Autonomous Lant Experimental Robotic Testbed (ALX)

Design Review

University of Minnesota

Prepared for the Third Annual International Ground Robotics Vehicle Competition

May, 1995

Section One

INTRODUCTION

The National Traffic Safety Board has indicated that in excess of 40% of heavy truck accidents are due to driver fatigue. It is necessary to provide systems which prevent such fatigue by assisting human drivers with integrated speed and headway regulation, roadway following, obstacle detection and collision avoidance, or serve as backup systems capable of taking over control of the vehicle when drivers are incapacitated. These autonomous vehicle control subsystems form a portion of the advanced vehicle control systems (AVCS) which will be required by future intelligent transportation systems (ITS). With the Minnesota Department of Transportation (Mn/DOT), we at the University of Minnesota are developing an autonomous vehicle control system for a semi tractor-trailer experimental testbed. For safety reasons, it is important to test potentially high risk innovative vehicle sensing and control strategies on small vehicles before implementation on the semi tractor-trailer. The *Autonomous Land Experimental Robotic Testbed* (ALX) was first designed to provide such a safe platform.

The ALX is based on an electric golf cart chassis, and is equipped with control computers, actuators and feedback sensors, dead-reckoning sensors for position estimation, an array of up to sixteen ultrasonic sonar range sensors with overlapping fields for obstacle detection, and a vision system for roadway sensing. It is designed to solve a specific problem: navigating an unknown outdoor road which is delimited with standard paint-stripe lane markers, while avoiding obstacles of unknown size, shape, number, and placement.

To examine our design and find out potential problems of the current ALX system in order to improve its performance, we plan to participate the Third Annual International Ground Robotics Competition.

Section Two

SYSTEM DESIGN

2.1 System Devices Overview

The Autonomous Land Experimental Robotic Testbed (ALX) is designed around a centralized computer system. The system design started with a golf cart chassis with dimensions shown in Figure 2-1.

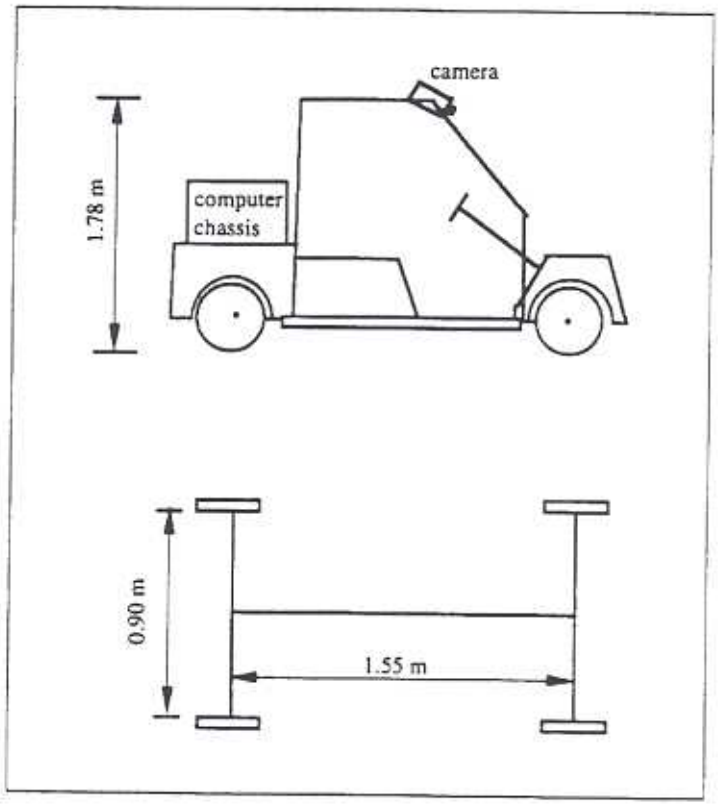

Figure 2-1. Dimensions of the ALX.

As any other autonomous land robotic vehicles, the ALX has a very complicated hardware design which consists of several subsystems as shown in Figure 2-2. These subsystems are described briefly as follows.

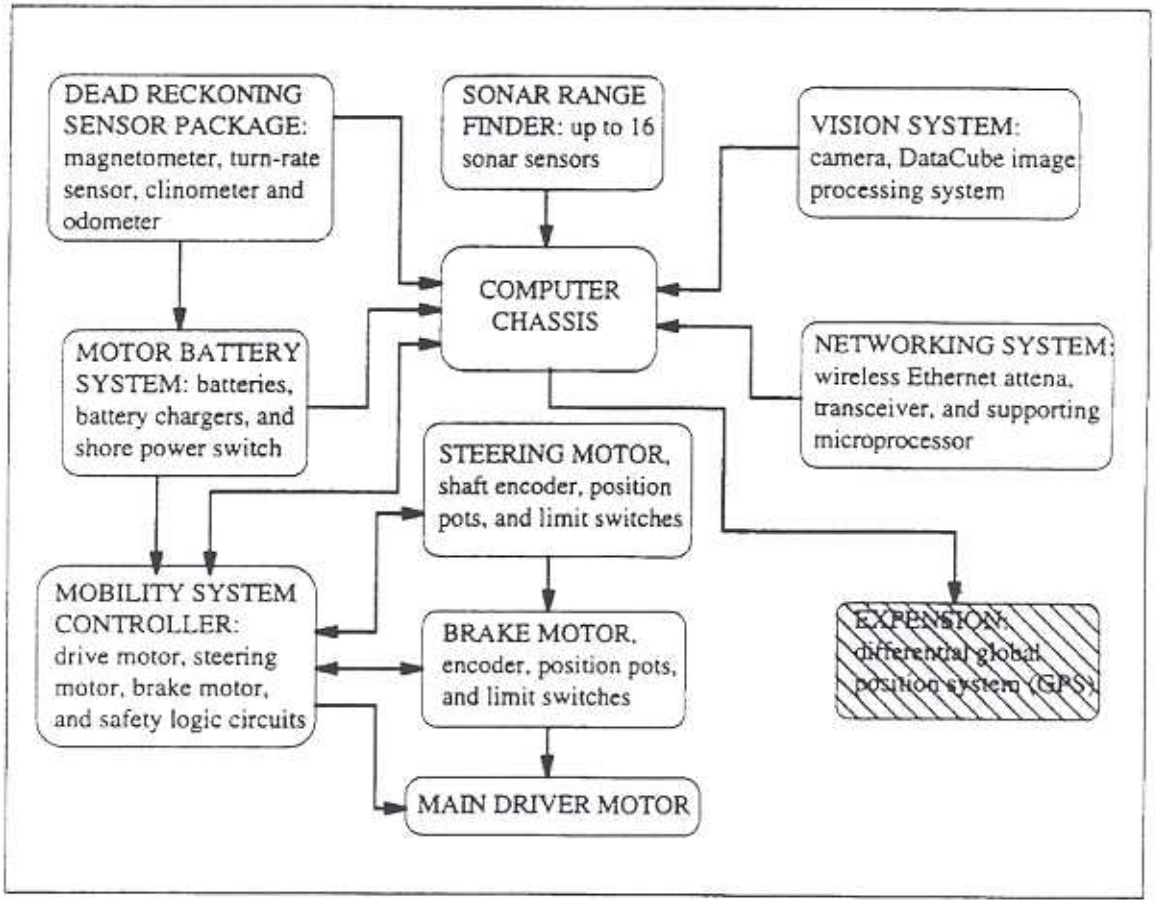

Figure 2-2. Hardware system layout of the ALX.

A. Dead Reckoning System

The dead reckoning system consists of a set of dead-reckoning sensors, which include one 2-axis fluxgate magnetometer, one angular rate sensor, two clinometers and one odometer (wheel encoder). The magnetometer, clinometers, and angular rate sensor are co-located in a single package.

The term "dead reckoning" refers to a mathematical process of projecting a point from a known location into a Cartesian coordinate space. The dead-reckoning sensors measure quantities needed by the position estimation program for the calculation of the ALX's current position ((x, y) coordinates with respect to a frame initially set up at the beginning of the ALX's operation) and orientation. These quantities are provided by the sensors, in analog

signal format, to the computer chassis for processing. Other entities sensed by the ALX (roadway edges, obstacles, etc.) will be referenced by using the ALX's current position and orientation information, i.e., dead-reckoning information.

B. Ultrasonic Sonar Range Finder

The ultrasonic sonar sensor range finding system is used to detect obstacles and allow the ALX to find a way around these obstacles. The information collected is accumulated and updated to build a dynamic virtual map which provides the ALX the local environment information. The sonar sensors are arranged on the ALX, as shown in Figure 2-3, to provide the ability to detect obstacles in front and behind.

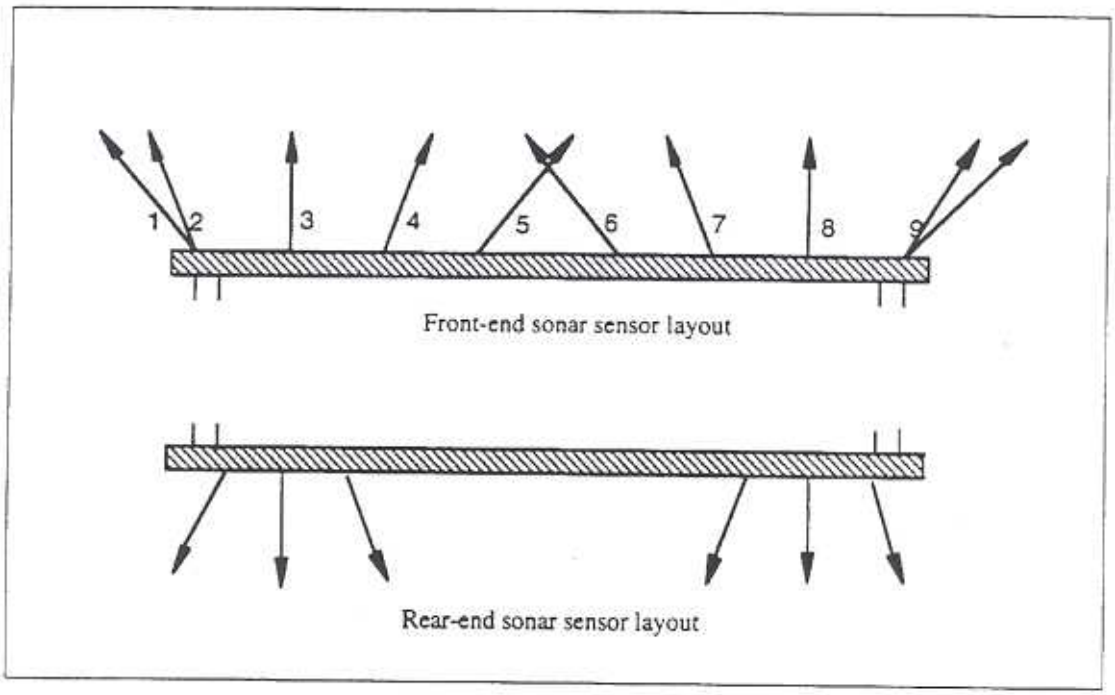

Figure 2-3. Sonar sensor layout.

Figure 2-4 depicts the relationships between various components of the designed sonar sensor processing board. Each sonar sensor assembly is based on a Polaroid experimenter's kit. This kit includes two Polaroid 604142 transducers, two signal processing/control circuit

boards, and two flextape connectors. This sonar processing board uses the National Semiconductor HPC-46003 16-bit integrated microcontroller, which has a precision timing ability. The CPU on board uses an 8-bit address latch and an 8-bit EPROM for program storage. Program RAM is internal to the CPU (256 bytes). All other HPC I/O pins are free for the sonar application I/O. The function of the sonar processing board is to coordinate the firing of the sensors and avoid potential interference of each other, measure the time-of-flight of the ranging echo pulse, and report range information to the main processor via a serial link RS232-C which is operated at a maximum rate of 9600 baud.

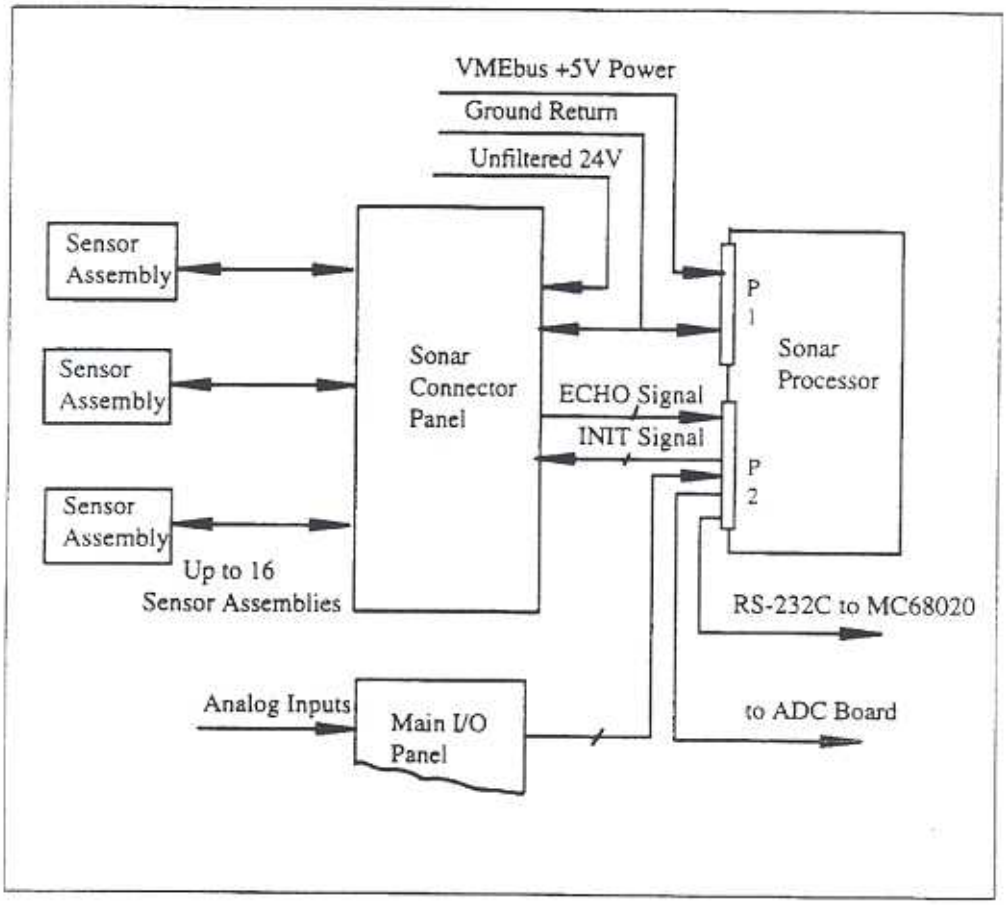

Figure 2-4. Sonar sensor processing board.

The current design of the sonar system polls at a fixed 10 Hz rate for each sensor group (up to 16 sensors for each group). A firing sequence of sensor groups can also be programmed.

C. Vision System

The vision system provides visual perception of the roadway for the ALX. It first determines the edges of the road from the images snapped by a CCD camera, then passes this information to the microprocessor on the ALX.

A 12V DC, 60Hz NTSC *JAVELIN* camera is used for the vision system. It is a color MOS solid state camera with high resolution (500 TV lines). The lens chosen for this camera is a 6.5mm TV lens which gives a 60 degrees horizontal angle of view.

The image processor for the vision system is the MaxTD 200 from DataCube, Inc. The communication link between MaxTD 200 and the main microprocessor (a MVME 130 XT) on the ALX was established via wireless Ethernet (refer to Figure 2-5).

D. Motor Battery System

The Motor Battery System on the ALX contains six 6-volt lead-acid deep cycle batteries used for the main drive motor of the ALX. It also provides power to the brake and steering motors. The six batteries are connected in series to provide nominal 36V power.

E. Mobility System Controller

The Mobility System Controller (MSC) contains the high-power controls for the main drive motor, the steering motor, and the brake motor. The MSC uses pulse-width modulation (PWM) to control the speed of the main drive motor. PWM frequency and duty cycle are supplied as inputs. The nominal PWM frequency is 10 KHz.

The MSC provides protection against illegal input commands (e.g., both "forward" and "reverse" commands simultaneously active). It also provides a mechanical relay disconnect for the main drive motor when in a safe state. The disconnect must be energized to supply power to the main drive motor. The MSC detects the brake and steering limit switch states and cuts off drive power (of the polarity driving to the limit) to the appropriate motor when a limit is reached.

2.2 Real-time Control Architecture

In classical sequential control, system actions are strictly ordered as a predefined time sequence: the behavior of the program depends only on the effects of the individual actions and their order; the time at which action is taken is not of consequence. Traditional sequential control systems are not appropriate for applications such as the autonomous navigation of a vehicle because of the on-going occurrence of multiple continuous and discrete events in the environment around the vehicle. Responding with actions in a pre-defined order usually will not satisfy time constraints for successful vehicle control.

We chose to use VxWorks real-time operating system (RTOS) to construct our computing architecture. By using VxWorks, we were able to develop an embedded real-time control system which differs from traditional control systems in that the sequence of actions is not determined by the designer but by events occurring in the outside world in real time.

A. VxWorks Real-time Operating System

The Wind River's VxWorks real-time operating system is a close partner for Unix. VxWorks handles the critical real-time chores, while Unix is used for program development and for non-time-critical applications. As a cross-development host, Unix is used to edit, compile, link, and store real-time code, which is then run and debugged on VxWorks.

VxWorks has a powerful development environment for real-time applications. It includes a fast run-time system, testing and debugging facilities, and a Unix cross-development packages, at the heart of which lies VxWorks's extensive Unix-compatible networking facilities, which allow VxWorks and Unix to combine to form a complete, integrated development and operational environment.

B. Cross-development Environment

To understand the ALX computer program cross-development environment, it is useful to outline the equipment first. The hardware for the ALX computer system includes one Unix

host system (on a SGI workstation), a MV133xt (MC68020) microprocessor which is a single-user VxWorks target system and is located on the ALX, and a DataCube vision processor which locates on the ALX. The SGI workstation is used for control software development and also contains the "*image*" of VxWorks RTOS, which is used to boot up the MV133xt. Radio frequency (RF) wireless Ethernet is used for inter-computer communications. The development setup is illustrated in Figure 2-5.

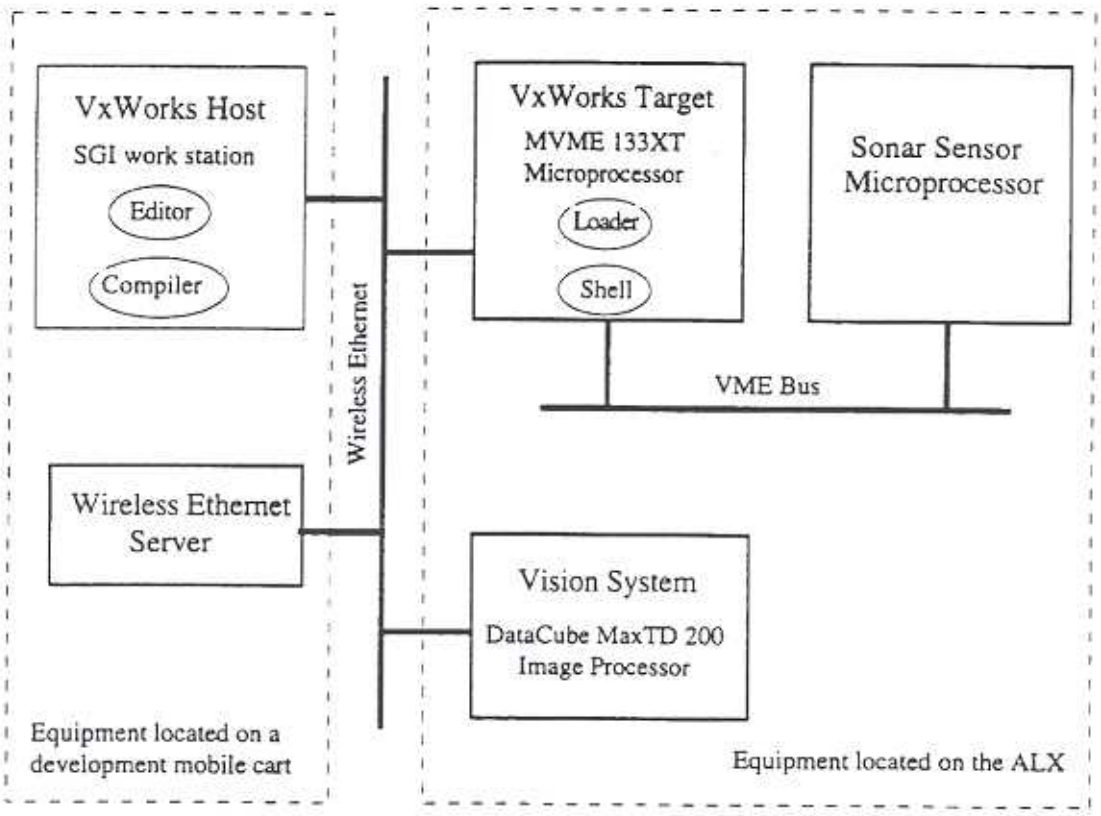

Figure 2-5. Cross development setup.

Software development for the ALX begins on the Unix host, the SGI workstation. Program modules are designed and implemented by using development and management tools on Unix such as text editor, the *make* utility, etc. Program modules are written in C language with use of the many library functions supplied by VxWorks, and are compiled with C cross-compiler provided with VxWorks. The program modules do not need to be linked with the

VxWorks system libraries or even with each other. Instead, VxWorks is capable of loading the generated object modules directly to the target MV133xt, using the symbol table contained in object modules to dynamically resolve external symbol references.

Selected modules can be dynamically loaded across the network for testing and debugging. The VxWorks shell program can then be used to interactively invoke and test individual subroutines, or complete tasks. The shell remembers the symbol tables from previously loaded object modules, giving symbolic access to data and subroutine names. User can examine data variables, call subroutines, spawn tasks, disassemble code in memory, set breakpoints, obtain subroutine call tracebacks, and so on, all using the original symbolic names. Source-level debuggers are provided that allow the application to be viewed and debugged in the original source code.

The cycle of building, downloading, and testing modules is iterated until the program is ready for the real-world testing environment.

C. Multi-tasking Control Structure

Modern real-time applications are constructed as a set of independent but cooperating tasks. The real-time computer control system for the ALX is based on the complementary concepts of *multitasking* and intertask communications.

Multitasking provides the fundamental mechanism for an application to control and react to multiple, discrete real-world events. The basic multitasking environment is provided by the VxWorks real-time kernel. Multitasking creates the appearance of many programs executing concurrently when, in fact, the kernel interleaves their execution on the basis of a scheduling algorithm. Each apparently independent program is called a *task*. Each task has its own *context*, which is the CPU environment and system resources that the task sees each time it is scheduled to run by the kernel. In the ALX control program, there are several tasks running concurrently, with three messages queues established for intertask communication, as shown in Figure 2-6.

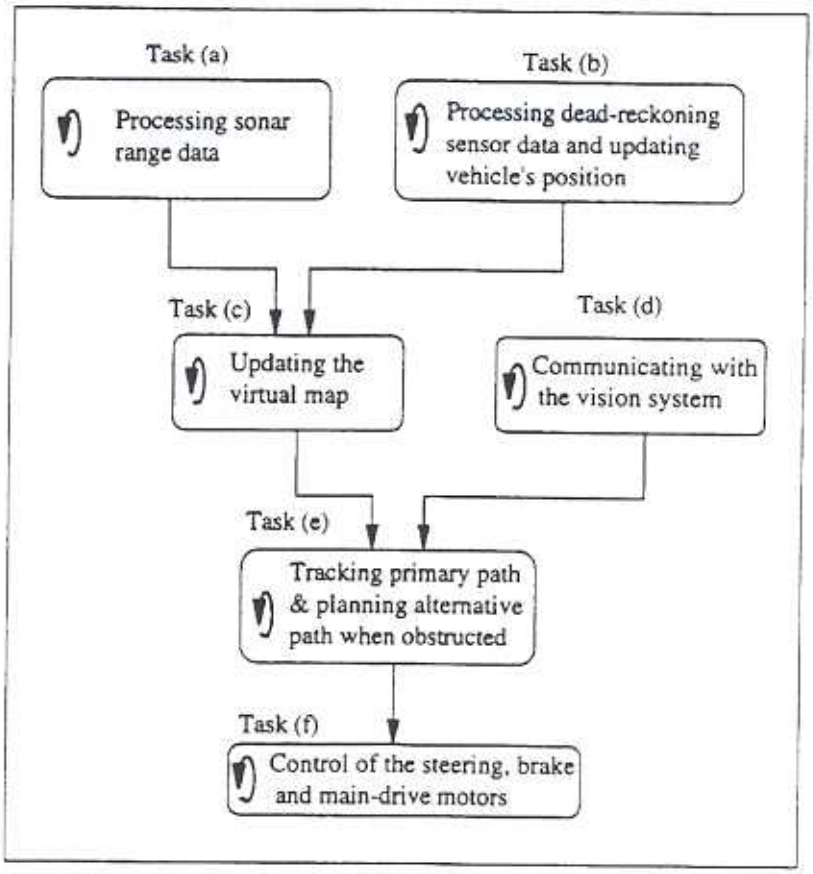

Figure 2-6. ALX multi-tasking structure.

The VxWorks multitasking kernel uses interrupt-driven, priority-based task scheduler, which features fast context switch times and low interrupt latency. The scheduler is preemptive in that if a task that has higher priority than the current task becomes ready to run, the kernel will immediately save the current task's context and switch to the context of the higher priority task. Under VxWorks, any subroutine for the ALX control may be *spawned* as a separate task, with its own context and stack. Other basic task control allows tasks to be suspended, resumed, deleted, delayed, and moved in priority. With this preemptive priority-based scheduler, each task is assigned a priority and the kernel ensures that the CPU will be allocated to the highest priority task that is ready to run.

The intertask communication facilities allow these tasks to synchronize and communicate in order to coordinate their activity. The primary intertask communication

mechanism used in the MV133xt microprocessor on the ALX is *message queues*. Message queues allow a variable number of messages, each of variable length, to be queued in FIFO (First-In-First-Out) order. Any task can send messages to a message queue. Any task can receive messages from a message queue. Multiple tasks can send to and receive from the same message queue.

Section Three
CONTROL ALGORITHMS

The control of the ALX consists of two parts: path tracking and obstacle avoidance.

3.1 Path Tracking

We have developed a goal-point based path tracking method which first uses image processing to extract and reconstruct roadway edges, then uses goal generation scheme to form a path defined by goal points, while in the meantime ~~conducts~~ provides vehicle heading control by using a pure-pursuit algorithm.

A. Visual Perception of the Roadway

The solo purpose of the vision algorithm for the on-board camera is to extract the information of the white lines from the captured roadway images. The processing procedures used to re-construct the roadway edges are depicted in the image processing flowchart shown in Figure 3-1.

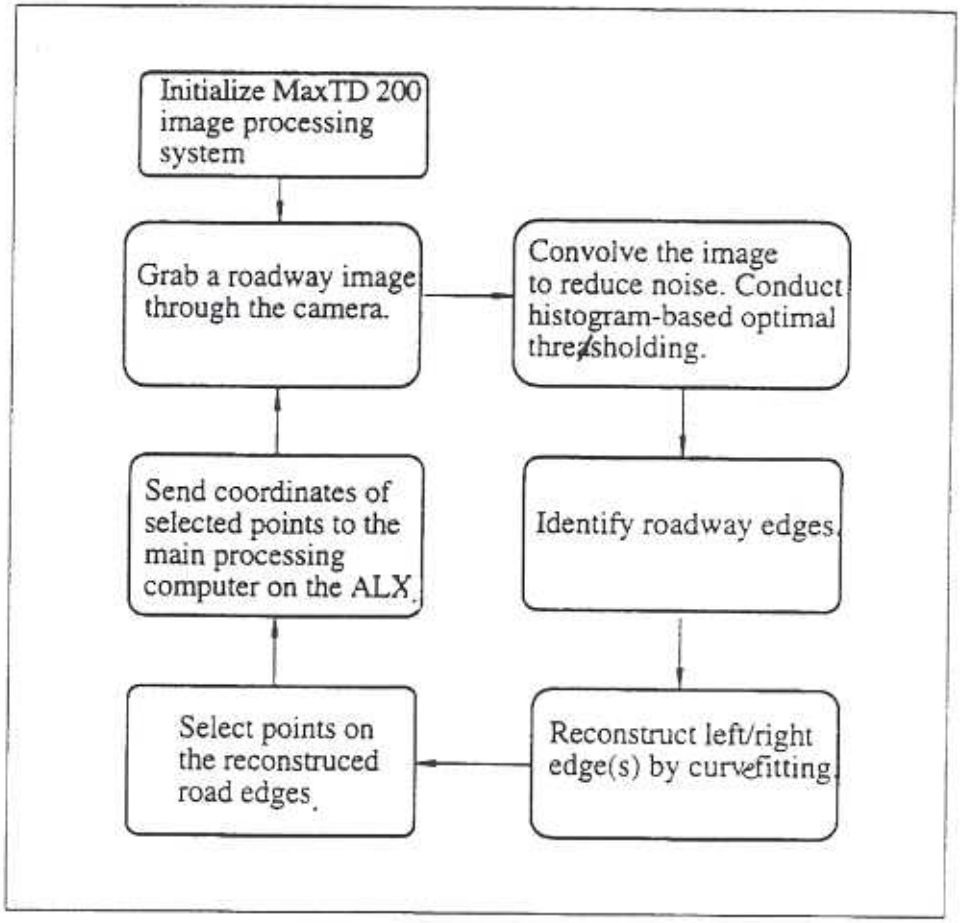

Figure 3-1. Flow chart of image processing.

B. Path Generation and Tracking

In our path tracking algorithm, any path, either generated from visual perception result or from alternative path planning algorithm when the vehicle is obstructed, is represented by a finite number of points on the center-line of this path. These points are called *goals*. The goals are defined in the world coordinates, and they represent the positions which the vehicle is directed ~~desired~~ to reach. Goals are generated from the vision information, and their coordinates are stored in a datastructure called *goal stacks*. Figure 3-2 shows an example of a path and its corresponding goal stack.

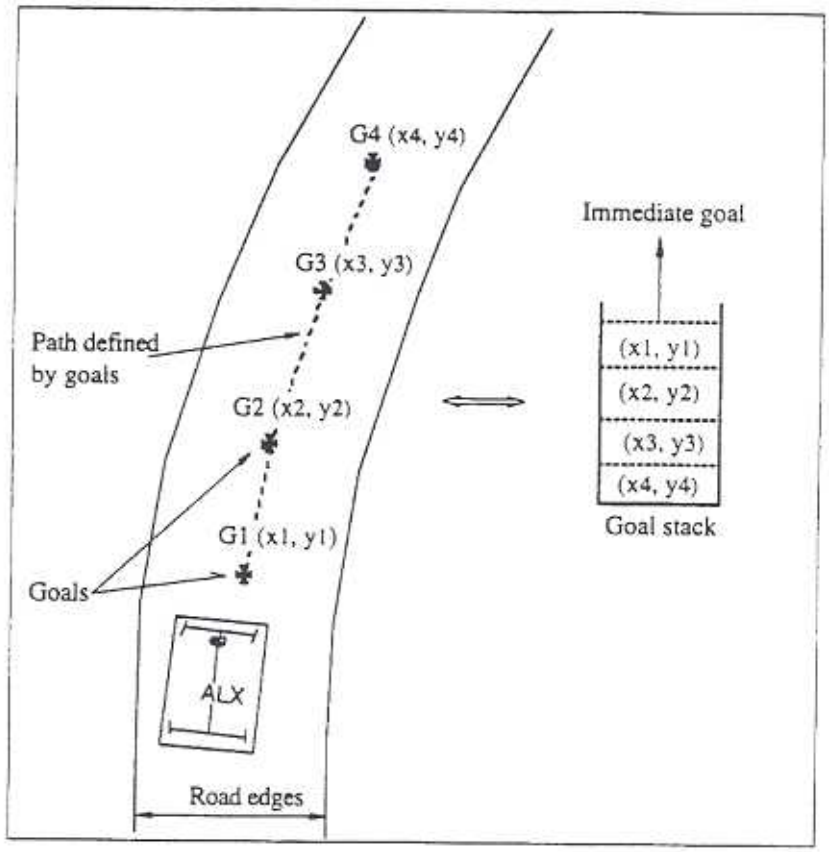

Figure 3-2. Illustration of the goal stack concept.
Goals are generated and stored in the goal stack.

Once the immediate-ahead goal is selected, the ALX uses a pure-pursuit method to conduct heading control. This method continually calculates the heading errors and issues steering command accordingly.

C. Obstacle Detection and Collision Avoidance

One of the key issues for the ALX's autonomous operation is its ability to detect imminent obstacles ahead in the roadway, and to plan detouring path(s) accordingly. We have successfully developed an obstacle detecting system based on a suite of up to sixteen ultrasonic sonar sensors. These sensors provide spatial information of the ALX's local environment and register this information into a Cartesian grid map (called *"virtual map"*.) Because of the relative wide beam width of the ultrasonic sensors, incorrect and inconsistent

interpretations of the raw sonar data may occur during range data processing. We developed a multi-value labeling scheme to remove these systematic errors and minimize the ambiguities of the obstacle positions. Once the ALX is determined to be obstructed, to find a detouring path, a graphic searching on the virtual map using binary-tree searching approach is then conducted (Figure 3-3).

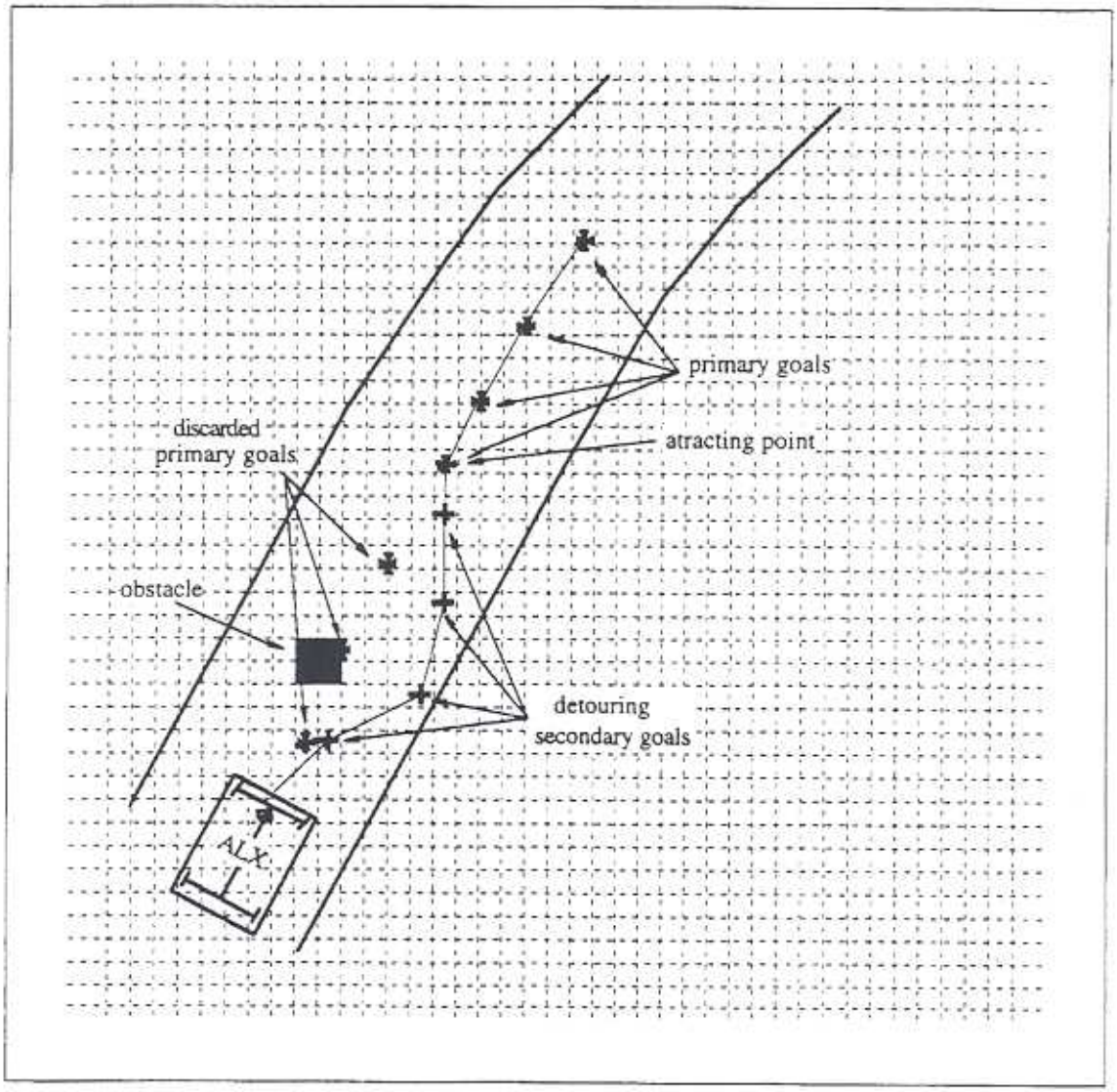

Figure 3-3. Searching for alternative path.

D. Overall Control Algorithm

As shown in Figure 3-4, the overall control algorithm of the ALX is based on the operation of goal stacks. The ALX attempts to travel towards the top goal on the primary forward goal stack, which is constructed from the vision information, and then constructs a detouring goal stack when an obstacle is detected by using the virtual map.

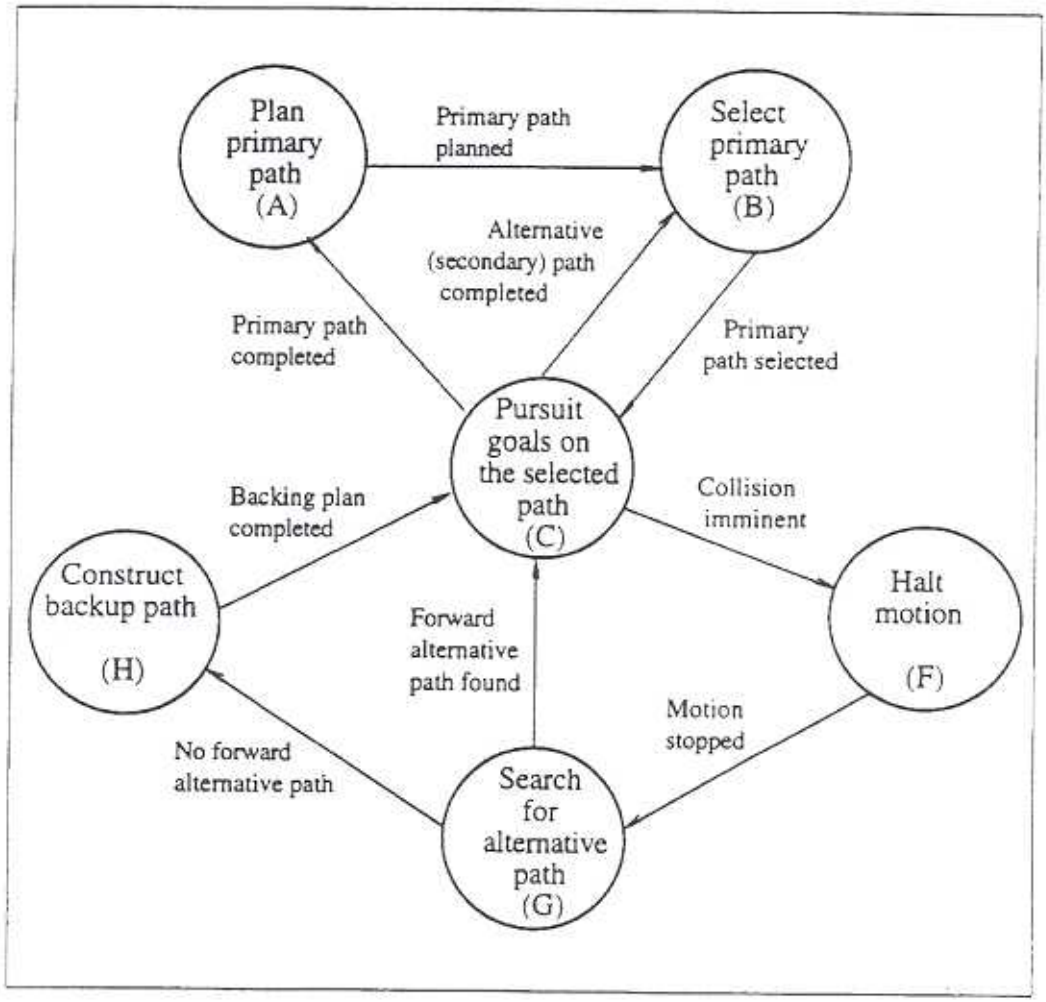

Figure 3-4. Strategy transition state diagram of the overall control algorithm.