The Development of Localisation Capabilities and Control for a Low-cost Robot

Kate McWilliams

18 February 2008

Prepared for:
Stephen Marais
Department of Mechanical Engineering
University of Cape Town

Declaration

This dissertation is submitted in complete fulfilment of an M.Sc (Eng)(Mechanical Engineering) at the University of Cape Town.

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. Each contribution to, and quotation in this dissertation from the work(s) of other people has been attributed, and has been cited and referenced.

I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off ask as his or her own work.

Signed by candidate

Kate McWilliams
MCWKAT001
February 18, 2008

Acknowledgements

I would like to thank the following people for their help and support during the course of this project.

My supevisor, Stephen Marais, for his constant support and suggestions without which I could not have completed this project.

Marc Brooker for his incredible support, assistance with software and giving up weekends to help with testing.

My research group members, Sally Levesque, Ian Baldwin and Tracy Booysen for all the advice, assistance and entertaining tea-breaks.

Graham Brooker, for organising my time at the Australian Centre for Field Robotics which played such a key role in this work.

My parents for all their support over my many years of study.

This project would not have been possible without the financial assistance provided by the following:

The National Research Foundation (NRF)
The Harry Crossley Foundation
The Department of Mechanical Engineering
The University of Cape Town

Opinion expressed and conclusions arrived at in this dissertation are those of the author and are not necessarily to be attributed to the NRF.

Abstract

A fully autonomous robot which can perform dangerous or mundane tasks is the ideal outcome of robotic research. A variety of commercially available household robots such as robotic vacuum cleaners exist but are limited in their navigation ability. In general, they tend to use random search patterns to navigate a room and overestimate the time required to clean the room in order to ensure covering the entire area. The ability to map the environment and then use this map to navigate is an essential step towards total autonomy, and would greatly improve the efficiency of these household robots.

Autonomous mapping is a complex problem as the robot must use sensor readings to generate a map while at the same time using that map to locate itself and navigate. One component of the mapping task is localisation. This is the process of determining position and orientation from sensor data given a known map. This was the focus of this work as a first-step towards an autonomous mapping robot.

This project continued the work of an undergraduate thesis in which a robot vacuum base was built. Using this base, the sensing and control systems were developed. The selection of a suitable controller was an important aspect of the development. It had to be suitable not only for this task but allow for expansion of the control capabilities should the project be extended. The Gumstix/Roboaudiostix embedded system was chosen and performed successfully. Its extremely small size and low power requirements are a feature of the system.

Another requirement was the control of the robot and sensor systems to provide data for the localisation algorithm. This included speed control and odometry using optical encoders and an ultrasonic object sensing system. These systems were successfully implemented and provided reliable data which was sufficient for localisation under certain conditions. A method

for reducing interference between ultrasonic transducers operating simultaneously was also investigated. This would be an advantage if more sensors were added to the robot or if multiple robots were operating in the same area.

The localisation software was a key component of the development. The extended Kalman filter localisation algorithm was implemented offboard in MATLAB for ease of testing and debugging but was intended for eventual implementation on the Gumstix platform.

As a first test for the localisation algorithm, a full simulation of the robot was developed. It simulated robot motion and sensor data and provided realistic data for the localisation algorithm. It also provided a way to determine sensor and vehicle accuracy requirements for accurate localisation. The algorithm performed well with simulated data and was therefore tested with experimental data.

To test the localisation algorithm, the robot had to drive along a path taking odometry and ultrasound readings with beacons in known positions, providing a map of the environment. The sensor data was stored and fed to the algorithm once the test was complete to estimate the robot's path. To verify the results, the robot's actual path was also recorded. This was extracted from video footage of the tests. A comparison of the localisation output, the odometry data and the actual path was then made to measure the performance of the algorithm.

It was successful in cases where correspondence was known, with the estimated path closely matching the actual path. Correspondence is the process of matching a sensor reading to a beacon. If an observation was manually attributed to the correct beacon, the performance was excellent. If however, the correspondence was unknown and the algorithm had to estimate which beacon was observed, the result was unsatisfactory. This was due to large odometry errors during turning, causing a large orientation error which resulted in incorrect data association between observations and the stored map.The correspondence problem is well-known and much work has been done in this field. Various recommendations such as adding additional sensors were made for improving performance.

This work demonstrated that localisation of a robot in a known environment using low-cost sensors is feasible. The extended Kalman filter was an effective method for the task if correspondence between observations and the

known map could be determined. This has laid the foundation for future development of a mapping capability on this system.

Recommended future work includes the implementation of the extended Kalman filter localisation algorithm on the Gumstix embedded system followed by simulation and implementation of a simultaneous localisation and mapping algorithm.

Contents

List of Figures

List of Tables

Glossary of Terms

ACFR	Australian Centre for Field Robotics
ADC	Analogue to digital Converter (Conversion)
Correspondence	Associating measurements with beacons
DAC	Digital to analogue Converter (Conversion)
EKF	Extended Kalman Filter
Kalman Filter	Recursive filter used for localisation
PI	Proportional integral control
PID	Proportional integral derivative control
Pose	Position and orientation
PWM	Pulse Width Modulation
SLAM	Simultaneous Localisation and Mapping

List of Symbols

$X(k)$	System state at time k
$u(k)$	Control input
$v(k)$	Gaussian process noise
$F(k)$	State transition matrix
$B(k)$	Control transition matrix
$G(k)$	Noise transition matrix
$z(k)$	Observation
$H(k)$	Observation model
$w(k)$	Observation noise
$\hat{X}(k)$	System state estimate
$P(k)$	Error covariance matrix
$Q(k)$	Process noise covariance
$\bar{y}$	Innovation
$S(k)$	Innovation covariance
$R(k)$	Measurement noise covariance
$K(k)$	Kalman gain
∇f_X	Jacobian of state transition matrix
∇h_X	Jacobian of observation model
ϵ_{sim}	Simulation error
ϵ_{ekf}	Estimate error used by EKF
θ	Orientation Angle
ϕ	Bearing to observation
k	Controller gain

Chapter 1

Introduction

The idea of a robotic servant, taking care of mundane household tasks has long been present in science fiction. Although not quite at the stage of a human-like, fully autonomous robot, capable of carrying out a variety of tasks, household robots are no longer a technology of the future. With sophisticated controllers and a variety of sensors, robots can carry out specific tasks without human intervention.

Current commercially available robot vacuum cleaners, while having many desirable features are limited by their navigation ability. One of the most widespread products is the Roomba from IRobot [4].

Figure 1.1: IRobot's Roomba Scheduler

Figure 1.1 above shows the Roomba. It is an autonomous vacuum cleaner which can detect and avoid any obstacles in a room and cover the entire floor

area, including many hard to reach places under furniture due to its compact size. It can also clean near to walls, detect and pay extra attention to dirtier areas and avoid stairs or sharp drop-offs.

Although this product has all the features necessary for the task, the navigation system is not efficient. It uses a random search to cover the floor area and in order to clean the entire floor, a point is covered an average of four times [4, 5].

The ability to map the room would be an advantage as the cleaning path would be much more efficient. For a robot, building a map of the environment is not a trivial task and much research has been undertaken in this field. An overview of environment mapping is provided in appendix A.3.2 with further detail in section 2.4 on page 9. Even localisation, using a given map and available sensor information to determine position, is a fairly complex procedure although this problem is well-understood. See appendix A.3.1 for further information. This project focused on this localisation task as a first-step towards the eventual goal of an autonomous robot which can both map the room and use this map to navigate.

The development of this system was begun at the University of Cape Town's Mechanical Engineering Department as an undergraduate thesis by Philip Young [6]. In his project, a robotic vacuum base was designed but the control aspects were not completed. This project continued the development, using the existing design for the mechanical components. These included the chassis, vacuuming and drive systems and the power supply. The original project did include collision sensors but these were removed and a new sensing and control system developed.

The objectives of this work included selecting and implementing a suitable controller for the robot. This had to be sophisticated enough to not only perform the currently required tasks but also had to be extendable for future mapping and navigation tasks. A Gumstix embedded computer [3] and microcontroller combination provided this platform and although the initial implementation was complex, the powerful capabilities of this system provide a solid foundation for future upgrades and continuation of this work. The selection process is outlined in appendix B.

The sensor system also had to be developed. This system consisted of ultrasound transducers and optical encoders measuring wheel-speed. Although not implemented on the final system, a method of avoiding interference be-

tween ultrasound transducers was investigated. This is outlined in appendix G.

To provide a working system to test the localisation software, much circuitry and embedded software had to be developed. Speed control had to be implemented to provide odometry data to the localisation algorithm and the support circuitry and software for the ultrasound transducers had to be designed, implemented and calibrated. As many of the robot processes had to take place in real-time, timing was critical in the design of the microcontroller software.

Another component of the work was simulation of the system in order to test the localisation algorithm and determine sensor and vehicle accuracy requirements. This included developing a realistic model of the robot motion and sensor data in which parameters could be varied and the effect on the system determined. Although intended to eventually run on the robot's onboard controller, the localisation algorithm itself was implemented on an offboard PC in MATLAB to test and calibrate the software in a user-friendly environment. This algorithm was initially tested with the simulation data, and later with actual robot test data which was stored onboard and then processed in MATLAB.

The final objective was to process the test results, and so determine any hardware requirements required for an effective localisation system.

This report begins with background theory on the localisation and mapping process followed by a discussion of work done at the Australian Centre for Field Robotics (ACFR). This work includes the implementation of an acoustic tracker on an autonomous vehicle which utilised the Gumstix system and acted as an ideal base for testing the controller. Courses in mapping and navigation taken at the ACFR also lead to the utilisation of the localisation algorithm implemented in this work. Further sections describe the hardware used in the system, an overview of the software including both the embedded and simulation software, followed by the simulation itself and testing. Results are then discussed, conclusions drawn and recommendations made. A set of appendices provides greater detail on the various aspects of the project.

Chapter 2

Background Theory

2.1 Simultaneous Localisation and Mapping

The goal of this project was to provide the localisation capability of the robot as a foundation for building a mapping capability in the future.

Localisation and mapping were traditionally viewed as separate problems. Localisation is determining the robot pose (position and orientation) given perfect knowledge of a map of the surroundings and sensor information, and mapping is building up a map given robot pose. It is however almost impossible to know the robot pose accurately enough from odometry data and vehicle models so errors tend to grow without bound [2][P.200 Chapter 14]. This makes mapping a difficult problem. Another added complication was that localisation and mapping would have to be done at the same time if truly autonomous robots were ever to explore unknown environments. Fortunately, it was found that the localisation and mapping problems didn't have to be treated separately. In fact, there is a convergent solution when the processes are combined and so they must not be performed independently. This was a major breakthrough in autonomous robotics research and has lead to many improvements in navigation and mapping ability. The outcome was a procedure known as SLAM (simultaneous localisation and mapping) [7][8][9]. SLAM is a very powerful technique although there are still many challenges to be faced such as the data association problem outlined in section 2.5.

At the heart of localisation and SLAM is the Kalman filter and a basic knowledge of this and other probabilistic techniques is required to understand

the process [10].

2.2 Kalman Filter

The Kalman filter (introduced by R.E. Kalman in 1960 [11]) is a recursive filter which is used to estimate the state of a system from a process model and measurement data. It is used in many applications including radar tracking and satellite navigation (a notable example is the Voyager spacecraft orbit determination at Jupiter [12]) and is widely used in robotic navigation and mapping. Because the filter is recursive, it requires only the information from the previous time-step and does not require all past states to be stored. The total information from all these states is incorporated in the current value for the state.

The following sections describe the details of the linear and extended Kalman filters. This is a summary of information from a number of sources. Further details can be found in [13][14][10][8].

2.2.1 Linear Kalman Filter

The linear Kalman filter requires a system which can be described by a discrete-time, linear equation of the form

$$X(k) = F(k)X(k-1) + B(k)u(k) + G(k)v(k) \tag{2.1}$$

where $X(k)$ is the system state at time k, $X(k-1)$ is the state at the previous timestep, $u(k)$ is the control input and $v(k)$ is the Gaussian process noise. $F(k)$, $B(k)$ and $G(k)$ are the state, control and noise transition matrices respectively.

The observation equation is also a linear equation of the form

$$z(k) = H(k)X(k) + w(k) \tag{2.2}$$

where $X(k)$ is the state at time k, $H(k)$ is the observation model and $w(k)$ is the Gaussian observation noise.

The state of the filter at time k consists of the system state estimate $\hat{X}(k)$ and the error covariance matrix $P(k)$. $P(k)$ is a measure of the accuracy of the estimate.

The Kalman filter is comprised of two phases, the prediction and the update. In the prediction phase, the current system state is predicted using the process model, $F(k)$ and control inputs. The input for this prediction is the estimate of the state at the previous time-step. The covariance, $P(k)$ is also predicted. During the update stage, measurement information is used to improve these predictions, forming the new state and covariance estimates.

Prediction Stage

The state and covariance predictions are calculated from the state and covariance estimates at the previous timestep.

$$\hat{X}(k|(k-1) = F(k)\hat{X}(k-1|k-1) + B(k)u(k) \tag{2.3}$$

$$P(k|k-1) = F(k)P(k-1|k-1)F(k)^T + Q(k) \tag{2.4}$$

where $Q(k)$ is the covariance of the process noise distribution.

Update Stage

The innovation is the difference between the actual measurement and the expected measurement based on the predicted state. It is calculated from

$$\bar{y} = z(k) - H(k)\hat{X}(k|k-1) \tag{2.5}$$

The innovation covariance is calculated from

$$S(k) = H(k)P(k|k-1)H(k)^T + R(k) \tag{2.6}$$

where $R(k)$ is the covariance of the measurement noise distribution.

The Kalman gain is calculated from

$$K(k) = P(k|k-1)H(k)^T S(k)^{-1} \tag{2.7}$$

The state estimate and the covariance are then updated

$$\hat{X}(k|k) = \hat{X}(k|k-1) + K(k)\bar{y}(k) \tag{2.8}$$

$$P(k|k) = (I - K(k)H(k))P(k|k-1) \tag{2.9}$$

2.2.2 Non-Linear Filter (Extended Kalman Filter)

In many cases, the process and observation models for the state to be estimated are not linear and so the Kalman filter shown above can not be applied. For non-linear problems, the extended Kalman filter is used. This has the same form as the linear filter except that the process and observation models are linearised around the current state. The extended Kalman filter solves the problem of non-linear processes but is particularly sensitive to initial conditions and errors in the process and observation models. Computational cost is also increased considerably.

In the extended Kalman filter, the process model is a non-linear function of the state

$$X(k) = f(X(k-1), u(k), k) + v(k) \tag{2.10}$$

where $X(k)$ is the system state at time k, $X(k-1)$ is the state at the previous timestep, $u(k)$ is the control input and $v(k)$ is the Gaussian process noise. $f(k)$, is a non-linear function which transforms the previous state and control input to the current state.

The observation model is also non-linear

$$z(k) = h(X(k)) + w(k) \tag{2.11}$$

where h is a function which maps the state to the observations and w is measurement noise.

The functions, f and h can be used to calculate the predicted state and predicted measurement as in the linear filter. They however, cannot be directly applied in the covariance calculations. Instead, the Jacobians (∇f_X,∇h_X) of f and h are used. This effectively linearises the functions around the current estimate

The same procedure as the linear filter is followed with the Jacobians replacing the transition functions in the covariance calculations.

Prediction Stage

$$\hat{X}(k|k-1) = f(\hat{X}(k-1|k-1), u(k)) \tag{2.12}$$

$$P(k|k-1) = \bigtriangledown f_X(k) P(k-1|k-1) \bigtriangledown f_X^T(k) + Q(k) \tag{2.13}$$

Update Stage

$$\bar{y} = z(k) - \bigtriangledown h_X(k) \hat{X}(k|k-1) \tag{2.14}$$

$$S(k) = \bigtriangledown h_X(k) P(k|k-1) \bigtriangledown h_X^T(k) + R(k) \tag{2.15}$$

$$K(k) = P(k|k-1) \bigtriangledown h_X^T(k) S(k)^{-1} \tag{2.16}$$

$$\hat{X}(k|k) = \hat{X}(k|k-1) + K(k)\bar{y}(k) \tag{2.17}$$

$$P(k|k) = (I - K(k) \bigtriangledown h_X(k)) P(k|k-1) \tag{2.18}$$

2.3 Kalman Filter Localisation

A localisation simulation was conducted to demonstrate the effectiveness of the Kalman filter and to test the system's response to various parameters. To use the Kalman filter for robot localisation given a known map, the system state is a vector which describes the robot pose (position and orientation). There is a known map of beacons which is a vector of their x and y coordinates. Observations are made of these beacons and they will depend on the sensor used, commonly range and bearing.

2.3.1 Process Model

A process model of the robot motion is required. For this robot, the process model for the robot from state $X(k-1)$ to $X(k)$ given information up to $k-1$ and under control inputs, V_L and V_R is

$$\begin{bmatrix} \hat{x}(k|k-1) \\ \hat{y}(k|k-1) \\ \hat{\theta}(k|k-1) \end{bmatrix} = \begin{bmatrix} \hat{x}(k-1|k-1) - \frac{1}{2}\Delta T(V_L(k)+V_R(k))\sin\theta(k-1) \\ \hat{y}(k-1|k-1) + \frac{1}{2}\Delta T(V_L(k)+V_R(k))\cos\theta(k-1) \\ \hat{\theta}(k-1|k-1) + \frac{V_R-V_L}{B}\Delta T \end{bmatrix} \tag{2.19}$$

The derivation of this model is detailed in appendix F on page F-1. The vehicle model is illustrated in figure 2.1 and the symbols used can be found in table 2.1.

X	State Vector
x	x Position of Centroid
y	y Position of Centroid
θ	Orientation Angle
V_L	Left Wheel Rotational Velocity
V_R	Right Wheel Rotation Velocity
B	Distance between Left and Right Wheels
ΔT	Change in Time

Table 2.1: List of Symbols

Assumptions

- Zero wheel slip
- For small angles $\arctan(x) = x$
- Independent movement of left and right wheels over small timesteps

2.4 SLAM Solution

SLAM can be thought of as an extension of Kalman filter localisation. In this case however, the map is not known beforehand. Instead the map is included in the state vector and therefore is estimated in the same way as the

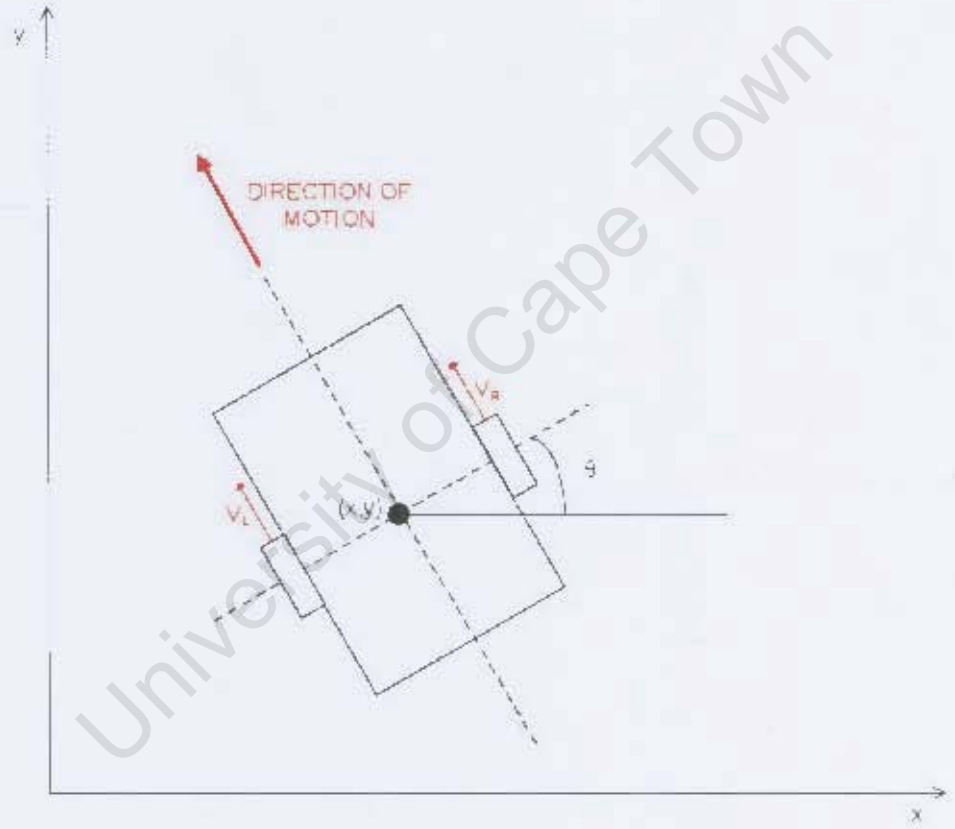


Figure 2.1: Vehicle Model

robot pose. The state vector therefore grows as landmarks are detected and included in the map. As these landmarks are re-observed, the uncertainty of their positions decreases and in the limit, the map converges to a relative representation of the actual environment, with some constant error determined by the initialisation error. The entire map can therefore be perfectly known as soon as the position of one of the landmarks is perfectly known.

The robot performs localisation exactly as in the Kalman filter method, but instead of having a stored map of beacons, it uses the changing map of landmarks. As time progresses, the estimate for the map becomes more certain and the localisation estimates also improve.

2.5 Correspondence

One of the most difficult components of localisation and SLAM is correspondence (also known as data association). This is the process of determining whether an observation corresponds to a stored landmark or whether it is a new landmark. This is a complex problem especially when the robot "closes the loop" (returns to a previously visited location). It is also important for the algorithms to ignore spurious readings and an incorrect association or the incorporation of a false landmark in SLAM can have a catastrophic effect on performance [9][14].

There are a number of methods used to address this problem and much research has been conducted in this area [15][16][8]. A standard method and the one utilised in this work is the maximum likelihood correspondence, which determines which is the most likely association and assumes this value. This method however can break down if there are many hypotheses which may be equally likely. A solution to this is to select beacons that are distinctive and sufficiently far apart from each other to avoid confusion. This however means that pose errors can grow quickly in the large areas between beacons where no sensor observations are incorporated into the estimate [10][P.215, Chapter 7][17]. Another solution is using multiple sensors to measure different properties of the beacons making it easier to distinguish them from each other [16] or to track multiple hypotheses but this requires considerable computational power [17].

Chapter 3

Tracker Implementation

3.1 Introduction

This chapter discusses work done during the two and a half months spent as an occupational trainee at the Australian Centre for Field Robotics (ACFR) at the University of Sydney in 2007. This consisted of the implementation of a combined infrared and acoustic beacon tracker on an autonomous vehicle. Course work in mapping and navigation undertaken at the ACFR also lead to the implementation of the Kalman filter localisation algorithm in the vacuum cleaning robot project.

3.2 Project Overview

The aim of this work was to develop a low-cost vehicle which could track and follow a person carrying a beacon at a given distance. A combined infrared and acoustic tracker was designed for this task at the Australian Centre for Field Robotics [1]. My contribution to this project was the implementation on an autonomous vehicle. The following sections summarise the operation of the tracker and the implementation and testing of the vehicle system. A detailed account of the work can be found in the paper "Combined Infrared and Acoustic Beacon Tracker Implementation on an Autonomous Vehicle" which was presented at the 2nd International Conference on Sensing Technology in 2007. A copy of the paper is included in appendix H.

3.3 Tracker Operation

The tracker consists of a sensor mounted on the vehicle and a beacon carried by the operator. The beacon transmits a pulse of infrared light and then a burst of ultrasound. The receiver detects these signals and the range to the beacon is determined by the difference in arrival time of the infrared and ultrasonic signals. It is assumed that the light arrives instantaneously so the time of flight is measured from this instant until the ultrasound is detected. This is illustrated in figure 3.1.

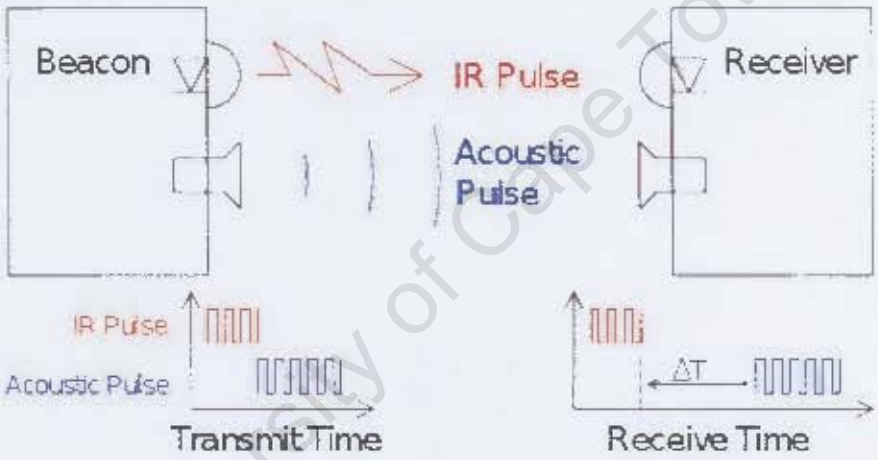


Figure 3.1: Diagram Showing Range Measurement Technique [1]

Two methods are used to determine the angle of the beacon. The first uses the infrared signal. There are two infrared receivers on the sensor which can detect the pulse. They are separated by an opaque barrier so if both are illuminated, the beacon is straight ahead while if only one is illuminated, the beacon lies in that direction. If neither is illuminated, the vehicle would have to rotate until a signal is found. This method has clear limitations in that the size of the angle error is not determined and therefore, only bang-bang control can be implemented.

The second method uses the ultrasonic receivers and can provide a more accurate measure of the angle. The receivers are angled slightly away from each other. This means that if the beacon is at an angle, the amplitude of the received signal at one transducer will be greater than at the other. Only

if the beacon is directly in front of the receiver, will the signals be equal. The difference between the two signals is therefore used to calculate the angle of the beacon. Figure 3.2 shows this error signal as the angle of the beacon varies. There is a fairly linear region between $\pm 20^{\circ}$ which can be used for proportional control.

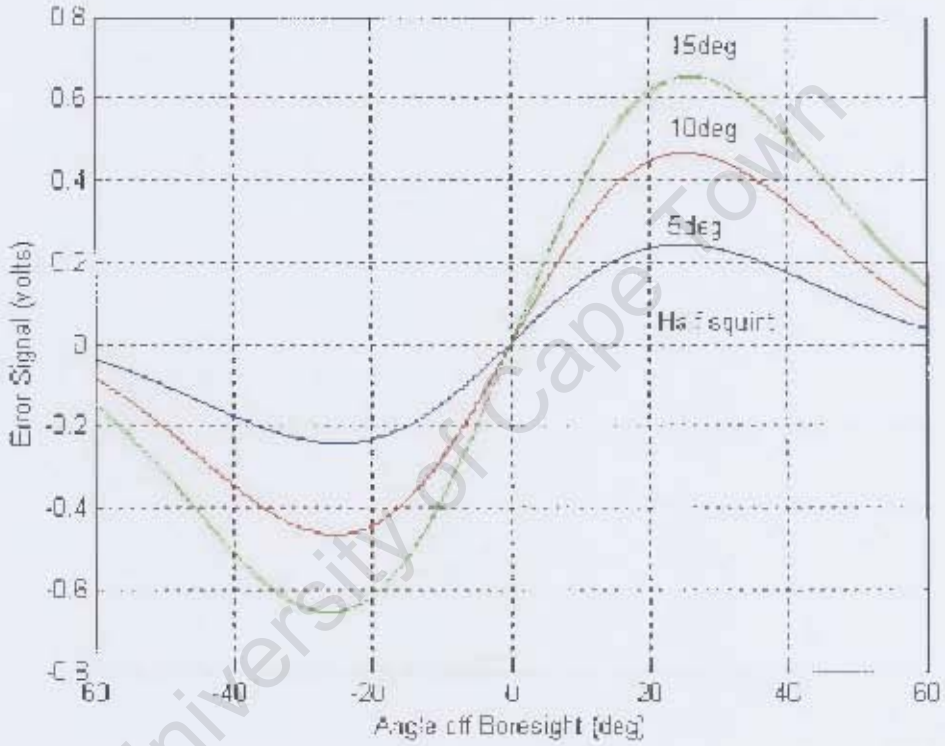


Figure 3.2: Angle Error Transfer Function [1]

The sensor implementation is discussed in [1]. The following signals are provided by the sensor to the controller. There are two infrared signals, one from each receiver and also a signal from each of the ultrasonic transducers. There is a clock signal when valid data is present. This data is valid if at least one infrared signal and the ultrasonic signal is present. There is also an analogue voltage which is proportional to the range to the beacon.

3.4 Controller

The Gumstix/Roboaudiostix system was used in this work which served as an ideal test platform for this controller. Although only the Roboaudiostix was used on the vehicle, the Gumstix was used to program it and therefore the entire system had to be correctly set up and was tested in this application.

3.4.1 Gumstix/Roboaudiostix System

The selection of a suitable control system for the vacuum cleaning robot was a large part of this project. This same system was used to control the tracker vehicle to test its operation. The combination of a Gumstix embedded computer and Roboaudiostix microcontroller was chosen. The Roboaudiostix runs the low-level software while the Gumstix is used to program it and can perform high-level tasks. This combination allows for the the system to be extended and upgraded in the future while still allowing low-level behaviour to be programmed quickly and reliably. The decision is outlined in appendix B.

3.5 Vehicle

The tracker was mounted onto a radio-controlled vehicle. This can be seen in figure 3.3 below. The Roboaudiostix board is also shown as well as an opto-isolation board. This configuration was used to control the vehicle in a similar setup to that of the vacuum cleaning robot shown in figure 4.1 on page 20.

3.6 Control Algorithm

3.6.1 Valid Data Determination

The first task in controlling the vehicle was to determine whether the signals received from the sensor were valid. The valid data clock had to be read and interpretted by the microcontroller. This was done by feeding the signal to an external interrupt pin. An 8-bit timer was used in conjunction with the interrupt to determine whether the signal was present. If it was absent, the sensor data was ignored and appropriate behaviour initiated.

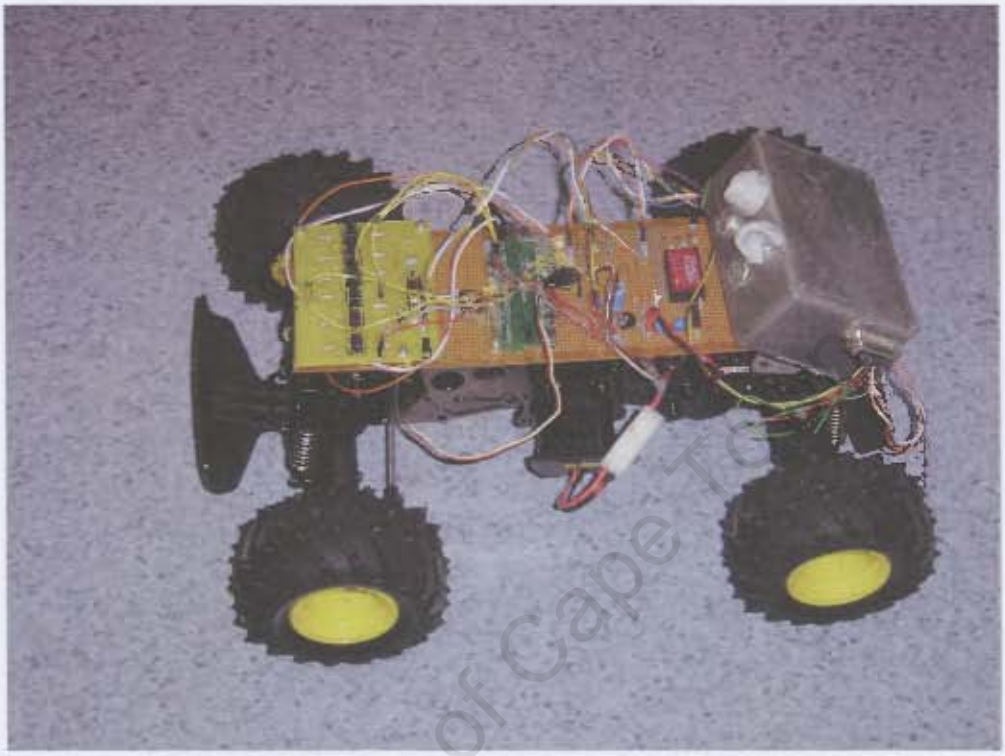


Figure 3.3: Tracker on Robotic Vehicle with Roboaudiostix and Opto-isolation Board [1]

3.6.2 Tracking Algorithm

The tracking algorithm consists of two parts, the range control and steering angle control. Both parameters are controlled using proportional control. The range voltage reading is digitised by the analogue to digital converter and used in the following equation to calculate the wheel speed.

$$R = K_1(V_{range} - V_{set}) \tag{3.1}$$

where R is the speed of the wheels, K_1 is a constant, V_{range} is the range voltage and V_{set} is the range set-point.

The objective of the steering tracking algorithm was to keep the beacon in the centre between the two ultrasound transducers. When it is in the centre, the two signals are equal. The offset error was estimated and normalised using the left and right receiver voltages according to the equation.

$$\triangle Az = K_2 \frac{V_{left} - V_{right}}{V_{left} + V_{right}} \tag{3.2}$$

where $\triangle Az$ is the normalised angle error, K_2 is a constant, V_{left} is the left receiver signal and V_{right} is the right receiver signal. The normalisation is required to provide accurate steering control independent of the amplitude of the received signals.

The amplitude of the received signals is also considered to avoid amplification of noise which would cause errors at low signal levels. Therefore, if the received signal is below a threshold, this data is ignored. A small dead-band was introduced to prevent oscillation about the centre-point.

3.7 Results

3.7.1 Noise and Multipath

Initial testing of the system showed noise present on the ultrasound signals. This was eliminated using a digital low-pass filter. Averaging the signals resulted in smooth steering behaviour. A number of possible sources of the noise were identified. These included interference from the servo motors, digital noise from the sensor circuitry and environmental noise.

During testing, the vehicle would occasionally track a false target. This was found to be due to multipath errors caused by the cluttered environment. The fact that the vehicle stopped tracking the false target as soon as the beacon was switched off proved that it was in fact tracking a reflection.

3.7.2 Control Algorithm

The proportional control used for both the steering and range parameters was sufficient for this system. A more complex control algorithm was unnecessary due to the considerable damping added by the mechanical components of the system.

There was however some overshoot at maximum speed if the beacon came to a sudden stop. If high vehicle speed and sudden changes in speed were required, a PI or PID controller could be implemented to solve this problem. This

would however come at a higher computational cost and was not necessary for this particular application.

3.8 Concluding Remarks

The implementation of the tracker on the vehicle was successful. The vehicle was capable of following a fast moving person which demonstrated the accuracy of the sensor. This work also served as a testbed for the Gumstix/Roboaudiostix control system which proved to be a suitable controller for the robot vacuum cleaner system. The microcontroller was able to drive servo motors and perform control actions in real-time which is similar to the tasks it would need to perform on the robot vacuum.

Microcontroller code for motor control, external interrupt and timer operation was written which was later used in the robot vacuum cleaner system. Testing this software on the tracker system was an excellent introduction to the hardware and a good way to test the controller in isolation from the rest of the system. The software routines developed in this work considerably sped up the development of the robot vacuum software.

The information obtained through coursework in mapping and navigation led to the implementation of the EKF localisation algorithm in the vacuum cleaning system as a first step towards eventual autonomous mapping and navigation. An overview of this system and the results obtained are discussed in the following chapters.

Chapter 4

System Overview

This chapter describes the robot vacuum cleaner system including hardware and software components and their interactions. Since the focus of this project was the implementation of a localisation algorithm and the design of the sensing and control system of the robot, the mechanical aspects of the existing platform were not greatly modified in this work.

Figure 4.1 on the next page shows a breakdown of the entire system including both onboard and offboard components.

Figure 4.2 shows the robot with the placement of the various components indicated. These components are described in the following sections

4.1 Hardware

4.1.1 Robot Platform

The original robot base [6] consisted of a perspex chassis, two rear driven wheels, a front caster and a brush and fan cleaning mechanism. The drive system consisted of a motor directly coupled to each wheel driven by a LMD18200 H-bridge chip [18]. The original sensors and control circuits were removed and replaced.

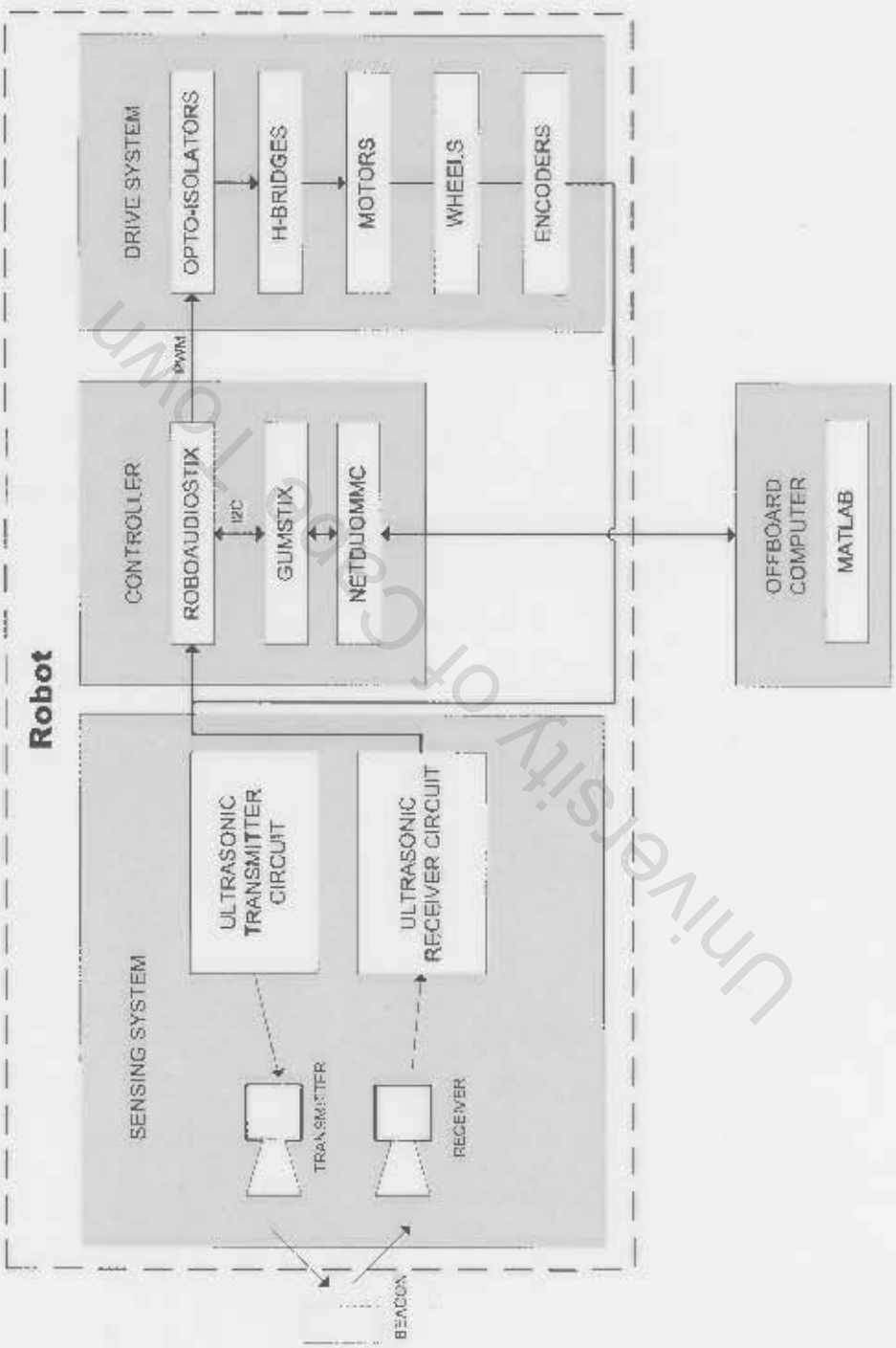


Figure 4.1: System Overview Block Diagram

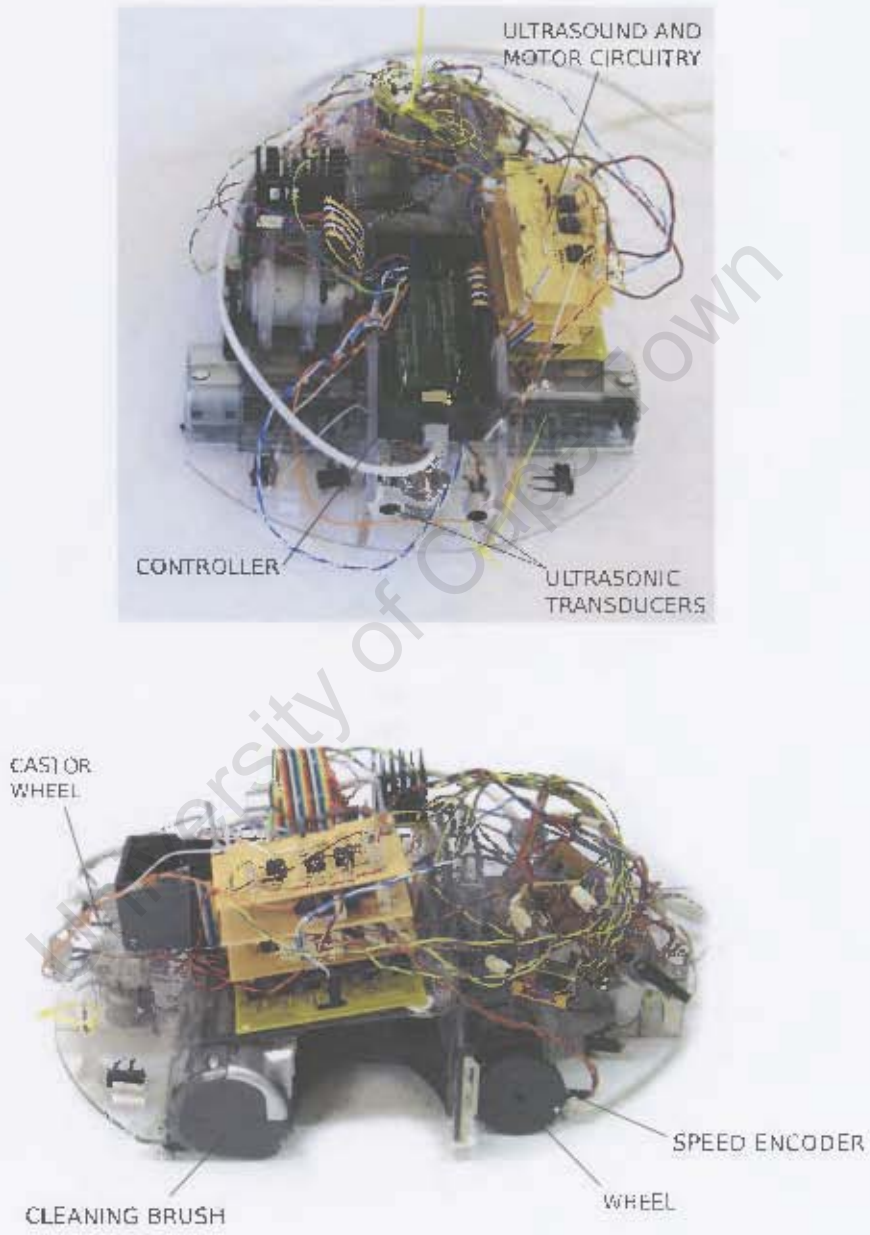


Figure 4.2: Vacuum Cleaning Robot

4.1.2 Controller

The controller consisted of a Gumstix, a Roboaudiostix and a NetMMC board contained in a protective housing. The Gumstix was used for high-level tasks, the Roboaudiostix for low-level tasks while the NetMMC was used for communication with the offboard computer. This was necessary for the prototype but would be removed in the final system.

Figure 4.3 shows the controller. Its extremely small size is a notable feature considering the available computational power.

Figure 4.3: Controller Mounted in Housing

4.1.3 Sensors and Circuitry

The object sensors were Murata 40kHz ultrasonic transducers [19] mounted on the front of the robot. Separate transducers were used for transmitting and receiving. The transmitter circuit, described in appendix E consisted of a 40kHz oscillator controlled by the microcontroller. The receiver circuitry included an amplifier, rectifier and comparator. Details of its operation are described in appendix E. The output of this circuit was fed to the microcontroller which determined the time-of-flight of the received signal.

The H-bridge chips were driven by the microcontroller through an opto-isolation board. Wheel speeds were measured by H21A1 phototransistor optical interrupter switches [20] in conjunction with slotted disk encoders attached to each wheel.

4.1.4 Power

The power supply was fairly complex due to the different voltage requirements of the controller and support circuitry and the need to isolate the sensitive circuitry from the motors. Figure 4.4 shows a schematic of the power distribution system. For convenience, in this prototype, a tether was used to supply power to the robot.

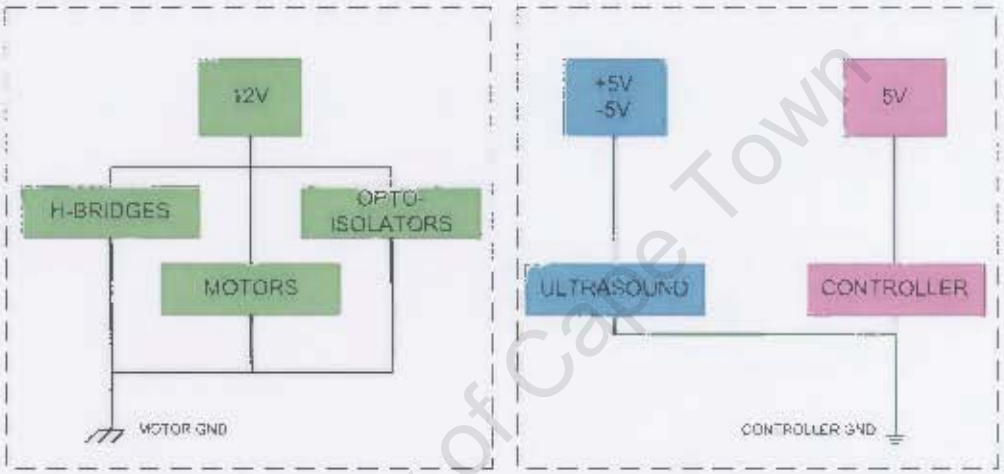


Figure 4.4: Power Supply Schematic

4.1.5 Offboard Computer

The system also included a PC running MATLAB which was used to simulate the robot and then to process test data and compare it to the simulation. This processing is intended to eventually occur on the Gumstix embedded computer but for simplicity during testing, the offboard computer was more suitable.

4.2 Embedded Software

As the low-level controller, the Roboaudiostix was responsible for driving the robot and operating the sensors. Driving the robot consisted of providing a pulse width modulation (PWM) signal to each H-bridge, reading and interpretting the wheel encoder signals and executing a control loop to maintain the given speed. The PWM signals used a 16-bit timer unit with a dedicated

PWM module [21]. Obtaining the wheel speeds from the encoders also required a timer unit (8-bit) and utilised two external interrupt pins into which the encoder outputs were fed. Detail of this is provided in appendix D.

The ultrasonic system required the microcontroller to both initiate the chirp and detect the response. A further 8-bit timer module was used to measure the time between the transmission and the received return signal. An external interrupt was used to detect this return pulse and software rejection of false triggers was included.

Because the speed and ultrasonic data had to be recorded at specific fixed intervals in order to be used by the localisation algorithm, another 16-bit timer module was utilised. This timer overflow value determined the time-step length and caused an interrupt when this time had passed. At each time-step, the speed, direction, ultrasound reading and a time-stamp were recorded to a circular buffer. This buffer was then read by the Gumstix at regular intervals and the data written to a file to later be processed offboard.

Communication between the Roboaudiostix and the Gumstix used the I^2C communication protocol. This allowed the Gumstix to send speed and direction information to the Roboaudiostix and for the Roboaudiostix to transfer stored data back to the Gumstix. The communication method and code is discussed in appendix C.

The system was designed so that the Gumstix could perform high-level tasks such as localisation and mapping although in this work, the complex operations were performed offboard to test the performance and feasibility of the system. The Gumstix did however play a role in this system. Firstly, it was used to program the Roboaudiostix microcontroller using I^2C. Secondly, it communicated with the Roboaudiostix during operation, periodically reading the data storage buffer, writing it to file and providing the robot with the required speed values. The reliability and ease of use of the Gumstix system was demonstrated and indicated the suitability of the system for future development.

4.3 Simulation Software

A large part of this project involved developing a simulation of the system to test the algorithms and develop specifications. This simulation was performed in MATLAB. An implementation of the Extended Kalman filter

localisation algorithm was also written in MATLAB and run on an offboard PC. This was done to test the algorithm in an easy-to-use environment before implementation on the embedded system. Details of the computer modelling are discussed in the following chapter.

Chapter 5

Computer Modelling

5.1 Introduction

X	Robot State Vector
EKF	Extended Kalman Filter
ϵ_{sim}	Simulation Error
ϵ_{ekf}	Estimated Error used by EKF
z	Sensor Measurement
P	State Error Covariance Matrix
R	Measurement Noise Covariance Matrix
Q	Process Noise Covariance Matrix
θ	Orientation Angle
ϕ	Bearing to Observation

Table 5.1: List of Symbols

A computer model of the robot and its environment was developed to test the operation of the localisation algorithm and the effect of various system parameters on its performance. It was desirable that the localisation algorithm could be tested on simulated robot motion and sensor data as well as on data from actual tests. Therefore, the localisation algorithm was developed as an isolated unit which required robot odometry and sensor data as an input. This data could come from physical tests or from a computer simulation. The computer simulation was developed initially to test the localisation algorithm. A further requirement was to determine the required

specifications for the robot sensors and driving accuracy to ensure successful localisation with the extended Kalman filter algorithm.

The structure of the computer model is shown in the block diagram, figure 5.1.

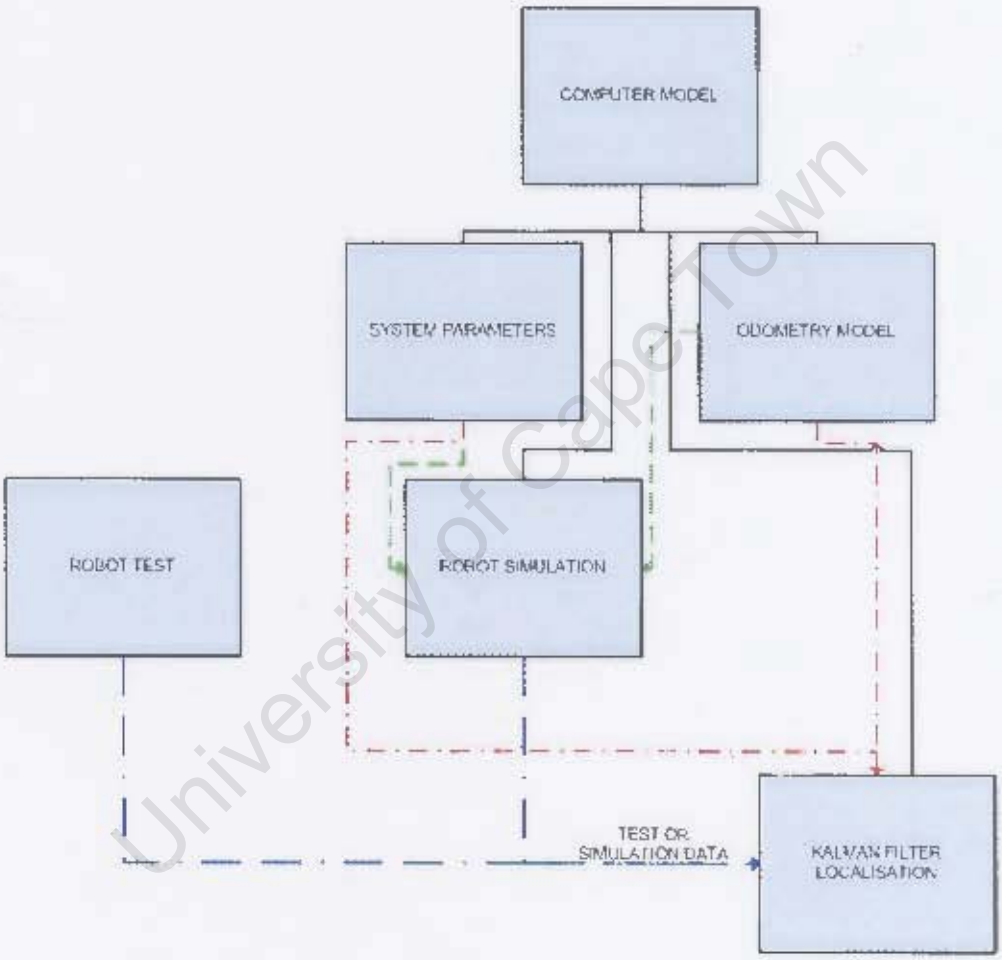


Figure 5.1: Computer Model Block Diagram

The localisation algorithm and the robot simulation share some basic system parameters which are initialised at the start of the computer model.

5.2 Initialisation

The first part of the computer model consists of initialising the parameters which define the environment. The system is modelled by dividing the total time into small timesteps and calculating the system parameters at each step. This is done in both the robot simulation and the EKF localisation so the timestep length and total number of timesteps are initialised at the start of the model.

There is a known map of the environment which contains the x and y co-ordinates of beacons which can be detected by the sensors and used to correct the robot position. The robot itself has a state, X, $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ which consists of the x and y co-ordinates and orientation angle, θ. This is set to a known initial value which is used in both the simulation, odometry model and EKF algorithm which are described below.

5.3 Error Modelling

An important component of both the system simulation and the EKF algorithm is the addition of various error terms. There is error used in the simulation to create a realistic path and sensor data. This will be referred to as ϵ_{sim} and consists of both the vehicle pose error as well as the sensor error. There is also error used in the EKF algorithm which is used in the covariance matrices and is the estimated error in the system. This will be referred to as ϵ_{ekf}. The difference between these error terms and their relationship is described below.

Firstly, there is error in any model of a real-world system so in order to create a realistic model of a robot moving in an environment, this error has to be included. The simulation of the robot uses a vehicle process model (Appendix F on page F-1) to determine the robot path based on wheel velocities. This however would never be entirely accurate due to wheel slip, inaccuracies in the velocity values and assumptions and approximations made in the process model. To compensate for this and create a more realistic path, an error term, ϵ_{sim}, is added to the vehicle centroid at each timestep. This error is a random Gaussian error with a specified standard deviation which would be determined from testing.

The fact that there is error in modelling systems plays a key role in the EKF and it is therefore important to quantify this error. In a real-world system, the various errors would be measured as far as possible and an estimate for the standard deviations, ϵ_{ekf}, would be used in the EKF. The closer these values are to the actual error, the better the performance of the filter. This is demonstrated by the results in section 5.8 on page 34.

In an ideal system, ϵ_{ekf} would be equal to the actual error in the system. Of course, in a simulated system, the error is fully known because it is specified and used to create the simulated data. It would therefore seem that ϵ_{ekf} should be set equal to ϵ_{sim}. This however would give the EKF better performance than could realistically be expected as this value could never be accurately known. ϵ_{ekf} and ϵ_{sim} were therefore separated in the code so that the affect of differing these values could be tested.

5.4 Simulation

A simulation of the robot moving in the environment and gathering sensor data was performed to provide realistic data for the EKF and to test its performance. The simulation requires the wheel velocities which would be applied to the wheels at each timestep. This is specified in the computer model but in reality would be the odometry data from the wheel encoders.

5.4.1 Path Simulation

The simulation begins with the robot at the initial position. At each timestep the new state vector is calculated using the velocity information. The process model (appendix F on page F-1) is used to calculate the new state. An error is then added to this state to simulate wheel slip and errors caused by odometry inaccuracies. The position and orientation at each timestep is stored to create the simulated robot path.

5.4.2 Sensor Simulation

This simulated path is used to simulate the sensor readings. At each timestep it must be determined whether the robot can see a beacon with the given sensor specifications. If there is more than one beacon visible, the sensor data will be that of the closest beacon. As the robot currently only has one ultrasonic sensor, only one beacon is seen at a time, but this could easily be

extended if further sensors were added.

Determining the Visible Beacons

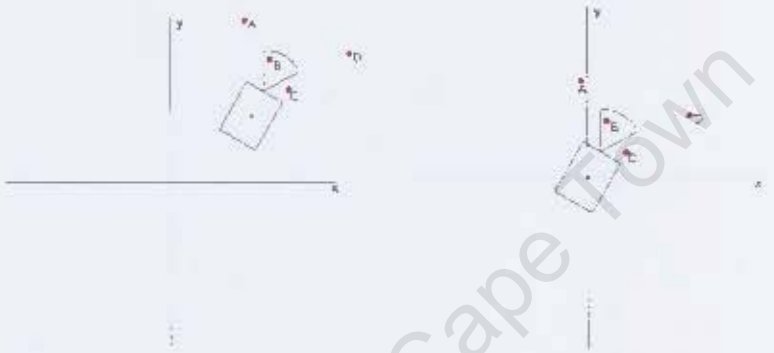
In order to determine whether a beacon is in sensor range, the maximum range and field of view of the sensor are specified. This creates an arc within which a beacon can be seen, referred to here as the visibility envelope. To determine whether the beacon falls into this arc, the beacon position must be related to the robot position. This is done by subtracting the robot centroid from the beacon co-ordinates and then rotating it about the origin by $-\theta$. This effectively places the robot centroid at the origin with an orientation angle of zero which simplifies the calculation of the visibility envelope. The range and bearing of the beacon with respect to the origin are then determined. If these fall within the specifications of the sensor, the beacon is visible to the robot and the co-ordinates of the beacon are passed back to the simulation.

The procedure for determining if a beacon is visible is illustrated in the following diagrams.

Figure 5.2(a) shows the robot and beacons in their initial position. The beacon labelled B is in the robot's field of view. Figure 5.2(b) shows the robot and beacons after the centroid has been subtracted, placing the robot at the origin. It can be seen that the beacon positions relative to the robot have not been altered. Finally, figure 5.2(c) shows the beacons and robot after the rotation has been performed. This sets θ to zero, aligning the robot with the co-ordinate frame to simplify calculations. The beacon positions relative to the robot are still unchanged.

Determining Sensor Data

Once it has been determined that a beacon is visible to the robot during a particular timestep, a sensor reading, $z = \begin{bmatrix} r \\ \phi \end{bmatrix}$ is created. This consists of a range, r, and bearing, ϕ, to the target beacon. Gaussian errors, $error_{range}$ and $error_{bearing}$ are added to each of these values to simulate sensor inaccuracies. The range and bearing are therefore determined by



(a) Initial Positions of Robot and Beacons

(b) Robot and Beacons with Centroid Subtracted

(c) Robot and Beacons after Rotation about the Origin

Figure 5.2: Determination of Visible Beacons

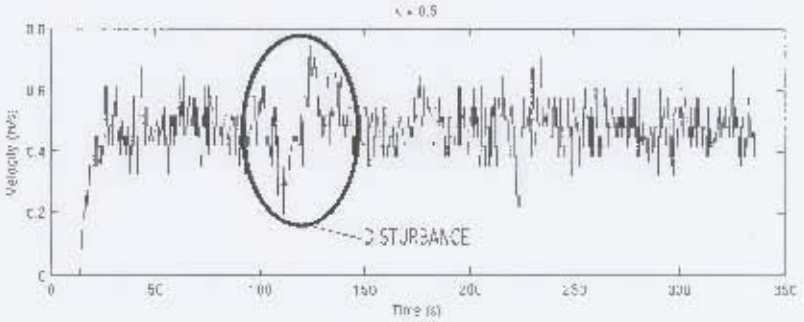


Figure 6.3: Step-test with k = 0.5

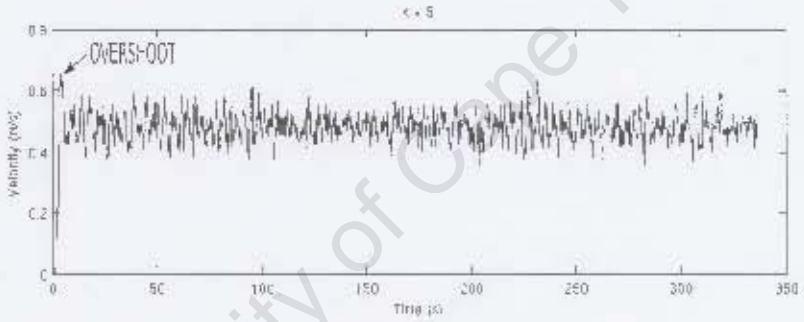


Figure 6.4: Step-test with k = 5

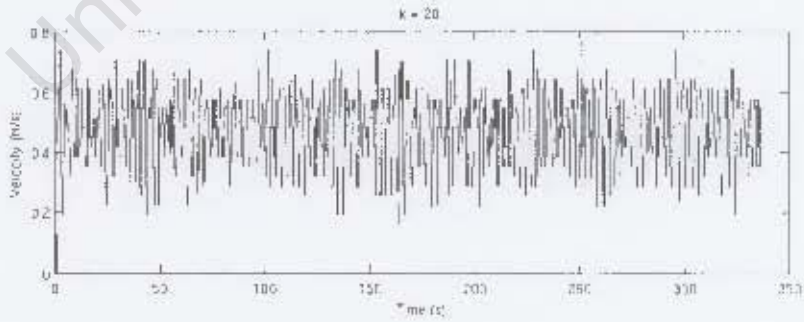


Figure 6.5: Step-test with k = 20

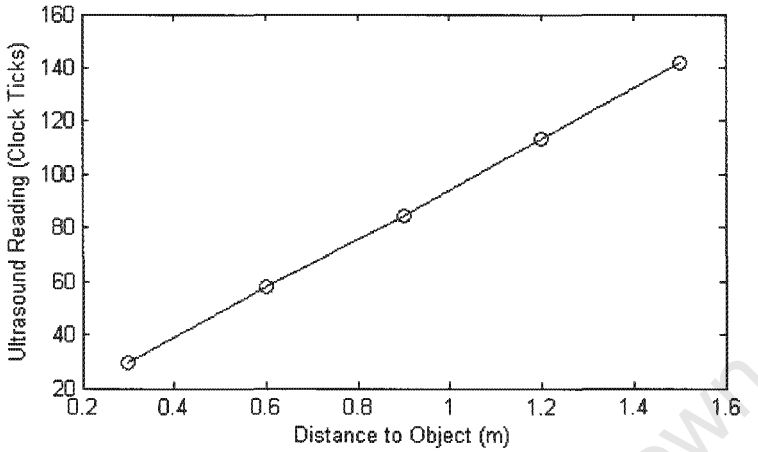


Figure 6.6: Ultrasound Calibration

The robot could detect a rectangular target with an area of $0.075m^2$ at a range of $1.2m$ with a probability of detection of 86% and at 1.5m with a probability of detection of 20%.

Another test considered ultrasonic readings as the robot moved. The sensor readings were recorded as the robot approached a target. Figure 6.7 shows a graph of this result. Initially the target was not visible but after some time, the target was observed. The distance to the target then decreased with time as expected before remaining constant once the robot had come to a stop.

Figure 6.8 shows the result of a similar test with two beacons. The robot approached the first beacon and then turned away before approaching the second beacon. The result is as expected with no targets visible initially. Sample 22 shows the first sighting of the first beacon, followed by a decreasing range as the robot moved closer to the beacon. It then lost sight of the first beacon until sample 66, where the second beacon was sighted. This range also decreased as the robot approached the beacon before reaching a constant value when the robot was stopped.

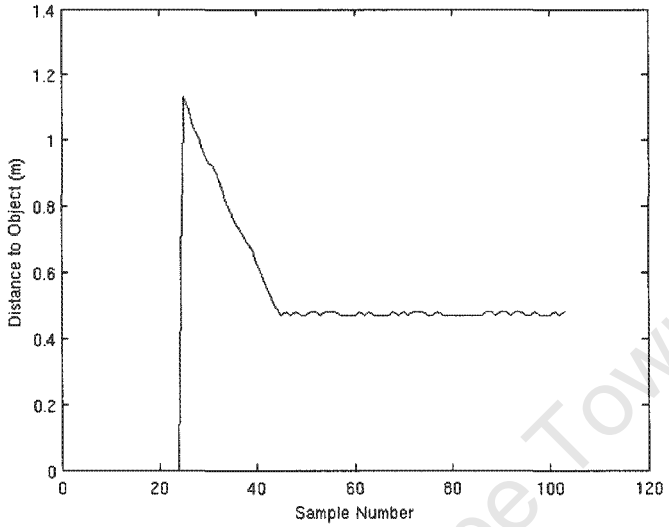


Figure 6.7: Ultrasound Test with Robot Moving - One Target

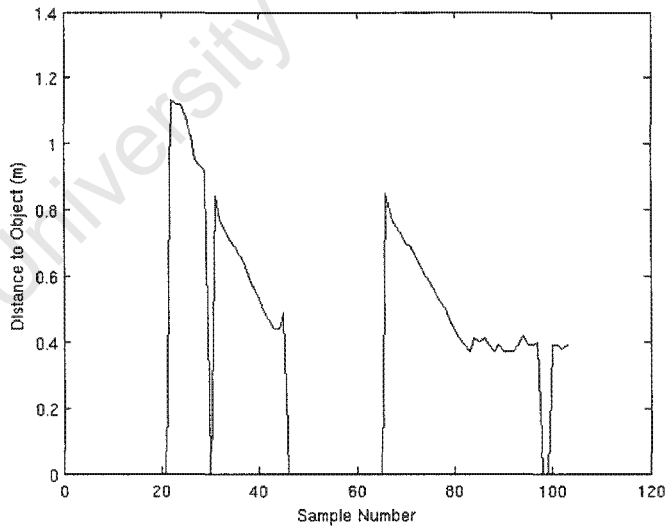


Figure 6.8: Ultrasound Test with Robot Moving - Two Targets

6.3 Localisation

To test the localisation algorithm, the robot had to drive along a path taking odometry and sensor readings. Beacons were placed in known positions to provide a map of the environment from the which the robot pose could be determined.

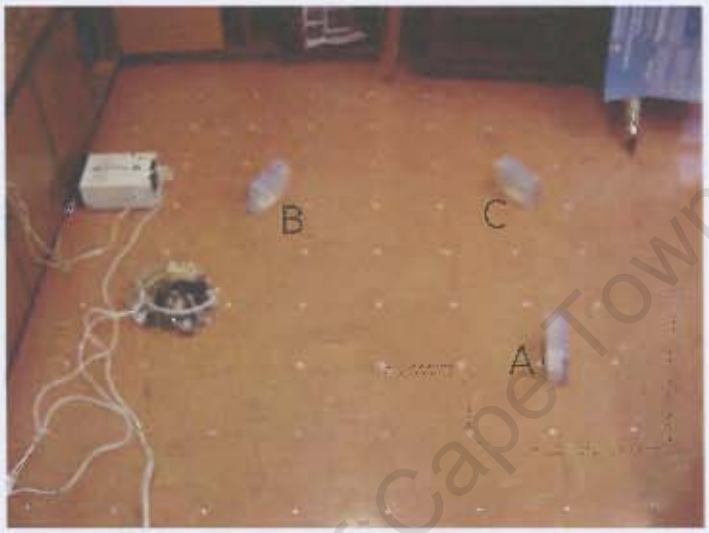
In order to verify the localisation results, the actual path the robot took had to be determined. This of course included any errors due to wheel slip and so could not be measured directly by the robot. To record this path, a video of the test was taken and a grid marked on the ground in the test area. This grid was used in post-processing of the video frames. Each frame was altered to compensate for the camera angle and provide a rectangular grid for accurate measurement of the robot position. An affine transform [22] was applied using the ImageMagick package [23] to perspective correct the images. The four corners of the grid were mapped to the corners of the corrected image. This procedure is demonstrated in figure 6.9. The image before correction is shown in 6.9(a) while the corrected image is shown in 6.9(b). Although the second image looks distorted, the white dots now form a regular grid which can be used to manually determine the robot co-ordinates. Figure 6.10 shows the camera set up on a tripod above the test area.

Figure 6.11 shows the video stills from a sample test using three beacons. In these images, the robot begins at the start position and then moves towards beacon A. It then turns left and starts moving towards beacon B. Finally, it turns right and heads towards beacon C before coming to rest. The results of this and other tests are discussed in the following sections. Videos of these tests can be found on the accompanying CD.

Data Processing

Before the raw test data could be used in the EKF localisation algorithm, some data processing was required. The ultrasound data had to be converted from microcontroller clock cycles to range measurements. This was done using the calibration factor determined from figure 6.6. Any range readings greater than 1.25m were ignored because this was outside the test area and was attributed to environmental clutter.

The wheel speed readings also had to be converted from encoder ticks per timer cycle to velocity in ms^{-1}. The direction of the wheels also had to be



(a) Test Area before Perspective Correction

(b) Test Area after Perspective Correction

Figure 6.9: Correcting Video Image for Camera Angle

$$z = \begin{bmatrix} \sqrt{(Beacon_x - Robot_x)^2 + (Beacon_y - Robot_y)^2} + error_{range} \\ \arctan(\frac{Beacon_y - Robot_y}{Beacon_x - Robot_x}) - Robot_\theta + error_{bearing} \end{bmatrix}$$

5.5 Simulation Output

During the simulation, the robot state, X and the sensor readings are stored at each timestep and form the output of the simulation stage. The sensor data is then used by the EKF localisation stage to update the state estimate and the robot state data is used as a comparison to test the performance of the algorithm. The closer the EKF output to the simulated path, the better the performance of the algorithm.

5.6 Odometry Model

An odometry model is also created for comparison with the localisation output. This is the estimated robot path based on odometry alone. A comparison between this odometry output and the EKF output gives an indication of the performance of the filter and the requirements of the sensors. This odometry model can take the real odometry data from an actual test or it can take the generated velocities used for the simulation.

5.7 Extended Kalman Filter Localisation

The extended Kalman filter localisation algorithm described in section 2.2.2 was implemented. It takes inputs from the robot simulation or from actual test data. Its output is the state of the robot at each timestep.

The EKF itself has various parameters which have to be initialised. These are the estimated sensor errors and estimated process errors and state error covariance. The state error covariance matrix, P, is initialised to

$$P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This represents the uncertainty of the initial position. In this case, it is assumed that the initial position is perfectly known. If the position is not

perfectly known, the diagonal elements of the P matrix are set to the variances of the error terms.

The measurement noise covariance matrix, R is initialised to

$$R = \begin{bmatrix} \epsilon^2_{range} & 0 \\ 0 & \epsilon^2_{bearing} \end{bmatrix}$$

where ϵ^2_{range} and $\epsilon^2_{bearing}$ are the estimated variances (squared standard deviation) of the measurement noise.

The process noise covariance, Q, is determined by

$$Q = \begin{bmatrix} \epsilon^2_x & 0 & 0 \\ 0 & \epsilon^2_y & 0 \\ 0 & 0 & \epsilon^2_\theta \end{bmatrix} \tag{5.1}$$

where $\epsilon^2_x, \epsilon^2_y$ and ϵ^2_θ are the estimated variances (squared standard deviation) of the process noise. These values however cannot be calculated initially as the state error depends on velocity. The Q matrix is therefore recalculated at each timestep based on the current velocity.

At each timestep, the velocity is retrieved, either from actual odometry data or from the simulation input. The sensor data (actual or simulated) is also retrieved. The process error is then calculated based on the velocity at the current timestep and this error is used to determine the covariance of the process noise, Q.

A prediction for the current state is then determined using the previous state and the process model described in appendix F on page F-1. The jacobian, ∇f, of the process model is determined at the current timestep, linearising the function around the current state. The variance prediction is determined from Equation 2.13 on page 8.

If there is a valid observation, this observation needs to be related to a beacon. The range and bearing are converted into x and y co-ordinates of a landmark as observed from the current predicted position. The closest beacon to these co-ordinates is assumed to be the visible beacon. This method had to be updated after testing. Because orientation errors after turning were so large, the wrong beacon was being tracked. The modification made was to disregard beacons which were not in the robot's field of view. If there was

no visible beacon, the sensor data was simply ignored and only the process model update performed.

The innovation and Kalman gain are then calculated from Equations 2.15 and 2.16 respectively. Finally, the variance and state are updated and the final estimate for the state is stored.

5.8 Results

A sample simulation and EKF result is shown in figure 5.3. It is clear that using odometry alone to determine position is insufficient as errors grow without bound and no correction is made. This is in contrast to the EKF result. Because of the combined odometry and ultrasound sensor data incorporated in this estimate, it is much closer to the actual path than with odometry alone.

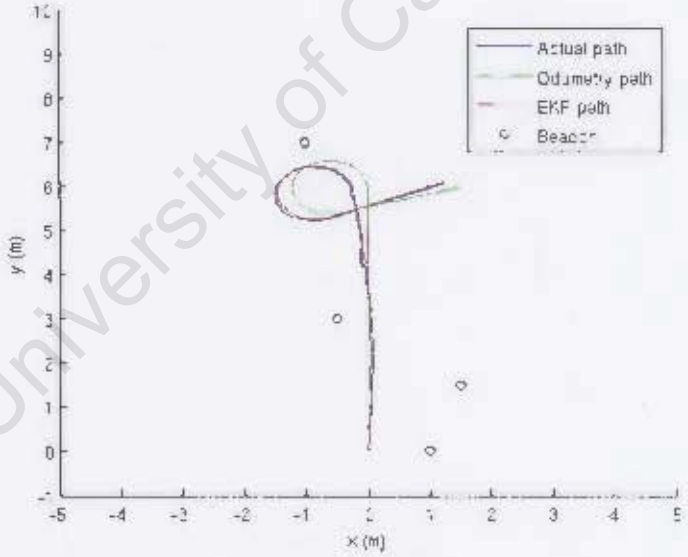


Figure 5.3: Result of EKF localisation using simulated data

The result of this simulation demonstrated the correct operation of the EKF localisation algorithm. The method is however sensitive to changes in the chosen error values. Figure 5.4 shows another simulation. In this run however, the sensor error terms used by the EKF were overestimated.

In this case, the filter puts little faith in the sensor data and the output is

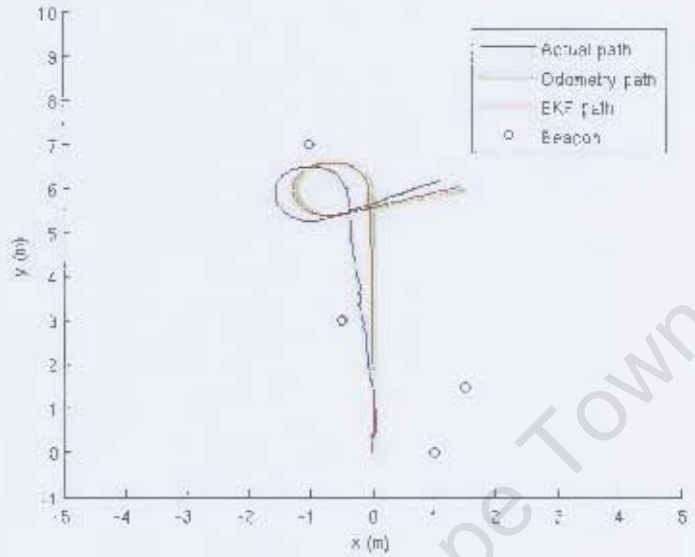


Figure 5.4: Result of EKF localisation with overestimated sensor error

very similar to the odometry path. This of course does not match the actual path because the sensor data was more accurate than assumed by the filter and should have been more heavily weighted in the position estimate.

The opposite is true if the EKF pose error term is overestimated. Figure 5.5 shows this case. As the odometry data is not heavily weighted, when sensor data is obtained, the filter follows this information more closely. This is demonstrated by the abrupt change in position shown in the EKF localisation path. At this point, sensor data was obtained and because the odometry data was assumed to be inaccurate, the filter was able to alter the estimate in favour of the sensor information.

These results demonstrated the successful implementation of the extended Kalman filter localisation algorithm and the robot simulation. The EKF algorithm however, had to be tested with actual experimental data to verify its suitability for this system. These and other system tests are described in the following chapter.

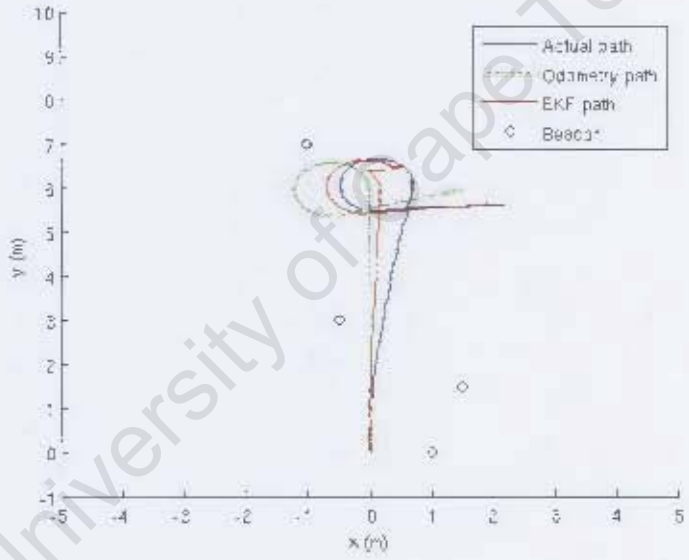


Figure 5.5: Result of EKF localisation with overestimated pose error

Chapter 6

Testing

The testing of the system was broken down into a number of tests of various aspects. These included the speed control system, the ultrasound system and verifying the operation of the extended Kalman filter localisation algorithm with experimental data.

6.1 Speed Control

The speed control algorithm consisted of a proportional controller, illustrated in figure 6.1.

The speed control tests consisted of testing the operation of the control algorithm and then tuning the controller gain (k) value by performing step-tests. The procedures and results are outline in appendix D. The result of a step-test performed with the chosen k value is shown in figure 6.2. This value was chosen because it provided an adequate speed of response while limiting overshoot and oscillation.

The following figures show the system responses for various k values to illustrate the effect of the controller gain on the system and to explain the chosen value.

Figure 6.3 shows the system response with $k = 0.5$. The response is extremely slow and is unable correct itself quickly after environmental disturbances. The circled area shows a disturbance such a slight incline, dropping the speed below the setpoint. In this area on the graph, it can be seen that a

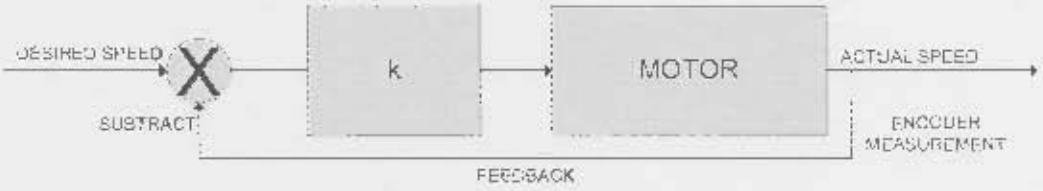


Figure 6.1: Proportional Controller [2][P.57, Chapter 4]

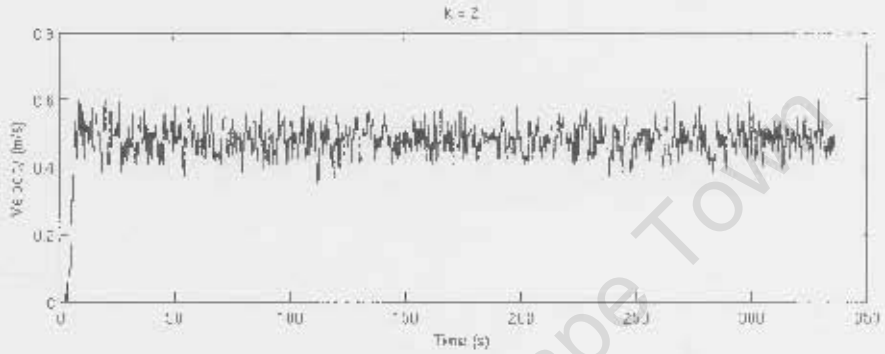


Figure 6.2: Step-test with k = 2

significant amount of time is taken for the system to return to the setpoint. By this time, the environment has changed, now level or a slight decline, and the system is unable to respond fast enough so overshoots the setpoint. This is an indication that a value of $k = 0.5$ is too low for this system.

Figure 6.4 shows the system response with a k value of 5. The response is much faster but there is visible overshoot which is undesirable.

Figure 6.5 shows the system response with k greatly increased to a value of 20. The system is clearly oscillatory and does not stabilise and reach the setpoint.

6.2 Ultrasound

Initially ultrasound calibration tests were performed with the robot stationary. A target was placed at varying ranges in line-of-sight of the ultrasonic transducers and the maximum range and likelihood of detection determined. These results are shown in figure 6.6. It should be noted that the transducers could only provide range data and not bearing.

Figure 6.10: Camera Set Up on Tripod Above Test Area

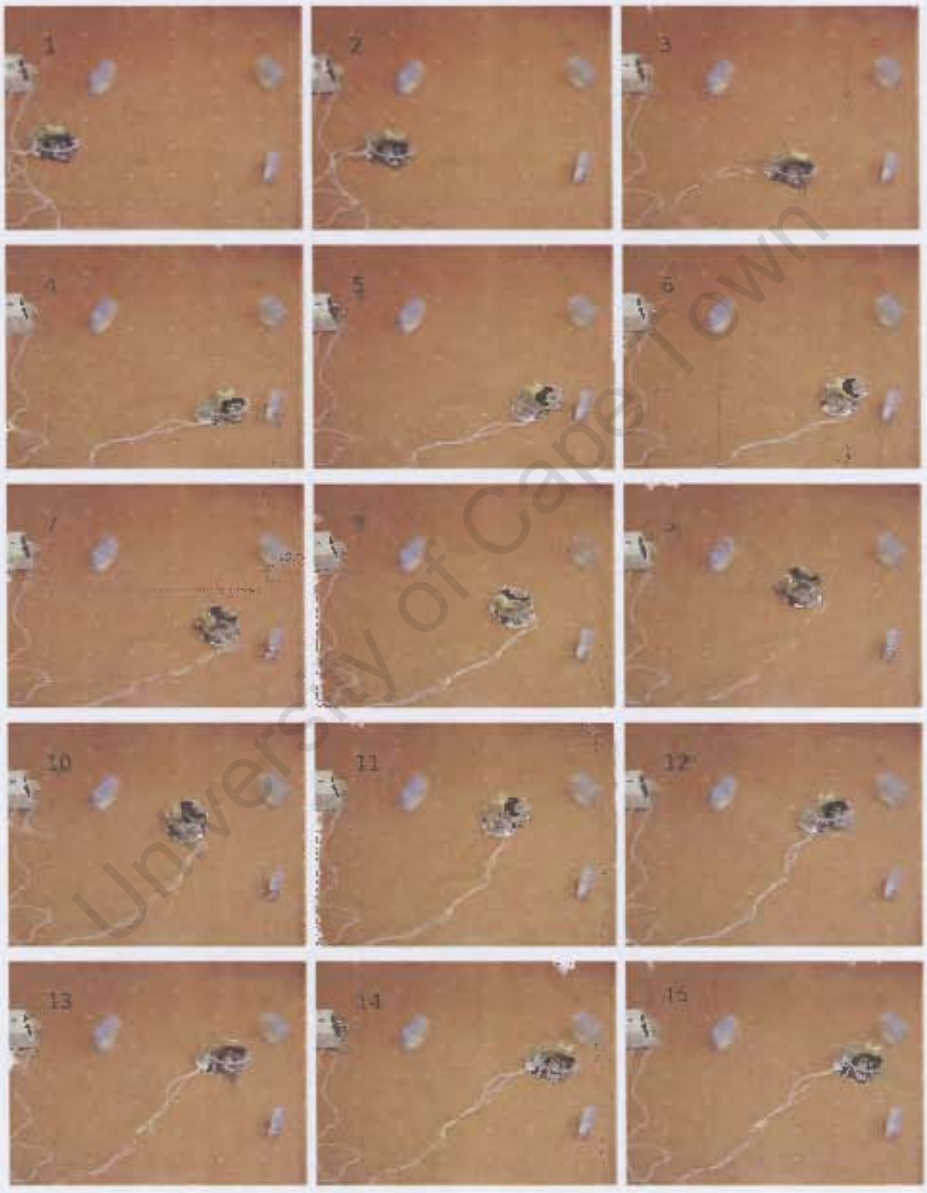


Figure 6.11: Video Stills of an Example Test Run

taken into account. In the EKF algorithm, the sign of the velocity value indicated direction whereas in the microcontroller code, the speed was a scalar value and a second variable specified direction. These had to be combined to give a signed velocity.

Single Beacon

In the initial test, the robot drove towards the single beacon and stopped. Figure 6.12 shows the actual path, the path determined by odometry data alone and the path determine by the extended Kalman filter localisation algorithm. The localisation algorithm was extremely successful, producing results far closer to the actual path than from odometry alone. The accuracy of the method is highlighted by the fact that the end point of the path determined by the extended Kalman filter matches the actual final position.

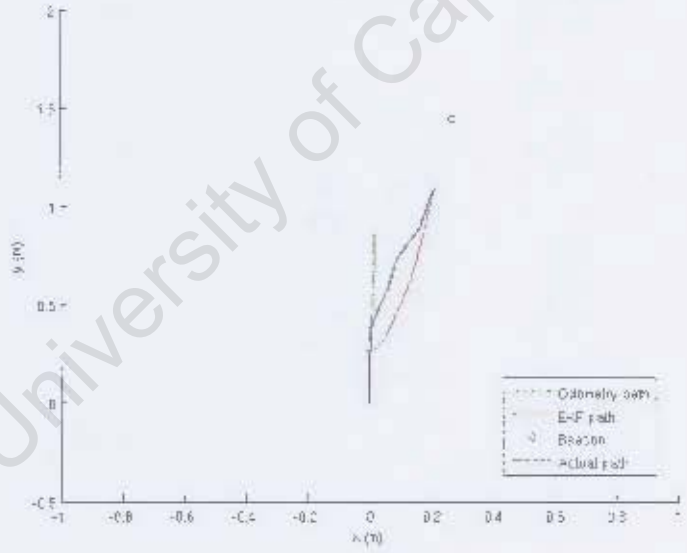


Figure 6.12: Comparison of Odometry and Localisation Data with Actual Path - 1 Beacon

It should be noted that the extended Kalman filter estimate follows the odometry for a while before branching off towards the actual path. This point where the EKF path branches to the right can be attributed to the first observation of the beacon. This observation was used to correct the estimate and therefore move the estimated position closer to the actual position. This is confirmed by ultrasound readings from the test. These are shown in figure

6.13. Range values greater than 1m from the transducers (1.25m from the wheel base) were discarded because they were outside the range of the test area and were therefore attributed to environmental clutter.

Figure 6.14 shows the velocity readings for both wheels. Comparison with the ultrasound readings is as expected. Initially, the beacon was out of range of the transducers. The wheel velocities increased and the robot moved forward. After thirteen samples, the beacon was observed at a range of approximately 1m. The wheel velocities remained constant and so the range to the beacon decreased steadily until sample 23 where the velocities dropped to zero and the robot stopped. The ultrasound readings then remained constant, with the robot stationary, approximately $0.36m$ from the beacon.

Two Beacons

Figure 6.15 shows a similar test with two beacons. This was not successful as the algorithm was not aware of which beacon was being observed at a given time. The way in which the algorithm attributes an observation to a beacon in the map is as follows. The current estimate for the robot pose is used and from that, and from the ultrasound data, the likely position of the beacon is determined. The closest beacon in that area is then determined to be the observed beacon. This however did not work as the error during turning was so large, the algorithm chose the wrong beacon.

The code was then modified to only accept beacon measurements if that particular beacon could realistically be in view. If the algorithm did not know which beacon to attribute the measurement to, it disregarded the ultrasound data and only followed the odometry data. This result is shown in figure 6.16. Again the filter was unsuccessful although it can be seen that instead of observing an incorrect beacon, the algorithm simply follows the odometry data. This performance was still unsatisfactory and in fact, because of the shape of the path, the filter performed worse than odometry alone. The task of matching up observations with beacons is known as correspondence and is one of the most difficult challenges in localisation and mapping. It is discussed in section 2.5 on page 11.

Figure 6.17 shows the same test except with known correspondence, i.e. the algorithm knows which observations belong to which beacons. This gave much improved results. As in the single beacon test, the EKF filter result is far superior to odometry alone and is relatively close to the actual path. The divergence from the actual path and then sharp turn in the EKF path is

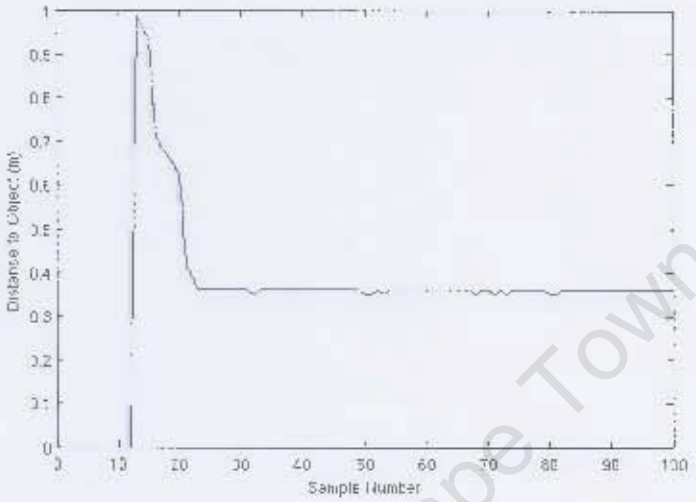


Figure 6.13: Ultrasound Readings for Single Beacon Test

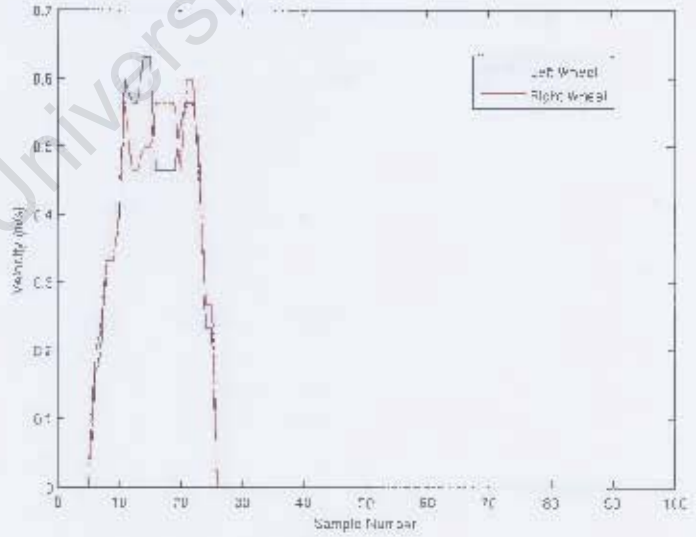


Figure 6.14: Wheel Velocities for Single Beacon Test

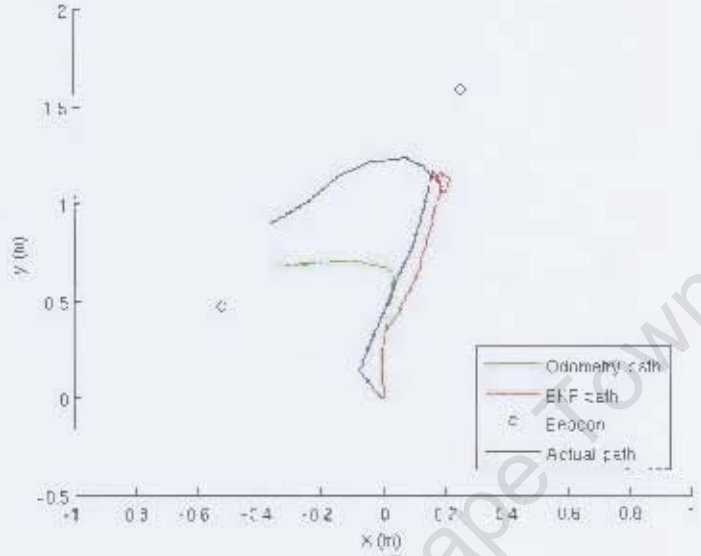


Figure 6.15: Comparison of Odometry and Localisation Data with Actual Path - Unknown Correspondence

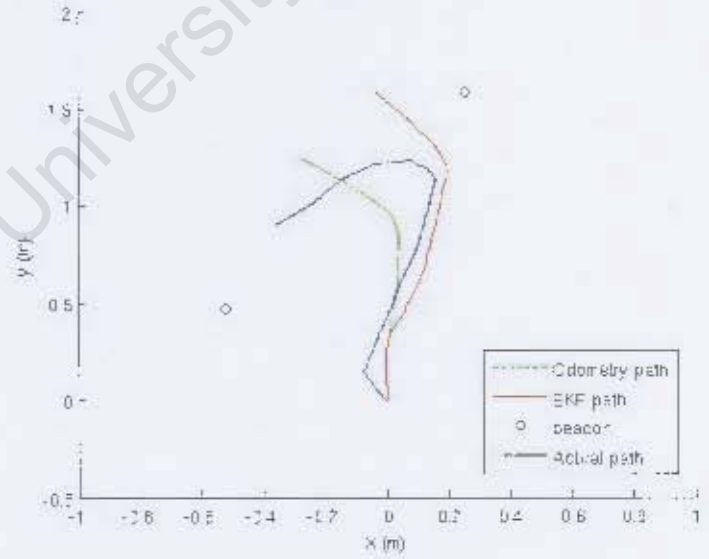


Figure 6.16: Comparison of Odometry and Localisation Data with Actual Path - Unknown Correspondence Corrected

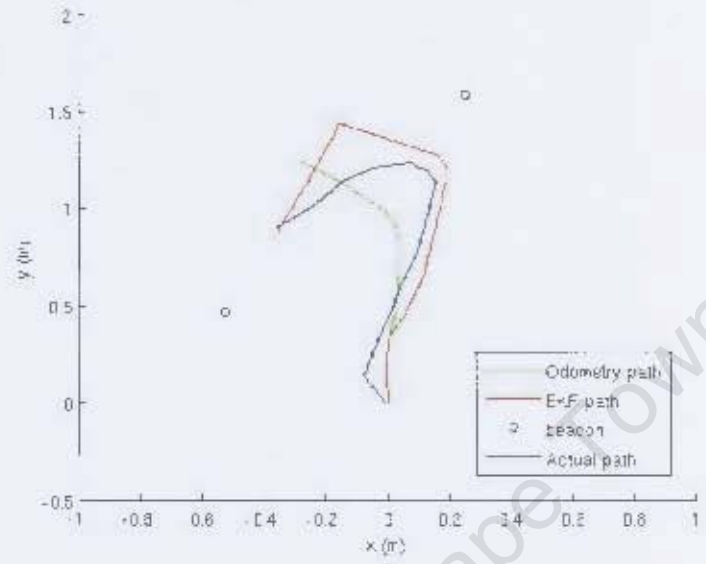


Figure 6.17: Comparison of Odometry and Localisation Data with Actual Path - Known Correspondence

where the ultrasound signal is lost and then regained and this used to correct the path. This is as expected as the robot turns the corner and loses sight of the beacons and is confirmed by the ultrasound data in figure 6.18.

These readings are as expected as nothing is seen initially, then the first beacon comes into view, followed by another loss in signal and finally the second beacon comes into view. There is some noise on the signal but this is compensated for by the filter and does not adversely affect the result.

Three Beacons

The test was once again performed but with a map of three beacons. Figure 6.19 shows the result with unknown correspondence. Once again, the result is unsuccessful with the algorithm unable to determine which beacon a particular observation belongs to and following the odometry information resulting in considerable errors.

Figure 6.20 shows the same test but with known correspondence. Again a vast improvement is shown with the EKF localisation algorithm closely matching the actual path.

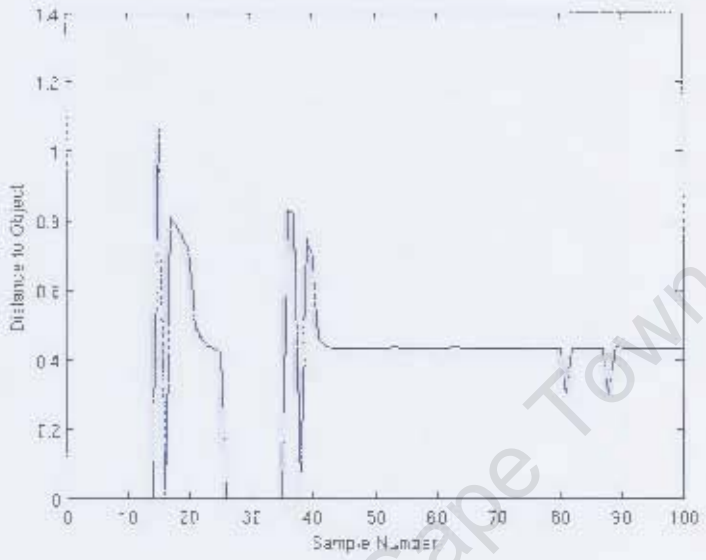


Figure 6.18: Ultrasound Readings for Two-Beacon Test

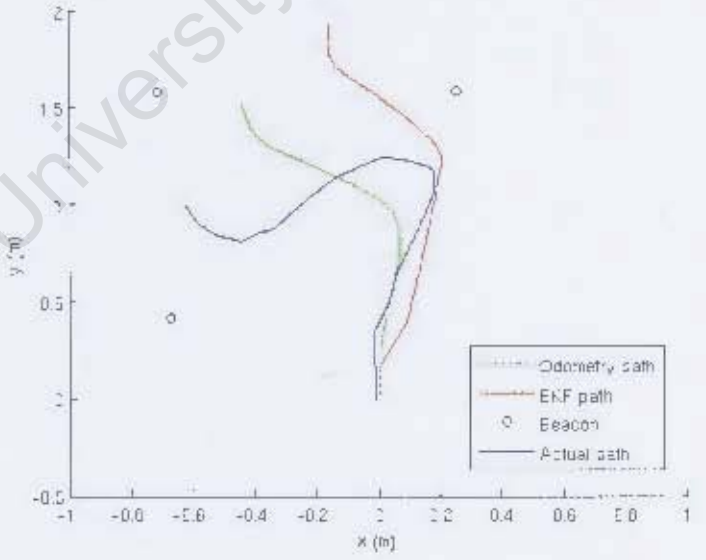


Figure 6.19: Comparison of Odometry and Localisation Data with Actual Path - Unknown Correspondence

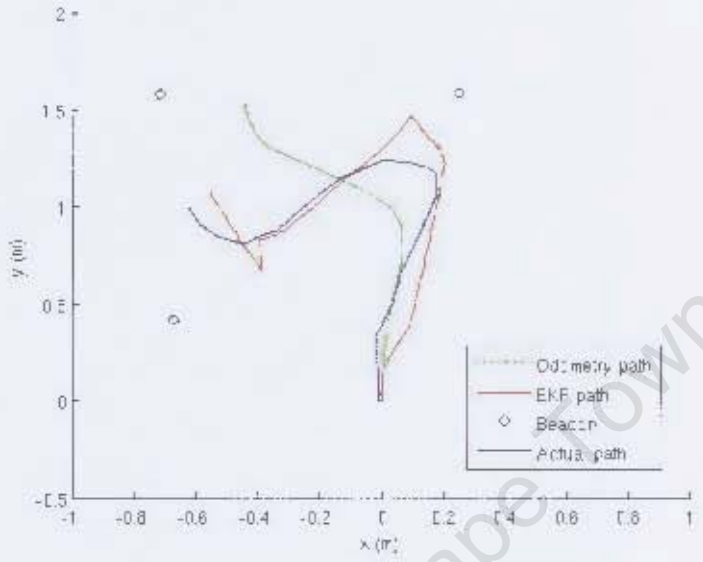


Figure 6.20: Comparison of Odometry and Localisation Data with Actual Path - 3 Beacons

The ultrasonic data is shown in figure 6.21. This is again as expected with three distinct spikes corresponding to the three beacons, with the range value decreasing as the robot approaches each beacon and ending with a constant range when the robot is stationary. This graph can be compared to figure 6.20 where it can be seen that the places where the EKF algorithm loses accuracy are where there is no ultrasound data available and the filter then follows the odometry.

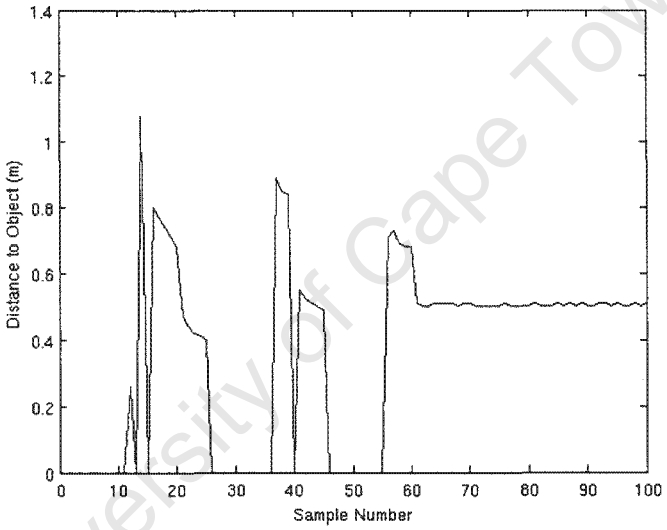


Figure 6.21: Ultrasound Readings for Three-Beacon Test

Chapter 7

Conclusions

7.1 Extended Kalman Filter Localisation

The system demonstrated that localisation of a robot in a known environment is possible using only low-cost ultrasound sensors and wheel speed encoders. The extended Kalman filter proved to be a highly effective method for this task if correspondence between observations and map information could be determined.

The method failed if correspondence was unknown due to large odometry errors especially during turning. When correspondence was known, the algorithm was extremely successful, giving results far superior to odometry alone. The effectiveness of the algorithm was demonstrated both with simulation and experimentation.

7.2 Speed Control

Speed control was implemented using optical encoders for rotational speed measurement. From the speed measurement, odometry was implemented both in the EKF algorithm and separately, for comparison with the filter output.

7.3 Ultrasound

An effective ultrasound range measurement system was developed and implemented. It had a maximum reliable range of approximately $1.2m$ for the given transmission pulse length. The effectiveness of this system was demonstrated by the EKF localisation results. When the sensor data was available and incorporated into the filter, the estimate of the robot position was much closer to the actual position than with odometry data alone. It should also be noted that the ultrasound data added considerable value to the system even though it did not provide bearing information.

A method of using multiple ultrasonic sensors simultaneously without mutual interference was investigated and appeared feasible. This would be highly advantageous if more sensors were added to the robot or if multiple robots were operating in the same area.

7.4 Controller

The Gumstix/Roboaudiostix embedded system was utilised as the controller for the robot. Although not all the features of this controller were used in this work, considerable potential for expansion was provided by choosing this platform. This will allow the future goal of running the EKF localisation algorithm onboard and eventually performing mapping to be implemented on the current hardware.

Chapter 8

Recommendations and Future Work

Solving the data association problem is a key area for future research. There are a number of methods which could improve the current system.

The first possibility is to add more sensors. A sensor which can detect a distinguishing property of a beacon, such as a camera detecting colour or shape would allow the algorithm to better determine which beacon a particular ultrasound reading corresponds to.

Another option is to include a direct orientation sensor such as a compass instead of determining orientation indirectly from the change in position. The reason for this is that a large portion of the error in data association was due to large errors in odometry during turning leading the algorithm to assume an incorrect orientation.

Self-identifying beacons, such as transmitters with a unique signal are another solution although this is not ideal for this application with the ultimate goal of autonomous mapping and navigation in an unstructured environment.

The investigation of the coded signal method of avoiding interference between conflicting ultrasonic sensors discussed in appendix G should be extended. The feasibility of performing the required signal processing on the Gumstix system should be investigated further. The performances of the methods should also be tested in cluttered environments as would be encountered in the final system.

The EKF localisation method used in this work should be implemented on the Gumstix and run in real-time during robot operation. This may require optimisation of the algorithm. As previously mentioned, the ultimate goal of this research is SLAM. EKF SLAM should be simulated as the localisation algorithm was in MATLAB to allow for easy testing. The SLAM algorithm could build on the completed localisation code. The final step would be to implement the SLAM algorithm on the Gumstix and then use this map and localisation information for efficient navigation of the environment.

References

[1] G. Brooker, C. Lobsey, and K. McWilliams, "Combined infrared and acoustic beacon tracker implementation on an autonomous vehicle," 2007.

[2] T. Braunl, *Embedded Robotics*. Springer, 2006.

[3] "Gumstix." `http://www.gumstix.com/`.

[4] "Irobot - home robots, roomba." `http://www.irobot.com/sp.cfm?pageid=122`. Last Accessed: 22 March 2006.

[5] "How stuff works - how robotic vacuum cleaners work." `http://home.howstuffworks.com/robotic-vacuum.htm`. Last Accessed: 22 March 2006.

[6] P. Young, "The development of an autonomous vacuum cleaning robot." 2005.

[7] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics,"

[8] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, 2001.

[9] S. Thrun, "Robotic mapping: A survey," 2002.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachusetts,United States of America: The MIT Press, 2005.

[11] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[12] J. K. Campbell, S. P. Synnot, and G. J. Bierman, "Voyager orbit determination at Jupiter," *IEEE Transactions on Automatic Control*, vol. ac-28, pp. 256–268, 1983.

[13] H. Durrant-Whyte, "Introduction to estimation and the kalman filter," January 2001.

[14] S. Riisgaard and M. R. Blas, "Slam for dummies: A tutorial approach to simultaneous localization and mapping,"

[15] J. Neira and J. D. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 890–897, 2001.

[16] J. A. Castellanos, J. Neira, and J. Tardos, "Multisensor fusion for simultaneous localisation and map building," *IEEE Transaction on Robotics and Automation*, vol. 17, pp. 909–914, 2001.

[17] D. M. Montemerlo and D. S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics.* Springer, 2007.

[18] "Lmd18200 datasheet." `http://www.national.com/pf/LM/LMD18200.html`.

[19] Murata, "Murata ma40 series datasheet." `http://www.murata.com/`.

[20] "H21a1 datasheet." `http://www.datasheetcatalog.com/datasheets_pdf/H/2/1/A/H21A1.shtml`.

[21] Atmel, "Atmega128 datasheet." book, 2006.

[22] E. W. Weisstein, "Affine transformation." From MathWorld — A Wolfram Web Resource. http://mathworld.wolfram.com/AffineTransformation.html.

[23] "Imagemagick." `http://www.imagemagick.org/script/index.php`.

[24] "Product reviews and info - Electrolux Trilobite." `http://www.onrobo.com/reviews/At\textunderscoreHome/Vacuum\textunderscoreCleaners/on00trilobroele/index.htm`. Last Accessed: 22 March 2006.

[25] "Welcome to the Electrolux Trilobite." `http://www.electroluxusa.com/node70.asp`. Last Accessed: 22 March 2006.

[26] "Product reviews and info - rc3000 cleaning robot." http://www.onrobo.com/reviews/At\textunderscoreHome/Vacuum\textunderscoreCleaners/on00rc3000rokac/index.htm. Last Accessed: 22 Marc 2006.

[27] "Friendly rv400." http://www.robotsandrelax.com/Vacuum.html. Last Accessed: 23 March 2006.

[28] "Friendly robotics robotic vacuum rv400." http://www.therobotstore.com/s.nl/sc.2/category.-109/it.A/id.43/.f. Last Accessed: 23 March 2006.

[29] N. Morrison, *Introduction to Fourier Analysis.* John Wiley & Sons, Inc., 1994.

[30] D. Hylands. http://docwiki.gumstix.org/Robostix_samples.

[31] L. Kleeman and R. Kuc, "Mobile robot sonar for target localization and classification," *International Journal of Robotics Research*, vol. 4, pp. 295–318, 1995.

[32] J. Borenstein, H. Everett, and L. Feng, *Where am I? Sensors and Methods for Mobile Robot Positioning.* 1996.

[33] "Cd4047 datasheet." http://www.datasheetcatalog.com/datasheets_pdf/C/D/4/0/CD4047.shtml.

[34] E. W. Weisstein, "Jacobian." From MathWorld-A Wolfram Web Resource. http://mathworld.wolfram.com/Jacobian.html.

[35] S. Kingsley and S. Quegan, *Understanding Radar Systems.* United States of America: SciTech Publishing, Inc., first ed., 1999.

[36] "Pulse compression." http://www.met.rdg.ac.uk/radar/ufam/pulsecomp.html. Last Accessed: 17 October 2006.

[37] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing.* Prentice Hall International, Inc., 1999.

[38] F. Stremler, *Introduction to Communication Systems.* Addison-Wesley, 1992.

[39] J. Walsh, "A Closed Set of Normal Orthogonal Functions," *American Journal of Mathematics*, 1923.

[40] K.-W. Jorg, M. Berg, and M. Muller, "Using pseudo-random codes for mobile robot sonar sensing," 1998.

Appendix A

Literature Survey

Contents

Figures

A.1 Introduction

An investigation of currently available autonomous vacuum cleaning systems was conducted to provide background information for this design. The literature survey was focused on the sensing abilities and behaviour mechanisms of the vacuum cleaners, as the mechanical design was already completed. Possible methods of achieving and improving these capabilities were also investigated.

A.2 Commercially Available Systems

The autonomous capability of commercial systems was investigated to provide information on the current state of the art and therefore to aid in the development of system specifications.

A.2.1 IRobot - Roomba

Figure A.1: IRobot's Roomba Scheduler

Figure A.1 shows IRobot's top-of-the-line robotic vacuum, Roomba Scheduler. It has the following capabilities:

- Object detection
- Active dirt detection
- Stair avoidance system

- Scheduling capability
- Automatic recharging

A.2.2 Floorbotics Floorbot

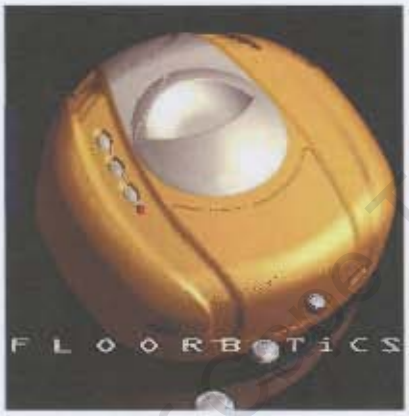


Figure A.2: Floorbotics Floorbot

Figure A.2 above shows the Floorbot from floorbotics. This robot also has the ability to detect objects and avoid drop-offs. It however does not have automatic recharge capability. The main advantage of this product is that it does not use a random search pattern but actively maps the room.

A.2.3 Electrolux Trilobite

Figure A.3: Electrolux Trilobite

The Electrolux Trilobite is shown in figure A.3[24]. It uses ultrasound to detect objects and navigate a room. It also has the ability to detect a low battery and recharge itself and can detect and avoid stairs or drop-offs[24, 25].

A.2.4 Karcher RC3000

Figure A.4: Karcher RC3000

Figure A.4 shows the Karcher Robocleaner RC3000[26]. It navigates by detecting obstacles using a contact sensor. It returns to base when a recharge is required and also to empty the dirt bin. Cleaning is continued once this has been completed.

A.2.5 RV400

Figure A.5 shows the RV400 from Friendly Robotics[27]. It uses ultrasound to navigate and avoid obstacles. It also has the ability to detect and avoid

Figure A.5: Friendly RV400

stairs and drop-offs[28].

A.3 Mapping and Navigation

The problem of autonomous navigation is not a trivial one. In order to be truly autonomous, a robot has to be able to not only sense objects within its environment but make logical decisions about the most effective action to take. This requires taking information from a system of sensors providing the raw data on the external environment. This information must then be assembled into a usable 'image' of the robot's surrounds. Finally, using this image, logical decisions about the best course of action to take must be made by the navigation algorithm. This entire process is dynamic and decisions must constantly be updated as the sensor information changes.

A.3.1 Localisation

Localisation is the process of determining robot pose (position and orientation). This is done with sensors and control information and vehicle process models. Localisation is obviously necessary for path-planning as the robot cannot choose a route to a destination without first knowing its position. It is also essential for the mapping process as in order to know the positions of landmarks, the vehicle position must also be known.

Environment Sensors

Localisation can be done using global or local sensors. In systems such as GPS, the beacons transmit a signal which is received by the robot. From the difference in arrival time between these signals and given that the position of the beacons is known, the position of the robot can be determined. The orientation however must be estimated from the change in position [2][P.198, Chapter 14].

In many environments, it is impractical for the beacons to transmit signals. In these cases, the transmitters could be mounted on the robot and the beacons could simply be reflectors, either natural or artificial but in known positions.

Odometry

Odometry is the process of using wheel encoder data for localisation. The starting position of the robot is known and future positions and orientations are calculated by integrating the wheel velocity data. This method is not reliable over large distances because errors grow without bound as there is no way to compensate for an incorrect measurement or wheel slip [2][P.200, Chapter 14]. It can however be used very successfully in conjunction with external environment sensors such as in the Kalman filter localisation algorithm described in section 2.2.2. Here odometry error is corrected by sensor measurements and also provides a way of compensating for errors in the sensor readings themselves. These two methods in conjunction provide a much more reliable localisation estimate than either would separately.

Probabilistic Localisation

There is uncertainty in all robot sensor measurements, movement and also in robot software because of approximations of the real world. Probabilistic localisation uses the mathematics of probability to estimate robot pose from all previous sensor and vehicle model data. This method includes not only estimating the most-likely current configuration but also the probability of this estimate. This means that because the current pose is a probability distribution rather than a single value, the robot can revise its estimate if a hypothesis becomes less likely when further measurements become available [10][P.4, Chapter 1][2][P.201, Chapter 14].

The advantage of probabilistic methods is that the required accuracy of the sensors and robot models is greatly reduced. There is however a high computational cost.

Extended Kalman filter localisation is often used in robotics. The extended Kalman filter is described in section 2.2.2. It requires motion and measurement models and a predefined map of beacons. The state of the robot is updated by the motion and measurement data, while the algorithm also keeps track of the likelihood of the estimates.

The algorithm also has to determine which beacon in the given map an observation corresponds to. This can be a difficult problem if there are many possible answers with a similar probability. This can sometimes be solved by structuring the environment, such as increasing the space between beacons. This however has the disadvantage of decreasing the likelihood of observing a beacon at a particular time-step and therefore increasing robot pose uncertainty [10][P.215, Chapter 7].

A.3.2 Simultaneous Localisation and Mapping (SLAM)

Simultaneous localisation and mapping (SLAM) is the process where neither the robot pose nor a map of the environment is known. These both have to be estimated at the same time, significantly complicating the procedure.

SLAM is done in a similar way to localisation, using the extended Kalman filter, but instead of only estimating the robot pose, the map is also estimated. As landmarks in the environment are re-observed, the estimates of their positions become less uncertain. The localisation part of the algorithm is exactly the same as in the localisation problem except instead of a fixed map, there is a constantly changing map of landmarks which is updated as new measurements are taken [8][14][10][P.309, Chapter 10].

Appendix B

Control System

Contents

Figures

Tables

B.1 Introduction

As it was a requirement that the robot be completely autonomous a fairly sophisticated controller was required. There are many possible controllers available, so a list of specifications was drawn up to aid in the controller selection.

B.2 Controller Selection and Layout

The controller requirements can be divided into two separate capabilities. The first requirement is the basic control of the sensors and actuators on the robot. This includes driving the motors and reading and storing data from various sensors such as wheel-speed encoders and proximity sensors. The second requirement is a higher level of control to perform room-mapping and navigation tasks.A decision had to be made as to whether these tasks should be controlled by a single central unit or whether the tasks should be split between two controllers, each suited to a specific task.

The advantages of a single controller over two separate controllers are as follows. A single controller would be simpler to implement in terms of hardware requirements. Having a central controller limits the amount of interfacing and communication required which greatly reduces the complexity of the system.

The disadvantages of this layout however, include the increased complexity in programming required and finding a controller suited to both low-level, real-time tasks as well as more abstract computation. The increased programming complexity comes about because while the complex navigation and mapping algorithms are carried out, the controller must still handle the simple yet essential tasks such as reading sensors and driving motors. These tasks must be completed in real-time and so make the scheduling of tasks within the program extremely complex. Added to this is the fact that different types of controllers are more suited to specific tasks. Microprocessors are suited to more simple tasks but can provide accurate timing very easily. Embedded computers can perform more complicated and abstract tasks but direct interfacing with hardware becomes more problematic. Choosing a single controller to perform both tasks would require compromise on the suitability of the controller to the given tasks.

An advantage of using two separate controllers is that they can run in parallel. This reduces the complexity of the required programs and timing difficulties.

Also, as mentioned above, specialised controllers can be chosen for the different tasks rather than trying to use an intermediate solution which is not ideally suited to either task. Another advantage is that the principles of subsumption architecture can be implemented. Because the basic control and behaviour of the robot is under the control of a dedicated microprocessor, if the more complex higher-level controller fails, the robot can still operate correctly. Although some of the functionality may be reduced, the robot could detect the failure of the higher system and take appropriate action. This redundancy is very desirable for this application which requires a fairly robust system.

The disadvantages of this layout are the increased cost associated with an additional controller although this is fairly minimal. There is also increased complexity in communication because the controllers have to communicate with each other in two directions. The low-level controller has to pass on sensor data to the central controller and behaviour instructions have to be sent back.

For these reasons, a system consisting of a low-level microcontroller to perform the simple yet essential tasks and a higher-level embedded system to perform the more complicated mapping and navigation algorithms was chosen.

B.3 Controller Requirements

B.3.1 Pulse Width Modulation

Pulse width modulation (PWM) is needed to control the speed of the motors driving the wheels. For this reason at least two PWM channels are required. There is a third motor which is used to drive the vacuum fan which may also need speed control to compensate for different floor surfaces. At least three PWM channels are therefore required for the system.

B.3.2 Analogue to Digital Conversion

The data from the ultrasound transducers will have to be read in using an analogue to digital converter. Other sensors may also require analogue to digital conversion

B.3.3 Digital to Analogue Conversion

As it is necessary to generate a complex transmission pulse for the ultrasound transducers, a digital to analogue converter is required to output this signal (which is generated in software) to the transmitter.

B.3.4 Minimum ADC/DAC conversion rate - 88kHz

The maximum frequency of the ultrasound transducers is 44kHz [19]. If the data is to be read directly into the microcontroller, the ADC must have a conversion rate greater than the Nyquist frequency of 88kHz (double the maximum frequency) [29]. This however is only a desirable feature as the ultrasound signal may be heterodyned to a lower frequency with external circuitry before being read into the microcontroller. This is also desirable for the the digital to analogue converter which needs to provide the transmitter signal at 44kHz.

B.3.5 General Purpose Input/Output

General purpose pins are required to read and control various sensors. An example of such a sensor is one which would detect sharp drop-offs or stairs so the robot could avoid these areas. At least one digital input pin would be required to read each drop-off sensor. Other sensors may also be controlled or read using digital input and output capabilities.

B.3.6 Communication

Two-way communication between the high-level controller and the microcontroller is necessary. This means that the controllers will have to be selected in conjunction with each other, ensuring that they have compatible communication systems.

B.3.7 Size and Power

To maximise the battery life of the robot, minimizing power consumption is essential. Using low power controllers as well as reducing the overall mass of the robot increases the battery life. Apart from the space constraints implicit in the mechanical design of the robot, it is desirable to have a controller which

is as small as possible. This allows for design changes and upgrades without concern for space constraints.

B.3.8 Development software

Open source development software is desirable as it eliminates licensing costs.

B.3.9 Programmable in C or C++

It is desirable for the microcontroller to be programmable in a higher-level language than assembler for ease of programming. As the final program will be fairly complex, a higher-level language such as C will make debugging simpler.

B.4 Controller Selection

B.4.1 Final Solution

The Gumstix embedded computer was chosen along with the Roboaudiostix and NetMMC expansion boards.

The Gumstix is shown in figure B.1 and its main specifications are shown in table B.1 on page B-8.

Figure B.1: Gumstix 400XM Connex [3]

The Gumstix is the high level controller in this system while the Roboau-

Table B.1: Gumstix Connex 400XM Main Specifications

400MHz Processor
64MB SDRAM
16MB FLASH
Low Power (draws < 250mA at 400MHz)
Operating System - Linux kernel 2.6.10
Open Source Software
Dimensions - 80mm x 20mm x 5.9mm
Weight - 8g

diostix performs the low-level tasks. Figure B.2 shows the Roboaudiostix board.

Figure B.2: Roboaudiostix Expansion Board [3]

The Roboaudiostix consists of an Atmega128 8-bit microcontroller as well as a UCB1400 audio codec. It can run as a standalone microcontroller or in conjunction with the Gumstix as in this system. The main specifications of the Roboaudiostix board are given in table B.2 on page B-9.

The NetMMC board shown in figure B.3 on page B-9 provides an ethernet connection and an MMC slot for MMC, SD or SDIO cards. This is used to interface with the host computer for programming.

The Roboaudiostix runs the low-level software which performs real-time tasks such as speed control and reading the ultrasound sensors. It then sends the required sensor data to the Gumstix embedded computer to be stored or processed by a localisation or mapping routine.

Table B.2: Roboaudiostix Main Specifications

ATmega128 AVR processor at 16MHz
41 AVR GPIO lines at 5V
6 x 16-bit PWM outputs
8 x 8 or 10-bit ADC
2 AVR UARTs at 5V
2 Gumstix UARTs at 5V
UCB1400 audio codec (AC97-compatible)
10 UCB1400 - GPIO lines at 3.3V
8 UCB1400 dedicated ADC channels at 3.3V
I^2C bus

Figure B.3: NetMMC Expansion Board [3]

As seen in the above table, the Gumstix system meets all the required control specifications as well as many of the desired features. The fact that the Gumstix and Roboaudiostix are designed to be used in conjunction with each other greatly simplifies the interfacing task. The extremely small size, low mass and low power of these controllers is an added advantage.

B.4.2 Other systems considered

PC104

A PC104 was considered as the central controller for this application. This was however eliminated due to the relative size and weight of the PC104

compared to the Gumstix system. It would also have a higher cost and would require additional boards for sensor input and motor control.

Two Microcontrollers

Using two microcontrollers such as the Motorola MC68HC08GT16 for the control system was considered. This was eliminated because a microcontroller such as this is not ideally suited to the high-level processing tasks of the central controller. The code required to perform the necessary calculations would be extremely complex and might require a number of microcontrollers running in parallel. This increased complexity was undesirable and would make fault-finding problematic.

Appendix C

Embedded System Software

Contents

Figures

C.1 Introduction

The software running on the robot was required to perform a number of tasks, many virtually simultaneously. For this reason, the microcontroller software was largely interrupt-based and so not easily described in terms of a linear program. In this section therefore, the different functions are described as separate units and details of their interaction with other units and the program as a whole is included. This was in fact how the program structure was built, with each unit being programmed and tested in isolation before being incorporated into the main code.

Parts of the code are based on software written by Dave Hylands [30]. The I^2C and circular buffer routines in particular were largely unchanged. The modifications that were made to the I^2C code were made in collaboration with Tracy Boosen, Mechanical Engineering, University of Cape Town. All code is included on the accompanying CD.

C.2 Speed Control

The initial task required for speed control is sending a PWM signal to the H-bridge chips. The circuitry and layout of the drive system is described in appendix D. A 16-bit timer with an incorporated PWM function is used to drive the motors. The frequency of the timer is set to 20kHz and the PWM duty cycle varied by changing the timer output compare register (OCR). The speed of the motor depends on the duty cycle of the signal to the H-bridge.

Once the timer register has been properly initialised for PWM function, no further action by the program is required to provide a PWM signal. This function simply runs in the background and provides a constant signal no matter what the current processor operations. To change the speed however, the duty cycle of the wave has to be changed which requires editing the OCR. This is done inside the control loop.

In order to control the speed of the wheels accurately, feedback is required. This is provided by optical encoders on the wheels, described in appendix D. Each encoder circuit provides a square wave with a frequency proportional to the wheel speed.

A traditional method for measuring the encoder signal and relating it to speed is to use input capture on the microcontroller. This is a function of

the timer modules which allows the time between successive events (a change in the logic level of an input pin) to be recorded.

Instead of using input capture, external interrupts are used to count the rising edges of the encoder signals. These are counted over a specified time determined by one of the timer modules. This allows for speed control on both wheels using only one timer module for encoder input and one for PWM generation and leaves two timers available for other purposes.

The encoder signal is fed to an external interrupt pin which is set up to trigger an interrupt on each rising edge. These edges are counted and a second timer (8-bit) is used to provide a speed control interval. After each interval, the number of interrupts is recorded and from this the current speed determined. The speed error is calculated and the control algorithm determines the required PWM duty cycle which is then written to the OCR. The direction of rotation of the wheels also has to be considered and this is specified by a global variable.

C.3 I^2C Communication

The Gumstix and the Roboaudiostix communicate with each other using I^2C. This communication is initiated by the Gumstix at an interval of approximately one second. At this time the Gumstix sends the required speed and direction for each wheel to the Roboaudiostix. The speed and direction is stored in a single 16-bit variable and so has to be interpretted correctly by the microcontroller. The direction is stored in the most significant bit of the variable and the speed in the 15 least significant bits. This method minimises communication time required. The Roboaudiostix also sends the data stored in the circular buffer back to the Gumstix to be written to file.

C.4 Ultrasound

The operation of the ultrasonic sensors is controlled by two functions, a chirp function and an external interrupt function triggered by a received ultrasound signal.

The chirp function is called by the main program and initiates the transmitter, waits for a specified chirp length and then switches off the transmitter. This is done using an 8-bit timer and an output pin. The transmitter circuit

is described in appendix E and requires an input from the microcontroller to switch on and off. When the chirp function is called the timer is reset and the transmitter switched on. The timer is then polled until the specified chirp length has been reached and then the transmitter is disabled. The sonar reading variable is also reset in this function.

When the sonar received interrupt is triggered, the value of the timer is recorded and stored in this variable. Due to noise in the receiver circuitry, sometimes a number of interrupts are triggered for each received signal. These would give false readings so have to be rejected. This is done by counting the number of receiver interrupts that have occurred since the chirp function was called. If this value is zero, the signal is valid and the sonar reading is recorded. If not, the data is ignored.

The ultrasound measurement procedure is outlined in figure C.1.

C.5 Data storage

Because accurate timing is required for the localisation algorithm, all data capture has to happen at fixed intervals. A 16-bit timer is used to achieve this. The timer interval is set and when the timer overflow occurs, an interrupt is triggered. Inside this interrupt routine, a flag is set. The main program checks this flag and when it is set, the data capture and storage routines are called. At each time-step the required data is stored in a circular buffer by the Roboaudiostix. This data consists of an index, the right wheel direction, the right wheel speed, the left wheel direction, the left wheel speed and the sonar reading. Because the chirp function is called at the beginning of each time-step to obtain an ultrasound reading, the program has to pause for enough time for the reflected signal to reach the receiver. This is done with a software loop. If a signal hasn't been received by the end of this pause, no ultrasound reading is recorded. After a number of time-steps this data is requested by the Gumstix and transferred using I^2C. The Gumstix then writes the data to a text file to be processed at a later stage.

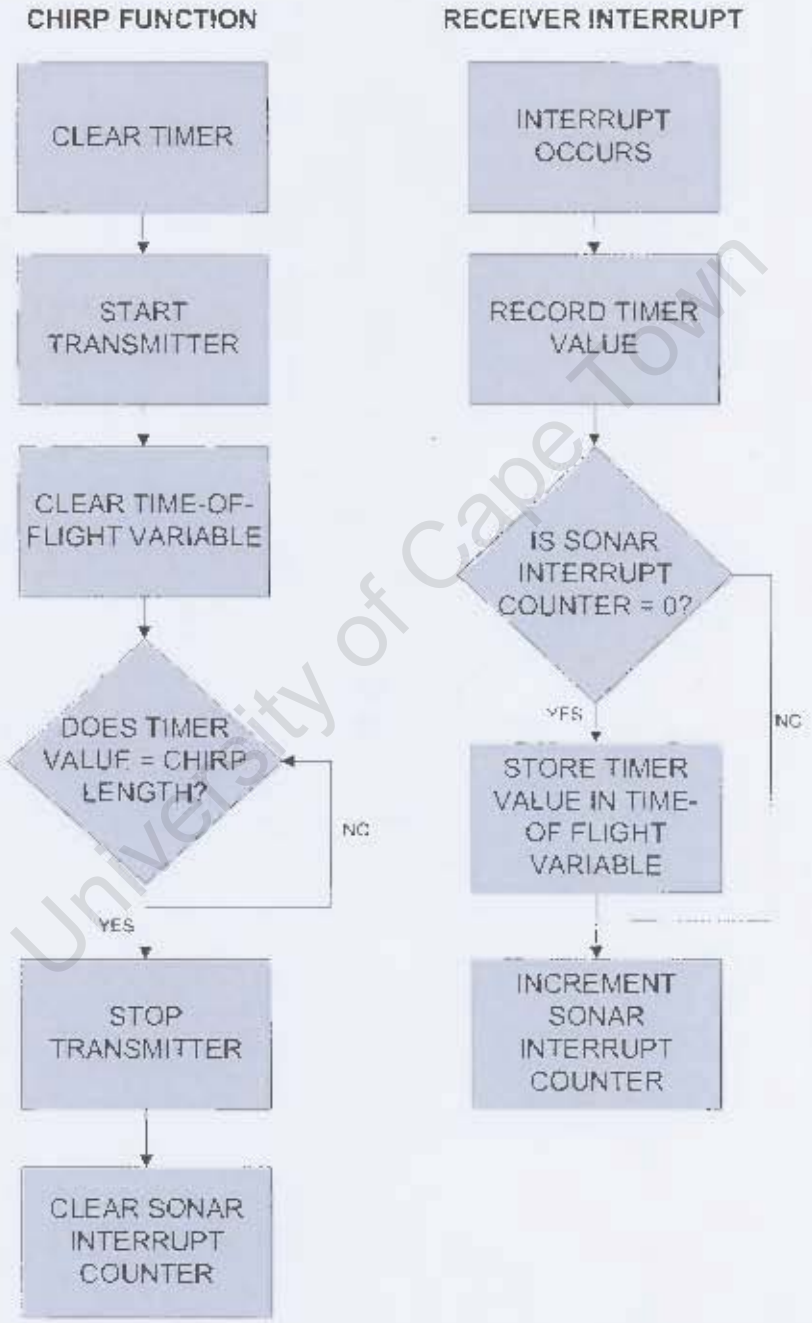


Figure C.1: Flow Diagram Showing Ultrasound Measurement

Appendix D

Speed Control

Contents

Figures

D.1 Introduction

Accurate speed control was required to allow the robot to move in a predictable manner and perform odometry. Odometry is the process of estimating position from speed and direction information. This was required for localisation and mapping. The actual speed of the wheels had to be measured by sensors so the motor driver could compensate for changes in load or supply voltage. This formed a closed-loop controller.

D.2 Drive System

The drive system consisted of a motor directly coupled to each wheel driven by a LMD18200 H-bridge chip [18]. This in turn received a signal from the Roboaudiostix microcontroller through an opto-isolation board to protect the controller from noise generated by the motors. Wheel speeds were measured by H21A1 phototransistor optical interrupter switches [20] in conjunction with slotted disk encoders attached to each wheel.

Figure D.1 shows the break-down of the drive system.

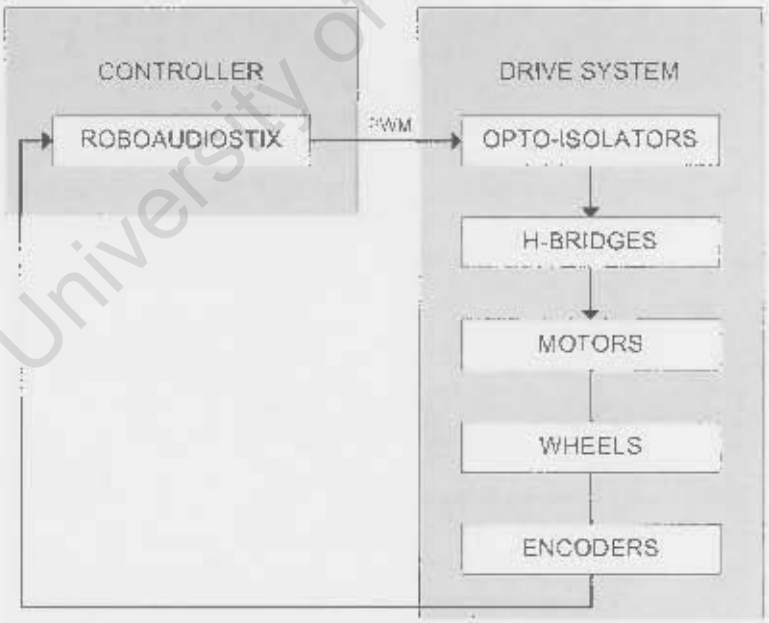


Figure D.1: Interaction of Drive System Components

D.3 Speed Measurement

The encoder circuit used for speed measurement is shown in figure D.2. The output of the encoder stage is a sinusoidal signal with a frequency proportional to the rotational speed of the wheel. This signal is then fed to the comparator stage which converts it to a square wave for input into the microcontroller interrupt pin. The microcontroller determines the wheel speed by counting the number of pulses per timing cycle. The details of this are outlined in section C.2 in appendix C.

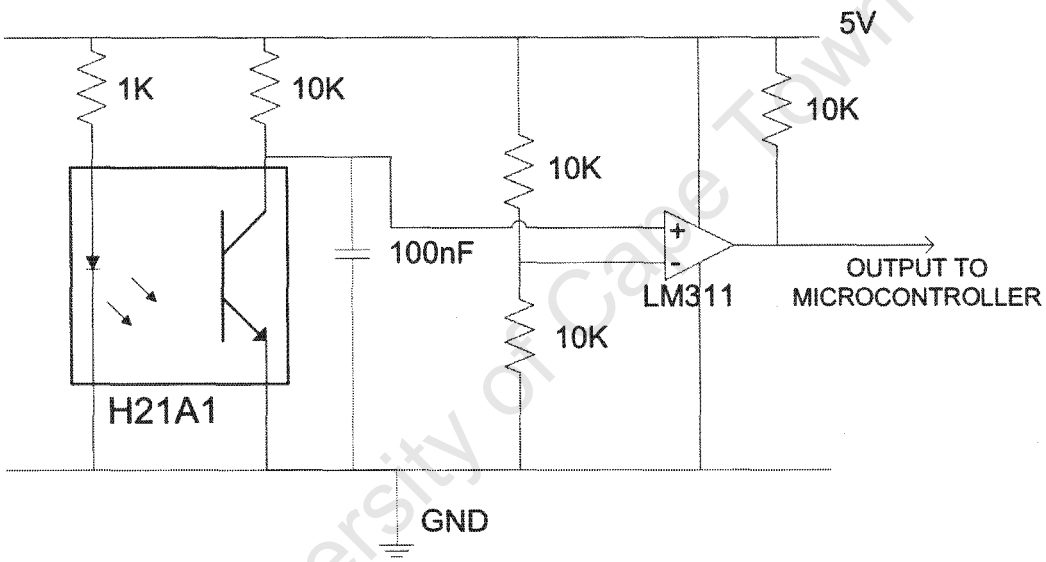


Figure D.2: Speed Measurement Circuit

D.4 Proportional Control

In this system, a proportional controller was implemented. The equation for proportional control is

$$R(t) = k(v_{des}(t) - v_{act}(t)) \tag{D.1}$$

where $R(t)$ is the motor output function, $v_{des}(t)$ is the desired motor speed, $v_{act}(t)$ is the measured motor speed and k is the constant controller gain [2][P.57, Chapter 4]. This procedure is illustrated in figure D.3.

By altering k, the behaviour of the system can be manipulated to obtain the required response. Increasing k increases the speed of the system but

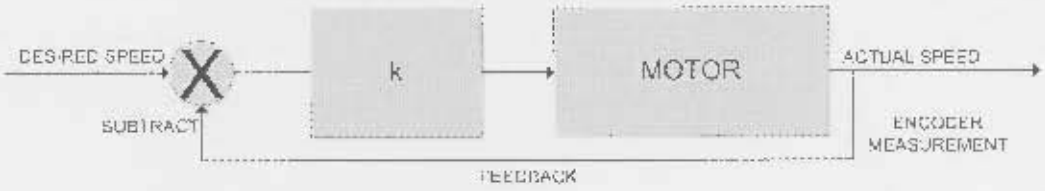


Figure D.3: Proportional Controller [2]

if k is too high, the system overshoots and may become oscillatory [2 [P.56, Chapter 4].

D.5 Implementation

An 8-bit timer module was used to set the time between control loop iterations. Of course for effective speed control, the direction of motion of the motors had to be considered. This however could not be measured by the encoders. It was instead defined by a variable in the microcontroller code and the control algorithm used this to select the appropriate output to the motors. The implementation of the algorithm on the microcontroller is discussed in section C.2 in appendix C.

D.6 Tuning

To optimise the speed control algorithm, the correct value for the controller gain, k had to be determined. A number of step-tests were performed where the robot was sent a required speed value and the wheel speeds measured from starting at zero until the setpoint was reached. The value of k affects how the wheels approach the setpoint.

If the k value is low, the system response will be slow and the system will take a long time to reach the setpoint. This system is overdamped but may be suitable for applications where it is essential that no overshoot occurs.

At intermediate k values, the system response is faster but the system may overshoot before reaching the setpoint.

If the value of k is high, the system overshoots and then oscillates about the setpoint. The response of the system is faster but this increased speed causes the overshoot which may not be desirable.

There is always a trade-off between system response and oscillation, and the optimal value for k depends on the particular system requirements.

D.6.1 Step-tests

The step-tests were performed for varying values of k. Each test was performed a number of times and averages taken to minimise the effect of environmental influences which were not constant over different tests. The robot was driven from a stationary position up to the set speed while the speed on one wheel was measured and stored. This data was analysed and plotted to determine the effect of the k value on the system.

Floating Point Calculations

The fact that k could take on a fractional value, added a level of complexity to the software. Because the control loop runs on an 8-bit microcontroller without a floating point calculation unit [21], special care had to be taken with these calculations. Fixed point arithmetic was used by multiplying the k value by a constant, being careful not to exceed the maximum possible size of a variable, performing the required control calculation and then dividing by the constant.

Resolution

The accuracy of the speed measurements was limited by the resolution of the speed encoders. Even at constant speed, the measurement could vary slightly. This varying speed value at constant speed should not be confused with oscillation of the system caused by the control algorithm. This means that even with an ideal controller and the system remaining at a constant speed, the graph of the wheel speed would show minor oscillation about the setpoint.

Results

The step-test results are shown below.

For a low value of k, $k = 0.5$, the system response is extremely slow. As can be seen in figure D.4, there is a significant amount of time taken initially to reach the setpoint. There is however no overshoot and although it at

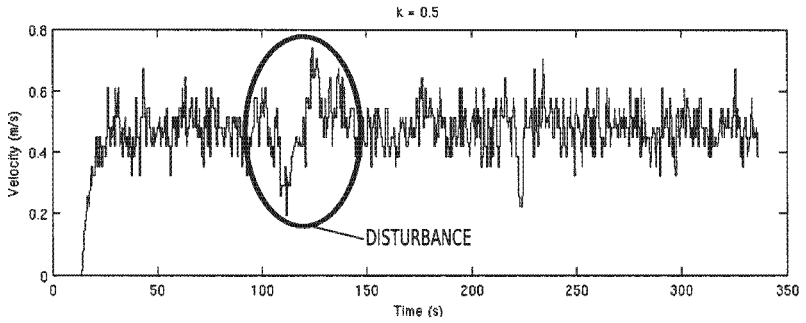


Figure D.4: Step-test with k = 0.5

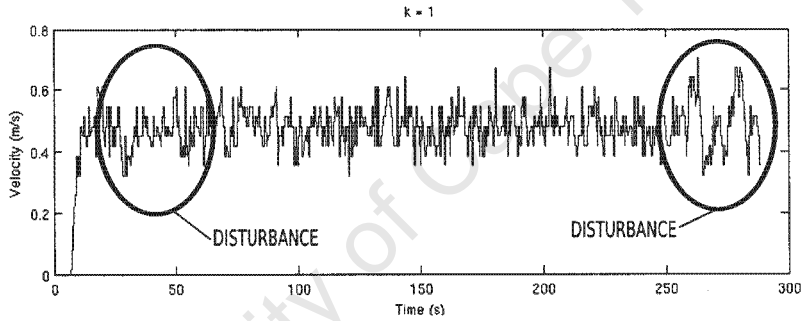


Figure D.5: Step-test with k = 1

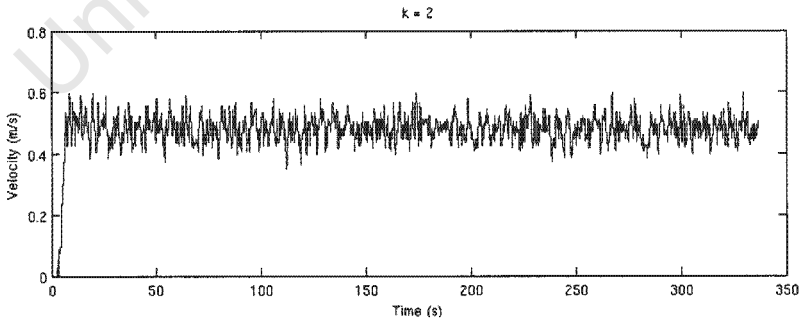


Figure D.6: Step-test with k = 2

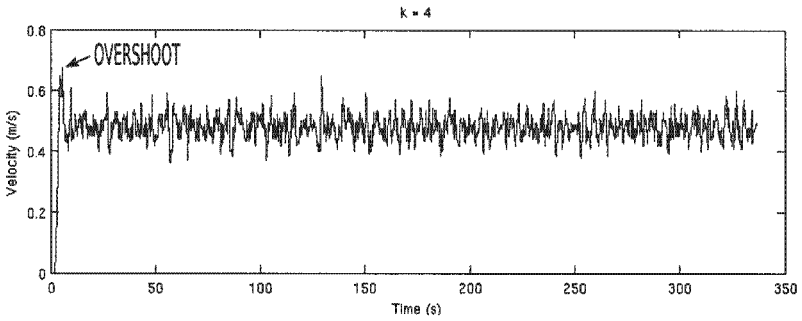


Figure D.7: Step-test with k = 4

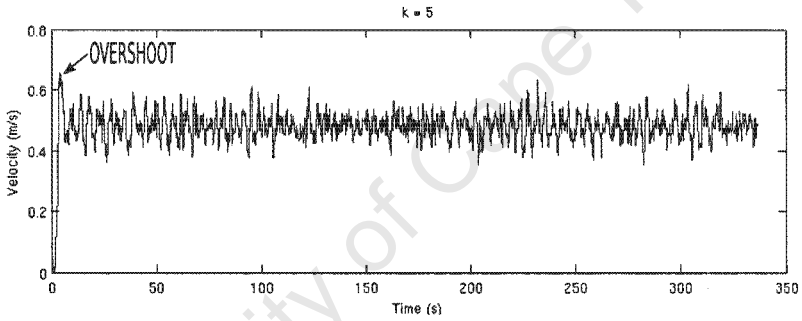


Figure D.8: Step-test with k = 5

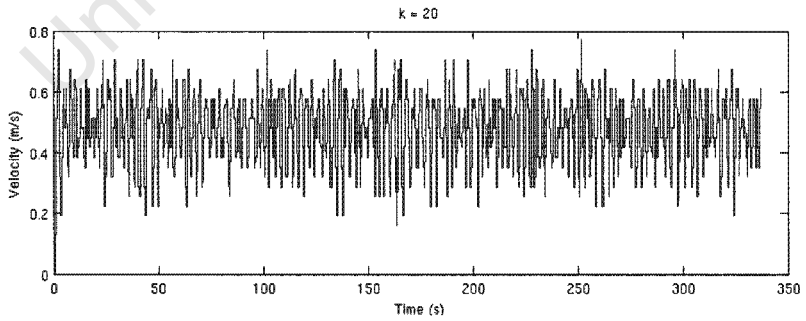


Figure D.9: Step-test with k = 20

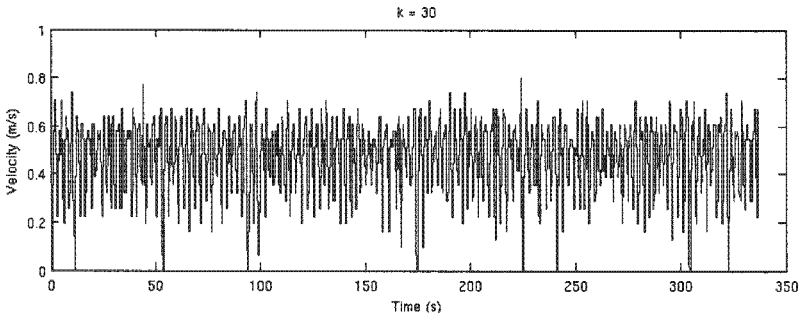


Figure D.10: Step-test with k = 30

first appears as if the system is oscillatory, this is not due to overcorrection by the control algorithm. On the contrary, the slow response of the system is the cause. This can be explained by the following example. The circled region shows a disturbance caused by the environment, for example a slight incline, causing the speed to drop well below the setpoint. The system is very slow to respond and get the speed back up to the setpoint. By this stage, the environment has again changed, now a level area or slight decline, and the system is unable to respond fast enough and so passes the setpoint. This behaviour can be seen in a number of places on the graph and is an indication that k is too low for this system.

For $k = 1$ (figure D.5), it can be seen that the system response is faster but still does not overshoot the setpoint. The system does take a fair amount of time to respond to disturbances especially when the speed is below the setpoint. This is demonstrated by the circled area.

Figure D.6 shows the result for $k = 2$. This response was faster as expected but still showed no overshoot.

For higher values of k, $k = 4$ (figure D.7) and $k = 5$ (figure D.8), the system response is greatly increased. There is however overshoot initially and some degree of oscillation although this is minor compared to the inherent resolution limitations of the speed encoders.

With k increased as high as 20 (figure D.9) and 30 (figure D.10), it is clear that the system has become unstable and oscillatory. The system overshoots the setpoint and oscillates indefinitely, never settling around the required value. This is as expected and indicates that these values are too high.

D.6.2 Conclusions

As explained above, choosing an appropriate k value is a trade-off between speed of system response and stability and lack of overshoot. Therefore, the requirements for the particular system must be decided on before the optimal value can be selected.

For this system, extremely accurate speed control was not required as long as the actual speed at any given time was recorded for use by the localisation algorithm. A controller gain value of $k = 2$ was chosen because it provided an adequate speed of response while limiting overshoot and oscillation.

Proportional control was sufficient for the requirements of this system. Because extreme accuracy was not required, a more complex control algorithm such as PI or PID was deemed unnecessary. This could however easily be added if it is required at a later stage.

Appendix E

Ultrasound Sensor System

Contents

Figures

E.1 Transducer Selection

For the robot to create a map of and then navigate within its environment, an efficient sensing system was required. Detecting objects at a distance was an important requirement for this system. This would allow the robot to decide on the best course of action, taking into account a number of obstacles and also to build up a map of their locations.

A number of remote sensing methods were considered for this application. Infrared sensors could be used to detect distant objects. The difficulty with this method however is that if the ambient light varies within the environment, the sensor would have to compensate for this. This is possible but makes the system more complex.

Laser sensors are traditionally used for environment mapping. These are costly and as the ultimate goal of this project is the design of a low-cost robot, they are not suitable for this application.

Cameras could also be used to detect objects and build an environment map. This however requires large amounts of sophisticated programming and signal processing to interpret the received images.

Ultrasonic sensors were ideal for the task as they are relatively low-cost, have a suitable range and should not be affected by noise in the environment because they operate in a narrow bandwidth outside the audible range.

E.2 Overview

In order to provide observation data for the localisation algorithm, a time-of-flight ultrasound system was implemented. This system is illustrated in figure E.1.

These sensors were designed to detect the presence of an object and calculate its range. The transducers used in this system did not have the accuracy required to implement a bearing measurement system but a possible method is discussed in "Mobile Robot Sonar for Target Localization and Classification" [31]. The object was just assumed to be directly in front of the transducer although in reality it could lie anywhere within a 50°arc around this point. The results of the testing however, demonstrated that localisation was possible without an accurate bearing measurement.

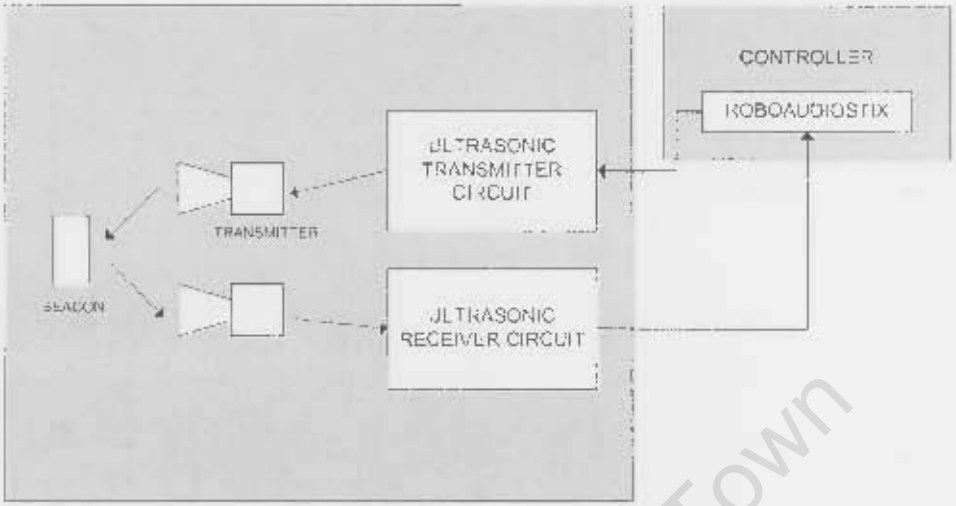


Figure E.1: Ultrasonic Sensor System

The range to an object was determined by the time taken for the ultrasonic transducer to receive a reflected pulse. The distance was calculated according to the equation

$$d = \frac{C \times t}{2} \tag{E.1}$$

where d is the distance to the object, C is the speed of sound and t is the time between transmission and received pulse [32][P. 95].

A timer on the microcontroller was started when the transmission signal began and was read when a return signal was received to determine the elapsed time.

E.3 Hardware Setup

Murata 40kHz ultrasonic transducers [19] were used in this system. The main specifications of these transducers are shown in table E.1.

Figure E.2 shows an overview of the operation of the time-of-flight ultrasound circuitry.

When the chirp routine was called, the Roboaudiostix activated the oscillator for a given chirp length. When a return signal was received, it was amplified by a two-stage amplifier with a total gain of 100 determined experimentally.

Table E.1: Ultrasonic Transducer Main Specifications

Centre frequency	40kHz
Resolution	9mm
Directivity	50°

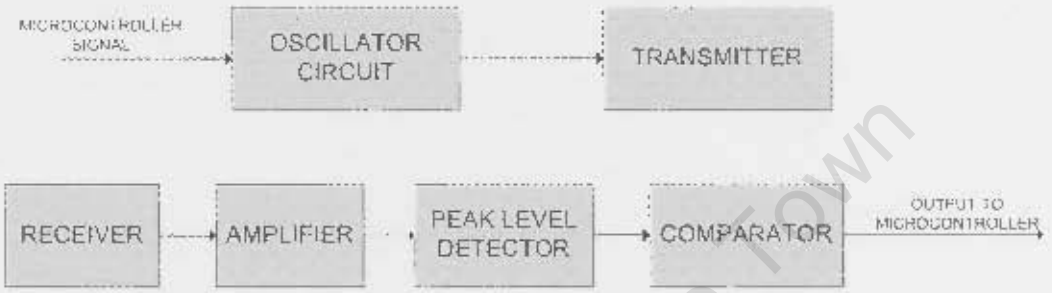


Figure E.2: Ultrasound Circuitry Block Diagram

This gained signal was then fed to a peak-level detector which converted it to a DC voltage. The DC voltage was then fed to a comparator. The comparator compared this voltage to a reference voltage and the output was fed to a Roboaudiostix external interrupt pin.

The operation of the individual system components is explained in detail below.

E.3.1 Oscillator

The transducer required a 40kHz signal. This was generated by a CD4047 astable multivibrator chip [33]. The frequency could be adjusted with a potentiometer to ensure the correct operation of the transducers. The length of the transmitted pulse was controlled by the microprocessor which sent a signal to the reset pin on the CD4047.

The oscillator circuit is shown in figure E.3.

As can be seen in the figure, the transducer was connected between Q and $\bar{Q}$. As these signals were in antiphase, this effectively doubled the supply voltage, increasing the range of the transducer.

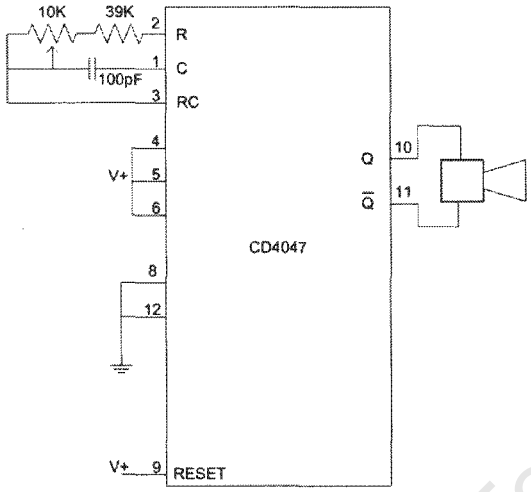


Figure E.3: Oscillator Circuit

E.3.2 Receiver Amplifier

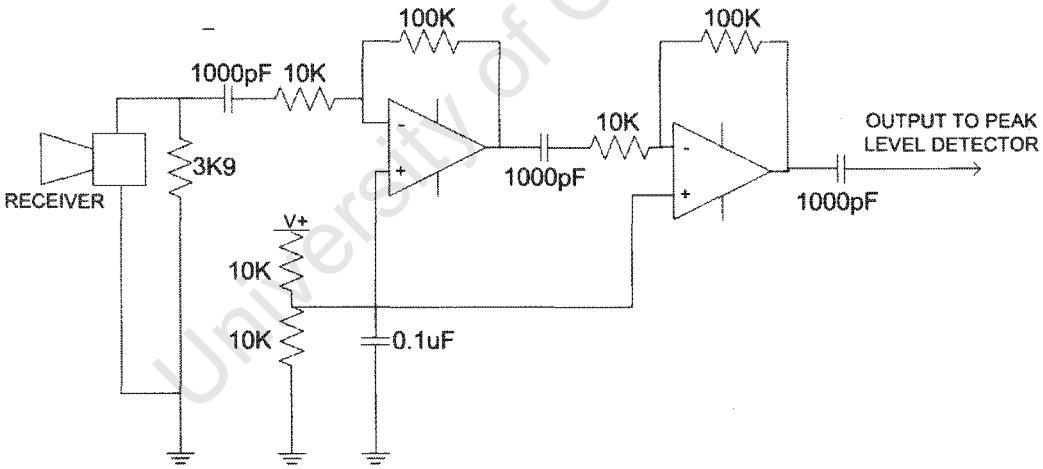


Figure E.4: Receiver Amplifier Circuit

Figure E.4 shows the receiver amplifier circuit. As the frequency was fairly high, high slew-rate opamps were required. The circuit had a gain of 100 and the output was a 40kHz sine wave.

E.3.3 Peak-level Detector

In order to determine if a target was present, it was necessary to compare the received signal to a reference. For this reason, a DC voltage was required. A peak-level detector was used to convert the received AC signal to a DC signal.

The frequency response was set so that the 40kHz peaks were not followed and the signal was effectively smoothed. The frequency was chosen to provide adequate smoothing with a sufficiently fast response time.

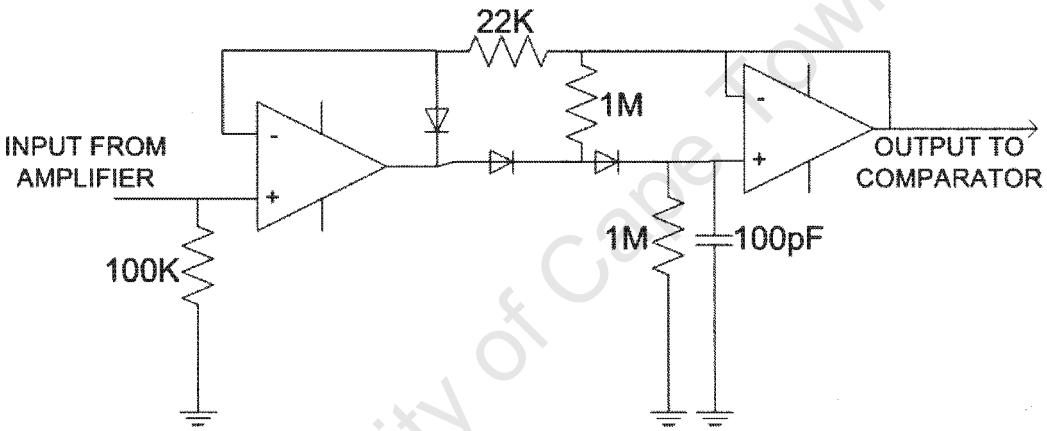


Figure E.5: Peak Level Detector Circuit

Figure E.5 shows the circuit diagram. High slew-rate opamps were again required.

E.3.4 Comparator

The comparator compared the output from the peak-level detector to a reference voltage. This reference voltage needed to be low enough to trigger as fast as possible after the signal was received to avoid timing errors, yet high enough to prevent false returns due to noise. This value was determined experimentally. During initial tests, a number of false triggers occurred. This was found to be due to the shallow gradient near the reference voltage level as the signal decayed. This error was rectified by adding hysteresis to the circuit as shown in figure E.6.

E.3.3 Peak-level Detector

In order to determine if a target was present, it was necessary to compare the received signal to a reference. For this reason, a DC voltage was required. A peak-level detector was used to convert the received AC signal to a DC signal.

The frequency response was set so that the 40kHz peaks were not followed and the signal was effectively smoothed. The frequency was chosen to provide adequate smoothing with a sufficiently fast response time.

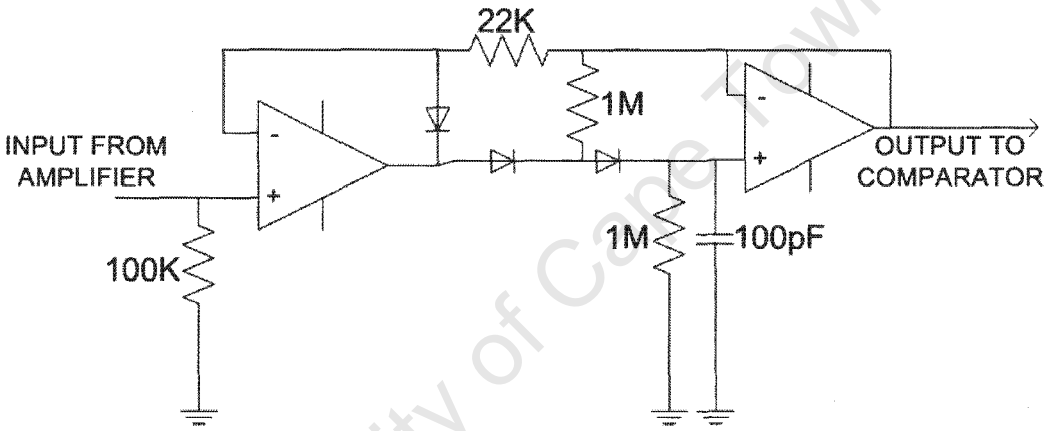


Figure E.5: Peak Level Detector Circuit

Figure E.5 shows the circuit diagram. High slew-rate opamps were again required.

E.3.4 Comparator

The comparator compared the output from the peak-level detector to a reference voltage. This reference voltage needed to be low enough to trigger as fast as possible after the signal was received to avoid timing errors, yet high enough to prevent false returns due to noise. This value was determined experimentally. During initial tests, a number of false triggers occurred. This was found to be due to the shallow gradient near the reference voltage level as the signal decayed. This error was rectified by adding hysteresis to the circuit as shown in figure E.6.

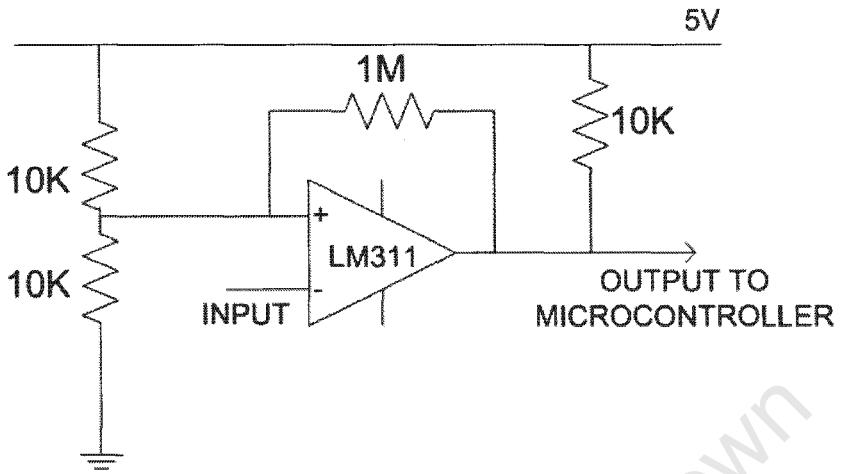


Figure E.6: Comparator with Hysteresis

E.3.5 Roboaudiostix Microcontroller

The Roboaudiostix had two functions in this system. On the transmitter side, it controlled the length of the chirp. The chirp function was called by the main routine at the beginning of each measurement cycle. The Roboaudiostix reset a timer and set the oscillator reset pin low. When the chirp time had elapsed, the oscillator reset pin was set high to end the chirp.

If a signal was received, the comparator output went low. This negative edge triggered an external interrupt on the Roboaudiostix. The timer was then read and the time of flight recorded. The details of this software are discussed in section C.4 in appendix C.

Appendix F

Vehicle Model

Contents

Figures

F.1 Introduction

In order to use a Kalman filter for localisation and mapping, a process model for the robot is required. This must be able to predict the future state of the robot, given the current state and control inputs. This prediction is then combined with sensor information to obtain an estimate for the actual robot state.

F.2 Model Derivation

The state of the robot is defined by its location and orientation. This is described by a state vector, X, $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$

where x and y are the co-ordinates of the centroid and θ is the angle of the line joining the left and right wheels with respect to the x-axis. The centroid in this case is defined as the centre point of a straight line joining the left and right wheels. This vehicle model is illustrated in figure F.1.

X	State Vector
x	x Position of Centroid
y	y Position of Centroid
θ	Orientation Angle
ΔT	Length of Timestep
R	Rotation Matrix
W_L	Left Wheel Position Vector
W_R	Right Wheel Position Vector
ω_L	Left Wheel Rotational Velocity
ω_R	Right Wheel Rotation Velocity
B	Distance between Left and Right Wheels
r	Wheel Radius

Table F.1: List of Symbols

The requirement is to find an equation for the centroid and orientation angle at the current time in terms of the previous centroid and orientation angle and the current control input, the rotational velocity of the left and right wheels.

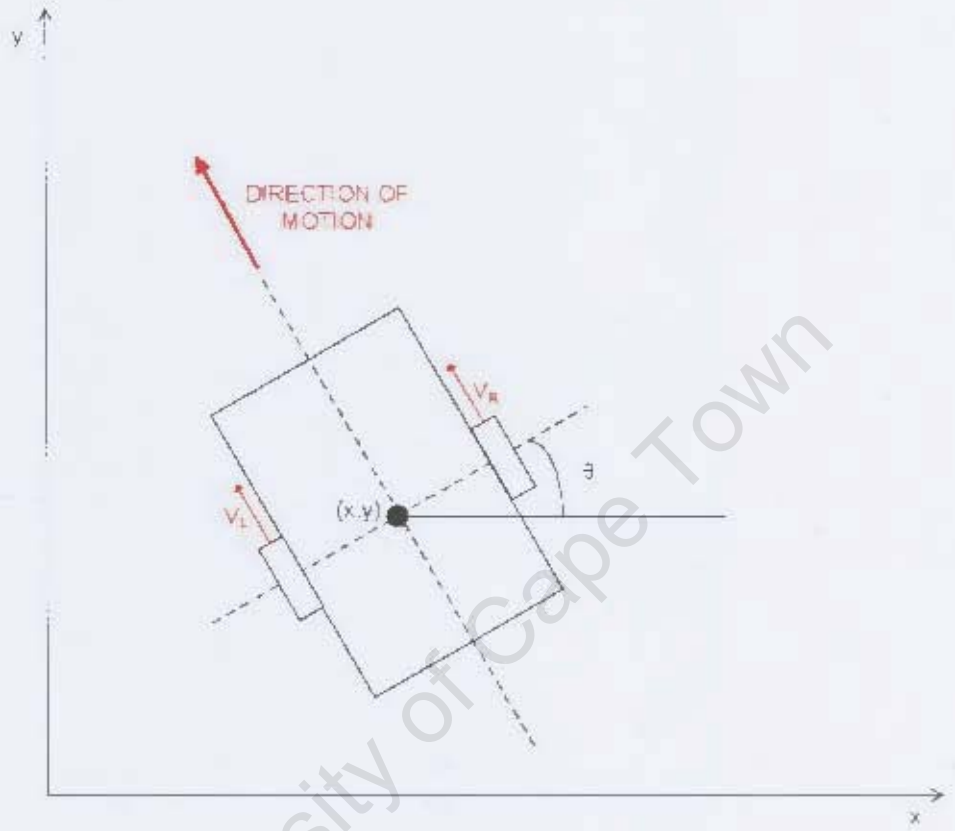


Figure F.1: Vehicle Model

To do this the vehicle motion is broken down into very small time steps and the change in position and orientation over that timestep is determined. It is assumed that if ΔT is sufficiently small, the left and right wheels can be approximated to move independently, in a straight line with the current orientation. After the change in positions of the two wheels, the new centroid and orientation are calculated from the vector joining the two positions.

F.2.1 Assumptions

- Zero wheel slip
- For small angles $\arctan(x) = x$
- Independent movement of left and right wheels over small timesteps

F.2.2 Derivation

The wheel positions are determined from the current centroid and orientation. Initially the robot centroid is placed at the origin with $\theta = 0$. The positions of the wheels are therefore,

$$W_L = \begin{bmatrix} -0.5B \\ 0 \end{bmatrix}$$

$$W_R = \begin{bmatrix} 0.5B \\ 0 \end{bmatrix}$$

A rotation matrix based on the current orientation is calculated.

$$R = \begin{bmatrix} \cos\theta(k-1) & -\sin\theta(k-1) \\ \sin\theta(k-1) & \cos\theta(k-1) \end{bmatrix}$$

The wheel positions are then rotated by the current θ and the centroid vector added to determine the current wheel positions. This process is illustrated in figure F.2.

$$W_L = R \begin{bmatrix} -0.5B \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}_{\mathrm{C(k-1)}} \tag{F.1}$$

$$W_R = R \begin{bmatrix} 0.5B \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}_{\mathrm{C(k-1)}} \tag{F.2}$$

From the wheel positions at $k-1$, the wheels move independently, with velocity, $V = \omega r$, the control input for the current timestep. The velocity is multiplied by the timestep to get the distance moved. This is transformed into the direction of vehicle motion and added to the wheel position.

$$W_L(k) = R \begin{bmatrix} -0.5B \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}_{\mathrm{C(k-1)}} + V_L \Delta T \begin{bmatrix} -\sin\theta(k-1) \\ \cos\theta(k-1) \end{bmatrix} \tag{F.3}$$

$$W_R(k) = R \begin{bmatrix} 0.5B \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}_{\mathrm{C(k-1)}} + V_R \Delta T \begin{bmatrix} -\sin\theta(k-1) \\ \cos\theta(k-1) \end{bmatrix} \tag{F.4}$$

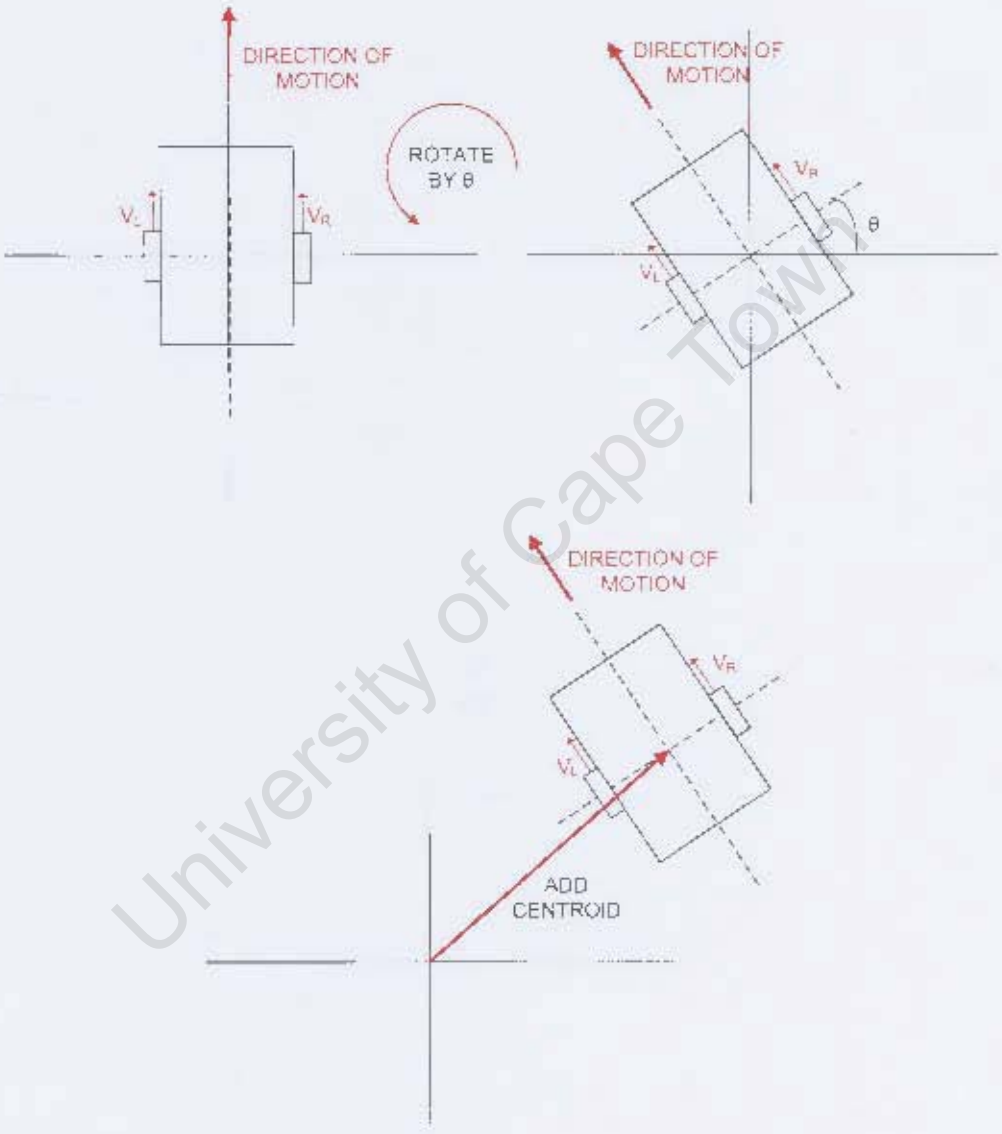


Figure F.2: Obtaining the Robot Position

From these new wheel positions, the centroid at time k is calculated.

$$\begin{aligned}\begin{bmatrix}x\\y\end{bmatrix}_{\text{C(k)}} &= \frac{W_L + W_R}{2}\\ &= 0.5\left(R\left(\begin{bmatrix}-0.5B\\0\end{bmatrix}+\begin{bmatrix}0.5B\\0\end{bmatrix}\right)+2\begin{bmatrix}x\\y\end{bmatrix}_{\text{C(k-1)}}\right.\\ &\quad\left.+\begin{bmatrix}-\sin\theta(k-1)\\ \cos\theta(k-1)\end{bmatrix}\Delta T(V_L+V_R)\right)\\ &= \begin{bmatrix}x(k-1)-0.5(V_L+V_R)\sin\theta(k-1)\Delta T\\ y(k-1)+0.5(V_L+V_R)\cos\theta(k-1)\Delta T\end{bmatrix}\end{aligned} \tag{F.5}$$

$\theta(k)$ is determined from

$$\theta(k) = \theta(k-1) + \Delta\theta \tag{F.6}$$

where $\Delta\theta$ is the change in angle.

$$\theta(k) = \theta(k-1) + \arctan((\frac{V_R - V_L}{B})\Delta T) \tag{F.7}$$

For small x, $\arctan(x) \approx x$, so with ΔT sufficiently small,

$$\therefore \theta(k) = \theta(k-1) + (\frac{V_R - V_L}{B})\Delta T \tag{F.8}$$

The final process model for the robot from state $X(k-1)$ to $X(k)$ given information up to $k-1$ and under control inputs, V_L and V_R is

$$\begin{bmatrix}\hat{x}(k|k-1)\\ \hat{y}(k|k-1)\\ \hat{\theta}(k|k-1)\end{bmatrix} = \begin{bmatrix}\hat{x}(k-1|k-1) - \frac{1}{2}\Delta T(V_L(k)+V_R(k))\sin\theta(k-1)\\ \hat{y}(k-1|k-1) + \frac{1}{2}\Delta T(V_L(k)+V_R(k))\cos\theta(k-1)\\ \hat{\theta}(k-1|k-1) + \frac{V_R - V_L}{B}\Delta T\end{bmatrix} \tag{F.9}$$

The extended Kalman filter also requires the calculation of the Jacobian of the process model. The Jacobian is a matrix of all first-order partial

derivatives of a vector-valued function [34]. It is used to linearise the function around the current state. This is discussed in section 2.2.2 on page 7.

In this case, the Jacobian at x(k) = $\hat{x}(k-1|k-1)$

$$\bigtriangledown f_x(k) = \begin{bmatrix} 1 & 0 & -\frac{1}{2}\Delta T(V_L(k)+V_R(k))\cos\theta(k-1) \\ 0 & 1 & -\frac{1}{2}\Delta T(V_L(k)+V_R(k))\sin\theta(k-1) \\ 0 & 0 & 1 \end{bmatrix} \tag{F.10}$$

where $f_x(k)$ is the process model described above.

Appendix G

Transducer Interference

Contents

Figures

G.1 Introduction

The final ultrasound system will consist of many ultrasonic transducers, supporting circuitry and an interface to a controller using analogue to digital converters. It was desirable to be able to test the performance of the transducers and the optimum setup of the system without all these components in place. These tests could be used to determine which parts of the system were suited to analogue processing and to what extent digital methods would be preferable. MATLAB was chosen to perform the signal processing and determine which digital methods would be valuable in the final system. The chosen methods could later be efficiently written into the controller program. The generation of the transmission signal was also performed in MATLAB and the onboard PC soundcard was used to input and output the data.

Because the maximum sample rate of the sound card was only 44100 samples per second, it was unable to produce or receive the required 40kHz of the transducer. The output signal was therefore produced at 10kHz and upmixed using an analogue multiplier to produce the required frequency. This process was also performed on the received signal to return to the 10kHz which was then read into the sound card.

Figure G.1 shows the system layout.

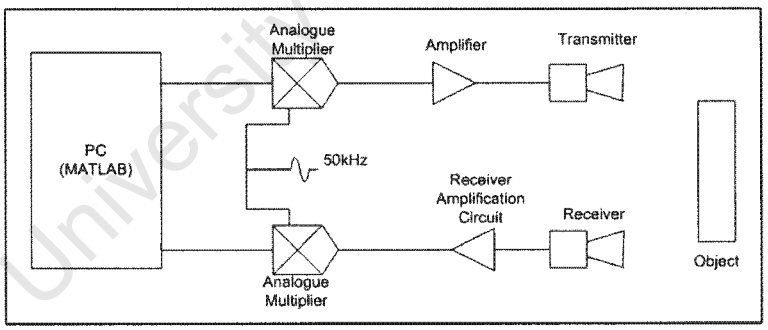


Figure G.1: Block Diagram of System Layout

G.2 Transmission Pulse Design

For time-of-flight sonar with monotonic transmitted pulses, the range resolution depends on the length of the transmitted pulse [35]. If two targets have a separation distance which is less than this length they cannot be

distinguished. Making the pulse shorter will improve this problem but will also reduce the maximum range of the sensor. Fortunately, the range resolution does not actually depend on the length of the pulse, but rather on the time-bandwidth product. For simple pulses, shortening the pulse increases the bandwidth leading to improved resolution. However, if the pulse can be designed in such a way that the bandwidth is high, the duration can be longer and therefore have good resolution and maximum range. Range information can be extracted from long, high-bandwidth pulses using the pulse compression technique [35],[36].

There are many methods for increasing the bandwidth of a signal, such as binary phase coding and frequency modulation. As it was desirable to have increased range resolution while maintaining an acceptable maximum range, three of these methods were investigated.

G.3 Frequency Modulation (Chirp)

A chirp signal was chosen as a starting point for the testing of the ultrasound system as they are commonly used in sonar and radar and because it is a simple way to utilize the bandwidth of the transducer. A chirp is a frequency modulated signal and can also be generated using analogue circuitry which was a possibility for the final system.

The chirp signal was generated in MATLAB. This signal had a linear frequency modulation, a centre frequency of 10kHz, a bandwidth of 4kHz and a duration of 1ms. The centre frequency and bandwidth are determined by the transducer itself while the duration can be varied.

Figure G.2 below shows the transmitted chirp signal.

G.3.1 Simulation of Return Signal

Before the testing of the system was conducted, a simulation was performed. In this simulation, the likely return signal was generated and this used to test the signal processing procedures and for later comparison with the actual received signal. This signal is shown for simulated targets at a distance of 0.5m and 1m in figure G.3 below. The signal which would be received directly from the transmitter is not included.

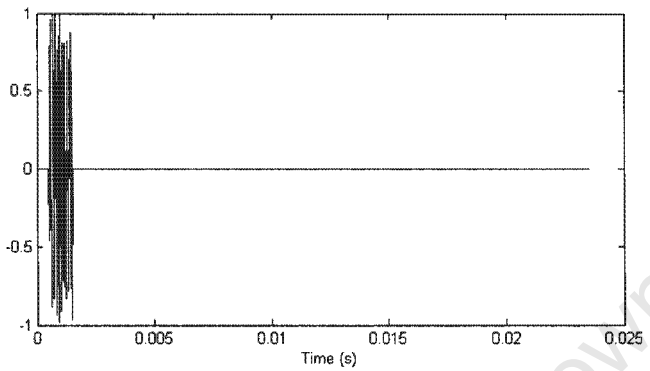


Figure G.2: Transmitted Chirp

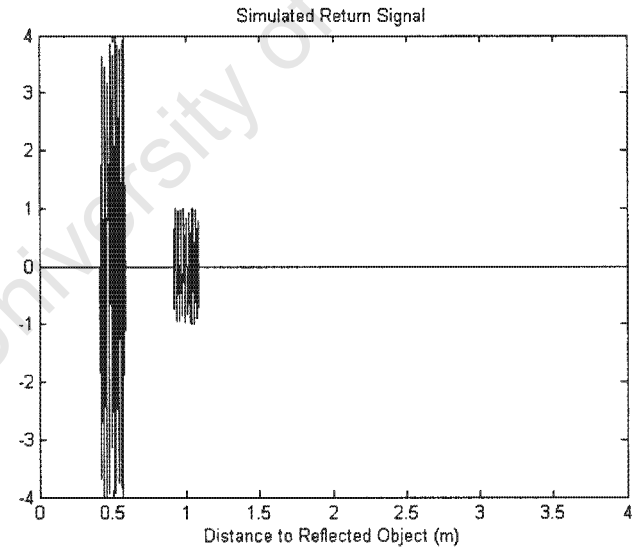


Figure G.3: Simulated Return Signal

G.3.2 Matched Filter

The simulation above shows an ideal return signal. This however is not likely in reality. There will be added noise and reflections from objects will not be so clearly defined. A technique for eliminating noise in received signals (i.e. maximizing the signal to noise ratio) is applying a matched filter. This is the process of searching a received signal for similarities to the transmitted signal by determining the correlation of the signals [35].

Figure G.4 below shows the return signal as above with the matched filter applied.

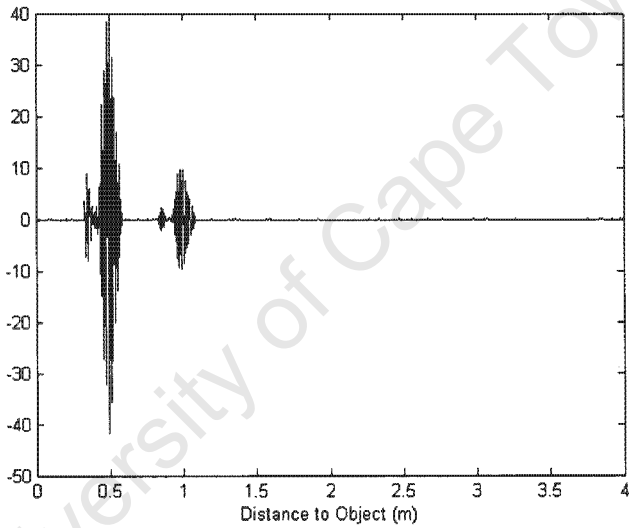
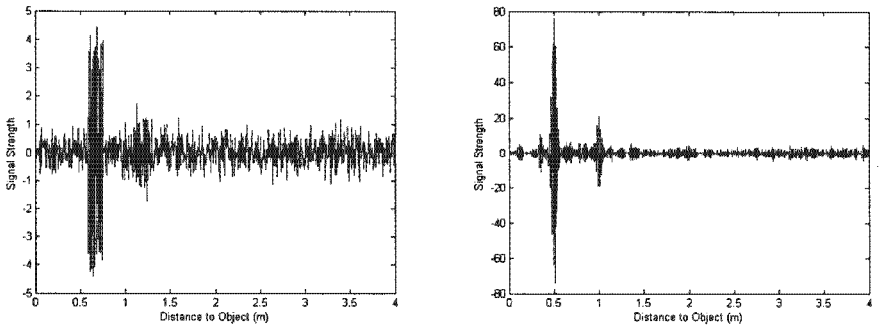


Figure G.4: Simulated Return Signal with Matched Filter Applied

To demonstrate the effectiveness of the matched filter, Gaussian noise was added to the simulated return signal. The filter was then applied to this and the result plotted. The target information (obscured with added noise) is clearly shown once the filter has been applied.

The figures below show how applying a matched filter to a received signal with added Gaussian noise can increase the signal to noise ratio. Figure G.5(a) shows the simulated return above with Gaussian noise added and figure G.5(b) shows the signal once the matched filter has been applied. The original signal can clearly be seen and the reflections detected.



(a) Return Signal with Noise Added

(b) Noisy Signal with Matched Filter Applied

Figure G.5: Noise Rejection of Matched Filter

G.3.3 Testing

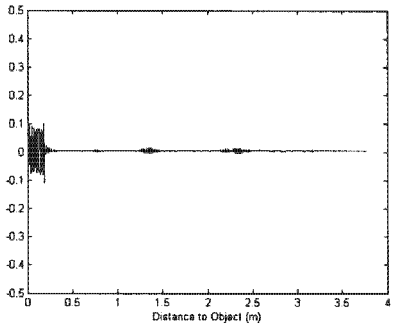
To test the actual performance of the system, the chirp pulse was transmitted and then the return signal fed into the soundcard as described above. This was repeated with objects at varying distances from the receiver.

G.3.4 Results

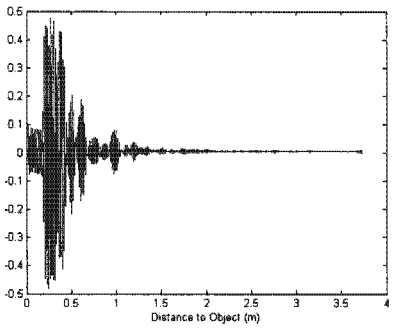
Figure G.6 shows the results of these tests before any signal processing was performed. The initial transmitted chirp can be seen in all the signals and was used to compensate for any delays caused by the soundcard and MATLAB allowing the system to be calibrated. In figure G.6(b), the initial chirp and the reflected signal overlap. This is expected as with the chosen chirp length, the minimum range was 170mm to avoid overlap.

In the graphs above, it is clear where the return signals occur. In practice, the signals may have higher levels of noise which obscure this. It was necessary to determine whether applying a matched filter to an actual received signal would improve the signal to noise ratio as demonstrated in the simulation.

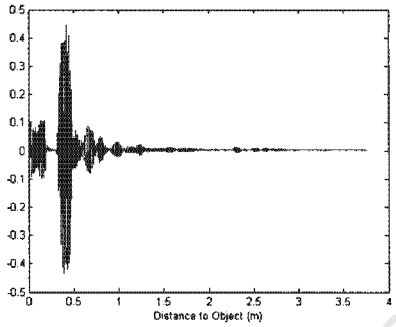
Figure G.7 shows the performance of a matched filter on a noisy return signal. The return signal from an object at 500mm can be easily detected once the filter has been applied.



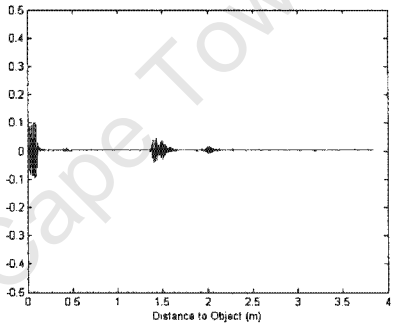
(a) No Reflecting Objects



(b) Object at 150mm

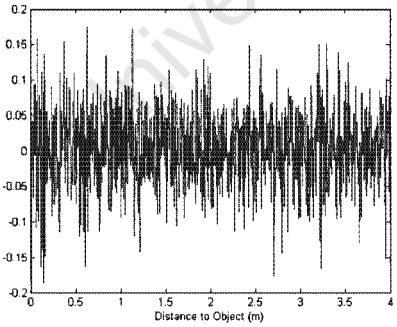


(c) Object at 300mm

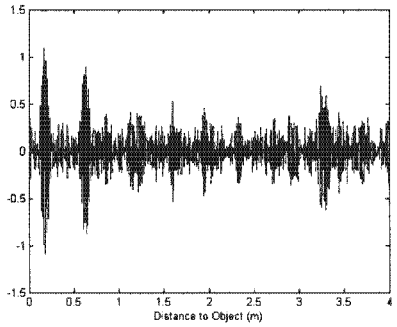


(d) Object at 1.4m

Figure G.6: Return Signals from Objects at Various Distances



(a) Received Signal with Noise



(b) Signal with Matched Filter Applied

Figure G.7: Application of a Matched Filter to Actual Received Signal with Added Noise

G.4 Eliminating Crosstalk

The methods discussed above are successful at eliminating white Gaussian noise [37] present in the received signal. This is because the additive white Gaussian noise is uncorrelated with the received signal. In this system however, it would be advantageous to be able to filter out noise from other sonar transducers transmitting at the same time. These could be from other transducers on the robot itself or from other robots operating in the same area. This problem of similar transducer's interfering with each other is known as crosstalk. The most common methods of preventing this interference are time division and frequency division but these were not suitable for this application.

Time division is simply ensuring that each transducer operates in turn, not transmitting while another is operating. This method has two disadvantages, the first being the considerable delay while allowing other transducers to operate. Another consideration is if there were multiple robots operating in an area, their sensors would have to be co-ordinated which would be extremely problematic due to the added system complexity and communication required.

Frequency division is operating the transducers at different frequencies. These must have enough separation to avoid interference. This therefore, requires high-bandwidth transducers which are costly and can require more computing power.

It would be ideal if the transducers could operate at the same time and in the same frequency band while being able to eliminate crosstalk. This is possible if each transmitter puts out a unique signal which can be extracted from the noisy return. The process of searching for similarities to a known signal in another signal is cross-correlation. The cross-correlation is high if the signals are similar and low if not. Autocorrelation is the cross-correlation of a signal with itself. As long as the various transmitted signals have a low cross-correlation with all the other signals and a high auto-correlation, the individual returns will be discernible. This technique is widely used in communications, allowing many signals to share the same transmission line without interfering with each other [35],[38].

Various methods of generating unique signals were considered. These are discussed below.

G.4.1 Phase Modulation using Walsh Codes

Phase modulation is one method of encoding the transmitted signal. The phase in each signal is modulated in such a way that any two signals do not correlate well with one another. The phase modulation is determined by a unique code which is assigned to each transmitter. If the codes used are orthogonal (their inner-product is zero), the signal will have an extremely low correlation with any other transmitted signals and any false returns will look like noise [35].

Walsh codes [39] are sets of orthogonal mathematical codes used in telecommunications to identify particular communication channels [38]. Although in direct sequence spread spectrum (DSSS) telecommunication, the signal being sent is digital, the codes can also be used to encode an analogue signal by modulating the phase of the signal depending on the code digit. This was done as follows:

$$Transmit = \cos(2\pi f0t + \pi/2 \times Wc)$$

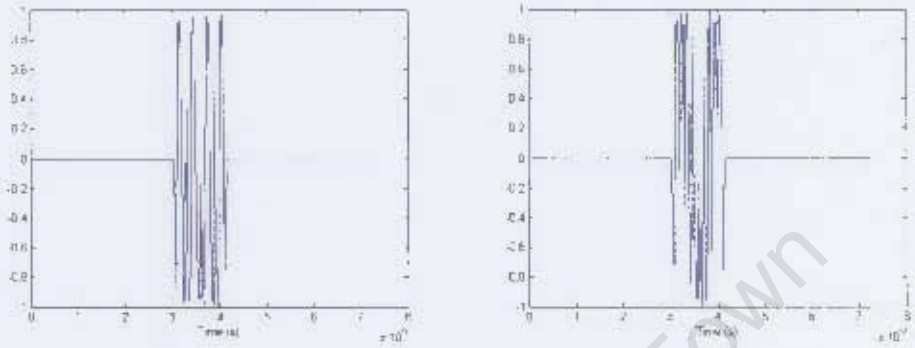
where Wc is the relevent Walsh code, f0 is the centre frequency and t is the timestep.

Each digit in the Walsh code is either positive or negative one, so this equation gives a maximum phase shift for each change in the code digit.

Testing

16-bit Walsh codes were chosen and the transmission signals encoded using two different codes by the method described above. These signals are shown in figure G.8 below where the phase modulation caused by the encoding can be seen.

Figure G.9 shows the auto- and cross-correlation of two transmitter signals encoded with Walsh codes. The blue shows the auto-correlation which has a clearly defined spike indicating that the signals correlate well. The red shows the cross-correlation with another Walsh coded signal. As expected, these have a low cross-correlation with the maximum being approximately five times lower than the auto-correlation maximum. This should allow the two signals to be easily distinguishable.



(a) Signal 1 (b) Signal 2

Figure G.8: Transmission Signals Encoded with Orthogonal Walsh Codes

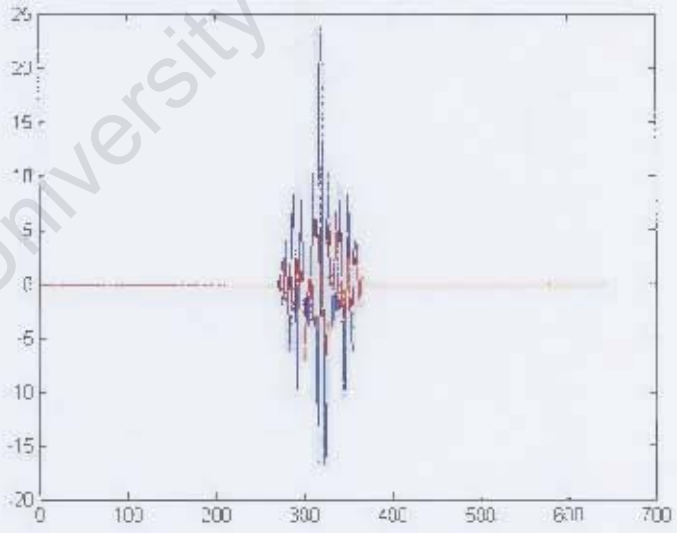


Figure G.9: Auto- and Cross-correlation of Phase Modulated Signals using Walsh Codes

Results

Figure G.10 shows the a return signal which has been correlated with the transmitted signal and with another incorrect signal. This correlation procedure was successful as the blue shows the correct signal while the red which is the incorrect signal has lower energy and looks like noise. The object reflection can be clearly identified at 300mm.

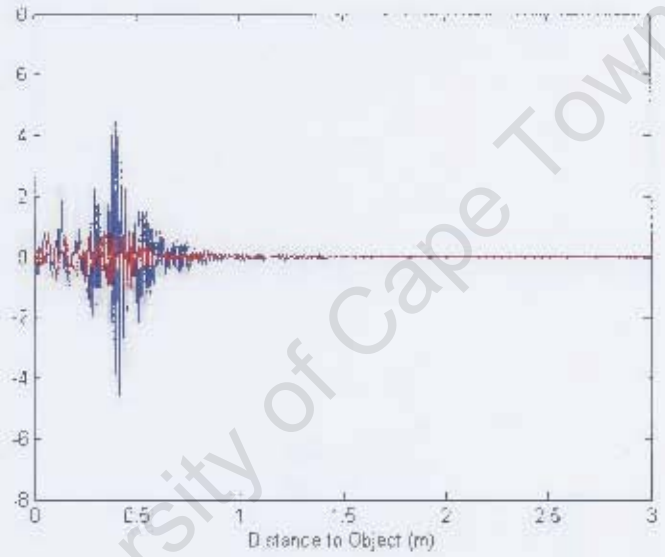


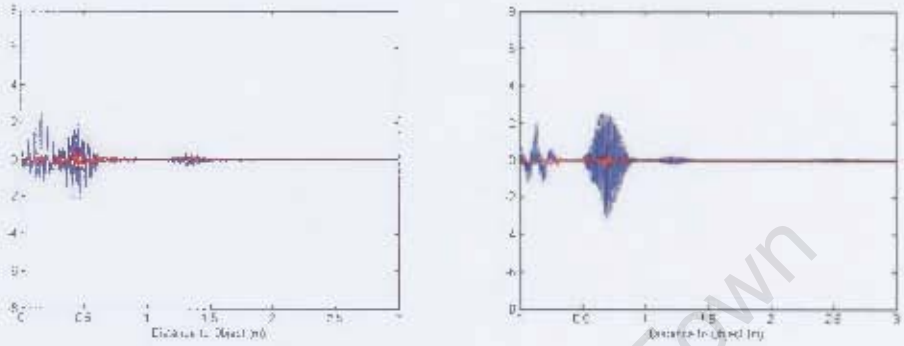
Figure G.10: Auto- and Cross-correlation of Return Signal

To test whether this method would work with interfering signals, returns from two different transmitted signals were added together. The correlation operations were then performed on the combined signal in an attempt to isolate the individual returns.

Figure G.11 shows the two different signals used to create the combined signal. The correlation with the correct signal is shown in blue while the incorrect one is shown in red. The distances at which the objects were positioned can be clearly seen.

Figure G.12 shows the combined signal. The targets are not distinguishable due to interference from conflicting transmitters. This signal was correlated with each transmitted signal to obtain the following result.

These figures clearly show the same targets shown in G.11. This information



(a) Signal 1 (b) Signal 2

Figure G.11: Individual Return Signals from Different Transmitters

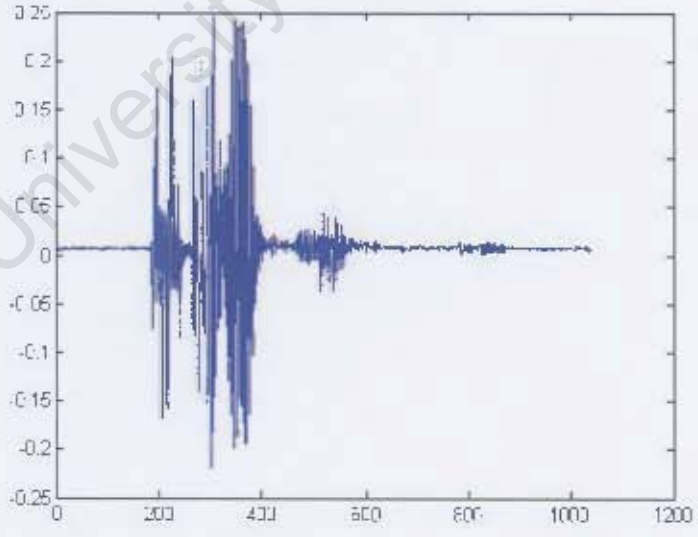
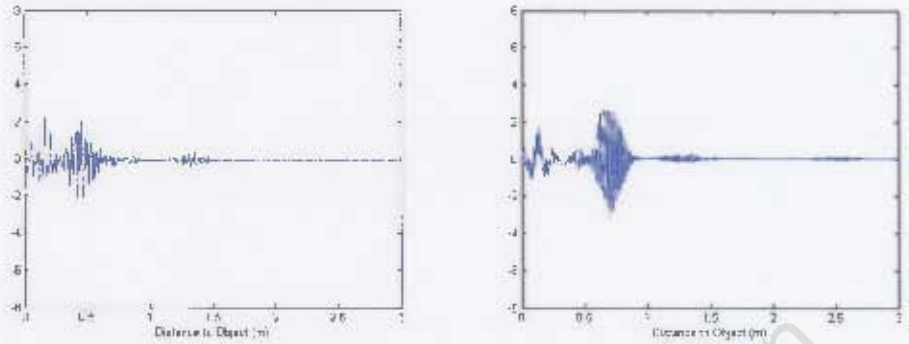


Figure G.12: Combined Signal



(a) Signal 1 Correlation (b) Signal 2 Correlation

Figure G.13: Signals Recovered by Correlation

was however recovered from the noisy combined signal. This demonstrates that the Walsh codes effectively identified the required signals and made the system immune to interference from multiple transducers.

G.4.2 Pseudo-Noise Signals

Another method of generating a unique transmission signal is to use a pseudo-noise signal. This is where a band-limited random signal is used to identify a particular transmitter. As long as the various transmitter signals have a low cross-correlation, they can be uniquely identified. Good correlation properties have to be determined experimentally as there is no deterministic method for generating two fixed length random signals with a low correlation [40].

Testing

Pseudo-noise signals were generated and their correlation tested. Two with a low cross-correlation were chosen as the transmitter pulses and used to attempt to identify objects at varying distances.

Results

Figures G.14, G.15 and G.16 show the correlated signals for objects at varying distances. The blue shows the correlation with the transmitted signal while the red shows the correlation with an incorrect signal. The targets can be

clearly identified by the spike in the correlated signal. The problem with these results, that made this method unsuitable for the task, was that the spikes for distant targets were below the noise level (correlation with incorrect signal) for closer targets. If the two signals were combined, the correlation procedure would be unable to pick out the distant target because it would be obscurred by the noise from the closer target.

This could be improved by increasing the bandwidth of the PN signals. Unfortunately this was not a possibility in this system because the bandwidth was limited by the transducers. Another possibility was using sophisticated signal processing to differentiate between a distant target and noise by comparing the maximum and average amplitude of the signal. This however would become problematic in more complex environments with more than one legitimate target.

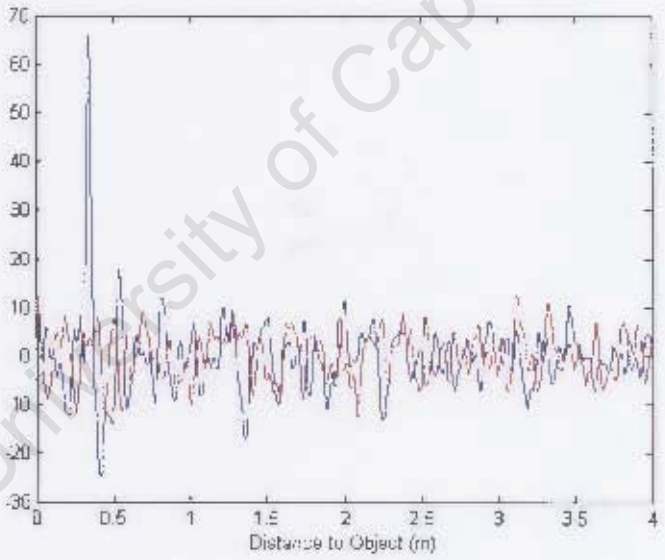


Figure C.14: Object at 300mm

C.4.3 Barker Codes

Barker codes are phase modulation codes which give the best range resolution because of their auto-correlation properties so these were considered. They are however unsuitable for a system where crosstalk is a factor because the signals cannot be distinguished from one another [35].

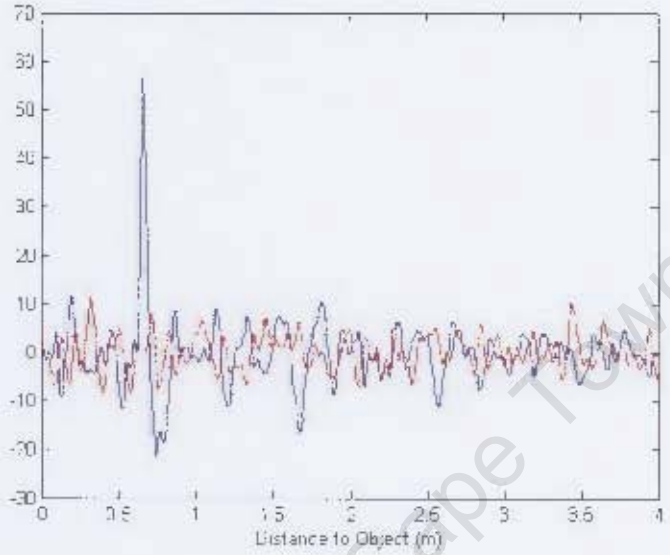


Figure G.15: Object at 600mm

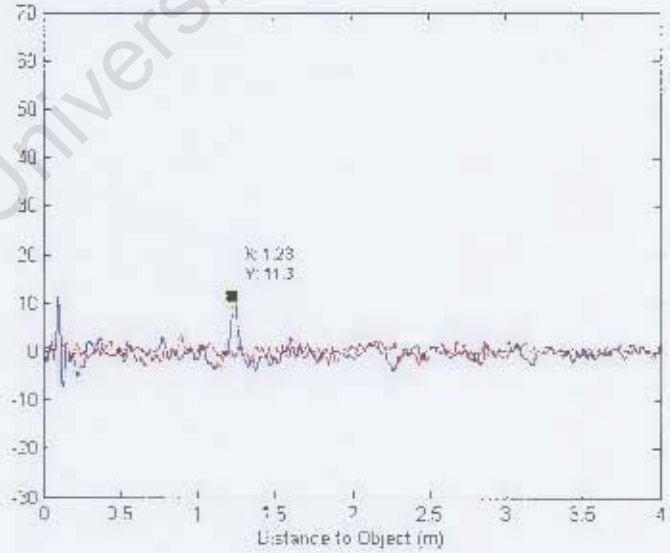


Figure G.16: Object at 1.2m

G.5 Conclusions

Encoding each transmission signal using Walsh codes to determine the phase modulation was found to be the most effective method of eliminating crosstalk and improving range resolution in this sonar system. Although they do not give the best performance for range resolution (as with Barker codes), they are a good compromise between good cross-correlation properties and sufficient resolution.

The bandwidth of the transducers proved to be a limiting factor in implementing the pseudo-noise pulses. Because of this limited bandwidth it was necessary to intentionally generate signals with low-cross correlation in order to eliminate crosstalk.

Appendix H

Combined Infrared and Acoustic Beacon Tracker Implementation on an Autonomous Vehicle

Combined Infrared and Acoustic Beacon Tracker Implementation on an Autonomous Vehicle

Graham, Brooker, Craig Lobsey,
Australian Centre for Field Robotics, University of Sydney, Australia
gbrooker@acfr.usyd.edu.au, clob7844@usyd.edu.au
Kate McWilliams, University of Cape Town, South Africa
katemcwilliams@gmail.com

Abstract

This paper describes the theoretical background and implementation of a dual IR and acoustic range and angle tracking system that allows a small robotic vehicle to find and follow a moving beacon. The sensor is integrated onto a modified radio control vehicle to demonstrate this capability.

Keywords: infrared, acoustic, time difference of arrival, beacon, tracking

1 Introduction

A low-cost system was required that would allow a number of small robotic vehicles to find and then accurately follow their human "owners". RFID and GPS based options were considered initially, however they were rejected as unsuitable because the measurement of relative angles and small range differences accurately is not cost effective using these techniques. More conventional time-of-flight techniques were deemed to be more suitable. Range measurement could either be undertaken using standard pulsed ultrasound, or more effectively using a beacon that transmitted an electromagnetic and an acoustic signal simultaneously, and then measuring the difference in the arrival times, a technique known as time difference of arrival (TDOA).

The main advantage of this method is that it allows the fast electromagnetic signal to carry a unique ID code which would identify a specific beacon to a specific receiver. TDOA systems have been developed for indoor localisation over the past decade. These include the well known MIT Cricket [1] and various other pulsed systems [2, 3]. Issues with multipath and acoustic interference have resulted in the use of spread-spectrum techniques [4, 5] originally pioneered for sonar sensing in robots [6], or a sequential approach [7].

Initial experiments were conducted using RF modules designed for keyless entry. However, it was found that most of these have a substantial (>100ms) and variable period between the transmission of the encrypted signal and its acknowledgement making them unsuitable for the measurement of arrival time differences of less than 15ms required for a 5m maximum range.

A viable alternative to this problem as illustrated in Figure 1 is discussed in the following sections.

2 Operational Principles

An alternative, and apparently unique, technique using standard infrared remote control hardware was implemented to produce a range measurement system which works as follows:

- The beacon transmits a short pulse of IR light with a wavelength of 960nm on-off modulated at 38kHz.
- It then transmits a 2ms burst of ultrasound at a frequency of 40kHz.
- The receiver detects the IR pulse, which, to all intents is instantaneous, followed shortly afterwards by the slower acoustic pulse.
- Because the speed of sound is reasonably constant (about 340m/s) and slow, it is easy to measure the time difference, ΔT, from which the range can be calculated.

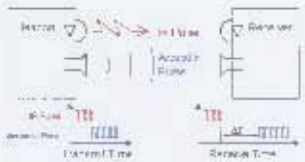


Figure 1: Conceptual diagram of the system showing the range measurement technique

The relative angle of the beacon with respect to the receiver is determined using a variation of the conventional amplitude comparison monopulse technique developed during WWII for tracking aircraft using radar [8, 9]. Though this process is usually applied using electromagnetic signals, a number of acoustic applications have been made [10, 11], where its use as an aid to the blind has been quite successful.

In this prototype, both IR and acoustic implementations of the angle measurement are made. In the IR implementation, two receivers are placed between an opaque septum as shown in Figure 10 so that as the angle of arrival varies, first the one receiver is illuminated, and then both, and finally the other. This produces four states: No signal (if the beacon is outside the angle limits of both receivers), angle left (if one receiver is illuminated), angle straight on (if both are illuminated) and angle right (if the other receiver is illuminated). This can be used to drive a bang-bang controller to adjust the coarse steering of the robot.

The size of the septum can be adjusted to control the angle during which both receivers are illuminated, and so define the "dead" band of the controller.

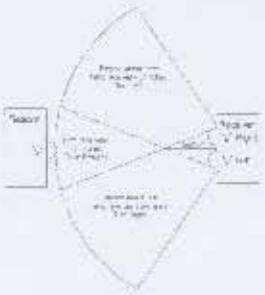


Figure 2: Conceptual diagram showing digital angle measurement

Because of the finite size of the receive aperture lens and diffraction around the septum, there will be a fuzzy region of transition between the receiver seeing the beacon and not. This will depend on the intensity of the beacon, and the ambient light level. It has been estimated that a dead band of much less than 10° could not be reliably obtained by this method.

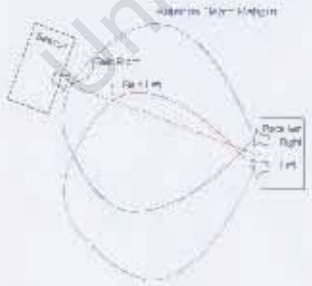


Figure 3: Conceptual diagram showing analog angle measurement

In the case of the acoustic system, because the amplitude of the received signal is determined by the receiver beam pattern, a far more subtle measure of the angle is possible. Two receiver transducers are mounted with a slight outward squint so that if the acoustic signal arrives at an angle, the signal amplitude received by one of the receivers will be higher than that received by the other as illustrated in Figure 3. Only when the illumination is symmetrical will the two signals be equal and so there will be no dead band.

A simulation model was built using the ultrasonic transducer normalised gain pattern supplied by the manufacturer. The difference between the signal voltages received from the left and the right transducer are used to produce the receive antenna transfer function shown in Figure 4. It can be seen that the peak magnitude of the error signal is dependent on the squint angle. It can also be seen that there is a region of about +/-20° over which the relationship between the voltage and the beacon angle is almost linear.

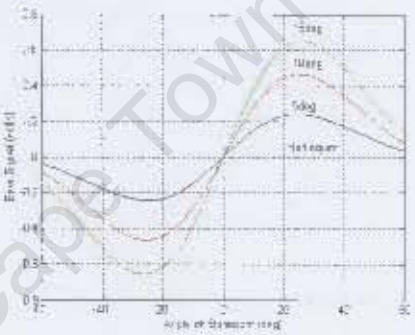


Figure 4: Angle error transfer function with squint angle as a parameter

The slope of the error signal at the origin is a good indication of the potential tracking accuracy, and it can be seen that this is proportional to the squint angle. The down side is that if this squint angle exceeds 15°, the on boresight signal amplitude decreases with a resultant reduction in the signal to noise ratio (SNR).

In addition, the amplitude of each of the received signals decreases considerably as the beacon angle increases, and in the end a threshold will be reached below which the SNR will be too low for reliable measurements to be made.

Because the signal level is also dependent on the range to the beacon, this angle limit threshold will be range dependent.

3 Tracker Implementation

3.1 Beacon

To make the beacon as light and unobtrusive as possible, it consists of a PIC micro-controller (PIC16F62X). From the schematic shown in Figure 5, it can be seen that it drives two IR LEDs with a peak current of 80mA for 1000µs pulse modulated at 38kHz. The pulse repetition interval is 50ms making the duty cycle 0.02 × 0.5 = 1% and reducing the average current consumption to 0.8mA per LED.

The micro-controller also drives two ultrasonic transducers at 40kHz through switching transistors and step-up transformers to provide each with a 20Vpp signal which is their maximum rated drive level. From the schematic of the beacon is shown in Figure 5, it can be seen that two transducers are used to widen the azimuth coverage to ensure that the beacon will be seen by the receiver even if it is at an angle.

A visible LED, not shown in the schematic, can be included if necessary and strobed for a brief period once every second or so to indicate that the beacon is switched on and working.

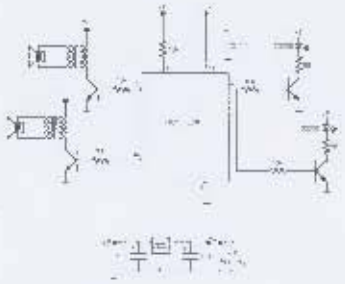


Figure 5: Circuit diagram of beacon

In this prototype the 9V supply was regulated down to 5V using a linear regulator. This is inefficient and it is proposed that a small DC/DC converter should be used to improve the battery life.

Figure 6 shows the layout for the beacon transducers. It is important that the IR LEDs and the Ultrasonic Transducers be squinted in the horizontal plane to improve angular coverage. It is also important that the two Ultrasonic Transducers be mounted one above the other, as close together as possible to reduce the magnitude of the elevation lobing.

Figure 6: Layout of beacon transducers

At this stage the beacon will operate reliably for about 6 hours using one 9V alkaline battery. This period would be doubled if an efficient DC/DC converter replaced the linear regulator.

3.2 Receiver

The receiver comprises a pair of ultrasonic transducers amplified by a conventional stereo pre-amplifier as shown in Figure 7. It is suggested that this is replaced by a pair of analog receiver IC's manufactured by SenseComp if better range performance is required.

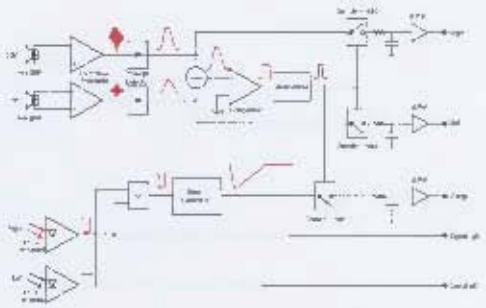


Figure 7: Receiver schematic block diagram

The amplified acoustic signals are then detected and filtered to produce an envelope of the pulse shape as shown in Figure 8.

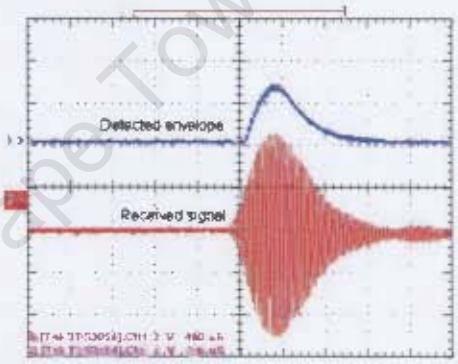


Figure 8: Received acoustic signal and the detected envelope

The sum of the envelope voltages from the left and right channels is then converted to a digital pulse using a comparator configured with a Schmidt trigger input and with an adjustable threshold level. Because the length of this pulse varies with the received signal amplitude, it triggers a monostable with a constant duration pulse of 500µs.

This pulse then performs a number of functions. In the first instance it officially marks the receipt of the acoustic pulse for the range measurement circuitry which is explained later. Secondly, it triggers a sample and hold which samples the detected envelopes of both the receivers, and holds the final value for the 50ms inter pulse period.

The reason that the monostable pulse duration is 500µs is because that is approximately the rise time of the received pulse (as can be seen in Figure 8). This ensures that the output of each sample and hold, holds the peak received amplitude.

The sample and hold circuitry consists of an analog switch followed by a resistor and a capacitor the output of which is buffered by a non-inverting follower. The high input impedance of the buffer, and the high off resistance of the analog switch ensure that

the charge remains on the capacitor for the complete 50ms inter pulse period with minimal droop.

The IR circuitry consists of a pair of buffered IR receivers that produce a TTL low on receipt of an IR signal modulated at 38kHz. These signals are read into a PIC micro-controller (not shown in the schematic) for further processing.

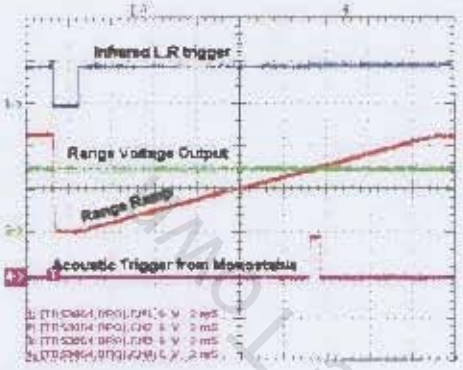


Figure 9: Analog range signals

An output, which is the logical OR of the two IR receiver inputs serves as a trigger for the analog range voltage generator. The range circuit comprises a constant current generator that charges a capacitor to produce a linear voltage ramp with a slope of about 0.75V/ms. The capacitor is discharged on receipt of the IR input and then charges for the full 50ms. This voltage, which is directly proportional to the elapsed time, is sampled at the instant that the acoustic signal is received. The output is therefore directly proportional to the range from the beacon to the receiver.

3.3 DC Power Supply

A pair of DC:DC converters convert the battery supply from the vehicle into +/-15V and 15V required for circuit operation. Because these systems generate switching transients at both their input and outputs, they should be filtered using the appropriate LC networks. In addition, the +/-15V rails are regulated to +/-12V to reduce this noise still further.

This discussion has shown that the receiver output comprises two digital signals from the IR receivers which are the wide angle capture signals and three analog signals which are the fine angle and range control signals.

4 The Vehicle

To demonstrate the capabilities of the receiver, it was mounted onto the chassis of an off-road radio-controlled vehicle (Tamiya Overlander) as shown in Figure 10.

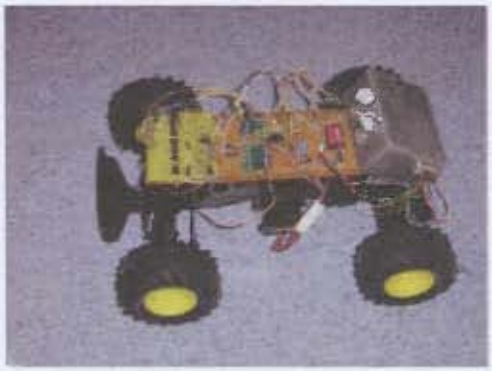


Figure 10: Radio controlled car chassis with sensor and prototype control electronics

An Atmel Atmega-128 8-bit microcontroller runs the software which performs the tracking function. In order for the vehicle to properly track the target, the presence of the acoustic signal and at least one infrared signal is required. An additional output from the sensor indicates this presence and notifies the controller that the incoming sensor data is valid. With this valid data, the vehicle can track the target using proportional control. The microcontroller digitises the range and the two angle voltages as well as sampling the digital outputs of the sensor.

Figure 11 shows a flowchart of the tracking algorithm executed by the microcontroller. The digitised range voltage, V_{range}, is subtracted from a set-point, V_{set}, to produce the difference which controls the speed of the drive-wheels, R.

$$R = K_1(V_{range} - V_{set}) \quad (1)$$

To control the vehicle steering, a simple linear tracking algorithm was implemented by estimating the offset error, ΔAz, using the left and right receiver voltages (V_{left} and V_{right}).

$$\Delta Az = K_2 \frac{V_{left} - V_{right}}{V_{left} + V_{right}} \quad (2)$$

The value is normalised to full scale and used to set the steering angle. Normalisation of the error signal is necessary to provide accurate steering control independent of signal amplitude.

At low signal-to-noise ratios, however, amplification of noise introduces unwanted inputs into the control algorithm. For this reason, the amplitude of the received acoustic signal is considered and if it drops below a threshold, the acoustic data is ignored. The algorithm also includes a small dead band to prevent oscillation around the centre point.

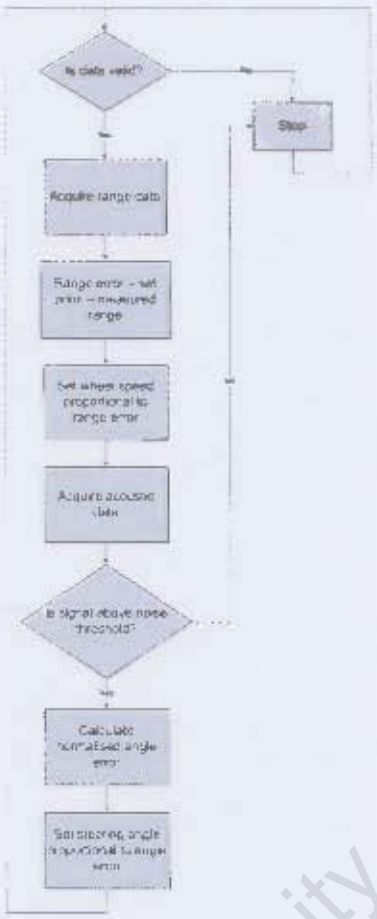


Figure 11: Flowchart showing the tracking algorithm implemented on the Atmega-128 microcontroller

If it is desirable that the vehicle track the beacon even when the acoustic data is not present, the infrared signals can be used. The vehicle could rotate until the infrared signal illuminates both optical sensors. If the acoustic signal is out of range, the vehicle could move forward using a bang-bang controller based on the infrared tracking signal until the acoustic signal is acquired.

It must also be noted that due to the mechanical configuration of the vehicle, it can only track the angle when it is also moving forward.

4.1 Mounting Position

For the vehicle to be used for this application, the receiver can only be mounted about 5cm off the ground as shown in Figure 10.

This mounting position has a number of drawbacks:

- The unit will have to look up toward the beacon if it is mounted on the owner's belt (about 1m off the ground)
- The angle to the beacon will vary with distance and will require a fairly wide elevation beamwidth to compensate (about 34° for a ground range between 1m and 5m)
- There is a chance that the unit will look directly into the sun which might have a detrimental effect on any IR sensors used.

Figure 12 shows the normalised elevation gain for the ultrasound transducer as a function of beacon range for a number of tracker tilt angles. It can be seen that at short range where the relative angle is large, the gain is low, but at longer ranges it remains almost constant as the angle changes very little with range.

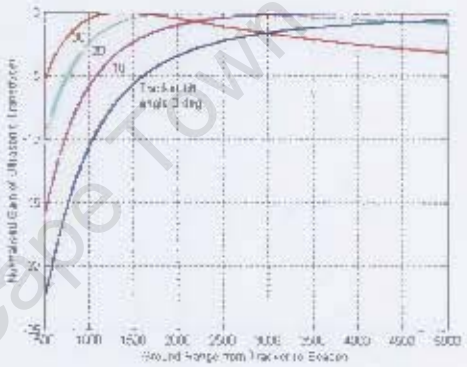


Figure 12: Effect of beacon range on ultrasonic receiver gain

A reasonable compromise would be to use a tilt angle of 20° as that maximises the gain at a ground range of 2m without having it roll-off too quickly at longer ranges where the signal level will be lower.

This angle could be optimised further by modelling the actual received signal level as a function of range instead of just using the antenna gain.

4.2 Vibration

Because the receiver comprises two sensitive ultrasonic transducers followed by a pair of high gain amplifiers, it is very sensitive to loud noises in the vicinity, and in this prototype, vibration of the housing. To address the vibration issue, the transducers should be isolated from the housing, and as an added precaution, the housing should be mounted on the vehicle using shock mounts.

4.3 Beacon Mounting Effects

There are two restrictions to the beacon mounting position. Firstly because of the limited angle of view of its transducers, it must be mounted so that it faces the vehicle while the owner is walking, and secondly, because neither the IR nor the acoustic signals will penetrate clothing effectively, it must be mounted in a position where shirts and jackets cannot obscure its view.

5 Preliminary Results

5.1 Noise and Multipath

Tests of the system showed substantial noise on the acoustic signals. The noise led to instabilities in the steering which were eliminated using a digital low-pass filter. A ten-sample average was found to be sufficient to create smooth steering behaviour with minimal computational cost. There are many possible sources of this noise, including interference from the PWM controlled servo motors, digital noise from the sensor circuitry and environmental noise.

Although the vehicle could detect a loss of signal and behave appropriately, it would occasionally track a false target. This was due to multipath errors from the acoustic signals in a cluttered environment. If the beacon was switched off, the vehicle would immediately stop tracking the false target, demonstrating that the vehicle was indeed tracking a reflection.

5.2 Control Algorithm

Proportional control was implemented for both the steering and range parameters, which was sufficient as the slow response of the mechanical components added considerable damping to system.

The beacon following capability of the system was good, if the beacon remained within the receiver beam. The vehicle was capable of following a person carrying the beacon at a fast walk, and was capable of performing a 180° turn with a radius of less than 3m.

At maximum vehicle speed, there was some overshoot if the beacon came to a sudden stop. For applications that require high vehicle speed and sudden changes in control input, a PI or PID controller could be incorporated to improve response. This, however, would increase the processing requirements of the controller.

5.3 Beacon Position

The sensor was found to be particularly sensitive to the position and angle of the beacon relative to the receiver. This sensitivity is due to the beam pattern of the acoustic transducers and could be improved by using transducers with a larger beamwidth. This characteristic highlighted the advantage of including the bang-bang controller using the infrared sensors which would allow the vehicle to return to a position where the acoustic signal is usable.

6 Conclusions

This paper has discussed the design and implementation of a simple dual IR and acoustic tracker and its evaluation on a radio controlled car chassis. The implementation is unique insofar as it combines the broad acquisition angle of IR sensors (nearly 180°) with the precision angle measurement capability of an acoustic monopulse tracker. Additionally it uses the difference in the time of flight of the IR and acoustic signals to measure the range to the beacon as well.

Preliminary results of the tracker performance were promising with the vehicle capable of following a fast moving person.

7 Acknowledgements

We would like to thank the ACFR (University of Sydney), the University of Cape Town and Itech Corporation for supporting this project.

8 References

[1] N. Priyantha, "The Cricket Indoor Location System," in *Computer Science and Engineering*, vol. PhD: MIT, 2005.

[2] L. Girod and D. Estrin, "Robust Range Estimation Using Acoustic and Multimodal Sensing," presented at Intelligent Robots and Systems (IROS 2001), Maui, Hawaii, 2001, 1312-1320.

[3] A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Personal Communications*, pp. 42-47, 1997.

[4] G. Bortolotto, F. Masson, and S. Bernal, "USRPS - Ultrasonic Short Range Positioning System," presented at IX Reunion de Trabajo en Prosesamiento de la Informacion y Control (RPIC 2001), Santa Fe, Argentina, 2001, 658-663.

[5] R. Palmer, "A Spread Spectrum Acoustic Ranging System - An Overview," presented at IEEE Canadian Conference on Electrical & Computer Engineering, 2002, 1242-1245.

[6] K. Jorg and M. Berg, "First Results in Eliminating Crosstalk & Noise by Applying Pseudo-Random Sequences to Mobile Robot Sonar Sensing," presented at EUROBOT'96, 1996, 40-45.

[7] Y. Fukuju, M. Minami, H. Morikawa, and T. Aoyama, "DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Environment," presented at IEEE Workshop on Software Technologies for Future Embedded Systems (WSTFES'03), 2003, 53-56.

[8] A. Leonov and K. Fomichev, *Monopulse Radar*: Artech House, 1986.

[9] M. Skolnik, *Introduction to Radar Systems*, 2nd ed: McGraw-Hill Kogakusha, 1980.

[10] R. Aguilar and G. Meijer, "Low-Cost Ultrasonic Fusion Sensor for Angular Position," presented at SeSense 2002, 2002, 594-597.

[11] R. Kuc, "Binaural Sonar Electronic Travel Aid Provides Vibrotactile Cues for Landmark, Reflector Motion and Surface Texture Classification," *IEEE Trans. on Biomedical Engineering*, vol. 49, pp. 1173-1180, 2002.