1-9-2008

A software architecture for mobile robot navigation

Phillip J. McKerrow
University of Wollongong, phillip@uow.edu.au

Sherine M. Antoun
University of Wollongong, sherine@uow.edu.au

A software architecture for mobile robot navigation

Abstract

Directed sensing poses the problem of sensing in specific directions in synchronisation with robot motion while avoiding collisions with objects in other directions. The rebuild of an outdoor mobile robot, with the goal of mimicking a blind person navigating with echolocation, has provided the opportunity to experiment with a state machine based software architecture for landmark navigation. In this paper, we discuss the rebuild of the robot, the software architecture and an initial experiment in collision avoidance.

Disciplines

Physical Sciences and Mathematics

Publication Details

This conference paper was originally published as McKerrow, PJ, Antoun S, A software architecture for mobile robot navigation, Proceedings TAROS'08, Edinburgh, 1-3 September 2008, 185-192.

A software architecture for mobile robot navigation

Phillip McKerrow, Shérine Antoun and Patricia Worth[1]

Abstract— **Directed sensing poses the problem of sensing in specific directions in synchronisation with robot motion while avoiding collisions with objects in other directions. The rebuild of an outdoor mobile robot, with the goal of mimicking a blind person navigating with echolocation, has provided the opportunity to experiment with a state machine based software architecture for landmark navigation. In this paper, we discuss the rebuild of the robot, the software architecture and an initial experiment in collision avoidance.**

I. INTRODUCTION

Blind people have demonstrated exceptional ability when navigating with mobility aids using **C**ontinuous **T**ransmission **F**requency **M**odulated (CTFM) ultrasonic sensing [1, 2]. Our research goal is to understand how they use echolocation to navigate by mimicking their navigation with a mobile robot. Achieving this goal requires a software architecture that supports directed sensing synchronised with robot motion.

In this paper we describe the rebuild of a mobile robot for this project. The rebuild involved updating both the computer system and the navigation software. To demonstrate the capability of a state machine based approach to directed sensing, we present the results of a simple collision avoidance experiment. The purpose of this experiment is to study the question of whether the robot can avoid collision in a particular direction while only sensing in that direction occasionally. It confirms that the software architecture is suitable for exploring a new approach to navigation that mimics how humans find their way.

In heavy industry it is common for a machine to outlive its control system. A cold rolling mill in a local steel plant was originally installed in 1955. Its control systems have been updated every decade. It continues to produce high quality cold rolled steel. Fifty-three years after initial installation most of the original mechanical components and electric motors are still in continuous use.

Many of us have mobile robots in our laboratories that were switched off years ago. While possibly in need of a little maintenance, the mechanical components and motors are still operational. Many of the sensors still work. But the computers, software and interface electronics have failed and spares are unavailable. After the last student finished his project, the robot was turned off and new students are not interested in projects with obsolete equipment.

1. Authors are with the School of S Computer cience and Software Engineering, The University of Wollongong. email: phillip@uow.edu.au

In 1998, we built an outdoor mobile robot (Titan), from a 4-wheel drive wheelchair [3]. As a Segway™ RMP 400 4-wheel Robotic Mobility Platform [4] for £16,000 plus customs and delivery was beyond what we could afford, we were faced with the challenge of either continuing to work with obsolete and failing hardware and software or updating.

The decision to rebuild (Fig. 1.) resulted in a requirement to develop new software from the old to work with new input/output (i/o) drivers. Also, it created an opportunity to improve the software architecture and libraries. As we had developed our own software we chose to go the code reusability route [5] rather than re-code in a standardised mobile robot language [6] such as CARMEN, [7].

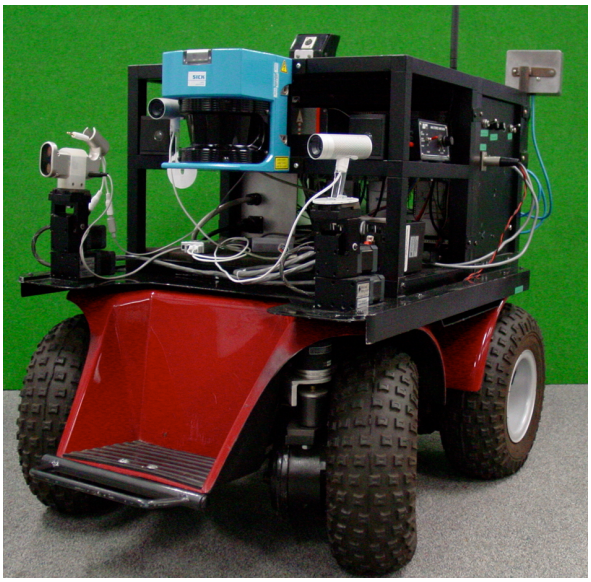

Fig. 1. Titan 4-wheel drive outdoor robot with K-sonar sensor attached to pan and tilt unit on left.

The software for echolocation and robot navigation had been written by research students from scratch. It was monolithic and difficult to understand (one student gave up and quit his PhD). To overcome this problem we put a lot of work into developing libraries of low-level routines with program templates as a starting point for new applications. We wanted to keep the libraries and redesign the templates to enable a new set of research projects.

In the next three sections, we review the ability of a blind man to navigate with echolocation. A key skill that blind people develop is directed sensing by synchronising sensor panning with their body motion. The implications for mobile robot navigation are explored in Section V. From these, we developed the design requirements for the mobile robot. Its rebuild

McKerrow, P.J., Antoun S. and Worth, P. 2008. A software architecture for mobile robot navigation, Proceedings TAROS'08, Edinburgh, September 1-3, pp 185-192.

is described in Sections VI - IX. Finally, Section X reports on using the software architecture to control a simple experiment where collision avoidance is achieved with minimal sensor data.

II. Navigation by blind people

> *"... I, as a thirty-six-year-old blind person, am able to thread my way through heavy pedestrian traffic smoothly, gracefully, and without collision, and can find an empty seat on the bus, an empty desk in a classroom, or an empty booth or table in a restaurant ..."*
> Gissoni, 1966

Fred Gissoni [1] is a blind man who has learned to navigate among sighted people using echolocation. Other blind people have also achieved significant navigation ability with CTFM ultrasonic mobility aids [8]. With the audio information produced by these aids they can perceive the geometric structure of the environment with sufficient clarity to enable them to move among both stationary and moving obstacles.

The abilities that blind people develop to navigate include: detection of object presence, recognition of object type, perception of object motion, prediction of their own motion, directed scanning of the sensor and synchronisation of perception with motion. Detecting object presence can be as simple as checking that there is an empty space to move into, to as complex as detecting that a door is ajar or a chair is empty.

Perceiving an object's type involves recognizing that the object is a wall, a chair or a pot plant for example. Perceiving object motion is required to follow a person or to walk along a busy footpath. Predicting your motion is planning your next move(s) before you move. Directed scanning is sensing in the direction that you are going to move prior to moving. Synchronization of sensing with motion is required to successfully navigate at a walking pace in any cluttered environment.

Fred Gissoni made 10 audio lessons on how to use a CTFM sensor to navigate, because the sighted trainers of the blind did not understand how to use it. Sighted trainers cannot see the ultrasonic beam and struggle to identify which parts of the environment produced the echoes that they hear in the audio output of the CTFM mobility aid.

III. Echolocation

Echolocation is a sense of perception that is normally associated with bats not with humans. Bats provide a model of what is possible. By contrast, people were not created with echolocation as a normal sensing skill, so they have to learn it. Because echolocation is outside the experience of sighted people, many are sceptical of the ability of blind people to safely find their way with it.

Some blind people have learned to sense the environment with audible clicks vocalised from their mouth [9, 10]. As these are at audio frequency, their ability to discriminate between objects is probably limited to detecting large geometric differences where the range difference is sufficient to create a time delay sufficient to create reverberation.

CTFM ultrasonic sensors work at an order of magnitude higher frequency and thus should give an order of magnitude better discrimination. They continuously sweep down from 100 to 50 kHz [8]. Echoes from objects are demodulated with the transmitted signal to produce an ensemble of audio tones in the range 0 to 5 kHz. The frequency of each tone is proportional to the range to the surface feature that reflected that component of the echo.

In previous research [11], we demonstrated recognition of plants [12] and recognition of surfaces based on roughness [13]. All were done by extracting features representing surface geometry from the echoes. As the best results were for classification of rough surfaces, we are conducting research into navigating paths based on sensing roughness. It is for this research that we are upgrading our outdoor mobile robot Titan.

IV. Directed sensing

Blind people purposefully pan the CTFM mobility aid to hear echoes from objects in the environment so that they can both recognise those objects and determine the spatial relationships between them. Carefully thought-out movement of the sensor in synchronism with the motion of their body becomes a habit that helps them turn meaningless tones into meaningful sounds from which they build spatial maps of the environment.

An example from Gissoni's lessons is how to detect a corridor [14]. Hold the sensor horizontal and pan slowly from side to side. Near the ends of the pan you should hear tones caused by echoes from the walls. Straight ahead you should hear nothing. Adjust the angle over which you pan so that you can clearly separate the walls from the corridor in the middle. Continue the horizontal pan and slowly tilt the sensor down until the gap in the middle is replaced by a tone of a different quality (a soft swishing sound). This signal is the floor. Finally, tilt the sensor back up, while horizontally scanning, until the signal from the floor is barely audible. Then when a person walks towards you their echo will be clearly distinguishable from the softer swishing sound of the floor.

Based on this description we set out to develop a software architecture that enables directed sensing in synchronisation with robot motion. As the robot moves, the sensor should be panned so as to detect the empty space in the corridor through which the robot plans to pass. This involves detecting the walls on both sides to define the corridor, detecting the floor to ensure there are no down steps or low lying objects and detecting objects in front to avoid collision.

V. Mobile robot Navigation

To give a mobile robot the ability to navigate like a human will be a major innovation. We have chosen not to use a common SLAM (simultaneous localisation and mapping) approach with Kalman filters because there is no evidence that the human brain uses Kalman filters and our goal is to mimic human navigation.

Second, while ultrasonic sensor based navigation systems have been developed that use Kalman filters for odometer correction [5] and localisation [16], they have to reduce the sensed information to point features in order to use the Kalman filter. Observations of blind people navigating indicate that humans use an alternate approach that relies on the quality of their sensing of landmarks. Echolocation data provides a richer description of objects than points and we wish to use that additional information in the navigation system.

Achieving similar navigation capability to people with a mobile robot requires the ability to sense the location of objects, to track those objects, to recognise them and to decide which objects are important in the current navigation task. While many of our ideas build on prior research, combining rich echolocation data with directed sensing makes this approach new.

Part of our research is to determine what information is useful for navigation and how to represent it in an echolocation map. We have observed that blind people, like sighted people [17], increase their speed of navigation by reducing the sensed information to the minimum required by the task enabling them to increase the update rate by panning over a smaller angle.

To achieve the goal of programming a mobile robot to mimic human navigation, we are redeveloping the software on Titan to achieve the following navigation architecture. At the top level a command is given to carry out a task (such as "fetch a hoe"). Achieving this command requires a number of functions to be performed, including a navigation function (such as go to the garden shed).

First, we decompose this navigation function into a sequence of simpler navigation tasks from a set of available tasks stored in a map in a graph data structure [18]. We are examining how to represent this sequence as a set of connected states as a solution to the problems of planning and tracking the robot's motion from one navigation task to the next. For example, a navigation task may be to navigate along a brick path from the garden to the shed. While navigating the path the robot is in the brick path navigation state. The first commercial mobile service robot, the Helpmate, decomposed navigation tasks into sequences of hallway navigation commands [19].

Based on its current perception of where it is on the path, the robot will predict where it will travel in the next three intervals of time if it continues along the current trajectory. An interval includes the time to pan the sensor and to scan the environment (a form of model predictive control [20]). Then it will move along the trajectory specified in the first interval, if the space is clear.

At the same time, it will direct the sensor to view the region in the second interval (i.e. predict where to sense) [21]. On a path, the sensor has four sensing positions to choose from: ahead (empty space), right border, ahead declined (path) and left border. To achieve directed sensing the robot will pan the sensor to scan each of these regions.

The robot will synchronise the panning speed with its velocity so that the space in the next interval is sensed before the robot attempts to move into it. Then it will adjust its velocities so that it continues to track down the path by turning to avoid obstacles and slowing down in narrow spaces.

VI. Robot rebuild

We ran tests to determine what had failed, what was obsolete and what was operational. We found that the wheel-chair mechanics, motors and power amplifiers worked. The pan and tilt units, the encoders and the gyro-stabilised compass were also working.

The traction batteries needed replacing, a PCI interface card had failed, and other PCI interface cards were not supported in new software. We had written the control software in LabVIEW™ 5 running on a G3 Macintosh™ Powerbook under Mac OS9. The interface cards were PCI cards plugged into a Magna™ expansion chassis. We had modified it for battery operation using aircraft quality inverters. The inverters were operational.

In the last decade, computer technology has changed dramatically. As we wanted to retain as much as we could of the software, particularly libraries that we had thoroughly tested, we chose to update to LabVIEW 8.5. But this also meant changing to new serial drivers (VISA) and to new hardware drivers (NI-DAQmx_Base).

Titan was controlled by an on-board Macintosh Powerbook. So all software processes (development, control, data collection and some analysis) were performed on the robot. However, this meant that when the robot was moving we had to run after it to see the graphical user interface and to change command parameters.

In our new design (Fig. 2.) we have chosen to control the robot with an Apple™ Mac Mini with no keyboard, mouse or monitor on the robot. Communication with the user is done via a remote Apple MacBook over an IEEE802.11g wireless adhoc (peer to peer) network using remote desktop software. We wrote an application to set up the Mac Mini as the network server on start up. We also added an aerial to extend the range of the network (Fig. 2.).

Setting the robot up as the network host enabled us to stop all other network traffic from it, particularly applications that go looking for a network. We had

found that applications that went looking for a network and couldn't find one (because the robot was in a field) hung the network drivers and caused delays in communication to the remote desktop. These delays lasted until the drivers timed out so they were long enough to be dangerous. Because the Mac Mini on the robot has no connection to any network, there is no network for these application to use so they do not call the drivers to access a network.

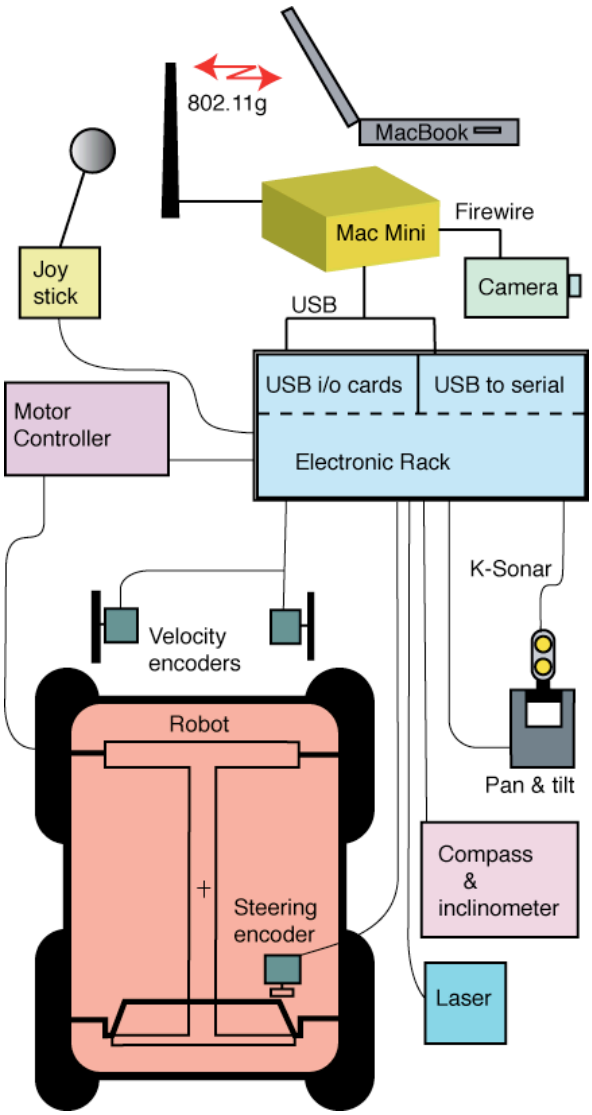

Fig. 2. Block diagram of Titan's sensors, actuators, and electronics.

The user of the robot can sit in one spot with access to all software running on the Mac Mini while the robot runs around the field. Software development tools and control, data collection and analysis software all run on the Mac Mini. This has proved to be a very workable arrangement. Now we only run after the robot to push the emergency stop button when experiments result in a potential collision.

Both computers run Mac OSX. It has useful features including remote desktop and real-time threads. The main issue caused by the operating system upgrade was that the i/o drivers were replaced.

The serial PCI card that we used does not have a Mac OSX driver, so now all serial inputs are handled with USB to serial converters. Quality converters (e.g. Keyspan™) have individual serial numbers so they can be uniquely identified by the software. Cheep converters don't, which creates problems when a USB card is unplugged because the USB dynamically reconfigures and the software relies on serial numbers to identify individual serial ports.

With one PCI card failed, one without a driver, and the Magma PCI expansion chassis having to be replaced to work with Mac OSX, we decided to do all the i/o with USB cards and Firewire for vision. We had had very good experience with the Keyspan USB to serial cards. On another project we had used one to read a 25 character packet from an IMU every 20msec, with excellent performance.

The change to USB i/o required rewiring the interface to the sensors and actuators. The change to a Mac Mini required the installation of an additional power inverter. One issue that we have to investigate further is the significant increase in power consumption. We get about an hour of continuous movement from a full battery charge.

VII. Software Libraries

We put a lot of work into developing and testing libraries of low-level routines for the previous version of Titan [22]. We wanted to keep what we could of these libraries. We found that, in the main, we could keep the logic but we had to rewrite the software that read data from the sensors and wrote commands to the actuators. The new LabVIEW hardware drivers are lower-level than the previous ones, so we had to develop routines to configure the USB i/o ports.

We developed 3 libraries: an In/Out library, a Control library and a Feature library. The In/Out library (Table 1.) includes routines to read all sensors and calculate values in physical units. Some, e.g. the Steering angle, return a single reading. Others, e.g. Odometer, run as a parallel process and produce a set of readings every 100msec.

Two important functions in the In/Out library are Logic and Control. The Logic function ensures the safe operation of the robot. It interacts with the hardware logic to switch to computer control and will return control to manual when any stop button is pushed. Also, the logic hardware switches the analogue hardware to choose manual or computer calculated outputs from the Control function to the motor power amplifiers.

Every 100 msec, the Sonar Producer process reads 1024 echo samples for each frequency modulated (fm) sweep transmitted by the K-Sonar [2] and places this

echo array onto a queue for a sonar consumer to read and process. The Feature Library has routines for calculating the Power Spectral Density of the echo and extracting features from it for object recognition and environment mapping.

Table 1. Applications and libraries developed to control Titan.

Applications	Control Library	In/Out Library
Test sensors and actuators	Linear velocity controller	Logic
Controller tuning	Angular velocity controller	Control
Square	Bearing controller	Odometer
Corridor follow	Bearing fusion	Pan Tilt
	Steering controller	Sonar Producer
	PID loop	Video grab
	PID loop cyclic	Compass
	Motion	Steering angle

The Control library groups the Logic and Control functions into a Motion process that runs in parallel with other processes to provide safe control in applications. The Control library includes closed loop controllers for linear velocity, angular velocity, steering and bearing control. Two PID (Proportional Integral Derivative) functions are included. A separate PID controller is used for bearing because the feedback is cyclic.

Bearing varies clockwise from 0^o (North) to $<360^o$. Thus, in turning left from north the value jumps by 360. The gyro-stabilised compass produces this step. The odometer calculation of bearing was modified to produce this step, also. This step results in a step in the error between reference and feedback in the region of North which the PID controller has to handle.

VIII. Applications

The first application to be written was to enable testing of all sensors and actuators. It enables manual operation of each actuator from a graphical user interface and manual inspection of all sensor values. This application proved invaluable in testing the In/Out library and is regularly used to confirm that the robot is operating correctly.

The second application is for tuning the control loops. Again a graphical user interface is used to control the robot, change tuning parameters and observe step responses. We jacked the robot up on blocks for testing of this software and for initial tuning of the loops. While on the blocks we were able to test the parallel operation of the processes, the synchronisation between the processes and the all important global stop function (stops all software).

However, loops such as bearing control can only be tested and tuned on a moving robot. Due to the size of the robot, we had to go outdoors onto a sports field to tune the loops. As expected, the loops tuned on the blocks were over damped when driving on grass.

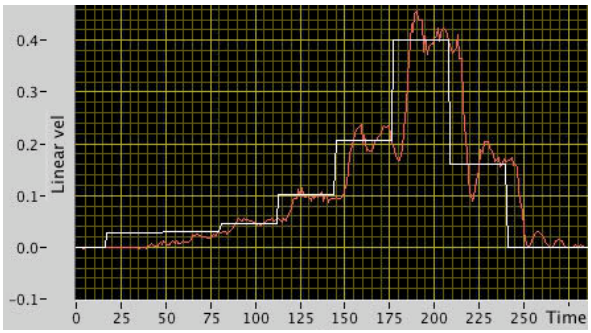

Fig. 3. Linear velocity reference and actual linear velocity showing change in tuning at different velocities. Divide numbers on horizontal axis by 10 to get seconds.

Tuning has not proved to be as easy as we hoped (Fig. 4.). The dynamics of the robot changes with the energy level of the batteries, the speed at which it is travelling, and the surface over which it is rolling. Loops tuned outdoors on grass are under damped when the robot is traversing carpet in the laboratory.

Another problem that has to be considered when tuning is that traditional methods of tuning are for small signal linear systems. Typically a 10% step is added to the reference to study the response over an operating region that is assumed to be linear. But a mobile robot is often given large signal commands, such as turn through 30 degrees or speed up by 50%. One way to reduce this problem is to tune for small signal response at the normal linear velocity and ramp any changes in references so that they appear as a series of small steps.

To answer the navigation research questions, we are developing corridor and path navigation applications. The corridor navigation application (Fig. 5.) includes 6 parallel processes that communicate through common data (LabVIEW Globals). It includes a global stop function that starts and stops all the other processes and a state machine. The state machine calculates the velocity and steering references based on the range data from the sonar. The Linear Velocity control is synchronised with the odometer to minimise delays between reading and controlling the velocity.

Directed sensing of a K-sonar sensor is controlled with a pan and tilt unit whose motion is specified by the state machine. The sensor direction is combined with the sonar range readings and a time stamp to

produce a map of the region scanned by the sensor. The state machine is synchronised with the sonar update to minimise delays between range reading and velocity control.

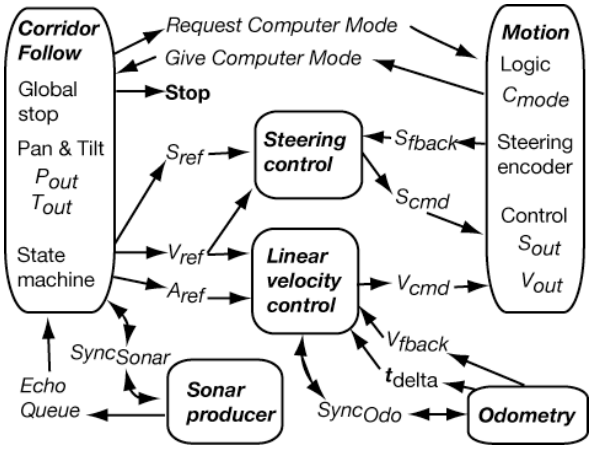

Fig. 4. Architecture of corridor following navigation software, with 6 parallel processes communicating through common variables, where V_{ref} = velocity reference, V_{fback} = odometer velocity, V_{cmd} = output of velocity control loop, V_{out} = velocity command to motor controller, A_{ref} = acceleration per time step, S_{ref} = steering angle reference, P_{out} = pan angle reference, T_{out} = tilt angle reference.

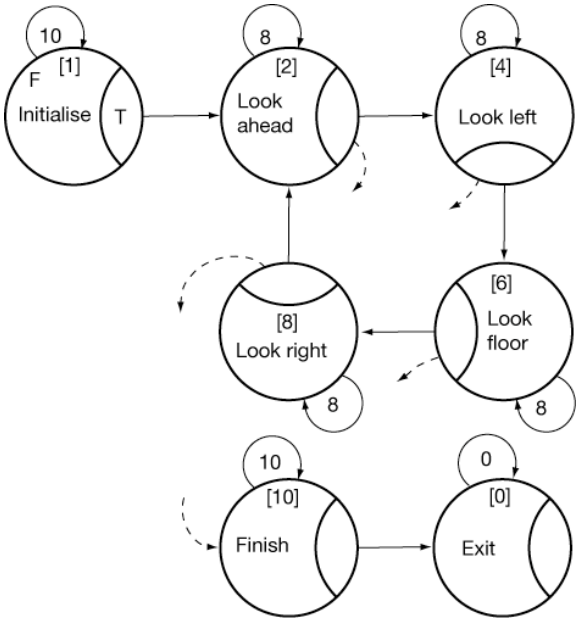

Fig. 5. Time based state machine for path scanning. [Value] is the number of 100 msec time steps in state.

IX. State Based Architecture

A state machine was first used as a high-level controller for mobile robot navigation in the ROBOL/0 language in the Yamabico robots [23]. The first program that students had to write was a state machine to drive the robot in a square. We are using a state machine to combine directed sensing with motion control.

One example of a directed panning pattern is the corridor following pattern where the sensor pans front, left, floor and right in sequence. When following a corridor the left and right pans are at waist level to detect walls. By contrast when following a path they are depressed to path level to detect the path edges. Another is a wall follow pattern where the sensor pans front, left, and floor.

The state machine is time based. It calculates a new state every time step (100msec). Other processes run during the thread wait between time steps. The time that it stays in a state can be determined either by time or by events. In the case of a sensor panning pattern it is determined by time.

The corridor following state machine in Fig. 5. has 7 states. In each state it loops for a number of time steps (change state = F) and then on the last step (change state = T) it calculates the output commands to the pan and tilt unit and the robot to move into the next state.

Table 2. State machine set up for collision avoidance experiment. Key: C = copy previous value, E = go to final state, F = false, L = loop at state, T = true, Tr = transition to next state, V_T = velocity target mm/sec, En = enable output to Pan and Tilt unit

State	Next	Time	Event	V_{ref}	A_{ref}	S_{ref}	En	P_{out}	T_{out}
1L	1	10		0	0	0	F	0	0
Tr	2			C	C	C	T	0	0
2L	2	8		C	C	C	F	0	0
Tr	4			% V_T	0.1	0	T	50	0
E	10		T						
4L	4	8		C	C	C	F	0	0
Tr	6			C	C	C	T	0	-20
E	10		T						
6L	6	8		C	C	C	F	0	0
Tr	8			C	0.1	C	T	-50	0
E	10		T						
8L	8	8		C	C	C	F	0	0
Tr	2			C	C	0	T	0	0
E	10		T						
10L	10	2		0	0.1	0	T	0	0
Tr	0			C	C	C	F	0	0
0L		2		0	0.1	0	F	0	0

State 1 is an initialisation state that initialises the robot to be ready to start moving. States 2, 4, 6, and 8 perform the directed panning and robot motion. At any time the user can press a switch in the GUI to stop the state machine. At the end of the current state it transitions to state 10 which shuts the robot down, and then it exits the state machine (state 0).

X. Avoiding Collisions

A concern with directed sensing is that the robot may collide with an object in one direction while it is sensing in another direction. When a particular direction is only sensed occasionally, will the robot detect an object in that direction in time to avoid collision?

To answer this question and in the process determine whether the collision avoidance that blind people achieve with directed sensing can be done with a mobile robot, we set up a collision avoidance experiment. The parameters of this experiment were chosen to represent a "hard" case scenario so that if the robot can be programmed to achieve it then it can be programmed to achieve simpler cases.

The parameters of the experiment are:

- The sensor pans a region in front of the robot 100° wide and 20° high using a corridor follow panning pattern of look ahead (0°, 0°)(pan, tilt) look left (-50°, 0°); look floor (0°, -20°); look right (+50°, 0°); repeat. A complete pan cycle involving 4 motions with stops between them takes 3.2 seconds. An angle of 50° to a surface is greater than the 40° used for excellent recognition of rough surfaces [13].
- The minimum amount of sensor data is used: range to the nearest object from a single echo during the 3.2 second pan cycle. The sensor scan used is the one pointing directly ahead of the robot. As the sensor emits an frequency modulated sweep every 100 msec, only 1 out of 32 echoes is used. Also the ultrasonic sensor is set to minimum range where it senses up to 2 metres.
- The robot's linear velocity target is set to 400 mm/sec. While this is considerably less than its maximum velocity of 1 m/sec, it allows us to conduct the experiment in the laboratory and not have to go outside. So in 3.2 seconds the robot travels 1.28 m. As a result, at target velocity, the sensor reads the echo at least once while travelling the 2m maximum range of the sensor.
- The robot's linear velocity reference is calculated based on the distance to the object reported by the sensor. It is set at 100% of target velocity for ranges greater than 1.7m, at 0% for ranges less than 0.8 m and at a percentage between 0 and 100 for ranges between 0.8 and 1.7m.
- As only a single sensor reading is used, the object is placed about 4 m directly in front of the robot, out of range of the sensor, but within the sensing cone of the sensor when it is in range. The object is a tall (780mm high, 570 mm wide) thin (55mm) block of dense styrofoam.

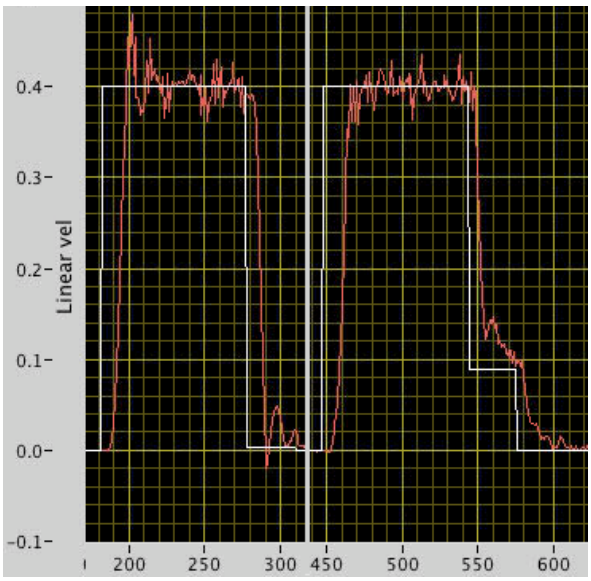

Fig. 6. Two runs of collision avoidance experiment. The robot stops before hitting the object. Vertical axis is in m/sec and horizontal axis is in sec*10.

The state machine is set up as shown in Table 2. In state 1 it initialises the robot. At the end of state 2, it reads the range directly ahead, calculates the linear velocity reference and issues the command to pan left. The robot accelerates to the reference velocity and continues to pan under the control of the state machine. Each time it reads directly ahead it recalculates the linear velocity reference. When it detects the object it slows to a stop, usually with 2 steps in the reference (Fig. 6.).

On every experiment that we did, when the robot detected the object it stopped. However, on the initial runs, just before the robot stopped it would nudge the block and knock it over. On the next pan, the robot would detect empty space and accelerate again, running over the object, and coming to a stop with its bumper against the filing cabinet at the end of the laboratory (Fig. 7.).

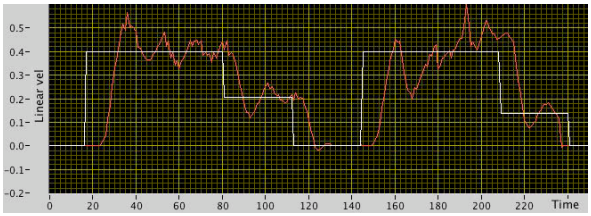

Fig. 7. Continuous run where robot knocks object over and accelerates again. On the right the robot collides with filing cabinet. Vertical axis is velocity in m/sec. Divide numbers on horizontal axis to get time in seconds.

The robot was stopping too late because the linear velocity control was taking longer to settle than we had allowed for. The gains of the controller were tuned when running on grass. We reduced the gains to obtain less overshoot on carpet and we increased the range for 0% from 600 to 800mm. Most times it stops a small distance from the object (up to 100mm).

XI. Conclusion

The above experiment demonstrates that the software architecture enables the development of navigation programs to use directed sensing. Also, it shows that it is possible to avoid collision when only sensing occasionally in the direction of travel. At other times, the sensor can be panned to sense in other directions, for example to detect the border of a path to control steering.

The above experiment is a hard case with minimal sensor information, limited sensor range, and maximum update time. It can be made more robust. First the update time can be halved by looking for objects in echoes when sensing the floor, and using their range when calculating the reference velocity as well. Second, the amount of sensor information used can be increased significantly by using echoes from several sensor scans both sides of the direction of interest and by using more features from the echoes.

Third, in many tasks the panning angle can be decreased from 50^{o} to reduce the total cycle time, particularly when you only want to detect the presence of a path border and not recognise whether it is grass or leaf mulch, etc. Fourth, the range of the sensor can be doubled to 4 m but at the cost of doubling the update time to 200 msec.

To achieve our goal of mimicking human navigation using echolocation we have a lot more work to do. The state machine has to be made easier to program to make setting up experiments easier. The next experiment we plan to do (corridor following) requires the fusion of echo data from all the echoes in a pan cycle to control both linear velocity and steering. Then we have to add a higher level, where several navigation functions are combined to achieve a navigation task.

References

[1] Gissoni, F., 1966. My "Cane" is Twenty Feet Long, The New Outlook for the Blind, February.

[2] BAT, 2008. Bay Advanced Technology - http://www.batforblind.co.nz/

[3] Ratner D. and McKerrow, P.J. 1999. Dynamics of the Titan four-wheel drive mobile robot with floating Ackerman steering, Proceedings Australian Conference on Robotics and Automation, ARAA, Brisbane, pp 144-149.

[4] Segway 2008. Robotic Mobility Platform (RMP 400) http://www.segway.com/business/products-solutions/robotic-mobility-platform.php

[5] Cote, C. Letourneau, D. Michaud, F. Valin, J.-M. Brosseau, Y. Raievsky, C. Lemay, M. Tran, V. 2004. Code reusability tools for programming mobile robots, Proc IROS'04, vol 2, 28 Sept - 2 Oct, pp1820-1825.

[6] Montemerlo, M., Roy, N. and Thrun, S. 2003. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In P*rocIROS'*03.

[7] CARMEN, 2008 - http://carmen.sourceforge.net/

[8] Kay, L., 2000. Auditory perception of objects by blind persons, using bioacoustic high resolution air sonar, Journal of the Acoustical Society of America, Vo11O7,No 6, June, pp 3266-3275.

[9] Nagel, T. "What is it like to be a bat?" *Philosophical Review*, vol. 83, pp. 435-450, 1974.

[10] Kish, D. and Bleier, H. 2000. ECHOLOCATION: What It Is, and How It Can Be Taught and Learned, Presented CAOMS, http://www.worldaccessfortheblind.org/publicationandInfo.htm

[11] McKerrow, P.J. and Antoun, S.M. 2007. Research into Navigation with CTFM Ultrasonic Sensors', ION 63rd annual meeting, Cambridge, Massachusetts April, pp 674-680

[12] McKerrow, P.J. and Harper, N.L. 2001. Plant acoustic density profile model of CTFM ultrasonic sensing, IEEE Sensors Journal, vol. 1, no. 4, pp. 245–255.

[13] McKerrow, P.J. and Kristiansen, B.E. 2006. Classifying surface roughness with CTFM ultrasonic sensing, IEEE Sensors Journal, vol. 6, no. 5 (October), pp. 1267–1279.

[14] Antoun, S.M. and McKerrow, P.J. 2007 Perceiving A Corridor With CTFM Ultrasonic Sensing, in: Proceedings of Australasian Conference on Robotics and Automation, Brisbane, 2007.

[15] Kurz, A. 1996, Constructing maps for mobile robot navigation based on ultrasonic range data, IEEE Trans Syst Man Cybern B Cybern, Vol 26(2), pp 233-42.

[16] Chong, K.S. and Kleeman, L. 1999. "Feature-based mapping in real, large scale environments using an ultrasonic array", International Journal Robotics Research, Vol 18, No. 1, Jan 1999, pp. 3-19

[17] Berthoz, A. 2007. Simplifying principles of perception, action, locomotion and navigation, Plenary talk, Digest IEEE ICRA'07, Rome, April, pp. xviii–xix.

[18] Antoun, S. and McKerrow, P.J. 2006. Landmark navigation with fuzzy logic, Proceedings ACRAA'06, Auckland, December (CD Rom).

[19] Evans, J., Krishnamurthy, B., Pong, W., Croston, R., Weiman, C. and Engelberger, G. 1989. Helpmate : A Robotic Materials Transport System, Transitions Research Corporation.

[20] Wei, S., Zefran, M., Uthaichana, K. and DeCarlo, R.A. 2007. Hybrid model predictive control for stabilization of wheeled mobile robots subject to wheel slippage, Proceedings IEEE ICRA'07, Rome, April, pp. 2373–2378.

[21] Berthoz, A. 2000. The Brain's Sense of Movement. Harvard Univ. Press.

[22] McKerrow, P.J. and Ratner, D. 2002. Calibrating mobile robot odometry with ultrasonic sensors, in: Proceedings of IROS'02, Lausanne, October, pp. 859–864.

[23] Suzuki, S. and Yuta, S. 1991. Analysis and Descriptioin of Sensor Based Behavior Program of Aotonomous Robot Using Action Mode Representation and ROBOL/0 Language, Proc. IROS'91, Nov 3-5, Osaka, pp 1497-1502.