e_i = y_i - (c + m\,x_i) \tag{6.3}$$

Then the sum of the n squared deviations is considered

$$Q = \sum_{i=1}^{n} (y_i - c - mx_i)^2 \tag{6.4}$$

and according to the method of least squares values, for c and m can be found that minimize the criterion Q for the given sample of (x_i, y_i) [6.3]. Using calculus, the values of c and m which minimize Q can be derived by differentiating (6.4) with respect to c and m

$$\frac{\delta Q}{\delta c} = -2\sum (y_i - c - m\,x_i) \tag{6.5}$$

$$\frac{\delta Q}{\delta m} = -2\sum x_i\,(y_i - c - m\,x_i) \tag{6.6}$$

These equations can be solved and the values for c and m can be found. After simplification, the values for *c* and *m* are calculated as described by Rajagopalan in [6.1]:

$$c = \frac{\sum x^2 \sum y - \sum x \sum xy}{n \sum x^2 - (\sum x)^2} \tag{6.7}$$

$$m = \frac{n \sum xy - \sum x \sum y}{n \sum x^2 - (\sum x)^2} \tag{6.8}$$

Once *c* and *m* are computed the regression line is found. This line can then be used for guidance purposes as it is explained in chapter 7.

6.5 Summary

This chapter presents an algorithm that analyzes the image and extracts the coordinates of the track. Once these coordinates are found, a regression line is computed to represent these points into a mathematical expression that will be used in the guidance control loop. A linear regression is implemented using the least squares method to fit the line. A linear fit is chosen for its simplicity and also because it simplifies the computation of the position and orientation of the AGV.

There are two identical versions of this algorithm running concurrently into the two C40 processors that execute the image processing algorithms. It is advantageous to perform the image analysis routine within the same processor because the array containing the image is already into the local C40 module's memory, therefore considerable time can be saved by not sending the array to another processor. Also, the time required to analyze the image is negligible as compared to the image processing routines.

CHAPTER 7

GUIDANCE CONTROL SCHEME

7.1 Introduction

This chapter presents the guidance control scheme used by the vehicle for road following. The vehicle follows the track profile based on the present track information received from the camera. This is a feedback system that compares the vehicle path with the track information and takes corrective action to insure that the two paths are the same. The guidance system implemented uses a simple proportional controller to keep the camera window in the centre of the track. With this scheme, the rear end of the vehicle may not be aligned with the track all the time [7.1].

Since this scheme uses the present camera information to guide the vehicle, the camera window has to be of a certain size; otherwise as the vehicle travels faster, it could lose control because there would be gaps in the snap shots of the road that provide the control information. This can be seen in Figure 7.1 where two camera windows are compared. Window A is a smaller camera view of the road than window B. As the speed is increased, window A starts to miss viewing portions of the road at an arbitrary speed labelled 5, whereas window B views the road up to a higher speed labelled 8. The number represents the amount of times the speed is doubled. The faster the vehicle moves, the more road is travelled in the same amount of time because the sampling time Δt is constant. However, the camera view cannot be too large because the further the camera sees, the less resolution it has. Each pixel would represent more information. This can be see again in Figure 7.1.

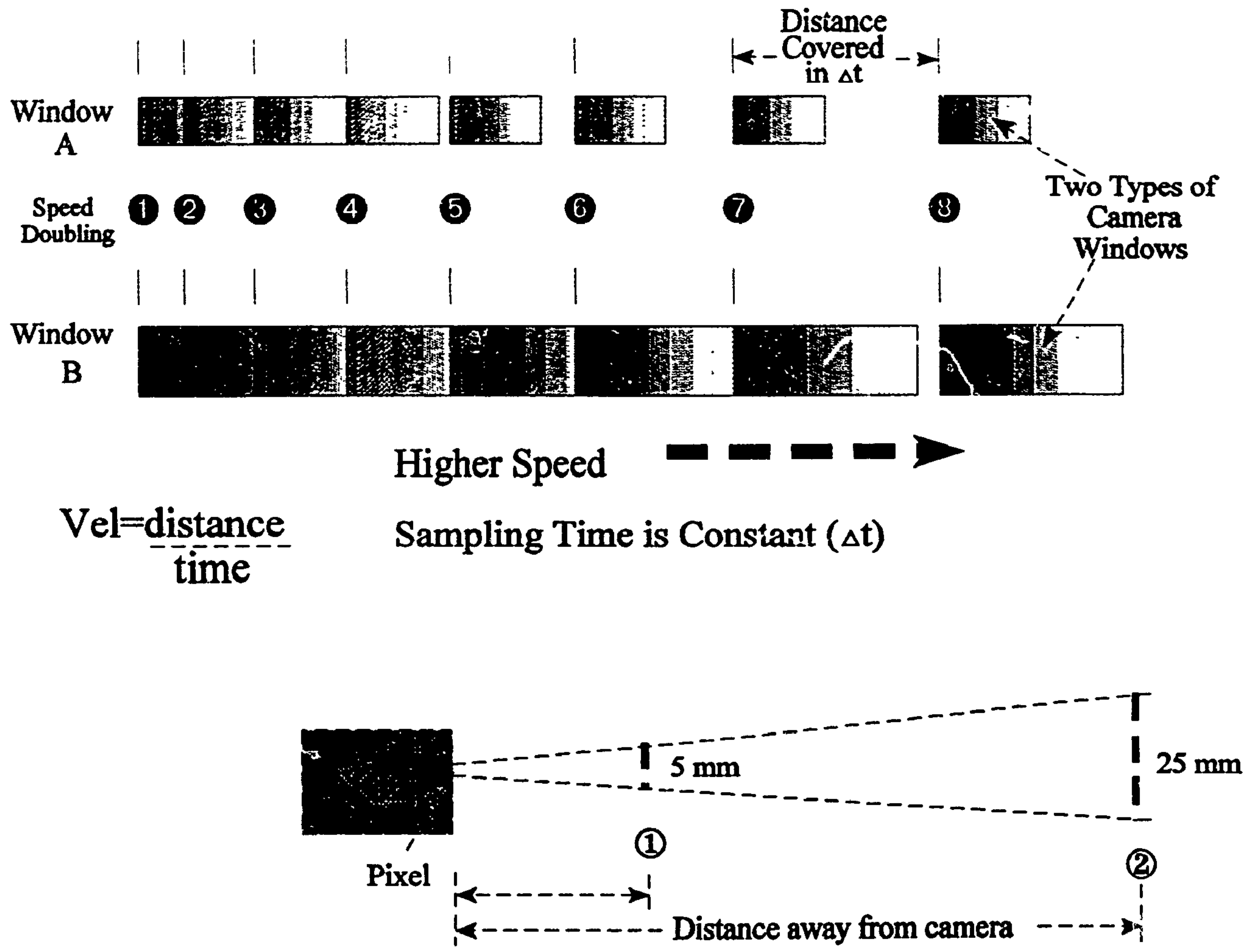


Figure 7.1 Camera View Size Relations

7.2 Definition of Position and Orientation Offsets

In order to provide guidance control to the vehicle, its position and orientation with respect to the guidance line need to be found. The approach developed by Rajagopalan [7.1] to compute the position and orientation offsets is used in this thesis. Figure 7.2 describes graphically the meaning of position and orientation offset. Position offset (ε_d) is the perpendicular distance between the centre line of the track image and the centre line of the camera image window [7.1]. Orientation offset (ε_θ) is the angle between the direction of travel by the vehicle and the direction pointed to by the track centre line [7.1]. Given that the centre line equation is

$$y = m\,x + c \tag{7.1}$$

where *m* and *c* are parameters found to represent the best fit for the coordinates of the centre line, ε_d and ε_θ can be found. Using equation 7.1 and substituting y=0 and x=ε_d, an expression for ε_d in terms of m and c is computed as follows

$$\varepsilon_d = - c / m \tag{7.2}$$

Looking back at Figure 7.2 the expression for ε_θ is obtained as

$$\varepsilon_\theta = \tan^{-1} (1 / m) \tag{7.3}$$

These two guidance parameters are going to be combined to provide a steering angle for the vehicle.

7.3 Computation of Steering Angle using a Proportional Controller

The information regarding the vehicle's position and orientation needs to be combined and used to compute a steering angle to correct for any existing offsets. The control law to determine the steering angle θ_s is as follows [7.1]

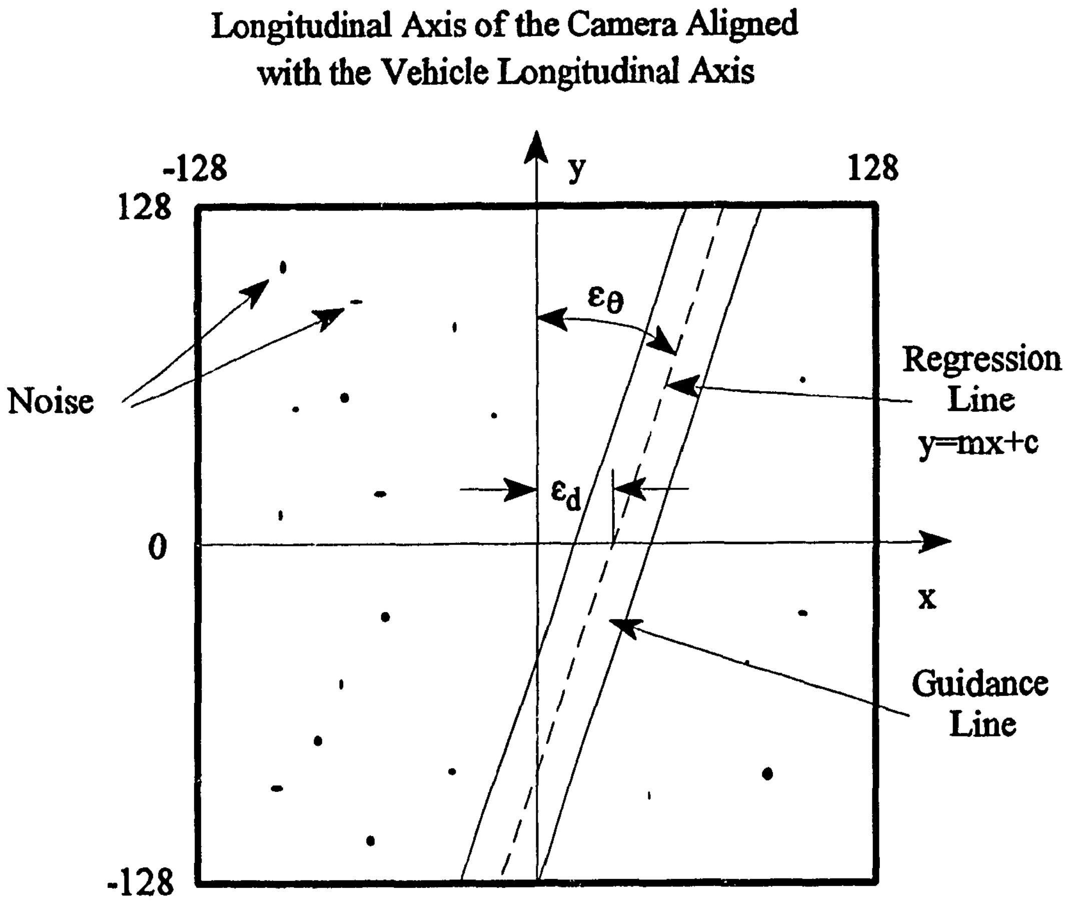


Figure 7.2 Camera Window with Guidance Line Parameters

$$\theta_s = G_1 \, \varepsilon_d + G_2 \, \varepsilon_\theta \tag{7.4}$$

where G_1 and G_2 are gains that multiply the offset values. The values for G_1 and G_2 will be determined experimentally. These gains have to be properly selected otherwise instabilities could occur. Figure 7.3 shows the overall control loop of the system. The feedback is provided by the camera as a position offset and an orientation offset. These offsets are combined in the proportional controller and sent to the inverse kinematic model to determine the wheel speeds and steering angles. The low level closed loop motor control is executed by the LM628 motion controllers.

7.4 Vehicle Wheelbase Geometry

The vehicle is configured like a car with the front wheels driving: it has two driving and steering wheels in the front and two driven wheels in the back. The mechanical configuration of most wheeled land vehicles falls into one of two categories: steered-wheeled vehicles or differential-drive vehicles. Cars are part of the steered-wheeled vehicles, whereas a military tank would be part of the differential-drive vehicles [7.2]. CONCIC-4 belongs to the steered-wheeled vehicle category. Since it has a fixed rear axle and two front steered wheels that can be approximated by a single wheel at the centre, it has a tricycle configuration [7.2].

Using a tricycle configuration the inverse kinematics is obtained and the steering commands are calculated for a fictitious centre wheel along the longitudinal axis of the vehicle and then decomposed for each front wheel. The same applies for the velocity commands: the velocity calculated for the fictitious centre wheel is recalculated for each individual wheel. The inverse kinematics relates the vehicle's velocity and steering to the wheels' velocities and

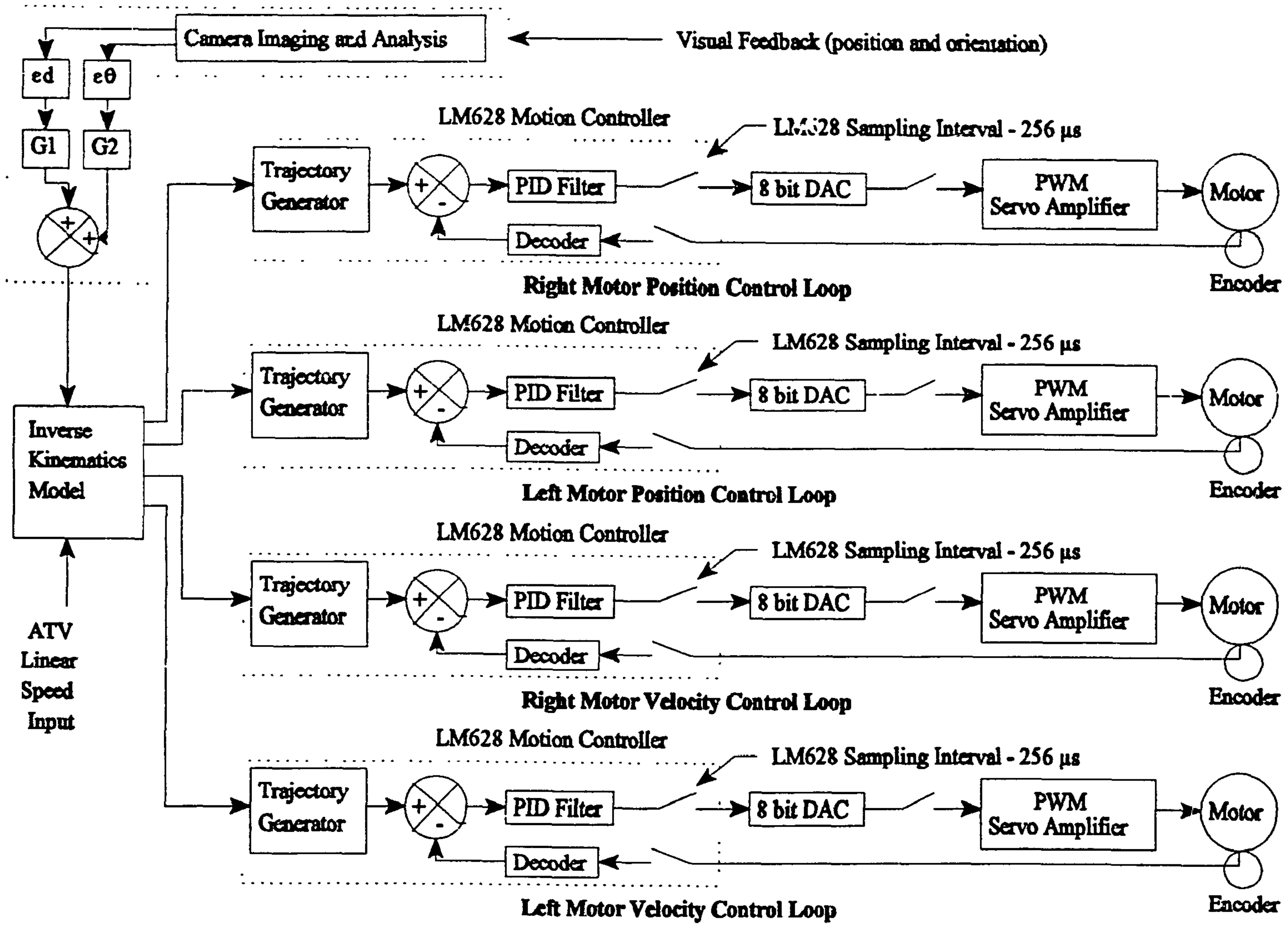


Figure 7.3 Control Loop with Visual Feedback

steering. If the wheels' velocitics and steering angles are not matched, then slippage can occur. In order to compute the correct wheels velocities and steering angles, the inverse kinematics of the vehicle need to be derived. A geometric approach is used to find the inverse kinematic relations for determining wheel speeds and steering angles, given the vehicle's desired velocity and direction. Figure 7.4 describes the wheelbase geometry and the various dimensions used to derive the inverse kinematic formulas.

7.4.1 Computation of Steering Angles

This computation involves resolving the steering angle assigned to a fictitious wheel located at point M of the diagram into two different angles, one for the left steering wheel and one for the right steering wheel. These actual steering wheels are located at point D and C respectively. Also, the steering angle for the fictitious wheel is labelled θ_s, whereas the left steering angle is called θ_{s1} and the right one θ_{s2}. Point O is the centre of rotation from which all angles are measured.

In order to find θ_{s1} and θ_{s2}, some geometry is used as described by Rajagopalan [7.3]. Starting with θ_{s2} the following relations are applied:

$$OP = \frac{L_{ac}}{\tan\theta_s} \tag{7.5}$$

where L_{ac} is the wheelbase of the vehicle. Then

$$OQ = OP - \frac{L}{2} = \frac{L_{ac}}{\tan\theta_s} - \frac{L}{2} \tag{7.6}$$

and the tangent of θ_{s2} can be found as

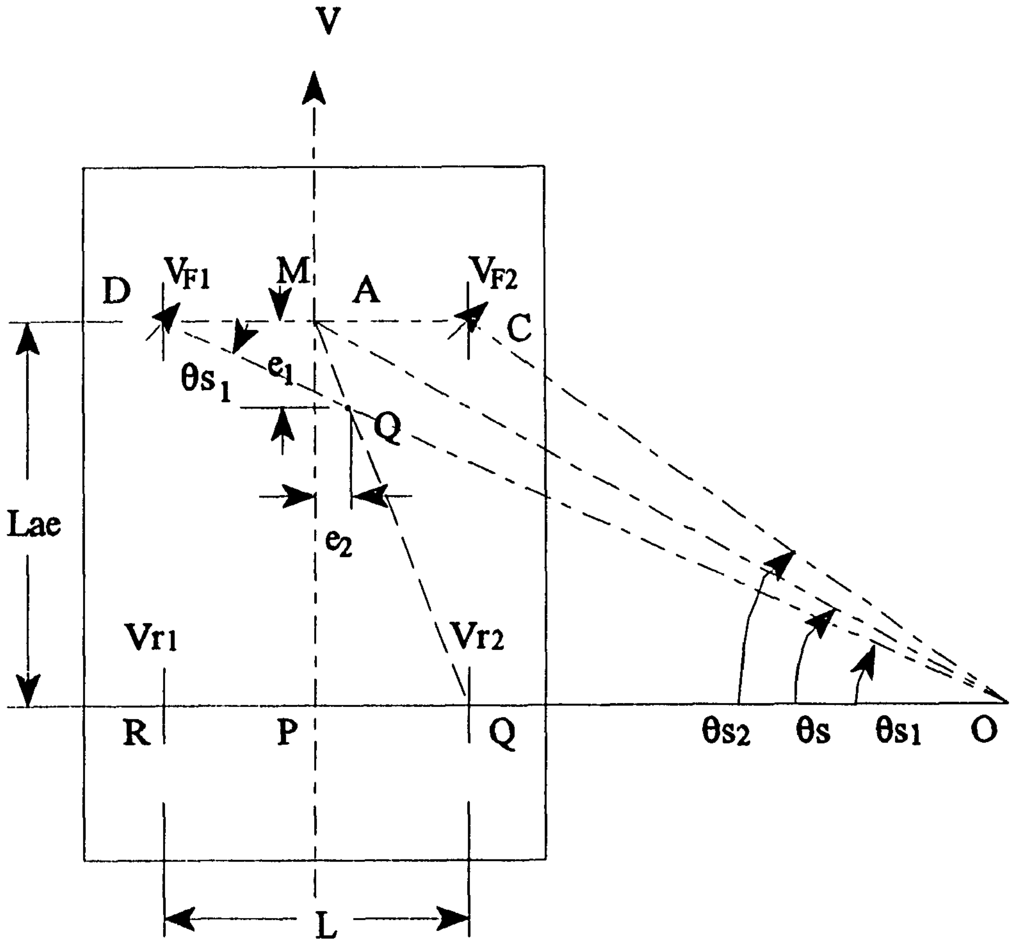


Figure 7.4 Geometric Approach to Find Inverse Kinematics of Vehicle [7.3]

$$\tan\theta_{s2} = \frac{L_{ae}}{OQ} = \left[\frac{L_{ae}}{\dfrac{L_{ae}}{\tan\theta_s} - \dfrac{L}{2}}\right] \tag{7.7}$$

and finally θ_{s2} is equal to

$$\theta_{s2} = atan\left[\frac{L_{ae}}{\dfrac{L_{ae}}{\tan\theta_s} - \dfrac{L}{2}}\right] \tag{7.8}$$

To calculate θ_{s1} a similar geometric approach is used. Referring to Figure 7.4

$$\cot\,\theta_{s1} = \left(\frac{L}{2} + e_2\right) \,/\, e_1 \tag{7.9}$$

$$\cot\,\theta_{s2} = \left(\frac{L}{2} - e_2\right) \,/\, e_1 \tag{7.10}$$

$$\cot\,\theta_{s2} - \cot\,\theta_{s1} = -\,2\,e_2 \,/\, e_1 = -\frac{L}{L_{ae}} \tag{7.11}$$

Separating the term with θ_{s1}

$$\tan\,\theta_{s1} = \left[\frac{1}{\dfrac{1}{\tan\,\theta_{s2}} + \dfrac{L}{L_{ae}}}\right] \tag{7.12}$$

and finally θ_{s1} can be calculated as

$$\theta_{s1} = atan\left[\frac{1}{\dfrac{1}{\tan\,\theta_{s2}} + \dfrac{L}{L_{ae}}}\right] \tag{7.13}$$

7.4.2 Computation of Wheel Speeds

The velocity of the vehicle has to be decomposed into wheel velocities. While the vehicle is turning, the outer wheel to the turn will have a higher speed than the inner wheel. The steering angle of the inner wheel will be larger than the outer one. Figure 7.5 shows how the wheel speeds ω_{x1} and ω_{x2} are derived from the vehicle's velocity V. The wheel speeds have to have the right proportions or otherwise slippage could occur. The angular velocity of the vehicle is

$$\psi = \frac{V \tan \theta_s}{L_{ae}} \tag{7.14}$$

The front wheel velocities are

$$V_{F1} = \frac{\psi \, L_{ae}}{\tan\theta_{s1}} \tag{7.15}$$

$$V_{F2} = \frac{\psi \, L_{ae}}{\tan\theta_{s2}} \tag{7.16}$$

but they can be simplified by substituting the angular velocity of the vehicle

$$V_{f1} = V\frac{\tan\theta_s}{\tan\theta_{s1}} \tag{7.17}$$

$$V_{f2}= V\frac{\tan\theta_s}{\tan\theta_{s2}} \tag{7.18}$$

Finally the angular wheel velocities can be computed as follows

$$\omega_{x1} = \frac{V\dfrac{\tan\theta_s}{\tan\theta_{s1}}}{r} \tag{7.19}$$

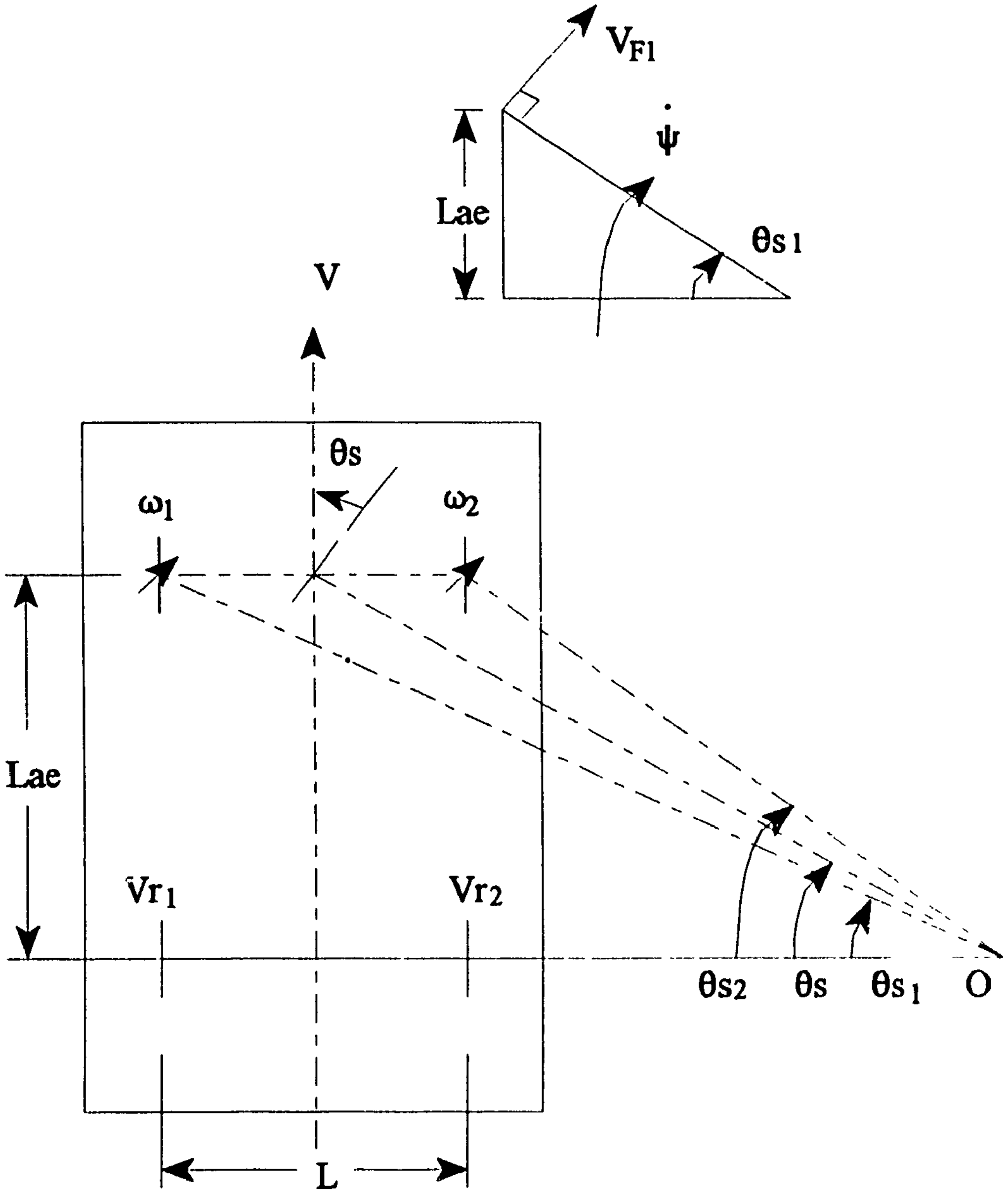


Figure 7.5 Resolving Vehicle's Velocity into Wheel Speeds [7.3]

$$\omega_{x2}=\frac{V\,\dfrac{\tan\theta_{s}}{\tan\theta_{s2}}}{r} \tag{7.20}$$

where *r* is the radius of the wheel.

7.5 Low Level Motion Controllers

Once the wheel speeds and steering angles are determined they are used to instruct the low level motion controllers to perform closed loop control on each wheel. There are four independent motion controllers: two performing velocity control and two executing position control. Figure 7.6 shows how the closed loop motor control is done. An explanation of the LM628 motion controller follows.

The host computer specifies the speed of each driving wheel. The motion controller performs closed loop control and keeps the wheel rotating at the desired speed. For the steering motors the host computer sends the angular position. The motion controller by functioning in position mode, as opposed to velocity mode as before, keeps the wheel at the specified steering angle. Each wheel assembly is comprised of two motors: one for driving and one for steering.

The LM628 is a dedicated precision motion controller that can be used with a variety of DC motors as long as it is provided with a quadrature incremental position feedback signal [7.4]. The advantage of using such a controller is that the host computer only has to instruct the motion controller of what to do and the controller performs the operation without further assistance or supervision by the host computer. Therefore the host computer is free to execute other functions. The host computer, in this case is a C40 DSP. The DSP sends the

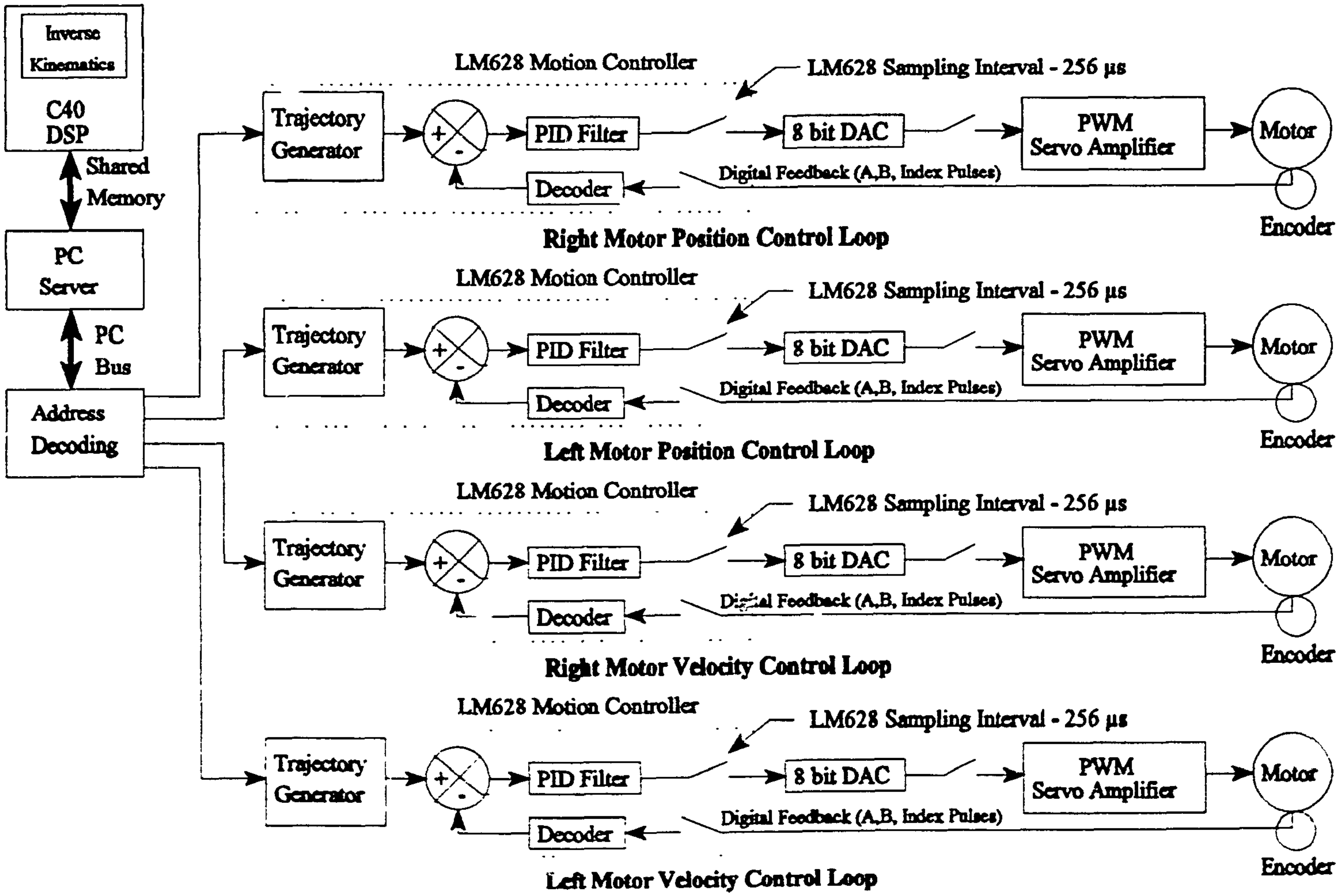


Figure 7.6 Diagram of Closed Loop Motion Control

contro! commands for the LM628 to PC where a server program relays this information to the PC bus where the LM628 interface card receives it and routes it to the appropriate motion controller (refer to Figure 7.6).

There is a sequence of instructions that have to be sent to the motion controller before it can perform any velocity or position control. The motion controller has to be reset to clear any previous settings and to verify that it is functional; in fact it returns a code with a status report. Then the controller is informed of the type of digital to analog converter (DAC) used: either 8 bits or 12 bits. The coefficients of the digital PID filter are sent next. The mode of operation, either velocity or position mode, along with the trajectory data, acceleration, velocity and position (only for position mode), follows. It should be noticed that new values for velocity and position can also be given during motion whereas acceleration cannot be updated on the fly, the motion has to be stopped before. Both the position and velocity values can be relative or absolute. For a more detailed description of the LM628 refer to Appendix B. A block diagram of the various elements present in the LM628 can be seen in Figure B.1 Internally the LM628 contains a communication interface for the host computer, a velocity profile generator, a digital PID filter, feedback encoder pulse decoding circuitry, and output lines for a digital to analog converter (DAC).

In order to send commands to the LM628 the host computer writes to the I/O port. However before writing any command or data, the status of the LM628 needs to be checked to determine whether it is busy to receive. This is done by reading the status byte from the LM628 and verifying the least significant bit or the busy bit. The busy bit goes high as soon as a command is written or data is read or written, and it remains high for at most 100 µs. During this time the LM628 cannot receive or send any information.

After the summing junction a digital PID filter is used as a controller. The equation representing this filter is given by National Semiconductor [7.4] in discrete form as

$$u(n\ \delta t) = K_p e(n\ \delta t) + K_i \sum_{j=0}^{n} e(j\ \delta t) + K_d[e(k\ \delta t) - e(k-1)\delta t] \quad (7.21)$$

where $u(n\delta t)$ is the command that will be sent to the motor and K_p, K_i, K_d, are the gains of the proportional, integral, and derivative terms respectively. The filter gains can be changed even during motion. The sampling time n is $2048/f_{clk}$, where f_{clk} is the LM628 operating frequency which is 6 or 8 MHz. The derivative sampling time interval $(k\ \delta t)$ can be set from $2048/f_{clk}$ to $65536/f_{clk}$ and it does not have to be equal to the controller sampling time. The proportional and integral terms have a fixed sampling time of $2048/f_{clk}$. For CONCIC-4 f_{clk} is set at 8 MHz which yields a sampling time of 256 μs. The position/velocity error term $e(n\delta t)$ is computed from the desired position/velocity and the actual position/velocity as

$$e(n\delta t) = d(n\ \delta t) - a(n\ \delta t) \quad (7.22)$$

at the sampling time $(n\ \delta t)$. The integration sum limit is I_i and it is given by

$$I_i(n\ \delta t) = K_i \sum_{N=0}^{n} e(j\ \delta t) \quad (7.23)$$

This limit is needed to prevent overshooting of the output signal due to the integration term.

The LM628 contains a position feedback processor. It decodes the quadrature A and B pulses from an incremental position encoder and the index (I) pulse. Because of the quadrature pulses, the resolution of the position is four times the number of pulses per revolution of the encoder disk. Therefore, the actual position can be read from the LM628 along with an internally computed velocity. The internal velocity profile generator provides for a trapezoidal velocity profile: an acceleration period, followed by a constant velocity period, and a deceleration period.

The trajectory parameters that instruct the LM628 to move the motor to a new position or to keep the motor turning at a certain velocity, need to be in a position count format. The following conversion equations are used to determine the correct acceleration, velocity and position parameters in LM628 format. All of these parameters are 32 bit values, where position is positive or negative; velocity and acceleration are only positive values. Of the 32 bits, the 16 most significant bits represent the integer part of the value and the 16 least significant bits are the fractional part. For CONCIC-4 the desired steering position is expressed in terms of degrees and the equation to convert degrees to position counts is

$$P_c = \frac{\theta\, s\, n}{360} \qquad (7.24)$$

where P_c stands for position counts which is the value to be sent to the LM628. θ is the desired angular position in degrees as an absolute value from the centre position. The parameter s is the encoder resolution multiplied by four and it is equal to 1024. The gear ratio n is the motor's gear reduction and for the steering motor it is 10. For the steering motors the values of velocity and acceleration are constant and predetermined.

The drive motors receive velocity commands. The system uses the units of cm/s which is a linear velocity. To convert the cm/s into position counts per second, the following equation is used

$$V_{LM} = \frac{S\ (\delta t)\ F\, N\, V}{2\pi r} \qquad (7.25)$$

where V_{LM} is the velocity in LM628 format. S is four times the encoder resolution and it is 4096. δt is the sampling time of the system which is 256 μs. F is used to adjust for the fractional part of the velocity conversion and N is the gear ratio which equals to 24. V is the linear velocity of the vehicle. The conversion for the acceleration command is as follows:

$$A_{LM} = S \, \delta t \, \delta t \, F \, N \, A \tag{7.26}$$

A_{LM} is the acceleration in LM628 format of position counts per second per second. The value A is the linear acceleration. While the LM628 is executing a particular motion, the actual values for the position and velocities can be read back. To obtain the position in degrees and the velocity in cm/s, the same equations can be used but solved for the other parameters.

There are some limitations associated with the LM628. Acceleration cannot be changed on the fly while some motion is being executed. This creates problems when accelerating in a turn because the outer wheel has to reach a higher velocity than the inner one in the same time period, instead both wheels accelerate at the same rate causing wheel slip. This problem can be averted by artificially modifying the velocity commands as to obtain a different acceleration profile for each wheel. Instead of providing the final velocities to each wheel, a ramp of intermediate velocities is given. Another problem encountered is the lack of resolution while reading back the velocity from the LM628. Since the velocity read back is only a 16 bit value there is an error of ± 2.5 cm/s. This can be solved by reading the position values, which are 32 bit values, and differentiating them as described by Rajagopalan [7.1].

7.6 Variable Position and Orientation Gains

Referring back to the equation for the control law,

$$\theta_s = G_1\ \varepsilon_d + G_2\ \varepsilon_\theta \qquad (7.4)$$

the steering angle θ_s is computed from the sum of the position offset ε_d and the orientation offset ε_θ. The gain coefficients G_1 and G_2 are determined experimentally to allow for stable control of the AGV in various conditions. Initially, fixed values were tried. It has been established experimentally that a set of values that would work well for straight line tracking at low speed (70 cm/s) would not result in a stable performance for higher speeds. Experience has shown that the position offset gain G_1 has to be reduced with increasing travelling speed. This makes sense as a car driver, when driving on a highway at higher speeds he/she makes less adjustments for slight offsets in position. Therefore, the driver effectively reduces his sensitivity and makes less corrections for small position offsets. When driving at slower speeds, accuracy is more important and the position offset gain can be increased for better road following. It has been shown experimentally that too high of a gain for G_1 at any particular speed leads to oscillatory behaviour, and too low of a gain causes large drifts and sluggish response. This is illustrated in Chapter 9 in Figures 9.6 and 9.7.

While experimenting with curved roads, another interesting behaviour has been observed. The angular offset gain G_2 needs to be adjusted as well, depending on the radius of curvature of the turn. The tighter the turn, the higher the value for G_2 has to be. Again, if G_2 is too high then oscillatory behaviour can be observed while turning. If G_2 is too low then the vehicle does not turn enough and the road is lost from the camera view. Figure 7.7 represents graphically some aspects of the above discussion.

Hence the original control equation has been modified to a'iow the AGV to perform

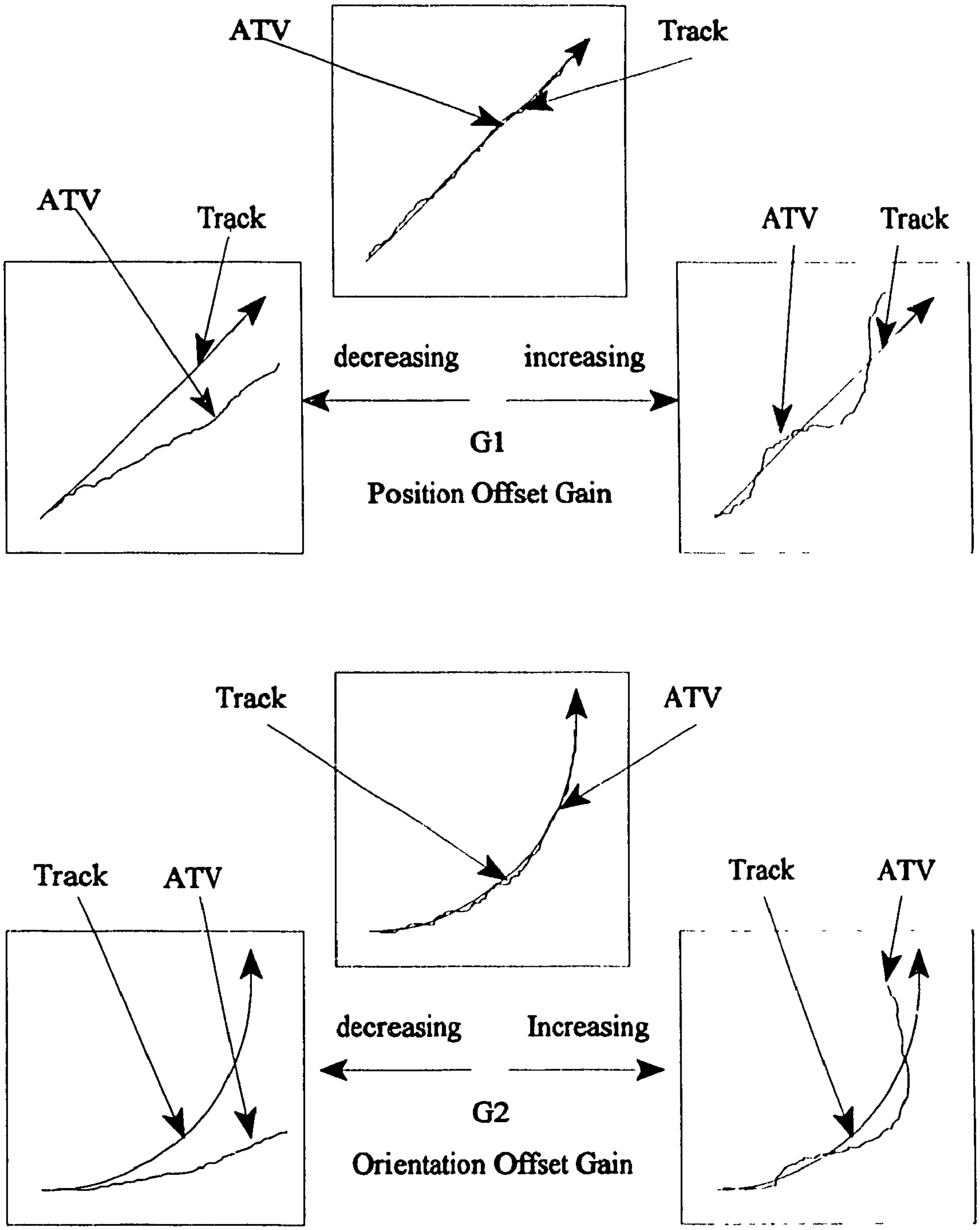


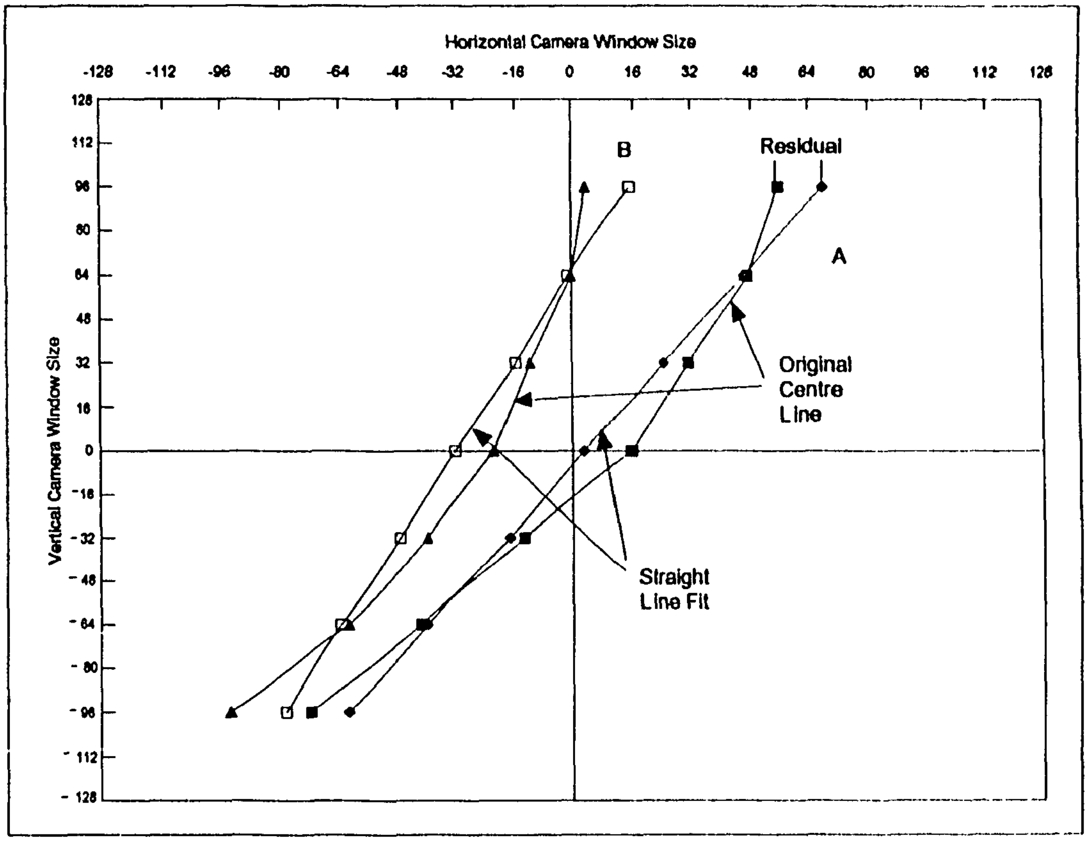
Figure 7.7 Schematic Showing the Effects of Position Gain G1 and Angular Gain G2

well at high or low speeds and in straight or curved roads. The control law has the following form

$$\theta_s = G_1(V)\ \varepsilon_d + G_2(R)\varepsilon_\theta \qquad (7.27)$$

where G_1 is a function of the vehicle's speed, G_2 is a function of the road geometry, V is the linear forward velocity of the vehicle, and R is the turning radius. The function that relates the position offset gain and the vehicle's speed was found experimentally. The function relating the angular offset gain and the road geometry was also found experimentally. These relations, along with the performance of the AGV using this control law, can be found in Chapter 9.

Relating the position offset gain to the vehicle speed is simple because the AGV's velocity is known at all times. On the other hand, the road geometry needs to be identified before the angular offset gain can be related to it. The actual shape of the road is not relevant; what is important is the variation of the road from a perfect straight line. The more the variation, the more the angular offset gain has to be increased. The centre line extraction routine provides the coordinates of the centre line. Then a straight line is fitted through the centres. In order to find if the road is straight or not, the difference between the points on the straight line fitted and the original centre line coordinates is computed. These differences are called *residuals*. Then, by summing the square of the residuals, an indicator of how far the original centre line is from a perfect straight line can be computed. This can be seen in Figure 7.8. The sum of the residual squared increases as the original centre line gets more curved. Therefore, the sum of the residual squared is used as the indicator to relate the angular offset gain G_2 and the road geometry. These equations are derived and shown in section 9.6.



Actual Curved Track A Coordinates			Computed Straight Line	Residual	Residual ^{2}
-72	-96		-62	-10	100
-42	-64	Straight Line Fit Computed	-40	-2	4
-14	-32		-18	4	16
15	0	F(x)=0.667x+3	3	12	144
31	32		24	7	49
47	64		46	1	1
56	96		68	-12	144
			Sum of Residuals squared	------▶	458

Actual Curved Track B Coordinates			Computed Straight Line	Residual	Residual ^{2}
-64	-96		-79	15	225
-62	-64	Straight Line Fit Computed	-64	2	4
-40	-32		-48	8	64
-22	0	F(x)=0.493x-32.3	-32	10	100
-12	32		-16	4	16
0	64		-1	-1	1
4	96		15	11	121
			Sum of Residuals squared	--- ---▶	531

The more the track is curved the higher the value for the sum of residuals squared is
Track B has more of a curvature than Track A and the sum of residuals squared is higher

Figure 7.8 Estimation of Road Geometry Using the Sum of Residuals Squared

7.7 Summary

In this chapter the guidance control scheme of CONCIC-4 is presented. The definitions for position and angular offsets are described. These offsets are used in the guidance control loop. The camera images provide the feedback in terms of distance and orientation of the vehicle on the road. Once a steering angle is computed, it has to be decomposed into two steering angles: one for the left wheel and one for the right one. Also the individual wheel speeds have to be computed, because the inner wheel in a turn turns slower than the outer wheel. Once the steering angles and wheel speeds are computed, then these values are passed on the LM628 motion controllers. These controllers, one for each motor, perform the closed loop position or velocity control.

Experimentally it has been found that the gains for the position and angular offsets had to vary with speed and road geometry. The higher the speed, the lower the position offset gain. The sharper the turn, the higher the angular offset gain has to be. Therefore, the position offset gain is inversely proportional to speed, whereas the angular offset gain is proportional to the sharpness of the turn. The sharpness of a turn is estimated using the sum squared of the differences between the straight line fitted and the original centre line.

CHAPTER 8

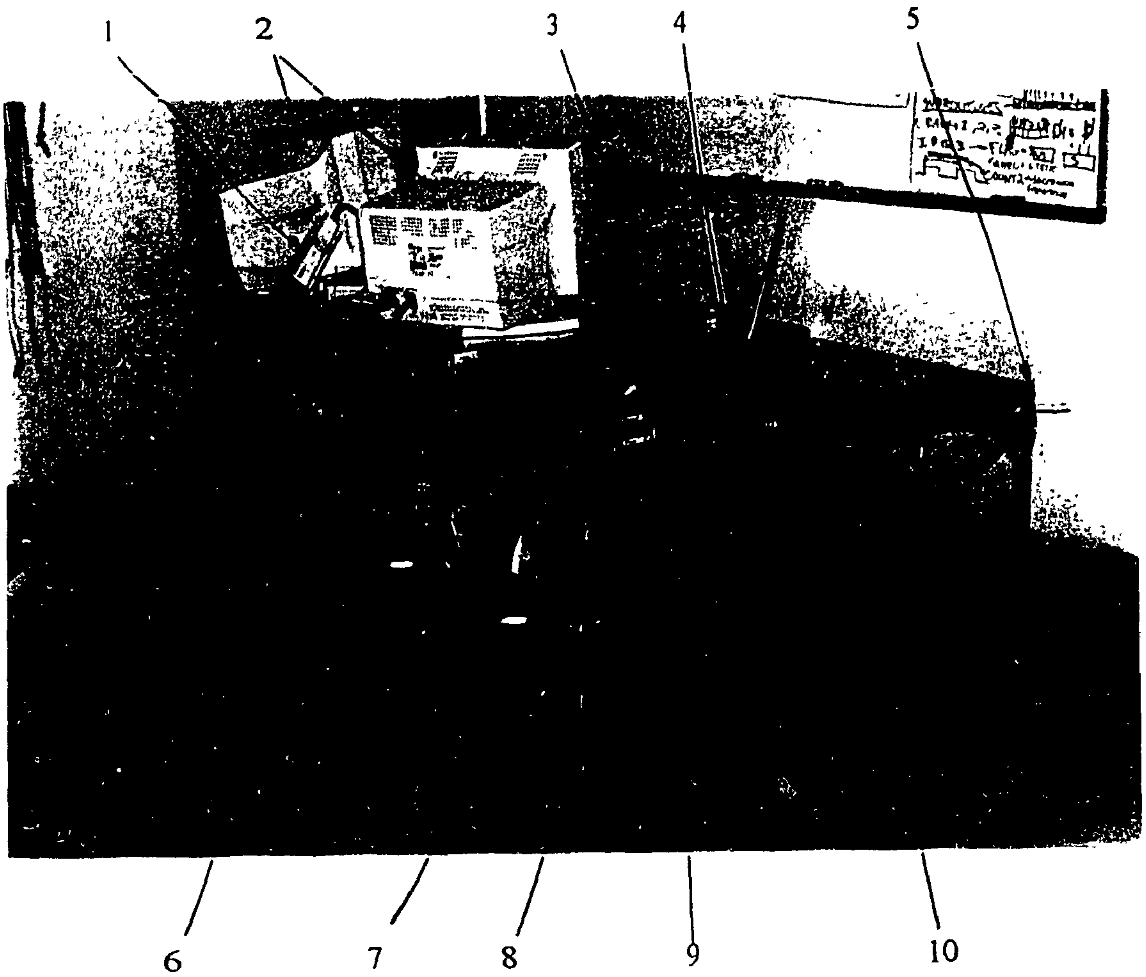
DESCRIPTION OF AN EXPERIMENTAL VEHICLE

8.1 Introduction

This chapter describes the various components that make up CONCIC-4. CONCIC-4 is an experimental vehicle built to implement and test the CCD camera vision guidance for road following discussed in the previous chapters. The mechanical design, the electronics, the power source, the guidance control system, the data acquisition, and the safety features are all discussed in some detail. The mechanical design of CONCIC-4 was already developed and the chassis already built by M. Pharand and J. Campanelli under the supervision of Drs. Rajagopalan and Cheng. The drive and steer motor sets, as well as the C40 parallel processors, were also already selected by the researcher then involved in the developement of the prototype vehicle. All the other systems were designed and added to the original chassis design to produce a fully functional automated vehicle. Since there is no documentation available on the architecture of this vehicle, this chapter presents the various components of the vehicle and the interfaces designed for this thesis work.

8.2 Components of CONCIC-4

Figure 8.1 is a picture of CONCIC-4 that shows most of its main components. The structure consists of an aluminium frame. This frame supports and mounts all other components, notably the motor assemblies in the front, the casters in the back, the batteries in the middle, the computers and monitors on top, and the power amplifiers to drive the motors besides the computers. Last but not least, the CCD video camera is attached on the



1. CCD Video Camera
2. Monitors
3. Safety Switches
4. RF Remote Kill Switch
5 Chassis
6. Computers
7. Servo Amplifiers
8. Wheel Motor Set
9. Batteries
10. Caster

Figure 8.1 Picture of CONCIC-4 ATV

front of the AGV looking down at the road. There are also other less noticeable items, such as safety features, control switches, relays, wires, and the operating system software that are necessary for the functioning of CONCIC-4. In the following sections, a detailed description of each component and its function is given.

8.2.1 Structure

CONCIC-4's chassis is made up of welded square cross-section aluminium tubing. The chassis is reinforced with cross-braces on all sides, which make the structure very rigid. Aluminium sheets are used to mount most other components. There are four main supports where the motor assemblies in the front and the casters in the rear are attached. It is through these supports that the weight of the vehicle and its cargo is passed to the wheels and eventually to the road. In the middle of the vehicle there is a battery bay, where up to six batteries can be placed. The batteries contribute to a large portion of the overall vehicle weight. In the front of the vehicle there are two levels, the lower one housing the computing unit and the servo amplifiers and the upper one supporting the monitors and keyboard The CCD camera is located at the front top of the vehicle and it is mounted on a swivelling base so that the camera view can be adjusted. Figure 8.2 is a sketch detailing the most important dimensions of CONCIC-4. The overall length of the vehicle is 166 cm, the width is 97 cm, and the height without the monitors is 90 cm. The ground clearance is 10 cm. The wheel base, which is the distance between the front and rear wheels, is 101 cm. The distance between the left and right wheel is 46 cm, which is also called the wheel span. The total weight of CONCIC-4 with all the components mounted is estimated at 700 lbs. The drive wheels are integrated wheel sets from Schabmuller Corp. specifically designed for automated

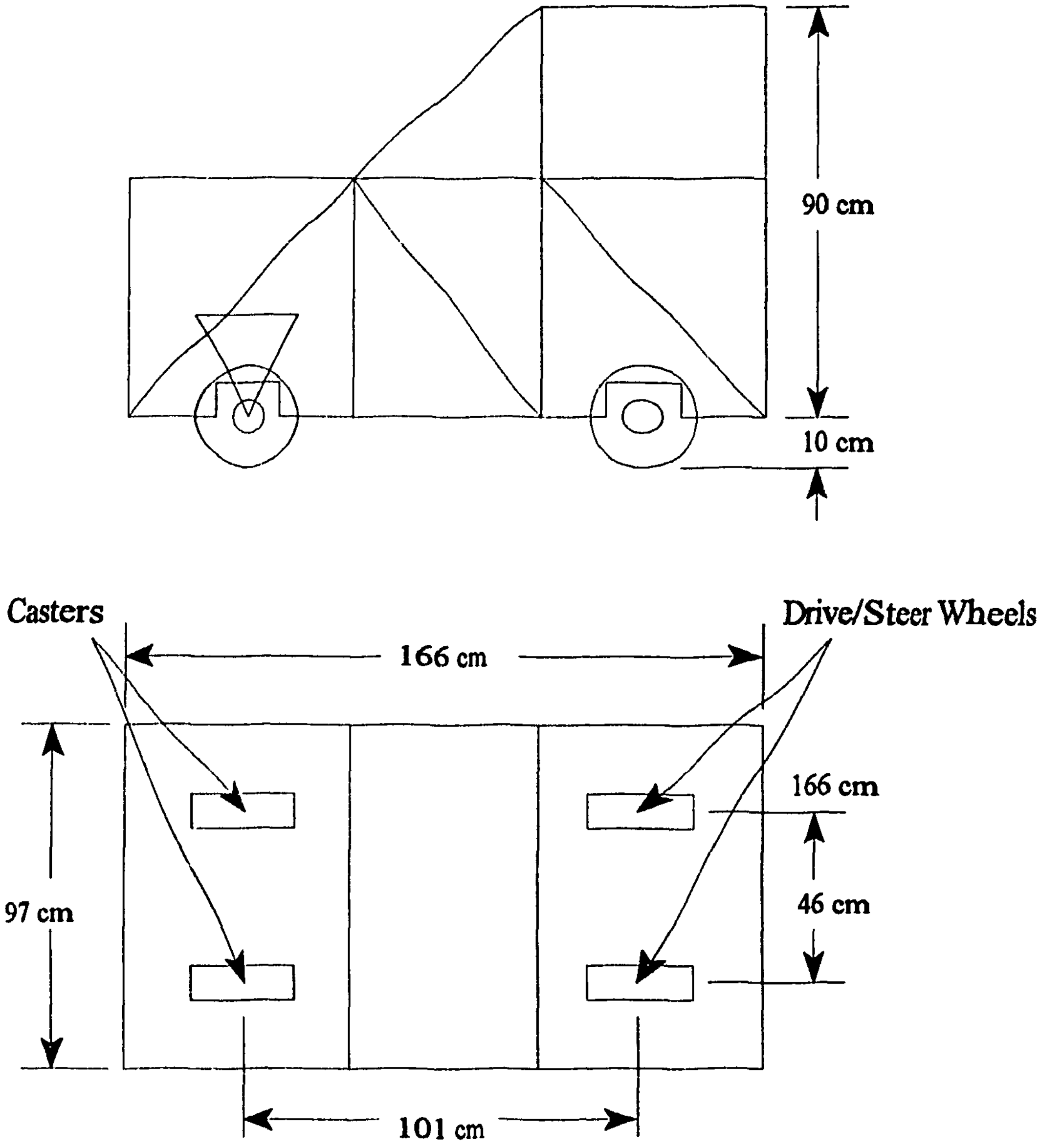


Figure 8.2 Sketch of CONCIC-4 With Overall Dimensions

vehicles.

8.2.2 Control Components

The control system for CONCIC-4 is based on three interrelated sub-systems: a 486 66MHz computer, a five C40 DSP network, and four LM628 precision motion controllers. The 486 hosts the two other sub-systems through the ISA bus. The C40 network performs most of the system controls: it digitizes the images from the CCD camera, it processes the images to reduce noise and identifies the location of the centre line of the road, it computes the guidance control loop, it communicates with the 486 and the LM628s. The LM628s, as described in the previous chapter, perform the closed loop motion control of the drive and steer motors. The 486 is used as user interface allowing for keyboard input, displaying of information, and hard drive access. For more details on the hardware setup refer to chapter 4 in section 4.2 of hardware implementation.

8.2.3 Servo Amplifiers

Four amplifiers are utilized in driving the motors. All made from the same manufacturer Advanced Motion Controls two different models are used. The main drive motors are power by the 100 Series capable of delivering up to 100A. The steer motors require less current and the 50 Series is used still capable of providing 50A. These amplifiers all share the same design. They are pulse width modulation (PWM) amplifiers switching at a frequency of 33 KHZ. The output stage uses MOSFET devices and it is configured as H-type, allowing for bi-directional motor rotation. The power input voltage has to be in the range of 30 to 200 volts: four batteries in series provide 48 volts. The control input is an analog voltage of ± 5 volts which yields a maximum output voltage of about 44 volts.

Each amplifier is equipped with two inhibit inputs so that the motors can be stopped from turning in either direction. This is useful for the steering motors, where limit switches can switch off the amplifier in the two directions. The continuous and peak currents can be limited by adjusting a potentiometer. A fault led lights up if there are any problems with the amplifier.

8.2.4 Guidance System

The guidance system is based on visual recognition of the road. A CCD camera sends images of the road ahead to a frame grabber board that digitizes the images. These images are then processed for background noise removal. Next, the road centre coordinates are extracted and the position and orientation of the vehicle, with respect to the road, are computed. This visual feedback is used to steer the vehicle in order to reduce the deviation from the correct course.

The CCD camera provides standard NTSC interlaced video at 30 frames per second. The frame grabber can digitize images up to 1024x1024 pixels in full 24 bits colour. In this case the images captured are 512x512 in 256 grey scale tones (8 bits). The images before being processed are further reduced to 256x256 pixels. The overall system allows for a frame rate of 15 images per second.

The camera's view is located in the front of the vehicle. It is of trapezoidal shape, due to the fact that the camera is mounted at an angle with respect to the ground. Figure 8.3 highlights the various dimensions involved. The view of the camera starts about 82 cm from the front wheels and extends for 75 cm. The width at the top is 90 cm and at the bottom is

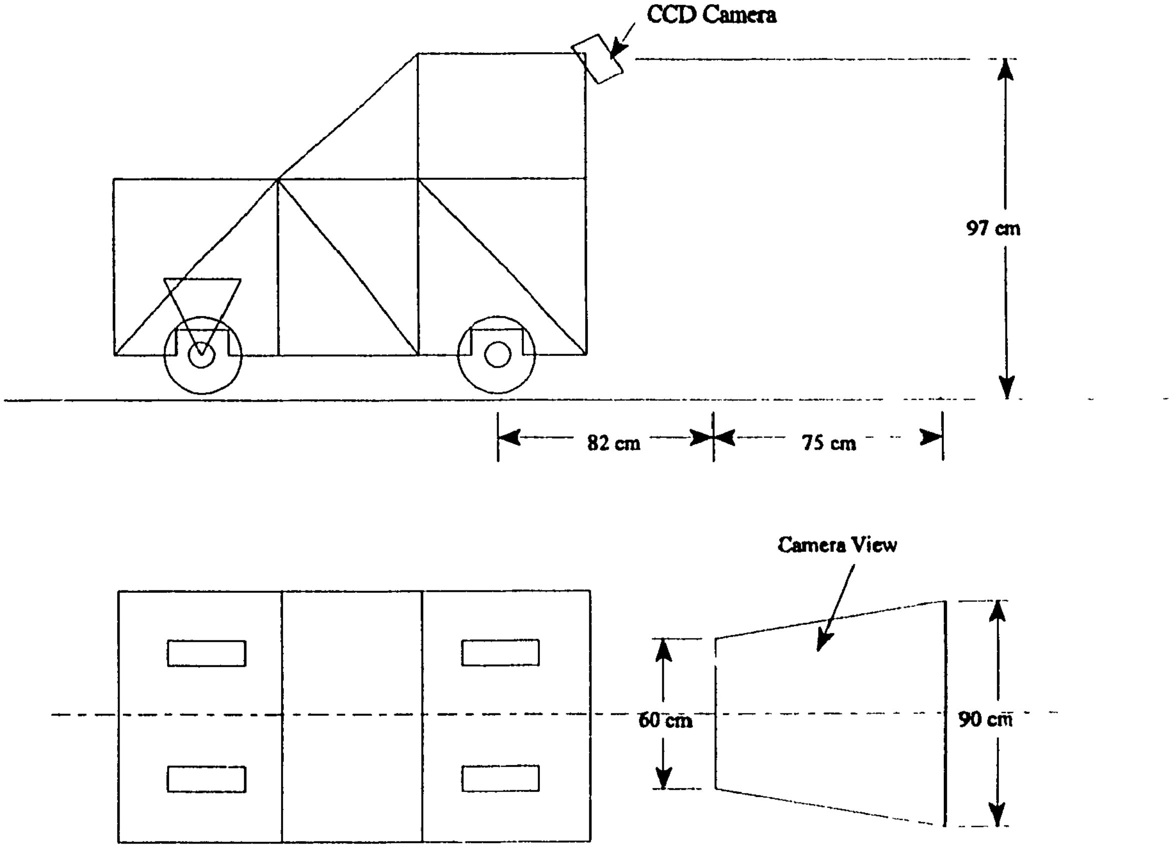


Figure 8.3 CCD Camera Location and View

62 cm. The camera is angled at approximately 65 degrees with the horizon. If the camera is tilted less, then it can view more road; however the perspective effects will be more pronounced and the resolution will also decrease. If the camera points straight down the field of view will be reduced and the maximum speed could not be as high. The camera's view can also be increased or decreased by adjusting the zoom on the lens. The same principle applies: with a shorter view, there is more resolution and detail but lower maximum speed, with a longer view the maximum speed is higher but the resolution is decreased.

8.2.5 Power System

The power source for CONCIC-4 is a set of six 12 volt deep cycle batteries. They are two main power systems: one working at 48 volts and another at 12 volts. Figure 8.4 provides a schematic of the power system. The motors are powered with four batteries in series at 48 volts through the amplifiers control. The 12 volt system, consisting of two batteries in parallel, is used for powering the lights and a power inverter. The inverter converts the 12 volts DC to 120 volts AC to power the computer and the two monitors. The inverter can provide up to 550 watts, which is enough for the 250 watts computer and the 150 watts for the monitors. The lights are three 40 watt halogen bulbs.

During the testing stages of CONCIC-4 it has been observed that the batteries can provide power continuously for more than two hours. The largest consumption is by the drive motors when running at high speed, heavy load, or demanding large accelerations. The motor assemblies are made up of two motors: a larger one for driving and a smaller one for steering. The drive motor is a 500 watts permanent magnet DC motor running at 24 volts. The steer motor is a smaller version rated at 125 watts from a 24 volt power source. In order

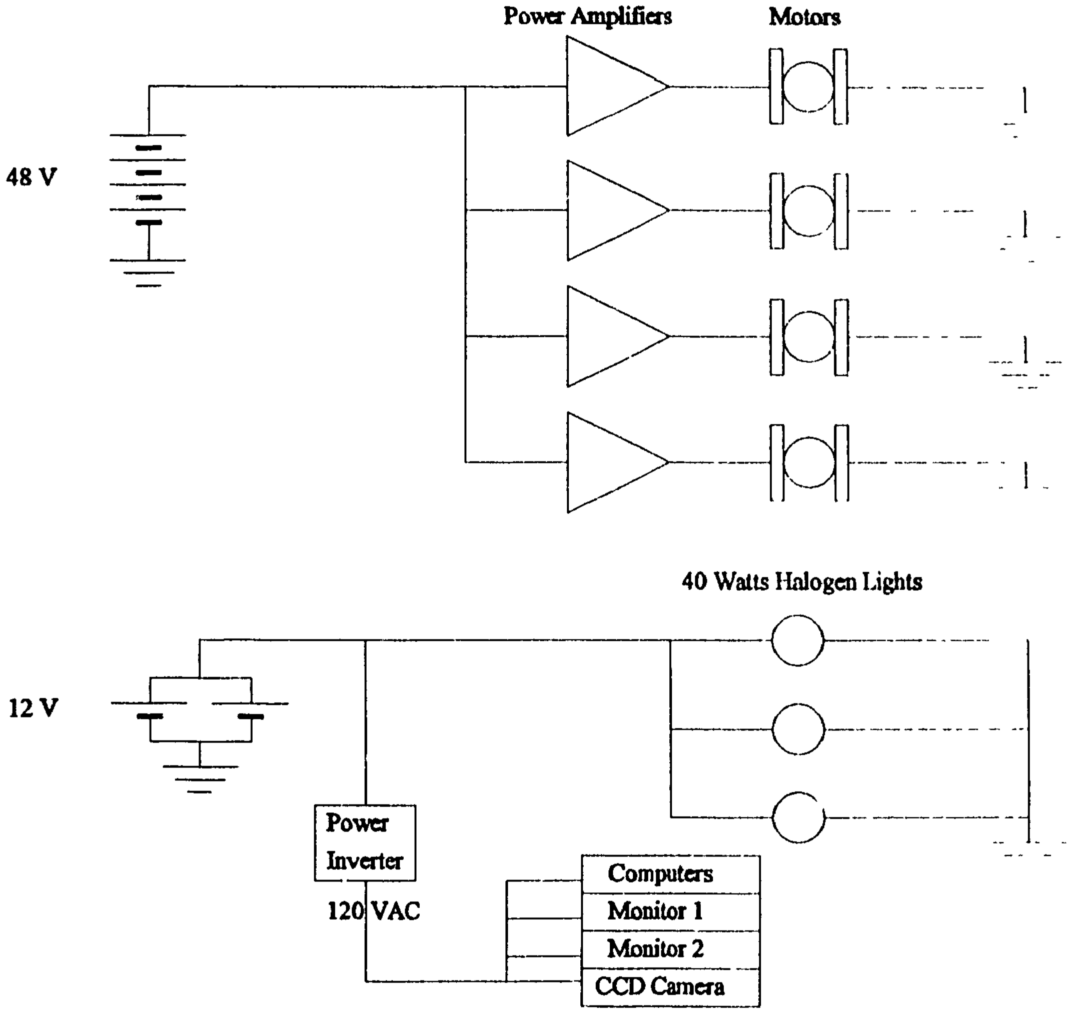


Figure 8.4 Electrical Power Systems

to achieve higher speeds the motors are run at 48 volts. With the increased voltage, the drive motors can run the AGV at 2.6 m/s instead of the rated 1.0 m/s. In the long run, the motors can get damaged from the excessive power dissipation, however for testing purposes it should not be a problem.

8.2.6 Data Acquisition

The data acquisition is comprised of 11 software channels. It is executed within the C40 network, and the operating system software can support many more software channels if more data needs to be monitored. The data acquisition is necessary in order to study the behaviour of the vehicle while actively performing road following. The following data are recorded:

- vehicle's linear speed
- vehicle's steering angle
- desired wheel speeds
- actual wheel speeds
- steering angles
- position and orientation offsets
- centre of the road coordinates

These data are stored in memory every time the main control loop is executed, that is every 67 ms. The user can select the number of samples to record, as well as starting and stopping the data acquisition any time on demand. The C40 processor that communicates with the host PC is the one that carries out the data acquisition. Once the vehicle has stopped, the data is send to the PC to be saved to a file on the hard disk, where it can be retrieved later and analyzed.

8.2.7 Safety Features

There are some safety features implemented both in hardware and software. The hardware safeties include an RF remote control motor stopper, and two emergency kill switches on the vehicle. Both the RF remote control and one of the emergency kill switches act on the inhibit input of the servo amplifiers. By inhibiting the inputs, the amplifiers give zero output, effectively stopping the motors from turning. The other emergency kill switch has two functions, it powers down the amplifiers, but it also disengages some relays that disconnect the motors from the amplifiers so that no back e.m.f. will be sent to the amplifiers by the motors still turning. This back e.m.f. can severely damage the amplifiers.

The software safety features include detection of motion controllers malfunctioning, stopping of vehicle when no road is detected, and stopping of vehicle using several keys on the keyboard. The LM628 motion controllers are responsible for performing closed loop control of the motors. If one of the LM628s stops working, it can be detected because the status byte would have a wrong value. The system can then be stopped and a message appears to the user to be aware of the problem. The vision system also stops the vehicle after not detecting the road for four frames. The user can stop the system by pressing one of the following keys: the 'ESC' key, the 'S' key, the 'spacebar', and the 'enter' key. In all cases the vehicle stops quite rapidly.

8.3 Summary

In this chapter the design of CONCIC-4 is presented. The test vehicle is made up of an aluminium frame where all the other components are mounted. The power to the vehicle is provided by six batteries that allow for two hours of testing. The vehicle is guided by a vision system based on a CCD camera and a five C40 DSP network. Motion controllers perform the low level closed loop motor control supervised by the C40 network. The motion controllers feed the servo amplifiers that provide the power to the motors. The behaviour of the vehicle can be studied using the data acquisition system. Several basic safety features are implemented both in hardware and software. The vehicle can be stopped by an RF remote control, by direct emergency kill switches or by software control when there is no road or the stop key on the keyboard has been pressed. It should be reminded that this is just a test vehicle and more safety features would have to be implemented for a more 'commercial' version of this vehicle.

CHAPTER 9

VALIDATION OF GUIDANCE CONTROL BY EXPERIMENTATION

9.1 Introduction

This chapter presents the experimental results obtained while testing the vehicle under various operating conditions. Tests are performed on a straight track, on a straight track with an offset, and on curved tracks. Operating parameters are chosen after extensive testing of the vehicle in each of these conditions. It is found that on a mixed track, made up of straight and curved sections, some operating parameters need to be varied dynamically according to the track profile.

The chapter is organized as follows: the calibration of the PID filters for the motion controllers is first presented, then results of straight track and offset track following are shown, along with the tuning of the position and orientation gains, next the results for curved tracks are discussed, the need for dynamic adjustment of position and orientation gains is d^escribed for mixed tracks and finally, the results of high speed tests are shown.

9.2 Calibration of PID Filters of Motion Controllers

For accurate closed loop control of the motors, the LM628 motion controllers need to have the correct values of the PID filters. These filters determine the response of the motor given a command, which could be a velocity or a position command. The response of a step input is analyzed to identify what type of response the motor gives: the underdamped response produces quicker but more oscillatory behaviour and larger overshoot, the overdamped has no overshoot, no oscillations but slower response, and the critically damped has no overshoot and no oscillations with good response time [9.1]. In the case of a servo motor system, the response can be a bit underdamped so that faster action can be achieved [9.2].

The loop compensation filters of a PID controller are usually tuned experimentally, especially if the system dynamics are not well known [9.3]. An experimental tuning procedure is described by National Semiconductor in the LM628 Programming Guide [9.3]. There are five parameters that need to be determined to tune the system: the three gain coefficients, K_p, K_i, and K_d, the integration limit coefficient i_l, and the derivative sampling coefficient d_s. In this case the derivative sampling coefficient d_s is chosen to be the smallest possible value 256 μs. The derivative sampling coefficient should be at least 10 times smaller than the mechanical time constant of the system. Therefore, four parameters have to be systematically varied until reasonably good response characteristics are obtained.

The value for K_p is initially set to 1 and the coefficient K_d is varied. After extensive testing, a value of 800 for K_d was found to give good response: no oscillations, fast response, some overshoot. Then K_p is varied; however any value above 1 caused oscillations and ringing in the motor, even when standing still. The integral coefficient K_i is then varied along with the integral limit i_l. It was found that by using no integral action, better results can be obtained. The integral action caused oscillatory behaviour. Figure 9.1 summarizes some of the test results while tuning the PID filters of the motion controllers for the drive motors. In Figure 9.1, just the shape of the curves should be observed; the actual units of the velocity and time are irrelevant for this discussion. A step input was used to monitor the motor's response. The final parameters for the drive motors are: K_p =1, K_i =0, K_d =800, i_l =0, d_s =256 μs. The response to various step inputs using these parameters can be seen in Figure 9.2 .

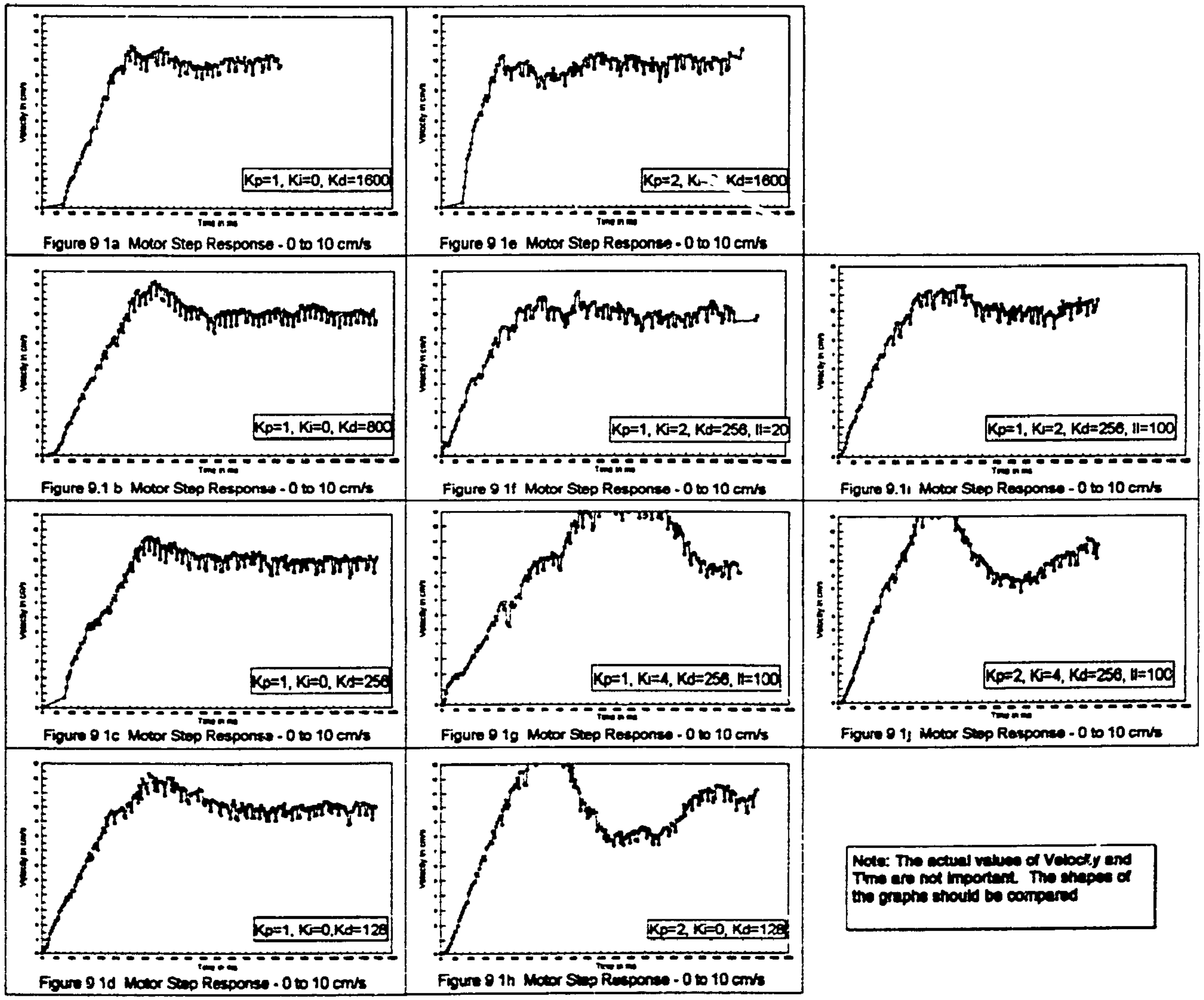


Figure 9 1a Motor Step Response - 0 to 10 cm/s

Figure 9.1 b Motor Step Response - 0 to 10 cm/s

Figure 9 1c Motor Step Response - 0 to 10 cm/s

Figure 9 1d Motor Step Response - 0 to 10 cm/s

Figure 9 1e Motor Step Response - 0 to 10 cm/s

Figure 9 1f Motor Step Response - 0 to 10 cm/s

Figure 9 1g Motor Step Response - 0 to 10 cm/s

Figure 9 1h Motor Step Response - 0 to 10 cm/s

Figure 9.1i Motor Step Response - 0 to 10 cm/s

Figure 9 1j Motor Step Response - 0 to 10 cm/s

Note: The actual values of Velocity and Time are not important. The shapes of the graphs should be compared

Figure 9.1 Tuning of PD Coefficients for Drive Motors

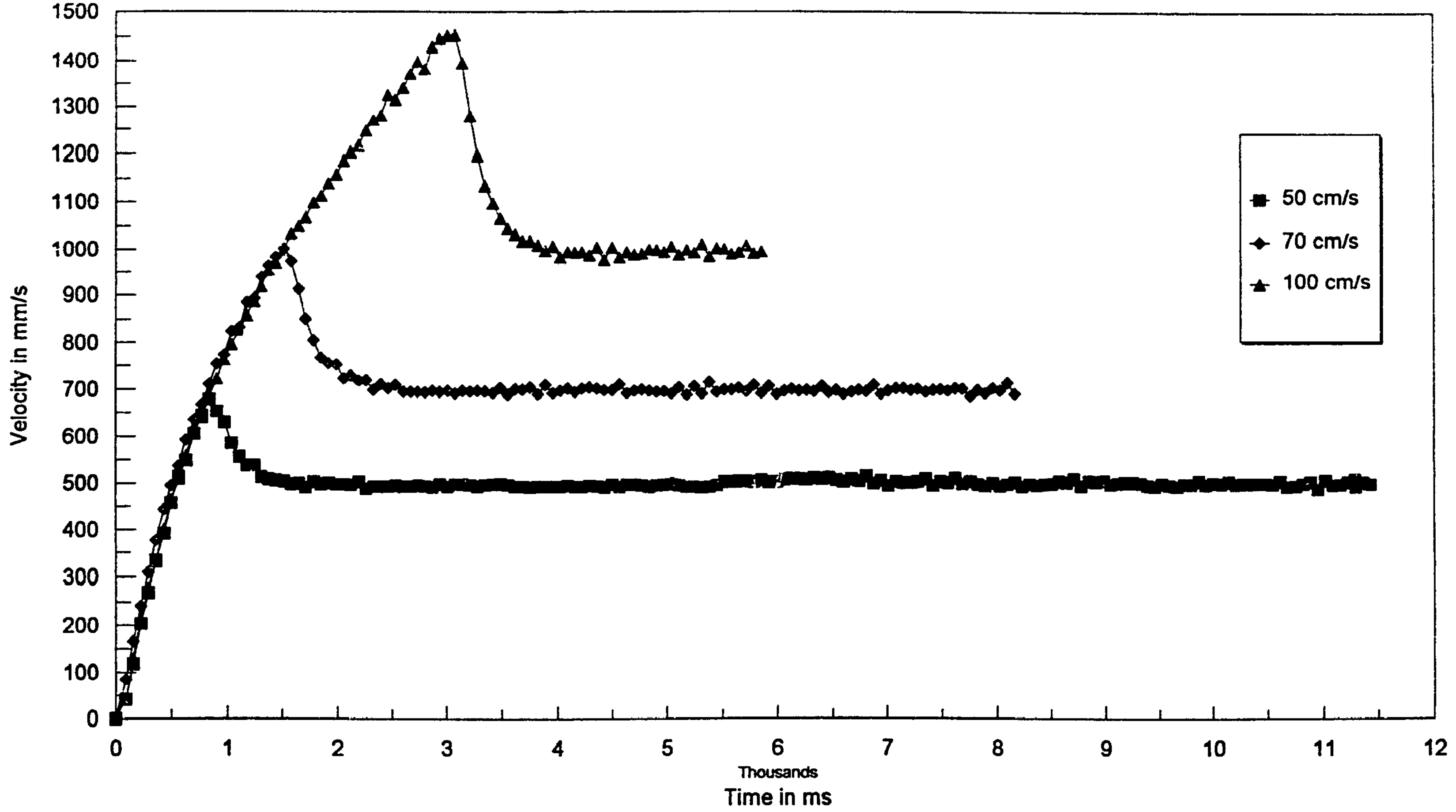


Figure 9.2 Step Input Motor Response Kp=1, Ki=0, Kd=800

In all cases an overshoot is present but the steady state condition is reached right after, without oscillations and noticeable offsets. The steer motors are tuned similarly and the final parameters are chosen as: K_p=35, K_i =100, K_d =10, i_l =100, d_s =256 μs.

9.3 Track Following Performance for Straight Tracks

The first set of tests are performed on a straight track. The vehicle speed is maintained low, 30-70 cm/s, so that the position and orientation offset gain coefficients could be experimentally approximated. After the range has been defined more tests are performed. Figures 9.3a and 9.3b show the tracking performance for a straight line track at 70 cm/s and 100 cm/s vehicle speed respectively. The position offset is within ± 2 cm. The orientation offset also remains within a range of ± 5 degrees. To reach the speed of 70 cm/s the vehicle needs about 2 s, whereas to reach 100 cm/s it requires 3.5 s. In Figure 9.4 the tracking performance for a straight line track at 240 cm/s can be seen. The position offset is maintained within a narrow range of ± 4 cm. The orientation offset also remains within a range of ± 7 degrees. The vehicle requires about 6 s to reach the speed of 240 cm/s. It should be noted that the position gain coefficient for the test at 240 cm/s is not the same as for the tests at 70 or 100 cm/s. The position gain coefficient has to be reduced as the speed is increased. A detailed discussion of the position gain and orientation gain coefficients can be found in section 9.4.1.

9.4 Dynamic Response for Sudden Lane Change

In order to study the robustness of the guidance control of the vehicle, a sudden change in the location of the geometry of the track is introduced. The straight line track is

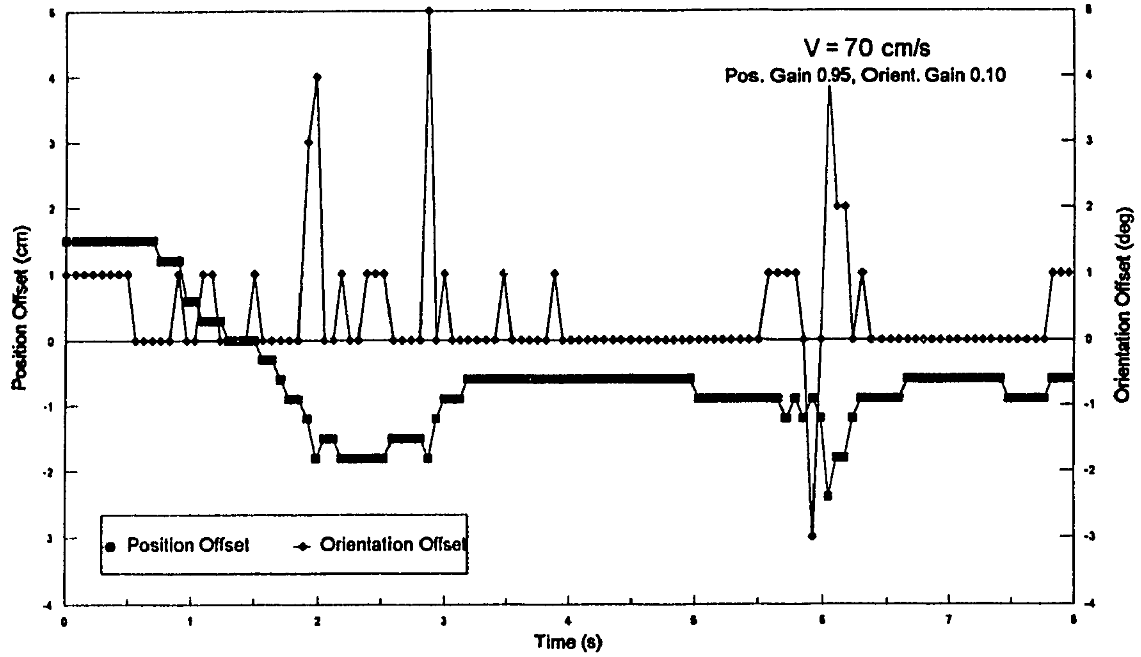


Figure 9.3a Tracking Performance for a Straight Line Track at 70 cm/s

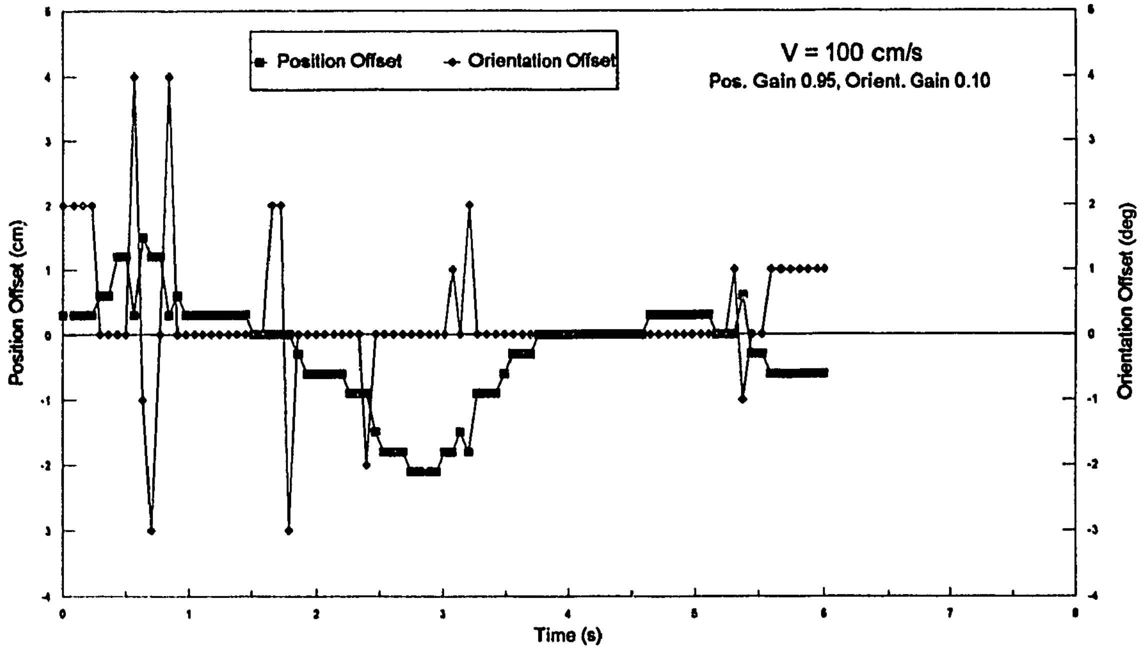


Figure 9.3b Tracking Performance for a Straight Line Track at 100 cm/s

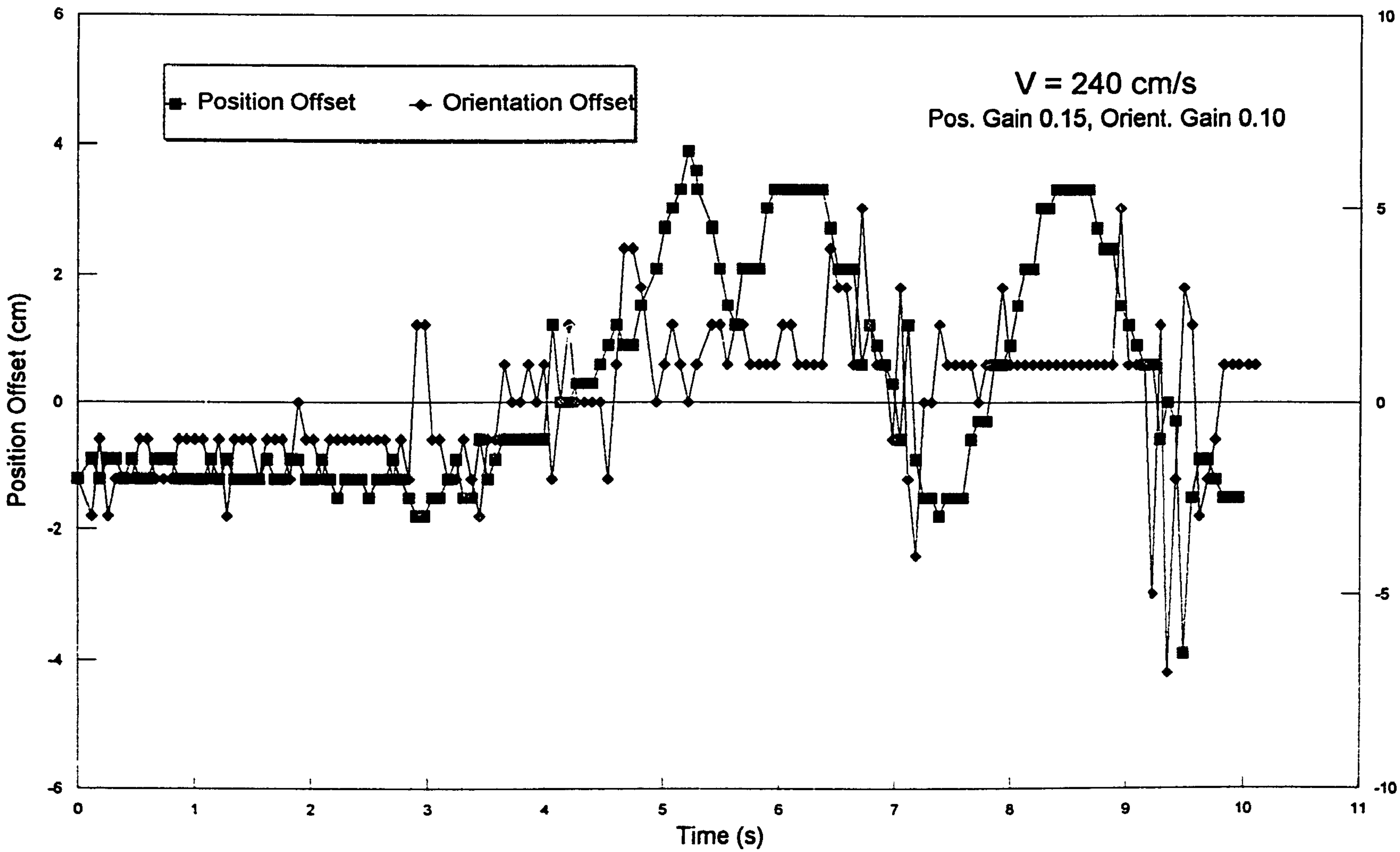


Figure 9.4 Tracking Performance for a Straight Line Track at 240 cm/s

offset by a distance of 39 5 cm. This lane change occurs after the vehicle has reached its nominal test velocity. This kind of test allows for the observation of the dynamic response of the vehicle. In particular, various position and orientation gain coefficients can be evaluated and the optimal ones selected.

Figures 9.5a-9.5d show the behaviour of the vehicle as it changes lanes at various speeds. In all tests the vehicle started from the same initial position and stopped at the same distance where the track ends. It should be noticed that at lower speeds the lane change is more gradual since the new track position gradually enters the camera view and the vehicle starts to compensate for the offset. This can be seen particularly in Figure 9.5a after the time from 6 s to 8 s. At 8 s the old track position disappears from the camera view and the vehicle adjusts for the new track with a large steering command that overshoots the track. Then the vehicle reduces the position offset and returns to following the track. At higher speed (refer to Figure 9.5d), the jump to the new track is more pronounced since the transition from the old to the new track is much faster. As a result, during the compensation stage, the vehicle oversteers past the new track and then it has to hunt for the track. In all cases the vehicle is able to follow the new track without stopping. In figure 9.5d the graph stops after the 6 s because the space at the test site is limited, the vehicle would have resumed following the track with a small position offset like in the other cases.

9.4.1 Tuning of Position and Orientation Gain Coefficients

It is found during the tests that the position offset gain has to be changed according to the velocity the vehicle is travelling at. This can be seen for the graphs in Figures 9.6 and 9.7 Starting with Figure 9.6, the position offset is plotted for various values of the position

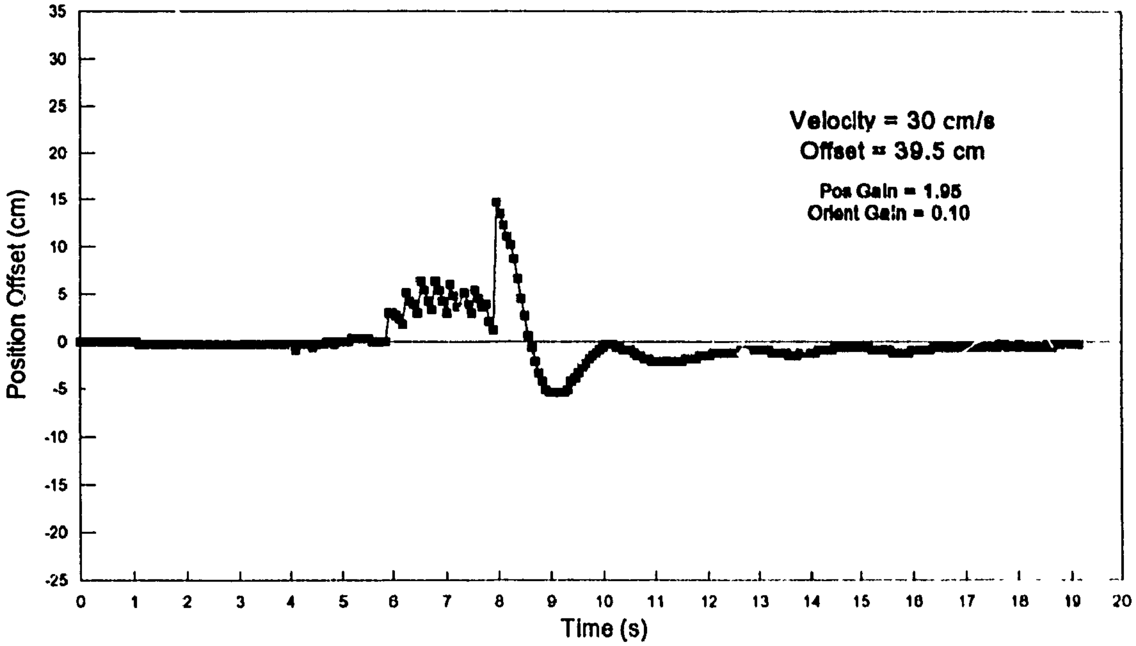


Figure 9.5a ATV Dynamic Response to Straight Line Track With Offset

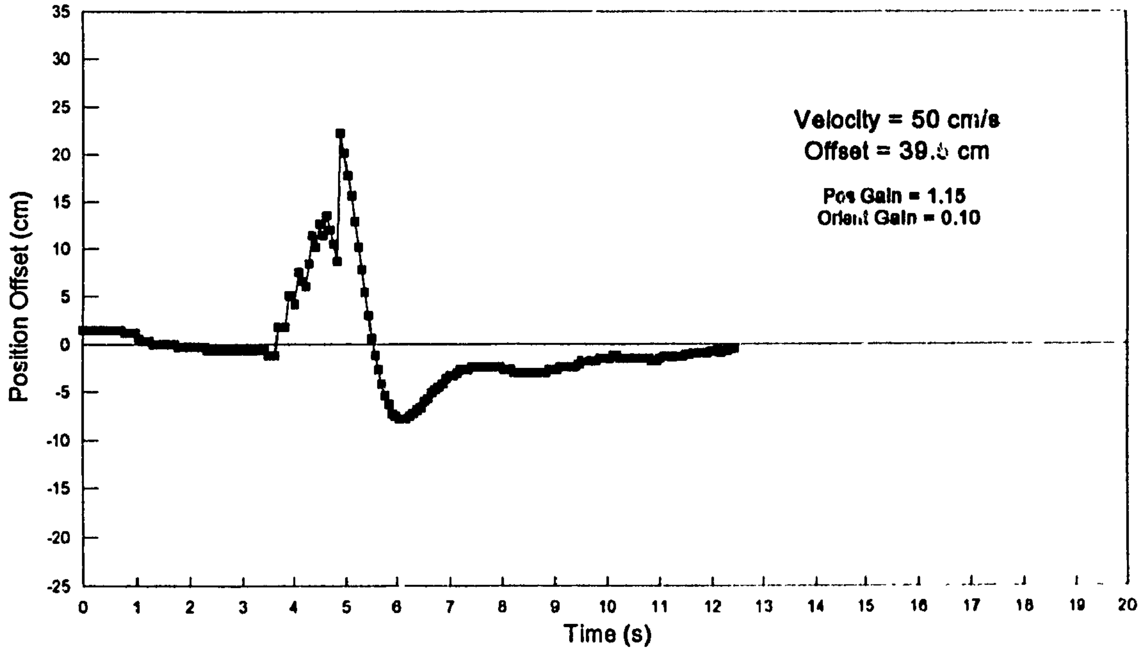


Figure 9.5b ATV Dynamic Response to Straight Line Track With Offset

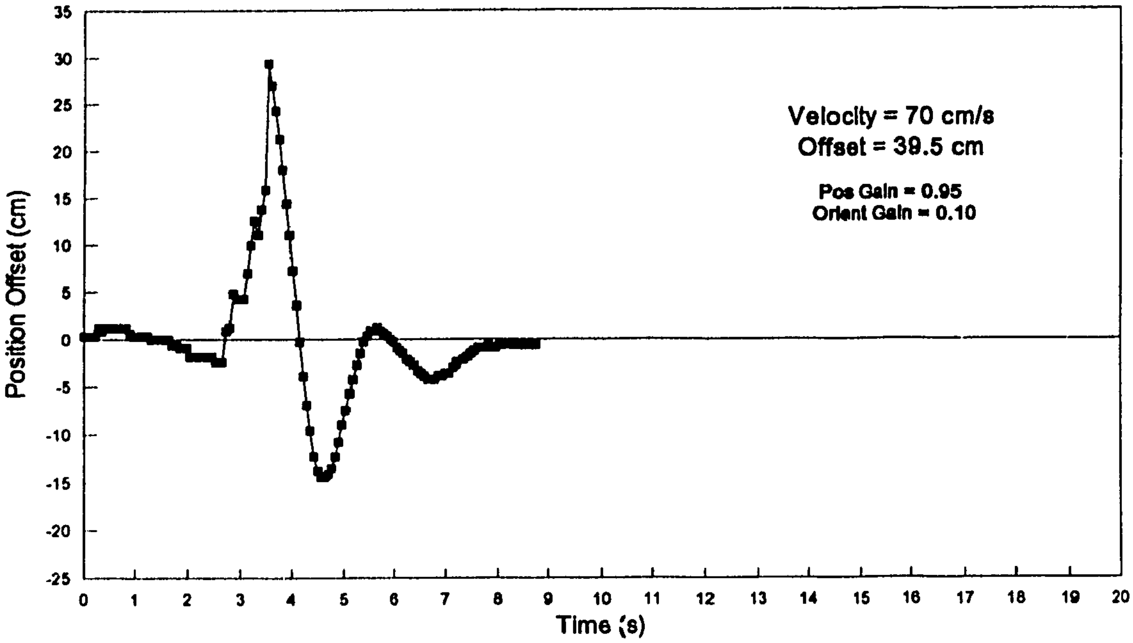


Figure 9.5c ATV Dynamic Response to Straight Line Track With Offset

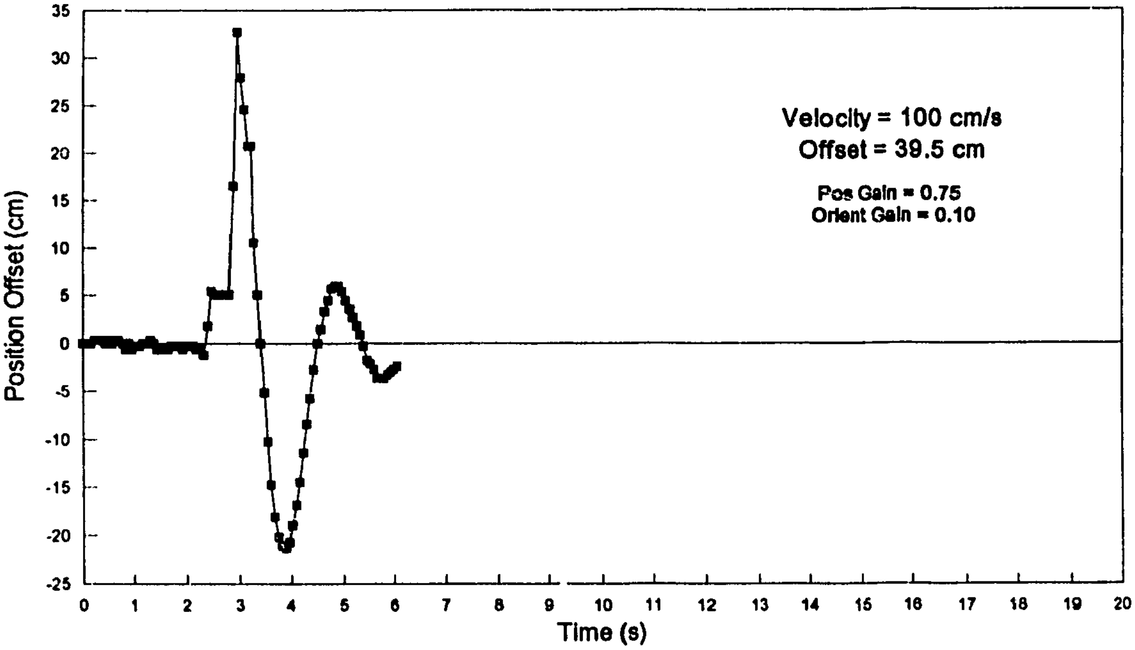


Figure 9.5d ATV Dynamic Response to Straight Line Track With Offset

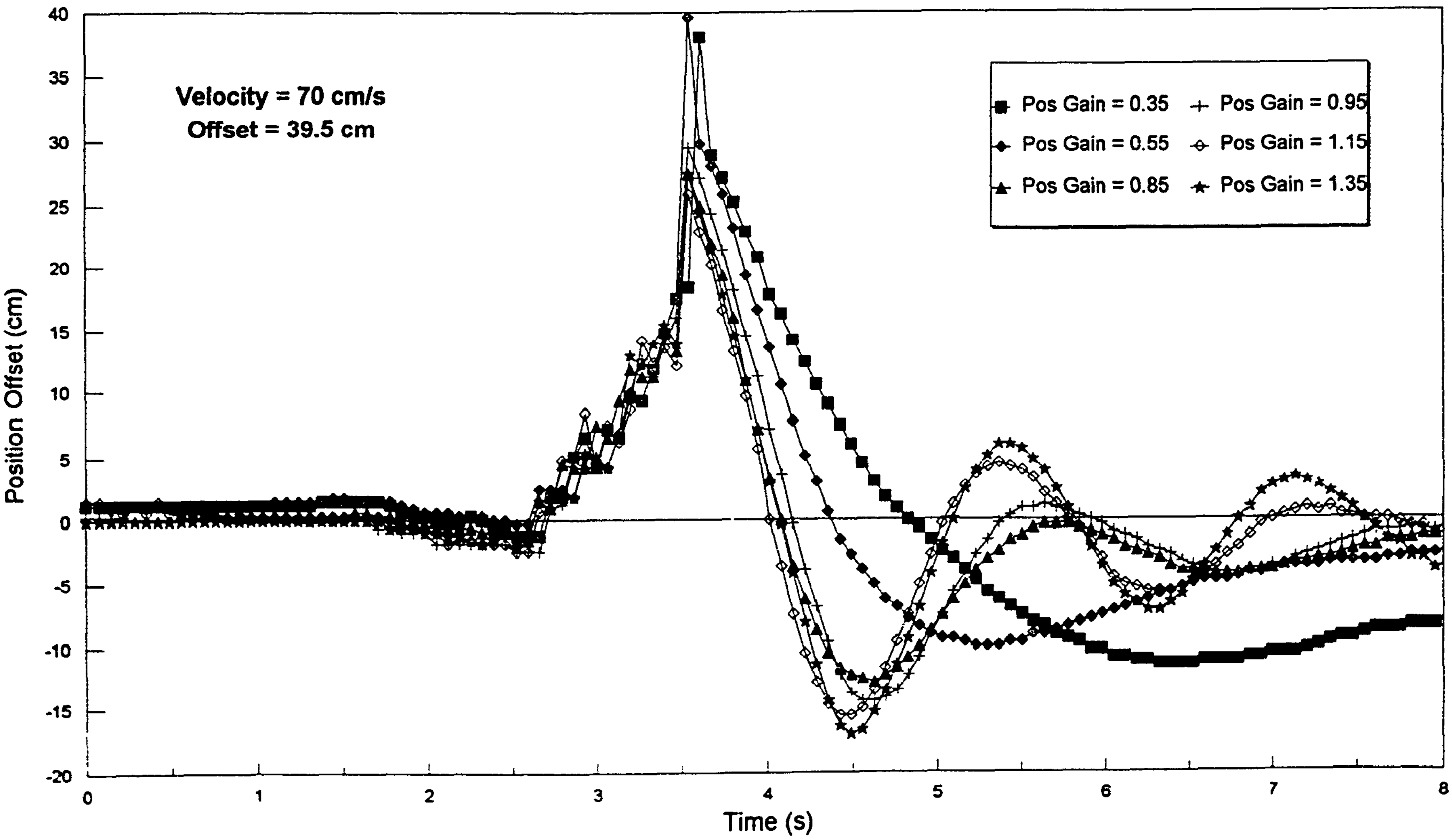


Figure 9.6 Dynamic Response of ATV to a Straight Line Track With Offset

offset gain. The vehicle's velocity is 70 cm/s and the track offset is still 39.5 cm. The orientation offset gain is kept constant at a value of 0.10. For low values of position offset gain, for example 0.35, the response is slow, overdamped, and it takes a long time for the vehicle t, properly follow the new track. As the position offset gain value is increased, the response becomes faster, with larger overshoots and damped oscillations observed. A position gain of 0.95 results in good performance for a velocity of 70 cm/s.

In Figure 9.7 the behaviour of the AGV to the same track offset at a lower speed of 30 cm/s can be observed. The shape of the plots is similar to the previous figure, however the values for the position offset gains are much higher. For good response, a value of 1.95 is required for the position offset gain. This indicates that the position offset gain has to increase as the velocity is reduced and it has to be decreased as the velocity is increased. Section 9.6 of this chapter describes the relationship between velocity and position offset gain.

The other parameter in the control equation is the value for the orientation offset gain. Figures 9.8a and 9.8b show the effects of different values of orientation offset gains keeping the position offset value at 0.95. It was previously found that a value of 0.95 for position offset gain gives good performance at 70 cm/s. The orientation offset gain does not greatly affect the performance of the AGV while following a straight line track with an offset. Figure 9.8a plots the position offsets for various values of orientation offset gains. The higher the value for this parameter, the more responsive the control is to changes in orientation. This responsiveness can be observed for values of orientation offset gains of 0.30 and 0.50. The control system starts to correct earlier, but more oscillations are observed. In all cases the vehicle is back following the track after a time of 8 s. In Figure 9.8b the orientation offset is

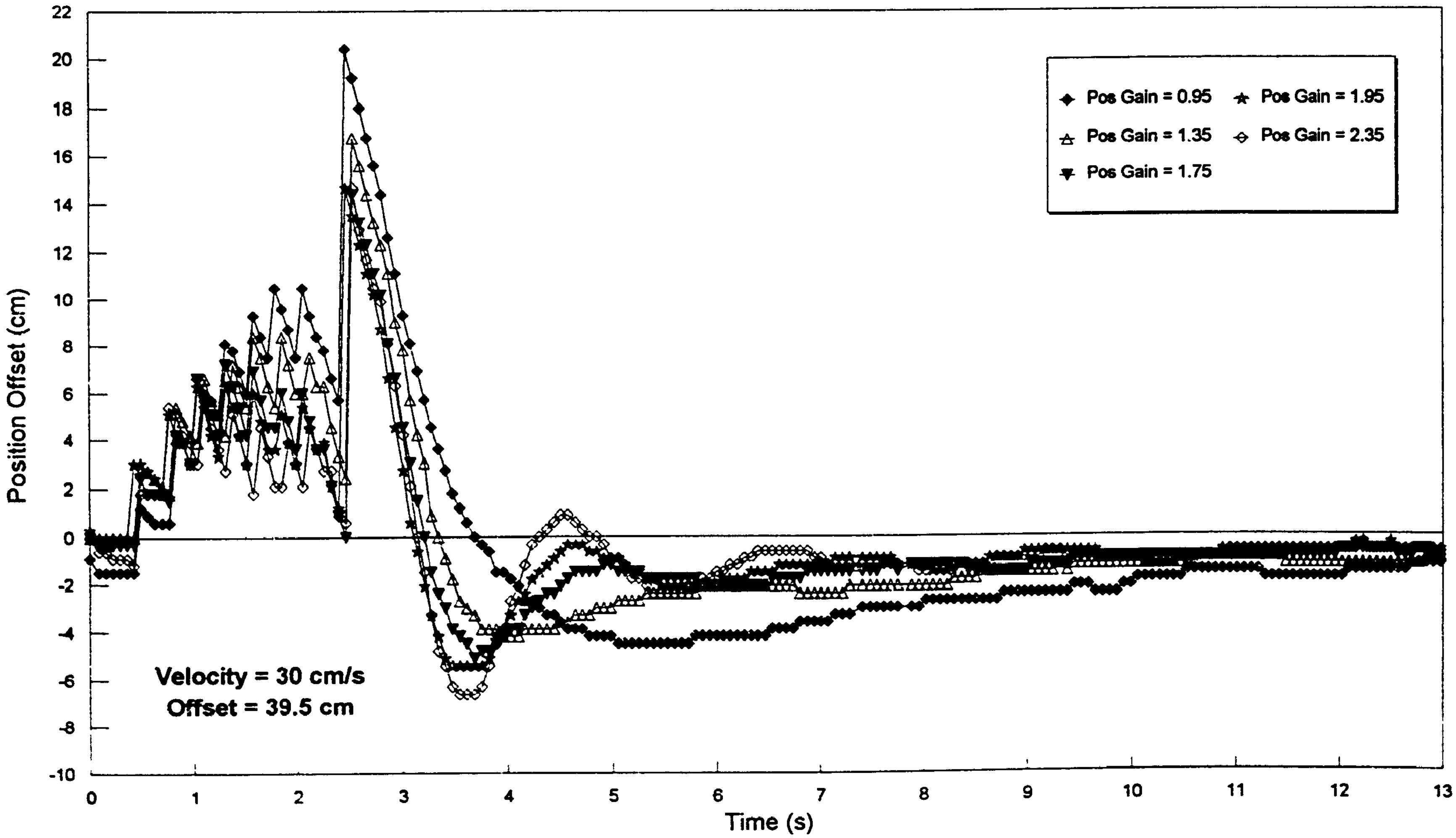


Figure 9.7 Dynamic Response of ATV to a Straight Line Track With Offset

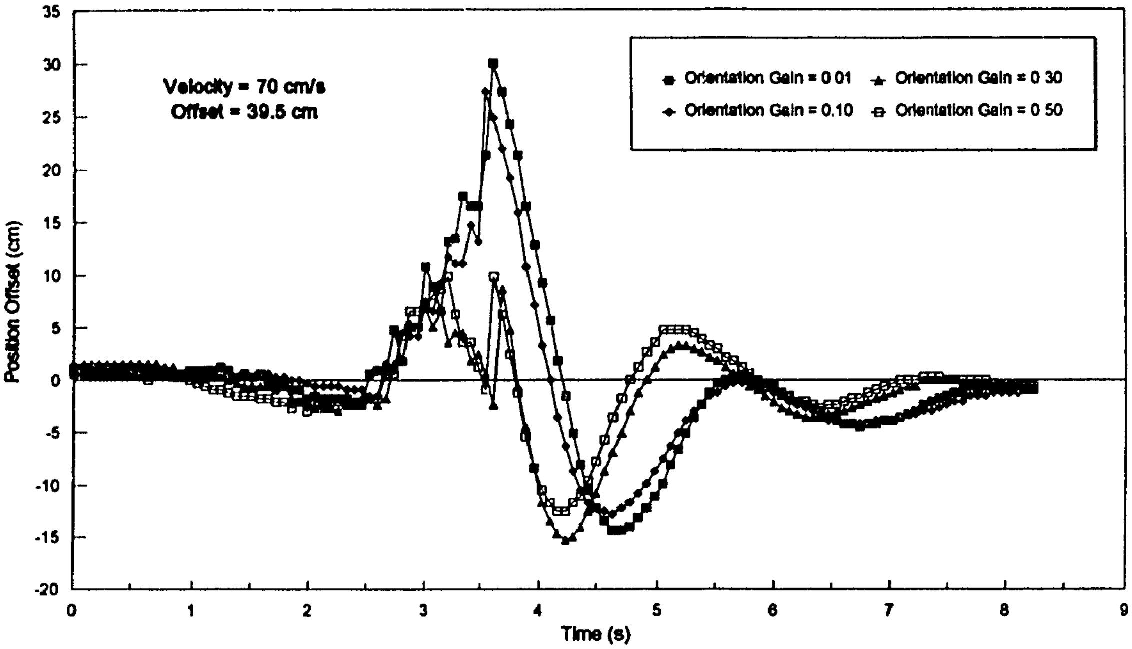


Figure 9.8a Effects of Orientation Gain to ATV's Response to Lane Change

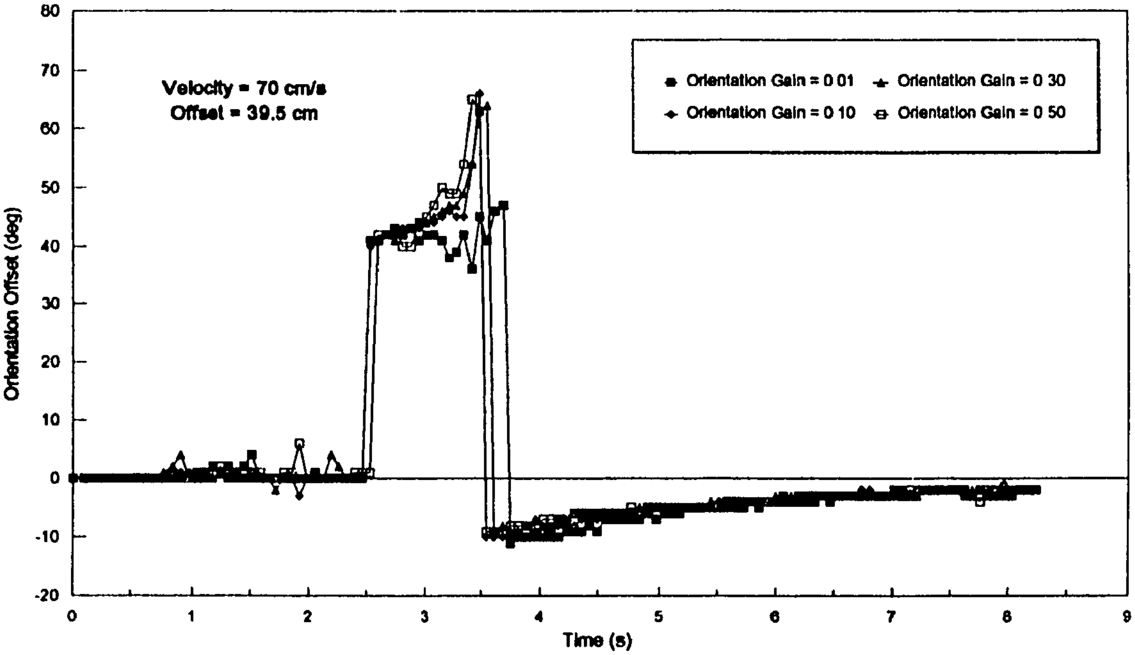


Figure 9.8b Effects of Orientation Gain to ATV's Response to Lane Change

plotted for various values of orientation offset gains; there is basically no difference among these curves.

The observations obtained from these tests are that for a certain velocity a corresponding best position offset gain has to be found. If the position offset gain is too high then oscillations and large overshoots can occur. If, on the contrary, the position offset gain is too low then slow response can be observed with larger position offsets. The orientation offset gain plays a smaller role in following straight tracks and a value of 0.10 is selected for straight tracks from the plots of Figure 9.8a and further experimentation.

9.5 Track Following Performance for Curved Tracks

The AGV's ability in following a curved track can be seen in Figure 9.9. The results shown are for a test in which the AGV follows a constant radius of curvature track at 100 cm/s. Figure 9.10 sketches the curved track followed. The AGV starts from a short straight line and enters the turn. At is can be seen, the position offset remains well within a range of ± 4 cm. The orientation offset keeps on building up as the AGV gets more into the turn and then it stabilizes. This is correct because once into a constant radius turn, the orientation offset should remain at a certain value to reflect the curvature of the track. It should be noted that the orientation offset gain is increased from the straight line value of 0.10 to a curve following value of 0.50. This increase of the orientation offset gain is necessary as it can be seen in Figures 9.11a and 9.11b. These graphs show the effects of changing the orientation offset gain to the position offset (Figure 9.11a) and to the orientation offset (Figure 9.11b). When the orientation offset gain is too low, for example 0.10, the AGV understeers and drifts further away from the track. If the orientation offset gain is too high, for example 0.70, the

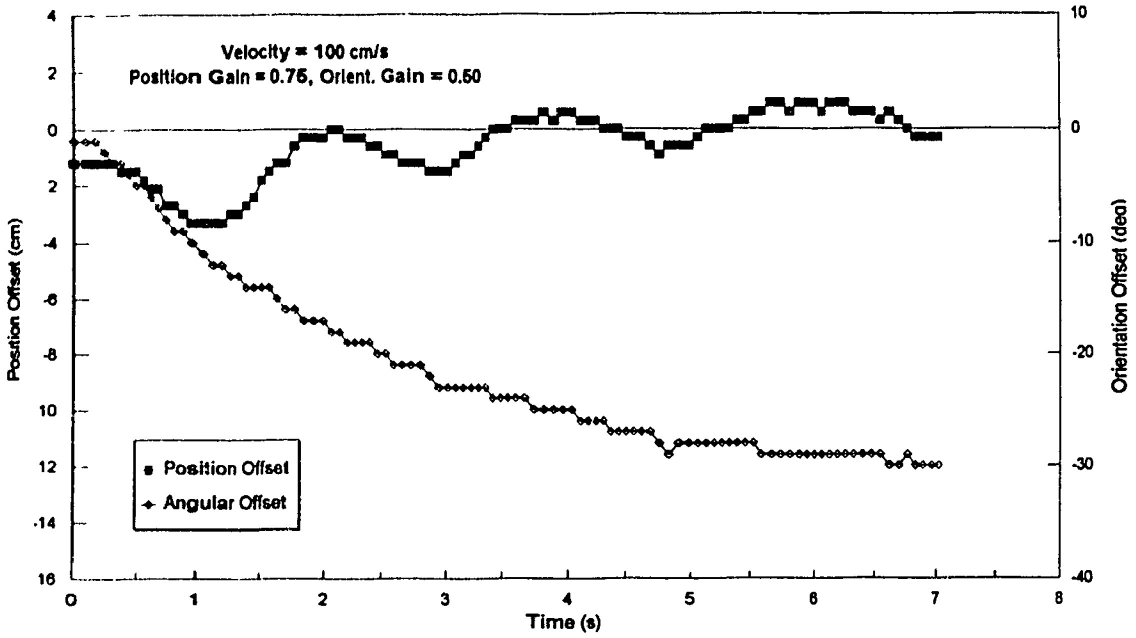


Figure 9.9 Curved Track Following (3.96 m radius of curvature)

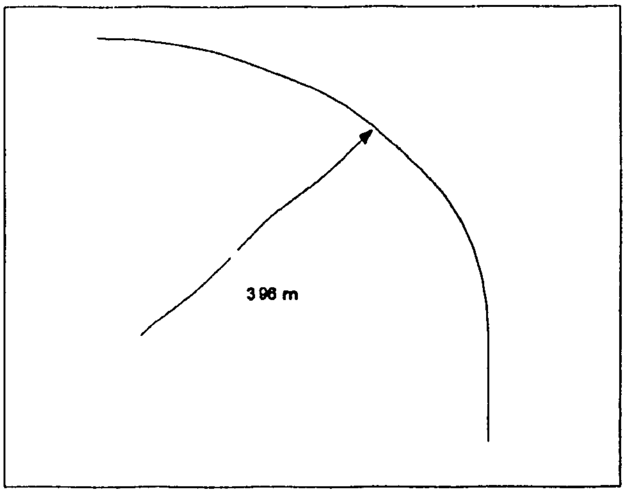


Figure 9.10 Sketch of Curved Test Track

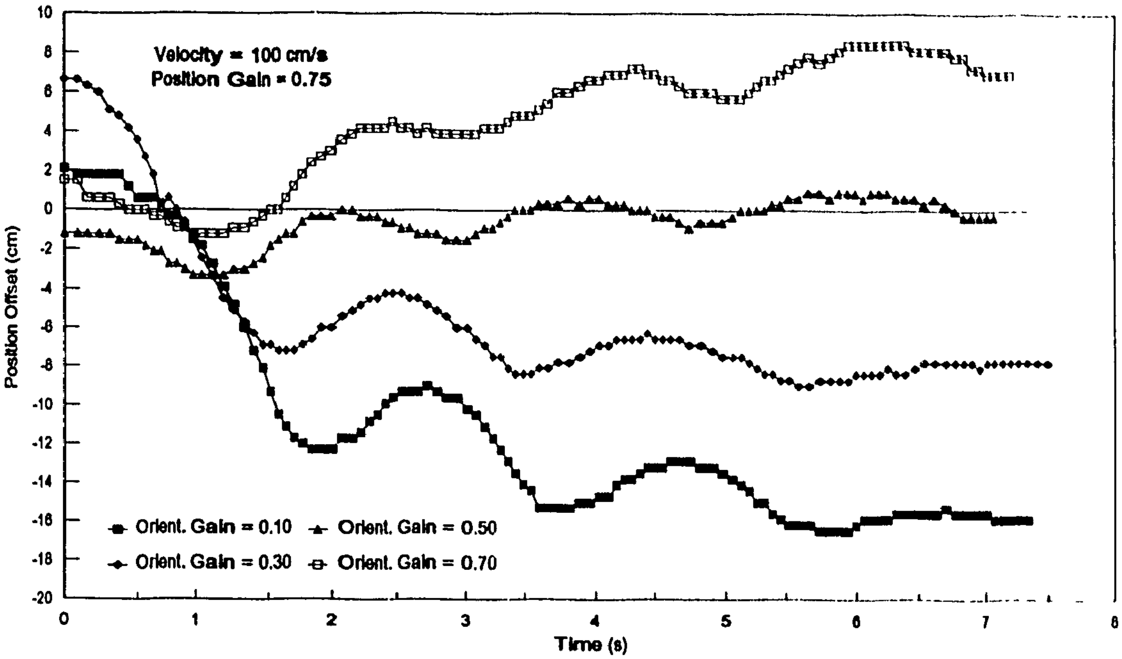


Figure 9.11a Effects of Orientation Gain to ATV's Response to Curved Track (3.96 m radius)

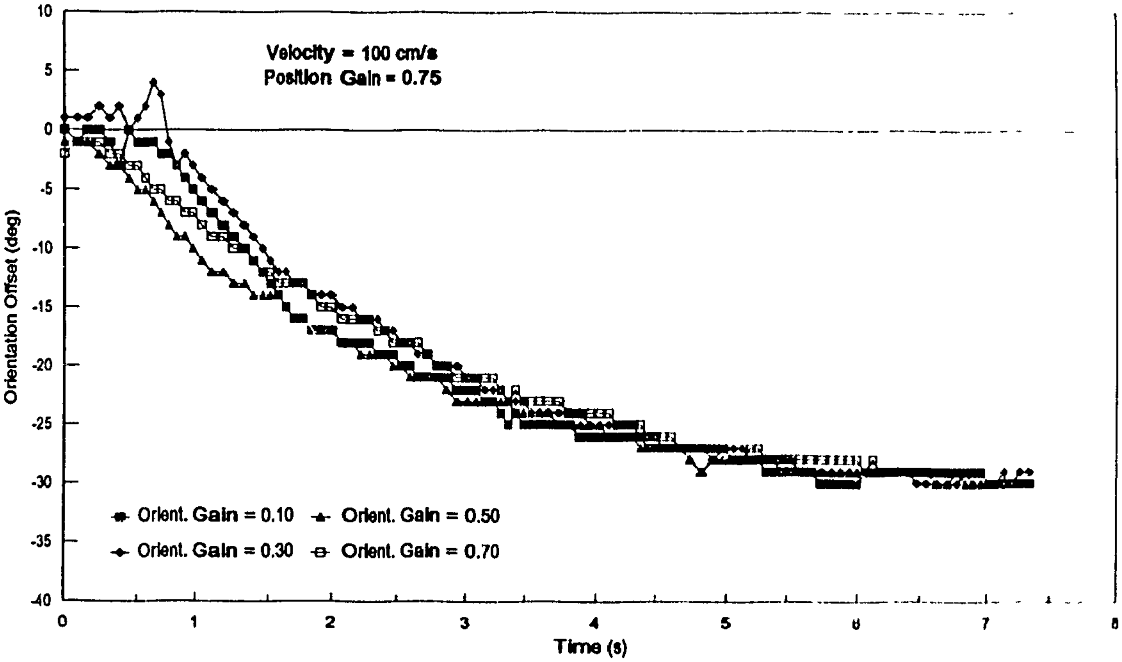


Figure 9.11b Effects of Orientation Gain to ATV's Response to Curved Track (3.96 m radius)

vehicle oversteers and cuts inside the turn. A value of 0.50 for the orientation offset is appropriate for this kind of turn. The orientation offset shown in Figure 9.11b is not affected by the changes in the orientation offset gain value. It should also be mentioned that there are limitations on the radius of curvature of the turns that the vehicle can follow. This is due to the fact that the CCD camera is fixed and it is mounted looking straight ahead of the vehicle. If the turn is too sharp then the track will disappear from the camera view and the vehicle will stop. This problem can be solved in several ways and a discussion can be found in Chapter 10 in the recommendations for future work section.

The next set of tests performed included the following of an S-shaped track. This track is made up of two turns blended into one another. The radius of curvature for each half-circle is 4.5 m and it is constant. The first turn is towards the left and the second one is towards the right. Figure 9.12 shows the results. The time at which the vehicle turns from left to right starts after about 6.5 s. At the inflection point, a temporarily large position offset and a transition of orientation offset from negative to positive can be seen. Not considering the transition period, the vehicle tracks the turns with a position offset in the range of ± 10 cm. This is still good because the vehicle is travelling at a considerably higher speed of 180 cm/s.

9.6 Dynamic Position Offset and Orientation Offset Gains

From the various test performed on different track profiles at different speeds it became clear that there is a need for continuously changing the position offset and orientation offset gains in response to the shape of the track and the travelling velocity. The position offset gain is related to the velocity of the vehicle and the orientation offset gain is related to

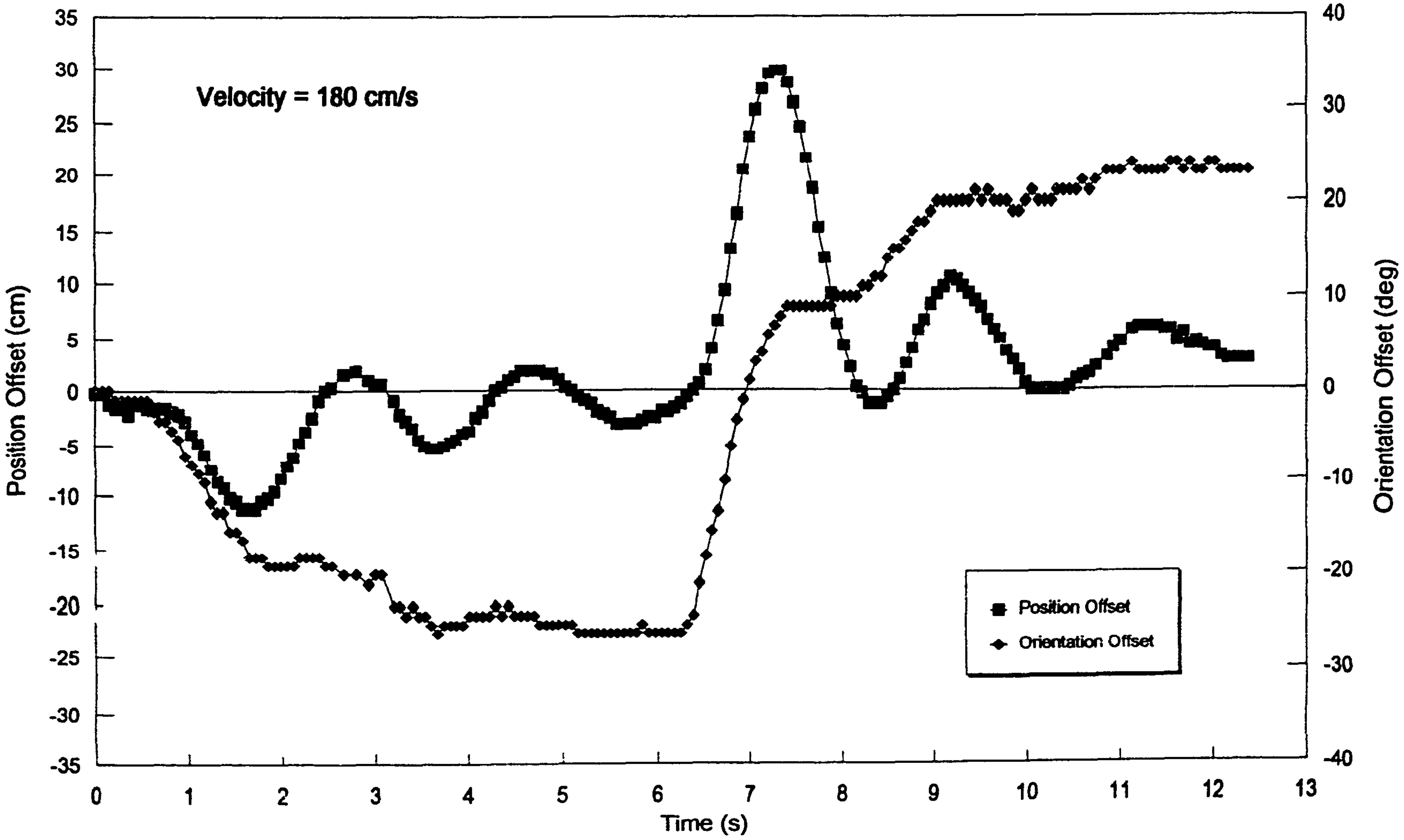


Figure 9.12 Tracking Performance for S-Shaped Track (4.5 m radii of curvature)

the curvature of the track. The modified control law equation was presented in Chapter 7 and rewritten here for convenience:

$$\theta_s = G_1(V)\ \varepsilon_d + G_2(R)\varepsilon_\theta \tag{9.1}$$

Where V is the speed cf the vehicle and R is the turning radius. The relations between position offset gain and speed and orientation offset gain and track curvature were determined experimentally from the data collected from the tests and extrapolated for a wider range of operation.

A curve fitting algorithm [9.4] is used to find the relation between position offset gain and speed from the data points available. The resulting equation is

$$G1 = -\frac{881.469}{V^2} + \frac{97.668}{V} - 0.222 \tag{9.2}$$

with the following conditions:

$$G1_{(min)} = 0.103; \qquad \text{and} \qquad G1_{(max)} = 1.954;$$

where Gl is the position offset gain and V is the vehicle's linear speed. The shape of this relation can be seen in Figure 9. 13a.

The relation between orientation offset gain and track curvature is a linear relation and the equation is

$$G2 = \frac{1.45\gamma}{10^{-6}} + 0.10 \tag{9.3}$$

with the condition: $G2_{(max)} = 0.50;$

where G2 is the orientation offset gain and γ is the sum of the residuals squared from the track curvature algorithm. This equation is again found experimentally and it is plotted in Figure 9.13b.

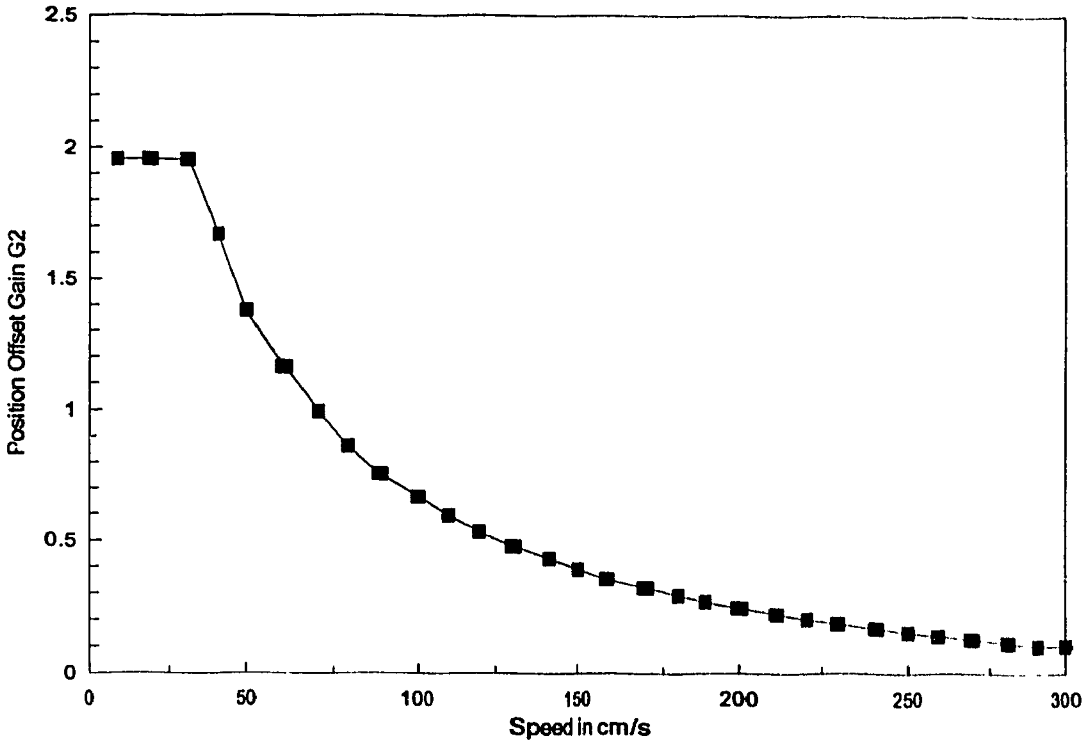


Figure 9.13a Relation between Position Offset Gain and Speed

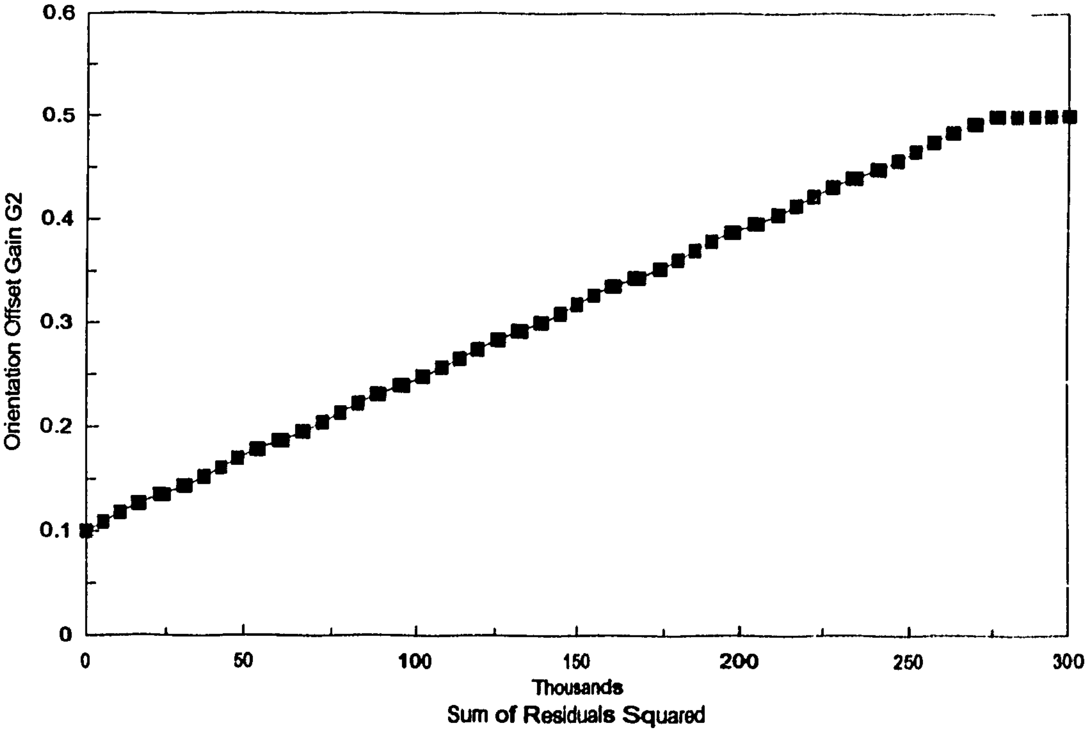


Figure 9.13b Relation between Orientation Offset Gain and Track Profile

The tracking performance of the vehicle obtained experimentally with these dynamic gains is shown in Figure 9.14. The shape of the test track can be seen in Figure 9.15. The AGV follows a straight track at 240 cm/s for the first 9.5 s. Then it slows down to 180 cm/s and enters the S-shaped track and stops at the end. The performance is good and the position offset is within a range of ± 6 cm. There are some spikes in the orientation offset which are noise, and are due to the vibration of the vehicle in response to the uneven pavement of the garage where the test was performed. The vehicle has no suspension and it is sensitive to large cracks in the pavement.

9.7 Summary

This chapter describes the various results obtained during the testing period of the vehicle. Initially the PID filters of the motion controllers were calibrated to provide a desired response from the driving and steering motors. Then tests were executed in order to see the response of the vehicle to straight track following. To adjust the gains of the guidance control loop, a straight track with an offset is used because it demonstrates the response of the vehicle to a step input. Once the gains have been selected then the curved track following is tested with success During the testing stages, the need for variable gains that responded to the conditions of the track and the travelling speed arouse and the guidance control loop was modified. The vehicle can follow any type of track at any speed, within the operating range, since the gains adjust themselves to track profile and speed. Good results were obtained on a mixed track made up of a long straight line followed by an S-shaped section.

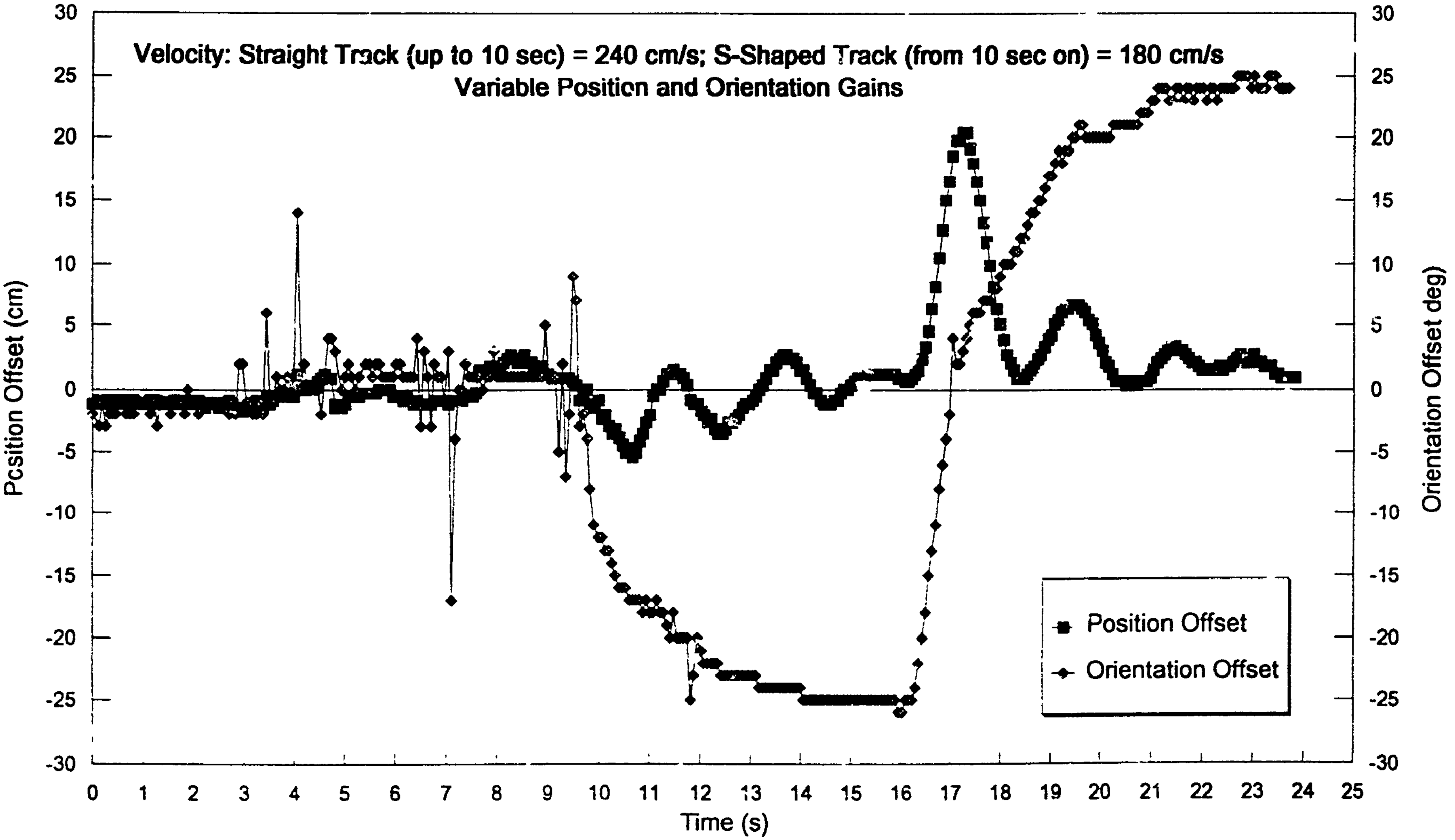


Figure 9.14 ATV's Performance on Mixed Type of Track (Straight & S-Shaped)

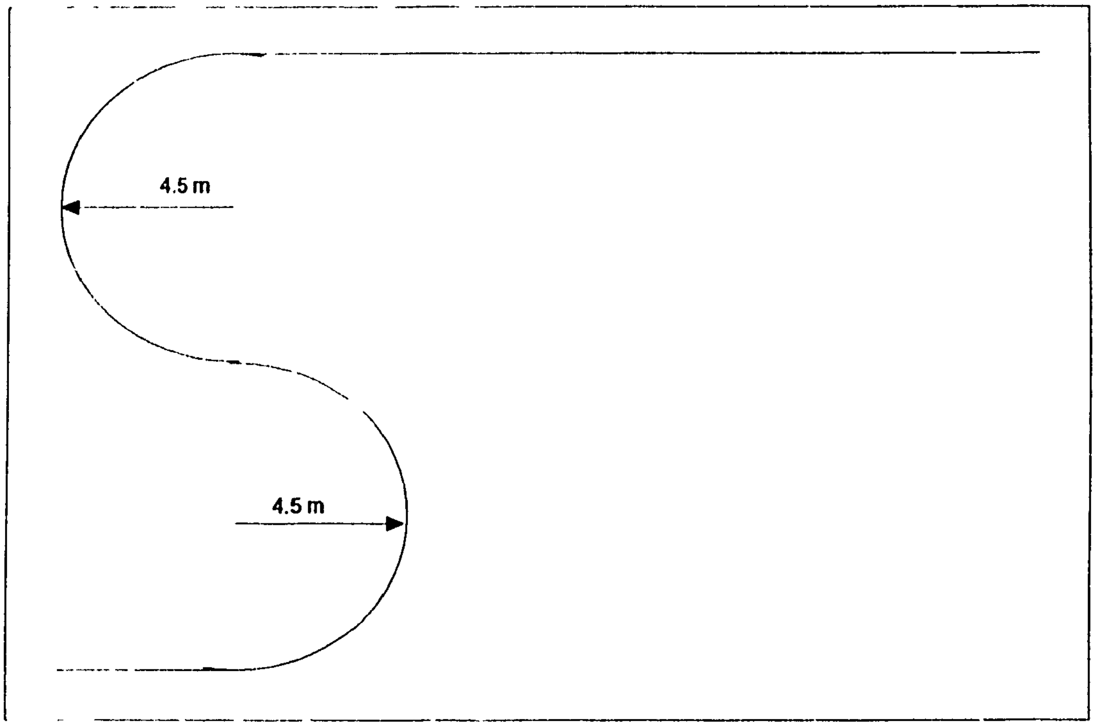


Figure 9.15 Sketch of Mixed Test Track

CHAPTER 10

CONCLUSIONS AND RECOMMENDATIONS

10.1 Introduction

In this thesis, the following research and development issues, pertaining to the development of high speed automated vehicles, have been successfully completed:

- Development of a High Performance Vehicle Controller.
- Development of a Guidance/Navigation technique using a CCD Camera.
- Development of a Parallel Processing based Hardware and Operating Software Interfaces for guidance and control of a Prototype CONCIC-4 Automated Transit Vehicle (ATV) available at CIC.

A high performance vehicle controller is developed using the latest parallel processing technology. An array of TMS320C40 parallel processing modules are integrated within a host PC. A third system consisting of LM628 motion controllers interacts with the other two systems to provide low level motor control. CCD camera vision is used for guidance of the automated vehicle. The images from the CCD camera are processed and analyzed to extract the necessary information for guiding the vehicle. Hardware and software interfaces, along with the operating system software, were developed for a prototype vehicle, CONCIC-4

This chapter is organized as follows. An evaluation of the systems performance is given. Then some suggestions on possible ways to improve the existing system are presented; this includes adding more cameras, having moving cameras, using a convolution board for hardware image processing, making some mechanical improvements, and using a simulation study to retune the parameters of the system for improved performance

10.2 Accomplishments

A high performance vehicle controller has been developed and tested. CCD camera images are used to provide the necessary guidance information to navigate the vehicle along a line on the road. Three main systems needed to interact to perform all of the above: the PC, which is the host for the other two systems; a five C40 DSP network that digitizes, processes, displays, analyzes the images and computes the guidance control loop and four LM628s precision motion controllers that perform the low level motor control. The operating system software, written using Parallel C and Turbo C, ties the systems together and controls their performance and interaction. The above mentioned hardware has been installed on an existing test vehicle to validate its performance in following tracks with varying radii of curvature. The results are encouraging, as it is shown in Chapter 9, and some improvements were identified. Overall, it should be said that good performance is achieved given the resources available.

10.2.1 Development of a High Performance Vehicle Controller

In order to develop a high performance vehicle controller, three hardware systems, a host PC, a C40 DSP network and four LM628 motion controllers, had to work and interact with one another First, the C40 DSP network was thoroughly tested on its own, to make sure that all processors could communicate with each other. Then, the host PC was used to access the hard drive information and to display messages from the C40 DSP network. The last step was to send data to the LM628 motion controllers from the C40 DSP network through the host PC Once the systems were able to interact with each other, the controller was built.

It should be mentioned that one of the advantages of this controller is that is it modular both in term of the hardware and software. The hardware is all off-the-shelf components and it can be easily replaced. More modules can be integrated and the network can be modified with minor changes in the software. The software is also modular and modules can be modified without the need to recompile the whole code. A bit mapped character set has also been developed to display information on the C40 graphical display module.

The controller's performance still had to be tested in controlling the test vehicle. Therefore the next step was to bring the vehicle to move under its own control. Many tests were conducted. Initially, the results were less successful, however as the various gains were experimentally determined, encouraging results were obtained. Some of the results are shown in Chapter 9. It can be concluded that the high performance vehicle controller can perform well in following various types of tracks. On straight tracks the position offset remained within a range of ±4 cm and the orientation offset within ±7 degrees, even at speeds of cm/s. A straight line with an offset was used to calibrate the position offset and angu gains. This simulates a step input for the guidance control loop. In curved track follo the vehicle performed well with a position offset within a range of ±4 cm. The vehicle was also tested on an S-shaped track. The position offset remained within ±10 cm while travelling at 180 cm/s.

10.2.2 Use of CCD Camera Vision for Vehicle Guidance/Navigation

The goal was to be able to provide to the control loop the camera feedback at least 15 times a seconds or every 67 ms. The big problem to overcome was the large amount of data (more than 7.8 Mbits/s) that had to be manipulated and processed. The speed and parallel processing of the C40 DSPs was a key element in resolving the problem. Many different configurations were tested yielding various results. Finally the following configuration was implemented. One C40 was used to digitize the CCD camera images and two other C40s in parallel, to manipuiate the images into the correct format and size; image processing is then performed to reduce the background noise. The road information is then extracted. Another C40 displays the processed images along with other system information. A fifth C40 computes the guidance control loop, interacts with the PC and instructs the low level motion controllers on how to perform the motion All of this is possible in 67 ms. The most time consuming operation is the image processing. The images are digitized from the camera at 512x512 pixels at 256 shades of grey. Then, they are reduced to 256x256 binary images. With this size, the images are processed at 15 frames/s, which is a good level of performance. At 15 frames/s the theoretical maximum travelling speed would be the camera view size times the frame rate which is 11.25 m/s or 40 km/h. Beyond this speed the camera would start to miss seeing sections of the road. Unfortunately, the mechanical design of the drive motors used in the prototype vehicle limits the maximum speed to 2.4 m/s (though the designed operating speed for the motors is only 1.2 m/s).

10.2.3 Performance Tests

The last goal was to provide smooth and accurate guidance control while following any track at any speed, without having to manually change the gains. It was found experimentally that proper dynamic coefficients for the position and orientation offset gains were necessary to achieve good control under different types of track profiles and speeds.

A method to compensate for curved track profiles was developed and tested From the results, that are provided in Chapter 9, it can be said that with proper dynamic gains, the position offset can be maintained within a range of ±6 cm for a mix of straight track and S-shaped track at speeds up to 240 cm/s.

10.3 Recommendations for Future Work

After having extensively experimented with CONCIC-4, it has been realized that there are several improvements that can be carried out to increase the operating speed, reduce tracking errors and negotiate curves with smaller radii of curvature. The major limitations are that the motors cannot run faster than 240 cm/s, and that the camera view is fixed, too small and therefore, sharp turns cannot be taken. The main new goals are higher speed, larger camera window, reduced tracking error and the ability to negotiate turns with smaller radii of curvature. In the following sections several suggestions are presented

10.3.1 Moving Camera System

In order to solve the problem of losing vision of the track in a sharp turn, a moving camera can be used. Presently, the camera is fixed and it views an area straight ahead of the vehicle. This causes a problem because as the vehicle takes a turn, the camera still looks ahead, however the track is now more on the side. If the turn is sharp enough the track will end up outside of the camera's view and the vehicle stops With a camera mounted on an auxiliary motor so that it can turn from left to right, this problem can be solved. As the vehicle makes a turn, the camera is rotated in the direction of the turn so that the track will remain in the camera's view, as the human driver does The control algorithm would have to be modified to take into account the rotation of the camera's view.

The camera could also be mounted on another motorized swivel so that it can look up and down. This would allow for the camera to view closer when the vehicle speed is reduced resulting in accurate track following and further away for high vehicle speeds to keep the vehicle in the middle of the road. An auto focus camera would also be required to compensate for the changing focusing distance. The control algorithm would also need to be slightly modified to take into account the different locations of the camera view. This moving camera system would also allow for sharper turns, since the camera can look closer to the vehicle, as the vehicle takes a turn and as a result, it would not lose the track from the camera view.

A combination of the two above mentioned systems could be implemented. The extra complexities in resolving for the location of the camera view can be computed by the C40 DSPs. Two small motors can be added and controlled by two more LM628 motion controllers in position mode. Two problems have to be solved: the vibration and the change in perspective. There would be more vibration because the camera is mounted on a more flexible assembly consisting of gears, motors, etc. The perspective problem is common to all implementations, involving a moving camera view.

10.3.2 Multi-Camera Vision System

When the vehicle is travelling fast the camera's view has to be further away to cover more road, when the vehicle is moving slower the view has to be closer to the vehicle to obtain good accuracy and resolution. An alternative solution to moving the camera is to use multiple cameras. Each camera is dedicated to a specific task. One can look further away, another can view closer to the vehicle. Also some cameras can be placed looking to the left and to the right when taking sharp turns. The cameras' images do not all have to be digitized, a multiplexer can be used to receive the images from one or both cameras depending on the type of information required. For example, if the vehicle is travelling at high speed then the camera viewing further away will be used and its images digitized by the C40 DSP. Multiple camera images can also be individually digitized and utilized when required by the system. The cost would be higher since the digitizing boards are more expensive than building a video switcher. Also with more images for the various cameras more C40 DSPs would have to be used to process the images.

10.3.3 Convolution Board for Hardware Image Processing

A dedicated board can be used to perform all of the required image processing algorithms used to filter the noise from the images. The image processing algorithms would be executed in hardware instead of software This would allow for very fast image processing and therefore in higher frame rates which would translate in higher operating speeds. The two main drawbacks of this alternative is the high cost of convolution boards and the fact that the image processing algorithms cannot be modified. To change the algorithms, a new board has to be purchased or the hardware has to be modified.

10.3.4 Retuning System Parameters Using a Simulation Study

Some of the system parameters, like the gains in the guidance control loop, could be tuned more precisely to improve the track following performance using a simulation. To write a software simulation, the forward and inverse dynamics of the entire vehicle have to be found. This could be a time consuming task. However, with a simulation package many more simulated test conditions can be experimented in less time. Also, with a simulation, the extreme cases in which the vehicle would lose control, can be identified so that they can be avoided during the actual tests. Many more combinations of position and orientation offset gains can be tried and better control may be achieved. A simulation can also resolve the problem of finding a suitable test site until the vehicle's parameters are theoretically tuned.

10.3.5 Mechanical and Drive Train Improvements

The mechanical design of the vehicle is the main limiting factor in testing the guidance control algorithm at higher speeds. The motor sets have to be changed for more powerful ones so that higher speeds can be reached. With the present drive motors, the vehicle can reach 240 cm/s. The guidance control scheme was designed for higher speeds and with faster motors the control's performance can be tested for speeds in excess of 240 cm/s. It should be mentioned that the present drive motors, when operated at 24 volts which is their nominal rating, they drive the vehicle at a speed of 120 cm/s. By doubling the voltage, a speed of 240 cm/s is reached. However this is not a recommended practice as it would reduce the life of the motor due to the extra heat generated; also the mechanical components like bearings may fail as the motors are being operated beyond the rated capacity.

Another problem of the mechanical design is the total lack of suspension. The aluminium frame structure is quite stiff and since the motor sets have a hard rubber surface, any small crack in the pavement is very much felt on the vehicle. This situation worsens at higher speeds where more vibration is felt every time when the ground surface is not perfect. This is shown in Figure 9.4 for an operating speed of 240 cm/s, where many spikes in the orientation offset are observed. Adding suspension is a difficult task because if the vehicle suspension is too soft, the camera view will swing all over the place and it would be complicated to provide to the control loop an exact vehicle position and orientation. Therefore a compromise has to be made so that some suspension is implemented without upsetting too much the camera view. All that is needed is to absorb the vibration from the small cracks on the road so that the camera does not feel these vibrations. Also, the computer equipment should not be exposed to harmful vibrations

To increase the stability of the vehicle, especially when taking sharp turns, the wheel span should be increased. The front motor sets and the read casters are mounted too close to each other towards the centre of the vehicle. Since there is some space within the chassis of the vehicle, the four wheels should be located more towards the edges of the vehicle. This provides higher mechanical stability since the wheel span is larger and faster turns can be taken, as well as the wheel base could be increased adding to the overall stability of the vehicle at higher speeds.

10.3.6 Curve Fitting Algorithm

Once the road centre coordinates are extracted from the images, they are combined to form an equation to represent the road location with respect to the vehicle's location. Presently, a linear approximation equation is used to represent the road. Unfortunately, this approximation is very coarse for tracks with curvature. The sharper the turn, the least accurate is the linear approximation for the road. By using better curve fitting to represent the road more accurately, improved curve following could be achieved. The extra computing required would not be a problem in terms of execution time, since the C40 DSPs excel in terms of executing computations fast.

Another alternative in order to describe the road more accurately is to divide the camera view into three or more sections. Then in each section, a linear approximation of the road can be used since the portion of the road in the section is fairly short and therefore can be approximated by a straight line. The section closer to the vehicle provides the immediate vehicle's position and orientation that will be used by the guidance control loop immediately. The other two sections could provide feedforward information to better predict the shape of the road and to provide guidance corrections, as required to improve the track following performance.

A different use of dividing the camera's view into three section is in the event that one camera frame has too much noise to detect the road. The section closest to the vehicle provides the present vehicle's position and orientation. If the next image cannot provide the road information then the next section of the previous image could provide an approximate vehicle' position and orientation for the guidance control loop. With this scheme, up to two or more bad images in a row can be tolerated and still allow the vehicle to function properly.

REFERENCES

Chapter 1

1.1 Electronics Today International, "Driving into the Future", *ETI*, pp. 10-19, November 1995 Issue.

1.2 R.M.H.Cheng, Y.Courbet, M.Surpaceanu, P.Favreau, A Fahim, "Investigation of an Automated Guided Vehicle (AGV) Driven by Camera Vision", Proceedings from the International Conference on Intelligent Autonomous Systems, pp.162-167, Amsterdam, The Netherlands, 8-11 Dec. 1986.

1.3 R. Rajagopalan, "Guidance Control for Automatic Guided Vehicles Employing Binary Camera Vision", PhD. Thesis, Concordia University, Montreal, Canada, Sept. 1991.

1.4 Data Sheets on Motor-in-wheel Drive Units, Schabmuller Elektromotoren, Feldkirchen, West Germany and NDC Automation, Charlotte, North Carolina.

1.5 Idetix Digital Vision System, Operator's Manual, Micron Technology Inc., Idahom U.S.A., 1986.

1.6 National Semiconductor, LM628/LM629 Precision Motion Controller, User Manual, March 1989.

1.7 PWM Servo Amplifier User and Operating Manual, Galil Motion Control, California, U.S.A., 1988.

1.8 M. Huang, "Dynamic Modeling and Simulation of an AGV (CONCIC-2)", Masters Thesis, Concordia University, Montreal, Canada, March 1991.

1.9 Mostafa Mehrabi, "Path Tracking Control of Automated Vehicles : Theory and Experiment", PhD. Thesis, Concordia University, Montreal, Canada, 1993.

1.10 N. Barakat, "Computed Torque Control of an AGV", Masters Thesis, Concordia University, Montreal, Canada, January 1996.

Chapter 2

2.1 Yong C.Cho and Hyung S.Cho, "A Stereo Vision-based Obstacle Detecting Method for Mobile Robot Navigation", *Robotica*, Volume 12 Part 3, pp. 203-216, May-June 1994.

2.2 S. Ishikawa, H. Kuwamoto, S. Ozawa, "Visual Navigation of an Autonomous Vehicle Using White Line Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 5, pp. 743-749, Sept. 1988.

2.3 R. Rajagopalan, R.M.H.Cheng, S.Lequoc, "A Guidance Control Scheme for Accurate Track Following of AGVs", *Proceedings of the 1992 IEEE International Conference* *on Robotics and Automation*, pp. 188-193, Nice, France, May 1992.

2.4 R. Rajagopalan, "Guidance Control for Automatic Guided Vehicles Employing Binary Camera Vision", PhD. Thesis, Concordia University, Montreal, Canada, Sept. 1991.

2.5 Ingemar J. Cox, "Blanche: Position Estimation for an Autonomous Robot Vehicle", *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp. 432-439, 1989.

2.6 U.K.Sharma, L.S.Davis, "Road Boundary Detection in Range Imagery for an Autonomous Robot", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 5, pp. 515-523, Oct. 1988.

2.7 M.A.Turk, D.G.Morgenthaler, K.D.Gremban, M.Marra, "VITS - A Vision System for Autonomous Land Vehicle Navigation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 3, pp. 342-361, May 1988.

2.8 Fabrice Noreils, Raja Chatila, "Plan Execution Monitoring and Control Architecture for Mobile Robots", *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 2, pp. 255-266, April 1995.

2.9 C.Thorpe, M.H.Hebert, T.Kanade, S.A.Shafer, "Vision and Navigation for the Carnegie-Mellon Navlab", *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, Vol. 10, No 3, May 1988.

2.10 Hans P.Moravec, "The Stanford Cart and the CMU Rover", *Proceedings of the IEEE*, Vol. 71, No. 7, pp. 872-884, July 1983.

2.11 K. Tani, N.Shirai, O.Matsumoto, K.Komoriya, S.Tachi, "Autonomous Control of Mobile Robots - An Approach to an Orienteering Robot", 1991 ISART.

2.12 T.Takeda, A. Kato, T.Suzuki, M.Hosoi, "Automated Vehicle Guidance Using Spotmark", *IEEE International Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, Dec. 1986.

2.13 S.Harmon, "The Ground Surveillance Robot (GSR): An Autonomous Vehicle Designed to Transit Unknown Terrain", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 3, pp. 422-435, June 1987.

2.14 Billur Barshan, H.F.Durrant-Whyte, "Inertial Navigation Systems for Mobile Robots", *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 3, pp. 328-342, June 1995.

2.15 Todd Jochem and Shumeet Baluja, "Massively Parallel, Adaptive, Color Image Processing for Autonomous Road Following", Carnegie Mellon University The Robotics Institute, Technical Report #CMU-RI-TR-93-10, May 1993.

2 16 M Rygol, P.McLauchlan, P.Courtney, "Parallel 3D Vision for Vehicle Navigation and Control", *Transputing '91, Proceedings of the World Transputer User Group (WOTUG) Conference*, April 1991.

2 17 A. Tachibana, K. Aoki, "TOYOTA Automated Highway Vehicle System", TOYOTA Technical Review, Vol. 43, No. 1, pp. 18-23, Sept. 1993.

2.18 F.Blais, M.Rioux, J.Domey, "Optical Range Image Acquisition for the Navigation of a Mobile Robot", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991.

2.19 R. Cok, J.Gerstenberger, C. Lawrence, H.Neamtu, "Applications of a Parallel Image Processor", *Transputing '91, Proceedings of the World Transputer User Group (WOTUG) Conference*, April 1991.

2.20 Henry Schneiderman, Marilyn Nashman, "A Discriminating Feature Tracking for Vision-Based Autonomous Driving", *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 6, pp. 769-775, Dec. 1994.

2.21 S.Tsugawa, "Vision-Based Vehicles in Japan: Machine Vision Systems and Driving Control Systems", *IEEE Transactions on Industrial Electronics*, Vol. 41, Issue 4, pp. 398-405, Aug. 1994.

2.22 M.E.Boudihir, M.Dufaut, R.Husson, "A Vision System for Mobile Robot Navigation", *Robotica*, Vol. 12, pp. 77-89, 1994.

2.23 A.M.Waxman, K. Lemoigne L.Davis, B.Srinivasan, T.Kushner, E Liang, T.Siddalingaiah, "A Visual Navigation System for Autonomous Land Vehicles", *IEEE Journal, Robotics and Automation*, Vol. Ra-3, No 2, April 1987.

2.24 Z.Zhang, O.Faugeras, "A 3D World Model Builder with a Mobile Robot", *The International Journal of Robotics Research*, Vol. 11, No. 4, pp. 269-285, August 1992.

2.25 H. Khalfallah, E.Petriu,F.C.A.Groen, "Visual Position Recovery for an Automated Guided Vehicle", *IEEE Transactions on Instrumentation and Measurement*, Vol. 41, No. 6, 906-910, Dec. 1992.

2.26 D.Monrgenthaler, S.J.Hennessy, D.DeMenthon, "Range-Video Fusion and Comparison of Inverse Perspective Algorithms in Static Images", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No 6, Nov/Dec. 1990

2.27 M.P.Ghayalod, E.L.Hall, F.W.Reckelhoff, B.O.Matthews, M.A.Ruthemeyer, "Line Following Using Omnidirectional Vision Control", *Proceedings of the SPIE - The International Society for Optical Engineering*, Vol. 2056, pp. 242-251, Sept. 1993.

2.28 Y.Zhao, C.V.Ravishankar, S.L.BeMent, "Coping with Limited On-Board Memory and Communication Bandwidth in Mobile-Robot Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, No. 1, pp.58-72, Jan 1994.

2.29 A.M.Flynn, "Combining Sonar and Infrared Sensors for Mobile Robot Navigation", *The International Journal of Robotics Research*, Vol. 7, No. 6, pp. 5-14, Dec 1988.

2.30 P. van Turennout, G.Honderd, L.J. van Schelven, "Wall-Following Control of a Mobile Robot", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pp 280-285, Nice, France, May 1992.

2.31 R.D.David Thiel, Andrew Russell, A.Mackay-Sim, "Odor Sensing for Robot Guidance", *The International Journal of Robotics Research*, Vol. 13, No. 3, pp.232-239, June 1994.

Chapter 3

3.1 Ian Graham & Tim King, The Transputer Handbook, Prentice Hall, Cambridge, England, 1990.

3.2 Warren B. Cope, "Selecting a DSP is no simple chore", *Electronic Products*, July 1990.

3.3 Texas Instruments, TMS320C4x User's Guide, 1991.

3.4 INMOS, Transputer Reference Manual, Prentice Hall, Cambridge, England, 1988

3.5 3L, Parallel C User Guide Texas Instruments TMS320C40, 3L Ltd., Edinburgh, Scotland, 1992.

3.6 Steve Heath: "Microprocessor Architectures and Systems RISC, CISC & DSP", Newnes, Manchester, England, 1991.

3.7 Robert L. Hummel: "PC Magazine, Programmer's Technical Reference: The Processor and Coprocessor", Ziff-Davis Press, Emeryville, California, 1992.

Chapter 4

4.1 R. Rajagopalan and F. Perelli, "Image Processing Performance of a Digital Signal Processing System for Guidance Control of Automated Transit Vehicles Using CCD Vision", *2nd International Conference on Road Vehicle Automation, ROVA'95 INTERNATIONAL,* Sept. 1995.

4.2 Texas Instruments, TMS320C4x User's Guide, 1991.

Chapter 5

5.1 Gregory A. Baxes: "Digital Image Processing a Practical Primer", Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

5.2 Sinectonalysis, Optimized Image Processing Library for TMS320C40/C30/C31, West Newton, MA, 1994.

5 3 Craig A. Lindley: "Practical Image Processing in C", John Wiley & Sons, 1991.

5 4 Rafael C. Gonzalez & Paul Wintz: "Digital Image Processing", Second edition, Addison Wesley, 1987.

5 5 R. Rajagopalan, F. Perelli, "Image Processing Performance of a Digital Signal Processing System for Guidance Control of Automated Transit Vehicles Using CCD Vision", *2nd International Conference on Road Vehicle Automation, ROVA'95 INTERNATIONAL*, Sept. 1995.

Chapter 6

6.1 R. Rajagopalan, "Guidance Control for Automatic Guided Vehicles Employing Binary Camera Vision", PhD. Thesis, Concordia University, Montreal, Canada, Sept. 1991.

6.2 D.M. Etter: "Engineering Problem Solving with Matlab", Prentice Hall, Englewood Cliffs, New Jersey,1993.

6.3 J. Neter,W. Wasserman, M.H. Kutner: "Applied Linear Statistical Models", Second Edition, Richard D. Irwin, Homewood, Il, 1985.

Chapter 7

7.1 R. Rajagopalan, "Guidance Control for Automatic Guided Vehicles Employing Binary Camera Vision", PhD. Thesis, Concordia University, Montreal, Canada, Sept. 1991

7.2 I.J. Cox, G.T.Wilfong: "Autonomous Robot Vehicles", Springler-Verlag, 1990.

7.3 R.Rajagopalan, "A Generic Kinematic Model for Automated Vehicles", to be submitted to the *Journal of Robotic Systems*, Dec. 1995.

7.4 National Semiconductor, LM628/LM629 Precision Motion Controller, User Manual, March 1989.

Chapter 9

9.1 Naresh K. Sinha: "Control Systems", Holt Rinehart and Winston, 1988.

9.2 Noriyuki Hori, Control Systems, Course notes, McGill University, 1990.

9.3 National Semiconductor, LM628 Programming Guide, Application Note, March 1990.

9.4 Eureka, Mathematical Software.

Appendix B

B.1 National Semiconductor, LM628/LM629 Precision Motion Controller, User Manual, March 1989.

APPENDIX A

OPERATING SYSTEM SOFTWARE DESCRIPTION

A.0 Overview

When the user starts the vehicle's software, the power to the motors should be off' Once the motion controller board has been in initialized then a messagge appears on the screen that the motors can be safely turned on. Then the front wheels, one at the time, rotate until the centre position is found. At this point the main menu options are available to the user. Each option is described in the following sections

A.1 Show the Default Setup

With this option the user can see the default setting for various parameters of the system. The first is the number of lines to scan in the images in order to extract the centre line coordinates. Usually a value of 10 lines yields good results. Next the PID filter values for the drive motors are displayed. This includes the Kp, Ki, Kd, and Il. The PID filter values for the steering motors are shown next. Lastly the values for the coefficients of the position offset and orientation offset gains are displayed.

A.2 Modify Default Setup

This option allows the user to change all of the above parameter values Care should be taken when changing the PID values for the driving motors since small changes in the proportional gain value can make the motors oscillate. Also the position and orientation offset gains should be modified in small increments so that the guidance control loop reaction can be somewhat predicted. The scan lines can be modified within a range of 3 to 18. To save the new parameter values and exit this menu the '0' key has to be pressed.

A.3 Data Acquisition Menu

This menu permits the user to setup the variables to be recorded by the data acquisition system. The user just has to select the number of the parameter to record and the on/off indicator will toggle. The number of samples to be recorded has to be defined with a maximum of 1000 samples. Once the user is finished it presses '0' and the parameters with ON sign beside them will be recorded by the data acquisition system. During camera mode operation samples of data are recorded on average every 67 ms, which means that a maximum of 67 seconds can be recorded. The starting and stopping the data acquisition can be selected at any time during the camera mode of operation by pressing the 'F5' key and 'F6' key respectively.

The data that can be selected to be recorded is as follows:

1. Desired vehicle speed
2. Desired steering angle
3. Desired left wheel speed
4. Desired right wheel speed
5. Left wheel steering command
6. Right wheel steering command
7. Actual left wheel speed
8. Actual right wheel speed
9. Position offset

10. Orientation offset

11. Centre line coordinates

The recorded data is saved as an ASCII text file under the name 'DATA0000 DAT' and it can be read by any text editor.

A.4 Keyboard Control Mode

Under keyboard control the user can manually move the vehicle without camera control. The up and down arrow keys are used to increase or decrease the vehicle's speed. The down arrow can also be used to backup the vehicle. Every time one of these keys is pressed the speed is increased or decreased by 10 cm/s. To steer the vehicle to the left and to the right the left and right arrow keys are used. Increments or decrements of 5 degrees are used. There are several ways to stop the vehicle depending on the kind of stop desired or required. A smooth stop can be achieved by pressing the 'S' key, upper or lower case 'spacebar' stops the vehicle by turning off the motors. In case of a panic stop the 'ESC' or the 'ENTER' keys can be pressed. The 'ENTER' key is also used to exit the keyboard control mode. To activate the data acquisition the 'F5' key can be pressed at any time. To stop the data acquisition the 'F6' key is used.

A.5 Camera Calibration Mode

This mode is used to calibrate the camera for optimum imaging. The lighting conditions can be tested in this mode because the processed images are displayed but no control action is given. The camera's aperture can be opened or closed to improve the quality of the images. The front vehicle lights can also be adjusted while in this mode. To exit the camera calibration mode 'ENTER' has to be pressed.

A.6 Camera Mode

To make the vehicle perform road following this mode has to be selected. Initially the vehicle is stopped and the track should be clearly visible on the display monitor. The wheels will be steered to the correct position. Then to advance and increase the speed the up arrow is used. The speed is increased by 20 cm/s increments. The down arrow does the opposite and slows down the vehicle in steps of 20 cm/s. The vehicle cannot go backwards in camera mode because the control algorithm would drive it away from the line since the steering commands would be executed in the opposite direction. To record with the data acquisition the 'F5' key can be pressed at any time. To stop the data acquisition the 'F6' key has to be pressed at any time. The 'ENTER' key has two functions: to stop the vehicle in case of a problem and to exit this mode and to return to the main menu options.

APPENDIX B

LMó28 PRECISION MOTION CONTROLLER

B.0 Overview

This appendix gives an operational overview of the LM628 motion controller and how it is used to implement position or velocity control of a DC motor. The motion controller can be used with any kind of DC motor as long as a quadrature encoder feedback signal is returned to the LM628. The main advantage of using a dedicated motion controller is that the host computer just has to download the trajectory parameters and the controller executes the closed loop motion control without host assistance. Therefore the host is free to execute other tasks. The internal components of the LM628 are shown in Figure B.1 and they are a position feedback processor, a trajectory generator, a digital PID filter, and a host interface. The following sections explain how the motion controller is programmed and used. For more complete specifications refer to the LM628/LM629 Precision Motion Controller User Manual [B.1].

B.1 Position Feedback Interface

The LM628 motion controller reads the position of the motor through the signals from an incremental encoder mounted on the motor. Two quadrature signal inputs are provided (Phase A and Phase B) as well as an optional index pulse (I) input. The index pulse is triggered once per revolution. The quadrature signals are used to keep track of the absolute position of the motor. The position is incremented every time there is transition of one of the two quadrature pulses. This provides for four times the resolution that the encoder

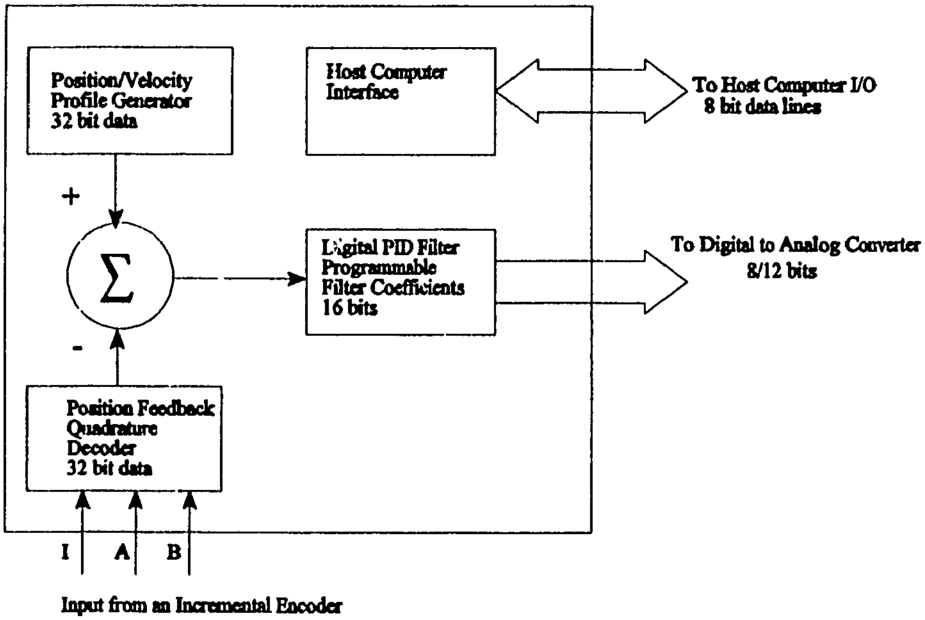

Position/Velocity Profile Generator 32 bit data
Host Computer Interface
To Host Computer I/O 8 bit data lines
+
Σ
-
Digital PID Filter Programmable Filter Coefficients 16 bits
To Digital to Analog Converter 8/12 bits
Position Feedback Quadrature Decoder 32 bit data
I
A
B
Input from an Incremental Encoder

Figure B.1 Elements of the LM628 Precision Motion Controller

disk can give. The motor position value can be read by the host computer as well as an internally computed motor velocity. In noisy environments precautions should be taken to provide 'clean' feedback pulses to the LM628.

B.2 Velocity Profile (Trajectory) Generator

The internal velocity profile generator computes a velocity profile that has a trapezoidal shape. It consists of an initial acceleration period, a constant velocity period, and a deceleration period. The acceleration and deceleration periods have the same duration. In position mode the values for acceleration, velocity, and position has to be specified. Once in motion the acceleration cannot be changed whereas the velocity and position can be modified. In velocity mode the values for acceleration and velocity have to be specified. Once in motion the acceleration cannot be modified. The motor has to be issued a stop command before the acceleration can be changed.

B.3 Sampling Interval

The sampling time for a digital controller is important because it can affect the control of the system. The proportional and integral terms of the PID filter have a sampling rate of $2048/f_{CLK}$ and since f_{CLK} is 8 MHZ for CONCIC-4, the sampling time is 256μs. The sampling time the derivative term is selectable within the range of 256 μs and 65,536 μs in steps of 256 μs.

B.4 Reading and Writing Operations

All commands and data written or read to/from the LM628 by the host computer is through the host I/O port, which is 8 bit wide. There are other pins on the chip that determine whether a command or data is on the I/O port and whether the I/O port has to write or ı ead. Commands are read or written when the Port Select input is logic low. Data is read or written with the Port Select logic high. To instruct the LM628 to write a byte, the Write pin has to be strobed. To read a byte the Read pin is strobed.

Before issuing any command or data, the host computer has to verify that the LM628 is not busy. This is done by checking the least significant bit of the status byte, which is called the busy bit. If this bit is set it means that the LM628 cannot read or write anything. The busy bit has to be checked before writing or reading any data or command. The busy bit is never set for more than 100 μs and on average is anywhere between 15 to 25 μs. The bit goes high as soon as a write command is issued, a read command is issued or a read/write data is given. When writing 32 bit data the busy bit has to be checked before sending the data and after 16 bits (2 bytes) are sent. If commands or data are written when the busy bit is set it will be ignored. If data is read during this time it will contain random values.

B.5 Motion Controller Commands

The LM628 motion controller has a set of high level commands that are used by the host computer to instruct the controller on what operations to perform. The commands are categorized as follows

1. Initialization Commands
2. Filter Control Commands
3. Trajectory Control Commands
4. Data Reporting Commands

5. Interrupt Control Commands

The various commands (excluding the interrupt control commands which are not used by CONCIC-4) and the corresponding hex value to be downloaded are grouped in Table B.1.

B.5.1 Initialization Commands

The LM628 provides four commands to initialize the system for use. The first is RESET and it is equivalent to a hardware reset. It clears all registers and performs a self check. All internal registers have to be redefined. The command PORT8 or PORT12 selected depending on the type of digital to analog converter (DAC) used. For an 8 bit DAC the PORT8 command is not necessary because it is the default. The DFH commands defines the home position which is the absolute zero position.

B.5.2 Filter Control Commands

There are two filter control commands. The first one LFIL is used the load the filter parameters whereas the other one, UDF, is sent to update the filter parameters. The LM628 stores the coefficients for the PID filter into a buffer. Once the update filter parameters command is sent, those coefficients are passed to the working registers so that these new PID filter values can be applied to the control loop.

Once the LFIL (load filter parameters) command is issued a filter control word is sent to identify which parameters will be sent. The filter control word uses 16 bits to allocate what is being sent. Table B.2 contains the allocation of the bits in the filter control word. Bits 15 through 8 are used to set the derivative sampling time and Table B.3 has the various selections. A bit set to high indicates a selected item.

Command	Type	Description	Hex Value	Data Bytes
RESET	Initialize	Reset LM628	00	0
PORT8	Initialize	Select 8-bit Output	05	0
PORT12	Initialize	Select 12-bit Output	06	0
DFH	Initialize	Define Home	02	0
LFIL	Filter	Load Filter Parameters	1E	2 to 10
UDF	Filter	Update Filter	04	0
LTRJ	Trajectory	Load Trajectory	1F	2 to 10
STT	Trajectory	Start Motion	01	0
RDSTAT	Report	Read Status Byte	None	2
RDSIGS	Report	Read Signals Register	0C	2
RDIP	Report	Read Index Position	09	4
RDDP	Report	Read Desired Position	08	4
RDRP	Report	Read Real Position	0A	4
RDDV	Report	Read Desired Velocity	07	4
RDRV	Report	Read Real Velocity	0B	2
RDSUM	Report	Read Integration Sum	0D	2

Table B.1 LM628 Command Summary

Bit Position	Function
Bit 15	Derivative Sampling Interval Bit 7
Bit 14	Derivative Sampling Interval Bit 6
Bit 13	Derivative Sampling Interval Bit 5
Bit 12	Derivative Sampling Interval Bit 4
Bit 11	Derivative Sampling Interval Bit 3
Bit 10	Derivative Sampling Interval Bit 2
Bit 9	Derivative Sampling Interval Bit 1
Bit 8	Derivative Sampling Interval Bit 0
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Not Used
Bit 4	Not Used
Bit 3	Loading Kp Data
Bit 2	Loading Ki Data
Bit 1	Loading Kd Data
Bit 0	Loading Il Data

Table B.2 Filter Control Word Bit Allocation

Bit Position								Sampling Interval µs
15	14	13	12	11	10	9	8	
0	0	0	0	0	0	0	0	256
0	0	0	0	0	0	0	1	512
0	0	0	0	0	0	1	0	768
0	0	0	0	0	0	0	1	1024
.	.	.	.	.	.	.	.	....
.	.	.	.	.	.	.	.	....
1	1	1	1	1	1	1	1	65536

Table B.3 Derivative Term Sampling Interval Selection Codes

B.5.3 Trajectory Control Commands

There are two trajectory control commands. LTRJ is load trajectory command and it is used to send the trajectory control commands (position, velocity, acceleration), the mode of operation (position or velocity), and for velocity mode the direction of rotation. A trajectory control word is sent after the LTRJ command to instruct the controller which parameters will follow. Table B.4 explains how the trajectory control word bits are allocated. Most of the selections are self explanatory. Bit 11, if set, tells the motion controller to operate in velocity mode otherwise position control will be executed. Bit 8, 9, and 10 are used to stop the motion. To turn off the motors and stop the motion bit 8 should be selected. To stop abruptly, like a panic stop, bit 9 should be set and the motor will stop using the maximum deceleration possible. For a normal stop that uses the same deceleration value as the acceleration, bit 10 has to be set to one.

Like for the filter control commands the values for acceleration, velocity, and position are initially placed in buffer registers. Once the command STT (start motion control) is sent then the desired trajectory parameters are copied into working registers and the motion is executed. It should be noticed that the acceleration, velocity, and position values can be relative or absolute.

B.5.4 Data Reporting Commands

There are several data reporting commands: RDSTAT (read status byte), RDDP (read desired position), RDRP (read real position), RDDV (read desired velocity), RDRV (read real velocity), RDIP (read index position), RDSIGS (read signals register), and RDSUM (read integration sum). It should be noted that read status byte has no command byte because the status byte can be read just by reading the I/O port. Table B.5 summarizes the status byte bit allocation. Also the real velocity is reported with only 16 bits of accuracy and in many applications the lost resolution is could be quite important. When reading the signal registers a 16 bit word is returned. Table B.6 describes the allocation of the bits of the signal register.

Bit Position	Function
Bit 15	Not Used
Bit 14	Not Used
Bit 13	Not Used
Bit 12	Forward Direction in Velocity Mode Only
Bit 11	Velocity Mode
Bit 10	Stop Smoothly (Decelerate as Programmed)
Bit 9	Stop Abruptly (Maximum Deceleration)
Bit 8	Turn Off Motor (Zero Output to DAC)
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Acceleration Will Be Loaded - Absolute
Bit 4	Acceleration Will Be Loaded - Relative
Bit 3	Velocity Will Be Loaded - Absolute
Bit 2	Velocity Will Be Loaded - Relative
Bit 1	Position Will Be Loaded - Absolute
Bit 0	Position Will Be Loaded - Relative

Table B.4 Trajectory Control Word Bit Allocation

Bit Position	Function
Bit 7	Motor Off
Bit 6	Breakpoint Reached (Interrupt)
Bit 5	Excessive Position Error (Interrupt)
Bit 4	Wraparound Occured (Interrupt)
Bit 3	Index Pulse Observed (Interrupt)
Bit 2	Trajectory Complete (Interrupt)
Bit 1	Command Error (Interrupt)
Bit 0	Busy Bit

Table B.5 Status Byte Bit Allocation

Bit Position	Function
Bit 15	Host Interrupt
Bit 14	Acceleration Loaded (but not updated)
Bit 13	Trajectory Update Executed (but filters not yet updated)
Bit 12	Forward Direction
Bit 11	Velocity Mode
Bit 10	On Target
Bit 9	Turn Off upon Excessive Position Error
Bit 8	Eight-Bit Output Mode
Bit 7	Motor Off
Bit 6	Breakpoint Reached (Interrupt)
Bit 5	Excessive Position Error (Interrupt)
Bit 4	Wraparound Occurred (Interrupt)
Bit 3	Index Pulse Acquired (Interrupt)
Bit 2	Trajectory Complete (Interrupt)
Bit 1	Command Error (Interrupt)
Bit 0	Acquire Next Index (Set Index Executed)

Table B.6 Signal Register Bit Allocation