Navigation and Localisation of Autonomous Vehicles

Matthew Ricci

SID: 9930385

Australian Centre of Field Robotics,

Department of Mechanical Mechatronic and Aeronautical engineering,

University of Sydney

Supervisor: Dr. Eduardo Nebot

November 2002

A thesis submitted in partial fulfillment of a bachelor of mechanical (Mechatronics) engineering

Acknowledgements

I would first of all like to thanks all those in the ACFR who helped me during the course of this thesis, especially to Juan Nieto who was always willing to take time out to explain things to me even when he might have been busy.

I also want to thank my fiancée Leah for her support during this time, I know it has been tough not being able to see me much, but I thank you for being supportive.

I would also like to thank Benedikt for being a great work partner, without him, working on this thesis would have been a lot more tedious. Thanks also to Jiri and Kenny for gracing us with your presence and providing me with someone to beat in AFL slime.

Finally I would like to thank Conrad for his input in this thesis

Statement of Student Contribution

- I wrote a C version of the encoder/GPS kalman filter
- I tuned this filter offline to handle various types of GPS data
- I implemented a real time version of the filter in hyperkernal and wrote a simple GUI to plot vehicle paths in real time
- I conducted tests to ensure that the filter worked reliably in real time
- I implemented a laser based dead reckoning routine based on matching consecutive features from laser scans in C
- I researched the maximal common subgraph algorithm for data association and implemented this on a sample laser data

The above is an accurate statement of the students contribution.

Dr. Eduardo Nebot

Thesis Supervisor

Abstract

This thesis aims to develop a reliable, robust and accurate navigation filter for use on an autonomous land vehicle. The filter developed was based on a kalman filter implementation fusing data from model-based predictions using velocity and steering information and the absolute position information provided by the GPS. The filter was tuned to be able to deal with multipath efficiently and still provide accurate position estimates. An offline simulation was developed and this was then ported to a real time implementation using the Hyperkernal real time operating system. Extensive testing on filter performance was conducted to ensure that the filter performed adequately under most conditions.

An investigation into whether consecutive laser scans could be used to localize the vehicle. A robust data association technique was developed based on a maximal subgraph between two graphs. Techniques for extracting features from raw laser data were also investigated and this data used to calculate pose change between consecutive scans.

Table of Contents

Table of Figures

Chapter 1:

Introduction

1.1 Background to the High Speed Vehicle (HSV) project

The Australian Centre of Field Robotics (ACFR) initiated the High Speed Vehicle (HSV) project in 1997 as a test bed for the development of an autonomous vehicle. The mission statement of the project was to develop an autonomous system capable of operating in a variety of unknown outdoor environments and potentially at high speeds (90km/h). The potential applications for such a system are enormous ranging from autonomous control of vehicle in dangerous environments to reduce the risk to a human operator, such as in the mining industry or to things such as autopilot type systems for suburban vehicles.

1.1.1 Test Vehicle

All testing and development is conducted on a late 90's model Holden utility, pictured in Figure 1.1, referred to from here on in as the Ute. The Ute has been fitted with a substantial suite of sensors and actuators to allow for autonomous control and navigation of the vehicle. For control and data logging purposes a PC (Pentium II 400mHz 64MB RAM) running the Windows NT operating system (with the Hyperkernal real time extension) along with data logging equipment such as an A/D converter have been installed in the Ute's tray. A flat screen monitor has also been installed in the Ute's cabin to facilitate online monitoring of system performance.

Figure 1.1: HSV test vehicle (the UTE)

A substantial suite of sensors has been installed to allow for a number of different navigation and control options. The sensors available are as follows:

- Global Position System (GPS) – The GPS returns absolute positioning information of the vehicle in terms of latitude and longitude based on signals received from a number of satellites orbiting the earth. The latitude and longitude signals must then be transformed to a local co-ordinate frame for vehicle navigation. Greater accuracy can be obtained using what is known as differential GPS (DGPS) whereby a base station of known position is set-up to transmit GPS data to the vehicle.

- Inertial Measuring Unit (IMU) – The IMU, located in the Ute's tray, returns acceleration and angular rotation rates about the Ute's three axes (X,Y,Z). This information can then be integrated to obtain the velocity, position and roll, pitch and yaw angles of the Ute.

- Optical Encoder – The optical encoder is located on the rear passenger side wheel and returns the angular position of the wheel. When combined with a timestamp the difference between consecutive encoder counts is typically used to determine vehicle velocity.

- LVDT – The linear variable displacement transducer is located on the steering column of the Ute and is used to determine the position of the steering column and hence the steering angle of the wheels.

- Potentiometers – Essentially variable resistors these devices are mounted onto the brake and throttle actuators and based on the voltage returned by the device, the positions of the brake and throttle can be calculated.

- Range and Bearing Laser – A SICK laser is mounted on the bulbar at the front of the Ute, on the passenger side. The laser returns range and bearing information over a 180° scan at a resolution of 0.5°. Based on these readings various different obstacle detection, mapping and navigation algorithms can be developed.

- Compass – A compass is also installed on the Ute to provide an alternate source of heading information. Typically a compass is used only for initialisation.

- Cameras – Three cameras are mounted atop the HSV cabin for testing and development of various different vision techniques like panoramic vision and stereovision.

Actuation facilities on the brake, throttle and steering have also been installed to allow the Ute to be driven autonomously. These are as follows:

- Steering – A 24V DC motor mounted next to the brake pedal (shown in Figure 1.2), is coupled directly onto the steering column via and electromagnetic clutch, to directly control the position of the steering wheel.

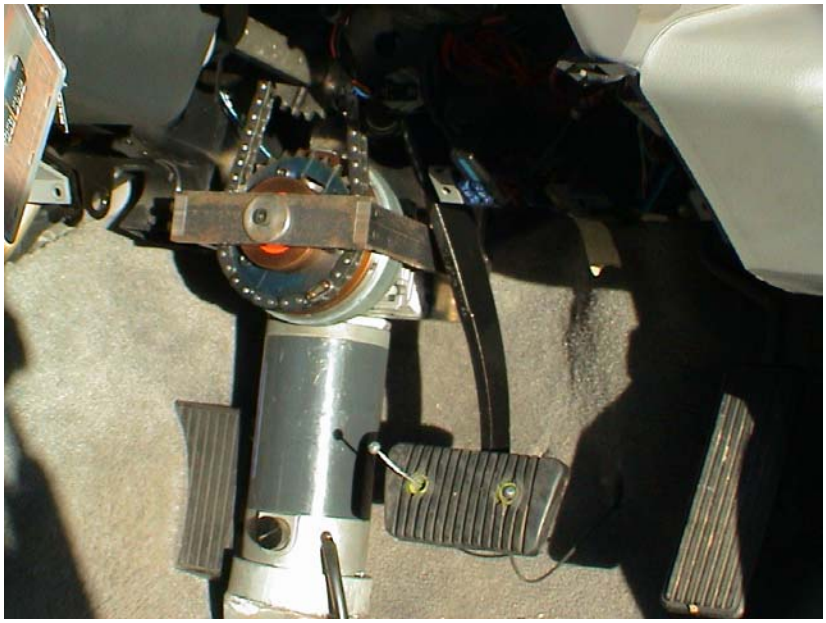


Figure 1.2: Steering control motor and clutch

- Throttle – Throttle control is achieved via a linear actuator that controls the displacement of the carburettor valve via a guide cable.

- Brake – Brake control is achieved by a linear actuator that controls the position of the brake pedal directly via two guide cables, shown in Figure 1.2.

1.1.2 Hyperkernal

It is convenient at this point to give a more detailed explanation of hyperkernal since the online implementation of some of the algorithms discussed is a major part of this thesis.

Hyperkernal is a real time subsystem for Intel-based PC's running the Windows NT operating system. Windows NT is a popular operating system but is simply not designed for real time applications. To overcome this problem the hyperkernal extension to Windows NT was installed on the Ute's computer. Hyperkernal incorporates all the usual functions of a real time operating system such as, multi-threaded programs with multiple priority levels, precisely timed events and interrupts and direct access to the hardware, all things that are needed for a real time system but can't be achieved using windows NT. Perhaps the biggest thing about hyperkernal, for this thesis anyway, is the shared memory database used for communication between hyperkernal and Windows NT

applications. Hyperkernal applications running at the highest priority level of the processor can read or write data to the shared memory database and Windows NT applications, running whenever Hyperkernal frees the CPU, can read or write the data to/from the shared memory. In this way hyperkernal applications and Windows NT applications can communicate as shown in Figure 1.3. A perfect example of this is a hyperkernal front-end program that reads the sensors, publishing the information into the shared memory, then a C++ program reads the shared memory and graphically displays the sensor information.

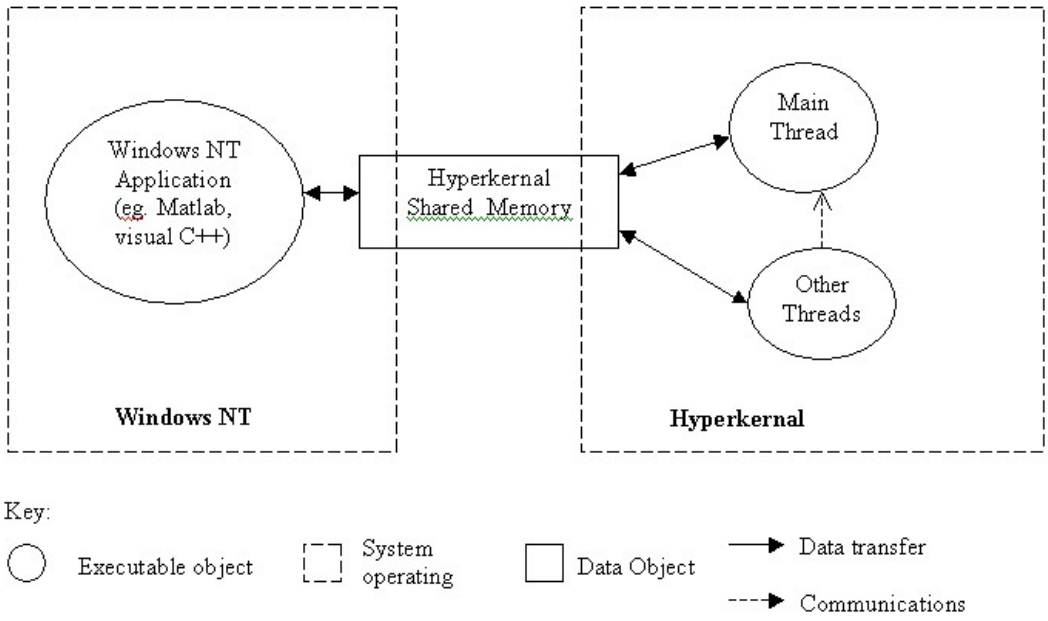


Figure 1.3: Hyperkernal shared memory architecture

The greatest attraction of the hyperkernal system as a real time platform lies in the shared memory communication between applications since programs can be written in the more user friendly and graphics friendly Matlab and run online on the vehicle. Also the Windows environment makes programs easier to run and debug. However Hyperkernal has some flaws the most obvious being that the system is very particular in that it only runs on Windows NT platforms and cannot be ported to other Windows operating systems such as Windows 2000 or XP. Hence any code developed on the hyperkernal system must be changed significantly before being ported to another operating system. Also any code that must run in the Hyperkernal part of the operating system must be

written in C and not C++. Despite its shortcomings hyperkernal was used as the base for the development and implementation of all the algorithms discussed in this thesis.

1.1.3 Past and Current Work

Since it's inception in 1997 large and varied amounts of research have been carried out on the HSV project using the Ute as a test platform. In the early stages, most of the work was focussed upon installing hardware and writing drivers to allow for efficient operation of the sensors and actuators. Much work has also been done on the low-level control aspects of the vehicles actuators, ensuring that the basic framework for autonomous navigation has been in place. More recently work has been undertaken on more high level aspects of vehicle control such as path planning, taking into consideration vehicle constraints, path tracking, obstacle detection using the laser and obstacle avoidance or reactive navigation. Work has also been undertaken in the area of using vision to identify obstacles and for vehicle localisation, as well as in the simultaneous localisation and mapping problem (SLAM).

At present the vehicle location for the high-level control modules such as path tracking and obstacle detection and avoidance comes solely from the GPS. Apart from GPS estimates being quite low frequency (5Hz), the GPS also suffers from other problems such as multipath when near trees or buildings due to reflection of the buildings and or trees or the GPS antenna being obscured by the environment. These errors make the GPS far from robust and as such the GPS can only be used reliably for navigation in open environments. Naturally differential GPS (DGPS) can be used to improve these errors, however DGPS may not always be available since it involves setting up a base station within range of the GPS receiver. As a result to allow the vehicle to operate in an autonomous fashion it is necessary to come up with other methods of vehicle navigation based upon the other sensors and available in real time.

1.2 Goals of this Thesis

The primary objective of this thesis is to provide a robust and reliable algorithm capable of providing accurate position estimates of vehicle position. The method must be available in real time so that other software modules may perform more complex tasks,

such as path tracking, obstacle avoidance or surveillance. A strong component of this thesis is to make the localisation routines very modular in design so that a user need have no knowledge of how the modules compute position; it can simply read the position information provided, given a template of how the position information is formatted and stored in memory. Another key component of the thesis is that the localisation modules are available in real time and as a result of this it was decided that all modules would be coded in C for fast operation and easy portability.

1.3 Thesis Outline

This thesis consists of two main areas, first a model based dead reckoning navigation loop is developed using a Kalman filter to fuse together position estimates based upon a vehicle model and also absolute position data made available by the GPS. The equations used by this process are analysed initially followed by a detailed analysis of the navigation loop under many different conditions. Fault detection is of prime importance to the navigation loop to ensure that even with GPS errors the navigation loop can still provide accurate position data.

The second part of this thesis constitutes an investigation into whether the laser can be used to provide a path estimate of vehicle motion by matching consecutive scans and tracking objects within these scans. Features are extracted from the raw laser data using various line and circle detection techniques and following this, a one to one mapping of features is found using the maximal common subgraph method.

Chapter 2:

Statistical Estimation

2.1 Introduction

In any localization or Navigation technique statistical estimation plays a crucial role since one can never be entirely sure of the vehicle position. Hence the position must be estimated reliably and so a robust algorithm is required to do this, so as to provide the position estimate with the least statistical error. The method used in this thesis is the Kalman filter.

The Kalman filter essentially consists of two stages, the prediction and the update. During the prediction stage the next state of the system is estimated based on the current state and some control input. This process is repeated until some absolute information about the system state is available, upon which an update is performed. The update simply fuses together the predicted state and the observed state to provide a least squared estimate of the vehicle state. A prime example used in this thesis is the prediction of vehicle position based upon velocity data taken from wheel encoders and steering data taken from the steering LVDT. The vehicle position is continually predicted using this data until GPS information becomes available upon which an update is performed and the data fused to provide a new position estimate of the vehicle.

This chapter simply outlines the mathematics and general principles behind the Kalman filter and the subsequent chapters it's application in this thesis.

2.2 Kalman Filter

The Kalman filter is a recursive algorithm that is used to provide statistical estimates of the system state given all the information preceding the current state. The nature of the Kalman filter is that it will minimize the least squared error of the system state during each iteration. The Kalman filter consists of two main stages the prediction stage, whereby the next state of the system is predicted based upon the current vehicle state and a system model, and the observation stage whereby the information provided by the prediction stage is fused with some external absolute information about the system states. The next section of this thesis will briefly outline some of the equations used to perform the prediction and update stages of both a linear and extended a Kalman filter. (It is noted that the equations used in the Kalman filter are well documented and for a more detailed description the reader is referred to [7], [21] and [24].)

The prediction part of the Kalman filter is performed using a discrete time linear model, described by Equation 2.1.

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{B}(k)\mathbf{u}(k)$$

Equation 2.1

Where:

$\mathbf{x}(k)$ is the predicted state of the system at time k

$\boldsymbol{F}(k)$ is the state transition matrix at time k, which relates the previous state $\mathbf{x}(k\text{-}1)$ to the current state $\mathbf{x}(k)$.

$\mathbf{u}(k)$ is the input control matrix at time k

$\mathbf{B}(k)$ is a matrix that relates the input control signals to the system states

The uncertainty of the system states is represented by a covariance matrix ($\mathbf{P}(k|k\text{-}1)$) that contains the expected values of the variance of the error at time k given all information up to time $k\text{-}1$. The covariance matrix is calculated as per Equation 2.2.

$$\mathbf{P}(k \mid k-1) = \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^{T}(k) + \mathbf{Q}(k)$$

Equation 2.2

Where:

P(*k*|*k-1*) is the new covariance matrix based on the model uncertainty and the previous covariance matrix (**P**(*k-1*|*k-1*)).

F(*k*) is the state transition matrix described in Equation 2.1

Q(*k*) is a system noise component related to the uncertainties in the model and the input control vector.

For each iteration of the Kalman filter, Equation 2.1 and Equation 2.2 are computed to provide estimates of the system state and uncertainty, until some absolute information about the system states, known as an observation, becomes available. Upon observation the information contained in the observation is fused with the current prediction of the system state via Equation 2.3 to produce a new estimate of the system state.

$$\mathbf{x}(k \mid k) = \mathbf{x}(k \mid k-1) + \mathbf{W}(k)\mathbf{v}(k)$$

Equation 2.3

Where:

x(*k*|*k*) is the new state after the observation

x(*k*|*k-1*) is the predicted state at time k

W(*k*) is a gain matrix produced by the Kalman filter

v(*k*) is the innovation vector

The innovation vector (**v**(*k*)) is defined as the difference between the observation of the states at time *k* and the predicted value of the states at time *k* (Equation 2.4).

$$\mathbf{v}(k) = \mathbf{z}(k) - \mathbf{H}(k)\mathbf{x}(k \mid k-1)$$

Equation 2.4

Where:

z(*k*) is the observation of the states at time k

H(*k*) is a transformation matrix that relates the states to the observations since generally not all states are observed.

x(*k*|*k-1*) is the predicted value of the state at time *k* obtained from Equation 2.1.

From Equation 2.3 it is clear that the updated state is simply the predicted state plus some weighting of the innovation vector. This weighting is known as the Kalman gain and is calculated so as to minimise the least squared error of the new state estimate (Equation 2.5).

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)\mathbf{S}^{-1}(k)$$

Equation 2.5

Where:

$\mathbf{P}(k|k\text{-}1)$ is the current covariance matrix

$\mathbf{H}(k)$ is the matrix that relates he observations to the states (Equation 2.4)

$\mathbf{S}(k)$ is a matrix known as the innovation covariance

The innovation of the covariance is calculated by Equation 2.6.

$$\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}(k)$$

Equation 2.6

Where:

$\mathbf{R}(k)$ is a noise matrix associated with the observation

After the prediction information has been fused with the observation an updated covariance or uncertainty matrix must be calculated (Equation 2.7).

$$\mathbf{P}(k \mid k) = (\mathbf{I} - \mathbf{W}(k)\mathbf{H}(k))\mathbf{P}(k \mid k-1)(\mathbf{I} - \mathbf{W}(k)\mathbf{H}(k))^T + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}^T(k)$$

Equation 2.7

The equations above describe the recursive Kalman filter algorithm when applied to a linear model that can be described by Equation 2.1. However in a large number of situations processes behave non-linearly or cannot be linearised efficiently and so the Extended Kalman filter was developed to handle such instances.

2.3 Extended Kalman Filter

Similar to the Kalman filter the extended Kalman filter has a prediction stage followed by an update stage with the main difference being found in the fact that the model is now non-linear and hence the equations describing the filter are significantly different. The following section describes the equations behind an extended Kalman filter.

For a non-linear system the state prediction equation is shown in Equation 2.8.

$$\mathbf{x}(k \mid k-1) = \mathbf{F}(\mathbf{x}(k-1 \mid k-1), \mathbf{u}(k))$$

Equation 2.8

Where:

$\mathbf{F}(\mathbf{x}(k\text{-}1|k\text{-}1), \mathbf{u}(k))$ is a non-linear state transition function using the current state and a control input.

The covariance matrix after a prediction is defined in Equation 2.9.

$$\mathbf{P}(k \mid k-1) = \nabla \mathbf{F}_x(k)\mathbf{P}(k-1 \mid k-1)\nabla \mathbf{F}_x^T(k) + \mathbf{B}(k)\mathbf{Q}(k)\mathbf{B}^T(k)$$

Equation 2.9

Where:

$\nabla \mathbf{F}_x(k)$ is the Jacobian matrix of the partial derivatives of **F** with respect to the state vector **X**.

$\mathbf{B}^T(k)$ is source error transfer matrix calculated as the Jacobian matrix of partial derivatives of **F** with respect to the control input **u**.

$\mathbf{Q}(k)$ is the process noise covariance.

Similar to the linear Kalman filter when an observation occurs the state vector is updated using Equation 2.10.

$$\mathbf{x}(k \mid k) = \mathbf{x}(k \mid k-1) + \mathbf{W}(k)\mathbf{v}(k)$$

Equation 2.10

The innovation, innovation covariance and gain matrices are calculated using Equations 2.11 below.

$$\mathbf{v}(k) = \mathbf{z}(k) - \mathbf{H}(\mathbf{x}(k \mid k-1))$$

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\nabla\mathbf{H}_x^T(k)\mathbf{S}^{-1}(k)$$

$$\mathbf{S}(k) = \nabla\mathbf{H}_x(k)\mathbf{P}(k \mid k-1)\nabla\mathbf{H}_x^T(k) + \mathbf{R}(k)$$

Equations 2.11

Where:

$\mathbf{z}(k)$ is the observation

$\mathbf{H}(x(k|k-1))$ is current observation model

$\mathbf{P}(k|k-1)$ is the current covariance matrix

$\nabla\mathbf{H}_x(k)$ is the Jacobian of the current observation model

$\mathbf{R}(k)$ is a noise matrix associated with the observation

The new covariance matrix after the observation is given in Equation 2.12.

$$\mathbf{P}(k \mid k) = (\mathbf{I} - \mathbf{W}(k)\nabla\mathbf{H}_x(k))\mathbf{P}(k \mid k-1)(\mathbf{I} - \mathbf{W}(k)\nabla\mathbf{H}_x(k))^T + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}(k)$$

Equation 2.12

2.4 Filter Accuracy and Fault Detection

The algorithms discussed above provide estimates of the states of interest to provide a minimum least squared error, however unless a truth model of the states is known it is

unknown whether the filter is producing accurate estimates. The only information regarding the outside world comes in the form of the observation.

For correct filter performance the innovation must have the property that it is both unbiased and white with a covariance of **S***(k)*. To test this, the innovations are normalised (Equation 1.13) and if the filter is operating correctly the normalised innovations are distributed as a χ^2 distribution.

$$\gamma = \mathbf{v}^T(k)\mathbf{S}^{-1}(k)\mathbf{v}(k)$$

Equation 2.13

However instead of using Equation 1.13 as a check of filter performance, it could be used as a gating function to reject wild observations. Each time the observation occurs the value of γ is checked and if it is less than some threshold the observation is accepted if not the observation is rejected. The value of this threshold is calculated based on the standard χ^2 tables based on the confidence level required. For a 95% confidence the threshold is 12.6.

Chapter 3:

Coordinate Frames

Throughout this thesis various different coordinate frames are used to describe navigation data. It is important to clearly define each of the frames and the transformations between coordinate frames. A brief description of some of the coordinate systems used in this thesis is appropriate at this point.

3.1 Earth Centred Earth Frames

Earth Centred Earth Frames or ECEF frames have their origins at the fixed to the centre of the earth. Two major ECEF coordinate systems exist and these are rectangular ECEF coordinates and geodetic coordinates. [7]

3.1.1 ECEF Rectangular coordinates

A standard right hand rectangular coordinate system can be defined having its origin at the centre of the earth and is referred to as rectangular ECEF coordinates (Figure 3.1). The *x*-axis extends through the intersection of the prime meridian (0° longitude) and the equator (0° latitude). The *z*-axis extends through the North Pole. The *y*-axis completes the set passing through the equator at 90° longitude. [7]

3.1.2 ECEF Geodetic coordinates

The ECEF geodetic coordinate system has three parameters λ longitude, ϕ latitude and *h* altitude defined relative to the earth's ellipsoid (Figure 3.1). Latitude is defined in [7] as the angle between the ellipsoidal normal of the earth and the equatorial plane. Longitude is the angle in the equatorial plane between the prime meridian and the projection of the point onto the equatorial plane. The altitude is the distance along the ellipsoidal normal from the centre of the earth. For more information regarding ECEF frames the reader should consult the references particularly [7].

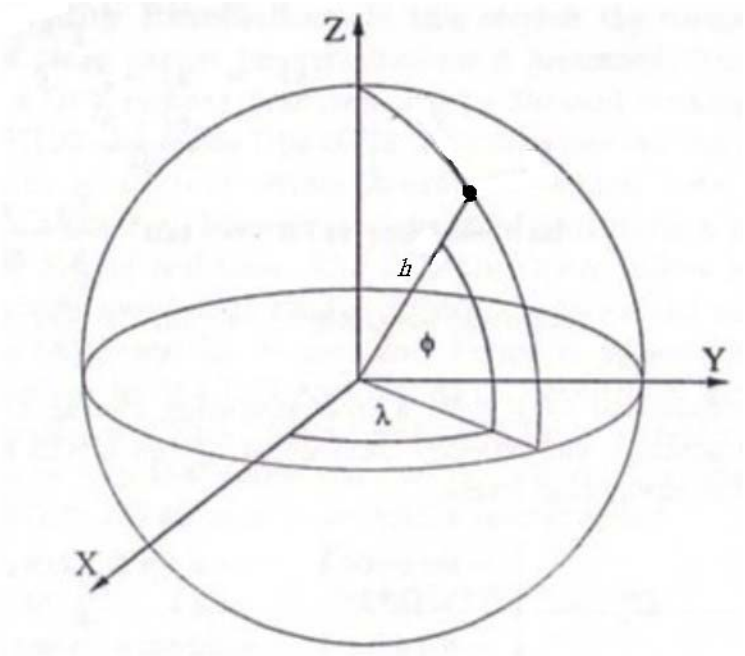


Figure 3.1: ECEF coordinate systems. ECEF rectangular (X,Y,Z) and ECEF geodetic (λ is longitude, ϕ is latitude and *h* is altitude

3.2 Local Navigation frame

It is often more convenient to work in a local navigation frame rather than the ECEF frames and so a local or navigation frame is defined. The local frame is the north, east and down rectangular coordinate system we use in our everyday lives.

The local navigation frame is defined by fitting a tangent plane onto the surface of the earth at the particular point of interest Figure 3.2. The plane remains fixed about this

point and this point becomes the origin of the coordinate frame. The *x*-axis points to true north, the *y*-axis points east and the *z*-axis points towards the centre of the earth. For this reason the local frame is often referred to as the NED (North East Down) frame. [7]

In this thesis the NED frame is the one used to define the navigation parameters and as such all measurements taken in the ECEF and body frames are converted to the NED frame.

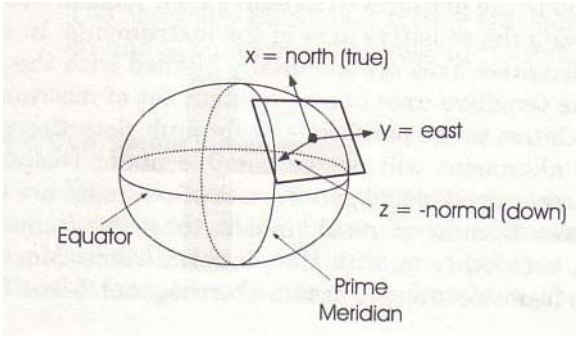


Figure 3.2: Local NED navigation frame [7]

3.3 Body Frame

Sensors reading from the vehicle are often taken with respect to the vehicle and so it is convenient to define the body frame as a vehicle centred rectangular coordinate system or polar coordinate system (**Error! Not a valid bookmark self-reference.**).

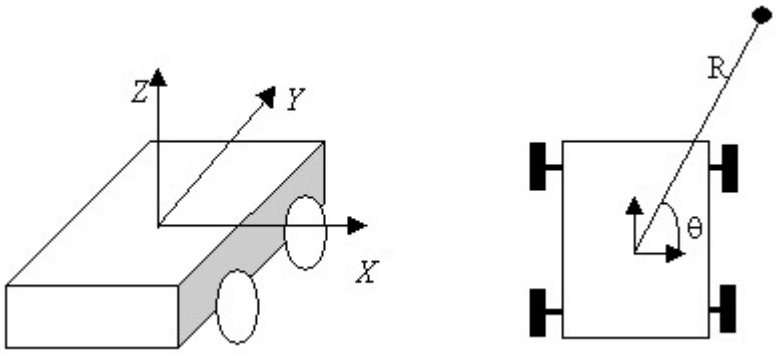


Figure 3.3: Cartesian and polar body frames

Chapter 4:

Model Based Dead Reckoning

Vehicle models can be used to make predictions about where a vehicle will be located at a particular time given information about the vehicles current position, velocity and steering angles. Naturally the error of such predictions will grow continually with time due to inaccuracies in the sensors used and in the model proposed. Therefore it is necessary not only to make predictions about vehicle states but also to have some absolute reference about the vehicle position. In this chapter the state predictions will be based on data from the LVDT on the steering column and the wheel encoders combined with a vehicle model and the absolute position information will arrive via the GPS.

The data will be fused together using an extended Kalman filter to provide accurate position estimates of the vehicle in a 2D environment.

This chapter first discusses the vehicle model used to perform the state predictions followed by how this model is incorporated into a Kalman filter. A brief analysis of aspects relating to filter tuning is also presented.

4.1 Vehicle Model

The vehicle model used is a simplified version of the Ackerman steering model assuming that both wheels have the same steering angle, so in truth the model more closely resembles a tricycle model. [9][11] Figure 4.1 shows the vehicle model used for dead

reckoning along with three coordinate frames: the navigation frame (X,Y), a vehicle frame centered about the centre of the rear axle (x_c,y_c) and a vehicle frame centered about the location of the GPS receiver (x_v,y_v).

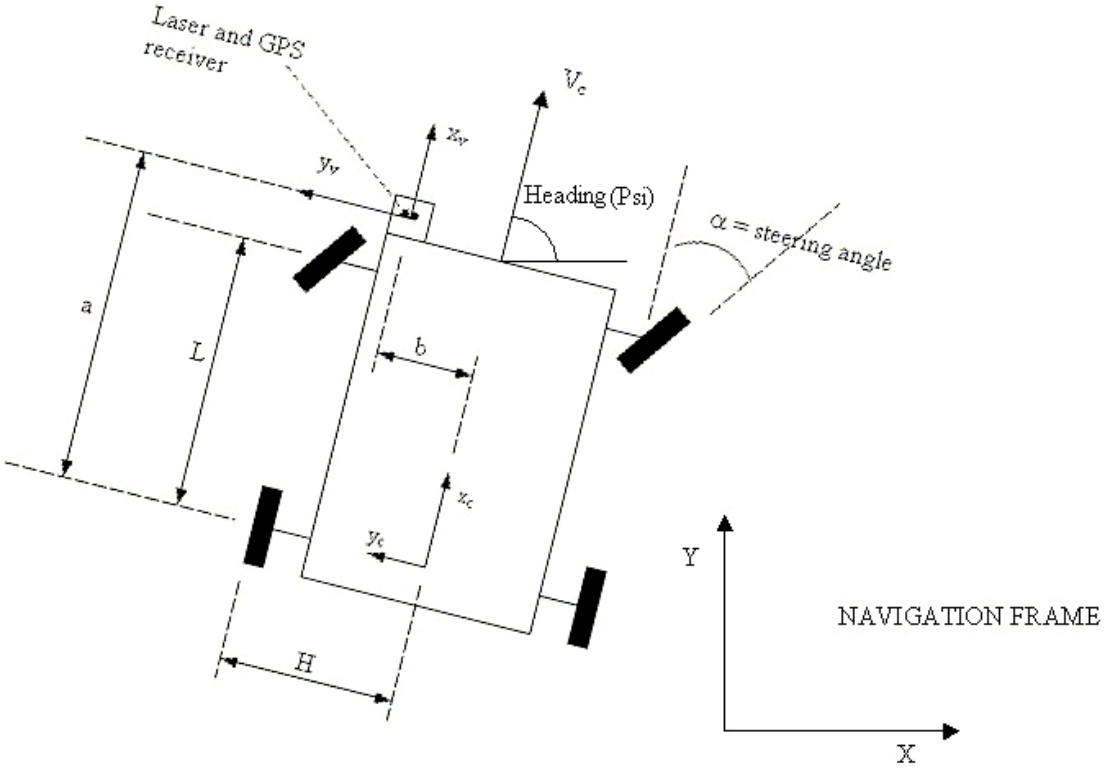


Figure 4.1: Vehicle model used for dead reckoning

Equation 4.1 shows the discrete time equations for predicting the position of the vehicle in the navigation frame based on the vehicle model. The new position of the vehicle is predicted based upon the last position and assuming a constant velocity and steering angle over the sampling interval. The current steering angle and velocity are sensed using the LVDT on the steering rack and the optical wheel encoders respectively at a sampling interval of 25ms. The model assumes that the direction of vehicle travel is always the same direction in which the vehicle is facing. In other words the model assumes no slip between the tyres and the ground, a fairly good assumption at low speeds. A derivation of Equation 4.1 can be found in appendix A.

$$\begin{bmatrix} X(k+1) \\ Y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{pmatrix} X(k) + \Delta t\left(v_c \cos\phi - \frac{v_c}{L}\tan\alpha\left[a\sin\phi + b\cos\phi\right]\right) \\ Y(k) + \Delta t\left(v_c \sin\phi + \frac{v_c}{L}\tan\alpha\left[a\cos\phi - b\sin\phi\right]\right) \\ \phi(k) + \Delta t\frac{v_c}{L}\tan\alpha \end{pmatrix}$$

Equation 4.1

Where:

X is the 'x' position of the vehicle in the navigation frame

Y is the 'y' position of the vehicle in the navigation frame

ϕ is the heading of the vehicle in the navigation frame

α is the current steering angle of the vehicle

Δt is the sampling interval of the system (25ms in this thesis)

v_c is the velocity at the centre of the rear wheels

L is the wheelbase of the HSV

a is the distance between the rear axle and the GPS receiver

b is the distance between the centreline of the vehicle and the GPS receiver

The encoder is located at the rear left wheel of the vehicle and as a result the velocity information must be converted the centre of the rear wheel axle. The conversion from the rear left wheel to the centre of the axle is given in appendix A. Equation 4.1 gives the position estimate at the location of the GPS antenna in the local navigation frame. Typically the origin of this map is given as the first GPS reading taken; hence all position estimates from then onwards are taken relative to this point.

4.2 Kalman Filter Implementation

Making dead reckoning predictions about the vehicle position using the model described by Equation 4.1 is extremely inaccurate over long ranges due to errors in the model and noise on the sensor readings. Since each prediction of vehicle position is based on the previous estimate position error will grow without bound until the position data is essentially useless. To correct this problem the GPS data, which provides absolute

position information, will be fused with the model based predictions using the Kalman filtering techniques discussed in chapter 2, Statistical Estimation.

4.2.1 Predictions

Predictions occur at a rate of 25ms, which is the sampling rate of the encoders and LVDT. Predictions are made using Equation 4.1 with the state transition matrix used by the Kalman filter (**F**(k)) being that defined by Equation 4.1. After a prediction the covariance matrix is updated according to Equation 4.2.

$$\mathbf{P}(k \mid k-1) = \nabla\mathbf{F}_x(k)\mathbf{P}(k-1 \mid k-1)\nabla\mathbf{F}_x^T(k) + \mathbf{B}(k)\mathbf{Q}(k)\mathbf{B}^T(k)$$

Equation 4.2

Where $\nabla\mathbf{F}_x$, the Jacobian matrix of partial derivatives of **F** with respect to **X** $\mathbf{X}(k) = \begin{pmatrix} x(k) \\ y(k) \\ \phi(k) \end{pmatrix}$ and is given by, Equation 4.3.

$$\nabla\mathbf{F}_{x[i,j]} = \frac{\partial\mathbf{F}_{[i]}}{\partial\mathbf{X}_{[j]}}(\mathbf{X}(k-1), \mathbf{u}(k))$$

$$\nabla\mathbf{F}_x = \begin{bmatrix} 1 & 0 & \Delta t(-v_c \sin\phi + \frac{v_c}{L}\tan\alpha[a\cos\phi - b\sin\phi]) \\ 0 & 1 & \Delta t(v_c \cos\phi + \frac{v_c}{L}\tan\alpha[-a\sin\phi - b\cos\phi]) \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 4.3

The source error transfer matrix, $\mathbf{B}(k)$, is defined as the Jacobian of the state transition matrix with respect to the control input $\mathbf{u}(k) = \begin{pmatrix} v_c(k) \\ \alpha(k) \end{pmatrix}$ and is given by Equation 4.4.

$$\mathbf{B}_{[i,j]} = \frac{\partial \mathbf{F}_{[i]}}{\partial \mathbf{u}_{[j]}}(\mathbf{X}(k-1), \mathbf{u}(k))$$

$$B(k) = \Delta t \begin{pmatrix} \cos\phi - \frac{\tan\alpha}{L}(a\sin\phi + b\cos\phi) & -\frac{v_c}{L\cos^2\alpha}(a\sin\phi + b\cos\phi) \\ \sin\phi + \tan\alpha(a\cos\phi - b\sin\phi) & \frac{v_c}{L\cos^2\alpha}(a\cos\phi - b\sin\phi) \\ \frac{\tan\alpha}{L} & \frac{v_c}{L\cos^2\alpha} \end{pmatrix}$$

Equation 4.4

The matrix **Q** is a process noise covariance matrix based upon the perceived accuracy of the encoders and the LVDT. The selection of noise values for the encoders and LVDT is an important factor and greatly effects filter performance and is discussed in more detail in section 4.5 Filter Tuning as well as chapter 7, Results. The process noise matrix is given by Equation 4.5.

$$\mathbf{Q}(k) = \begin{pmatrix} \sigma_{vel}^{\ 2} & 0 \\ 0 & \sigma_{steer}^{\ 2} \end{pmatrix}$$

Equation 4.5

Where $\sigma_{vel}^{\ 2}$ and $\sigma_{steer}^{\ 2}$ represent the variance on velocity readings (*m/s*) taken from the encoders and the variance on steering angle readings *(rad)* taken from the LVDT.

4.2.2 Update

GPS information arrives at a rate of 5 Hz (every 200ms) and when this occurs the GPS information regarding absolute vehicle position is fused with the predicted vehicle position in what is known as the update, in the Kalman filter.

Since the observation model is a linear one the observation equations are those used in the linear Kalman filter section of chapter 2. Some of the relevant matrices from the equations of section 2.2 Kalman Filter as applied to thesis are given below.

Since only the first two states (X and Y) are observed the observation matrix is given as:

$$\mathbf{H}(k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Equation 4.6

The noise matrix ($\mathbf{R}(k)$) defined in equation 2.6 represents the accuracy of the GPS.

$$\mathbf{R}(k) = \begin{pmatrix} \sigma_{long}^2 & 0 \\ 0 & \sigma_{lat}^2 \end{pmatrix}$$

Equation 4.7

Internally to the GPS is a Kalman filter that calculates the latitude and longitude information based on the signals returned from the satellites. The basic general form of the covariance matrix of this Kalman filter internal to the GPS is shown below in Equation 4.8.

$$\mathbf{P} = \begin{pmatrix} \sigma_{long}^2 & \sigma_{LatLong}^2 \\ \sigma_{LongLat}^2 & \sigma_{lat}^2 \end{pmatrix}$$

Equation 4.8

The GPS makes available the diagonal entries σ_{long}^2 and σ_{lat}^2 as the variance of the GPS data. However it is rare that the latitude and longitude are uncorrelated meaning that the cross correlation terms $\sigma_{LongLat}^2$ and $\sigma_{LatLong}^2$ are non-zero. Consequently some of the uncertainty information is contained in these variables. Exact techniques exist to de-correlate the data however the cross correlation must be known. The consequence of this is that the variance of the latitude and longitude returned from the GPS is smaller than the actual variance and must be used with caution. One method, suggested by Juan Nieto, of de-correlating the variance data for use in the filter is to multiply the correlated variance values returned by the GPS by a constant factor and use these new values to tune the filter. This is illustrated by Equation 4.9.

$$\mathbf{R}(k) = K * \begin{pmatrix} \sigma_{long}^2 & 0 \\ 0 & \sigma_{lat}^2 \end{pmatrix}$$

Equation 4.9

Where: σ_{long}^2 and σ_{lat}^2 are the variance values returned by the GPS and K is a scaling constant to account for the cross correlation. In this thesis both using the variance data as s and using it after being scaled were tried and the results outlined in chapter 7, results.

Two methods of selecting the observation noise are available either a constant noise based upon some a priori knowledge of the environment or using the variance values returned from the GPS receiver. Both methods are tested in chapter 7 and the choice of which method to use depends on the testing environment and desired filter performance.

4.3 Multipath

The latitude and longitude information from the GPS is calculated based on the time of flight of the signal form the satellite to the receiver. Often the path of this signal will be obstructed or will rebound off many surfaces before reaching the receiver. As a consequence the GPS will return erroneous position information. This typically causes a large jump in the GPS position estimates or in some cases the GPS information to drop out completely. This phenomenon is known as multipath. [7]

Hence for the GPS to return accurate position information, the receiver must be in a wide-open space away from buildings and trees or other potential objects that may obscure or reflect the signal. Figure 4.2 shows some faulty GPS data taken in from the area next to the ACFR building at the university of Sydney where there are several buildings in the area and two large palm trees under which the vehicle drove. Notice the large and discontinuous jumps in the GPS data each time the vehicle passes under a tree.

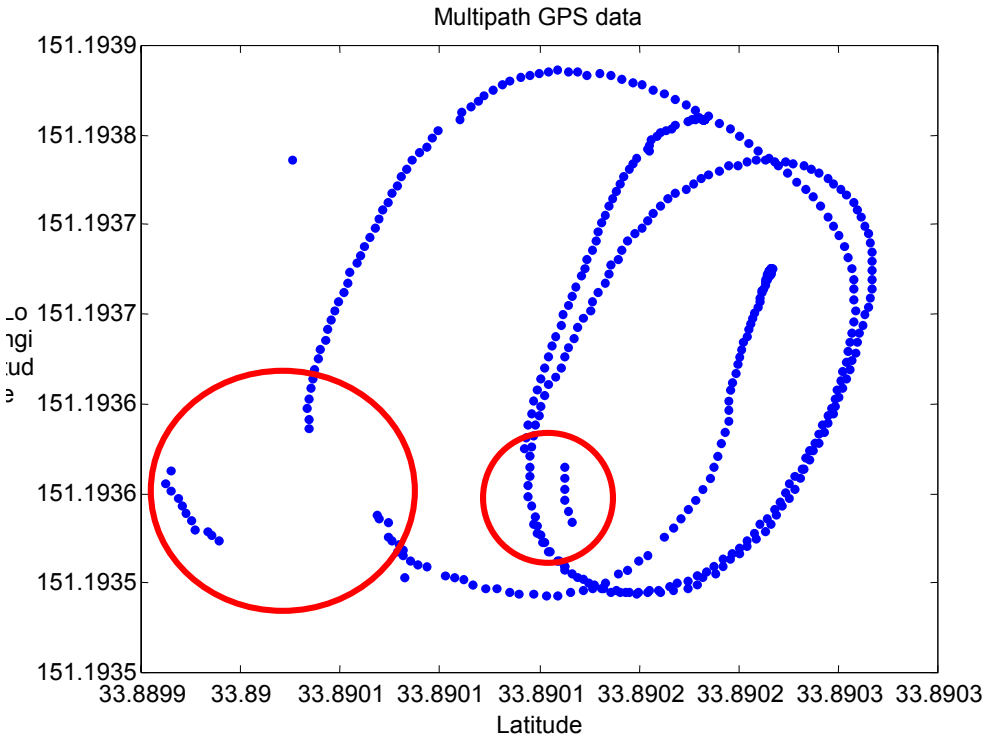


Figure 4.2: Faulty GPS data taken from the area next to the ACFR building, caused by the Ute passing under a tree and near a building.

4.4 Fault Detection

For accurate navigation the presence of multipath must be detected and when the GPS signal is deemed to inaccurate due to multipath the update stage should be skipped and only vehicle predictions made until the GPS information becomes reliable again.

The gate validation tests of section 2.4 Fault Detection can be used as a mechanism to detect multipath. The value of γ is computed at each update and if this value is greater than the threshold on 12.6 then the update is rejected.

4.5 Filter Tuning

The performance of the navigation filter is highly sensitive to the process noise matrix $\mathbf{Q}(k)$ and the observation noise matrix $\mathbf{R}(k)$. Hence the correct noise values must be chosen to ensure smooth filter operation.

The process noise **Q***(k)* is a measure of how well the model is trusted. A low process noise means that the model is perceived to be quite accurate and so upon observation the predicted position is trusted more than the observed position. Figure 4.3 shows the filtered position with a low process noise. The blue indicates the filtered position and the red the observed GPS position. Notice that since the process noise is low the model is trusted quite a lot and so the filtered position lies away from the GPS observations.

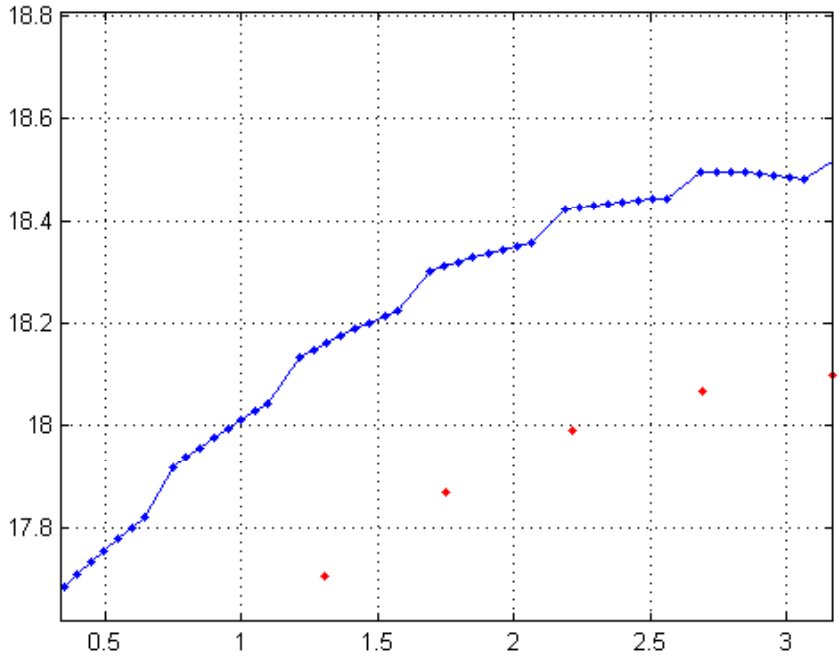


Figure 4.3: Low process noise. The blue indicates the filtered position and the red indicates the observed GPS position.

Figure 4.4 shows the filtered position with a higher process noise. The observation noise was kept constant. Notice how the filtered position lies much closer to the observations due to the fact that the uncertainty in the model is increased and so the predictions are not trusted as much as they were in Figure 4.3.

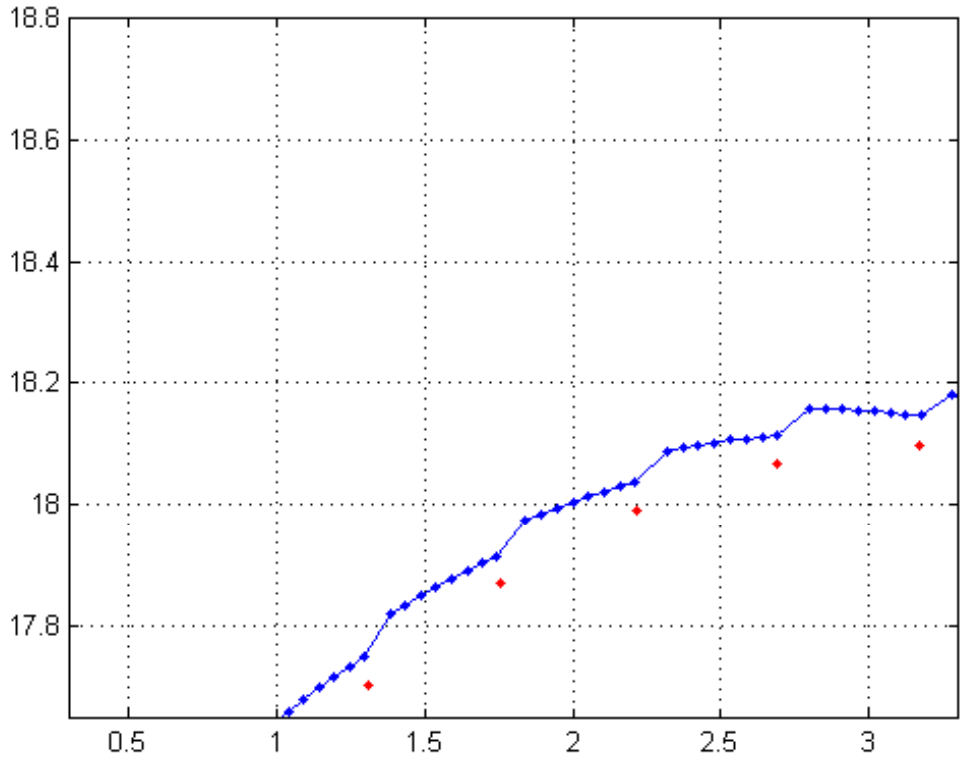


Figure 4.4: High process noise. The blue indicates the filtered position and the red the observed GPS position

Just as the process noise is a measure of how the model is trusted the observation noise **R***(k)* is a measure of how well the observations are trusted.

Figure 4.5 shows the filter operating with a very low observation noise. Notice how the position estimates fall virtually exactly on top of the GPS observations. This is because the GPS observations are interpreted to be extremely accurate.

Figure 4.6 shows the filter operating when the observation noise is very high. The position estimates lay away from the GPS observations since the accuracy of the observations is deemed to be low by the observation noise.

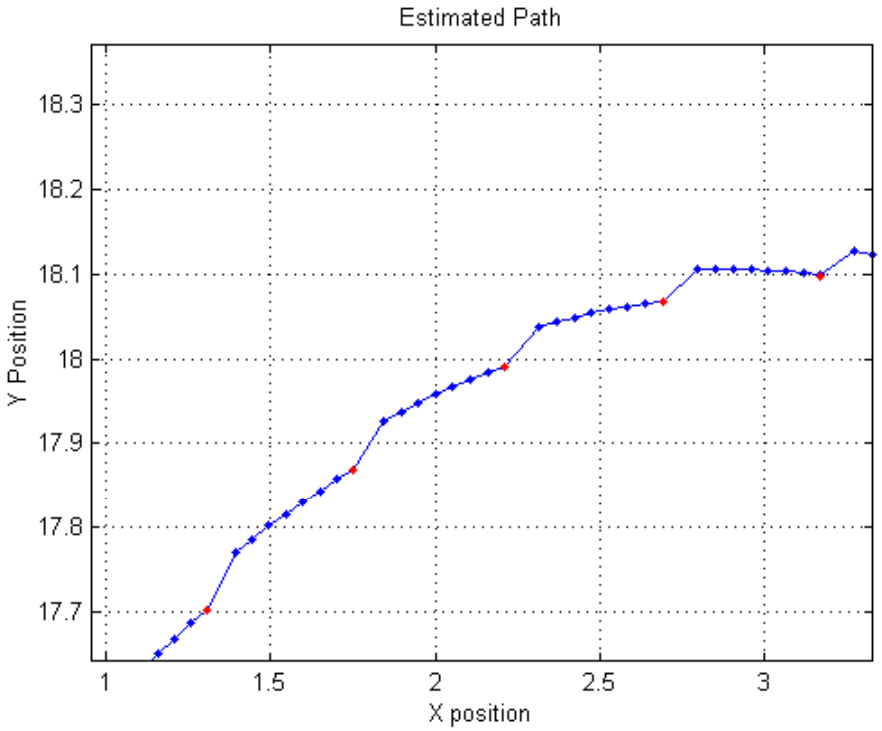


Figure 4.5: Low observation noise. Observed GPS data in red and filtered path estimate in blue.

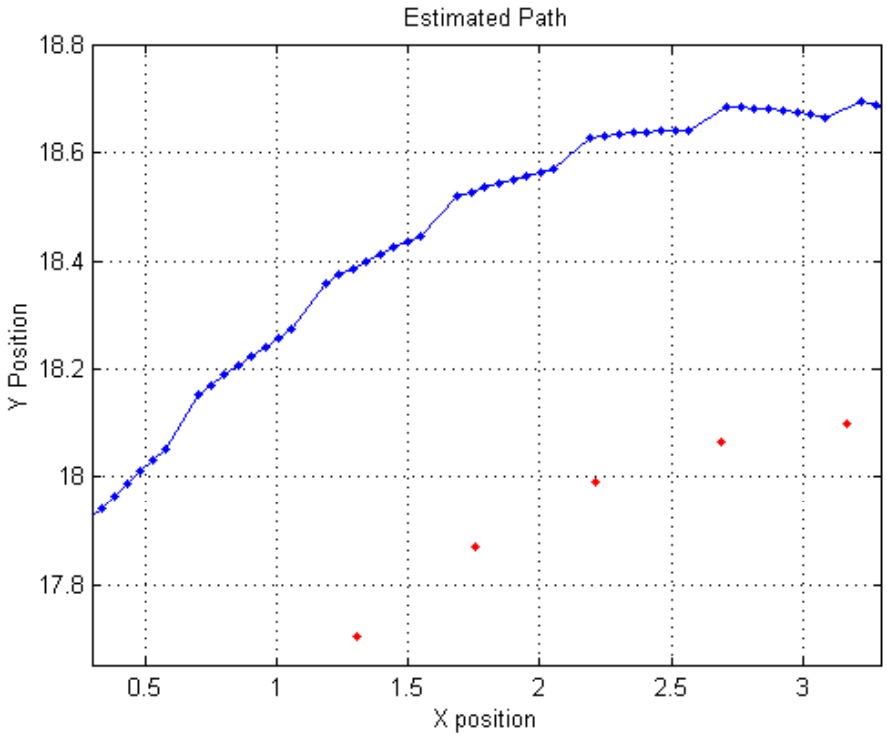


Figure 4.6: Large observation noise. Estimated path in blue, observations in red.

Clearly increasing the process noise has the same effect on filter performance as decreasing the observation noise and similarly decreasing the process noise will have the same effect as increasing the observation noise.

Another factor that must be considered in filter tuning is what happens to the gate validation parameter (γ) and hence the filters ability to detect and recover from the occurrence of multipath.

If the observation noise is too low and the process noise is also too low the filter will never recover once multipath is detected. Figure 4.7 shows the filter, with incorrect tuning, rejecting an update due to multipath and then rejecting all updates thereafter when clearly multipath has never occurred. Great care had to be taken in filter tuning to ensure that the observation noise was not set too low to trigger the gate validation tests erroneously.

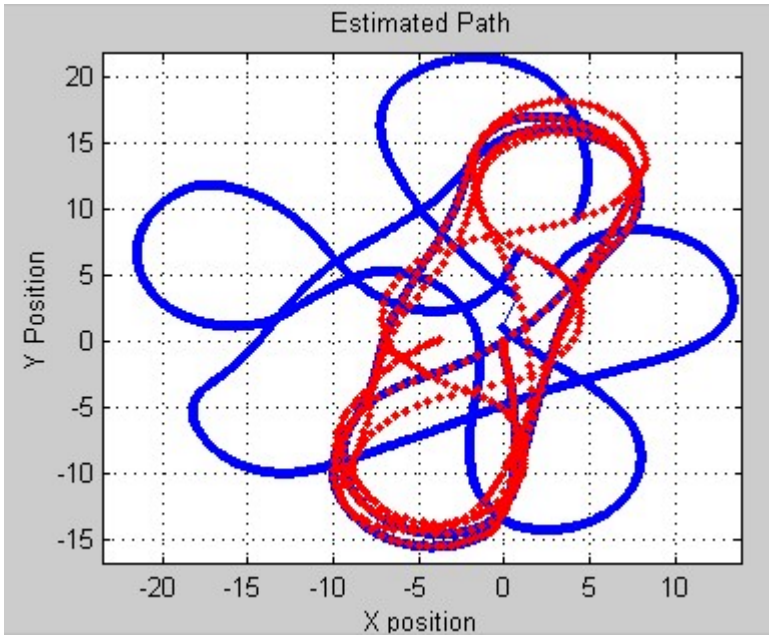


Figure 4.7: Filter never recovering form detection of multipath

Conversely if noise levels are too high in the observation noise matrix, the filter will never detect multipath and accept all observations. Figure 4.8 shows the filter failing to detect the presence of multipath in the GPS signal due to incorrect filter tuning. This posed less of a problem than the gate validation test detecting errors when there was none

since the filter generally still behaved acceptably and did not become unstable such as the case where the filter rejects all updates (Figure 4.7).

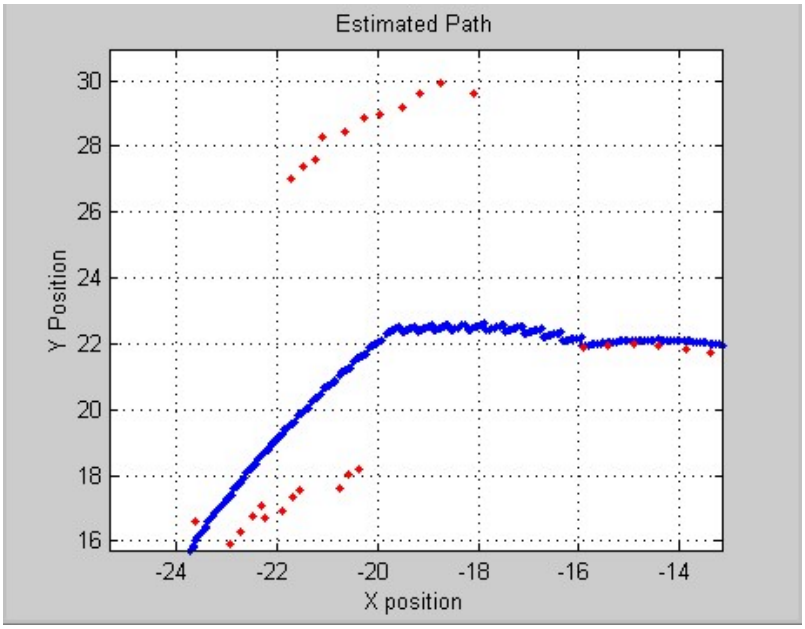


Figure 4.8: Incorrect filter tuning failing to recognise multipath effects

4.6 Initialisation

It is very important to initialise the filter correctly. For example the initial heading may be wrong causing the vehicle to move off in the wrong direction initially. If the filter is tuned incorrectly it may never recover and become unstable returning essentially useless position data. One method to correct this is to use a compass to find the initial heading and then not accept any updates until the vehicle has started moving. A second method is to set the initial covariance on the states (x,y,ϕ) to be extremely large, of the order of 10km and 360° for position and heading respectively. Then once the vehicle starts moving and a GPS update arrives the GPS information is trusted effectively 100% and the filter will begin moving at least for a while right along the GPS path. As a consequence it is vitally important to begin the filter in an area where the GPS data is reliable.

4.7 Limitations

The navigation filter presented in this chapter has several limitations associated with it. Primarily the filter is 2-D filter and as such will not function effectively in hilly or 3-D environments. Other methods such as inertial systems or slam must be used in 3-D environments.

The filter is also very reliant on reliable GPS information so can only be used in reasonably open environments. Although the filter can detect and recover from multipath in the GPS, extended periods of GPS information being unavailable will result in the position estimates being extremely inaccurate due to the problems in the vehicle model.

Also the module must have reliable GPS information to begin the filtering process otherwise the origin of the map will be wrong and the navigation loop will suffer as a result.

4.8 Conclusion

This chapter presents a 2D solution to the localisation problem utilising information about vehicle velocity and steering angles to predict the vehicle position and then fusing these estimates with absolute GPS position information. Reliability is guaranteed using the gate validation to reject multipath on the GPS signals providing a more robust navigation loop.

Chapter 5:

Laser Dead Reckoning

To date on the HSV project the 2-D range and bearing laser mounted on the Ute (see chapter 1) has been used for a number of different applications including identification of beacons in SLAM, 2-D and 3-D mapping of an environment and for obstacle detection in a reactive control module. However an interesting potential application for a 2-D range and bearing laser would be to determine vehicle pose changes from consecutive laser scans thus providing a dead reckoning path estimate of the vehicle or an estimate of vehicle velocity.

This chapter investigates several methods proposed for vehicle pose estimation based on consecutive laser scans. Firstly the Iterative Closed Point method and its variants are analysed. Following this various methods of feature extraction are discussed along with methods of feature matching between scans. Finally once features between consecutive scans are matched the vehicle pose change can be solved for.

5.1 Iterative Closest Point and Variations

The Iterative Closest Point (ICP) algorithm, described in greater detail in Besl & Mackay [3] (1992) and Lu & Milios (1994) [12], is a method of comparing two consecutive laser scans on a point-by-point basis to determine the pose transformation of the 2nd scan relative to the first. The pose transformation of the 2nd scan relative to the 1st is equivalent

to the pose change of the vehicle. The pose change is defined as a translation in X and Y (Tx, Ty) and a rotation (ω).

For each point in the second scan the closest point in the first scan is found and these points are matched together. Hence every point in the second scan will have a corresponding point in the first scan. Using all these corresponding point pairs the least squares problem is solved to produce a pose transformation of the 2nd scan relative to the first. The data from the 2nd scan is then transposed to be relative to the first and the process repeated until the transformation between scans converges to zero. The equations to solve the least squares problem once points have been matched are given in Appendix B (Equations from Lu and Milios (1994)).

Lu and Milios (1994) found that the closest point rule for matching points proved to be very effective in detecting translational movement but not so good at detecting rotational movement. So a second method for matching points between consecutive scans was developed known as the Iterative Matching Range Point method (IMRP). This rule groups points that are closest together in range. Using the iterative range point rule proved to be effective in identifying rotational movement but not so effective in detecting translational movement between scans.

To correct the problems of the two methods Lu and Milios (1994) developed a combined method known as Iterative Dual Correspondence (IDC). This method took the translation component of the ICP method and the rotational component of the IMRP method and was found to converge much faster than both methods individually.

Test were conducted on the ICP methods and their variants to see if the methods could be used to directly determine the pose change of the HSV on a point by point basis. It was found that the ICP methods performed well if all the points present in the 2nd scan are present in the 1st scan. Figure 5.1 shows the ICP method working well. Notice that the amount of points and features in the first and second scans (blue and red) is practically identical so each point in the second scan can be easily matched to the correct point in the previous scan.

Figure 5.2 shows the ICP method breaking down spectacularly due to the introduction of a corner feature in the second scan (red) not present in the first scan (blue). It is clear to the human eye that all points in the added line from the 2nd scan should not be matched to points in the 1st scan. However the ICP algorithm assumes all points present in the second are present in the first scan in some form or another and the ICP algorithm erroneously matches those point in the 2nd scan that don't belong to the 1st scan. Hence due to erroneous matching of points the ICP method is not a robust method at all for determining pose change from consecutive laser scans.

The ICP method is also very computationally expensive since it is a point-by-point based method and so a point pair must be found for every point in the 2nd scan. The ICP algorithm generally takes between 15-20 iterations before it converges (Lu and Milios) and this coupled with the fact the fact that it is a point-by-point method makes the ICP algorithm an extremely slow algorithm. Given that the ICP algorithm is not particularly robust in outdoor environments and that the algorithm is extremely slow, the ICP algorithm was rejected in this thesis as a method for matching consecutive scans.

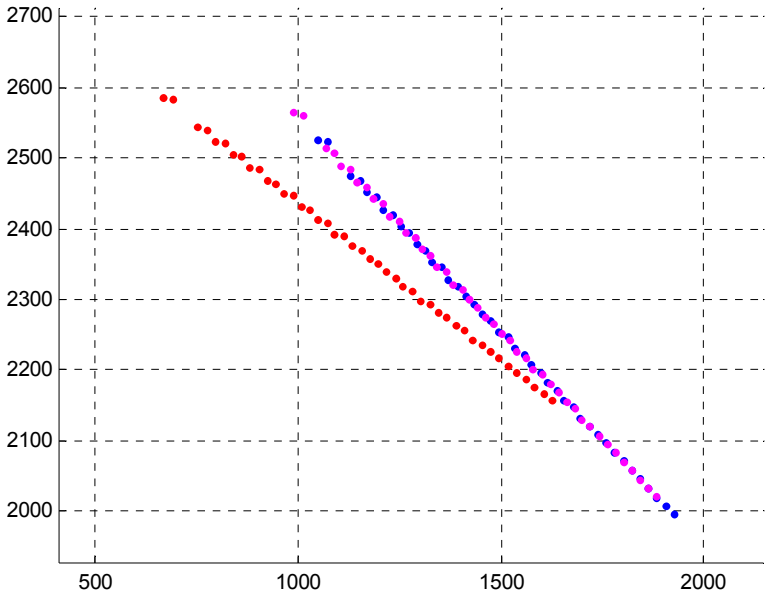


Figure 5.1: Shows the ICP method working well when both scans contain the same features. The blue scan is the first (reference) scan and the red shows the second scan. The purple represents the second scan moved to the same frame of reference as the first scan.

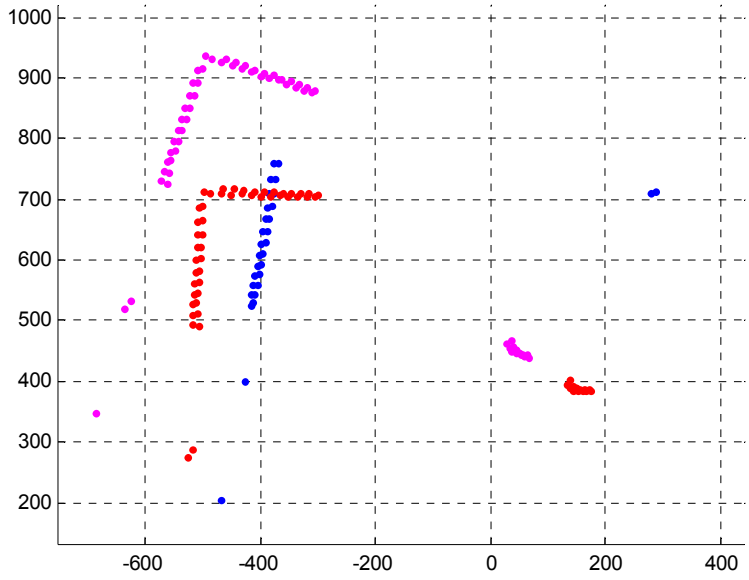


Figure 5.2: Shows the ICP method breaking down as new features and points are introduced into the second scan. The blue represents the first scan, the red the second and the purple the second scan transformed to be in the same reference frame as the first scan.

5.2 Feature Extraction

Since ICP at best provided a less than reliable method of determining pose transformation between consecutive scans, a feature extraction method was proposed to attempt to extract features from any given laser scan and then track the different features between scans. Based on the movement of features the pose change of the vehicle could be determined.

As proposed in Lee (2001), it was decided that the best features to be extracted were lines and circles. Hence the surfaces should be represented as a line or a series of lines and the trees and poles as circles.

5.2.1 Clustering

To correctly identify lines and circles from a laser scan it is first necessary to group the points into clusters with each cluster typically representing a single obstacle. The clustering techniques developed in Lee 2001 were used. Note that a single object or

cluster can be represented by more than one line or circle. Recall also from chapter 1 that the data returned from the laser is the form of an ordered array of increasing bearing angles, with each element in the array representing the distance to the nearest object.

To form a cluster of points each point in the laser scan is compared with the previous point in the scan and if the distance between the points is less than some threshold then the points are assumed to part of the same object or cluster. If the point is too far away from the previous point then a new cluster is started. Situations can arise where a point in the laser scan fails to return a value due to a reflection of an uneven surface and when this occurs the range reading is set to 8000, the maximum range of the laser. An example of this is shown in Figure 5.3. If the current point is compared only to he previous points it is clear that for the case where a pulse is lost a new cluster will be formed erroneously. So to correct this problem the current point is compared to points that are less than three degrees (6 bearing angles) away. Figure 5.4 shows the results of the clustering process with each cluster shown in a different colour.

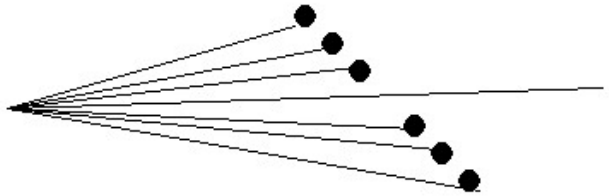


Figure 5.3: Showing the effects of how some laser pulses fail to return due to the pulse being lost

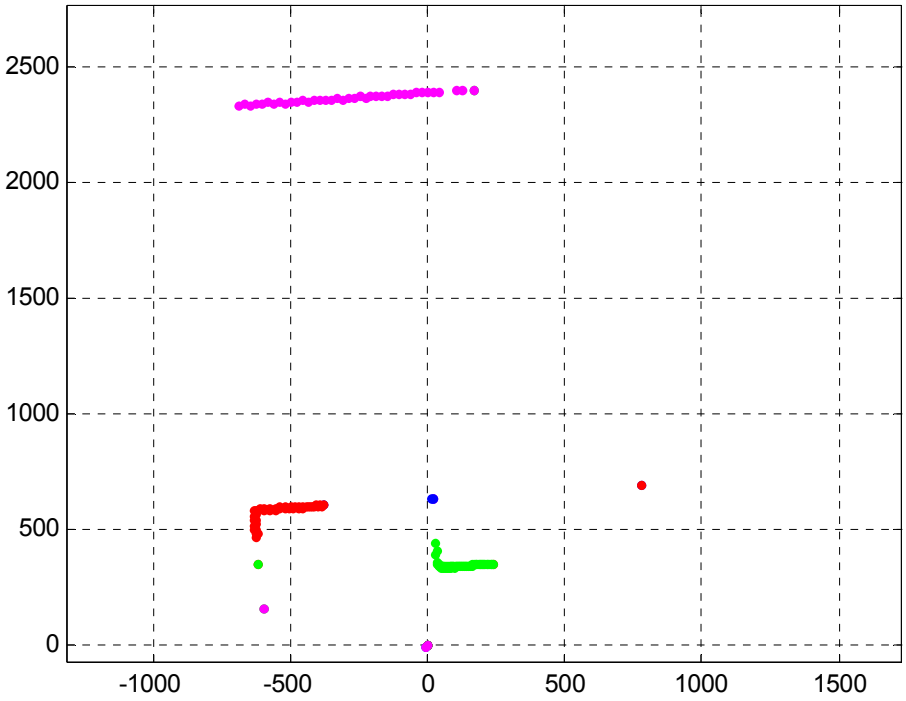


Figure 5.4: Example of successful clustering. Different clusters are shown in different colours. Note in the large purple cluster the three points to the right (separated by a small gap from the rest of the points) are still included in the cluster.

5.2.2 Classification of Objects

After clustering each cluster must be checked to see whether it is more likely to represent a line or a circle.

Firstly the distance between cluster endpoints is checked. If this distance is deemed to small it is decided that any line fitted to the cluster will be too short and so the object most likely represents a circle.

The second check involves checking the number of points in the cluster. If the number of points in the cluster is high it is very likely that the cluster will represent a line or series of lines and if the number of points is low then the cluster most likely represents a circle.

Problems may occur with small lines since they may be classified as circles, when really they should be very short lines. So for any short objects, with four or more points a line is

fitted and if the quality of this line is very high (co-efficient of determination discussed in next section was used), then a short line is fitted in place of a circle.

5.2.3 Line Detection

Fitting a line to a cluster of points provides more meaningful information about the object and reduces the amount of data required to represent the obstacle. Lines were decided to be a good object to fit to the raw laser data since the laser can only return the distance to the nearest object and so, many surfaces, such as walls and cars, can easily be represented by lines.

The method used to fit lines to a cluster of points is a statistical least squares method aimed at reducing the square of the error of the distance of the points to the line. Problems arise when the cluster represents more than one line as can be seen in Figure 5.4 where two of the clusters (the red and green) represent corners. Naturally when this situation occurs multiple lines must be fitted to the cluster since one line will not accurately describe the obstacles in the environment. This section of the report outline in greater detail the statistical methods used for line fitting as well as the methods used to extract multiple lines for a single cluster.

5.2.3.1 Line representation

Several methods exist for representing a line in Cartesian space. A line could be represented using the equation of the line ($y=mx+b$ or $ax+by+c=0$), the line could be represented by its end points or the line could be represented in a polar form by a distance to the origin r and an angle θ. The polar format shown in Figure 5.5 is the one used in this thesis since the line can be represented by two only parameters R and θ. Also the polar format is convenient when performing calculations on pose transformation at latter stage. The start and end points of the line are also stored mainly for graphical display purposes.

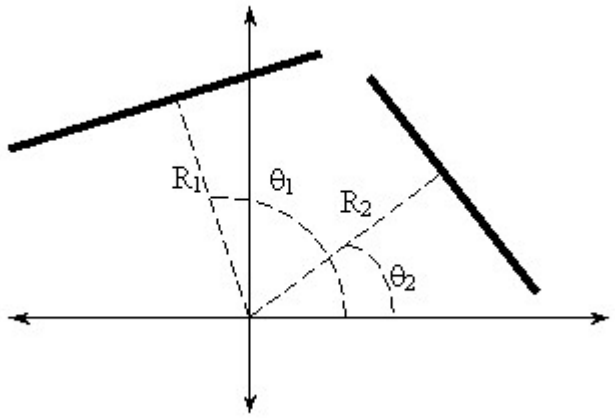


Figure 5.5: Polar representation of a line

5.2.3.2 Least squares regression parameters

Least squares regression or linear regression is the method used in this thesis to find the 'line of best fit' for a given number of points. The line of best fit is chosen in such a way as to minimise the square of the error of the points from the line. Linear regression is a well-documented technique for fitting lines to data sets and is commonly used in the fields of economics, finance and science. The following section describes the equations used to generate the least squares line model and it's application to this thesis. (For more information on linear regression consult any standard statistics book or more specifically [18].

Given a group of '*n*' points (the cluster in our case) where each point has two components an *x* component and a *y* component a least squares line can be formed through the points. The equation of this line takes the standard y-intercept form of a line shown in Equation 5.1. Depending on whether a regression is performed from *y* to *x* or from *x* to *y* the top or bottom equation is used.

$$y = \beta_1 x + \beta_0$$
$$or$$
$$x = \beta_1 y + \beta_0$$

Equation 5.1

where:

β_1 is the slope of the line

β_0 is the intercept of the line

The slope of the line (β_1) and the intercept (β_0) are calculated according to the equations below.

Regression *y* to *x*	**Regression *x* to *y***
$(y = \beta_1 x + \beta_0)$	$(x = \beta_1 y + \beta_0)$
$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$	$\beta_1 = \frac{SS_{xy}}{SS_{yy}}$
$\beta_0 = \bar{y} - \beta_1 \bar{x}$	$\beta_0 = \bar{x} - \beta_1 \bar{y}$

Where:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum (x_i)^2 - \frac{\sum x_i * \sum x_i}{n}$$

$$SS_{yy} = \sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum (y_i)^2 - \frac{\sum y_i * \sum y_i}{n}$$

$$SS_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - \frac{\sum x_i * \sum y_i}{n}$$

The polar parameters (r, θ) can be calculated from these regression parameters (β_1, β_2) using simple trigonometry. For a more detailed derivation see appendix C.

$$x = -\frac{\beta_1 \beta_0}{1 + \beta_1^2}$$
$$y = \beta_1 x + \beta_0 \qquad \text{where atan2 is the 4 quadrant arctangent.}$$
$$\theta = \text{atan2}\ (y, x)$$
$$r = \beta_0 \cos(\theta)$$

The above equation is for regression from y to x. For regression x to y simply swap parameters x and y.

The test to determine whether to perform a regression from y to x or vice versa is based on whether the line is more vertical or more horizontal. To determine this the values of *SSxx* and *SSyy* are compared. If *SSxx* is greater than *SSyy* the line is more horizontal and a regression from y to x is performed as per the equations on the left hand side above. Otherwise a regression from x to y is performed.

The above linear regression equations provide no indication of the goodness of fit of a line, in other words do the data points make a good line. To determine whether a line fits the data points well we use two additional parameters about the line, the variance and the coefficient of determination.

The coefficient of determination (R^2) is a measure of how well a variation in the independent variable can be used to predict the variation in the dependant variable. In layman's terms the coefficient of determination is simply a measure of the linearity of the points. The coefficient of determination always lies between 0 and 1 with 0 meaning that there is no linear correlation in the data set and 1 being a perfect line fit. Figure 5.6 shows typical values of the coefficient of determination for three different data sets.

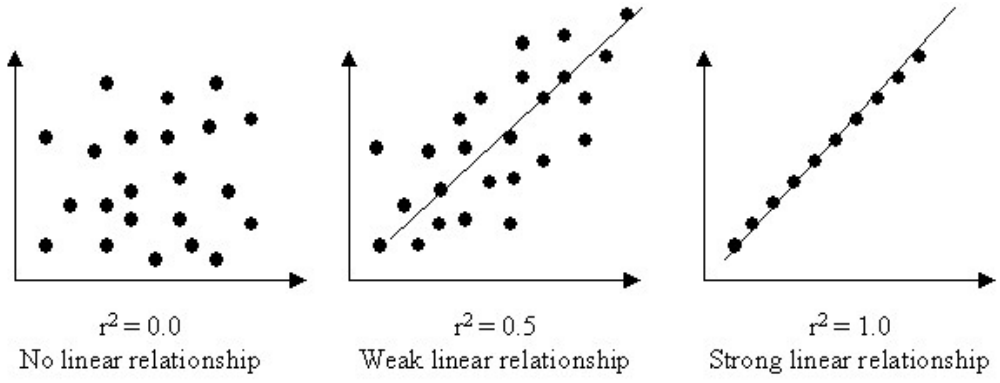


Figure 5.6: Typical values for the coefficient of determination (R^2)

The coefficient of determination is calculated as follows:

Regression *y* to *x*	**Regression *x* to *y***
$(y = \beta_1 x + \beta_0)$	$(x = \beta_1 y + \beta_0)$
$R^2 = 1 - \frac{SSE}{SS_{yy}}$	$R^2 = 1 - \frac{SSE}{SS_{xx}}$
where:	where:
$SSE = SS_{yy} - \beta_1 SS_{xy}$	$SSE = SS_{xx} - \beta_1 SS_{xy}$
is the sum of the squared errors	is the sum of the squared errors

One final piece of useful information that can be obtained from the regression information is the variance of the points about the line (s^2). Given from standard statistics that 90% of points lie within two standard deviations of the line and given that the variance is equal to the standard deviation squared, the variance gives a measure of how close to the line the points lie. For a regression of *y* to *x* the variance is based on the vertical distance between the points and the line and for an *x* to *y* regression the variance is based upon the horizontal distance between the points and the line as shown in Figure 5.7.

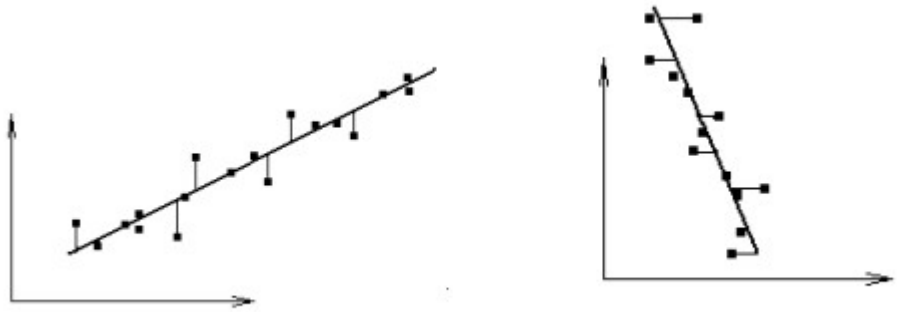


Figure 5.7: The variation in points from the line. On the left is the variation in *y* caused by a *y* to *x* regression and on the right is the variation in *x* caused by an *x* to *y* regression.

The variance is calculated according to the equation below.

$$s^2 = \frac{SSE}{n-2}$$

5.2.3.3 Coefficient of Determination Testing

Testing was done on the coefficient of determination (R^2), defined in the section Least squares regression parameters, to see if this information was useful in extracting lines from the cluster. Given that the coefficient of determination is a measure of the linearity of the points (Figure 5.6), it was postulated that the coefficient of determination could be used as a basis for accepting lines or for rejecting them.

To test if the coefficient of determination was useful in extracting line information from a cluster, a line was firstly fitted to the entire cluster and the coefficient of determination computed for this line. If the value of R^2 was less than a certain threshold, the line was accepted and added to the list of lines for that laser scan. If the value of R^2 was less than the threshold, the cluster was bisected according to the corner method discussed in section 5.2.3.5 Corner Method, and lines continually fitted until a suitable value of R^2 was reached.

It was found that the coefficient of determination was only a useful parameter if the laser points tested formed an almost perfect line. Figure 5.8 shows an example of a laser scan taken from a flat surface where all the points lie nearly perfectly on the line. The

coefficient of correlation for this example was 0.9995 indicating the excellent linear relationship between the points.

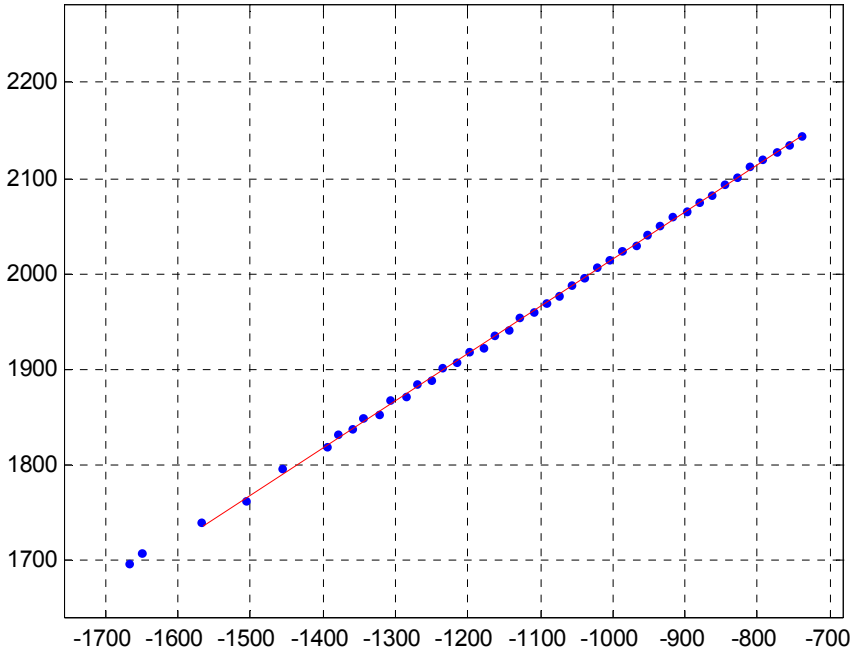


Figure 5.8: Laser points in a near perfect straight line, demonstrating a coefficient of correlation of R^2 = 0.9995

Although the R^2 test works well in the case where the data points spell out a near perfect line and a large portion of the points lie on the line, the test often fails to identify lines that clearly should be identified. Figure 5.9 shows a laser scan typical of a rough surface or perhaps a window. Clearly a linear relationship exists between the points in the middle section of the cluster, however the R^2 test fails to pick this up. To the human eye it can be seen clearly that it would be possible to fit two lines almost exactly through all the data points and that a line of best fit would be somewhere between these two lines passing through only a few of the points. For this test the threshold for R^2 was set to 0.5, quite a low value, and the linear relationship between the points was deemed to be too small for a line to be fitted.

In general using the coefficient of determination proved to be an unreliable method in detecting lines in that points which should be extracted, as lines were often not. However

whenever the coefficient of determination was high, it was sure that the points spelt out and excellent line.

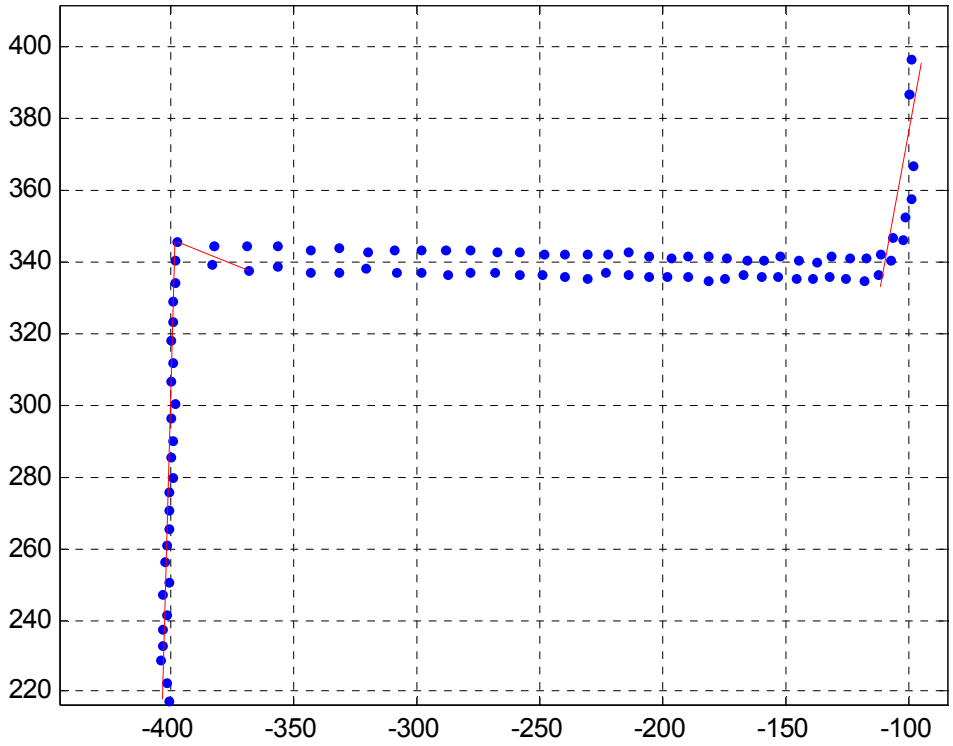


Figure 5.9: Scan of points showing where the R^2 tests break down across the middle section (R^2 threshold was set to 0.5 for this test).

5.2.3.4 Variance Testing

Given the failure of the coefficient of determination to accurately identify lines it was proposed the variance of the points about the line be used as a measure of whether a line fits the data points well.

Given that 90% of the points lie within 2 standard deviations of the line and that the variance is equal to the standard deviation squared, it was assumed that if the value of the variance less than a certain threshold (i.e. all the points are close to the line), then a good line can be formed from the specified points.

Figure 5.10 shows the variance method for detecting lines correcting the problems associated with the R^2 tests. The variance threshold was set to 15cm for this particular test.

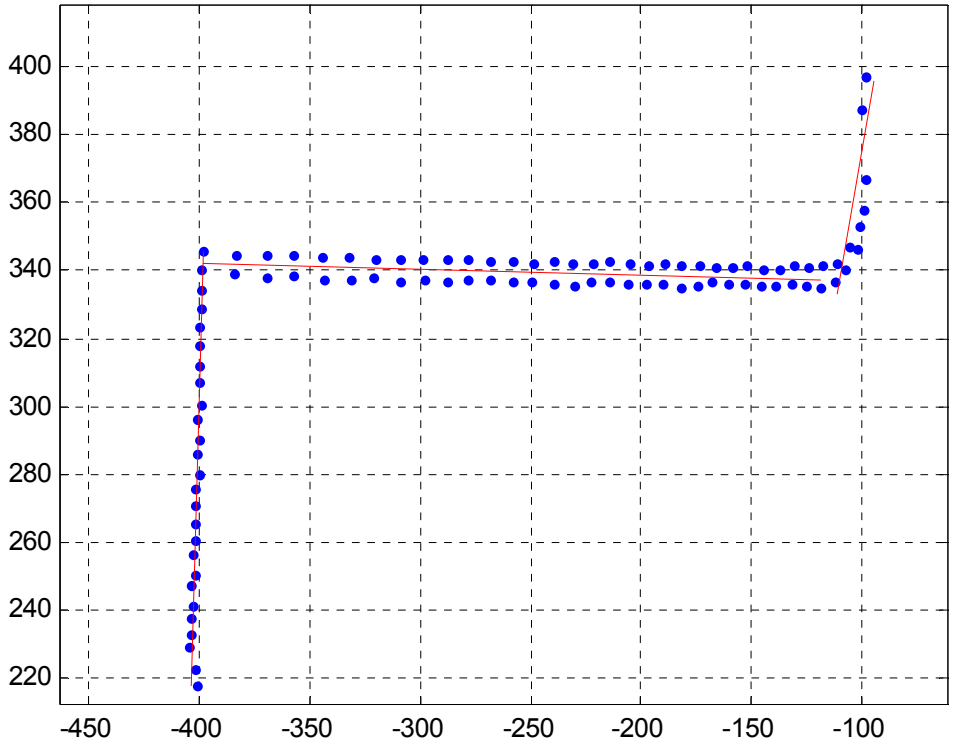


Figure 5.10: Showing the variance method correcting the problems associated with the R^2 test.

5.2.3.5 Corner Method

A large proportion of the time the points that make up a cluster define more than one line as shown in Figure 5.4. Obviously attempting to fit a line to the entire cluster group will fail and so a method must be chosen to break up the cluster into smaller groups. The thesis of Lee (2002) proposed a bisection and trisection method, whereby the points are simply trisected or bisected evenly and lines fitted to each section. If the gradient of the lines match up a line is fitted to the whole group, if not each section is further divided. This thesis uses a similar method however the method by which the cluster of points in bisected is based upon the most probable location of a corner.

Firstly a least squares line is fitted to all the points in the cluster and if both the coefficient of determination and variance criteria are not satisfied then the cluster of points is bisected. To bisect the points the two endpoints in the cluster are joined to form a line. Then the perpendicular distance from the line to each point in the cluster (excluding the end points) is computed and the point that is farthest from the line is taken to represent a corner. The points are then bisected using the corner point as the split point. A line is then fitted to each of the bisected segments and if this line satisfies the coefficient of determination and variance tests, then the line is accepted otherwise the line is further bisected until a suitable line is found.

Figure 5.10 shows the corner bisection method working well for the case of two fairly well defined corners.

5.2.3.6 Summary of Line fitting

The final method used to identify lines is a combination of the variance and coefficient of determination methods. Basically once a cluster has been identified as a line, a line is fitted to he cluster and the value of both the variance and the coefficient of determination are checked, and if either of these is satisfactory the line is accepted. If not the line is recursively bisected according to the corner rule until either a satisfactory line can be fitted or there are too few points in the bisection to form a line. If after much bisection suitable lines cannot be fitted then no line is fitted and that feature is ignored (Figure 5.11). Ignoring features is safe in this application since we are using the features for navigation purposes in an attempt to determine vehicle pose. However if the feature extraction were to be used for mapping purpose to determine regions where the vehicle cannot travel, then lines would need to be fitted to all data however good or bad.

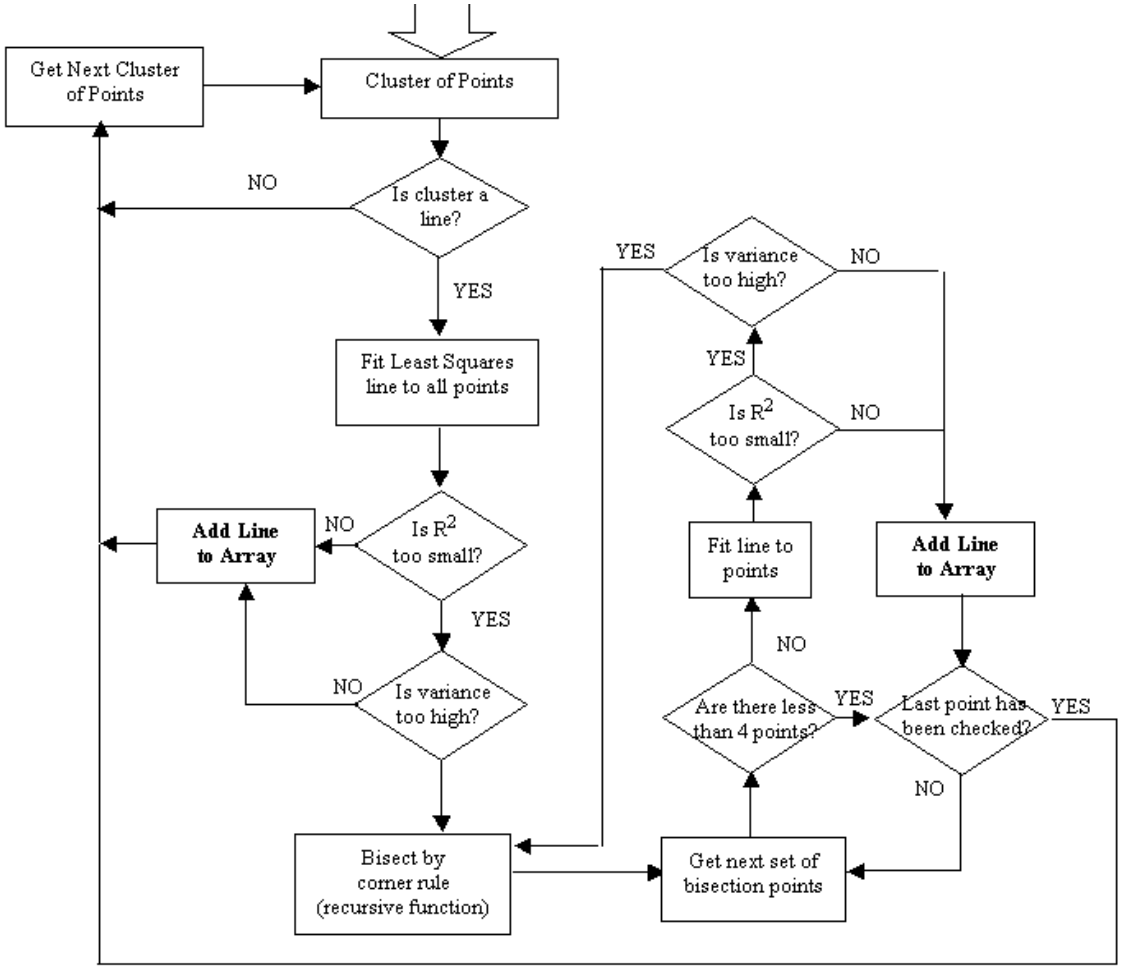


Figure 5.11: Line fitting flow chart

5.2.4 Circle Extraction

The method used to fit circles was based on the method used in the thesis of Lee 2001 developed by Jose Guivant and is detailed in the following section.

Given that the laser can only return the closest point in a surface a reflection off a circular object such as a tree or pole will, in an ideal case, return a semi circle representing the front half of the object (Figure 5.12). Given this information the diameter of the tree can be estimated by simple trigonometry as shown by Figure 5.12.

$$D = \Delta\theta * R$$

Where $\Delta\theta$ is the angle subtended by the cluster and R is the average range of the points in the cluster.

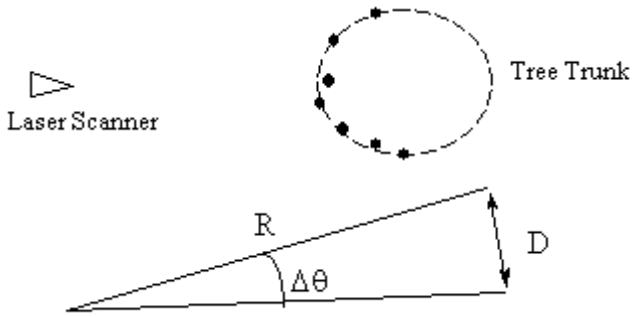


Figure 5.12: Laser return from a typical circular object, and showing how to estimate tree diameter [7]

Guivant proposed several different methods for finding the centre of the circle. Firstly it was assumed that the centre of the circle was collinear with the centreline of the obstacles bearing angles, Figure 5.13. The location of the centre could then be determined in one of two ways:

- The distance from the point nearest the laser to the centre is equal to the radius (Figure 5.14).

- The distance from the average range to the centre is equal to half the radius (Figure 5.14).

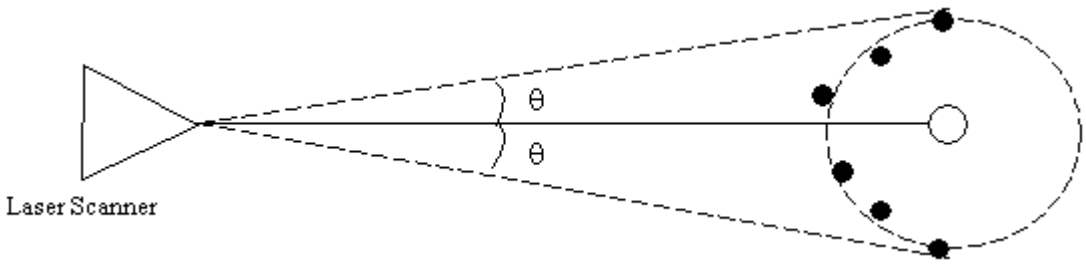


Figure 5.13: Circle Centre on centreline of bearing angles[7]

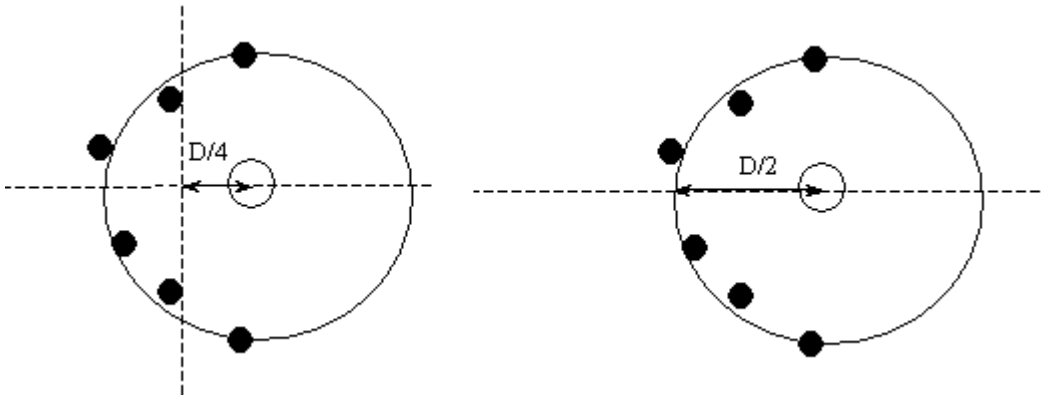


Figure 5.14: Shows the average distance to sensor method for estimating circle centre on left and closest point method on right. [7]

Based on the results shown in Lee 2001 [7] the closest range method was used for finding the centre of circles. Some results of circles extracted from typical laser data representing trees taken from Victoria Park are shown in Figure 5.15.

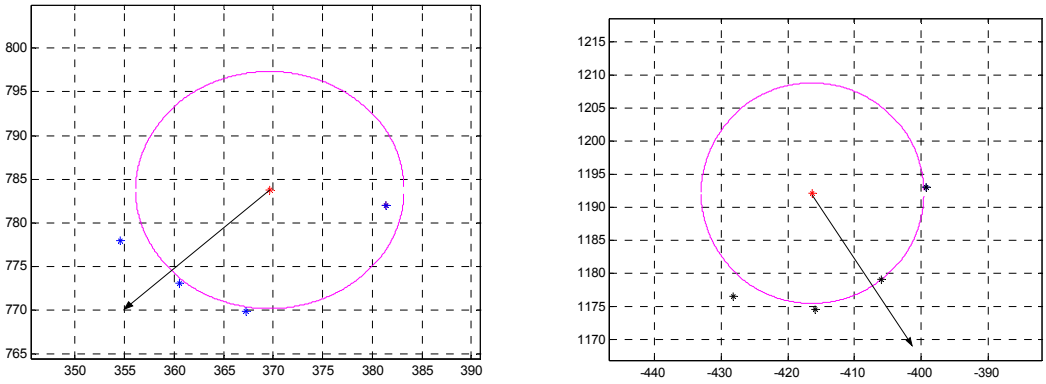


Figure 5.15: Typical circles extracted using Guivants method

Problems with Guivants method arise when the number of points in the cluster is one or two. When there is only one point in the cluster the circle fitted has a centre at this point and a radius of 5cm, the accuracy of the laser. In the case of two points in the cluster the diameter of the circle is defined in the same way but the centre of the circle is defined to be the centroid of the two points. Figure show circles extracted with only one or two points.

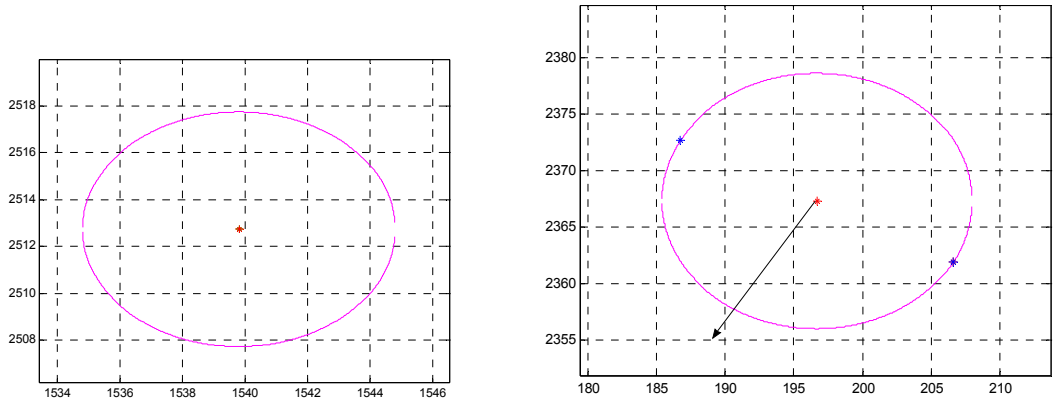


Figure 5.16: Circles extracted from laser scans with 1 or 2 points

5.3 Data Association

After features have been extracted from the lasers scans these features must be associated with features in the previous scan. Correct feature matching between scans is crucial to the laser based dead reckoning since in correct feature mappings will result in erroneous pose change estimates of the vehicle. Hence a reliable and robust method must be implemented to correctly map features from consecutive laser scans.

The thesis of Lee 2001 [11] proposed several methods for data association between scans that makes no use of the geometric relationship between scans and simply tries to match each line to the line in the previous scan that is a closest match to the line in question and tries to match each circle to the circle with a similar radius and is closest to the circle in question. This method proved to be very unreliable since often features would be present in one scan and not the second and so a false mapping would result. Also it became difficult to match features on a feature-by-feature basis in more cluttered environments.

An alternate approach used in Bailey [1] makes use of the geometric relationship between features and hence provides a much more robust algorithm for data association. This method is known as the graph theoretic approach.

5.3.1 Graph Theoretic Approach

The graph theoretic approach to the data association problem essentially consists of creating two graphs representing the features from current and previous scans and solving the maximum common sub-graph problem for these two graphs to provide a one to one mapping of features between consecutive scans. This section outlines the essentials of the graph theoretic approach to the data association problem as presented in Bailey [1].

5.3.1.1 Feature Graph Generation

The data from the feature extraction process can be represented as a graph where each node in the graph represents a particular feature (line or circle) and each edge is the relationship between two features. Edges must be defined for all pairs of features or nodes and as a result the feature graph generated will have every node connected to every other node. Logically there will be two different node types in the feature graph, lines and circles. Therefore edges must be defined as follows: if the edge connects two lines the edge represents the acute angle difference between the two lines, if the edge connects a circle and a line the edge represents the perpendicular distance between the centre of the circle and the line and for two circles the edge represents the distance between circle centres. Figure 5.17 and Figure 5.18 show both the geometric relationship between features and the corresponding feature graph representation for a trivial case of 2 lines and three circles.

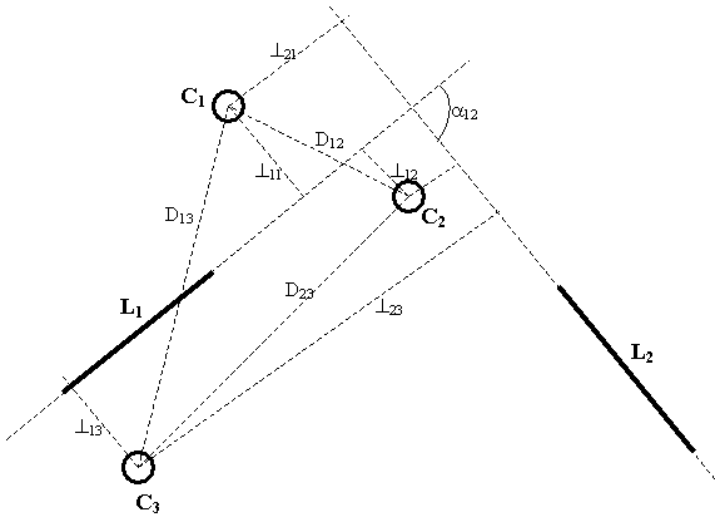


Figure 5.17: Geometric relationships between features

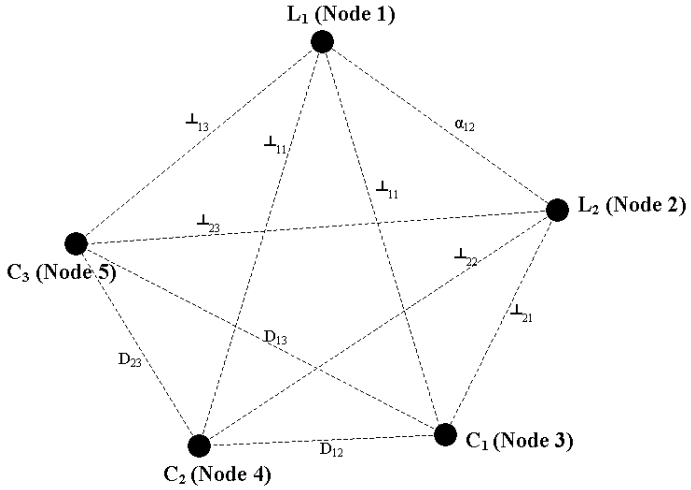


Figure 5.18: Graph representation of features in Figure 5.17

5.3.1.2 Correspondence Graph Generation

After generating two feature graphs, one for the current and previous scans, a correspondence graph between the two must be generated. The method for generating the correspondence graph is best illustrated by example.

Figure 5.19 shows two typical feature graphs A and B composed of two different node types *L* and *C* and with edge labels *a* through *n*. (Note in our case *L* represents a line and *C* a circle and the edge labels the geometric relationships between the lines and circles

but these node and edge labels could be generic labels for any type of graph.) Figure 5.20 shows the correspondence graph generated from graphs *A* and *B*.

To create the nodes of the correspondence graph we generate all permutations of same-labelled nodes (nodes of the same type). For example for node type *L* the possible permutations are *{A1,B1}, {A1,B2}, {A2,B1}, {A2,B2}, {A3,B1}* and *{A3,B2}*.

To generate the edges in the correspondence graph we must look for edge matches between the two graphs. Naturally for a real system an edge value will never be exactly the same in both scans and so edge matching will involve a certain threshold to determine if two edges can be matched.

If an edge match is found between two similar node types (*L* to *L* or *C* to *C*), then two edges are added to the correspondence graph. For example, in graphs *A* and *B* of Figure 5.19 the edge connecting nodes *A2-A3* matches the edge connecting nodes *B1-B2* and since these pairs have the same node label (L) it is unknown whether *A2* maps to *B1* or to *B2*. This ambiguity results in two lines being added to the correspondence graph thus joining node 3*{A2,B1}* to node 6*{A3,B2}* and node 4*{A2,B2}* to node 5*{A3,B1}*. (Figure 5.20)

If an edge match is found between two features of different types one edge is added to the correspondence graph. For example the edge connecting *A3-A4* and the edge connecting *B2-B3* match and since *A3* and *B2* are both of type *L* and *A4* and *B3* are both of type *C* this provides a 1 to 1 mapping of *A3-B2* and *A4-B3*. Hence nodes 6*{A3,B2}* and 7*{A4,B3}* are joined together in the correspondence graph.

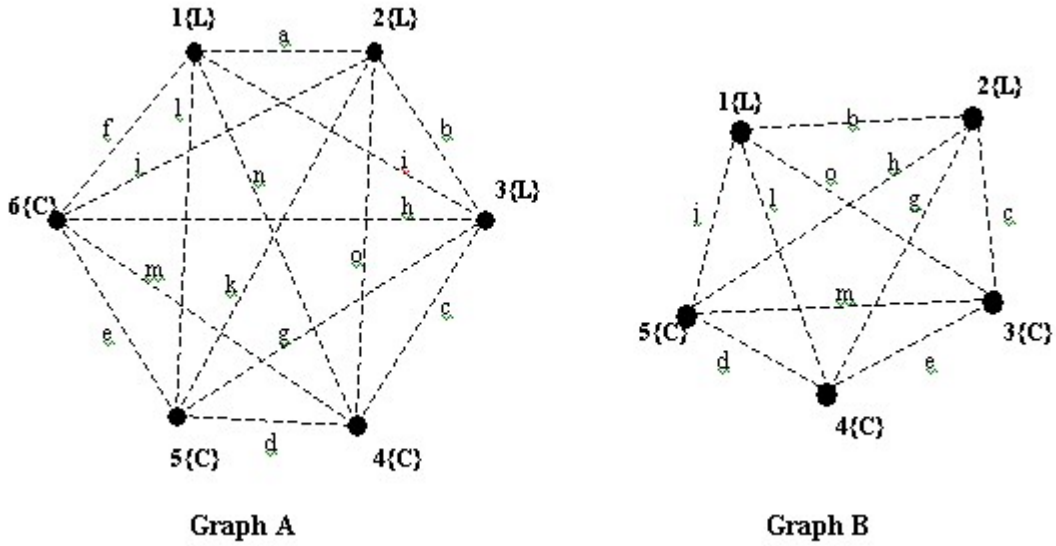


Figure 5.19: Example feature graphs A and B

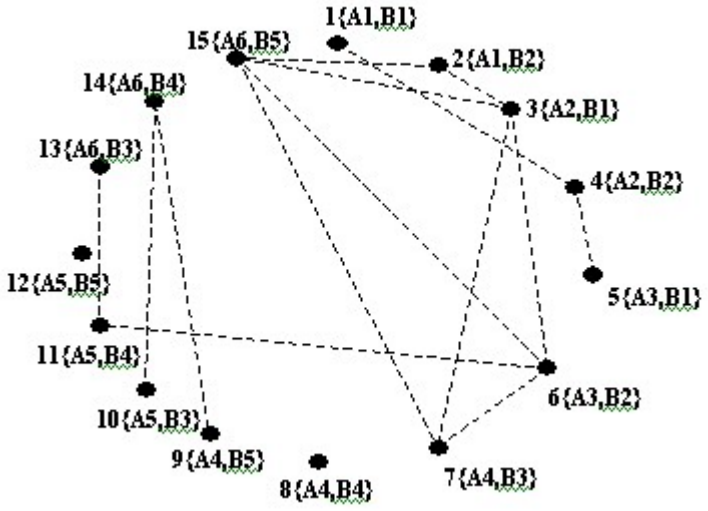


Figure 5.20: Correspondence graph of feature graphs A and B

Basically an edge in the correspondence graph indicates the possibility of mapping between two sets of points described by the nodes at the connection. A node in the CG with no edges into it means that there is no possibility of those two points being mapped.

5.3.1.3 Maximum Clique

Bailey [1] states that finding the best mapping of features between scans is equivalent to finding the maximal clique of the correspondence graph.

A clique is defined as a complete subgraph within a graph, in other words a subgraph in which every node is connected to every other node. The maximum clique for the correspondence graph is shown in Figure 5.21. The maximum clique shown indicates that the following features were matched together *{A2,B1}, {A3,B2}, {A4,B3}* and *{A6,B5}*.

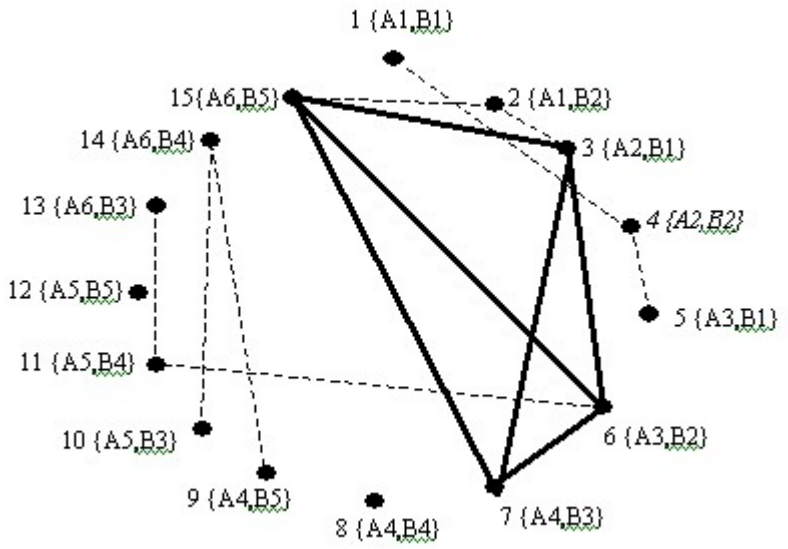


Figure 5.21: Maximum clique of the correspondence graph

Although it easy to see the maximum clique visually it is another matter entirely to compute the maximum clique algorithmically. The method used in this thesis was based on a backtracking search algorithm by Bron and Kerbosch [5]. Appendix D describes this algorithm in more detail and a code listing of the algorithm as implemented in this thesis can be found on the CD attached to this thesis.

5.3.1.4 Effectiveness of the Graph theoretic approach

The maximal clique method for finding correspondence between features proved to be quite effective in the majority of cases for clearly defining the best mapping between scans. However in many cases several maximum cliques all of the same size were found so it became difficult to determine which of these cliques contained the best mapping solution. This was caused mainly by symmetries in the environment and occasionally by the correspondence graph generation missing a match between graph edges. When this occurred the maximal clique algorithm always identified the correct feature mapping however several other false mappings were also identified with it.

Parallel lines posed quite a large problem since lines would often be paired incorrectly due to the fact that the only matching information about lines is the angles between them.

Consider the two sets of lines in Figure 5.22, $A\{L_{A1}, L_{A1}, L_{A1}\}$ and $B\{L_{B1}, L_{B1}, L_{B1}\}$, where scan B is imply formed by a rotation of scan A anticlockwise and a small translation forward and to the left, giving a correct matching of $\{L_{A1},L_{B1}\}$, $\{L_{A2},L_{B2}\}$, $\{L_{A3},L_{B3}\}$. However given that lines 1 and 2 are parallel in both scans the possibility exists for a false mapping. During the correspondence graph generation stage the angle between L_{A1} and L_{A2} will be correctly matched with the angle between L_{B1} and L_{B3} and incorrectly matched to the angle between L_{B2} and L_{B3}. This will causes the edges connecting $\{L_{A1}, L_{B2}\}$ to $\{L_{A3}, L_{B3}\}$ and $\{L_{A1}, L_{B3}\}$ to $\{L_{A3}, L_{B2}\}$ to be incorrectly added to the correspondence graph.

Whilst, for the example given in Figure 5.22, the presence of the parallel lines will not cause an incorrect mapping, there are many cases where this will cause a problem, especially when combined with errors relating to circle matches.

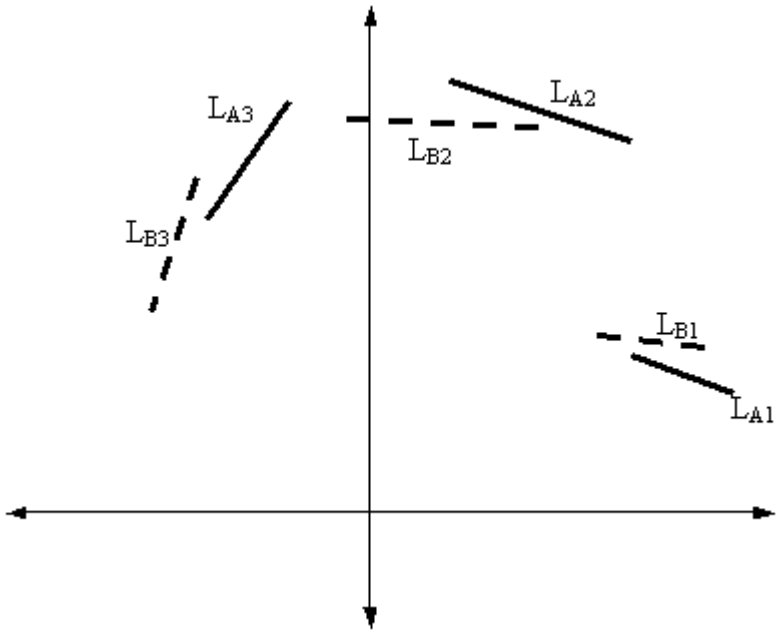


Figure 5.22: Incorrect line pairing due to parallel lines

Problems also arose with false mapping due to the distance between circles being equal and circles having the same distances to lines. For example consider the scenario depicted in Figure 5.23, showing the features extracted from two scans *A* & *B* each having 2 lines and 4 circles. Clearly the correct feature mapping between the two scans is, $\{A_{L1}, B_{L1}\}$, $\{A_{L2}, B_{L2}\}$, $\{A_{C1}, B_{C1}\}$, $\{A_{C2}, B_{C2}\}$, $\{A_{C3}, B_{C3}\}$, $\{A_{C4}, B_{C4}\}$. However notice that the distance between circles 1&2 is very close to the distance between circles 3&4 in both scans. This would result in the distance between circles 1&2 in scan *A* to be matched to both the distance between circles 1&2 and the distance between circles 3&4 in scan *B* during the correspondence graph generation stage. This could potentially cause an incorrect matching of A_{C1} & A_{C2} to B_{C3} or B_{C4}. Similar problems would occur when matching the distance between A_{C3} and A_{C4}.

Notice also that the distance between circles 1&2 and line 1 and the distance between circles 3&4 and line 2 is the same in both scans. Again the symmetries present due to the similar distances of the circles to the lines could result in an erroneous mapping of features from one scan to the next.

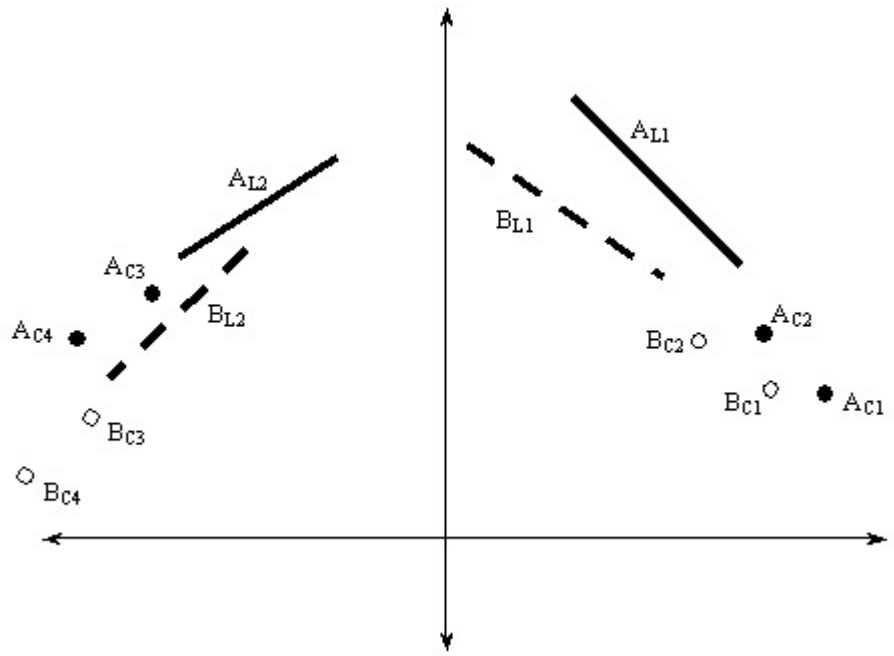


Figure 5.23: Potential false mapping due to circles being mapped incorrectly

It must be noted here that theoretically the occurrence of such symmetries should still result in only one maximal clique being generated with all other erroneous cliques being smaller and hence ignored. However experimental results indicated otherwise with a small yet significant proportion of the time multiple matching occurring.

The problems of symmetries in the environment causing false matching was solved at the correspondence graph generation stage by making use of the constraints of the vehicle motion.

Given that the scans that are to be matched are taken consecutively at an interval of about 200ms it can fairly safely be assumed that features that are far apart in Cartesian space are unlikely to represent the same feature. It was therefore decided that features (either circles or lines) having their range or bearing angles more than a specified distance threshold apart cannot be matched together. This threshold was calculated based upon the likely maximum velocity and turn rates of the vehicle and set at 5m and 50°, well above the calculated maximum movement between scans.

During the correspondence graph generation stage any edge that would connect a node in which the features could not be mapped was not added to the correspondence graph.

Therefore the correspondence graph only contained edges connecting feature pairings that were allowed by the constraints of the vehicle motion. For example consider the scenario depicted in Figure 5.23. The distance between A_{C1} and A_{L1} would at some stage be matched to the distance between B_{C3} and B_{L2}. This would cause an edge to be added between the nodes $\{A_{C1}, B_{C3}\}$ and $\{A_{L1}, B_{L2}\}$. However since A_{C1} cannot be matched to B_{C3} since they are too far apart and A_{L1} cannot be matched to B_{L2} then the edge is not added to the correspondence graph. As a consequence the correspondence becomes very sparse and a lot of memory and execution time is wasted on nodes that will never contain an edge. Therefore an improvement on the maximal clique algorithm would be to only generate the correspondence graph with node pairings that are allowed within the confines of vehicle motion. Thus greatly reducing the amount of time and space required to store and search the correspondence graph for a maximal clique.

Once the constraints of vehicle motion were taken into consideration the maximal clique method proved to be very effective in finding the best mapping of features from consecutive scans.

5.4 Pose Change Calculation

Once features from the current scan have been mapped to features in the previous scan it is then necessary to compute the transformation of the current scan relative to the previous scan. This will be equal to the change in robot pose over the interval between scans. The transformation will be calculated based upon a rotation ω and a translation T_x, Ty. This pose transformation will be calculated in a vehicle frame and will therefore need to be subsequently converted to the navigation frame.

A number of different methods for finding the pose transformation were used in this thesis depending on what kinds of features were extracted from the environment. These are outlined below.

5.4.1 Circle only Transformation

Since circles are treated as a point located at the circle centre, the only difference between this problem and the ICP problem is that in the ICP problem the point to point

mapping of features is guessed iteratively until the best match found, whereas in this case the point to point mapping is known via the maximal clique algorithm. Hence the pose transformation is simply the closed form solution to the ICP problem as presented in appendix B.

It must be noted at this point that the closed form solution for the ICP problem is only valid if more than one pair of features is mapped between scans. Naturally the more features that are correctly mapped the more accurate the method becomes but the method will work for as little as two circle pairings.

5.4.2 Line only Transformation

When the only feature mapped are lines the only information that can be discerned directly from the lines is the rotation of the vehicle since lines were modelled as infinite segments, during the data association process. It was found that tracking line start and endpoints was unreliable due to occlusions and other factors causing line endpoints to move a great deal between scans.

As a result it was proposed that the intersection of lines, or corners, could also be tracked as points features, providing the lines aren't parallel. The point does not necessarily have to represent an actual corner but would represent where a corner would be if the line segments were infinite (Figure 5.24).

So for a scan in which only lines were matched, the mean rotation is calculated based upon the change in gradient of the lines. When the average rotation is computed a simple weighting function is used based on the variance of each line. Lines with a lower variance are given a higher weighting and conversely lines with a higher variance are given a lower weighting according to Equation 5.2.

$$\Delta\theta_{mean} = \frac{1}{\sum_{i=1}^{n} \frac{a}{\sigma_i^2}} * \left(\sum_{i=1}^{n} \frac{a}{\sigma_i^2} \Delta\theta_i \right)$$

Equation 5.2

Where:

σ_i^2 is the average variance of the two lines for which the angle difference is computed

a is a weighting factor

$\Delta\theta_i$ is the angle difference between the i^{th} pair of lines

Once the mean rotation of the scans is calculated then a mean translation can be calculated using the closed form solution to the ICP problem presented in appendix B.

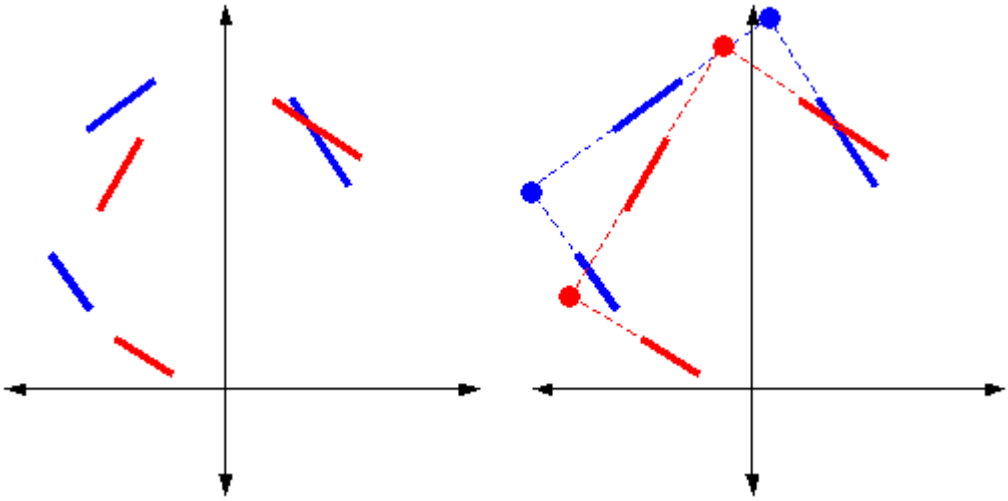


Figure 5.24: Pseudo corners from intersection of line segments

Clearly for a solution to be found the minimum number of lines features mapped must be two.

5.4.3 Line and Circle Transformation

For the case where there are both lines and circle matched between consecutive scans either or a combinations of both the methods discussed above may be used.

If the number of circles matched is two or more the ICP equations may be used to calculate the pose transform using only the matched circles. However an alternate approach is to use the lines to calculate the rotation and the circles to find the translation in an analogous fashion to using the corners.

Based on the three method discussed above it is clear that the minimum amount of features to get a valid solution for the pose transformation is either two circle pairings, a pair of circles and a pair of lines or two non parallel lines. If possible the pose change is calculated using more than one method and the average of these is taken as the pose change of the vehicle.

5.4.4 Coordinate Transform

By summing the transformations between scans a path estimate of vehicle motion can be built up or at the least the transformations can be combined with a timestamp to estimate vehicle velocity. However the pose transformations are calculated in a vehicular frame and so must be converted to the navigation frame (Figure 5.25).

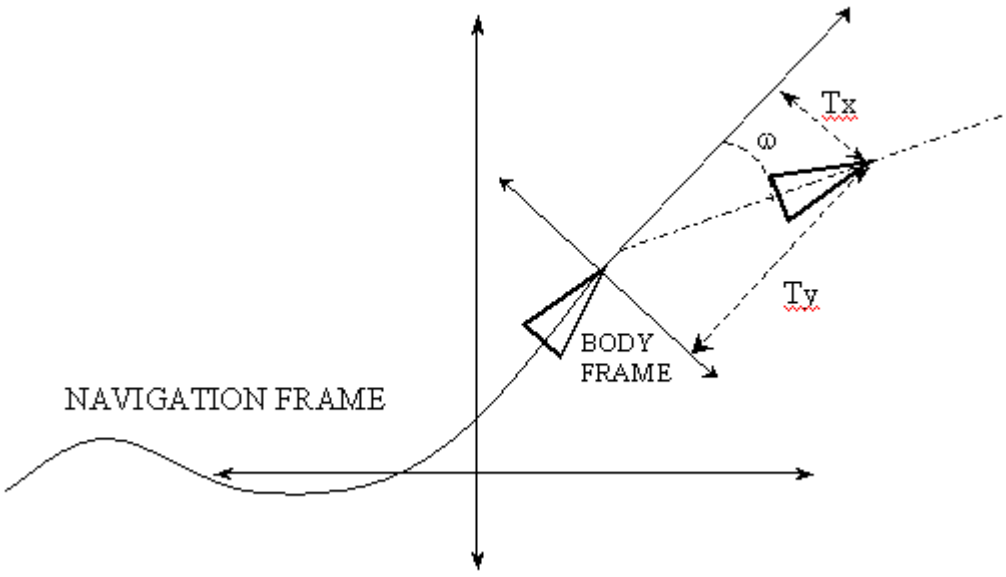


Figure 5.25: Vehicle pose change in body frame

The equations to transform the translation from the body frame to the navigation frame are given by simple trigonometry by Equation 5.3.

$$\begin{pmatrix} T_{XNav} \\ T_{YNav} \\ \omega_{Nav} \end{pmatrix} = \begin{pmatrix} \sin\phi & \cos\phi & 0 \\ -\cos\phi & \sin\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Tx_{body} \\ Ty_{body} \\ \omega_{body} \end{pmatrix}$$

Equation 5.3

Where:

- ϕ is the current vehicle heading in the navigation frame

Once the transformations are converted to the navigation frame building a dead reckoning path estimate of vehicle motion is simply a matter of summing the transformations.

5.5 Conclusion

Presented in this chapter are methods to extract features from raw laser data. Suitable features were deemed to be lines and circles since the majority of objects in the environment were planes and trees. Once features were extracted the maximal common sub-graph method was used to find the best one to one mapping of features between scans. Once the features were matched then a pose transformation could be computed and by summing these a crude dead reckoning path estimate could be built up.

Chapter 6:

Software Implementation

This chapter outlines the different software techniques used to implement the navigation algorithms discussed in this thesis. Firstly the kalman filter implementation for the model based dead reckoning is discussed along with the techniques used for real time implementation. Following this the software techniques for the laser based dead reckoning module are discussed. Not all source code is on the data CD attached to the rear of this thesis. Contained also on this CD is a detailed description of the code including major functions and variable and a description of how to use the software.

6.1 Model Based Dead Reckoning

Given that the ultimate goal of this navigation loop was to implement the system in real time, the C programming language was selected over C++ or matlab since the hyperkernal real time operating system used on the HSV could only run C code in the hyperkernal front end. Any C++ code had to run on the Windows NT side communicating to the sensors via the shared memory buffer. This will be discussed in more detail later.

The front-end program running on the HSV to read the sensors is written in C and is implemented using several hyperkernal threads, each of which reads a particular sensor and publishes the data into the shared memory region. [10] If written in C the code for

the dead reckoning filter could easily be added to these threads to be able to run in real time.

Given that a transition from the hyperkernal real time operating system to the QNX real time operating system is likely to occur in 2003 a C++ offline simulation was also developed to allow future students to easily use the filter in the future to provide position estimates for a control module. The source code has been included on the data CD at the back of this thesis.

6.1.1 Matrix Functions

To program a kalman filter it is necessary to have a small suite of simple matrix functions, namely matrix multiplication, addition, subtraction, inversion and the transpose of a matrix.

Using the algorithms in Numerical methods in C [25] a suite of basic matrix functions was written in both C and C++. Details of the functions implemented and how to use them can be found along with the source code on the data CD at the back of this thesis.

6.1.2 Real Time Implementation

As previously stated one of the main objectives of this thesis was to implement the navigation filter in real time. This section discusses the communication between the sensors and the filter in real time and the general programming concepts related to real time programming of the filter.

The encoders, brake and throttle potentiometers and steering LVDT are all sampled at a rate of 40 Hz or 25ms. GPS updates arrive at a rate of approximately 5Hz, 200ms. Hence each iteration of the filter must be able to run in under 25ms in time for the next set of prediction data to be used. Tests conducted on the Ute's computer using an offline simulation of the filter with sensor data arriving from a text file, indicated that the average iteration time of the filter was under 1ms meaning that the execution time would not pose a major problem to the running of the filter online.

The functionality of the filter in real time is outlined as follows. The thread sampling the sensors reads the information and publishes the sensor data into a known location in the shared memory buffer. Another thread, the filter thread, then reads this information from the shared memory buffer and performs a prediction or an update depending on whether the GPS information is current or not. The filter then publishes the position of the HSV back into the shared memory for another application such as control, or a user interface to use. Figure 6.1 shows this graphically.

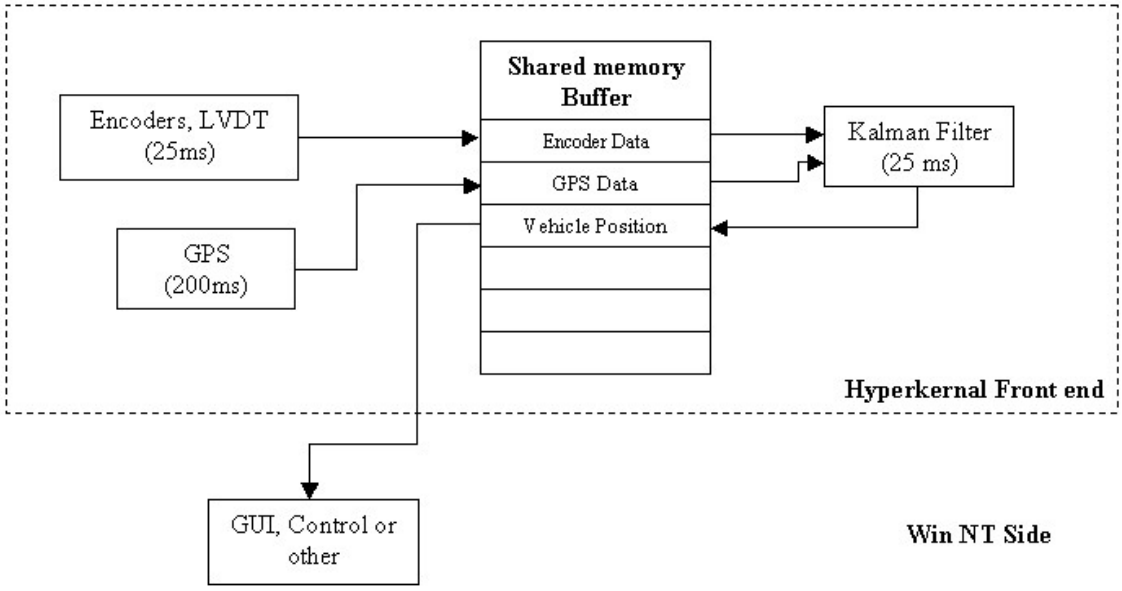


Figure 6.1: Software architecture and data flow for real time operation

When working with multithreaded programs such as this, care must be taken to ensure that the shared data used is up to date and isn't modified by another thread. For example in this application if the filter were to read the encoder information from the shared memory but before it could read the LVDT information the thread that reads the encoders and LVDT were to gain control of the machine, then the filter would be working with new encoder data but old LVDT data. A similar problem may arise when writing the position of the vehicle into the shared memory region. To ensure that thread is never interrupted during a data transfer to the shared memory buffer, calls to the functions *HKEnterCriticalSection()* and *HKLeaveCriticalSection()*, are made immediately before and after data transfer to disable and enable thread switching respectively.

6.1.3 Graphic User Interface

A simple graphic user interface (GUI) was designed to plot the path of the vehicle in real time. The layout of this interface is shown in Figure 6.2.

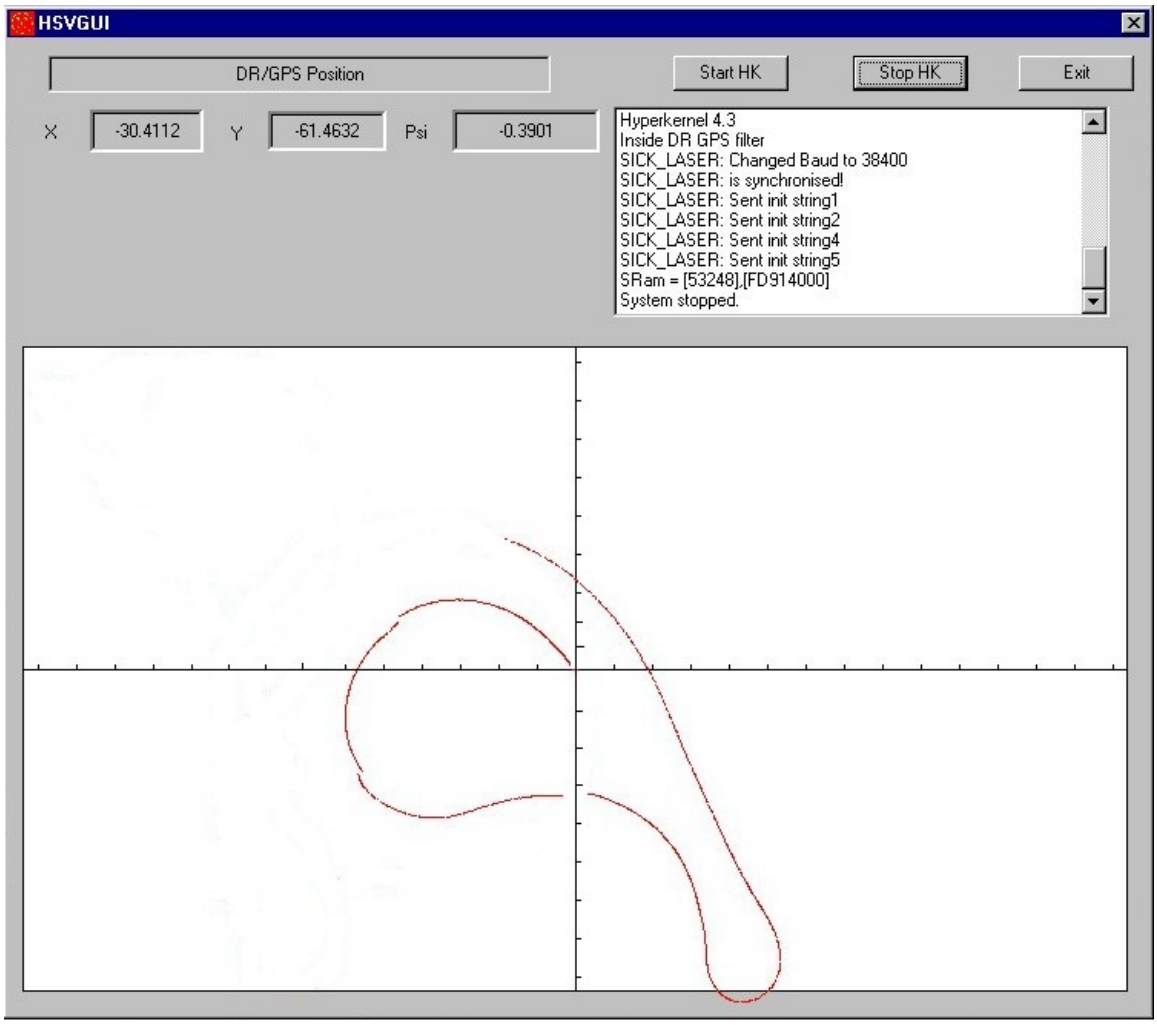


Figure 6.2: GUI to plot position

The GUI is a Win NT application and as such is non real time. The GUI was written in C++ using the Microsoft Foundation Class (MFC). The GUI simply stores a pointer to the region in shared memory where the position updates are written and periodically updates the display of vehicle position. Again source code for this GUI can be found on the data CD. Both an offline simulated version and online version have been included.

6.2 Laser Based dead reckoning

This next section discusses some of the software techniques used to implement the algorithms for laser dead reckoning as presented in chapter 5.

For the same reasons as the model-based dead reckoning the laser-based dead reckoning module was coded using the C programming language. However due to some of the constraints of the algorithms (discussed in chapter 5&7) it was not possible to put this module into real time operation on the Ute.

Again an outline of the functions used and major variables has been included on the data CD for the interested reader. However a brief mention about graph representation in software is discussed.

6.2.1 Graph Representation

[23] proposes two methods for representing a graph. The first is using a connectivity matrix of size $m_x m$ where m is the number of nodes in the graph. If two graph nodes are connected then a one is placed in the correct place of the connectivity matrix. For example if node one is connected to node two then a one is placed in elements *(1,2)* and *(2,1)* of the connectivity matrix. In general is node i is connected to node j then elements *(i,j)* and *(j,i)* will both be true, equal to one. From this it is clear that the connectivity matrix will be symmetric.

The second method proposed is using an array of linked lists. An array of all nodes is set-up and each element in the array is the head of a linked list containing all nodes to which the head node is attached. For example consider the graph shown in Figure 6.3. An array of four elements would be created where each element represents the head of a linked list of node structures. Element zero of the array would represent node one. This element would head a linked list whereby the first element in the list points to node two, the second points to node four and the third to null to indicate no more connections. Similar lists would be set up for each of the four nodes. This is shown graphically in Figure 6.4.

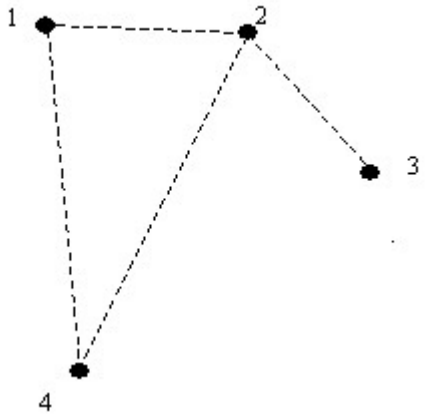


Figure 6.3: Example graph

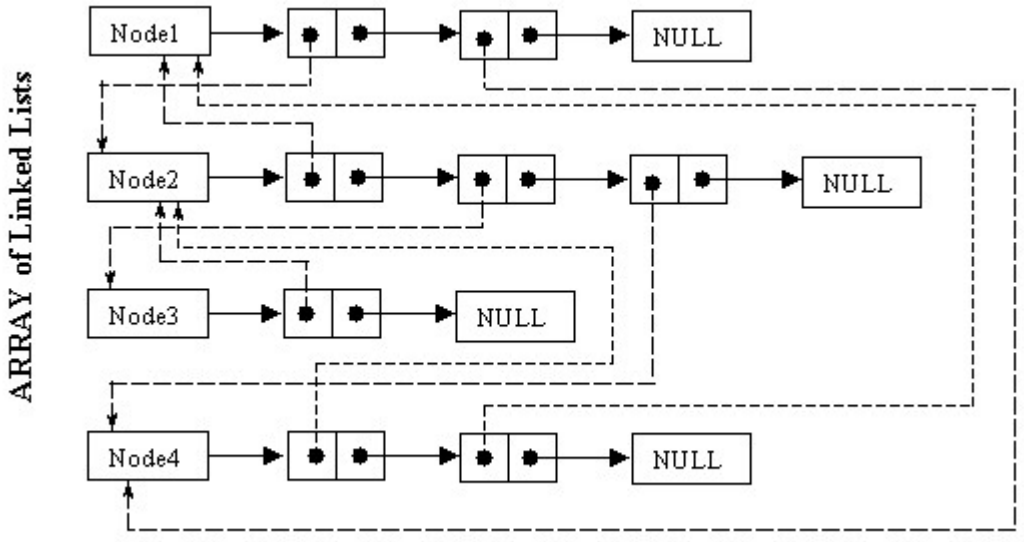


Figure 6.4: Graphical representation of graph linked lists

The linked list implementation requires less storage space than the connectivity matrix method since in the connectivity matrix half of the space is wasted since the matrix is symmetric. However in the linked implementation checking to see if two nodes are connected requires traversing a list whereas using a connectivity matrix connection between nodes can be found immediately simply by checking the correct space in the matrix. One other advantage of the linked list implementation is that it allows data to be stored at the nodes.

Despite the advantages of the linked list implementation a connectivity matrix was used to represent graphs in this thesis. The main reason behind this was that the maximum clique algorithm of [5] used a connectivity matrix implementation.

Chapter 7:

Results

This chapter outlines the results of all the different algorithms discussed in this thesis. Data from the Ute's sensors were logged in a variety of environments around the university. The Ute typically traveled at low speeds of less than 20km/h during testing, mainly for safety reasons. Some of the testing locations are outlined below:

- Seymore centre car park roof – Very open area no buildings nearby excellent GPS reception
- Victoria park – Many scattered trees
- Grass area next to ACFR building – Fairly open area enclosed by buildings, contains two large palm trees.

The a discussion of the performance of the Kalman filter discussed in chapter 4 is presented followed by a discussion of the practicalities of using a laser based dead reckoning procedure.

7.1 Model Based Dead Reckoning

The following results outline the performance of the Kalman filter in two main situations; with good GPS data and with a multi-pathed set. Both online and offline results are presented along with graphs related to the filter performance. All results are obtained from the C program that implements the filter and all plots are done in matlab using data saved to a text file by the C program.

7.1.1 Accuracy of the Model

The following data set was taken from the roof of the Seymour centre car –park at the university of Sydney using a differential GPS (DGPS) setup. Figure 7.1 and Figure 7.2 shows the accuracy of the vehicle model. The blue path indicates the predicted path from the model and the red path is the true path from the DGPS.

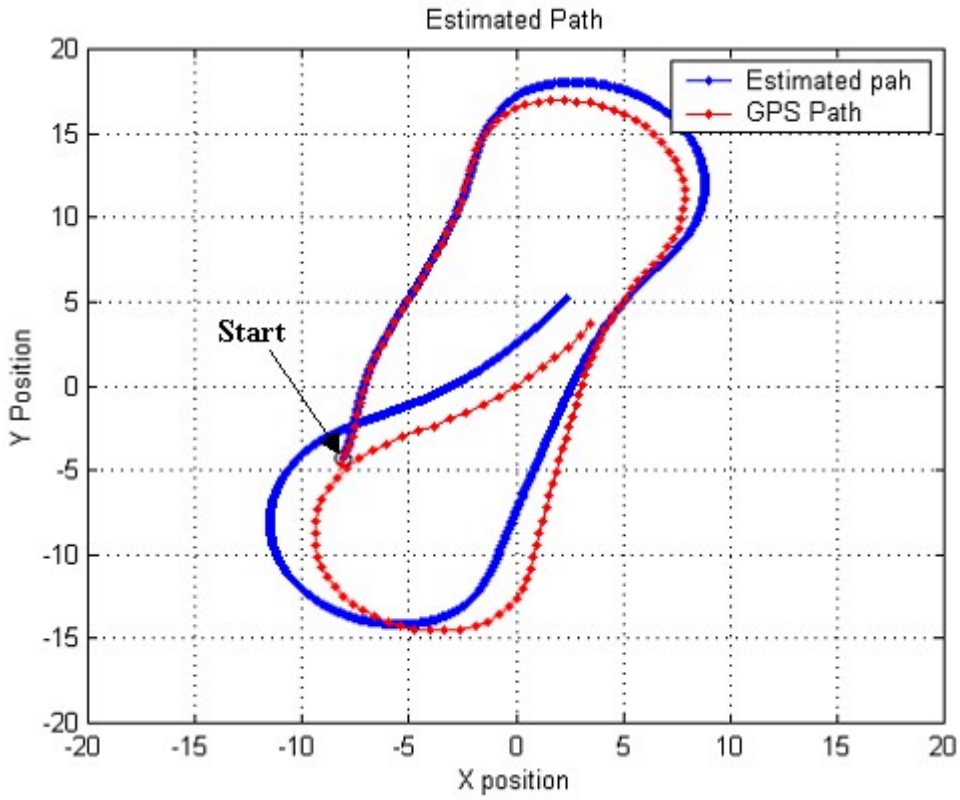


Figure 7.1: Accuracy of the vehicle model

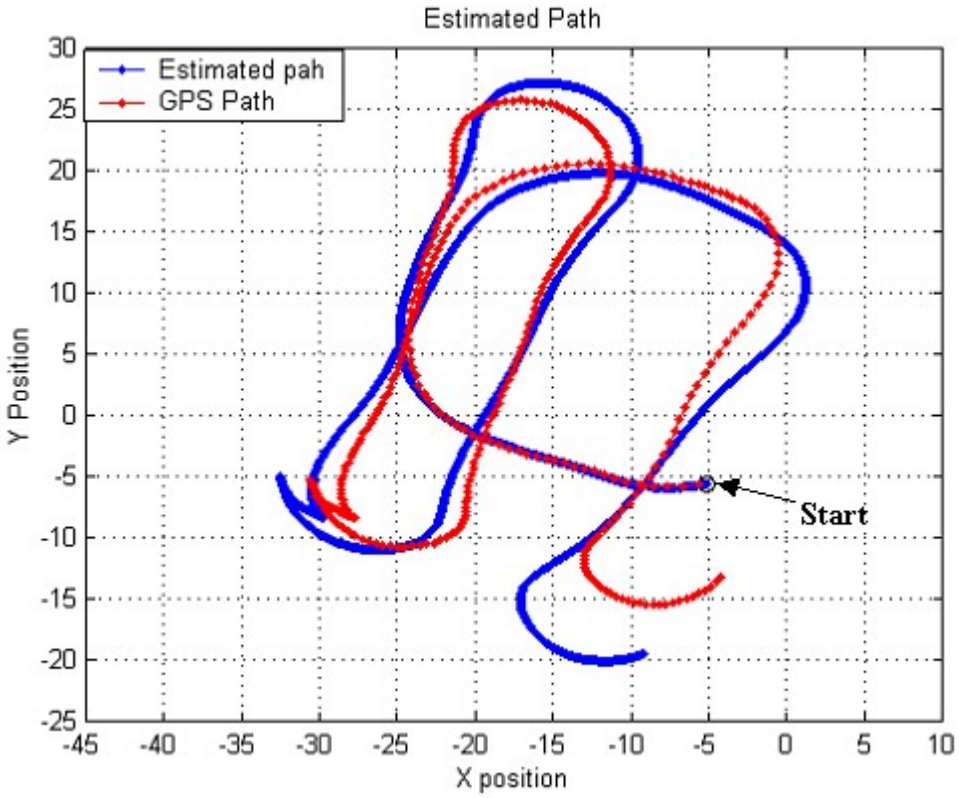


Figure 7.2: Accuracy of vehicle predictions using the model

The model performs very accurately over straight segments whilst a lot a less accurately over turns. This is because the model assumes a constant steering angle and velocity between predictions and when driving straight the steering angle is relatively constant but in the turns the steering angle and velocity are both changing somewhat. Naturally the accuracy of the model decreases as time goes on since the error is constantly growing without bound.

These results do indicate however that the model is reliable to predict vehicle positions over distance of several metres with only 1-2m, as seen in Figure 7.1. Therefore during multipath the filter should still be able to provide fairly accurate position estimates provided the GPS returns after a fairly short absence and also provided that multipath is detected accurately. Naturally if the HSV were to drive in something like a long tunnel causing extended periods of bad GPS then the position estimates would become extremely inaccurate.

7.1.2 Offline Filter Performance

The performance of the filter was first verified offline using data logged from the Ute's sensors. Scenarios in which the GPS data was excellent along with cases where the GPS data was faulty are both examined to determine the robustness of the filter. In particular this section focuses on aspects of filter tuning and how it affects performance.

7.1.2.1 Accurate GPS

Initially the filter was tested using ideal GPS data taken from the roof of the Seymour centre car park using a differential GPS setup, resulting in centimeter accuracy on the GPS data. Figure 7.3 shows this environment with a typical vehicle path shown.

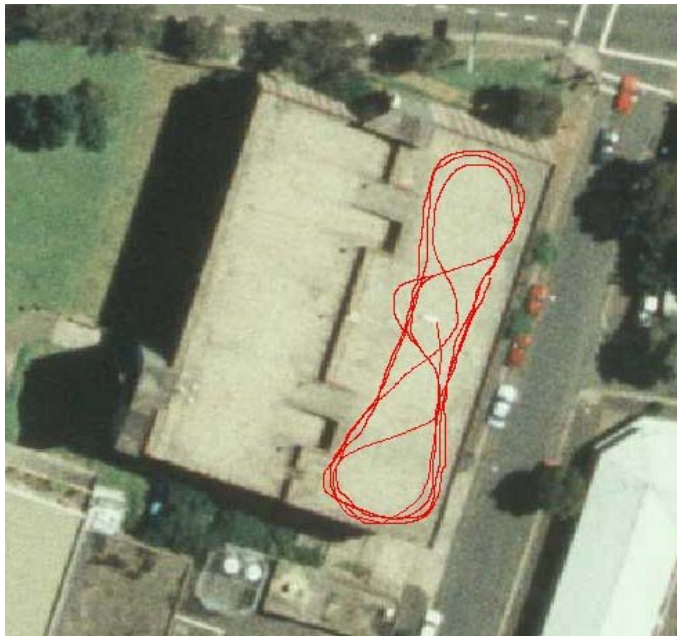


Figure 7.3: Seymour centre car park roof and typical vehicle path (photo courtesy of http://www.bearings.nsw.gov.au/)

Initially the performance of the filter was tuned using fixed noise values on the GPS. The best set of parameters found was as follows: $\sigma_{steer} = 3°$, $\sigma_{vel} = 0.3m/s$, $\sigma_{lat} = \sigma_{long} = 0.07m$. The process noise parameters (σ_{steer} and σ_{vel}) were not altered during these tests. Figure 7.4 and Figure 7.5 shows the vehicle path generated from this data set. The GPS position data is shown in red and the filtered estimates shown in blue.

The filter performed excellently providing a smooth path throughout that lay almost right on top of the GPS information.

Notice in Figure 7.5 the saw tooth shape so typical of Kalman filters caused by the frequent updating of the position estimates by the GPS data.

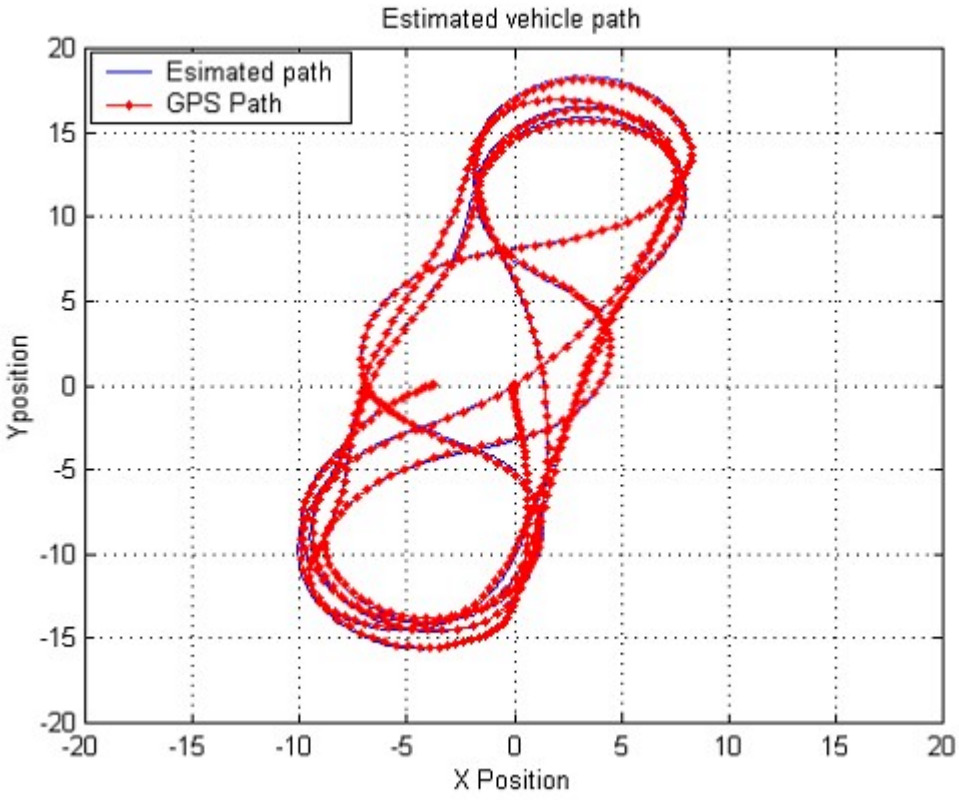


Figure 7.4: Estimated vehicle path. Data set taken from Seymore centre car park roof using DGPS. (fixed observation noise)

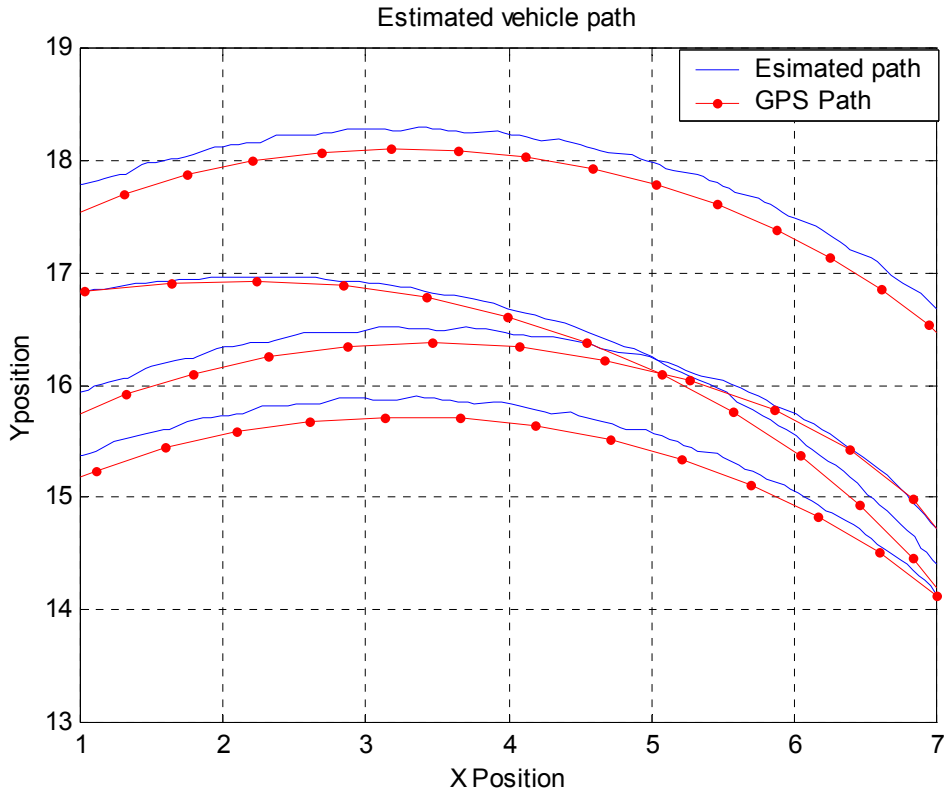


Figure 7.5: Close up of the estimated path shown in Figure 7.4

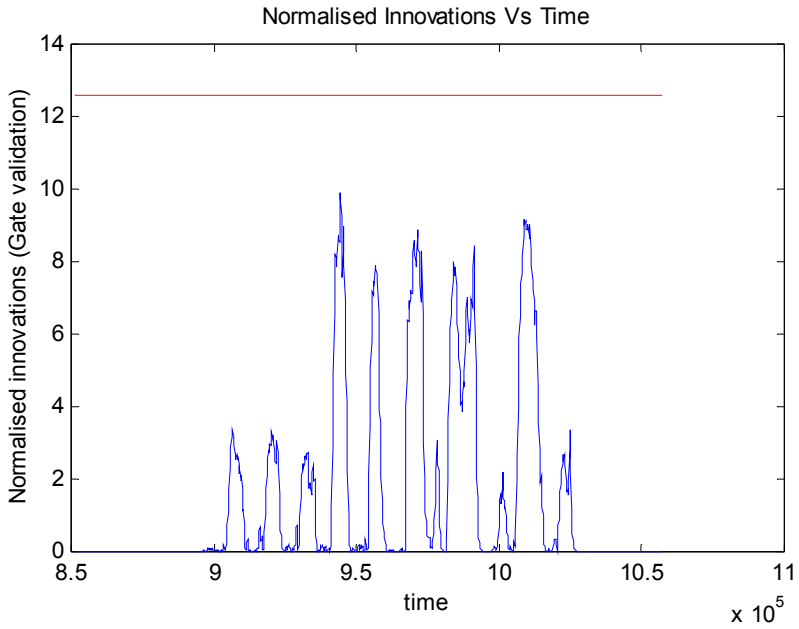


Figure 7.6: Normalised innovations (γ) of Figure 7.4

Shown in Figure 7.6 are the normalised innovations used in the gate validation tests for the detection of multipath effects. Notice that the normalised innovation never rises

above 12.6 meaning that the filter is operating well and that multipath is not an issue in this data set.

A second guarantee that the filter is performing correctly is that the innovations are both unbiased and white. Figure 7.7 shows the innovations and Figure 7.8 shows the autocorrelation of the innovations. It can be seen clearly from Figure 7.7 that the innovations are unbiased and the autocorrelation of Figure 7.8 has a mean of zero indicating that the innovations are uncorrelated and hence white.

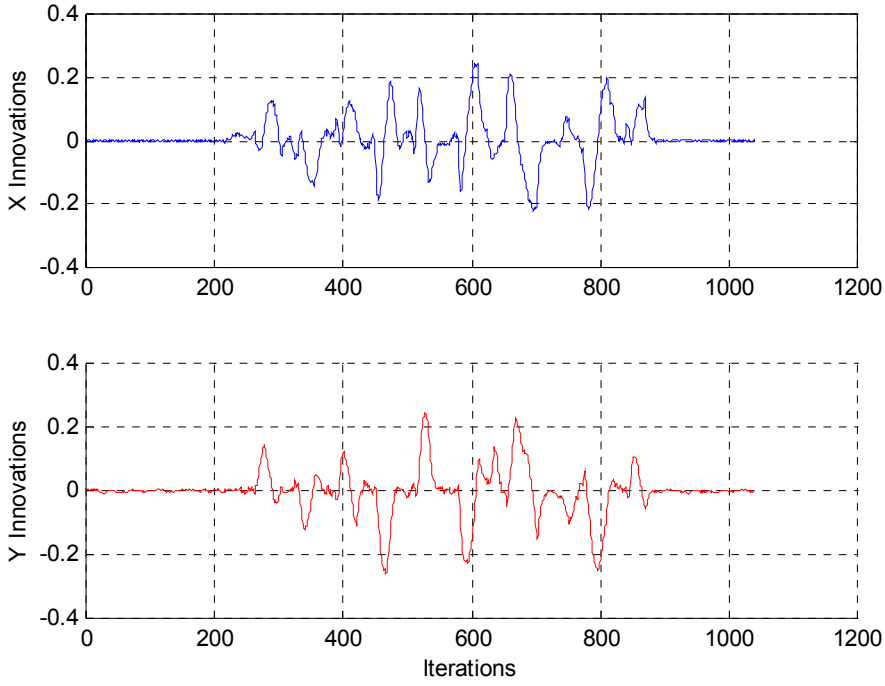


Figure 7.7: Innovations of Figure 7.4

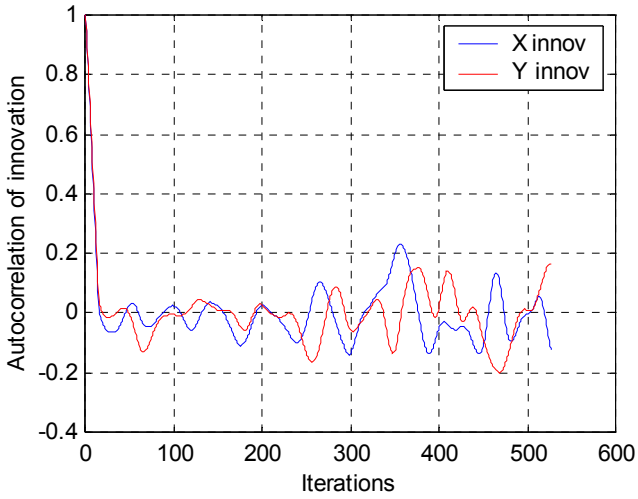


Figure 7.8: Autocorrelation of innovations of Figure 7.4

The performance of the filter was tested again using the same data set, only this time the process noise matrix *R(k)* was assigned using the variance information available from the GPS unit. To provide a stable filter output the variance from the GPS receiver had to be multiplied by a factor of 5. Any less and the gate validation test would detect an error and the filter would never recover.

Figure 7.9 shows the estimated path generated using an observation noise matrix with variance values returned from the GPS receiver and multiplied by a factor of five, a value deemed to large to be accepted. The filter performs well but notice compared to Figure 7.4 the path estimated is further away from the GPS data since the observation noise was set too high.

Figure 7.10 shows the variance values returned from the GPS receiver for both longitude and latitude. The mean variance is 0.01m, which results in an average standard deviation of 10cm. The maximum variance is approximately 0.038m which gives a standard deviation of approximately 20cm. Hence it is clear that the GPS information in this data set is extremely accurate and there is no reason why updates should be rejected as multipath errors.

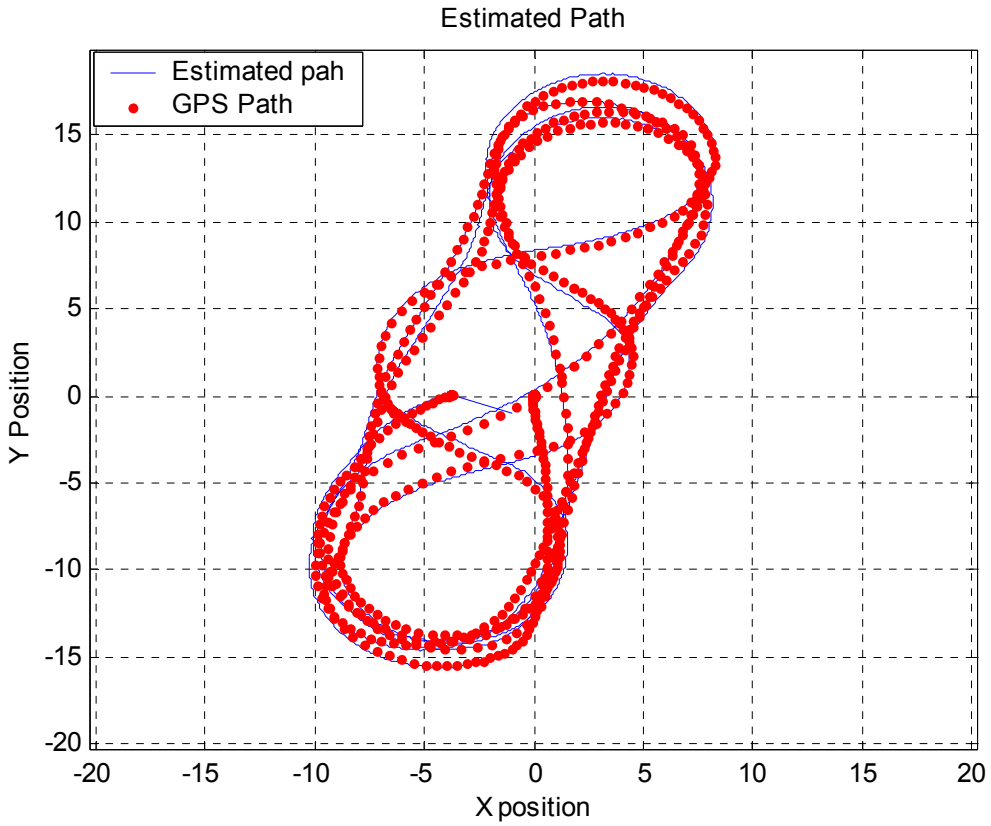


Figure 7.9: Estimated vehicle path using data set from Seymour centre car park. Observation noise set using variance returned by DGPS receiver. This noise was multiplied by a factor of 5.

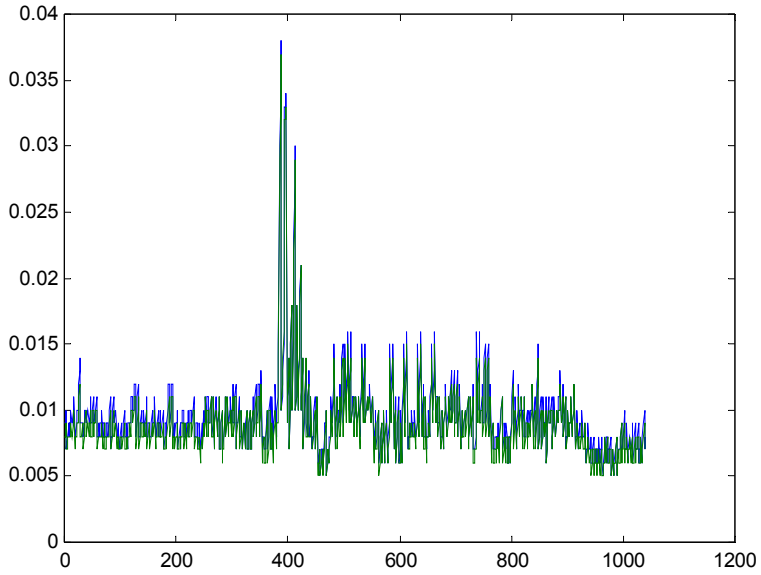


Figure 7.10: Variance of DGPS data as returned by DGPS receiver. Mean value = 0.01m

A further test was conducted turning off the gate validation criteria and using the raw variance values returned by the GPS receiver un-scaled. The performance of the filter was ideal giving an output graph practically identical to that obtained using the fixed variance in Figure 7.4. In fact while the gate validation was off the filter basically performed excellently for a whole range of observation noise values.

From these results is can be concluded that when the differential GPS is used in a wide-open space such that centimetre accuracy in the GPS signals is available, the filter may become unstable if using the gate validation test and the observation noise is set too low. This is because the gate validation test may detect an error, when there is none, and the filter may never recover from this error rejecting all updates from that point onwards. Hence the gate validation test may be turned off if the integrity of the GPS data can be assured. If using the gate validation with centimetre accurate GPS data great care must be taken to ensure that the filter is properly tuned or the gate validation test is turned off.

7.1.2.2 Inaccurate GPS

Given that the accurate performance of the filter under ideal conditions, the filter was then tested using a data set containing very inaccurate GPS. The purpose of this was to test the robustness of the filter in terms of rejecting multipathed GPS data using the gate validation tests.

The following data set was taken from the area next to the ACFR building. This area contains two large palm trees and is bounded on three sides by buildings. As a result the GPS can really only maintain an accurate signal out in the center of the test area. As soon as the vehicle passes under the palm trees or in some cases next to the buildings the GPS looses integrity. The data set was taken without using the differential GPS and as such the overall variance of the GPS is of the order of metres. Figure 7.11 shows an aerial picture of the testing environment, with a typical vehicle path.

Figure 7.11: Aerial shot of the ACFR building and surrounds, showing a typical vehicle path (photo courtesy of http://www.bearings.nsw.gov.au/)

Initially the filter is tested using a fixed observation noise of $\sigma_{steer} = 3°$, $\sigma_{vel} = 0.3m/s$, $\sigma_{lat} = \sigma_{long} = 0.4m$. Figure 7.12 shows the estimated path of the vehicle using the data set taken from next to the ACFR building. Compared to the accurate GPS data (Figure 7.4) the GPS data at times seems erratic and somewhat coarse in places. The orange circles identify two examples of multipath. Notice that in these cases the filter simply ignores the GPS data and makes predictions using the vehicle model until the GPS data is deemed to be reliable again. Notice when the updates are accepted the estimated path hugs the GPS path fairly closely since the observation noise is set at 40cm, still a fairly small value, when compared to the noise value returned by the GPS.

Figure 7.13 shows the normalized innovations used by the gate validation of this data set. Notice that when the GPS data has multipathed the normalized innovations become extremely high, well above the 12.6 threshold and once the GPS data returns to normal the value of the normalized innovations falls below the threshold.

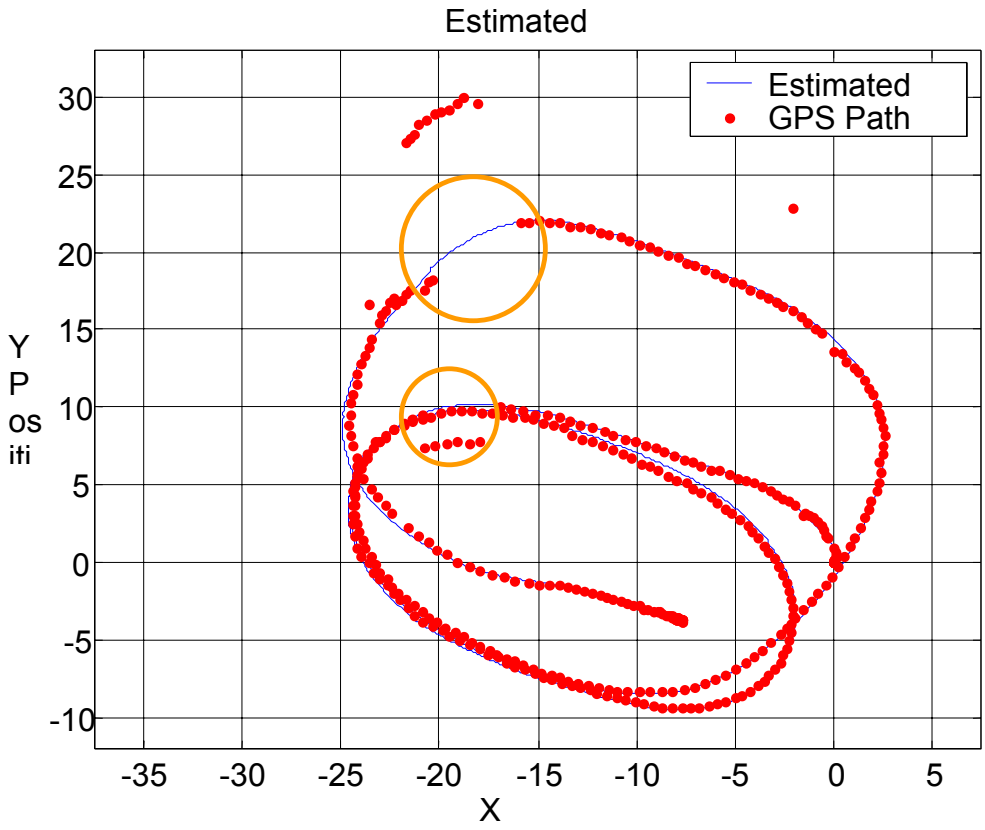


Figure 7.12: Estimated vehicle path, showing the effects of multipath rejection with a fixed observation noise. (Data set from area next to ACFR building)

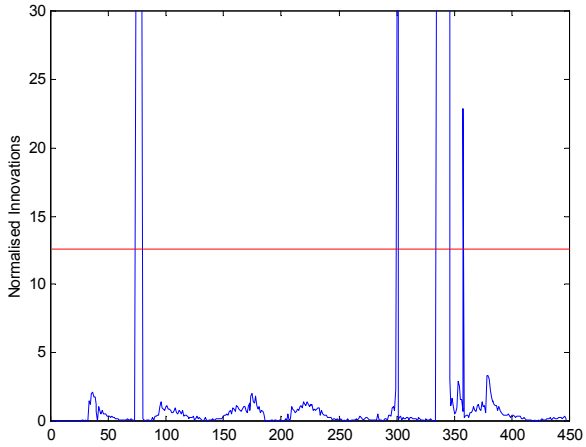


Figure 7.13: Normalised Innovations of Figure 7.12 (data set taken next to ACFR building)

The filter was then tested with a variable observation noise matrix using the variance values returned by the GPS. Figure 7.14 shows the variance data from the GPS receiver in the data set taken from next to the ACFR building. The mean variance is around 1.5m with peaks occurring in areas where the data has under gone multipath, most notably the two peaks at around 35m.

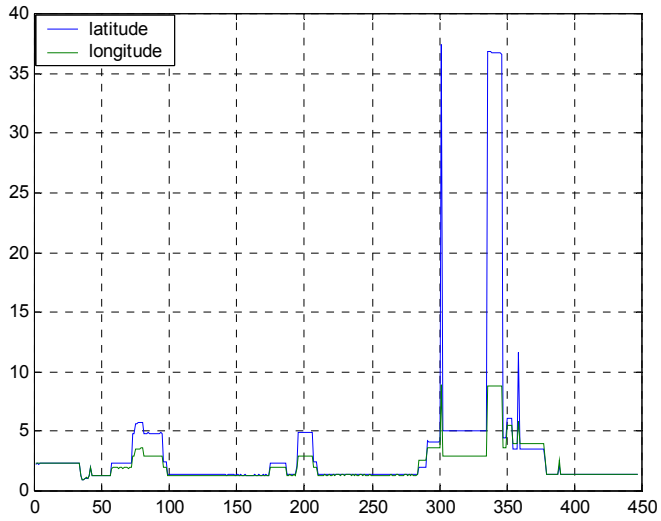


Figure 7.14: Variance on GPS data taken from next to ACFR building.

Figure 7.15 shows the results of the filter using the variance returned by the GPS receiver in the observation noise matrix. The noise values are used un-scaled. Again the filter performs admirably outputting a smooth vehicle path despite the large discontinuities in the GPS. Notice that in this case the estimated path deviates from the GPS path a little more than the fixed observation noise case (Figure 7.12). This is because the covariance returned from the GPS is bigger than the fixed value assumed in Figure 7.12. One thing of note is that when the variable observation matrix is used very few observations trigger the gate validation test, two in fact. Hence discontinuities in the path are avoided not by the gate validation test but by the variance being so large on multi-pathed data, that the GPS data is completely untrustworthy and so the observation hardly shifts the position at all.

Both Figure 7.12 and Figure 7.15 present viable solutions for the path of the vehicle and it is unknown which of these more closely represents the actual vehicle path since the

actual position of the vehicle cannot be known 100% due to the errors on the GPS. So ultimately it is up to the user to determine whether a fixed observation noise or a variable on utilising the variance data from the GPS is used.

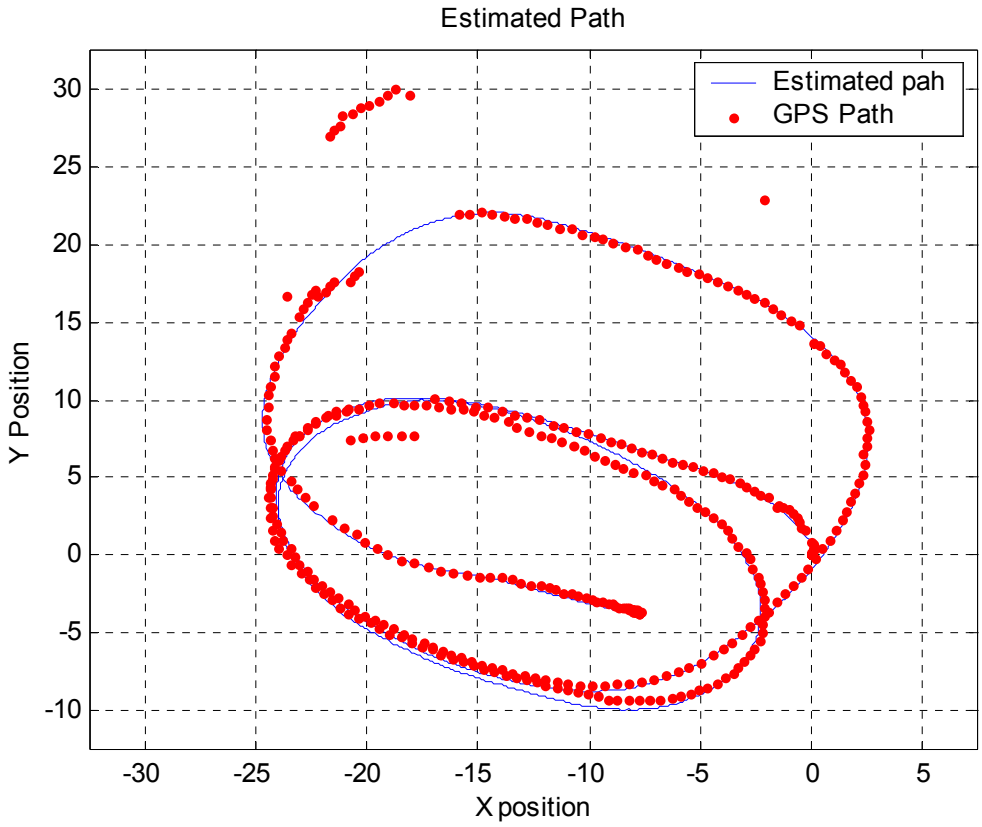


Figure 7.15: Estimated path using variable observation noise with variance data returned by the GPS receiver.

Finally a scenario is shown in Figure 7.16, whereby the GPS information is very bad. In some instance it is hard to tell which is the good GPS data and which the bad. This has been included to illustrate that even under the worst conditions the filter can still provide smooth vehicle path estimates that are fairly accurate, at least as accurate as possible. It must be noted however that this is not the ideal situation for the navigation filter, as it should be used in open environments with a very sparse scattering of trees or buildings.

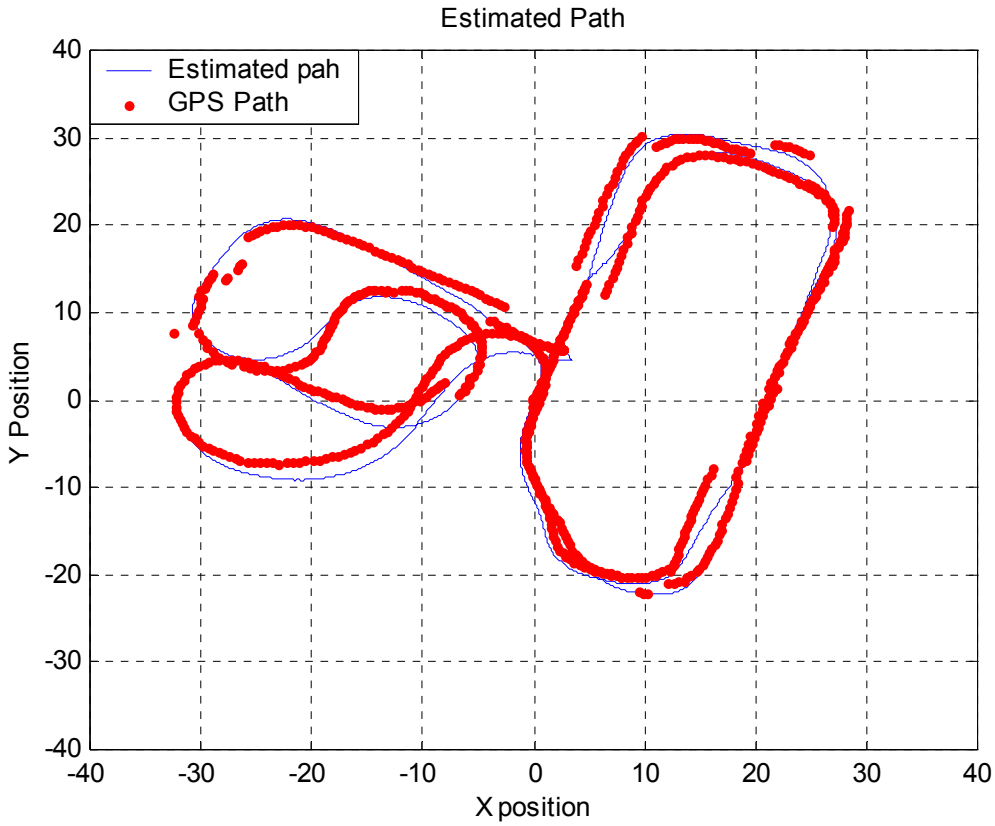


Figure 7.16: Estimated path using a data set with terrible GPS information. (Variable observation noise used)

7.1.3 Online Results

After verification of the results offline the filter was tested in real time using the GUI and the real time techniques discussed in chapter 6 software implementation. It must be noted that no useful information regarding correct filter operation can be gleamed online since no extra data regarding GPS accuracy and the gate validation and innovations is available. The only output is the estimated position of the vehicle. The only real indication about online performance is driving past the same point more than once and checking if it shows up as the same position in the graph. Basically if the filter works robustly in all simulations using logged data then it should be a trivial matter to convert the filter to a real time application. All that needs to happen is that the program reads and writes the data from shared memory instead of a text file.

Online testing yielded promising results indicating that the filter algorithms developed offline could be used to provide accurate position estimates of the vehicle in real time.

Figure 7.17 shows one such example of the filter working well in real time in the area next to the ACFR building.

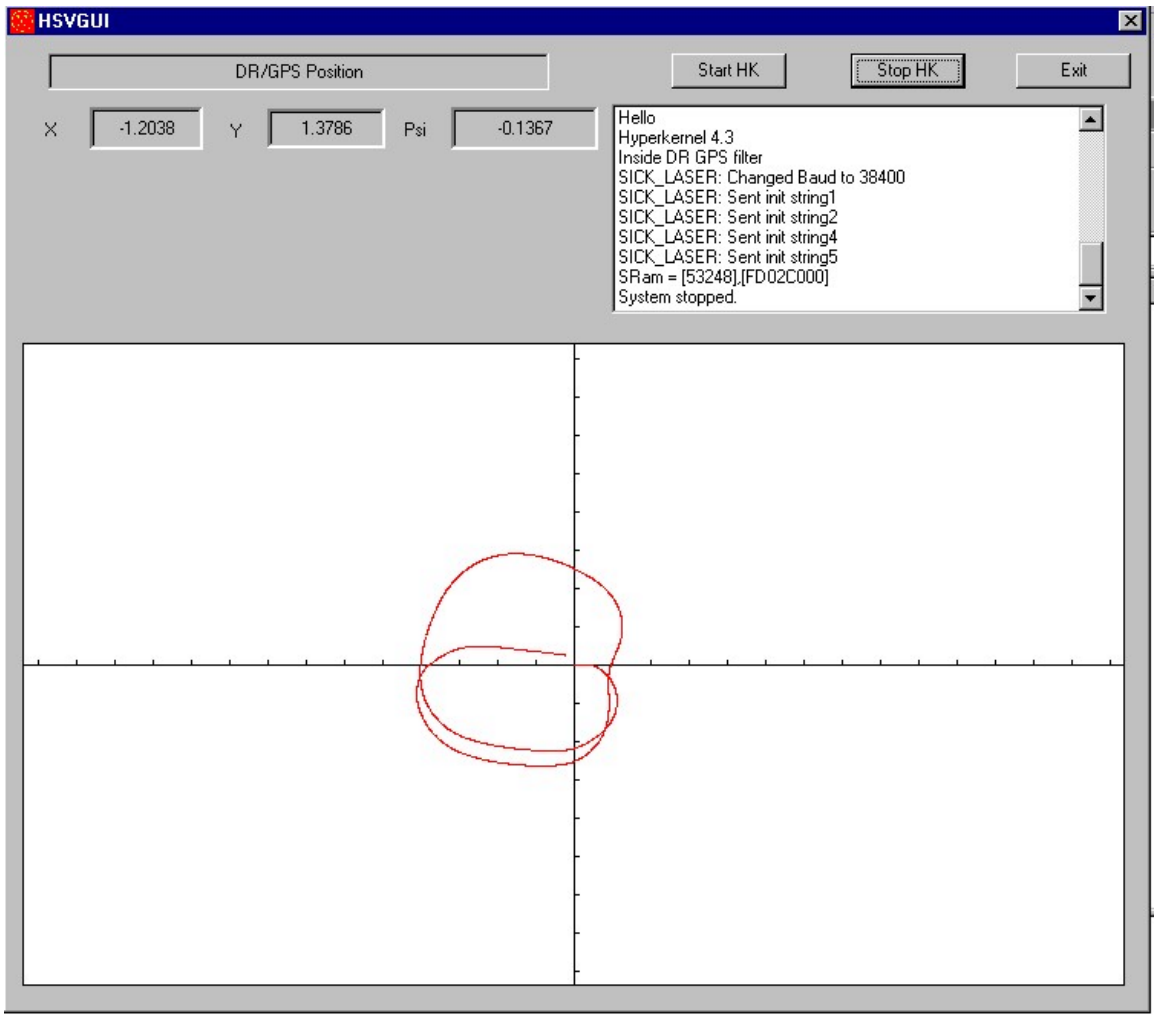


Figure 7.17: Simple vehicle path plotted online in the area next to the ACFR building

7.1.4 Summary

Given the results just discussed it is clear that the performance of this navigation loop is highly dependant on the tuning of the Kalman filter thus it is important to tune the filter for the specific application and environment.

The ideal value for the process noise was found to be $\sigma_{steer} = 3°$, $\sigma_{vel} = 0.3m/s$, and was based on the theoretical error of the LVDT and wheel encoders. This value was unchanged during the tuning processes of the filter.

In the situation where the base station is used to provide differential GPS resulting in extremely accurate GPS information, care must be taken when using the gate validation tests that the observation noise is not too low so as to cause the filter to reject all updates. It was found that the filter could use fixed noise values (standard deviation) typically of the order of 10-15cm with the gate validation turned on, and no problems would arise. An alternative to this was to turn the gate validation off and use the variance values returned by the GPS unit multiplied by 1.5. One method would be to leave the gate validation on but if the uncertainty of the GPS is lower than some threshold say 40cm then the gate validation is not used. Once tuned correctly it was found that the filter provided excellent position information with ideal GPS data.

For the more noisy GPS data both the fixed and variable uncertainty methods provided accurate position estimates of the vehicle. Typically fixed values of around 0.5-1.0m for the standard deviation were used to provide a reliable and smooth estimate.

In general however it was found that using fixed tuning was somewhat specific to the environment and the type of data logged, whereas using the uncertainty from the GPS receiver was more robust and worked in all environments providing that the gate validation was taken care of by disabling it when the uncertainty fell too low (less than 30-40 cm).

7.2 Laser Dead Reckoning

The following sections outline the results of the feature extraction and data association techniques presented in chapter 5.

7.2.1 Feature Extraction

7.2.1.1 Line Detection

The final line detection strategy used was based on the recursive bisection of a line until the either bisected segment meets the variance criteria or the number of points in the segment becomes too small for further bisection.

In general this method of line extraction worked very well especially in cases where the laser data clearly define a line or a corner. Cases in which the laser data was noisy often caused the method to extract lines that may not be an accurate representation of the object.

Figure 7.18, Figure 7.19 and Figure 7.20 show the line extraction algorithm working well, extracting lines that very closely represent the shape of the objects.

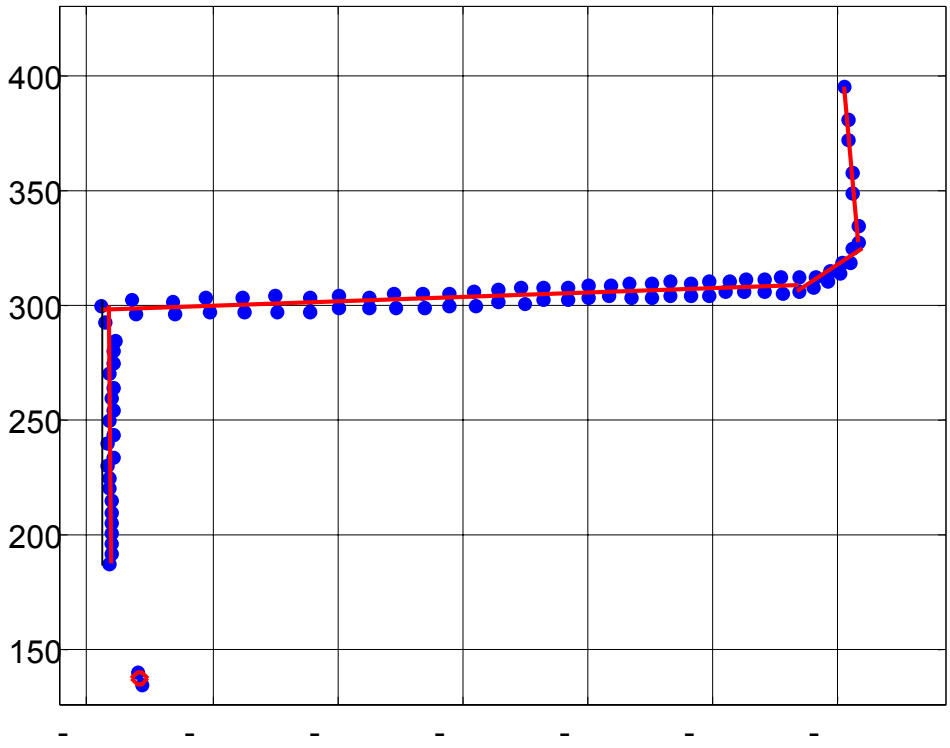


Figure 7.18: Successful line extraction (I)

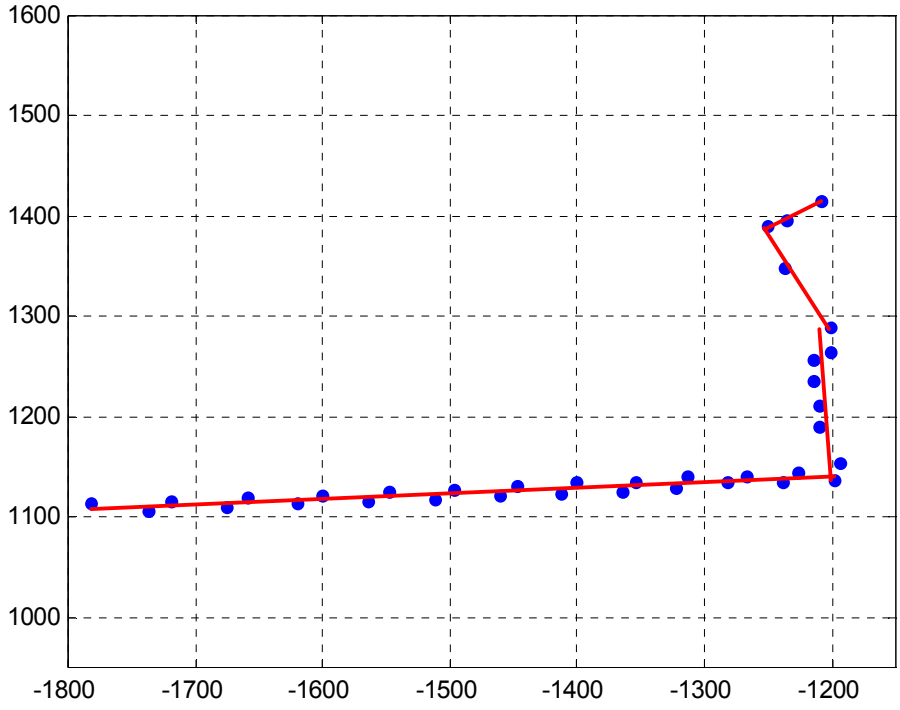


Figure 7.19: Successful line extraction (II)

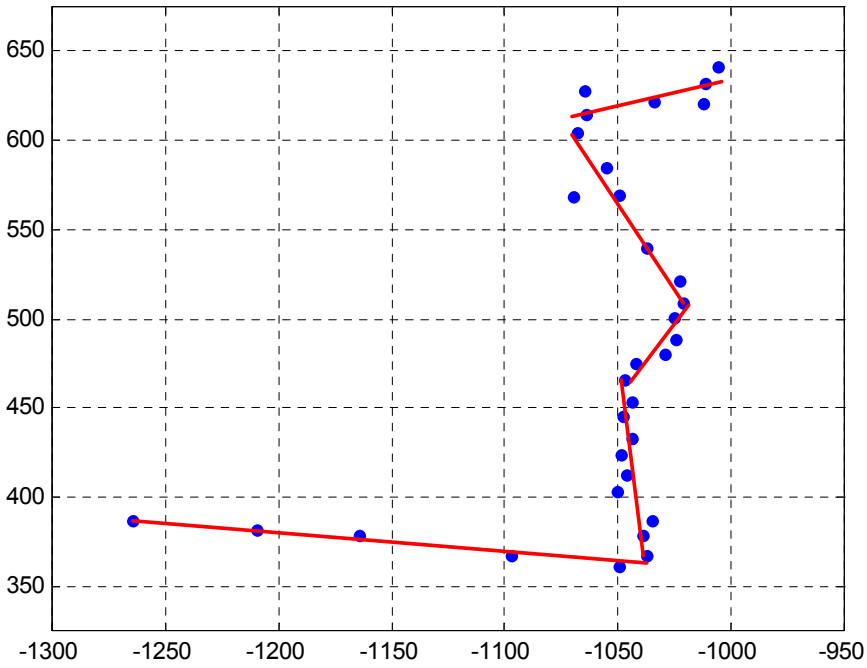


Figure 7.20: Successful line extraction (III)

Figure 7.21 and Figure 7.22 show the line extraction technique failing to identify lines correctly to represent obstacles. This is due to the fact that the shapes of the obstacles scanned are odd shapes and/or the laser scan data is very noisy. Unfortunately not much can be done about this and given that the line detection algorithm worked remarkably well in the majority of cases, the odd situation where erroneous obstacles were extracted was ignored. In any case lines that were extracted incorrectly were not matched to other lines in the data association stage.

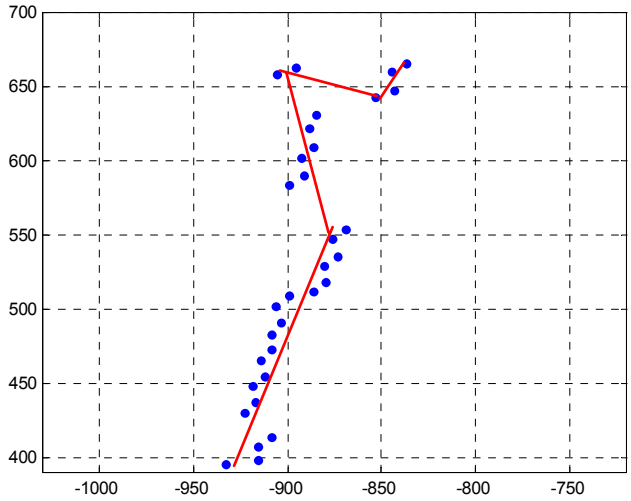


Figure 7.21: Line detection failing to some degree (I)

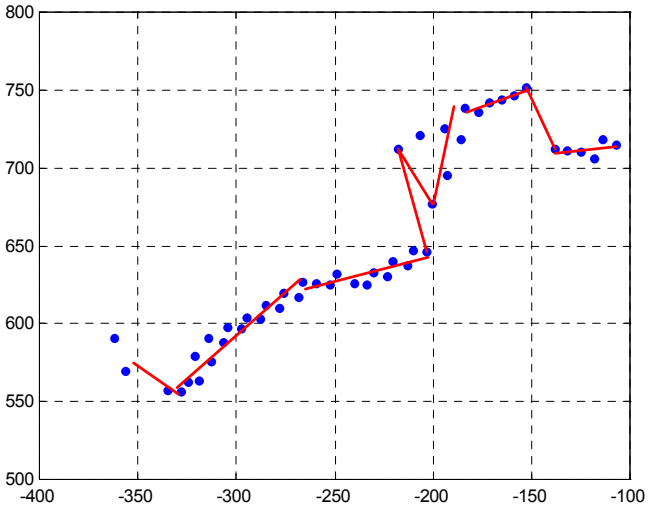


Figure 7.22: Line detection failing to some degree (II)

7.2.1.2 Circle Extraction

The circle extraction techniques proved very effective in identifying circles provided that the cluster was correctly identified as a circle. In some instances very short clusters that represented very short lines were identified as circles and the circle extraction algorithm fitted a bad circle. In most other cases however the circle extraction method discussed in chapter 5 performed very well. Figure 7.23 to Figure 7.25 shows the performance of the circle extraction method under different scenarios.

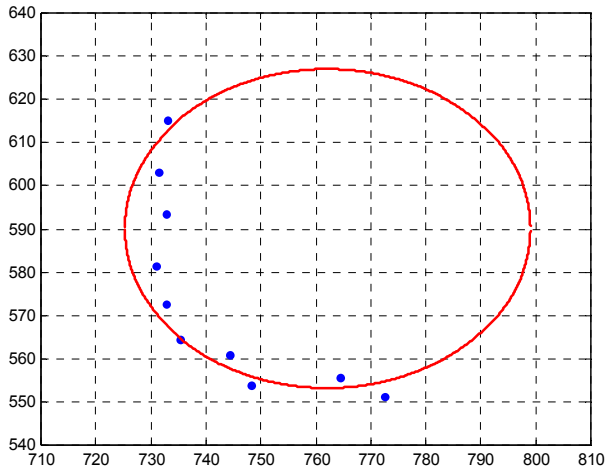


Figure 7.23: Accurate circle extraction

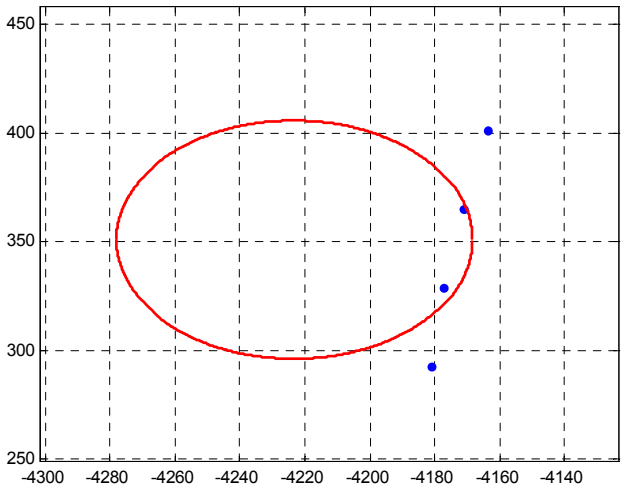


Figure 7.24: Extracting an erroneous circle since the data should be a short line

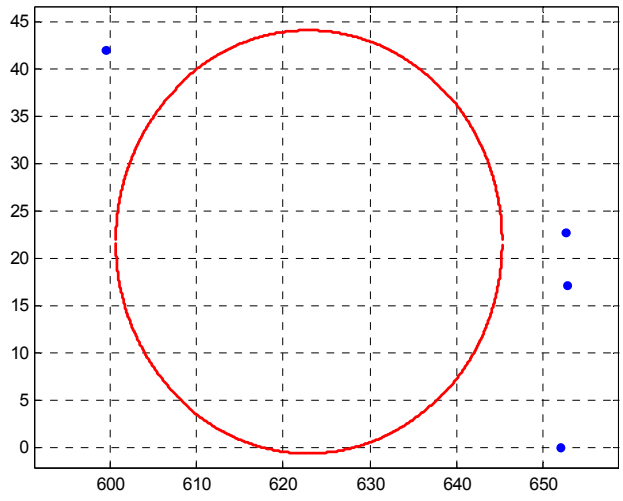


Figure 7.25; Extracting an erroneous circle since the data should be two circles

7.2.2 Data Association

Results of the maximal common subgraph data association techniques presented in chapter 5 are shown in this section. To verify these techniques the pose changes for two consecutive scans was calculated and then the second scan was moved to in the same reference frame as the first. Ideally the two scans would then be identical.

Figure 7.26 to Figure 7.31 shows three examples of consecutive laser scans and the results of the data association methods using the maximal common subgraph method. In each figure the red scan represents the current vehicle scan and the blue represents the previous scan. The features in red are matched to the features in blue using the data association techniques discussed in chapter 5. The pose transformation is calculated and the previous scan is then rotated to the same reference frame as the current scan to verify that correct features were matched and a correct pose transformation extracted.

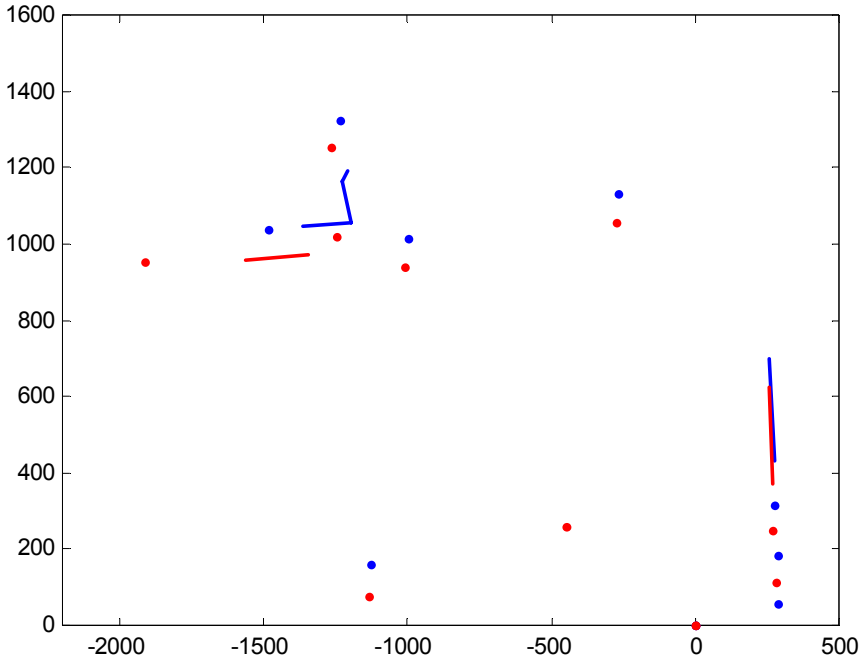


Figure 7.26: Consecutive scans from the laser in the Seymour centre car park after feature extraction. Red scan is the current scan blue scan the previous. Dots represent circle centres.

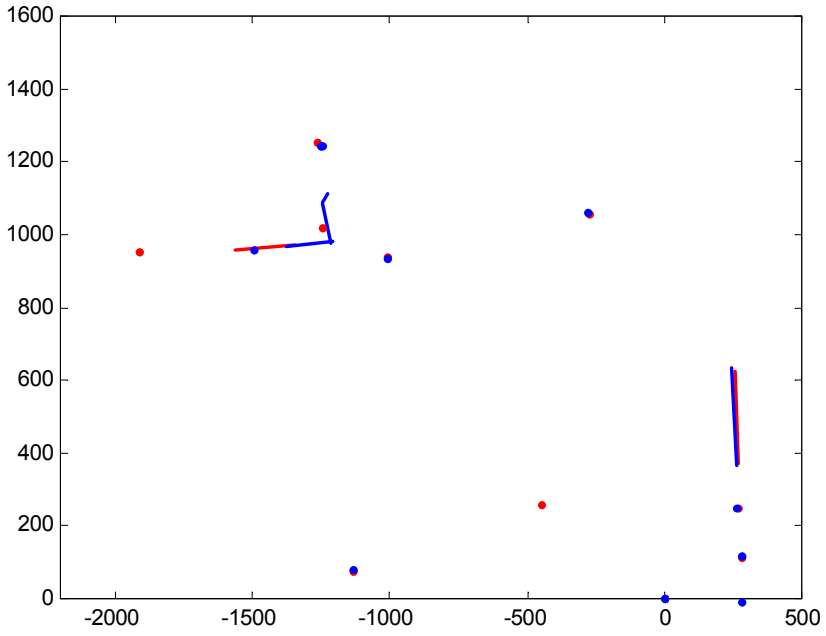


Figure 7.27: Scan from Figure 7.26 after data association. Features translated so both scans are in the same frame of reference. *(Tx = 7.4cm Ty = 68.9cm ω = -0.41°)*

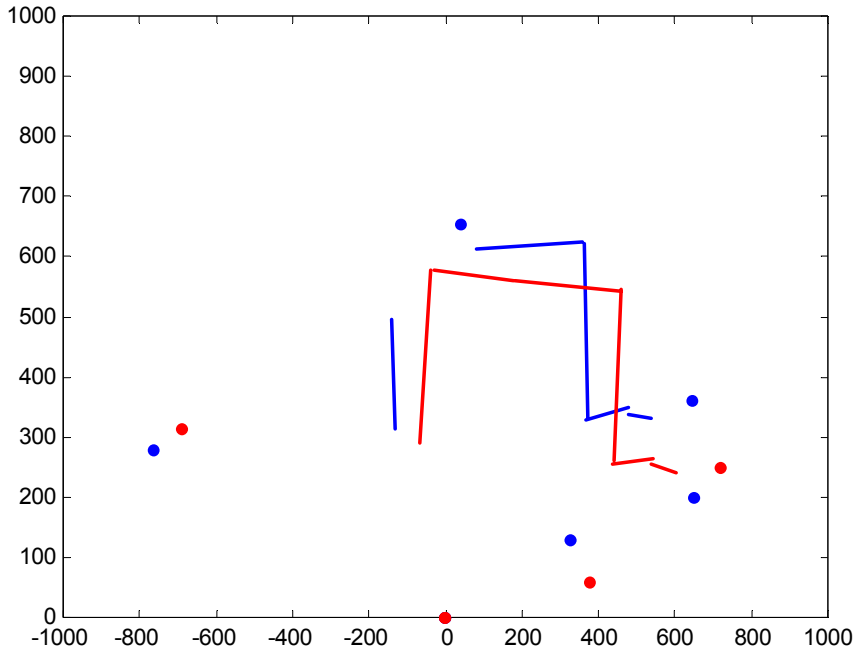


Figure 7.28: Consecutive laser scans from Seymour centre car park after feature extraction.

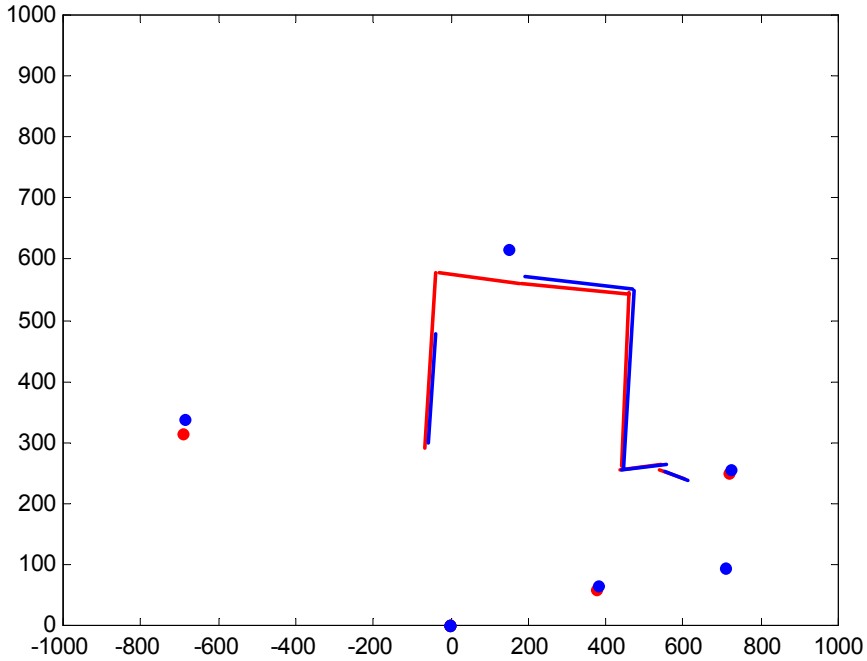


Figure 7.29: Data from Figure 7.28 after data association. *(Tx = -39.2cm Ty = 28.3cm ω = 6.6°)*

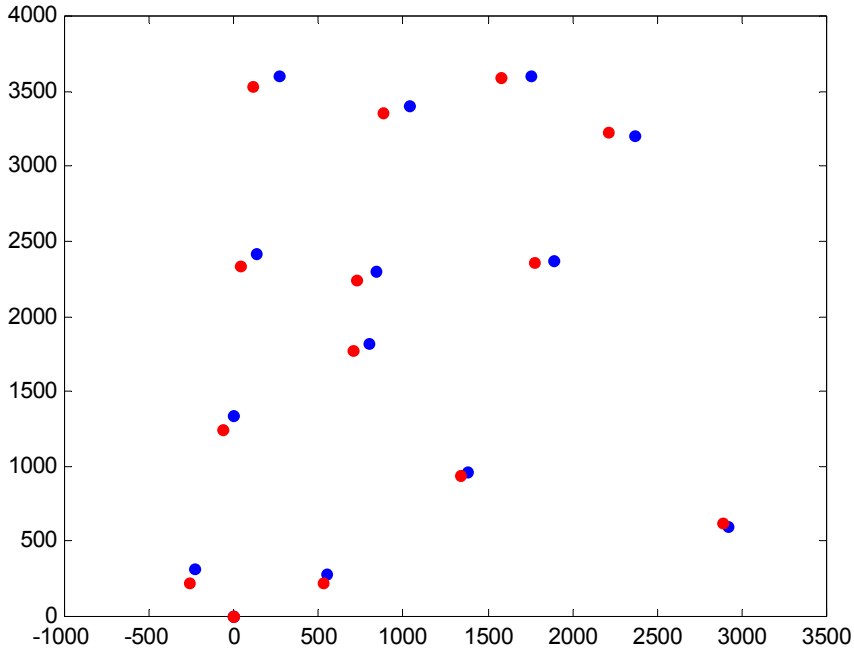


Figure 7.30: Consecutive laser scans taken from Victoria Park after feature extraction. Red scan is the current scan. Blue the previous scan. Dots represents circle centres.

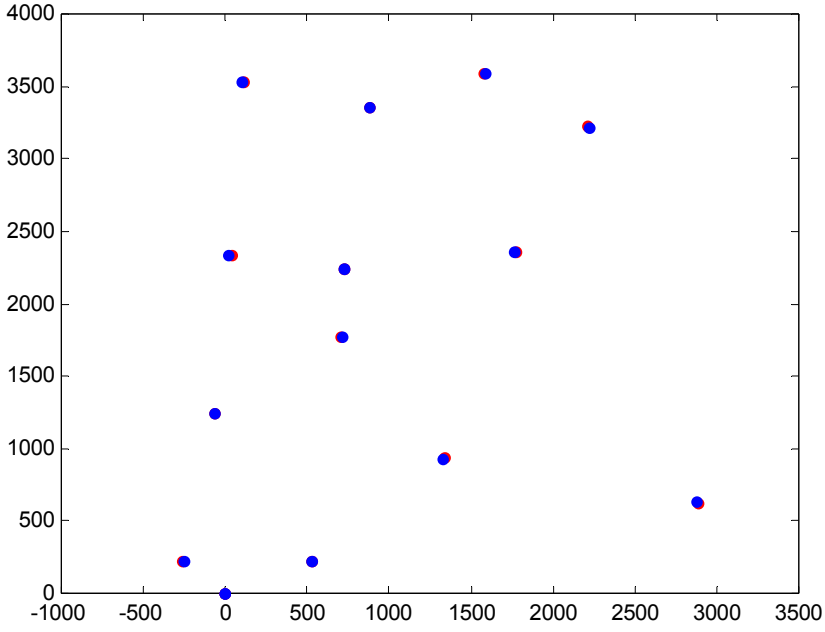


Figure 7.31: The scan of Figure 7.30 after the data association. The previous (blue) scan has been moved to the same frame of reference as the current scan. (*Tx = 12.33cm Ty = 85.6cm* $\omega = -2.47°$)

The results indicate that the maximal common subgraph method for data association is quite effective in determining a one to one matching of features between consecutive scans and hence the pose change of the vehicle based on this information.

These pose changes were then summed and converted to the navigation coordinate frame in an attempt to build up a path estimate of vehicle motion. Figure 7.32 shows the path estimate built up from consecutive laser scans using the data set taken from the Seymour centre car park (Figure 7.4 shows the actual path taken). Only the initial stages of the dead reckoning are shown since after this the path estimates become somewhat unreliable.

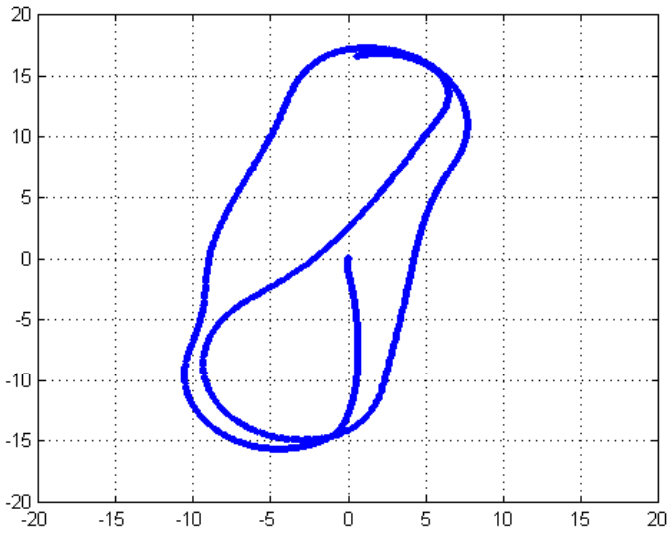


Figure 7.32: Laser Dead reckoning. Data set from seymour centre car park.

7.2.3 Summary

The techniques for data association and feature extraction presented here were quite effective in the sense that the feature extraction process reliably identified the correct features ion the majority of cases and the maximal common subgraph data association technique provided the correct feature mapping in the vast majority of cases.

Estimating the pose change from this feature matching information proved quite effective on an individual basis, i.e. simply matching two scans together. As can be seen from the

results the error was small when two scans were mapped onto one another. However when several feature mappings were strung together to form a path estimate of the vehicle the data became unreliable since all the small errors present just grew with time. As a consequence the laser dead reckoning path grew less and less accurate with time. As was to be expected since this was a dead reckoning path of the vehicle with no absolute information about the actual position.

It was hoped that the laser could be used as an alternative to the model for dead reckoning in situations where the GPS dropped out for extended periods of time. It was found that the laser dead reckoning was quite unpredictable and its performance could not be guaranteed reliably for all data sets. Also due to the time constraints of this thesis, the method was unable to be fine-tuned to produce a more robust algorithm.

Despite the relative failure of the using the laser to perform robust dead reckoning predictions effective feature extraction and data association techniques were implemented that could find many uses outside the scope of this thesis in areas of mapping and reactive control.

Chapter 8:

Conclusion

The kalman filter presented in this thesis to fuse vehicle state prediction based on encoder and LVDT data with GPS data performed reliably well in a number of different scenarios. A suitable filter tuning was to provide accurate position estimates when using the cm accuracy provided by the DGPS. Filter tuning was also provided to deal with the much less accurate GPS data when not using DGPS. Multipath was a common problem and testing showed that the filter was able to handle reasonable multipathed data. The online performance of the filter verified the testing performed on the offline simulation.

The ideal situation for the filter is a wide-open area with no trees or even a sparse covering of trees. The area must be relatively flat however since the navigation loop only gives position data in 2 dimensions. If hilly areas are involved then alternate strategies must be used such as an inertial system. Although the filter can handle the odd occurrence of corrupt GPS data, the filter still requires that on the whole the GPS information be good. Most importantly it is necessary to start the filter in an area where the GPS information is known to be accurate.

The feature extraction techniques implemented in this thesis proved to be very accurate in extracting features from raw laser data and even in these case where the features were not extracted entirely correctly the problem was not continued by the data association process.

The data association method implemented using a maximal common subgraph method to find the one to one feature mapping between scans was very effective in mapping features between consecutive scans. Initially it was postulated that laser based dead reckoning may provide a more accurate dead reckoning procedure since it is non model based. However the method proved to be often erratic if the correct number of feature were not matched between scans to compute a valid pose transformation.

Although the laser dead reckoning was somewhat inaccurate the data association method used could have many other potential applications such as tracking features in slam or in map building and exploration strategies.

Future work may include developing a three dimension navigation loop using an inertial unit, or using the navigation loop developed as an input to various control modules such as path following.

Chapter 9:

Bibliography

[1] Bailey T., Nebot E., Rosenblatt J., Durrant-Whyte H., *"Data Association and mobile Robot Navigation: A Graph Theoretic Approach",* ICRA 2000, SF, USA, April 2000

[2] Barton M, *"Controller Development and Implementation for Path Planning and Following in an Autonomous Urban Vehicle"*. University of Sydney, Undergraduate Thesis 2001

[3] Besl PJ, McKay HD, *"A method for registration of 3-D shapes"*, IEEE Transactions on pattern analysis and machine intelligence, vol 14 issue 2, Feb 1992, pgs 239-256.

[4] Bomze, IM, Budinich M, Pardalos PM, and Pelillo M, *"The maximal clique problem"*, Handbook of Combinatorial optimization (supplement volume A)m in D-Z Du and P.M. Pardalos (eds.) Kluwer academic publishers, Boston, MA, 1999

[5] Bron C., Kerbosch J., *"Algorithm 457 – Finding all cliques of an undirected graph*", Communications of the ACM, vol 16, pp 575-577, 1973.

[6] Deitel H, Deitel P, *"C: How to program"*, Prentice Hall 1994.

[7] Farrel J., Barth M., *"The Global Positioning System and Inertial Navigation",* McGraw Hill, 1999

[8] Fitzgibbon T., *"Remote Graphical User Interface for High Speed Vehicle"* University of Sydney, Undergraduate Thesis 1999

[9] Givant J, Baiker S, Nebot E, *"Localisation and map building using laser range sensors in outdoor applications"* Journal of robotic systems 17(10), pg 565-583, 2000

[10] Isikyildiz G *"The Hardware and Software Link Drivers for HSV"* University of Sydney, Undergraduate Thesis 1999

[11] Lee K, *"Reactive Navigation for an autonomous outdoor vehicle"*, thesis undergraduate university of Sydney 2001.

[12] Lu F., Milios E., *"Robot Pose estimation in unknown environments by matching 2D range scans",* Proc. IEEE Comp. Soc. Conf. On computer vision and pattern recognition, Seattle, WA (June 1994)

[13] Mallet A., *"A specification of generic robotics software components: future evolutions of GenM in orocos context"* LAAS-CNRS, to be presented at ICRA 2002

[14] Mallet L, *"Autonomous Rover Navigation on Unknown Terrains Functions and Integrations"* LAAS, To be published in International Journal of Robotics.

[15] McGregor J. J., *"Backtrack Search Algorithms and the maximal complete subgraph problem",* Software – Practice and Experience, vol 12, pp23-34, 1982

[16] Nebot E., Bailey T., Guivant J,. *"Navigation Algorithms for autonomouse machines in off road applications"* Journal of autonomous robots. Downloaded off the www at: www.acfr.usyd.edu.au

[17] Nebot E., Scheding S., "*Navigation System Design", Course notes for Navigation Systems Design (KC-4) ACFR,* March 27 2001 version 0.1

[18] Selvanathan A., *"Australian Business Statistics",* South Melbourne, Vic. Nelson, 2000, 2nd ed.

[19] ShouryA., Aboutabl M., *"Neural Network approach for solving the maximal common subgraph problem"* IEEE Transactions on systems, man and cybernetics, vol 26, no. 5, pp 785-790, 1996

[20] Sukkarieh S, Nebot EM, Durrant-Whyte H, *"Achieving integrity in INS/GPS Navigation loops for autonomous land vehicle applications"*, proceeding of the 1998 IEEE international conference on Robotics and Automation, Leuven Belgium 1998.

[21] Sukkarieh S., *"Introductory Estimation and the Kalman filter*", Mech 4701 Modern estimation and control lecture notes 2002.

[22] Sukkarieh S., Nebot E.M, and Durrant-Whyte H.F., *"A high integrity IMU GPS navigation loop for autonomous land vehicle applications"*, IEEE Transactions on Robotics & Automation, vol 15, no. 3, Jun 1999 pg 572-578

[23] Van Wyk, C.J., *"Data Structures and C programs"* Reading, Mass: Adison Westley, 1988

[24] Welch G, Bishop G, *"An introduction to the Kalman filter",* Department of computer science, University of North Carolina at Chapel Hill. Downloaded of the www at: http://www.cs.unc.edu/~welch/kalman/

[25] William H., *"Numerical Recipes in C"* Cambridge university press 1988, New York.

Abbreviations

DGPS – Differential Global Positioning System

GPS – Global Positioning System

HSV – High Speed Vehicle

IMU – Inertial Measurement Unit

LVDT – Linear variable displacement transducer

SLAM – Simultaneous Localisation and Mapping

ICP – Iterative Closest Point

IMRP – Iterative Matching Range Point

IDC – Iterative Dual Correspondence

ECEF – Earth Centred Earth Frame

NED – North East Down

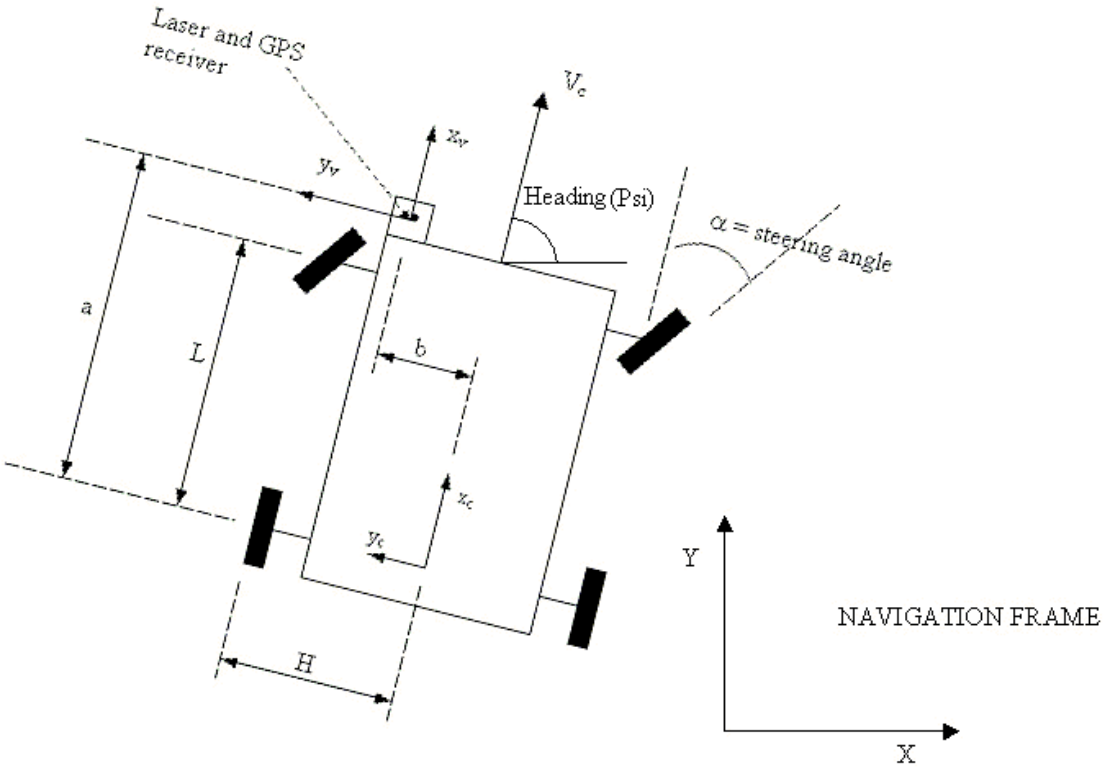
GUI – Graphical User Interface

QNX – A real time operating system

MFC – Microsoft Foundation Class

Appendix A

Derivation of Dead reckoning equations



The continuous differential equation to describe the system is given by:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} v_c \cos\phi \\ v_c \sin\phi \\ \frac{v_c}{L}\tan\alpha \end{bmatrix}$$

Where:

v_c is the velocity at the centre of the rear axle

Since the velocity is measured at the location of the encoder (v_e), rear left wheel, the velocity must be moved to the centre of the rear axle (v_c) using the following equations.

From simple geometry we can see the position of the centre of the rear axle (P_{Cx},P_{Cy}) with respect to the wheel encoder (P_{Ex}, P_{Ey}) is:

$$\begin{aligned} P_{Cx} &= P_{Ex} + H \sin \phi \\ P_{Cy} &= P_{Ey} - H \cos \phi \end{aligned}$$

Differentiating this to get V_c gives:

$$\begin{aligned} \dot{P}_{Cx} &= \dot{P}_{Ex} + H\dot{\phi}\cos\phi \\ \dot{P}_{Cy} &= \dot{P}_{Ey} + H\dot{\phi}\sin\phi \end{aligned}$$

Given that $\dot{X}_c = v_{Cx} = v_c \cos(\phi)$ and $\dot{Y}_c = v_{Cy} = v_c \sin(\phi)$ and $\dot{\phi} = \frac{v_c}{L}\tan\alpha$, the above equations reduce to:

$$v_c = \frac{v_e}{\left(1 - \frac{H}{L}\tan(\alpha)\right)}$$

The translation from the centre of the rear axle to the GPS antenna is given by:

$$\begin{aligned} x_v &= x_c + a\cos\phi - b\sin\phi \\ y_v &= y_c + a\sin\phi + b\cos\phi \end{aligned}$$

Differentiating this gives:

$$\begin{aligned} \dot{x}_v &= \dot{x}_c - a\dot{\phi}\sin\phi - b\dot{\phi}\cos\phi \\ \dot{y}_v &= \dot{y}_c + a\dot{\phi}\cos\phi - b\dot{\phi}\sin\phi \end{aligned}$$

Combining this with the original differential equations gives:

$$\dot{x}_v = v_c \cos\phi - \frac{v_c}{L}\tan(\alpha)[a\sin\phi + b\cos\phi]$$
$$\dot{y}_v = v_c \sin\phi - \frac{v_c}{L}\tan(\alpha)[a\cos\phi - b\sin\phi]$$

The discrete version of the above equation is:

$$X(k+1) = X(k) + \Delta t\left(v_c \cos\phi - \frac{v_c}{L}\tan\alpha\left[a\sin\phi + b\cos\phi\right]\right)$$
$$Y(k+1) = Y(k) + \Delta t\left(v_c \sin\phi + \frac{v_c}{L}\tan\alpha\left[a\cos\phi - b\sin\phi\right]\right)$$
$$\phi(k+1) = \phi(k) + \Delta t\frac{v_c}{L}\tan\alpha$$

Where:

X is the 'x' position of the vehicle in the navigation frame

Y is the 'y' position of the vehicle in the navigation frame

ϕ is the heading of the vehicle in the navigation frame

α is the current steering angle of the vehicle

Δt is the sampling interval of the system (25ms in this thesis)

v_c is the velocity at the centre of the rear wheels

L is the wheelbase of the HSV

a is the distance between the rear axle and the GPS receiver

b is the distance between the centreline of the vehicle and the GPS receiver

Appendix B

Closed-form Solution for Point Based Matching

For *n* pairs of points: $P(x_i, y_i), P'(x_i', y_i'), i = 1,...,n,$ a distance function between the transformed P's and the P'/'s is defined as the following:

$$E_{dist}(\omega, T) = \sum_{i=1}^{n} | R_\omega P + T - P' |^2$$

$$E_{dist}(\omega, T) = \sum_{i=1}^{n} \left((x_i \cos\omega - y_i \sin\omega + T_x - x_i')^2 + (x_i \sin\omega + y_i \cos\omega + T_y - y_i')^2 \right)$$

By minimizing E_{dist} we can obtain a closed form solution for Tx, Ty, and ω as given below:

$$\omega = \arctan \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}}$$

$$T_x = \bar{x}' - (\bar{x}\cos\omega - \bar{y}\sin\omega)$$

$$T_y = \bar{y}' - (\bar{x}\sin\omega + \bar{y}\cos\omega)$$

where:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\bar{x}' = \frac{1}{n}\sum_{i=1}^{n} x_i' \qquad \bar{y}' = \frac{1}{n}\sum_{i=1}^{n} y_i'$$

$$S_{xx'} = \sum_{i=1}^{n} (x_i - \bar{x})(x_i' - \bar{x}') \qquad S_{yy'} = \sum_{i=1}^{n} (y_i - \bar{y})(y_i' - \bar{y}')$$

$$S_{xy'} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i' - \bar{y}') \qquad S_{yx'} = \sum_{i=1}^{n} (y_i - \bar{y})(x_i' - \bar{x}')$$

Note also the coordinate transform to move the 2nd scan to the same frame of reference as the first is given by the equations below.

Let P1 be a point on scan 2 and P2 be the corresponding point on scan 1, i.e. they both represent the same point in the world. The two points are related by:

$$P2 = R_\omega P1 + T$$

where:

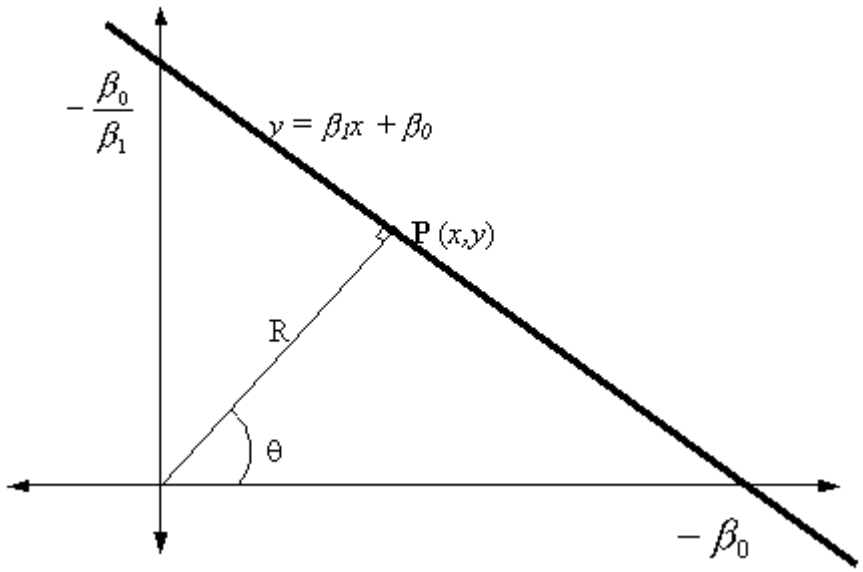
$R_\omega = \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix}$ is the rotation matrix

$T = \begin{pmatrix} T_x \\ T_y \end{pmatrix}$ is the translation vector

Appendix C

Polar Line Equations

Given the equation of a line in the form $y = \beta_1 x + \beta_0$ the polar parameters can be calculated as follows.



Form a line, perpendicular to the line of interest, that intersects with the origin. In the figure above this is a line connecting point **P** with the origin. The equation of this line is given by:

$$y = -\frac{1}{\beta_1} x$$

Point **P(**X_P, Y_P) is given by the simultaneous solution of the two line equations

$$y = -\frac{1}{\beta_1} x \quad \& \quad y = \beta_1 x + \beta_0$$

$$x_P = -\frac{\beta_1 \beta_0}{1 + \beta_1^2}$$

$$y_P = \beta_1 x_P + \beta_0$$

Having found the coordinate of point P the polar line parameters are simply given by simple trigonometry.

$$\theta = \text{atan2}\,(y_P, x_P)$$
$$r = \beta_0 \cos\theta$$

The above equations apply for a regression from y to x. For regression from x to y simply reverse x_p and y_p.

Appendix D

Finding all cliques of an undirected graph

The algorithm for finding all cliques of an undirected graph is presented in Bron & Kerbosch [5] and is outlined in this appendix. For more details consult Bron and Kerbosch.

The input graph is in the form of a symmetrical (*m* * *m*) matrix where *m* is the number of nodes in the graph. A 1 in entry *(i,j)* indicates that node *i* and node *j* are connected. The diagonal entries must all be 1.

The recursive algorithm is outlined below where the set *CompSub* indicates all possible nodes for the current clique, the set candidates indicates all points that may in time serve as extensions to *CompSub* and the set *Not* indicates all nodes that cannot be part of the current clique.

Global integer **N** Comment: **Size of the graph, number of nodes**
Global integer **c**

Comment: **The input graph, with N nodes, in the form of a connection matrix Connected**
Global boolean Matrix: **Connected[N][N]**
Global integer Array: **CompSub[N]**

```
Function FindMaxClique()
        integer Array ALL[1:N]
        integer i




        For i=1, step until i=N do
                ALL[i] = i
        end for




        c = 0




        ExtendVersion2(ALL, 0, N)



End FindMaxClique
```

Comment**: ExtendVersion2 is recursive the function to find all cliques of the graph Connected.**

```
Function ExtendVersion2(integer Array Old, integer ne, integer ce)
        integer Array new[1:ce]
        integer nod, fixp, newne, newce, i, j, count, pos, p, s, sel,
        integer minnod




        minnod = ce; nod = 0




        for i = 1 while i <= ce and minnod != 0  do
                p = old[i]; count = 0; j = ne;
```

```
            for j=j+1 while j <= ce and count < minnod do
                        if connected[p,old[j]] == 0 then
                                    count = count +1
                                    pos = j
                        end if
            end for

            if count < minnod then
                        fixp = p; minnod = count
                        if i <= ne then
                                    s = pos
                        else
                                    s = 1; nod = 1;
                        end if
            end if
end for

for nod = minnod + nod step –1 until nod = 1 do
            p = old[s]; old[s] = old[ne+1];
            sel = p; old[ne+1] = p;
            newne = 0;

            for I = 1 while I <= ne step 1 do
                        if connected[sel, old[I]] == 1 then
                                    newne = newne+1; new[newne] = old[I]
                        end if
            end for

            newce = newne;

            for I = ne+1 while I <= ce step 1 do
                        if connected[sel,old[I] == 1 then
                                    newce = newce+1;l new[newce] = old[I]
                        end if
            end for
```

```
            c = c+1; compsub[c] = sel;




            if newce == 0 then
                  Comment: Compsub contains a clique, save this clique
                  to check later if this clique is the maximal clique
            else if newne < newce then
                  ExtendVersion2(new, newne, newce)
            end if




            c = c-1
            ne = ne+1
            if nod > 1 then
                  s = ne;
                  s = s+1;
                  while Connected[fixp, old[s]] == 0 do
                        s = s+1
                  end while
            end if
      end for



end ExtendVersion2
```