Towards A Real-Time Architecture for Obstacle Avoidance and Path Planning in Mobile Robots

M.D. Adams, Huosheng Hu and P.J. Probert

Robotics Research Group
Department of Engineering Science
University of Oxford , U.K.

1 Abstract

We describe the design and partial implementation of a real time architecture for a mobile robot, aimed particularly towards a vehicle developed for factory automation.. We develop a layered design to equip the robot with a number of behavioural competences. We examine sensing and a potential field algorithm especially to achieve modification of behaviour at a speed close to the robot's operational speed. We show how the layered architecture interfaces to the original on-board architecture, which provided sophisticated localisation but no ability to deal with environmental exceptions.

2 Introduction

This paper describes our progress towards an architecture for real time navigation and obstacle avoidance for mobile robots used in a semi-structured factory or uncluttered office environment. We examine particularly the following issues:

- The interaction between a real mobile robot's control system and a newly developed obstacle avoidance capability.
- Architectures and technologies for appropriate sensors.
- The implementation of different behavioural capabilities through sensor-driven algorithms.
- A real-time potential field algorithm, developed specifically for mobile robot navigation whilst on the move.

We outline an overall distributed architecture, distributed both in sensing and in processing. A crucial feature is the provision of ***distributed, locally intelligent*** sensors [15].

The most obvious requirement in exception handling in this type of application is the capability to efficiently manoeuvre around unexpected obstacles in the vehicle's pre-planned path. Many researchers have previously treated the problem of obstacle avoidance in a purely static sense, meaning that upon detection of an obstacle, the robot stops and thinks before continuing any further manoeuvres [7], [8]. Our goal is to realise, through simple algorithms and fast sensing, obstacle avoidance while the vehicle moves at its operating speed of around 0.5m/sec. In detailed path planning we would expect that adjustments to the path might have to be made about every 5-10cm; ie every 100-200msec. This implies a bandwidth of 5Hz for the processes of collecting sensor data, interpreting it in terms of a new vehicle position and communicating with the vehicle. Allowing for any complexity in algorithmic processing, this is a stringent requirement in bandwidth. To achieve this, we are implementing a layered control system, which we examine this in the next section.

3 Overall Architecture

Our architecture draws from Brooks's work on subsumption architectures [5]. There are two reasons for our choice. First, building up behavioural competences is a natural way to implement a specification defined in terms of behavioural objectives. Secondly the possibility of using a distributed architecture avoids the bandwidth restrictions of a central controller and is a natural dual to our distributed sensing architecture. Finally distribution increases robustness.

We describe here our progress towards the design and construction of a three levelled system: level 2: Path plan from previously stored model, level 1: Route follow, and level 0: Reflex obstacle avoid. The details of level 2 are outside the scope of this work and are being examined by other workers at Oxford [3], [4]; we are involved only with its interaction with other layers. Level 1 attempts to reach a goal in the presence of obstacles whereas level 0 takes over using simple algorithms if an obstacle is detected at very close range. Level 0 is almost acting as a bumper, but attempts to take simple reflex-type action to get around an obstacle, or to back away from an encroaching target. Each level receives input sensor data, carries out its relevant processing and attempts continuously to output signals to the vehicle. The layers operate asynchronously and do not communicate with each other. All continuously send control signals to the vehicle, based upon their sensor observations, even though only one level actually controls the vehicle at a time. Individual layers work on individual goals concurrently. Currently level 0 has been implemented and the algorithms for level 1 developed.

We describe in detail in sections 4 and 5 the internal machinery within our levels of competence. We build in the 'upward' manner, as stated above, in order to allow the desirable effect of the 'graceful degradation' in the behaviour of a mobile. The algorithms and sensors in both levels are designed to be robust to individual errors in reading. Both sensors and algorithm can be 'fooled' by certain environmental/obstacle configurations (for example, highly light absorbing obstacles will be undetected by infra-red range finders, and cluttered environments can generate local minimum regions, which will falsely attract the robot). Since level-0 also continuously monitors the environment for obstacles, it will inhibit the output of level-1 from the robot, and replace it with its own, in the event of a mobile positioning itself near to an obstacle. Level-0 attempts only to avoid simple obstacles (using a sonar array), hence our mobile operates under a new 'degraded' behaviour.

Because of the highly parallel nature of our control strategy, we note here our choice of processor within the architecture - namely the Inmos T800 Transputer. As well as being a fast and powerful processor, the transputer facilitates the design of distributed systems with small, intercommunicating processes. In addition, expect to be able to put all the lower level tactical reasoning on board, and ultimately much of the higher level capability to provide a truly on-board architecture.

4 Sensors and Sensor Architecture

4.1 Requirements of Sensors

We define a sensor as consisting of a transducer, a data acquisition facility and a facility for local interpretation of data. The most crucial aspects of the sensors are reliability and bandwidth. Given these requirements, which sensors should we use? Cameras give us very full information but cannot are relatively low bandwidth because of the extensive computation required in data interpretation. Linear array cameras remove one dimension and are being investigated for real-time operation with some success [6]. However for simplicity and high bandwidth we are looking at implementations based on small, low cost distributed range sensors.

4.2 A Layered Perception Architecture

Here we present a layered perception architecture, shown in Figure 1. It is a high bandwidth sensing architecture based on a Transputer network. It supports active sensor control and distributed sensor data fusion so that real time control of a mobile robot can be achieved [12].

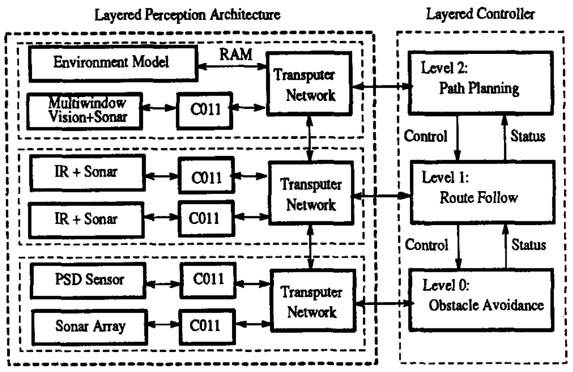

Figure 1: Block Diagram of Layered Architecture for Sensing

There are four crucial considerations in our design:

- Different reaction times and scanning range have been chosen for each sensing layer to meet the requirement of each control layer with different response speed.
- Where feasible multiple sensors are used in each layer so that we can combine the data from each sensor.
- The sensing layers are operated independently in parallel.
- Low levels of competence have no access to an environment model.

4.3 Sensor Implementation

We emphasise the importance of local intelligent sensors through incorporating a transputer within each sensor for local control and information processing. The concept of networks of intelligent sensors is being developed by Durrant-Whyte [15] in his work in distributed Kalman filters. With suitable network design there are advantages in reliability as well as bandwidth.

The next three sections describe the sensors we will be using in this control architecture.

4.3.1 Sonar array

Sonar has been used with success in several AGV projects, [11], [10], primarily as a low level sensor complementing vision. Sonar arrays have response times of ten to twenty milliseconds at the close range we expect in the obstacle avoidance layer. Sonar data is unreliable in certain conditions owing to specularity and poor angular resolution, but although these difficulties can be offset, or even exploited, by careful processing [14], the processing time then becomes a limitation on system bandwidth.

We use a sonar array in the low level reflex obstacle avoidance layer since it is a simple well developed sensor. The array incorporates its own processor for low level timing and overall control. It has a transputer as local intelligence at a higher level. The sonar array consists of 12 sonars, mounted at approximately 30cm intervals around the truck. Each sonar sensor is mounted on a stepper motor and can be rotated and fired individually.

4.3.2 The integrated infra-red/sonar sensor

To overcome some of the difficulties of sonar and to investigate possibilities in sensor integration, an integrated infra-red/sonar sensor has been developed to be available in any level. This sensor has an embedded transputer for dedicated control and processing. The infra-red operates through detection of returned amplitude. Infra-red and sonar are complementary in many respects and this sensor should give the advantages of both [9].

4.3.3 Infra-red range finder

Active infra-red sensors offer an alternative to vision and sonar. They may rely either on time-of-flight or on amplitude measurements. Simple time-of-flight sensors Cox [7] may be used for range measurement. The bandwidth is limited only by the electronics needed for signal transmission and detection. We are currently building such a sensor to use in level 1. The sensor will rotate to obtain a 360° view of the environment. Its speed in data acquisition is expected to be less than 1 second. The sensor has a dedicated transputer for control and processing.

5 Overall control system implementation

5.1 Architecture of the mobile robot

The mobile robot on which we are working has been lent to us by GEC Electrical Projects Ltd and is in the forefront of factory autonomous vehicles. We in fact have a small prototype of the commercial vehicle (Figure 2), which measures 0.9 x 1.2 m with a platform 0.6 m high.

The vehicle is fitted with vision and a sophisticated laser range finder for the high level path planning. For the work described here, it currently has fitted the sonar array. The other sensors should be fitted soon.

5.2 Low level control architecture

The vehicle was originally designed to operate through a radio link under the central control of a 'land-based' computer, which co-ordinates the passage of several vehicles in the factory. In the first phase of the project we are, as far as possible, leaving the original on-board facilities intact since they provide a powerful low level capability. We are therefore forced to use standard

Figure 2: The mobile robot test bed.

communication protocols. At a later stage we hope to realise a full transputer architecture.

In the original architecture, the land based controller provided all reasoning facilities. Each vehicle is given a predefined task, specified by end points and intermediate path points. The environment is assumed to be unchanging and there is no high level sensory feedback as the vehicle moves.

The low-level control architecture, which is built-in to the vehicle, provide a number of sophisticated facilities. It is based around a number of 8 bit and 16 bit microprocessors which are co-ordinated by a GEM 80 controller. The main sub-systems are as follows:

- A vehicle controller which controls steering and motion through a direct interface to the motors and shaft encoders
- A laser scanner system which runs a Kalman filter to provide absolute position information to the laser location system.

- A guidance system which performs low level path planning, using an approximate spline algorithm on a given set of route points
- An intelligent communications interface to communicate externally through a 20mA current loop at 9600 baud.

The GEM80 operates in a fixed 80msec cycle in our case. During each cycle it passes data between these subsystems and performs some internal system verification. The 80msec provides an absolute timing constraint at present.

5.3 Odometry, Localisation and Navigation

The vehicle holds an internal representation of its position at several levels of reasoning. In the GEC architecture, the land based controller updates an internal model of position as the vehicle moves. In addition the vehicle has hardware and software odometry and localisation to provide absolute positioning accuracy. Shaft encoders are used in the tight feedback loop to control the motors.

A laser scanner contributes to localisation. Upon start-up, the vehicle obtains an absolute reckoning of its position from the scanner using triangulation from positioned bar codes in the laboratory. Thereafter as it moves, results from both shaft encoders and laser scanner are input to a Kalman filter to update the vehicle's own representation of its position. If the variance is higher than a certain (preset) figure, the vehicle is defined as uncalibrated and a calibration sequence must be invoked before it can proceed. This technique ensures high positioning accuracy (to better than 2cm).

Communication with the vehicle from higher levels is through the intelligent communications interface. A message based protocol has been implemented for the definition of tasks and path points. Messages are packed or unpacked and verified by the communications interface. In the original implementation, the navigation guidance system takes the path points from the communicated message and uses a spline algorithm to provide a smooth path between them. Points are stored by the navigation guidance system into a position reference file, from which a new path point is sent out to the motor controllers each cycle time (80 msec) of the GEM80.

In the higher levels of our implementation something equivalent to the splining algorithm is required, possibly by externalising the navigation guidance system into the appropriate levels. However at the low levels, operating in a fast tight sensor based loop, we do not expect there to be time for path smoothing; instead each new point is calculated from immediate sensor data.

The GEM80 controller imposes limits on the speed and modes of operation of the vehicle. The cycle time limits the speed of communication with external devices, and hence the tightness of the control. It allows us to achieve our target bandwidth of 5-10Hz but may prevent us from exploiting the full capabilities of fast sensors for tight path control.

5.4 Implementation so far

At present one level is implemented in skeletal form: the reflex obstacle avoidance level. For demonstration purposes level 2 has been replaced by a task moving the vehicle to given end coordinates. With these, we are able to examine the interaction of control layers with the original control architecture and achieve simple obstacle avoidance. The importance of this implementation is to establish timing and interfacing principles between our control structure and the vehicle's original capabilities.

All the added capabilities are implemented using transputers. A transputer-20mA link allows communication between the higher levels of control and the on-board communications interface. The use of transputer technology throughout provides high bandwidth and simple communication protocols between different sub-systems in the controller and between the controller and sensors.

The highest level of control is currently simply a command to the vehicle of the form "go to (x, y, θ)" where (x, y) signify a point in the Cartesian co-ordinate system in which the bar code reflectors are placed and θ is the vehicle's heading. These are sent to the vehicle as a task, using the original message facilities. A further message is sent instructing the vehicle to begin moving.

The obstacle avoidance level is sensor driven through the

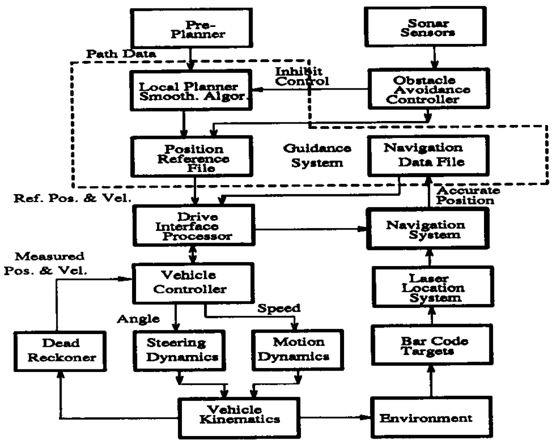

Figure 3: Block Diagram of Obstacle Avoidance Layer

sonar array, as shown in Figure 3. When an obstacle is detected within a certain range, it takes control of the vehicle from the higher level through sending a signal to the low-level guidance system via the communications interface and the GEM80 to inhibit execution of the pre-planned path. The obstacle avoid algorithm then creates new data and sends it to the position reference file as the vehicle negotiates around the obstacle under sensor control. Up to 30 set points at a time can be sent to the position reference file. The distance between successive set points must correspond to one 80msec cycle time of the GEM80; hence at a velocity of 0.5m/sec the set points are 40mm apart.

Figure 4 shows the implementation of the obstacle avoidance layer [12]. Data from the sonar array, controlled by an 8-bit processor, is processed by an on-board transputer. At present

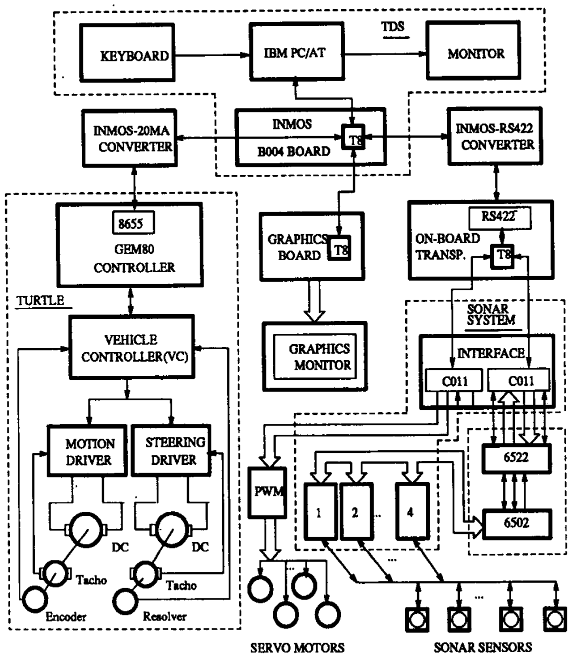

Figure 4: Implementation of Obstacle Avoid Layer using Transputers

this transputer is controlled by a development system on a PC, which also communicates with the GEM80 controller through the communications interface. The PC transputer will be replaced by another on-board transputer shortly. A graphics display is available through the PC.

For demonstration, a simple algorithm is used. Since the vehicle is designed to operate in the aisles of a factory frequently we expect to be able to make a local bypass around an obstacle, providing a rapid local diversion. A simple version of this algorithm has been implemented to validate our overall control philosophy and design, using the sonar array for sensory input.

Provided at least one of the sonar facing in the direction of travel of the vehicle (assumed here to be forwards) sees a clear path, the vehicle follows a piecewise linear path around the obstacle, as shown in figure 5. Initially the vehicle turns towards the free space detected by the sonar and moves to a point besides the obstacle. It then uses the side sonar to move along the obstacle, at a fixed distance. When the side sonar see that the obstacle is past, the vehicle returns to its predetermines path.

This is clearly a very simple scheme but has produced reliable results for obstacles such as chairs, boxes and people in a reasonably uncluttered environment. Its operation on the vehicle demonstrates stable switching between the layers of the control architecture and the feasibility of simple sonar processing in real time.

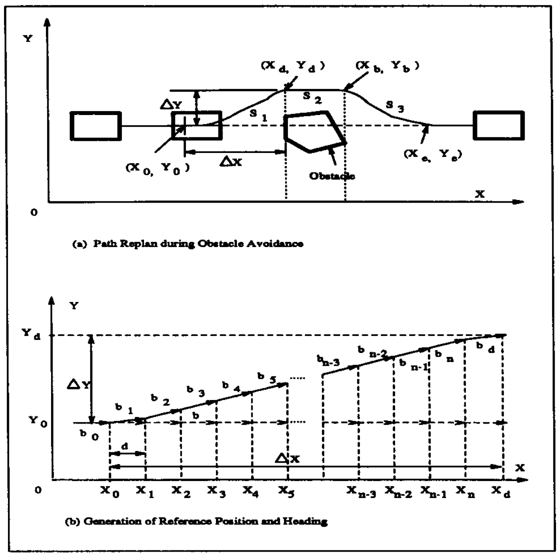

Figure 5: The principle of the obstacle avoidance algorithm.

6 A real-time potential field

The obstacle avoidance mechanism outlined above has been designed to give a safe lowest level of competence, offering a mobile robot protection from its environment, and more importantly the environment from the robot.

In this section we provide insight into our next level of competence within the layered control architecture. Instead of fulfilling the purpose of pure safety, level-1 is a strategic level, designed to successfully navigate a robot to a goal position. The level-1 controller requires no previously learned map of its environment, as it is designed to guide a vehicle using both past and present discrete depth readings, from continuously rotating infra-red range finders. As well as goal seeking, level-1 is designed to prevent a mobile positioning itself such that the obstacle avoidance controller, level-0, ever takes control. This design philosophy has been adopted in order to allow a 'graceful degradation' in the behaviour of a vehicle, in the event of a failure in level-1 performing correctly.

We attack the problem of navigation in our semi-structured environment, by applying an artificial potential field algorithm to sampled data provided by a continuously rotating sensor. Much previous work has focused upon the use of artificial potential fields in the navigation problem presented in mobile robotics research [5], [13], [16], but navigation using such an approach, whilst on the move, still appears to be a relatively unresearched issue.

6.1 The Potential Field Algorithm

At present our input sensor data takes the form of simulated infra-red time-of-flight range readings, with which we wish to model our environment, whilst 'on the fly'. These facts give us strict conditions to follow, when considering the possible rate of information extraction from the sensors, speed of algorithm execution, and the rate at which a vehicle will respond to updated velocity control signals. When dealing with a real vehicle, it would be ambitious to expect any response to changes in the desired velocity control signal, more than five times per second. On the other hand, because we are dealing with an infra-red light sensor, the speed of retrieving the depth information is restricted only to the minute electronic delays presented by the infra-red driving equipment [7]. These considerations have had a directing influence on our navigation strategy, which is explained in detail below.

With reference to figure 6 we consider the mobile to be the origin of a polar coordinate system, within which depth readings, provided by the continuously rotating sensor, are recorded.

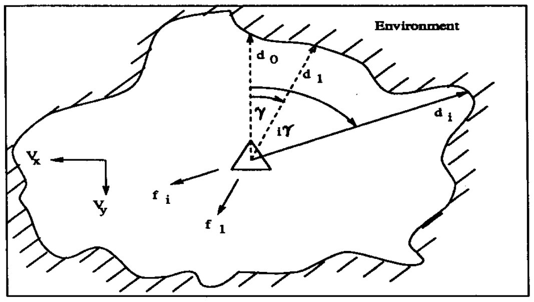

Figure 6: The mobile is the origin of a polar coordinate system, upon which force vectors are calculated directly from infra-red range readings.

The angle γ will be referred to as the *sampling angle*, and the radial distance d_i as the i th depth estimate, recorded directly from the sensor output. Corresponding to each range vector $\vec{d_i}$, a force vector $\vec{f_i}$ is immediately calculated obeying the simple artificial potential field equations:

$$|\vec{f_i}| = \frac{A}{|\vec{d_i}|^2} \tag{1}$$

$$\arg(\vec{f_i}) = \arg(\vec{d_i}) + \pi \tag{2}$$

where A is a fixed constant. At the end of a 360° sweep of the environment, the forces are *weighted*, *resolved* and *summed* in order to give an output which can be considered to be the desired new velocity components V_y and V_x, both parallel and perpendicular to the vehicle's centre line.

$$V_y = \sum_{i=0}^{2\pi/\gamma} \beta^{((2\pi/\gamma)-i)} f_i \cos(i\gamma) \tag{3}$$

$$V_x = \sum_{i=0}^{2\pi/\gamma} \beta^{((2\pi/\gamma)-i)} f_i \sin(i\gamma) \tag{4}$$

Note that the summation is completed when $i = (2\pi/\gamma)$ which must be an integral value when γ is measured in radians. The constant β is a weighting constant, to be explained later.

In order to assess the effect of the vehicle's velocity during the sensing process, we introduce the following mathematical model.

Consider a system which has a continuous input signal, dependent purely upon the shape of the environment ie: system input $= \frac{1}{|d_i|^2}$. We multiply this signal with a temporal cosine function, $A\cos(iT)$, which is non zero only at the discrete values of T when T equals γ, the sensor sampling angle. Then, pre-multiplying by the weighting term β and summing the above discrete signal values, we have:

$$\sum_{i=0}^{2\pi/\gamma} u(i) = \sum_{i=0}^{2\pi/\gamma} \beta^{((2\pi/\gamma)-i)} \frac{A}{|\vec{d}_i|^2}\cos(i\gamma) \tag{5}$$

which is identical to the y component of our output velocity. Hence by using a single rotating sensor, we can inter-relate temporal and spatial sampling within a non-recursive filter. Figure 7 shows a model for our discretised control system, for our y component of velocity only.

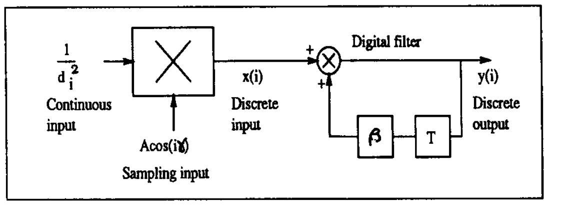

Figure 7: The sampling process used to extract forces from continuously rotating sensors.

The filter has an exact recursive equivalent, given by the difference equation:

$$y(i) = x(i) + \beta y(i-1)\ ,\ \ y(-1) = 0 \tag{6}$$

giving a z transfer function:

$$\frac{Y(z)}{X(z)} = G(z) = \frac{1}{1-\beta z^{-1}} \tag{7}$$

where z is the delay operator e^{sT}, or in our case $e^{s\gamma}$.

Our reason for choosing the weighting constant, β in the above equations, is to distinguish this method of navigation as a truly *real-time* process. A technique used in the past for eliminating the effect of vehicle motion during sensor data acquisition, was to take the vector sum of each depth reading and the vehicle's velocity [6]. This is used to predict what the corresponding depth reading would be, were the mobile stationary in its final position. In a real-time situation, to read in displacement measurements from the odometers, and calculate vector summations between successive depth estimates, would hinder the speed at which the infra-red sensor could rotate, and hence update the vehicle's velocity.

The constant β was introduced in order to take into account the translational motion of the sensor, due to the vehicle's velocity whilst scanning. If a scan were taken on a stationary vehicle, β would be set to unity. Physically speaking, β is a weighting factor used to make the velocity component V_y more dependent upon the force vectors placed upon the vehicle from in front, than those from behind. This is done in order to take into account the fact that the vehicle is moving with an old velocity V_y into the on-coming environment.

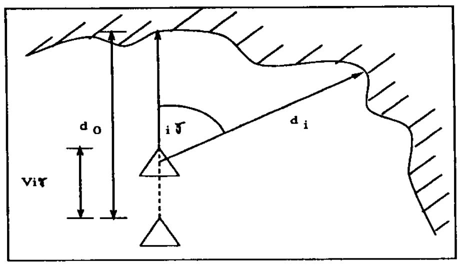

Figure 8: The effect of motion during sensor data acquisition.

Z-transform analysis of equation 7, for specific inputs $X(z)$, has produced interesting results, showing the dependence of the output signal $Y(n)$, the velocity component in the y direction, on the vehicle speed recorded during sensor data acquisition. Consider a general depth reading d_i, figure 8, and its dependence upon the environment and the vehicle's forward velocity V.

For any environment, each depth reading d_i is split into two terms of the form:

$$d_i = f(i\gamma) + Vg(i\gamma) \tag{8}$$

where $f(i\gamma)$ and $g(i\gamma)$ are functions of the sensor angle.

$$\frac{1}{d_i^2} = \frac{1}{f(i\gamma)^2}(1 + V\frac{g(i\gamma)}{f(i\gamma)})^{-2} \tag{9}$$

We can see that from figure 7 that our input $x(i)$ to the digital filter is:

$$x(i) = \frac{1}{d_i^2}A\cos(i\gamma) \tag{10}$$

so that, using 9

$$x(i) = m(i\gamma) + Vn(i\gamma) \tag{11}$$

where $m(i\gamma)$ and $n(i\gamma)$ are functions of the sensor angle.

Taking the unilateral z-transform of $x(i)$ and using equation 7, we can find the output signal. Upon taking the inverse transform and replacing $i = 2\pi/\gamma$ (since we are only interested in the last output value from the filter, for each scan) we get:

$$y(2\pi/\gamma) = p(\gamma,\beta) + Vq((\gamma,\beta) \tag{12}$$

Note that if the vehicle were stationary in its final position, then:

$$y(2\pi/\gamma)_{stat} = p(\gamma, 1) \tag{13}$$

since this is equivalent to the above case with $V = 0$ and $\beta = 1.0$. The error e in scanning whilst moving is given by:

$$e = p(\gamma,1) - p(\gamma,\beta) - Vq(\gamma,\beta) \tag{14}$$

which is minimised when:

$$\frac{de}{d\beta} = -P(\gamma,\beta) - VQ(\gamma,\beta) \tag{15}$$

Hence for minimum error, $\beta = T(\gamma, V)$, ie: β is a function of the vehicle's velocity and the environment into which the sensor points.

6.2 Experimental results

Figure 9 shows the results from a simulation of a mobile navigating in an office type environment.

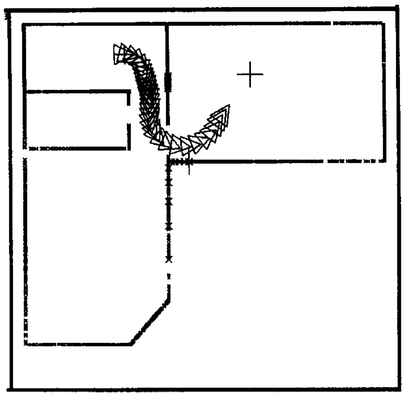

Figure 9: The mobile is moving under a potential field algorithm, and taking its scans whilst stationary.

In order to avoid the potential minimum (the point at which all of the forces placed upon the vehicle cancel), the goal is temporarily moved to the next depth reading beyond the edge of any gap recorded by the rotating scanner. The temporary new goal is shown as the lower large cross in figure 9. This method of *goal relocation* is explained in detail in [1]. The figure shows the path taken by a mobile, if it scans whilst *stationary* and then makes its move, and is the path that we would like our mobile to take if it moves during the sensing process.

Figure 10 shows the same start and goal positions for mobile scanning whilst moving, *without* weighting any of the depth readings ie: with β set permanently to 1.00.

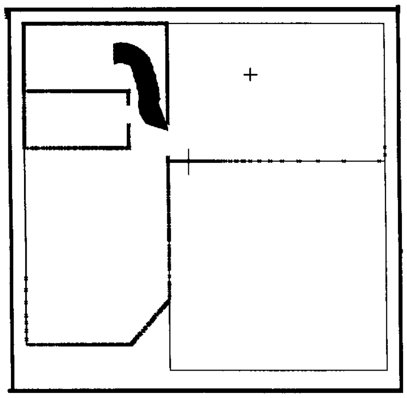

Figure 10: The distortion of scanning whilst moving is clearly shown at point P.

The distortion in the resulting velocity vectors is shown as the vehicle crashes at point P, where the environment dictates that accurate manoeuvres should occur.

6.2.1 Prediction of β

Equation 15 shows that for minimum error, β is a function of γ and d_i for a constant velocity. We are examining simple environments to obtain an approximation to this function. From simple heuristic arguments, as a first attempt we took the following function:

$$\beta = F(\sum_{i=0}^{2\pi/\gamma} d_i \cos(i\gamma)) \tag{16}$$

We have created a look up table for β, using this function as the basis of *gain scheduling* within the controller, [2]. This means that every time a new velocity is computed, a new value of β is also found, from the look up table, and used to influence the next manoeuvre. Figure 11 shows the same start and goal positions of figures 9 and 10. Although the function of 16 gives encouraging results, furhter work is needed to investigate its general applicability.

This time however, the value of β is continuously adjusted using values from the look up table, based on vehicle speed and the result of equation 16. The result is clearly shown, as the vehicle traverses a safe path towards its goal.

7 Conclusions

This paper has presented sensory and architectural aspects of a real-time obstacle avoidance capability and demonstrated its viability using a skeletal system. Transputers are used as the overall architecture to provide high bandwidth in communications, powerful processing capability and a unified, on-board architecture.

We believe that there are three vital areas of research to extend the capabilities of our system. One is associated with finding the appropriate sensors. Associated with this is the implementation of the obstacle avoidance strategy and its relationship with the sensory data it receives. Finally we intend to continue our investigation into the real time algorithms, an initial implementation of which has been demonstrated here using simulated data only. We have shown a possible method for greatly reducing the errors caused in environment modelling, whilst on the move, and in the near future we intend to extend our research using real

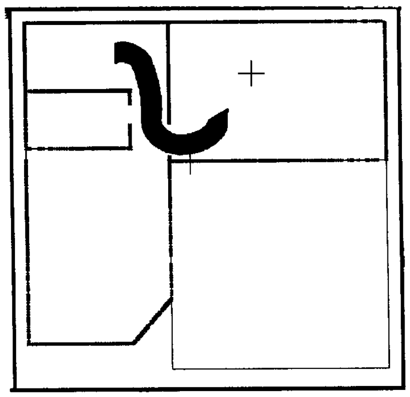

Figure 11: The effect of gain scheduling the value of β as each new velocity is computed.

time-of-flight infra-red range sensors.

8 Acknowledgements

We wish to thank GEC for the loan of the vehicle, and the many people within the Oxford A.G.V laboratory for contributing towards the project. Two of us (M.D.Adams and H. Hu) are supported by the SERC.

References

[1] Martin D. Adams. *A Layered Control System for a Mobile Robot.* Technical Report, Oxford University, Robotics Research Group, 1989.

[2] Karl J. Astrom and Bjorn Wittenmark. *Computer Controlled Systems.* Prentice-Hall, 1987.

[3] J.M. Brady, S. Cameron, H. Durrant-Whyte, M. Fleck, D. Forsyth, A. Noble, and I. Page. Progress towards a system that can acquire pallets and clean warehouses. In *Fourth Int. Symp. Robotics Research*, Santa Cruz, August 1987.

[4] J.M. Brady, H. Durrant-White, Huosheng Hu, J. J. Leonard, P.J. Probert, and B.S.Y. Rao. Sensor-based control of an agvs. In *Proc. IARP Conference on Domestic and healthcare robots*, Newcastle-upon-Tyne, U.K., September 1989.

[5] R.A Brooks. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation*, 2:14, 1986.

[6] R.A. Brooks. Self calibration of motion and stereo vision for mobile robots. In *Fourth Intl. Symp. Robotics Research*, 1987.

[7] Ingemar J. Cox. Blanche: an autonomous robot vehicle for structured environments. In *IEEE J. Robotics and Automation*, page 978 to 982, 1988.

[8] A. Elfes. A sonar-based mapping and navigation system. In *Proc. IEEE Int. Conf. Robotics and Automation*, page 1151, 1986.

[9] Anita M. Flynn and Rodney A. Brooks. M.i.t mobile robots - what's next? In *IEEE J. Robotics and Automation*, page 611, 1988.

[10] G. Giralt, R. Chatila, and M. Vaisset. An integrated navigation and motion control system for autonomous multisensory mobile robots. In *First Int. Symp. Robotics Research*, page 191, 1984.

[11] S.Y. Harmon. The ground surveillance robot (gsr): an autonomous vehicle designed to transit unknown terrain. *IEEE J. Robotics and Automation*, 1987.

[12] Huosheng Hu. *Distributed Architecture for Sensing and Control of A Mobile Robot.* Technical Report, Department of Engineering Science, University of Oxford, January 1990.

[13] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 500–505, St. Louis, March 1985.

[14] J. Leonard and H.F. Durrant-Whyte. *Navigation by Correlating Geometric Sensor Data.* Technical Report OUEL 1788/89, Oxford University, Robotics Research Group, 1989.

[15] B.S.Y. Rao and H. Durrant-Whyte. A fully decentralized algorithm for multi-sensor kalman filtering. In *IEEE J. Robotics and Automation*, 1989.

[16] Charles W. Warren. Global path planning using artificial potential fields. In *IEEE J. Robotics and Automation*, page 316 to 610, 1989.