VIRTUAL ROAD SIGNS

A thesis presented in partial fulfilment of the requirements for the degree of

Masterate of Technology

in

Computer Systems Engineering

at Massey University, Palmerston North, New Zealand.

Rhys David Nicholls

2001

ABSTRACT

Conventional road signs are subject to a number of problems and limitations. They are unable to disseminate dynamic information to road users, their visibility is heavily dependent on environmental conditions, they are expensive to maintain and frequently the target of vandals and thieves. Virtual road signs (VRS) differ from conventional signs in that they exist only in an information database – no physical signs exist on the roadside. By projecting virtual signs into a driver's field of view at the correct time, virtual road signs attempt to mimic conventional road signs. In addition, their visibility is independent of weather and traffic conditions, they can be tailored to specific driver and vehicle needs (such as truck drivers), and they cannot be vandalised like physical signs.

This thesis examines many of the major technical design decisions that must be made in implementing a virtual road sign system. A software prototype was designed and written to implement an experimental VRS system. The prototype served as a testbed to assess the technical feasibility of a VRS system and investigate alternative VRS designs. One limitation of the project was the lack of suitable display device that could display virtual signs inside a vehicle in real-time. Therefore, this project examined only the proof-of-concept. A test world was created around a university campus in which virtual signs were 'erected' to target a visitor to the campus. The prototype used a handheld GPS receiver to track a vehicle as it was driven around the campus. A Kalman filter was implemented to filter the GPS data and predict the motion of the vehicle when GPS data was unavailable. A laptop PC provided onboard processing capability inside the test vehicle. The prototype shows that technical implementation of virtual road signs is potentially feasible, subject to limitations in current display devices such as heads-up displays. Potential applications include signs custom-designed for tourists to indicate places of interest, bilingual signage, and aiding co-drivers in rally car driving. Before large-scale implementation can be considered, however, much research is needed, particularly with respect to systems acceptability to the public and road authorities.

This thesis is dedicated to my family,
who offered support, inspiration and
assistance as it progressed.

DECLARATION

I declare that this is my own, unaided work. It is being submitted for the degree of Masterate of Technology at Massey University. It has not been submitted before for any degree or examination in any other University.

Rhys Nicholls
28th day of February, 2001

ACKNOWLEDGMENTS

I would like to thank the following people for their assistance during the course of the research described in this thesis. Their contribution to this work is gratefully appreciated.

Dr Donald Bailey, for providing advice, guidance and comments which were extremely useful during the research, and who suggested a number of useful improvements to this thesis as it progressed. Donald's great sense of humour and never-ending jokes kept the spirits high at many of the meetings we had.

Dr Huub Bakker, for convincing me to delay my debut to the workforce for another year and return to work on this research. Huub also provided a number of useful comments regarding the material in this thesis.

My brother, Paul, for drawing a beautifully artistic plan view of a car, which appears in many of the report diagrams.

Veeral Patel, for assisting with the first experiment undertaken to test the software prototype.

Ken Mercer and Colin Plaw, for providing technical assistance during prototype development.

Alex Rowe, for providing the scale paper plot of the Massey University Ring Road, which was needed during prototype testing.

PREFACE

I first thought of the idea of virtual road signs while reading a newspaper some time ago. A current topic in the news at that time was the proposition of adding the Maori translation of road sign messages and placenames to the existing sign infrastructure. Advocates claimed that it is a Maori right to have bilingual signs, and that current signs are discriminatory. While bilingual signage seemed like a good idea to many, critics claimed that the sheer size of the task would make it prohibitively expensive and impractical. Only 15% of the population of New Zealand are of Maori ethnicity, and it is likely that a similar percentage can read the Maori language. Therefore, any campaign to duplicate sign messages would arguably apply to a small proportion of the total population.

Indeed, many countries in the world speak more than one major language, and it is desirable to display road signs in all of these languages. The use of internationally standard symbols overcomes the language barrier to some extent, but there are inevitably many signs that still use text to convey their message. Even if it is practical from a physical and economic point of view to express a sign's message in multiple languages, there will always be tourists who cannot read road signs adequately. A better solution is to somehow display a sign's message to suit the needs of individual drivers and their vehicles.

While reading the newspaper, I happened to think back to an overseas holiday to England with my family in 1993. One of the things that struck us about travelling on the roads in England, apart from the congestion, was the brilliant road sign infrastructure. Being unfamiliar with the road network, we were continually reliant on road signs for navigation. The signage was impeccable, with the layout of every major intersection displayed on huge signs well before the intersection was reached. The fact that I can only remember getting lost once is testimony to the excellent signage. I then thought of the situation in New Zealand, where there is only one major state highway, and signs are relatively sparse in comparison. The situation is gradually improving, but many will argue that there simply isn't the money nor the traffic volume in many areas of the country to justify the sign density seen in England. How can technology be applied to suit individual drivers in specific conditions, as well as create new signs at a practical cost? In the future, virtual road signs might just be one solution.

The motivation for this thesis is a vision in which virtual road signs are reality, a vision where virtual road signs contribute to improved road safety. Certainly, the idea of virtual road signs is not currently practical, even on a small scale. But one day, the required technology may develop to an extent that makes virtual road signs a feasible idea, rather than a figment of one's imagination. As we worked on the project described in this thesis, the comments we received from people who inquired about the project were very encouraging. At first, it was expected that people would find the virtual road sign idea too futuristic and approach the idea with some scepticism. But for most people, this was not the response received at all. They could actually see benefits of virtual signs, and wanted to know more about how they might work. A truck driver pointed out how roadworks signage often does not give the drivers of heavy vehicles enough time to slow down safely. He could see how virtual signs could help him. Other people were simply glad to hear of the prospect of extra signage.

In New Zealand, the penetration of automotive navigation systems is very low in private cars. In many overseas countries such as Japan, the United Sates, and many European countries, navigation systems are commonplace. New systems are being continually developed and released to cater for every information need of drivers, not just their need for navigation aids. Latest systems allow passengers to check their email, review the latest stock prices and surf the Internet. It seems only a matter of time before vehicles will be able to walk the dog and do the banking. It seems ridiculous to think that the vehicle manufacturers haven't already considered the idea of virtual road signs. Perhaps they have, and there are real reasons why they may not be practical. Or, perhaps they are simply waiting for the right technology. Perhaps drivers are not ready for the idea of virtual road signs yet. Time will tell.

The structure of this thesis is described as follows. Chapter 1 sets the scene by describing some of the problems and limitations of conventional road signs. The idea of virtual road signs (VRS) is introduced as one possible way of overcoming many of the problems associated with conventional road signs. The broad topic of augmented reality is introduced, which describes the way in which virtual signs are presented to drivers. The types of display commonly used for augmented reality are described, with particular reference to *heads-up displays*. A heads-up display is considered to be the most suitable device currently available for the display of virtual signs. As such, it is referred to frequently throughout this thesis. Some pros and cons expected of virtual road signs are listed, and the prototype software developed for this project is then introduced. Current traveller information systems are reviewed, followed by a review of some popular visual displays used by current systems to provide drivers with navigation and en-route trip information. The chapter concludes with a closer examination of some of the factors affecting perception of heads-up displays, in order to assess the suitability of such a display for virtual signs. Chapter 2 begins with an overview of the three functional components of a general VRS system. It then summarises the design and operation of the VRS prototype developed for this project. The coordinate systems of the prototype are briefly described, followed by a brief discussion of how these coordinate systems may be adapted for general VRS systems that cover a very large area.

Chapters 3-6 describe the three main subsystems of a general VRS system. With the exception of Chapter 6, each chapter begins by outlining some of the design issues that must be investigated when designing a general VRS system. The second part of each chapter is devoted to a discussion of the specific design decisions that were made for the VRS prototype. Chapter 3 addresses the positioning and tracking subsystem, which is responsible for measuring the location of a vehicle in the real world. It begins by examining some of the properties of positioning systems likely to be needed by a VRS system. Two different positioning techniques are presented. An experiment is conducted to verify the real-world accuracy of one satellite-based positioning technique. The remainder of Chapter 3 describes issues of positioning and tracking for the VRS prototype.

Chapters 4-5 address the second major VRS subsystem, the data subsystem. Chapter 4 investigates issues relating to the design of virtual worlds in which virtual road signs are contained. It begins by describing the role and design of sign regions, which are areas in the world where virtual signs shall be displayed. A discussion of various methods of representation of a virtual world is then presented. The important task of determining which sign regions are currently associated with a vehicle's position is described. This includes a discussion of one possible memory caching strategy, which improves the real-time performance of a VRS system. The remainder of the chapter is concerned with the specific design decisions made for the VRS prototype. The first part of Chapter 5 investigates issues relating to the design and processing of the most important objects in a VRS system, virtual signs. The chapter begins by discussing some important design aspects of virtual signs, such as their presentation style and the design of the sign face. Next, a description of how virtual signs may be processed in software is given. Like the virtual world in Chapter 4, one possible memory caching strategy is described to improve performance. The specific design decisions for virtual signs are described for the VRS prototype. The chapter concludes by presenting the database structure of the prototype, which encompasses the material in both Chapters 4 and 5.

Chapter 6 addresses the third and final major VRS subsystem, the display subsystem. This is primarily concerned with the display of virtual road signs. The chapter launches straight into the specific implementation of the VRS prototype. The user interface of the prototype is described, followed by an explanation of how the display subsystem is initialised and how the rendering of signs is optimised to best accommodate the limited screen space. The algorithms responsible for allocating signs to screen locations are then described, with examples.

Chapter 7 explains how the VRS prototype was tested using real-world data. A test world was constructed and a target user selected for the purpose of testing. A set of features were identified that may be of potential interest to the target user, and the coordinates of each feature were measured. A set of virtual signs was then designed to indicate the presence of the chosen features as the target user drove around the test world. The chapter goes on to explain how a set of sign regions were computer-generated in order to avoid the tedium of manual calculation. The testing methodology is described, followed by presentation of major prototype results. The results are primarily concerned with the performance and accuracy of the positioning and tracking subsystem. The chapter concludes with a thorough discussion of the results of prototype testing.

Chapter 8 appraises all major design decisions of the VRS prototype. It begins with an assessment of the positioning and tracking subsystem. Issues of accuracy are highlighted, and an improved subsystem is recommended for a real VRS system, based on the prototype. A discussion of the data subsystem and display subsystem follows. One of the key issues discussed in this section is the excellent scalability of the prototype – that is, it is noted that the prototype can be extended to handle very large virtual worlds with minimal degradation of performance. The storage requirements of virtual sign data are estimated for much larger worlds than the world tested in Chapter 7. Several other high-level data management issues are considered, such as how virtual sign information may be kept up-to-date. The hardware costs of the prototype are estimated, and these are compared with the estimated costs of a real VRS system using current technology, and the estimated costs that consumers might be willing to pay for a VRS system. Several potential applications of virtual road signs are proposed, and several issues of systems acceptability of a VRS system to the public and authorities are highlighted. The chapter finishes with a brief description of opportunities for future research. Chapter 9 presents the major conclusions of this project and concludes the body of the report.

TABLE OF CONTENTS

An online (searchable) copy of this thesis may be found on the CD-R disc contained in the back pocket of this thesis.

LIST OF FIGURES

CHAPTER 1
Introduction

Imagine for a moment the following scenario. It is 5:30pm at the end of a busy weekday and the city traffic is heavy. The sky is darkening as night descends. Outside, torrential rain is falling and the wind batters your vehicle as you hurtle down the highway. A short time later, you round a bend and see that two vehicles have collided at an on-ramp. Normally, you would brake hard and struggle to control the vehicle in such conditions. But this time your mind is at ease. You have already slowed to a safe crawl and moved into another lane opened just minutes ago to divert traffic. As you pass the crash scene, you adjust your speed to the slippery conditions, as do surrounding vehicles. You have never travelled this route before, yet you know exactly where you are, and before long you are back on the beaten track. Then a sign emerges out of the gloom, and you cannot help but smile: dinner will be served in ten minutes, just in time for your arrival home.

How plausible is this scenario? In the year 2001, the opening scene will be familiar to all seasoned road users. Of these, a fair proportion of drivers in Europe, America and Japan will be accustomed to navigating their vehicle with the help of some automated navigation system. But the driver in this scenario was not focussing their attention on a conventional display panel inside their vehicle. As they drove along the highway, they were seeing computer-generated signs in their field of view that did not actually exist on the roadside. This is the idea of virtual road signs (VRS).

1.1) Conventional Road Signs

Road signs are a major source of short-term road and traffic information to road users. The variety of signs is enormous: some signs are permanent, such as state highway signs that give directions and distance to settlements; others are temporary, such as roadworks and accident signs. Traffic signs control the flow of traffic; automatic signs can switch on to warn road users of icy conditions; and illuminated variable signs can adapt to variable traffic conditions. Whether a sign is intended to alert the road user to convenience, danger, or the law, its purpose is the same: to provide road users with useful information to improve the safety and enjoyment of their journey.

Let us define a conventional road sign as one which is not automated in any way, such as a wooden or metal sign on the side of a road. Conventional road signs are subject to a number of problems and limitations:

1) Signs are capable of providing static, unchanging information only.

Permanent sign data must be approximated to cater for all categories of road user. For example, speed signs erected at the beginning of curved roads suggest the recommended speed at which to traverse the curve. Often this speed is calculated to account for the worst road conditions that can reasonably be expected on a regular basis; for example, driving at night in a light vehicle such as a family sedan, in wet weather. In dry conditions, it may be safe to navigate the curve at a speed 15 kilometres per hour faster; in a heavy vehicle such as a truck, a speed 20 kilometres per hour slower might be the maximum safe speed. To an overseas tourist or a learner driver, it may be unclear which category of road user these signs target. Recently, the idea of variable speed limits on highways in New Zealand has been proposed (NZPA, 2000 [a-b]). Some people argue that using static signs to enforce variable speed limits would confuse drivers more than they are worth.

2) Dynamic information cannot easily be disseminated.

The minimum age of messages conveyed by conventional signs is typically in the order of hours. For information that needs to be conveyed more urgently, conventional signs are not a practical option. Even if they could be erected in time, the problem of selecting a standard sign still exists, as construction of a custom sign would take far too long. Temporary signs are difficult and cumbersome to continually update.

3) The environment has a significant influence on sign visibility.

Bad weather can have a detrimental effect on the ability of a driver to observe road signs. Large vehicles can conceal a sign completely. This is a common problem with road markings at major urban intersections. For example, in queued traffic, a driver unfamiliar with the lane configuration of an intersection fails to see the road-painted lane arrows and is forced to assume the most likely configuration. Unfortunately, lane configurations are often inconsistent between intersections, as urban drivers will attest.

4) Signs are expensive, intrusive and bulky.

The construction costs of signs are high and increasing. Permanent road signs clutter the landscape. Temporary road signs are bulky and hazardous to carry, and their placement on the road is potentially hazardous to road users.

5) Sign maintenance is an ongoing problem.

Road signs are often the target of graffiti and theft, endangering road users and incurring an even higher cost. Recently, this point has been highlighted by a number of newspaper articles. In one instance, thieves removed protective barriers warning road users of a large hole in the road (Nash, 2000). Only the quick action of a passing police patrol avoided the occurrence of a serious accident. In another instance, the occupants of a vehicle were killed when they failed to stop at an intersection because the Stop sign had been stolen. The labour cost of repairing damaged and vandalised signs is a significant proportion of total sign cost.

1.2) What are Virtual Road Signs?

In the last decade, an explosion in the telecommunications industry and the continuing convergence of telecommunications, computers and consumer electronics has been the driving force behind society's thirst for up-to-the-minute information. *In-vehicle navigation systems* have evolved to help fill the niche of dynamic information dissemination to road users that is beyond the domain of conventional signs. In-vehicle systems retrieve information about the current traffic and road situation and present it to drivers on a small display device inside their vehicles. This device is one component of the *human-machine interface (HMI)*. Research shows that current HMIs are far from perfect from a safe driving point-of-view, and a number of improvements can yet be made. This report describes a project that investigates one such improvement: to implement a HMI that realistically renders computer-generated road signs directly into a driver's field of view. The signs mimic real road signs in their appearance, but because the signs do not actually exist on the roadside, the signs are described as virtual. A driver views their surrounding environment as per usual, with signs superimposed onto the image they perceive. This technique is one example of *augmented reality*.

By their nature, virtual road signs convey visual information to a driver. Drivers can usually obtain more than 90% of the information needed for the driving task through the faculty of vision (Kuroki and Asou, cited by Okabayashi and Sakata, 1991). The advantage of visual information over auditory information is that it can be 'self-paced'; that is, the driver can glean the information when the traffic situation allows. In this thesis, the proposed purpose of virtual road signs is to disseminate two specific categories of information to drivers:

1) Navigation information, which instructs a driver how to navigate their vehicle for the purpose of route guidance. This may be static (historic) or dynamic (for example, responsive to other traffic to avoid congested areas).
2) En-route trip information that may be of interest to a driver. This second category consolidates a very wide range of information, some of which is currently disseminated to road users by conventional signs, some by other means (for example, traditional dashboard indicators such as excessive speed, seat belt reminder, low fuel) and some which is not conveyed at all. There are two main information types:

a) Notification of service facilities, such as nearby hotels, restaurants, workshops or parking spaces (OECD Scientific Experts Group, 1988).
b) Hazard warnings, which concern transient problems or specific topics relating to road conditions, the current traffic situation, rule compliance or vehicle manoeuvres (OECD Scientific Experts Group, 1988). Examples are roadworks, accidents, ice.

1.3) Augmented Reality: The Next HMI Frontier

1.3.1) Overview

Section (1.7) reviews the presentation styles of current in-vehicle HMIs that provide navigation and en-route trip information to drivers. Most current in-vehicle HMIs use colour displays mounted on or near the dashboard, known as instrument-panel (IP) displays. Regardless of the actual display type utilised, two key issues of in-vehicle IP displays affect the safety of drivers and their vehicles:

1) Distraction. The presence of an animated display close to the driver often attracts the attention of the driver even when he/she has no intention of interpreting the display.
2) The continual need to divert attention back and forth between the in-vehicle IP and outside world. Systems that require long glance durations to assimilate the desired information from a display are dangerous. Treat *et al.* (cited by Ashby and Parkes. 1993) highlight improper lookout, excessive speed and inattention as the leading causes of accidents. Smiley and Brookhuis (cited by Ashby and Parkes. 1993) argue that attention and perception errors predominate over simple response errors in accident causation. For all the benefits that they bring, current in-vehicle instrument panel HMIs inevitably aggravate this problem.

The challenge is thus to investigate how navigation and en-route information can be presented in a more seamless manner that makes the combined tasks of driving and assimilating information safer. The primary objective of virtual road signs is to address this challenge.

Augmented reality (AR) is one research topic that is proposed as an ideal medium to realise virtual road signs in practice. AR is described by Groves (1999) as "virtual reality overlayed on reality: a system allowing you to see virtual world objects overlayed on your physical surroundings." It overlays a computer-generated image onto the real-world scene observed by the user to assist the user in their environment (see Figure 1.1). In short, AR is what makes virtual road signs 'virtual'. In this context, the driver is already scanning their environment through the vehicle windscreen, a natural driving activity. AR would superimpose virtual road signs into the driver's field of view. Careful sign design and positioning would ensure that the driver's view of the forward road scene was not impaired for the purpose of safe driving (see Figure 1.2 for an example).

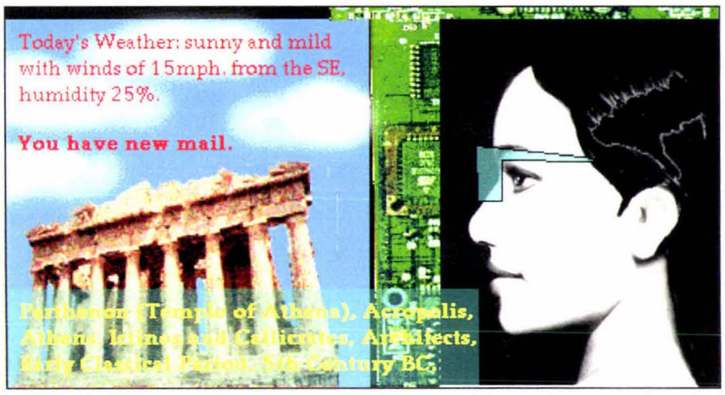

Today's Weather: sunny and mild with winds of 15mph. from the SE. humidity 25%.
You have new mail.
Parthenon (Temple of Athena), Acropolis, Athens, Ictinos and Callicrates, Architects, Early Classical Period, 5th Century BC.

Figure 1.1: Example of augmented reality: text is overlayed on a real world scene, as observed by a viewer (DeVaul, Rhodes and Schwartz, 2001)

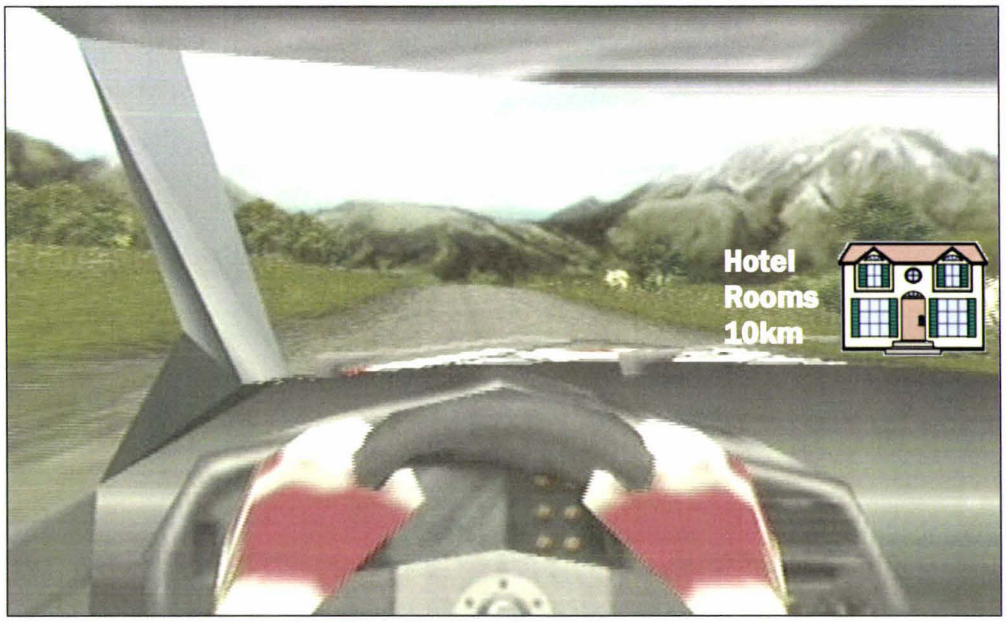


Figure 1.2: Example of how a virtual road sign might appear to a driver

By seamlessly augmenting a driver's forward view with computer-generated images, a driver no longer needs to divert their attention to a separate display of an instrument panel HMI. This reduced demand on the driver's focussed attention should translate into more time available for the driving task. The driver simply focuses on the task of scanning their outside environment as usual, periodically looking at the virtual road signs as they appear.

AR is perhaps the only way of implementing virtual road signs with sufficient realism. In theory, signs can be made virtually undiscernible from reality; that is, conventional road signs. If implemented correctly, virtual signs should present no more of a distraction to a driver than conventional signs. Another advantage is that, unlike current instrument panel HMIs, AR is semi-immersive: it does not completely surround the driver in an artificial world, yet it gives the impression that artificial images fit into the real world. AR can superimpose signs into the driver's field of view with an effective focal length of five metres or more, generating an illusion that the signs really do exist at some distance in front of the vehicle.

1.3.2) AR Display Devices

Current AR efforts use three categories of display device:

1) Video screens, in which live video input of a real scene is merged with live computer-generated imagery and the augmented result is viewed by the user on a computer monitor (Groves, 1999).
2) *Head-mounted displays* (HMDs), based on LCD, CRT or fibre optic technology (Pimentel and Teixeira, 1995). AR researchers have been working with two types of HMD: (a) *video see-through*, and (b) *optical see-through* (Vallino, 2000). The 'see-through' designation arises from the need for the user to be able to see the real world view that is immediately in front of them when wearing the HMD. A video see-through HMD uses video cameras that are aligned with the display to obtain the view of the real world. In this sense, a video see-through HMD is very similar to video screens described in (1), except that the user has a heightened sense of immersion. The optical see-through HMD eliminates the video channel that is looking at the real scene (Manhart, Malcolm *et al.*, cited by Vallino, 2000). Instead, the merging of real world and superimposed images is done by optics located in front of the user's eyes.
3) *Head-up displays* (HUDs). Automotive HUDs reflect a projected image off the inside windscreen surface of a vehicle into a driver's field of view.

Clearly, video screens described in (1) are much too bulky to display virtual signs inside a vehicle. The use of video see-through HMDs in vehicles is not practical because drivers want to see the real-world view with their own eyes. A HUD is considered a more practical option for displaying virtual signs because it does not require drivers to wear any headgear.

1.3.3) The Use of HUDs for Augmented Reality

HUDs were considered for widespread automotive use as early as 1985 (Enderby and Wood, 1992). HUDs have been used successfully in military jet fighters for many years to reduce the need of pilots to take their eyes off the airspace ahead; for example, during high-speed manoeuvring when timing is critical. Some advantages of automotive HUDs as a method of communicating information, such as that traditionally handled by the dashboard of a vehicle or navigation directions, are:

1) Reduced time to recognise the information content, compared to *heads-down displays* (HDDs) such as in-vehicle instrument panels. In one simulated experiment involving twelve drivers, the average response time to navigation information presented on a full-windshield HUD was 341 milliseconds faster than for an in-vehicle instrument panel (a 23% difference) (Steinfeld and Green, 1998).
2) Possible reduction of navigation error rates, compared to in-vehicle instrument panels (Green, Williams, Hoekstra, George and Wen, 1993, cited by Steinfeld and Green, 1998).
3) Reduced eyestrain, due to a shorter distance between the HUD image and the forward view of the outside world, compared to in-vehicle instrument panels (Todoriki, Fukano, Okabayashi, Sakata and Tsuda, 1994). The driver is not required to continually adjust their focal point from long-range (when scanning the outside environment) to short-range (this focal adjustment is known as *eye re-accommodation*).
4) Increased eyes-on-the-road time (Gish, Staplin, Stewart and Perel, 1999). In one experiment, Todoriki *et al.* (1994) found that the number of glances towards a HUD was significantly less, per minute, than for an in-vehicle HDD. They concluded that the application of a HUD to navigation systems could be expected to contribute to improved driver safety.

Given their advantages, integration of HUDs into automobiles has been slow over the last 15 years and driver acceptance remains low. There are several problems concerned with the readability of automotive HUDs which may help explain why this has been so. These include:

1) Contrast interference: difficulty seeing HUD symbology during the day due to lack of contrast between symbology and the background scene.
2) Increased driver workload required to interpret the HUD symbology while driving.
3) Propensity for capturing and holding driver's attention, away from the forward road view (Gish *et al.*, 1999).
4) A possible increase in navigation error rates, compared to in-vehicle instrument panels. In one simulated experiment involving twelve drivers, a full-windshield HUD produced almost twice as many errors as an in-vehicle instrument panel (Steinfeld and Green, 1998).

In addition, Gish *et al.* (1999) found no significant difference between the recognition rates of information presented on a simulated HUD, compared to a simulated HDD. In this experiment, the ability of 36 participants to detect targets external to a vehicle (such as pedestrians and other vehicles) and perform in-vehicle tasks (such as respond to navigation information) was tested. The HUD was mounted five degrees below the line of sight, whereas the HDD was mounted twenty degrees below the line of sight. Surprisingly, the recognition rates for the HUD and HDD were both very similar, although the response time of participants was slightly better for the HUD overall. This was true both when drivers performed one task at a time and when drivers were required to attend to dual tasks simultaneously.

Sviden (1993) describes one scenario that was predicted for the year 2000 in the early 1990s, as presented in Box 1.1. Although real HMIs like that described now exist, the level of ubiquity suggested by Box 1.1 has clearly not eventuated.

"Cars have now received an intelligent RSI (Road Service Informatics) visor, in front and above the driver. The RSI unit has display optics at its edge and can be adjusted in position and brightness to fit the individual drivers. The driver can see one row of symbols and a line of text above the traffic scene in front of him. He normally 'experiences' the symbols with his peripheral, rather than his direct, vision. The principal and standardized information on the RSI unit is combined with more detailed head-down information on the integrated dashboard displays."

Box 1.1: Implementation of in-vehicle HMIs, predicted by Sviden (1993) for the year 2000 (p30)

Even though there is no consensus as to the benefits of HUDs, a HUD would appear to be the most effective present-day device for displaying virtual road signs. The idea of using a projection device such as a HUD as an in-vehicle AR display device, specifically for the purpose of displaying virtual road signs that convey navigation and en-route trip information.

is not new (Chislenko, 1997; Spohrer, 1999). However, while plenty of ideas have recently been bandied about, few concrete implementations or prototypes appear to have been developed yet. The research project described in this thesis addresses this research opportunity.

1.4) Expected Benefits and Drawbacks of Virtual Road Signs

The true novel factor of virtual road signs is the style of information presentation. By combining this style of presentation with the functionality of current navigation systems, the expected benefits of virtual road signs are numerous. Box 1.2 describes some of these benefits with respect to the aforementioned problems associated with conventional signs, followed by some drawbacks of virtual road signs.

Benefits of virtual road signs:

1) Dynamic information can be communicated to road users very effectively:

   a) In real-time or near real-time. Sign messages can be easily updated in one or more databases as information comes to hand.

   b) On a per-user basis; for example:

   - Nearby landmarks or facilities e.g. supermarkets.
   - Reminder to take a break from driving on long trips.

   c) On a per-road basis:

   - Temporary road conditions e.g. roadworks, detours.
   - Adverse road conditions e.g. slippery surface (ice, oil), metal surface.
   - Adverse weather conditions e.g. snow, high winds, flooding, hot road shimmering.
   - Adverse traffic conditions e.g. vehicle collision, vehicle breakdown.

2) Dynamic and static information can be tailored to meet the needs of specific road user categories; for example:

   - Speed recommendations on curves and corners can be set separately for motorcycles, light vehicles (cars) and heavy vehicles (trucks) to suit traffic and weather conditions.
   - Learner drivers can receive additional driver assistance e.g. "Apply Brakes Now".
   - Vision-impaired drivers e.g. elderly drivers: virtual signs can be made as large as necessary.
   - Hearing-impaired drivers: aural information such as engine revs and ambulance sirens can be represented in visual form.
   - Emergency service drivers e.g. police, ambulance, fire: additional information can be provided for safer high-speed driving e.g. status of nearby intersections.
   - Tourists: a separate set of signs can indicate local attractions.

3) The visibility of virtual signs is excellent in all environmental and traffic conditions (subject to ambient light conditions), such as in heavy traffic and at night.

4) Construction costs are minimised as real materials are not needed. Signs are not manufactured in a physical sense.
5) Sign placement incurs a very low labour overhead as no physical erection is required. Signs can be easily "erected" in remote locations.
6) Virtual signs do not wear with time due to atmospheric exposure, nor is repair necessary.
7) Graffiti and theft is impossible, meaning a further cost saving in maintenance.
8) Signs have no impact on the surrounding environment. There is no chance of drivers colliding with virtual signs in an out-of-control situation. Environmental damage is non-existent.

Drawbacks of virtual road signs:

1) The possibility that virtual signs may cause excessive visual distraction while driving.
2) Issues of liability in case of system failure (for example, if the wrong signs are displayed).
3) Vehicles must be fitted with expensive system hardware to display virtual signs to drivers.

Box 1.2: Expected benefits and drawbacks of virtual road signs

Virtual road signs are presented here as an idea to supplement existing road signs, rather than replace them completely. Replacement of conventional road signs would only be feasible if all road users were provided with a means of viewing virtual road signs. Even if this becomes practical at some time in the distant future, issues of liability in the case of system failure reinforce the need for conventional road signs as a backup.

One of the attractions of virtual road signs is that even though the style of information presentation is quite different, many of the advantages of current in-vehicle HMIs apply, and thus many of the lessons learnt can also be applied. The information is there if and when the driver wants it, continually changing as a vehicle travels down the highway, and perhaps able to be edited and customised to a planned trip. Therefore, while the idea of virtual road signs is a potentially revolutionary idea with respect to in-vehicle HMIs, it can be considered an evolutionary phase in the development of in-vehicle HMIs overall.

1.5) Project Objectives

The objectives of the virtual road sign (VRS) project were:

1) To determine the technical feasibility of a VRS system, within the limits of current technology.
2) To investigate the major design decisions for a general VRS system.
3) To implement one particular VRS system, using real-world test data, for the purposes of demonstration.
4) To consider potential future applications and implications of the technology.

In order to fulfil these objectives, a working software prototype was developed. The prototype served as an ideal experimental project to brainstorm ideas, analyse and implement them quickly, and assess the result without commercial bias or burden. This report describes most aspects of the prototype, ranging from its high-level architecture down to lower-level details of implementation. The architecture and fundamental design decisions presented are intended to be sufficiently general such that they may be applicable to the development of other VRS systems. The final working prototype demonstrates an example of one possible VRS system implementation using a small set of real-world data. Although small scale and purely research-oriented, the prototype intends to spark the imagination with what might be possible on a larger scale; for example, a commercial end-user application.

The main limitation of this project was the unavailability of a suitable display device, such as a HUD, that could be incorporated inside a vehicle to present virtual signs to the driver. This meant that a full demonstration of a working VRS system was not viable. Therefore, the objective of the prototype was redefined: to assess the technical feasibility of virtual road signs by way of a restricted *proof-of-concept simulation*. The challenge was to decide on a practical design compromise that worked with available display devices, such as a computer monitor or a laptop LCD screen. One feasible option was to record the non-augmented view that would be observed by a driver using a video camera mounted inside the vehicle. On arrival back into the lab, virtual road signs could be superimposed onto the captured video and replayed to simulate the augmented driver's view. For example, Kim, Kim, Jang, Kim and Kim (1998) used this technique in an outdoors AR system that mounted a CCD camera on a radio-controlled helicopter. A tracking system measured the position and precise orientation of the camera, and virtual images were superimposed on the captured image in real-time using video overlay hardware. Another simpler option was to forego the video entirely and record only the vehicle tracking data during a journey. As we shall see, the lack of a suitable display device was successfully overcome.

1.6) Review of Intelligent Transportation Systems and Advanced Traveller Information Systems (ATIS)

1.6.1) Introduction

The phrase *intelligent transportation systems* (ITS) describes an incredibly diverse field combining advances in communication, tracking, information engineering, electronics and automobiles. The coupling of information technology and communications is increasingly known as *telematics*; therefore, ITS can be considered as the application of telematics to road transport. The goal of ITS is to apply technology to make transportation safer and more efficient, with less congestion, pollution and environmental impact (Zhao, 1997). The original phrase coined for ITS was *road transport informatics* (RTI), which later became known as *intelligent vehicle-highway systems* (IVHS) in the USA to embody the notion of an integrated system (Catling, 1994), and *advanced transport telematics* (ATT) in Europe. All of these acronyms remain valid and roughly synonymous with ITS. ITS has evolved from its most basic means back in 1910, when the Jones Live Map was advertised as a means of replacing paper maps, to a wide variety of modern-day systems that cater to every information need of drivers. A brief history is presented in Catling (1994) and Zhao (1997).

To organise ITS into manageable units, ITS activities are broken down into a set of user services (Drane and Rizos, 1998), such as:

1) Advanced traffic management systems (ATMS), which manage traffic on transport networks to reduce congestion and improve travel times.
2) Advanced traveller information systems (ATIS), which provide information directly to drivers, such as the best route to travel to reach a particular destination.
3) Commercial vehicle operations (CVO), which use automatic vehicle location systems linked with computer-aided dispatch systems to manage commercial vehicle fleets.
4) Advanced public transport systems (APTS), which are concerned with improving the efficiency of public transport information and control systems.
5) Automated highway systems (AHS), which use automated vehicle control systems to drive vehicles with minimal or no human intervention required (Whelan, 1995).

Virtual road signs provide *en-route driver information* (in contrast to pre-trip planning information) and belong in the ATIS category. ATIS is defined as systems that "acquire, analyze, communicate, and present information to assist surface transportation travellers in moving from a starting location (origin) to their desired location" (Whelan, 1995, p63). ATIS provides a variety of information that assists travellers in private vehicles or public transportation, such as location of traffic incidents, weather and road conditions, parking sites, optimal routes, recommended speeds and lane restrictions. One ATIS service is concerned with the task of *route guidance*. Route guidance is the process of guiding a driver along a predetermined route chosen because it is quicker, shorter or more convenient; *dynamic route guidance* uses real-time information such as traffic congestion to choose the route (Catling, 1994). The most basic envisaged use of virtual road signs is to apply virtual road signs to an existing route guidance system. As a driver travels along their chosen route, virtual road signs would display navigation instructions to keep the driver on track and up-to-date. A more general VRS system would not limit itself to the display of navigation information; rather, it would use virtual signs to convey almost any type of visual information to the driver.

Figure 1.3 shows the main components of a vehicle-based ATIS. There are two potential sources of information:

1) Relatively static information, including internal data, such as a description of the vehicle, and external environmental data, such as local road and intersection layout. Because this data changes infrequently, it can be stored on one or more onboard databases inside a vehicle. Typically, road layout for a defined area, such as a city, is stored in its own digital map database.
2) Relatively dynamic information, such as current lane status (open/closed) on a highway. Because this data changes frequently, it makes sense to transmit it from one or more offboard (remote) data stores, maintained by a variety of information providers, when it is needed in real-time. An ATIS-enabled vehicle receives the data via a wireless communication system such as radio.

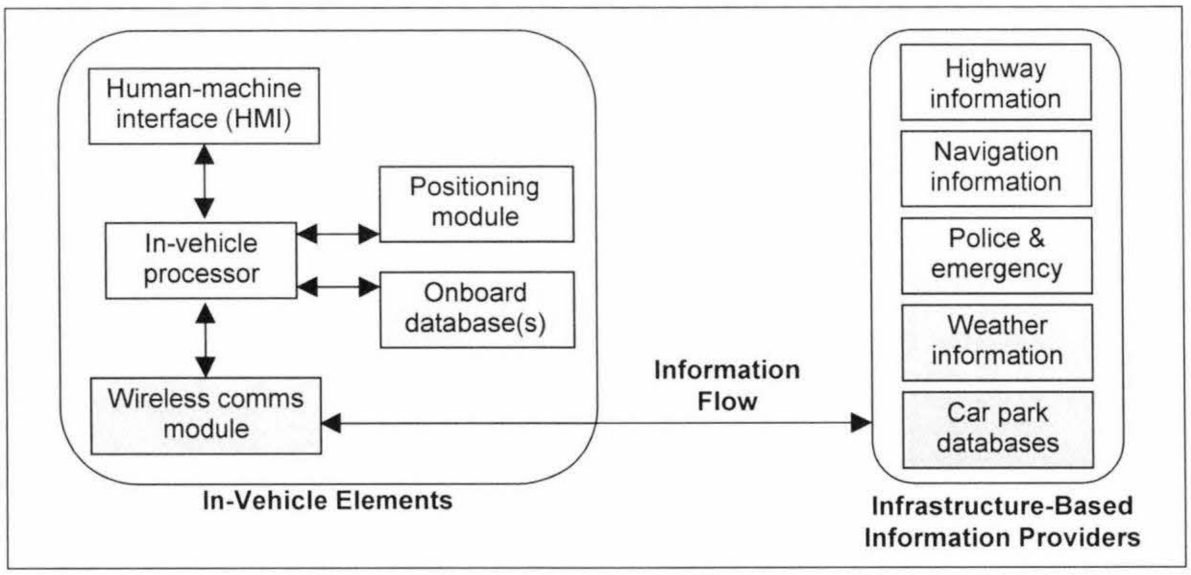


Figure 1.3: Components of an advanced traveller information system (ATIS) with onboard and offboard information sources.

A positioning module combines one or more sensor measurements to calculate the position of the vehicle relative to the Earth. An in-vehicle processor accepts the vehicle's current position as an input and performs various map-related functions, such as determining the messages that should be presented at the vehicle's given location. At the same time, real-time information may be received via an in-vehicle communication system and passed on to the processor. A *human-machine interface* (HMI) completes the system by presenting the data output by the processor in a human-readable form so that the driver can interpret it. In addition, the HMI provides one or more input devices such as a keyboard which allows the driver to communicate with the system; for example, to indicate their intention to select a new route to the nearest petrol station. The HMI is responsible for all interaction between the driver and the rest of the system.

1.6.2) Data Management

In this context, data management describes where the source of driver information is located and how the information will be communicated between the source and a vehicle. The two main data management philosophies of current ATIS systems are now described. Where appropriate, real-world systems are reviewed to exemplify current developments in ITS technology.

1) Onboard databases

The simplest category of ATIS encompasses *autonomous navigation aids*, which are essentially self-contained route guidance systems (OECD Scientific Experts Group, 1988). These *standalone* systems do not utilise any communication link with the environment external to a vehicle. Map data is stored in its entirety in each vehicle in an onboard, static (unchanging) format. Early projects such as DRIVEGUIDE and NAVICOM were based on this idea. Modern route guidance systems store all navigation information in a CD-ROM- or DVD-based database. For example, Alpine's DVD-based navigation system store uses map databases provided by NavTech to provide coverage of entire countries, such as Japan and the United States (Alpine, 2001). Drivers select the street that they seek, and the system directs the driver to that destination according to the driver's preference (for example, the quickest route, route that maximises travel on freeways, or route that minimises travel on toll roads). The system can also remember custom destinations specified by the driver.

While onboard map databases provide the luxury of almost instantaneous updates as a driver's travel plans change or side trips are added, they require a dedicated (and relatively costly) on-board processor (Arnholt, 2000). Data cannot be updated on-the-fly; rather, users must pay for periodic upgrades to the map database. In addition, such systems cannot disseminate real-time information.

2) Offboard databases

A more convenient and flexible data management option is to retrieve travel information from an offboard data source and relay it to vehicles via wireless communication as they need it. These mobile ATIS systems relay information between vehicles, sensors and land databases via powerful remote server computers. Offboard ATIS systems are steadily increasing in popularity due to their ability to provide up-to-date information to drivers (GPNN, 2000 [a]). Vehicle hardware is easier to update, more flexible in the range of options available to drivers and potentially less expensive to buy (GPNN, 2000 [b]). Numerous companies are currently exploring the offboard navigation model, including Motorola, Alpine and Visteon. In Japan, Denso already provides systems for Mercedes with dynamic route guidance via real-time traffic information input.

1.6.3) Wireless Communication

Wireless communication is an incredibly active research topic, encompassing radio, analog and digital cellular telephony, radio pagers, private land mobile radio, infrared and radio *wireless local area networks* (WLANs), microwave relays, and geostationary and low-Earth-orbit satellites (Elliott and Dailey, 1995). An excellent synopsis and comparison of wireless communication is presented by Elliott and Dailey (1995). Three major wireless communication methods dominate current ATIS systems: (1) beacons, (2) radio, and (3) cellular. An example of each method applied in the real world is described below.

1) Beacons

At the UK's first ever ITS national test project, detailed information on local traffic and weather conditions is transmitted to in-vehicle terminals provided by Alpine and programmed by Lucas Varity (M2 Presswire, 1999). Data is first sent from each vehicle via dedicated short wave communications (DSRC), to 80 roadside beacons. These high-speed DSRC beacons are positioned on gantries at key locations along the motorway. The information from beacons is then transmitted via a digital wireless data network, known as the RAM Network, to a central database managed by the national automobile association, RAC. The RAM Network provides two-way, real-time wireless data communication over dedicated radio frequencies. The RAM Network uses Mobitex, the international packet-switched standard for wireless data communications, originally developed by Ericsson. The central database interfaces with RAC's commercial travel and traffic information system (CATTiS). At RAC, travel and traffic data is retrieved and updated according to the central database. The results are transmitted back via the RAM Network to the beacons and via DSRC, to alert approaching vehicles to possible delays and adverse weather conditions ahead.

2) RDS-TMC

In Europe, the standard means of communicating real-time traffic information is via the Radio Data System Traffic Message Channel (RDS-TMC) (Kujawa, 1999). RDS-TMC is a protocol developed by the European Broadcast Union to employ subcarrier frequencies that are part of every publicly broadcast FM radio signal. Specific channels are reserved for traffic data. This data is normally gathered from roadside sensors strategically located around and within metropolitan areas. Data gathered by sensors is automatically tabulated into statistics that can be expressed by voice in simple terms for broadcast purposes. Traffic data can be retrieved within vehicles equipped with a properly configured audio system. Alternatively, the broadcast centre may elect to provide preprogrammed direction and caution information to drivers. The traffic sensor network is densest in Western Europe, particularly in Germany and France, and includes nearly every bridge in England. Nearly 65 per cent of audio systems sold with new cars in Europe have RDS-TMC circuitry installed. In the year 2000, Visteon and Clarion announced plans to launch systems that integrate real-time traffic data and dynamic rerouting information overlaid on in-vehicle navigation displays (GPNN, 2000 [c]). Both companies planned to use the RDS/TMC FM subcarrier networks and traffic codes.

3) Cellular

One limitation of RDS-TMC is that, being one-way, it does not support transmission of vehicle data back to the ATIS infrastructure. For example, this capability would be useful to provide more personalised information to a driver. Roadside beacons are also not well-suited to providing tailored information to specific vehicles. Cellular telephone networks address this niche. These are radio-based mobile communication systems that employ a set of base stations to transmit and receive signals (Drane and Rizos, 1998). The most commonly cited standard for cellular communication is the Global System for Mobile Communications (GSM). GSM is an open non-proprietary standard that supports a variety of digital services, most notably the digital cellular mobile telephone system in Europe, operating in the 900 MHz frequency band (Whelan, 1995). The market of personalised driver information was pioneered by GM's OnStar system, which offers pushbutton cellular connection to the OnStar service for emergencies, navigation and location services to personal vehicle users (Kujawa, 1999). It is now in its fourth year of service; in February 1999, over 40,000 subscribers had signed on in the US. Soon, OnStar and ATX Technologies will also provide "concierge" services to assist in the location of restaurants, gas stations, parking areas, hotels and motels

(Frenzel, 2001). The systems are equipped with GPS chipsets whose signals are transmitted to a service centre. In 1999, GM began shipping its OnStar 2 system that features three-button controls and a dedicated OnStar cellular network.

1.6.4) Internet-based ATIS

In the year 2000, manufacturers of ATIS systems embraced the Internet as an integrated means of providing real-time information to drivers. For example, InfoMove, a two-year-old company from Seattle, is looking at delivering customised Internet content and traffic data via handheld devices connected in cars (GPNN, 2000 [d]). A client-side browser provides the user interface, while a back-end server processes the data. Once the appropriate hardware is in place, InfoMove will enable automobiles of different manufacturers – using different computers or handheld devices, Web portals, and wireless carriers – to communicate with each other via the InfoMove database. UK companies CarCom Ltd. and Netcom Internet are combining to add Internet connectivity to CarCom's new in-car communication, navigation and computing platform. The CarCom system will use real-time traffic information provided by the UK's Metro service, which is sourced via the Internet. In March 2000, PSA Peugeot Citroaen and French communications giant, Vivendi, announced a joint venture to develop Wappi, a multi-access Internet portal (Electronic Times, 2000). Wappi aims to give motorists access to a wide range of up-to-date travel information via an in-vehicle screen and a GSM mobile telephone. Wappi includes assistance and emergency service links, user guides and maintenance tips, and will also have the ability to carry out a remote diagnosis on vehicle breakdowns. In the US, Utah's Department of Transportation provides traveller information and traffic updates to the public via email, which drivers receive automatically via their mobile phone or even their *personal digital assistant* (PDA) (Communications News, 2000).

The European Commission has launched a project known as the 'Multimedia Car Platform', which combines broadband Internet access with mobile telephony to present driver information on the dashboard (M2 Presswire, 2000). The project uses the latest mobile phone standards, such as GSM and third-generation cellular technology, 3G, to provide localised information (for example, describing traffic, parking and route guidance), regional information (such as news, weather and special events) and national information (such as news) to drivers and passengers. The system will support voice recognition control for the driver and hands/eyes control for passengers. This means that drivers can safely use the system on the move while passengers can enjoy a more direct interaction. The broadband information flow will use the IP protocol, which is a standard communications protocol used by the Internet. In the future the system could also be incorporated into trains, trucks, and buses, giving travellers all the facilities of the Internet and broadband broadcast-quality media while on the move. Another location-based service idea being considered is the opportunity to display advertising or special promotions to drivers as they drive within range of one or more businesses (Arnholt, 2000).

One of the problems with offboard databases is that collectively, offboard ATIS systems will require more bandwidth than is available in most current cellular networks. For this reason, it is predicted that true offboard navigation will not become mainstream for several years (GPNN, 2000 [a]). In the US, onboard-offboard hybrid systems are being developed which contain a smaller onboard database, reduced processing capability and onboard communications. In these systems, there is a degree of flexibility as to where information will be stored at any time. For example, most road layout and route guidance information may be stored remotely where there is sufficient storage capacity. Only when a vehicle enters a new region may it be necessary to download local road layout and route guidance information into onboard databases, to relieve the load on the communication system.

1.7) Review of Human-Machine Interfaces in Advanced Traveller Information Systems (ATIS)

1.7.1) Introduction

Zhao (1997, p143) defines the human-machine interface (HMI) provided for automotive navigation as a "module that provides the user (driver of a vehicle) with a means to interact with the location and navigation equipment computer and devices". The objectives of HMI design are to increase productivity and safety, maximise performance and ensure comfortable, ergonomic and effective human use. Inside a vehicle, the HMI permits a driver to indicate how they wish to proceed (for example, select a new route) and provides the driver with the stimulus they seek for navigation. In this context, the focus is on design and style of information presentation to the driver, specifically with respect to visual display-based interfaces, as it is this interface type that is capable of presenting virtual road signs. Ergonomics and non-visual interfaces, such as voice recognition, are not considered.

For any visual HMI to be effective, it must be timely, relevant to the current driver task, easily discerned without error and easily understood (Ashby and Parkes, 1993). Ideally it should match the expectations and experience of the user but not distract them so much that it jeopardises the safety of driver or vehicle. Thus it should not compete directly for the limited attention resources of the driver. In addition, the interface design should be consistent and minimise the memory load of the driver (Zhao, 1997).

An in-vehicle HMI concerns the two-way in-vehicle interaction between driver and equipment. An in-vehicle HMI consists of one or more display devices that are fully self-contained within a vehicle, such that the information displayed to each driver is personal and possibly unique. Examples include liquid crystal displays (LCDs), cathode-ray tube displays (CRTs), electroluminescent displays, plasma display panels and vacuum fluorescent displays (Zhao, 1997). The following elements are important in the design of an in-vehicle HMI (Catling, 1994):

1) The scope and type of information.
2) The range over which the system will display information.
3) The information content necessary to assist drivers, and information content sought by drivers.
4) The presentation style of information.
5) The timing of information provision linked with the motion of a vehicle.
6) The sequence in which information is displayed.

The nearest equivalent of virtual road signs using current technology is a hypothetical hybrid system that combines the dynamism and communication benefits of current in-vehicle HMIs with the presentation style of *variable-message signs*. Therefore, a review of both these types of HMI is now presented.

1.7.2) Variable-Message Signs

Variable-message signs (VMS) are physical signs positioned at strategic locations whose information content can be changed remotely. Variable direction signs indicate alternative routes as a function of the traffic situation on a network as a whole (Camus and Fortin, 1995). Alphanumeric signboards display short phrases relevant to current road or traffic conditions, allowing a wide variety of messages to be sent in an instantly-readable format, such as roadworks and road closures. The disadvantage is that foreign motorists may be unable to understand the messages displayed, as it is impractical to display the messages in more than one or two languages. A third category of VMS is pictograms which typically display schematic representations of the road network (such as in Japan) or traffic symbols. While the information conveyed is less explicit and may take longer to interpret than alphanumeric signs, this type of sign overcomes the language barrier (Camus and Fortin, 1995). The main advantage of VMSs is unquestionably their universal reception by all road users. This makes them more suitable for conveying safety-critical information. Once they are constructed at high financial cost, their effectiveness as a communication medium for collective dynamic control improves linearly as the number of road users increases. However, many drivers are tempted to think that such signs "only apply to others", particularly when dealing with dynamic route guidance information (Catling, 1994). Furthermore, the opportunity to convey information and the content of VMSs is very limited due to the short viewing time involved; ECMT (1995) suggests that most drivers cannot interpret more than seven words when travelling at speed. The high cost and large amount of space required by a VMS limits their potential implementation to specific stretches of road, such as busy highways or roads prone to seasonal weather. On highways, there may be no opportunity to turn around if the driver fails to observe the sign.

In Germany, variable-message signs aim to control traffic by showing ordinary traffic signs, such as signs that display admissible maximum speeds (Behrendt, 2000). The collection of traffic data is done conventionally through induction loops or radar sensors. Special sensor systems are able to determine the specific weather conditions (fog, heavy rain, black ice, etc). Unlike other European countries, Germany does not use alphanumerical verbal displays; instead, standard international traffic symbols are used. In the US, the Washington DOT has installed nine variable message signs that issue variable speed limits as well as information on road conditions, tyre chain requirements and highway closures (American City & County, 1998). The signs are part of Travel Aid, an intelligent transportation system designed to improve safety and minimise accidents. Wide aperture radar tracks current vehicle speeds, while six weather stations monitor temperature, humidity, precipitation, wind and road surface conditions. The information from all sources is gathered and transmitted by packet radio and microwave transmission to a control centre. Travel Aid then calculates safe speeds that are confirmed by DOT staff members and transmitted to the variable message signs.

1.7.3) In-Vehicle HMIs

In-vehicle HMIs offer several advantages over variable-message signs. Drivers can acquire their own personal information when necessary; for example, when he/she is lost and requires guidance back to the nearest highway. There is no chance of information transfer being impeded by other vehicles or bad weather conditions. Drivers can reconfirm information at their leisure, reducing the criticality of interpreting information at a specific time. The range of available information is much greater, but more importantly, drivers can obtain only that information they request. However, the flexibility of in-vehicle information systems always presents the possibility of information overload, which can be a particular concern for elderly drivers (Catling, 1994; Verwey, 1993). Schraagen (1993) makes the observation that dynamic graphical map displayed by the ETAK in-vehicle HMI did not reduce the number of navigation errors committed by a driver when compared with a conventional paper map. Also, most in-vehicle HMIs demand a fair amount of prior learning to be useful, which instantly reduces their potential market.

The need for drivers to divert their visual attention away from the external environment can be particularly problematic when they are learning how to use an in-vehicle HMI. The size of the display often aggravates this problem, forcing the driver to divert their focus back and forth between a small screen and the outside world. Popp and Farber (1991) investigated four different experimental display layouts of a simulated in-vehicle HMI, two of which displayed images very similar to roadside signs, and the other two displaying a sign and map simultaneously. All four displays forced drivers to observe them intensively for similar time durations and thus divert their attention away from traffic. In-vehicle instrument panel displays with limited contrast (such as LCDs) can delay information perception (Haller, 1991). Drivers' eyes are normally adapted to luminance levels of the outer forward view, which are typically much higher than the luminance of in-vehicle displays. Even after re-accommodation when the driver diverts their attention back to the in-vehicle display, their eyes may not be fully adapted to the reduced background light level of the display. Wierwille, Hulse, Fischer and Dingus (1991) found that drivers increase their proportion of time spent looking at the roadway centre (outside the vehicle) in heavy traffic conditions; similarly, they reduce the amount of time devoted to in-vehicle navigation displays in response to unanticipated scenarios (such as external vehicle accidents) when immediate control of the vehicle is more important. This suggests that an in-vehicle display is only effective for initial incident warning; once the driver knows of an impending incident, they prefer to rely more on what can be seen and heard outside the vehicle. Wierwille *et al.* (1991) state that during this time, navigation demands can in fact be postponed until the incident is brought under control.

Current in-vehicle visual HMIs present navigation and en-route trip information to drivers in a variety of styles. Wynalek favours the development of in-vehicle systems which use tiny displays that fit within the opening used by conventional car radios (Murray, 2001). Such displays accommodate only a few words of text. In a traffic-hazard warning system developed by Georgia Tech, known as the Safety Warning System, drivers are alerted to real-time hazards, dangerous weather and other traffic conditions by one of 64 pre-programmed text messages shown on an LED display (Wilson, 1999). These text-based displays can be read quickly and easily by drivers, but they are not optimal for displaying navigation information. For this information, arrows and symbols are preferred. A popular presentation style is known as *turn-by-turn*, in which arrow icons direct drivers at the correct time whenever a turn off the current track is required. Two navigation systems that use the turn-by-turn presentation style are shown in Figures 1.4.a and 1.4.b. Both systems indicate the direction of the turn to take, the name of the new street to turn onto, the distance to the turn-off and the distance to the driver's selected destination. In addition, NeverLost's display (Figure 1.4.b) presents an illustration of the road layout to further clarify each turn to a driver.

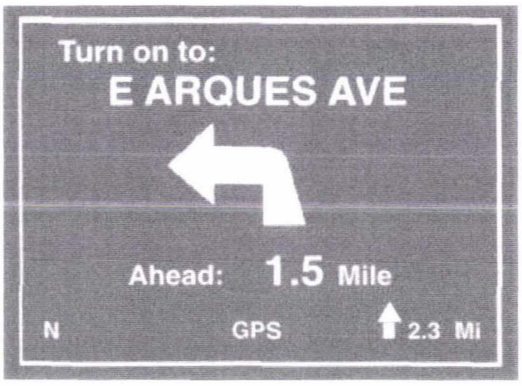


Figure 1.4.a: In-vehicle display of the TetraStar navigation system (Eby, 1999, p300)

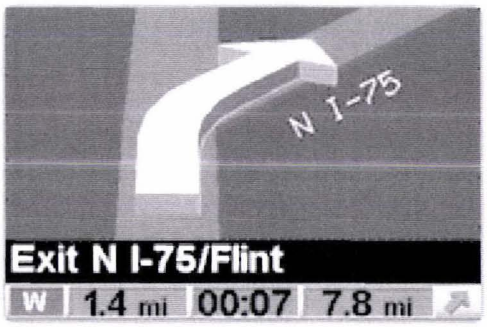


Figure 1.4.b: In-vehicle display of the NeverLost navigation system, recently fitted in Hertz rental cars (Bursa, 2000; image source: Hertz, 2001)

Other in-vehicle displays combine turn-by-turn instructions with an aerial planar view of the surrounding roads on a dynamic map. The advantage of these displays over turn-by-turn displays is that they resemble paper maps, allowing drivers to see exactly where they are, where they are going, and where nearby settlements, major roads and other facilities are located. Two examples of plan-view displays used by current navigation systems are shown in Figures 1.5.a and 1.5.b.

Recently, Nissan has announced details of its BirdView navigation system, which takes the viewpoint of an observer high above and slightly behind a vehicle (Nissan, 2001). The driver

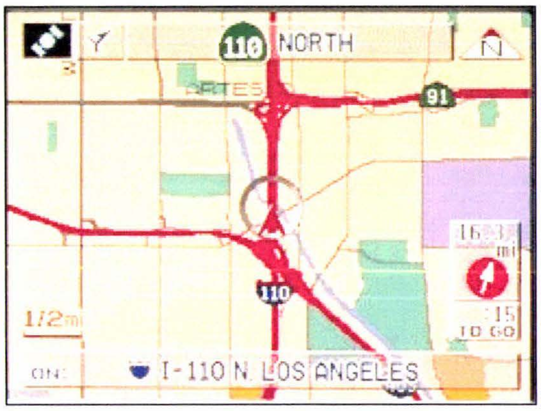


Figure 1.5.a: In-vehicle display of Alpine's DVD-based navigation system (Alpine, 2001)

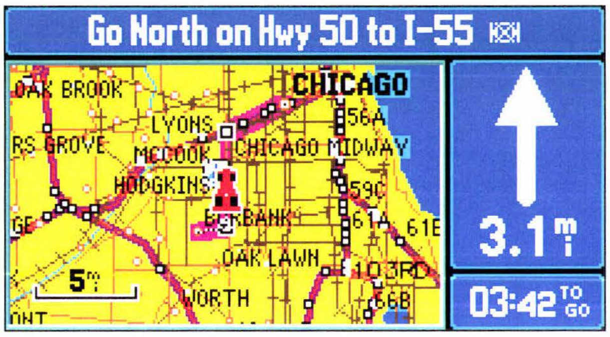


Figure 1.5.b: In-vehicle display of Garmin's StreetPilot ColorMap (Garmin Corporation, 2001)

can simultaneously see a detailed view of each intersection in the lower part of the screen for guidance on what turn to make, while the upper portion of the screen displays a zoomed-out view so the driver can anticipate the distance to the next intersection (Lewis, DeMeis, Mehri and Russelburg, 2000). The resulting upper display is a broad, panoramic three-dimensional map, shown on a HUD, which continues to the distant horizon ahead of the vehicle, as shown in Figure 1.6. The lower display (not shown) may closely remember a turn-by-turn display such as that shown in Figure 1.4.b. The idea is that by reducing the amount of zooming in and out required, the level of driver distraction is reduced. The system also displays real-time traffic information and warns of natural events such as floods or typhoons.

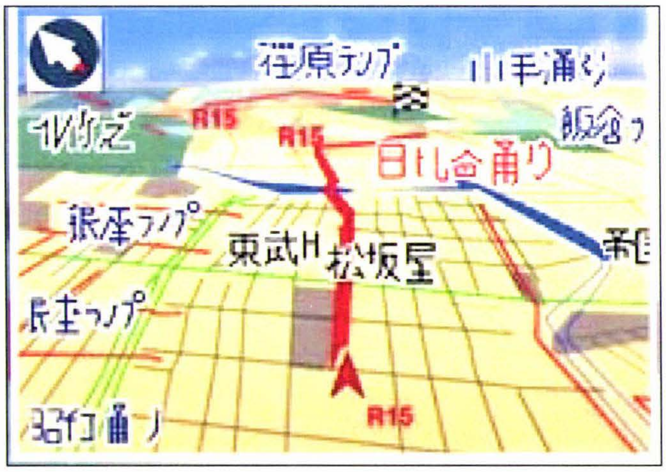


Figure 1.6: In-vehicle display of Nissan's BirdView navigation system (Nissan, 2001).

1.8) Suitability of a HUD as an In-Vehicle Display Device

1.8.1) Overview

As mentioned in (1.2), the aim of virtual road signs, as it is discussed in this thesis, is to mimic real road signs while overcoming many of the problems associated with conventional signs. By this definition, it is clear from (1.7) that virtual road signs do not match the style of information presentation used by current in-vehicle ATIS systems. Virtual road signs are opaque by their nature: their aim is to attract the driver's attention for a specific purpose at a specific time. Unlike turn-by-turn directions and aerial map views, virtual signs are only suitable for display on an AR display device, such as a HUD. One can argue that if AR allows drivers to interpret information in a more natural form, then virtual road signs are a more natural style of presentation than turn-by-turn directions and aerial map views.

The ways in which HUDs can be used to enhance the presentation of information to drivers have been re-addressed by vehicle manufacturers and researchers in recent years. Todoriki *et al.* (1994) proposed a route guidance system which overlaid a large guiding arrow onto the road ahead to direct drivers at intersections. They found that in a study of three participants, the average response time and number of glances were less for this "on-the-scene HUD" than for a conventional HUD that showed a plan view of the intersection. Steinfeld and Green (1998) tested a simulated HUD in which the outlines of roads at an intersection were superimposed to correspond exactly with the view of the intersection as observed by a driver. By labelling the superimposed outlines and displaying further route guidance information, the driver could determine which route to take. This idea is currently being tested as a means of aiding navigation in snowploughs (Trimble, 2000). When a snowplough driver cannot see the road ahead due to excessive snow or white-out conditions, a visual representation of the road is displayed on a HUD. As well as navigation applications, HUDs have also been used to enhance the night vision of drivers. The 2000 model of the Cadillac deVille used thermal imaging and a HUD to enhance the driver's view of the central part of the road scene. The system allowed drivers to spot pedestrians and animals on the road at night more easily (Pierce, 1999).

As mentioned in (1.3.3), of all the current display devices that may be suitable for displaying virtual road signs, an automotive HUD would appear to be the prime candidate at the present time. Thus, the perception of information presented on an automotive HUD is of particular interest. In order to gain a better insight into the suitability of this device for displaying virtual road signs, a literature review was undertaken of some of the factors affecting perception of information when displayed on an automotive HUD.

1.8.2) Field of View

The field of view of a display is the angle over which symbology may be displayed. A HUD with a narrow field of view can only display symbology in a narrow space, whereas a HUD with a wide field of view can spread its image over a wider area. Steinfeld and Green (1998) note that based on the results of one previous experiment (Williams and Green, 1992), the response times of drivers to navigation information were typically almost 10% less for a conventional small-angle HUD than for baseline in-vehicle instrument panels. Moreover, Steinfeld and Green (1998) found that response times decreased even more (at least 20%) over the instrument panel displays for full-windshield HUDs. The latter HUD configuration offered a wider field of view to drivers than the conventional HUD.

1.8.3) Angle of Depression

The angle of depression of a display device describes the angle at which the device appears below an observer's line of sight. Past experiments suggest that there is not necessarily an angle of depression for a HUD that optimises readability, but rather a number of trade-offs exist. In one experiment by Ward, Parkes and Crone (1995), the ability of thirty individuals to correctly identify the orientation of superimposed symbology was tested for two simulated HUDs. One HUD had a zero angle of depression, while the other HUD had a small non-zero (three degree) angle of depression. Participants were asked to track the location of a lead vehicle in video footage captured from the viewpoint of a real driver. At the same time, they were asked to identify the orientation of target icons displayed intermittently in different locations, for each HUD. Results showed that the legibility of the HUD with a three degree of angle of depression was superior. The reason suggested for this was that the latter HUD projected its image onto a less complex background scene (the road) that was more conducive to correct recognition than the other HUD. However, it was noted that the latter HUD increases the need for eye re-accommodation (Cole and Hughes, cited by Ward *et al.*, 1995). Furthermore, Sojourner and Antin (cited by Ward *et al.*, 1995) argue that a zero angle of depression permits faster response times.

Okabayashi, Sugie and Hatada (1999) investigated the effect of angle of depression on the ability of ten participants to recognise two simultaneous views. One simulated view represented objects observed in the forward view when driving; in this view, participants were asked to identify the orientation of simple shapes known as Snellen figures. The other simulated view was a superimposed HUD image of two green random numbers. Okabayashi *et al.* (1999) found that when participants correctly recognised the Snellen figures, their ability to also correctly recognise the HUD image decreased as the HUD angle of depression increased from zero to twenty degrees. However, when the task of Snellen figure recognition was removed, they found that correct recognition rates improved as the angle of depression increased from zero to five degrees. One suggested explanation for this finding was the significantly faster convergence response (eye re-accommodation) of the participants when the angle of depression was five degrees.

1.8.4) Distance to Image

Research suggests that the best recognition of HUD images is achieved when the projected image is collimated (focussed) at near-infinity. This configuration minimises the amount of eye re-accommodation needed as a driver diverts their attention between the HUD image and forward view of the road. In a simulated HUD experiment, Okabayashi and Sakata (1991) found that average correct recognition rate for a HUD image improved monotonically as the distance to the HUD image was increased from 0.7 metres to five metres. However, Okabayashi suggested in a paper written six years later (Okabayashi *et al.*, 1999) that very long display distances are not preferable. This finding was based on subjective evaluation data obtained from an experimental vehicle equipped with a HUD. Almost all drivers experienced a feeling of inconsistency between the HUD image and the forward view when the display distance was large (this distance was not specified). Okabayashi *et al.* (1999) also noted that the space restrictions of automobiles restrict the practical image distance to 1.5-2.5 metres.

1.8.5) Attention Tunnelling

Attention tunnelling describes an observer's fixation on a HUD image to the exclusion of other events in the real world, particularly unexpected events (Harrison, Ishii, Vicente and Buxton, 1995). This is perhaps the most commonly highlighted disadvantage of HUDs. In one aircraft-based experiment, Foyle, McCann, Sanford and Schwirzke (1993) tested the effect of information location on the ability of fourteen participants to concurrently process superimposed HUD symbology and "out-the-window" (real world) information. A computer simulated the out-the-window view by plotting shapes on the ground that participants were asked to track as closely as possible. At the same time, participants were asked to maintain a constant altitude by continually monitoring their current altitude displayed by a HUD. Simulated wind disturbances were presented during each trial. Foyle *et al.* (1993) found that when the altitude information was directly superimposed on the ground path, the ground-tracking performance of participants declined. The results showed no evidence that the trade-off in performance when the altitude was displayed in the lower position was due to visual masking (that is, reduced ability to see the ground shapes). This suggested that attention tunnelling was causing participants to focus too much attention on the altitude maintenance task. However, when the altitude information was superimposed in two other positions above the ground path, the performance of the altitude maintenance task did not change significantly, and the ground-track performance was no worse than when the altitude information was not displayed at all. Foyle *et al.* (1993) suggested that in the latter positions, the need for participants to visually scan their vision between known areas, rather than fixate their vision in one area, broke the effect of attention tunnelling. Gish *et al.* (1999) also found evidence of "HUD-induced cognitive capture" (p18), particularly in younger participants (25-54 years old). They state that it is possible that users would become less susceptible to cognitive capture as their experience with using the HUD increased. Harrison *et al.* (1995) also note that practice seems to improve the simultaneous monitoring performance of users.

1.8.6) Contrast interference

In a study of 36 participants by Gish *et al.* (1999), older observers (55 years and older) had particular problems seeing images displayed on a simulated HUD during simulated daytime trials. This suggested that the HUD contrast was somewhat interfered with by the brightness of the background. Gish *et al.* (1999) suggest that because the brightness in the simulation was less than what would be encountered while driving in the real world, their study underestimated the magnitude of the contrast interference problem.

1.8.7) Background complexity

The results of research into the effect of real-world background complexity on HUD perception are mostly predictable. The effect of background complexity was touched on in the discussion of HUD angle of depression in (1.8.3) above. In the same experiment by Ward *et al.* (1995) as that described in (1.8.3), the performance of participants for the target identification task was measured for varying background complexities. Recall that this task required identification of the orientation of target icons displayed intermittently in different locations on a simulated HUD. The background complexity was varied by changing the video footage of the actual driving scene observed. Ward *et al.* (1995) found that for the HUD placed at a zero degree depression angle, target identification performance significantly decreased as background complexity increased. Surprisingly, however, they found very little decrease in performance for the HUD placed at a three degree depression angle. The number of false positive identifications also increased as background complexity increased. In a separate task, participants were asked to identify specific changes in their current speed which was displayed on the HUD image. Again, the performance of participants for this task decreased significantly as background complexity increased. The number of false positive identifications was actually higher for a moderate complexity than a low or high complexity.

1.8.8) Implications for Virtual Road Signs

The number of factors that can affect HUD perception is greater than expected. The wider field-of-view of a full-windshield HUD would appear to make it the preferred option over a conventional HUD. A suitable angle of depression for display of virtual signs appears to be slightly lower than the driver's line of sight. This position is likely to make signs more readable due to the simpler background of the road scene, and may reduce the effect of attention tunnelling, even though it slightly increases the need for eye re-accommodation. The collimated distance to virtual sign images does not appear to be a critical factor. The effects of contrast interference and attention tunnelling are considered to be the most significant factors that will dictate whether it is practical to use a HUD, or indeed any AR device, to display virtual road signs to drivers.

It is worth pointing out that none of the experiments described in (1.8.2) to (1.8.7) used a real HUD, and most were conducted in simulated vehicle settings. Also, most of the experiments displayed simulated HUD images that were simplified for the purpose of experimentation, and which did not resemble the likely appearance of virtual road signs. The only sure way of working toward an answer as to whether virtual road signs may be feasible is to conduct our own experiments using real display devices in real vehicles. This was well beyond the budget and scope of this research. Therefore, development of a VRS system, as it is described in this thesis, was based on the assumption that virtual road signs may one day become a feasible option.

1.9) Summary

Conventional road signs are far from ideal as a means of communicating information to road users, and they are subject to a number of problems. Advanced traveller information systems are increasingly catering to the information needs of drivers, but there is still ample room for improvement to reduce the level of driver distraction and inattention they contribute to. Virtual road signs address these problems and aim to improve the safety of road users. They introduce a number of new opportunities for keeping road users informed. Augmented reality is the research area that makes virtual road signs possible. By developing a software prototype, the major design decisions involved in construction of a VRS system are examined and the technical feasibility of such a system can be assessed. A HUD is likely to be the most suitable display device for a VRS system. In the next chapter, we consider a high-level overview of the components of a VRS system and how such a system might operate.

CHAPTER 2
System Overview

This chapter outlines the basic requirements of a VRS system and the main functional components that comprise it. It then describes how the VRS prototype was developed for this project, followed by a high-level overview of how the prototype operates.

2.1) System Requirements

A VRS system renders graphical images of road signs on a display device incorporated inside a vehicle. Road signs must augment the driver's view of the environment in a way that minimises visual interference for the purpose of safe driving. Each road sign serves one of several purposes to enhance the driver's journey, such as to:

1) Indicate location of toilets, eateries and automotive service stations.
2) Identify landmarks such as buildings.
3) Warn of impending danger that could affect the driver's ability to control their vehicle.
4) Give prior notice of traffic conditions.

To cater for all categories of driver, vehicle, and road information, sign design must be entirely flexible, with provision for visual images as well as standard text. Timing of sign presentation, including the time when a sign first appears and the duration of visibility, is important to ensure a driver clearly understands the association between a sign and the feature it represents. Virtual signs must be easy to read to make them worthwhile, and they should become recognisable as early as possible (when they are still some distance from a vehicle). Correct synchronisation between vehicle position and displayed signs is essential to avoid giving a driver erroneous directions, or more significantly, compromising the driver's safety (for example, at intersections). For this project, technical feasibility means that the application must be attainable using current software and hardware. It must be developed to a standard suitable for public demonstration of how a VRS system might operate in practice, using real-world data.

In designing such a VRS system, the following characteristics are most important. The exact specifications of each characteristic will depend on each specific implementation and the subjectivity of the system designer, to some extent.

1) Accuracy and rapid access. Signs must appear in their correct location at the correct time.
2) Scalability. The system must support a wide range of operational areas, ranging from very small (for example, a university campus) to very large (an entire state or country).
3) Maintainability. The system must allow new signs to be added quickly and easily. Modification of existing data must also be straightforward.
4) Portability. All system hardware must be sufficiently small and robust to operate inside a vehicle.
5) Safety. Signs must not distract a driver or impair his/her vision beyond a degree considered necessary for safe driving. That is, the level of visual interference caused by virtual signs must be deemed acceptable. If, for some reason, the system malfunctions, it must degrade gracefully, perhaps by shutting off altogether.
6) Financial cost. The cost of the system hardware and software must be low enough to make the benefits of virtual signs worthwhile.

2.2) Functional Subsystems

Analysis of the system requirements identifies three subsystems that comprise a complete VRS system. Each subsystem is responsible for a cohesive, well-defined set of activities as described below:

1) The positioning and tracking subsystem is responsible for measuring the spatial coordinates and velocity (speed and heading) of a vehicle in the real-world, as it travels over land. The current vehicle position and velocity must be continuously measured to permit virtual signs to be displayed in synchronisation with the vehicle's motion.
2) The data subsystem takes the vehicle's current real-world position as an input and uses it as an index into a virtual "world". In a VRS context, a virtual world contains all road and sign information necessary to display virtual road signs. Although it is described as virtual, a virtual world is modelled on a set of roads in the real world. Each virtual sign is defined with respect to a road segment where that sign should be displayed. The output of this subsystem is a set of virtual signs that should be displayed at the present vehicle location and time. The data subsystem is divided into two parts:

    a) High-level data management, which encompasses design decisions such as where the road and sign information will be stored, who will be responsible for its upkeep, how the sign data will be collected, and how the data will be communicated between sensors and a vehicle.

b) Low-level data management, which is concerned with the form in which information is logically stored in each virtual world. A database is a convenient entity for defining the contents of a virtual world and subsequent retrieval of such data in a timely, efficient manner.

3) The display subsystem renders virtual road signs on an AR display device to make them visible to a driver.

2.3) Prototype Overview

2.3.1) Method of Implementation

Pimentel and Teixeira (1995) note that two common methods of implementing virtual worlds are:

1) By using a scripted language. A popular example is the Virtual Reality Modelling Language (VRML). VRML is a language used to represent three-dimensional objects and virtual worlds to be experienced via the World Wide Web (Huang and Lin, 1999). An excellent VRML tutorial is presented by Schneider and Martin-Michiellot (1998).
2) By using a compiled library or toolkit. A compiled library or toolkit typically provides more flexibility than a scripted language in creating a virtual environment. It is the tool of choice for software developers interested in developing complex, stand-alone virtual reality or AR applications (Pimentel and Teixeira, 1995). An example is the Open Inventor 3D Toolkit (Kim *et al.*, 1998).

The VRS prototype could have been implemented using either of these methods. For the purpose of research, a truly flexible approach was sought, and neither method was adopted. Instead, a VRS system was modelled and coded from scratch in a standard software development environment.

2.3.2) Software Engineering Modelling

Of the five high-level project types advocated by Pressman (1997, p160), the VRS prototype is a *concept development project* – its initiation is the product of a desire to explore an innovative application of technology. In the short timeframe available, a combination of the *prototyping* and *rapid application development (RAD)* software models was appropriate to develop the software prototype. The prototyping paradigm serves as a mechanism for identifying software requirements, and commonly follows an iterative process whereby a customer describes their requirements, the prototype is built or revised, the customer tests the prototype, and the cycle repeats (Pressman. 1997). In this context, the prototype serves to provide a real working example – an opportunity to design and implement one possible solution to show how a virtual road sign system might work. Time constraints and the lack of a specific customer limited the number of revisions that occurred in the prototyping cycle. The RAD paradigm is a linear sequential process model that emphasises an extremely short development cycle, made possible by using a component-based approach to construction and typically a *fourth-generation programming language* (4GL) (Pressman, 1997). In this context, the criteria of well understood requirements and constrained project scope were indeed true. The hybrid RAD-prototype paradigm permitted quick evaluation of the major design decisions and alternative designs for a VRS system. The result was the development of a fairly functional system within a very short time period.

An object-oriented paradigm was favoured over a traditional functionally-oriented paradigm. Due to the short project lifetime, maintenance of prototype code was not necessary, nor was the preparation of end-user documentation. Instead, efforts were made to simplify all coding for extensibility, should an intention to develop the application further (and possibly on a larger scale) arise in future. Formal estimation of application size and person-hours, perhaps using problem-based or process-based estimation (for example, the COCOMO model) (Pressman, 1997), was not undertaken at project initiation. An estimation of this information would undoubtably have been useful; in this instance, however, accurate estimation (perhaps for the purpose of remuneration) was not required, and it was more convenient to leave the scope slightly unconstrained and review it regularly, depending on progress.

2.3.3) Database Overview

The VRS prototype processes two main data stores, stored as two separate databases:

1) A *source world* database. This defines the complete set of virtual signs that will be displayed in one virtual world. In addition, the source world database defines the position and geometry of the areas in the world where virtual signs will be made visible to the driver of a vehicle. These areas are known henceforth as *sign regions*.
2) A *compiled world* database. This is derived from the source world database. One of the main purposes of the compiled world database is to represent a virtual world in a more efficient form than the source world database. The compiled database must be created prior to prototype execution. Then, by reading this optimised database at run-time, the VRS prototype executes more efficiently.

The topic of world representation and compilation shall be fully discussed in Chapters 4 & 5.

For the VRS prototype, the decision of which high-level data management approach to take was trivial. The test world that was used to test the prototype (as described in Chapter 7) was static and the volume was small. The intention was to store the source and compiled world databases on an onboard hard drive contained in an in-vehicle laptop PC. The author was responsible for collection, data entry and maintenance of all road and sign information stored by the databases. The fact that the databases would be completely self-contained inside a vehicle meant that an in-vehicle communication system would not be required. However, as shall be discussed in Chapters 6 and 7, the final VRS prototype used a decoupled approach whereby the databases were stored on a remote development PC.

2.3.4) Process and Data Flow Overview

Figure 2.1 (p. 2-6) illustrates the high-level process modules of the VRS prototype and the data flows between them. Standard *data flow diagram* conventions are adhered to: rectangles indicate data stores or data sources, rounded rectangles indicate algorithmic processes and solid arrows indicate data flows.

Vehicle tracking information is periodically received from an in-vehicle positioning device and displayed on-screen to register that the data has been received. Meanwhile, a high-resolution timer generates synchronised clock pulses at a fixed frequency. Each clock pulse is generated when it is time to update the set of virtual signs currently visible to a driver. For example, as the vehicle moves closer to a given virtual sign, that sign must be continually redrawn to give the realistic impression that the vehicle is approaching the sign. The screen refresh rate determines the fluidity and degree of motion of signs as perceived by the human visual system: a frame rate of five frames per second (fps) produces a rudimentary animation, yielding unconvincing motion and signs that appear to move abruptly. At the other extreme, a frame rate of 25 fps produces full-motion animation.

As each clock pulse is generated, the current vehicle data (position and velocity) must be known at that time in order to update the view observed by the driver. When a clock pulse coincides with data reported by the positioning device, the latter data is filtered to account for its continually-changing accuracy, and the filtered data describes the updated vehicle position and velocity. One problem is that vehicle data is reported at a much slower rate than the rate at which the screen is refreshed. To address this problem, a full set of predicted data is

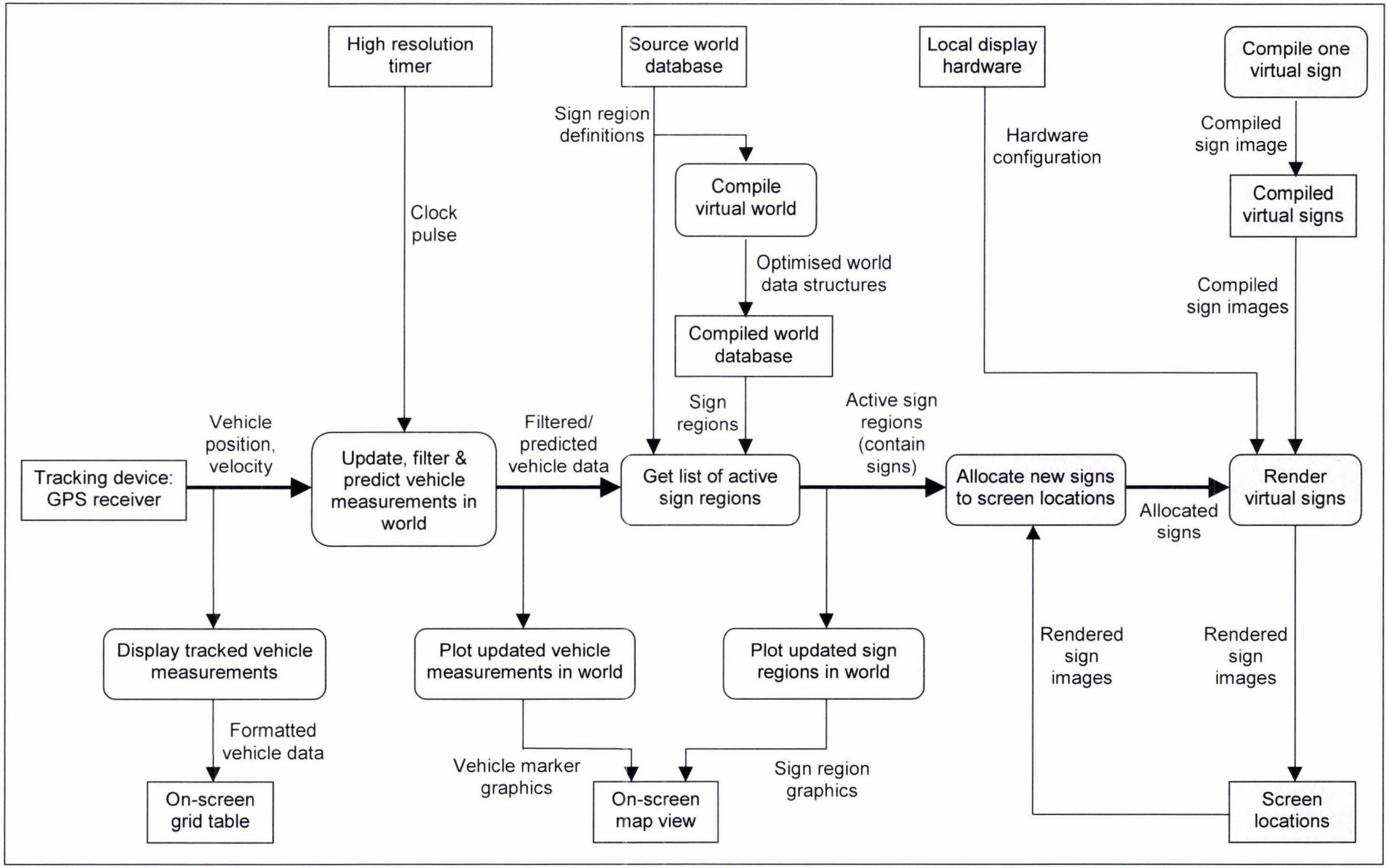


Figure 2.1: High-level Data Flow Diagram of VRS Prototype

calculated and stored every time new data is reported by the positioning device. The purpose of this predicted data is to anticipate the motion of the vehicle up until the time when the next set of vehicle data is received, based on past measurements. Then, when a clock pulse is generated during the vacant time interval when no data is reported by the positioning device, the next set of predicted data is retrieved from memory and returned as being the updated vehicle position and velocity.

At all times, a *map view* shows a plan view of the sign regions contained in one area of the virtual world (the specific area shown may be chosen by the user). In addition, the map view shows the current vehicle position and velocity. As each filtered/ predicted data set is generated, the updated vehicle position and velocity are plotted in the map view. At the same time, the filtered/ predicted data set is provided as an input to a process which looks up the list of sign regions associated with the updated vehicle position and velocity. The definition of association depends on how the world is represented and shall be discussed in Chapter 4. Once this list of *active* sign regions is known, the new signs associated with each sign region are allocated a screen location, ready to be displayed. At the same time, the map view is updated, highlighting the active sign regions to distinguish them from inactive sign regions.

Sign rendering on-screen completes the cycle. Like the compiled world database, virtual sign graphics are compiled, immediately before they are due to be displayed, to improve rendering performance. In this case, each compiled sign is represented as a specific type of data structure optimised for high-speed rendering. The final rendition of signs, in terms of their appearance and performance, is heavily dependent on the type of graphics hardware installed in the PC on which the VRS prototype is run.

2.3.5) The Choice of 3D APIs

Virtual road signs are three-dimensional objects rendered in a three-dimensional virtual world. An *application programming interface* (API) is a library of software functions and routines that provides specialised services to software developers. Two APIs that provide specialised services for the three-dimensional rendering of computer graphics are leading industry standards: OpenGL, developed by SGI in the early 1990s (Silicon Graphics Inc., 2001), and Direct3D, first released by Microsoft Corporation in 1995 (Microsoft Corporation, 1999). By using one of these APIs, the effort required to render signs is greatly reduced because a developer can make use of the functions provided by the API, rather than having to write the equivalent functions themself. A good technical comparison of the two APIs is given by unknown author C (2000). A more biased comparison in favour of OpenGL is given by Akin (1998).

Both OpenGL and Direct3D were examined as potential candidates for sign rendering in the VRS prototype. The author had no programming experience with either, and as such, access to detailed reference material was a major factor in deciding which to use. OpenGL was chosen due to its cross-platform and cross-language compatibility, excellent industry reputation, and most importantly, far wider range of tutorials and texts freely available (particularly on the World Wide Web: Silicon Graphics Inc., 2001; Warhol, 2001; Molofee, 2001; Unknown author B, 2001).

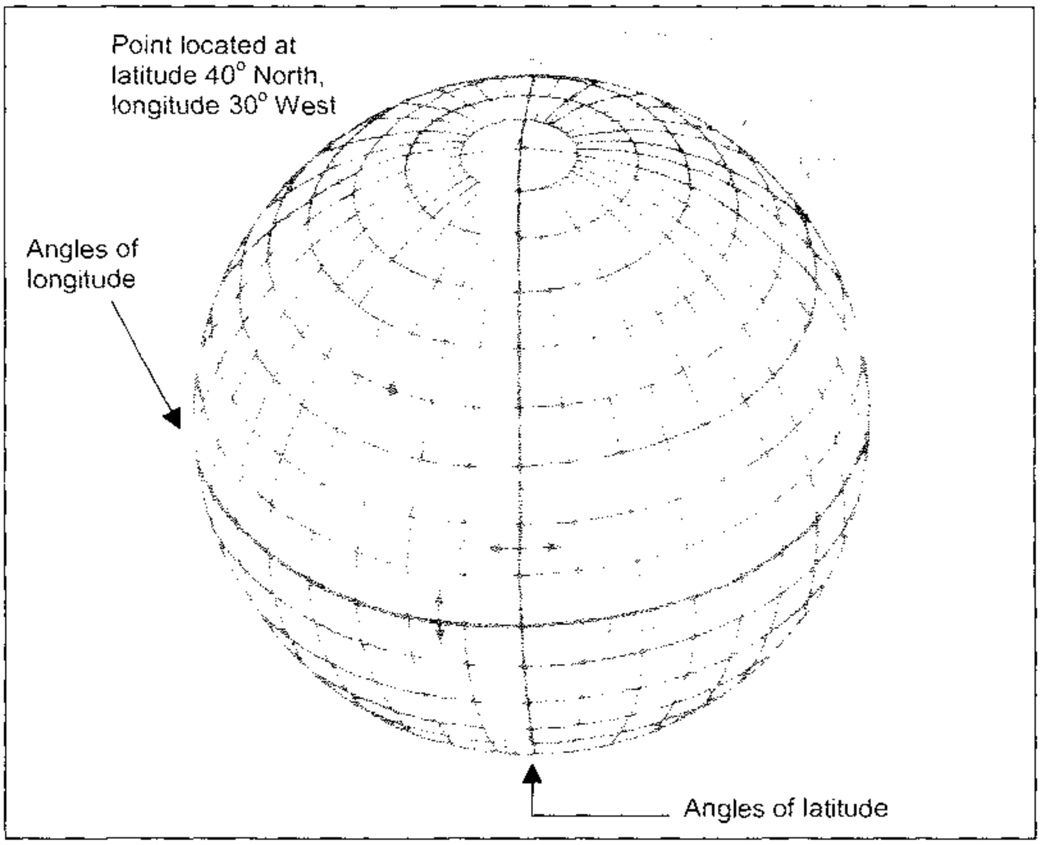


Figure 2.2: WGS-84 coordinate system (McHenry, 1992 [a], p184). All points on each parallel of latitude lie at the same latitude; all points on each meridian of longitude lie at the same longitude.

2.4) Coordinate Systems

2.4.1) Overview

The VRS prototype uses a *GPS receiver* as its positioning device, which is described in detail in Chapter 3. The frame of reference for GPS is known as the World Geodetic System 1984 (WGS-84). For this application, the position of a vehicle is measured in WGS-84 units of latitude and longitude. These describe the position of a point on the Earth's surface by two angles measured around the Earth's axes known as ellipsoid coordinates (Hofmann-Wellenhof, Lichtenegger and Collins, 1994). Figure 2.2 illustrates how the lines of latitude and longitude are arranged according to the WGS-84 coordinate system.

WGS-84 is a geocentric coordinate system: angles are measured around axes centred at the Earth's centre of mass (Drane and Rizos, 1998). Latitude has a range of 90^o North (at the North Pole) to 90^o South (at the South Pole), and the Earth's equator lies at a latitude of 0^o. Longitude has a range of 180^o West to 180^o East, where 0^o longitude is defined by a standard datum at Greenwich, London. This gives an origin of 0^o latitude, 0^o longitude lying a short distance south of Ghana and west of Gabon off the coast of Africa.

Typically, a virtual road sign world covers a very small region in comparison to the Earth's surface. In this thesis, planar coordinates are defined as (x, z). To make the coordinate system more manageable at a resolution of 0.001 minutes, vehicle position is expressed in a local planar frame of reference rather than a global ellipsoidal frame. There are many different projections that can be used to transform ellipsoid coordinates into planar coordinates, including conical projection (for example, the conformal Lambert projection), cylindrical projection (for example, the Transverse Mercator and Universal Transverse Mercator [UTM] projection systems), and azimuthal projection (Hofmann-Wellenhof *et al.*, 1994). Fortunately, when the size of a virtual world is small, a complex transformation from ellipsoid to planar coordinates is not required, and a much simpler approximation can suffice. One minute of latitude has a fixed angular length everywhere on the Earth's surface if we take the Earth to be a perfect sphere (a very good approximation). In contrast, one minute of longitude has a variable length that is a function of its latitude (Humerfelt, 2001). If the virtual world is small, then a small surface section of the perfect sphere can be well approximated by a finite plane. Therefore, the spherical coordinates can be roughly treated as planar coordinates. It is convenient for the horizontal and vertical units of the approximated planar coordinate system to be of the same length, however, so a transformation must still be performed.

The VRS prototype utilises three coordinate systems; the two main systems are summarised below. The third coordinate system is introduced in Chapter 7 for prototype testing. Full details of the three coordinate systems and transformations between them may be found in Appendix A.

1) GPS longitudinal and latitudinal coordinates (x, z), as described above. In general use, latitude and longitude are often measured in degrees, minutes and seconds, where an angle of one degree is equivalent to sixty minutes, and one minute is equivalent to sixty seconds. For simplicity, latitude and longitude are measured in units of minutes in this thesis.
2) Converted GPS coordinates (x, z), taken as approximated planar coordinates in minutes, in which a unit of longitude is taken to be of fixed length, defined as the length of one minute of longitude measured at the equator. This is approximately equal to the length of one unit of latitude.

To distinguish system (2) from system (1), system (2) coordinates are referred to as *equatorial latitude and longitude*, and the system itself is known as an *equatorial GPS coordinate system*. The system is localised by defining an offset (x_{offset}, z_{offset}) near the centre of the virtual world. All GPS coordinates reported by the GPS receiver in system (1) are immediately converted to system (2) coordinates and then redefined relative to the local offset. The final frame of reference has the point at [coordinates (x_{offset}, z_{offset}) in absolute system (2)] located at the origin. The purpose of utilising a local offset is to reduce the magnitude of coordinates in order to reduce the level of precision required to store them (in effect, to reduce the effect of floating-point round-off error at a given precision). This is effective because even medium-sized worlds will still be small relative to the entire surface of the Earth, perhaps spanning only a few degrees of latitude and longitude.

2.4.2) Coordinate System of Larger Worlds

The approximation of ellipsoid coordinates as planar coordinates over the area of a virtual world worsens as the range of latitude spanned by the world increases. As this range increases, the ability to approximate one minute of longitude as being of fixed length at all world latitudes decreases. For worlds beyond a certain size, the approximation will be unacceptably poor, and three options must be considered:

1) Break the world into a collection of smaller worlds, each small enough to yield a sufficiently good approximation but large enough to make the number of worlds manageable. Each smaller world has its own local offset (x_{offset}, z_{offset}) and therefore its own equatorial GPS coordinate system. As a vehicle travels, it will inevitably move between worlds. A general VRS system must ensure that the transition between coordinate systems is seamless as the vehicle moves between worlds.

2) Use an ellipsoid-to-planar projection, as listed in (2.4.1).

3) Represent sign regions in ellipsoidal rather than planar coordinates.

Of the three options, (1) is simplest followed by (2) and (3). Option (3) is perhaps the most elegant solution. For the VRS prototype described in this report, the small area of the test world (as shall be discussed in Chapter 7) did not warrant further investigation of these options.

2.5) Summary

A general VRS system is comprised of three basic subsystems, each responsible for a distinct area of functionality. One subsystem measures the position and velocity of a vehicle, one subsystem determines the virtual signs that should be displayed in a virtual world, and one subsystem displays the signs to the driver. Rapid application development of a software prototype allows one to assess the technical feasibility of a VRS system by presenting one possible implementation. The prototype employs two databases to store all world data, and uses a specific graphics API, OpenGL, to display signs on-screen. It utilises a GPS receiver to measure vehicle position and velocity. Because the test world of the prototype is small, the prototype coordinate systems are simplified.

The next four chapters 3-6 describe the three VRS subsystems in detail. The first section of chapters 3-5 describes and contrasts design decisions that may be applicable to a completely generic VRS system. In the second section of each chapter, specific implementation details as they relate to the VRS prototype are presented. The most important algorithmic processes and data structures are presented to illustrate how the internal mechanics of the VRS prototype combine to produce a working application.

CHAPTER 3
Positioning & Tracking Subsystem

The purpose of the positioning and tracking subsystem is to measure where a driver and his/her vehicle are currently located in the real world and where they are headed. The measured position coordinates are then passed to the data subsystem so that the correct set of virtual signs can be retrieved and displayed.

3.1) Positioning Characteristics & Systems

3.1.1) Introduction

Accurate and reliable knowledge of the position of a vehicle is an essential prerequisite for any ATIS system. Positioning involves determining the coordinates of the vehicle on the surface of the Earth (Zhao, 1997). Tracking is the ongoing positioning of a vehicle as it navigates a particular route. Five characteristics are important when choosing a positioning system:

1) Moderate accuracy, to ensure the correct signs are displayed at the correct time. The ideal accuracy of a VRS system is in the order of 1-3 metres to permit correct identification of individual lanes (or lane groups) of a highway. This would allow virtual signs to be correctly tailored depending on the lateral position of a vehicle on a highway. The minimum required accuracy is 10-15 metres to measure the location of buildings and intersections adequately.
2) High precision/ resolution, to ensure that small changes in vehicle position can be reported. The desired resolution of a VRS system is 1-3 metres to distinguish individual lanes of a highway and precisely resolve vehicle motion.
3) High sensitivity, to ensure that small changes in position are detected, such as a lane change.
4) Minimal temporal drift, to ensure that measurements are repeatable; that is, two position measurements of a stationary vehicle taken at distinct times must be very similar.
5) Availability & Continuity. Data must be reported at an adequate frequency, such that the time interval between measurements during which vehicle data is unknown is small. Ideally a truly continuous data source is desired, but may not be possible to achieve.

3.1.2) Navstar GPS

Satellite-based positioning systems use satellites orbiting above the Earth's atmosphere as active reference points whose coordinates can be accurately calculated. Of particular interest is the *NAVSTAR global positioning system* (GPS) operated by the US Department of Defense (Drane and Rizos, 1998). GPS is a satellite-based radio navigation system based on a network of 24 orbiting satellites in six orbital paths (Garmin Corporation, 2000 [a]). It provides a practical means of measuring position, velocity and altitude of a vehicle, as well as precise global time information. In the interests of vehicle location, two of the three main GPS functional areas are relevant: the space segment, concerned with satellites, and the user segment, concerned with the receivers that measure the data. Position measurement is based on the principle of time-of-arrival (TOA) ranging, in which the time interval taken for a signal transmitted from an emitter (satellite) is measured and multiplied by the signal velocity to obtain the emitter-to-receiver range (distance) (Zhao, 1997). The observation of at least four satellites simultaneously in the full constellation of 24 satellites permits multiple measurements to be triangulated, giving an accurate fix on the three-dimensional coordinates of the receiver. Mehaffey and Yeazel (2001) provide an up-to-date source of GPS information on the Web.

Here, GPS is considered to be the most suitable choice of positioning technology for a VRS system. Inexpensive consumer GPS receivers are readily available, portable and well proven. Unlike other tracking devices, a GPS receiver incurs a negligible operating cost and provides continuous coverage to every point on the globe, 24 hours a day, to both civilians and military users (Drane and Rizos, 1998). Considering its massive coverage area, the resolution of GPS is extraordinary, far exceeding any other consumer tracking device. Combined with its extremely high availability and reliability demanded for military purposes, GPS is easily the best option. Even the most inexpensive GPS receivers measure three-dimensional spatial coordinates, ground speed and heading, all of which must be known or estimated for correct virtual sign placement. A simple PC interface is standard on many receivers. In addition, GPS accuracy improved from approximately 100 metres (2RMS) to 15 metres (RMS) (Garmin Corporation, 2000 [b]) on May 1, 2000, as a result of the US Department of Defense disabling *selective availability* (SA), a technique intentionally employed to reduce accuracy of civilian GPS receivers.

One major advantage of GPS is that it scales perfectly as more users are added in increasingly larger virtual worlds. Satellites do all the computationally-intensive work and broadcast data to passive GPS receivers. The real-time performance of GPS receivers is constant regardless of the number of receivers in operation at any one time.

Let us review the capabilities of a consumer handheld GPS receiver. Measurements are reported at a fixed frequency of 1 Hz (once per second). The vehicle's position on the surface of the Earth is reported in nautical units of latitude and longitude. A common format describes position in terms of hours and minutes of latitude and longitude, at a resolution of 0.001 minutes. This resolution equates to approximately 1.85 metres when standing on the Earth's equator (Humerfelt, 2001), which lies within the precision range of 1-3 metres quoted in (3.1.1). However, the accuracy of positional coordinates is still limited to around 15 metres RMS for civilian use. The magnitude of this inaccuracy is constantly changing value due to movement of the GPS satellites (Hofmann-Wellenhof *et al.*, 1994). This means that the sign regions in which virtual signs exist must be intentionally made larger to guarantee, with a very high probability, that the reported GPS position will correctly lie within a given sign region. This is analogous to widening the road along which a vehicle is travelling in the real world. It must be noted that the error magnitude changes fairly slowly with respect to time; that is, it exhibits high temporal coherence (Kim and Oh, 2000).

Ground speed is reported in knots to an RMS accuracy of 0.1 knots (Garmin Corporation, 1999), where one knot is one nautical mile per hour (McHenry, 1992 [b]), 1.85 kilometres per hour, or 0.514 metres per second (3 dp). The high temporal coherence of positional errors is reflected in the fact that the speed error is considerably less than the positional error. This is explained by noting that speed is the unsigned derivative of position with respect to time. Providing the time difference Δt is small, positional measurements x_t and $x_{t+\Delta t}$ taken at time t and $(t+\Delta t)$ respectively will incur an error of very similar magnitude and direction. In subtracting $x_{t+\Delta t}$ from x_t and dividing by Δt, this common error is largely removed, leaving a very accurate speed value. In practice, a GPS receiver's speed calculation is likely to be more complex than this, perhaps based on multiple past measurements. Unfortunately, high speed accuracy is much less important than high positional accuracy when simulating road signs!

Ground heading is reported in degrees, where 0 degrees points due North toward the North Pole and 180 degrees points due South toward the South Pole. Altitude is reported in feet but may not be required for a VRS system. Estimated horizontal positional error (HPE), vertical error (VPE) and spherical error (EPE, which encapsulates HPE and VPE) are also reported, which are estimated magnitudes of the GPS error in the values of latitude and longitude, altitude, and overall three-dimensional position respectively. These values are estimates of positional error and not guarantees of maximum positional error. The techniques used to calculate HPE, VPE and EPE values vary between manufacturers of GPS units. For Garmin GPS receivers, the method of calculation is not disclosed; however, it is generally accepted that for each of the three values, there is an equal probability that the actual error is greater or less than the quoted value (Stone, 1997; Mehaffey, 2001). This is known as the 50% CEP value. As shall be described later, HPE turns out to be a very useful input for data filtering.

<u>**3.1.3) Dead Reckoning (DR)**</u>

Another positioning technique which can be used in conjunction with other positioning techniques (such as GPS) is dead reckoning. Providing that the starting location and all subsequent movements of a vehicle are known, it is possible to calculate the vehicle position at any instance. Dead reckoning is a primitive technique that achieves this by incrementally integrating distance travelled and direction of travel relative to a known reference point (Zhao, 1997). A commonly used device that measures distance is a differential odometer, which counts pulses generated by sensors embedded in the wheels of the vehicle. This device is particularly straightforward to implement in vehicles equipped with an *anti-lock brake system* (ABS) because wheel sensors already exist. Alternatively, a transmission pickup sensor, such as a variable-reluctance or Hall effect position sensor, can be attached to the vehicle speedometer or transmission housing (Zhao, 1997). A wide range of sensors are capable of measuring vehicle direction by noting its angle of rotation, including geomagnetic field sensor, magnetic compass, gas rate gyroscope, vibration gyroscope, fibre-optic gyroscope and accelerometers (Drane and Rizos, 1998; Catling, 1994; Ding, 1999). In addition, multiple sensors can be combined to yield a precision that would be difficult or impossible to obtain using a single sensor. Zhao (1997) gives a complete summary of linear and angular position sensors. The financial cost of a dead-reckoning system varies depending on the accuracy required; the fact that it offers simple, self-contained positioning and continuous position measurements are its major attractions. However, its main drawback is the accumulation of error that occurs as successive measurements are made. Although very high precision and sensitivity can be obtained, the effect is that its accuracy rapidly decreases due to drift. In practice, this limits the distance over which it can effectively be used to hundreds of metres before recalibration is needed (Zhao, 1997). In addition, the accuracy of many DR sensors is dependent on environment factors such as humidity and temperature (Ding, 1999).

3.2) Measuring Real-World GPS Accuracy

3.2.1) Overview

Comparisons of measured positional error magnitude before and after SA disablement are shown in Figures 3.1.a and 3.1.b respectively. It can be seen in Figure 3.1.a that the reduction in measured error magnitude due to SA disablement was more pronounced for vertical errors in altitude than horizontal errors. Figure 3.1.b shows that measured RMS magnitude of horizontal error before and after SA disablement was significantly less than the values quoted in (3.1.2).

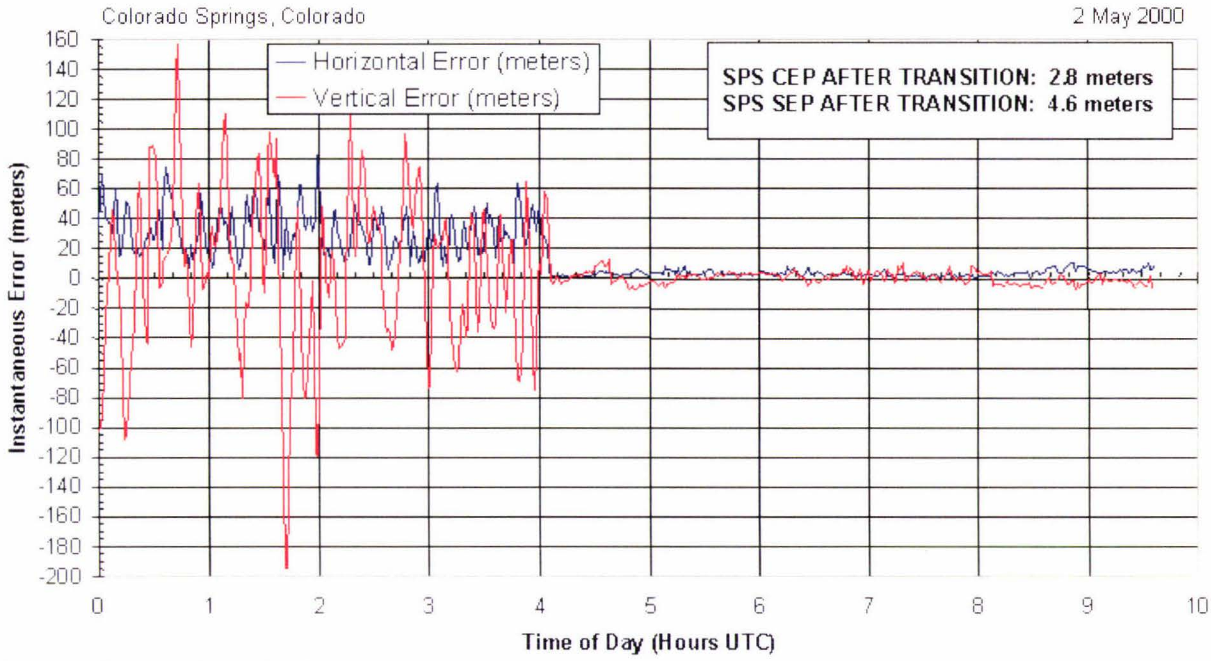
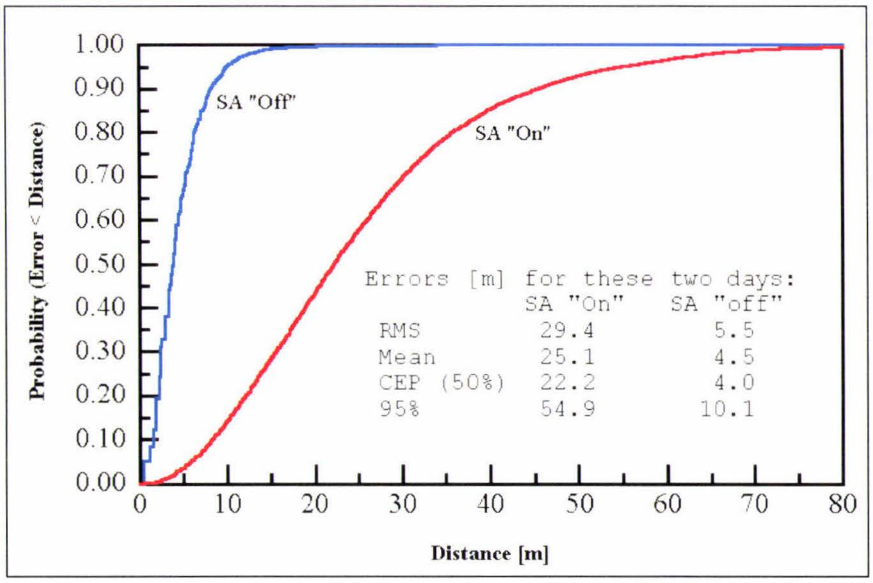


Figure 3.1.a: Graph of measured instantaneous GPS error over a ten-hour period taken at Colorado Springs on May 2, 2000 (Unknown author A, 2001). SA was disabled just after 0400 hours UTC.



```
Errors [m] for these two days:
             SA "On"   SA "off"
 RMS          29.4       5.5
 Mean         25.1       4.5
 CEP (50%)    22.2       4.0
 95%          54.9      10.1
```

Figure 3.1.b: Statistics of estimated GPS horizontal positional error with and without SA (Wilson, 2001 [a]). Measurements were recorded every two seconds over a 24 hour period using a Garmin 12XL GPS receiver.

Even with SA disabled, there was still some question as to whether the improved accuracy of GPS would indeed be sufficient for the precise vehicle tracking required for a VRS system. Figure 3.1.b suggests that the RMS error magnitude quoted in the literature in the absence of SA (15 metres) is conservative. Early data collection indeed showed that the positional error was highly correlated on a short timescale (seconds), but it was unknown exactly how this correlation varied over longer timescales (minutes and hours). To investigate this error behaviour in practice, a simple data logging experiment was conducted by securing a handheld GPS receiver to the roof of a house for 24 hours. The test conditions were ideal, in that the GPS receiver had a very wide, unobscured view of the sky. A laptop PC recorded the three spatial coordinates of the stationary receiver at regular intervals. In addition, HPE values estimated by the receiver were recorded so that they could later be compared with those values derived from the measured coordinates. In order to assess the degree of variation at different scales, the data was filtered using the *discrete wavelet transform* (DWT). One advantage of using an integral wavelet transform for signal decomposition rather than other integral transforms, such as the *Fast Fourier Transform* (FFT), is that the basis functions are localised with respect to both time and scale (Chui, 1992; Polikar, 2001). Therefore, the DWT allows the rate of change of GPS measurements to be easily analysed at different time scales of seconds, minutes and hours. The WaveLab library of wavelet functions (Donoho, Johnstone, Buckheit, Chen and Scargle, 1996) was utilised to perform this analysis. Code is listed on the CD-R disc contained in the back pocket of this thesis.

3.2.2) Results of GPS Data Logging Experiment

Log period	86401 seconds
# samples collected during log period	43148
Unfiltered latitude data: Standard deviation Range	 0.00194 minutes 0.019 minutes
Unfiltered longitude data: Standard deviation Range	 0.00227 minutes 0.036 minutes
Estimated horizontal error (HPE):	
Mean amplitude RMS amplitude Range	5.9 metres 6.1 metres 3.6 – 18.5 metres
DWT analysis: # samples analysed Wavelet utilised for analysis Number of DWT decomposition levels	 32768 Symmlet-4 16

Box 3.1: Results of GPS data logging experiment

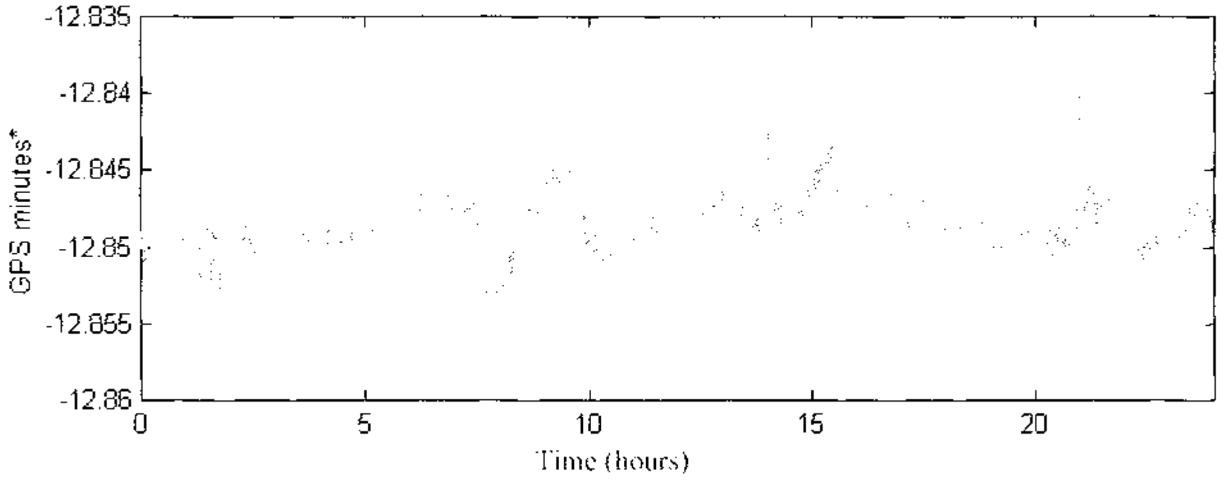


Figure 3.2.a: Graph of logged (unfiltered) GPS latitude data versus time.
*** Vertical axis values are relative to latitude 40 degrees South.**

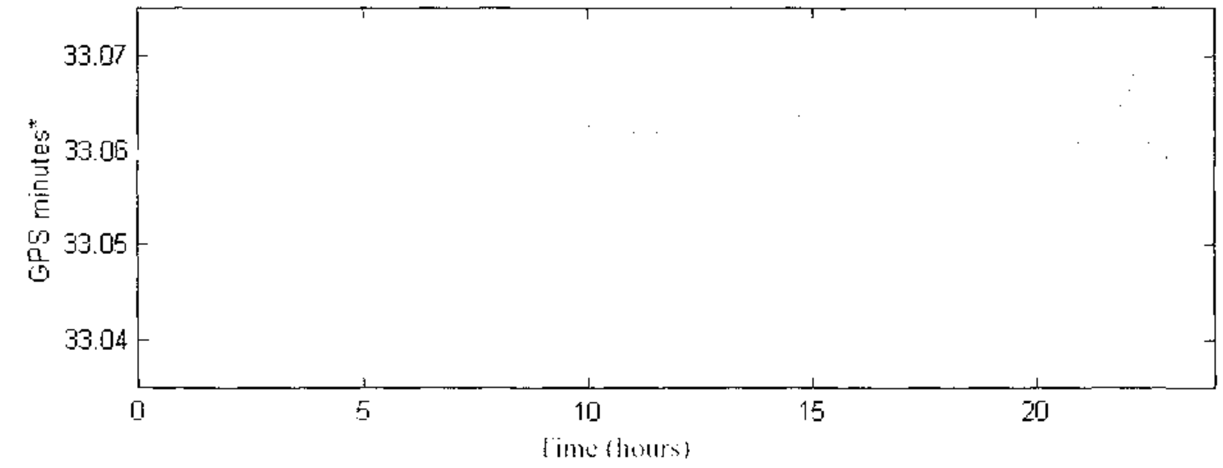


Figure 3.2.b: Graph of logged (unfiltered) GPS longitude data versus time.
*** Vertical axis values are relative to longitude 175 degrees East.**

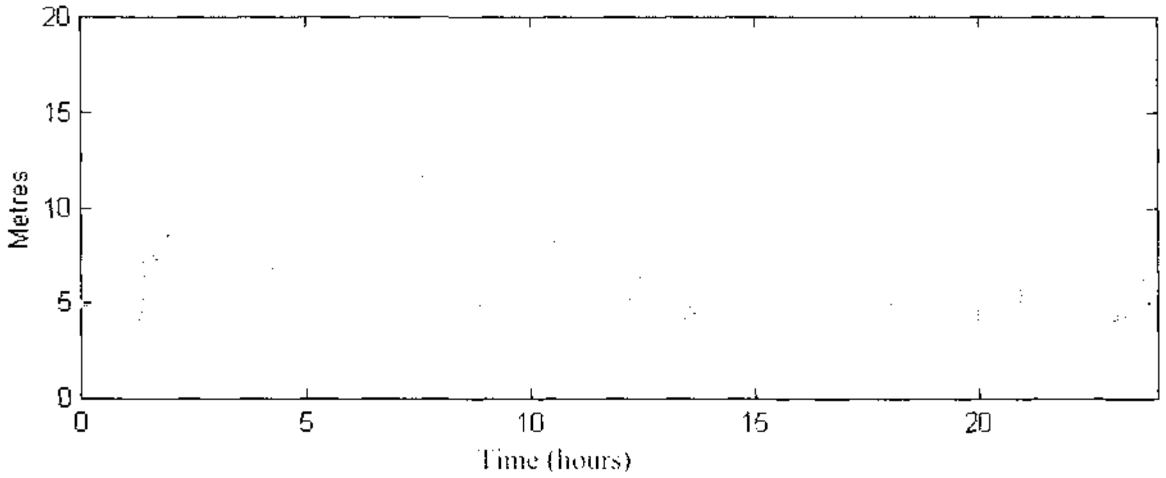


Figure 3.2.c: Graph of estimated CEP HPE versus time

Note from Box 3.1 that 32768 samples were chosen for analysis rather than the full 43148 samples. This was necessary because an integer power of two number of samples is required by the DWT (Polikar, 2001) (here $n = 2^{15}$). Logged latitude and longitude data was filtered at fifteen DWT decomposition levels (the DC coefficient at level zero was ignored); results are shown in Figures 3.3.a-b. Level one corresponded to the longest time scale and level fifteen corresponded to the shortest time scale.

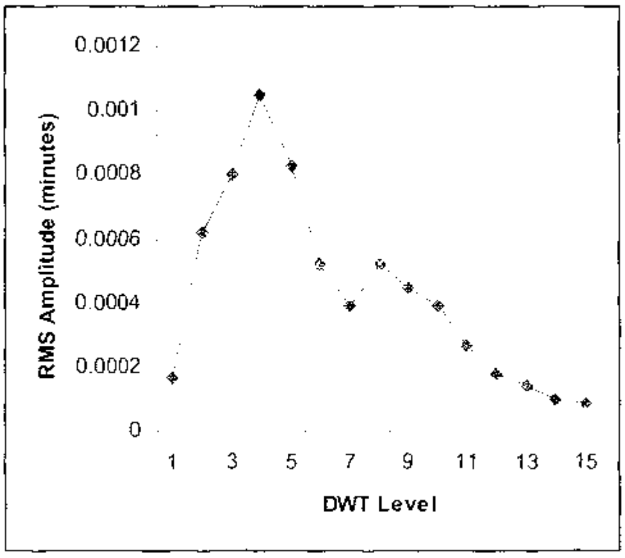


Figure 3.3.a: RMS amplitude of GPS latitude data, filtered at various time scales

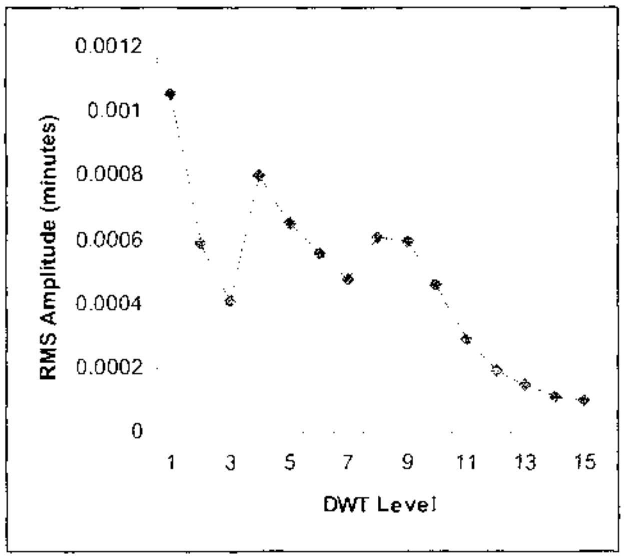


Figure 3.3.b: RMS amplitude of GPS longitude data, filtered at various time scales

<u>3.2.3) Discussion of Results</u>

The graphs of logged latitude and longitude show small amplitude variations of 0.00194 minutes and 0.00227 minutes (one standard deviation) over the full logging period. These values represent distances of only 3.6 and 3.2 metres respectively, or a horizontal positional error of $\sqrt{(3.6)^2 + (3.2)^2} = 4.8$ metres RMS. This gives close agreement with the value of 5.5 metres quoted in Figure 3.1.b for an independent experiment. However, the range of each value is significantly greater at 35 and 51 metres respectively. The mean and RMS estimated horizontal position error (HPE) of 5.9 metres and 6.1 metres respectively cannot be directly compared with the value of 4.8 metres calculated above, because the former are CEP values and the latter is an RMS value. To convert a CEP HPE value to an RMS HPE value, the CEP HPE value is multiplied by 1.20 (Wilson, 2001 [b]). When this calculation is performed for the HPE statistics in Box 3.1, mean HPE becomes 7.1 metres, RMS HPE becomes 7.4 metres and the HPE range becomes 4.3-22.2 metres. The two former values are approximately 50% greater than the measured value quoted above (4.8 metres) and approximately 30% greater than the value of 5.5 metres quoted in Figure 3.1.b of (3.2.1). The absolute difference is less than three metres, which is an insignificant difference. However, the HPE range is fairly large. The graph of logged CEP HPE shows that:

1) Almost all values lie between 4-10 metres (corresponding to 4.8-12.0 metres RMS), and
2) Local increases in reported HPE are reasonably correlated with the degree of local deviation of measured latitude and longitude. This is most clearly evident for the spike near time 14 hours, and less so for samples at times near 2, 7.5, 10 and 22.5 hours.

For filtered latitude, RMS amplitude increases from DWT level one to four, peaks and then decreases for higher levels, with a smaller peak occurring at level eight (see Figure 3.3.a). Similar behaviour occurs for filtered longitude, with the exception of a maximum RMS amplitude occurring at level one and reduced signal variation at levels two and three (see Figure 3.3.b). Translation of these results into dominant frequencies is only possible in a loose sense due to the irregular shape of the analysing wavelet. Examination of graphs of the filtered data (not shown) shows that roughly six complete cycles occur over the entire data set at DWT level four. The analysed data set represents a period of 18.2 hours, so peaks at levels four and eight roughly correspond to high signal variation for periods of 3.0 hours and 11 minutes respectively. The limited precision of logged latitude and longitude values prevents any further conclusions to be reached.

These experimental results suggest that in ideal conditions, the RMS positional error for a stationary GPS receiver is significantly better than the value of 15 metres quoted in the literature (Garmin Corporation, 1999). This agrees with the results shown in Figure 3.1.b of (3.2.1), although the researcher in that experiment did not elaborate on the test conditions. This finding assumes of course that the logged data is a good representative sample of data that can be expected in real-world use. It is also subject to the assumption that the mean values of latitude and longitude calculated over the entire sampling period represent the exact true position of the GPS receiver. The fact that very inaccurate values (bad position data) are very occasionally reported (as shown by Figures 3.2.a-b) explains the high observed ranges of latitude and longitude. There is no reason to believe that the HPE values estimated by the receiver are not good indicators of the actual positional error. However, in the absence of more accurate measuring equipment, it was not possible to definitively assess the accuracy of the measured positional coordinates or robustness of the reported HPE values.

Based on the results of this experiment, the RMS GPS positional error appears to be acceptable for a VRS system. An RMS positional error magnitude of 5-7 metres corresponds to an expected magnitude of 8.7-12.1 metres in 95% of all measurements (Wilson, 2001 [b] gives the conversion factors). This range borders onto the minimum required accuracy of 10-15 metres quoted in (3.1.1). In less ideal real-world conditions, we should expect a slightly greater error magnitude than the range of 8.7-12.1 metres quoted above. In rare instances (<5% of all measurements), the absolute positional error is expected to be unacceptably large; however, for the purposes of prototype testing, the likelihood of receiving very bad data is very low. The DWT results confirm that the positional error is slowly time-varying in the sense that variation on a per-second basis is negligible. This means that the relative accuracy over a short time and distance is very high, which has important implications for the VRS prototype. A given virtual sign is typically only visible to a driver for a matter of seconds: providing that the sign is correctly displayed for the first time, high relative accuracy should minimise the chance of the sign disappearing during that time when it should be visible.

It should be noted that these results will not automatically be applicable when tracking a moving vehicle. To improve accuracy of the final measurements, *map-aided positioning* techniques can be used in conjunction with a GPS receiver and indeed any tracking device. This is further discussed in Chapter 8.

Prototype Positioning & Tracking Subsystem

3.3) Overview

A Garmin GPS-12 handheld receiver provides all vehicle tracking measurements at prototype run-time. The receiver is connected to a laptop PC via the serial port. Rather than use Garmin's proprietary transfer protocol, the NMEA protocol is utilised to transfer data between PC and receiver unit at a fixed bitrate. The NMEA protocol is standardised by the National Marine Electronics Association and is an industry standard for low-level data transfer between navigation devices such as GPS units, radar, and depthfinders (Bennett, 2000). Appendix B describes the protocol in more detail. Dead reckoning sensors were not fitted to the test vehicle, as no such sensors were available.

3.4) NMEA Parsing & Decoding

The GPS receiver transmits its measured data to a PC using the NMEA-0183 v2.0 protocol (Garmin Corporation, 1999). NMEA allows a single 'talker' and several 'listeners' on one circuit. Data is transmitted in bursts when the GPS data becomes available in the form of a set of 'sentences'. Each sentence starts with a dollar sign, a two-letter *talker ID*, a three-letter *sentence ID*, followed by a number of data fields separated by commas, and terminated by an optional checksum and carriage return/line feed. A sentence may contain up to 82 characters. There are dozens of standard sentence IDs defined, as listed by Bennett (2000). In addition, NMEA allows manufacturers to define their own proprietary sentence formats.

Most NMEA sentences reported by a GPS receiver are ignored by the VRS prototype – only two sentences of interest are decoded. Because it is not possible to notify the GPS unit which subset of sentences are sought from the whole set, the entire burst must be parsed every time. Table 3.2 describes the syntax of these sentences for a real NMEA data burst received from the GPS-12 receiver inside a stationary vehicle. The complete NMEA data burst and explanation of all sentences contained within it are given in Appendix B.

NMEA data is received by the PC via a serial port. In the VRS prototype, low-level serial port communication is the responsibility of a Microsoft Communication ActiveX control which handles all serial port initialisation and buffering. Peacock (2000) gives an excellent explanation of low-level serial communications. By default, NMEA uses no handshaking, a bitrate of 4800 bits per second (bps), no parity, eight data bits per symbol and one stop bit. An input buffer size of 1024 bytes is sufficient to store almost two full NMEA bursts. Each burst is received into the buffer in 32-byte chunks; parsing of each burst proceeds when the first ID tag after the last useful sentence is received (`$GPGLL`).

`$GPRMC,011946,A,4012.853,S,17533.060,E,000.0,360.0,220600,021.4,E*68`
Standard sentence: `$GPRMC` = Recommended minimum specific GPS/Transit data

Data	Decoded	Data	Decoded
`011946`	Time of fix (1:19:46 PM)	`000.0`	Speed over ground (knots)
`A`	Data okay (V would indicate GPS data is not genuine),	`360.0`	True course made good (heading, in degrees)
`4012.853,S`	Latitude (40 degrees 12.853 minutes South)	`220600`	Date of fix (22 June 2000)
`17533.060,E`	Longitude (175 degrees 33.060 minutes East)	`021.4,E`	Magnetic variation (degrees)
		`*68`	Checksum

`$PGRME,16.6,M,38.9,M,42.4,M*1F`
Proprietary Garmin sentence: `$PGRME` = Estimated error values (see [3.1.2])

Data	Decoded	Data	Decoded
`16.6,M`	Estimated HPE (metres)	`42.4,M`	Overall EPE (metres)
`38.9,M`	Estimated VPE (metres)	`*1F`	Checksum

Table 3.2: Explanation of two sentences for a real NMEA data burst

3.5) Kalman Filtering

3.5.1) Introduction

At a bitrate of 4800 bps, GPS data is transferred in an NMEA burst sent by the GPS-12 receiver once every two seconds (0.5 Hz). This is the same configuration used by Kim and Oh (2000). For some unknown reason, this is half the rate observed by a user reading the same data from the GPS receiver's LCD user interface. One possible explanation is that the sheer volume of NMEA data cannot be transferred between receiver and PC fast enough at the fixed bitrate to sustain 1 Hz. At a screen refresh rate of 0.5 Hz, virtual signs would appear to be stationary most of the time and move abruptly every two seconds. The resulting animation would be completely unsatisfactory.

To improve the animation, a six-state Kalman filter was added to predict the motion of the vehicle during each vacant time interval when no data was sent by the GPS receiver. A number of different filters were available to perform general filtering, many of which are derivations from the classic weighted least-squares filter (Brookner, 1998). A Kalman filter was the logical choice as it is proven to generate optimum estimates of a random variable that satisfies a linear differential equation driven by white Gaussian noise (Brookner, 1998). Some of the commonly cited benefits of the Kalman filter are listed in Table 3.3:

Benefit	Elaboration
Generality	The filter is applicable and proven in a wide range of problems including weapons guidance systems and ballistic modelling.
Continuity	The filter provides a running measure of accuracy of predicted data e.g. predicted sign position.
Optimality	The filter permits optimum handling of accuracy measurements that vary with time, including missed measurements (rarely occur with GPS) and non-equal times between measurements (not applicable to GPS). The filter can combine multiple measurements to provide an estimation that is bounded with less variance than individual measurements alone (Ding, 1999).
Stability	Addition of a random-velocity variable to the filter forces it to be stable.
Versatility	The matrix representation of the filter is completely general and scalable. Its form allows specific filters to be quickly implemented to tackle specific problems.
Efficiency	The matrix representation of the filter permits fast calculations on modern computers (at the time of the filter's inception in 1960, its relatively large processing overhead was cited as one disadvantage).

Table 3.3: Benefits of Kalman filter (based on Brookner, 1998, p67)

The iterative process of Kalman filtering is summarised in Figure 3.4.

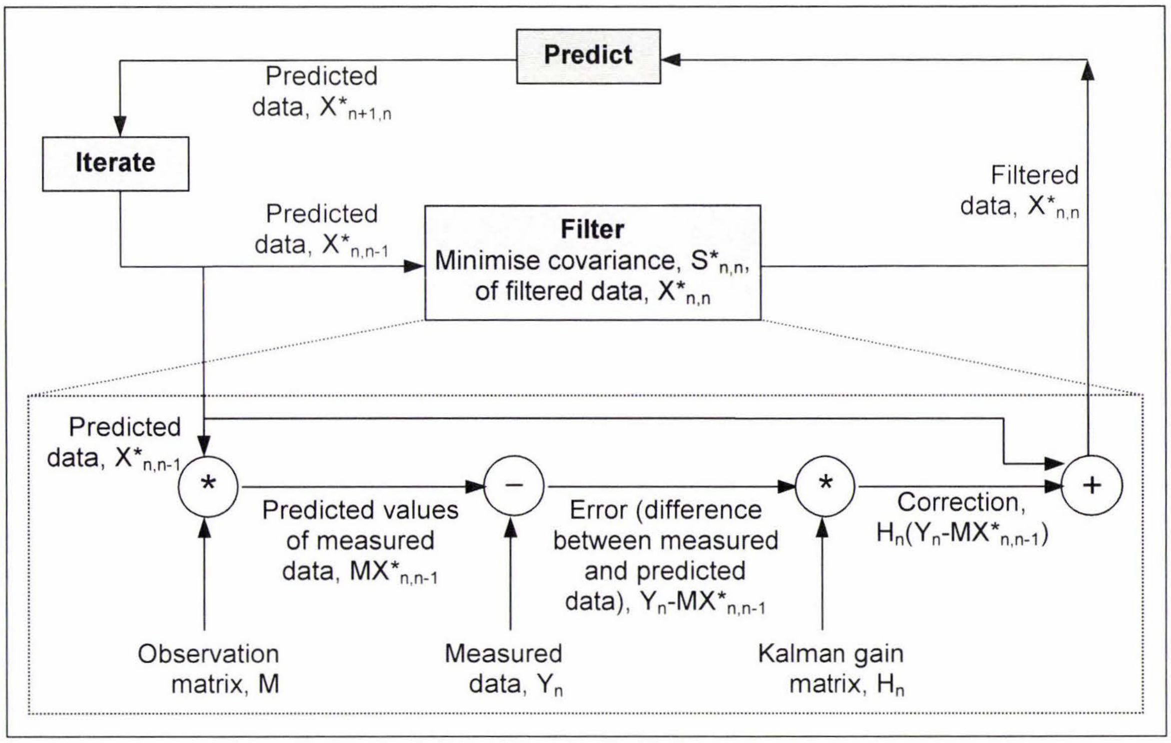

Predict
Predicted data, X*n+1,n
Iterate
Predicted data, X*n,n-1
Filter
Minimise covariance, S*n,n, of filtered data, X*n,n
Filtered data, X*n,n
Predicted data, X*n,n-1
Predicted values of measured data, MX*n,n-1
Error (difference between measured and predicted data), Yn-MX*n,n-1
Correction, Hn(Yn-MX*n,n-1)
Observation matrix, M
Measured data, Yn
Kalman gain matrix, Hn

Figure 3.4: Iterative Kalman filtering process (calculation of $S^*_{n,n}$ and $S^*_{n,n-1}$ is not shown). Notation: $X^*_{p,q}$: Estimated state value at iteration p, based on data at iteration q and before.

For a given iteration n, a Kalman filter estimates an optimum set of filtered parameters, stored in matrix $X^*_{n,n}$, based on a vector of measurements, Y_n, and a predicted estimate of filtered parameters made at iteration (n-1), stored in state vector $X^*_{n,n-1}$. The Kalman gain matrix, H_n, determines the proportions of Y_n and $X^*_{n,n-1}$ that contribute to $X^*_{n,n}$ by combining them in such a way that the estimated variance of $X^*_{n,n}$, represented by covariance matrix $S^*_{n,n}$, is minimised (Brookner, 1998). For example, if measurements Y_n are considered much more accurate than the estimates $X^*_{n,n-1}$, then Y_n will have a stronger influence on the value of $X^*_{n,n}$, and vice versa. Many texts have been written on the Kalman filter; Brookner (1998) presents an excellent introduction to the filter and a general history of tracking.

3.5.2) A Constant-Acceleration Non-Linear Filter

The VRS prototype uses a Kalman filter to predict the position and velocity of a vehicle between GPS measurements. The measurement vector Y_n for the Kalman filter contains the four data values measured by the GPS receiver at iteration n, as shown by Eq. 3.1.

$$Y_n = [x_n \quad z_n \quad s_n \quad \theta_n]^T$$

Eq. 3.1: Measurement vector, containing reported GPS measurements of vehicle: (x_n, z_n) is position, s_n is speed, θ_n is heading

Brookner (1998) gives example matrix notation for the Kalman filter in which the measurement vector and state vector are represented in Cartesian coordinates with an independent time variable. Kim and Oh (2000) describe a *constant-velocity* filter model that is commonly used to model the motion of vehicles. This is effective because a large proportion of vehicle motion is classified as being of constant velocity (Kim and Oh, 2000). It models the two horizontal components of velocity separately, and models the magnitude of each velocity component as a constant function of time over short time intervals. Some advantages of this constant-velocity model are its simple processing cost, very low steady-state error, and the lack of disturbance by higher states during normal operation. One limitation of this model, however, is that its tracking performance is sub-optimal if the vehicle does not travel in a straight line, even if the vehicle's motion is very simple. For example, many curved roads require a vehicle to manoeuvre around them at a nearly-constant turn radius. In this case, rather than splitting a velocity into two components, polar coordinates are more suited to the modelling task: the position of a vehicle's steering wheel dictates its rate of change of heading, θ, and the position of the accelerator pedal directly dictates its speed in steady state, s. A GPS receiver reports these polar coordinates directly.

Here, an improved constant-acceleration filter model is described and implemented to handle the cases when a vehicle travels in a straight line and when it navigates around curves. For the VRS prototype, the following assumptions are made:

- The rate of change of speed of a vehicle is constant between updates (an assumption of constant acceleration). This is likely to be true except when the vehicle intentionally accelerates or brakes suddenly.
- The rate of change of direction (heading) of a vehicle is constant between updates. For example, this is exactly true for a vehicle rounding a curved road shaped as a perfect arc. Many real-world curved roads can be well approximated as arcs over a short distance.

The estimated target trajectory state vector, $X^*_{n,n-1}$, for this constant-acceleration model is given by Eq. 3.2. It is simpler to process the position of a vehicle in terms of its equatorial GPS coordinates than it is in terms of its GPS coordinates. Therefore, the positional coordinates of the vehicle reported by the GPS receiver are first converted on-the-fly to equatorial GPS coordinates (as described in Appendix A) before being filtered. In addition, recall that the GPS receiver reports speed in knots. To yield dimensional agreement with the vehicle position, the reported GPS speed is also converted to units of minutes per second.

$$X^*_{n,n-1} = \begin{bmatrix} x^*_{n,n-1} & z^*_{n,n-1} & s^*_{n,n-1} & \dot{s}^*_{n,n-1} & \theta^*_{n,n-1} & \dot{\theta}^*_{n,n-1} \end{bmatrix}^T$$

Eq. 3.2: Target trajectory state vector: estimated state values at iteration n based on available data at iteration (n-1) and before: (x, z) position (in equatorial GPS minutes), speed s (in equatorial GPS minutes per second), rate of change of speed with respect to time, $\dot{s}$ (in equatorial GPS minutes per second per second), heading θ (in degrees), and rate of change of heading with respect to time, $\dot{\theta}$ (in degrees per second).

The equation that produces the set of filtered estimates at each iteration, $X^*_{n,n}$, is given by Eq. 3.3. Vector $X^*_{n,n}$ contains estimates of parameter values at iteration n which are a function of parameter values at iteration n and earlier, as the measurement vector Y_n is involved in its calculation. Each iteration is separated by a time T, which is the length of time between presentation of consecutive measurement vectors. Here T = 2 seconds.

$$X^*_{n,n} = X^*_{n,n-1} + H_n(Y_n - MX^*_{n,n-1})$$

Eq. 3.3: Kalman filtering equation used to calculate filtered estimate at iteration n (Brookner, 1998, p74)

The transition matrix, Φ, is given in Eq. 3.4. As shown in Eq. 3.5, this matrix is multiplied by the filtered trajectory state vector, $X^*_{n,n}$, to generate the predicted target trajectory state vector $X^*_{n+1,n}$. The transition matrix contains non-linear sine and cosine terms to accommodate the polar coordinates reported by the GPS receiver.

$$\Phi = \begin{bmatrix} 1 & 0 & T\sin(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T) & 0.5T^2\sin(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T) & 0 & 0 \\ 0 & 1 & T\cos(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T) & 0.5T^2\cos(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T) & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Eq. 3.4: Transition matrix

$$X^*_{n+1,n} = \Phi X^*_{n,n}$$

Eq. 3.5: Kalman predictor equation: anticipates future state parameters at iteration (n+1) based on available data at iteration n (Brookner, 1998, p74)

The equivalent set of predictor equations for Eq. 3.5 is expressed in a non-matrix form that is easier to understand by Eq. 3.6. These equations are adapted from the case of a linear Kalman filter to include the non-linear sine and cosine functions of heading θ.

$$x^*_{n+1,n} = x^*_{n,n} + \sin(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T).\left[T.s^*_{n,n} + 0.5T^2.\dot{s}^*_{n,n}\right]$$

$$z^*_{n-1,n} = z^*_{n,n} + \cos(\theta^*_{n,n} + 0.5.\dot{\theta}^*_{n,n}T).\left[T.s^*_{n,n} + 0.5T^2.\dot{s}^*_{n,n}\right]$$

$$s^*_{n+1,n} = s^*_{n,n} + T.\dot{s}^*_{n,n} \qquad \theta^*_{n+1,n} = \theta^*_{n,n} + T.\dot{\theta}^*_{n,n}$$

$$\dot{s}^*_{n+1,n} = \dot{s}^*_{n,n} \qquad \dot{\theta}^*_{n+1,n} = \dot{\theta}^*_{n,n}$$

Eq. 3.6: Predictor equations corresponding to Eq. 3.5, in non-matrix form

Observation matrix M is needed when the coordinate system of observed measurements does not match the coordinate system of filtered parameters. This is true in this case – the state vector $X^*_{n,n-1}$ estimates two states which are not reported by the GPS receiver ($\dot{s}$ and $\dot{\theta}$) – and thus M is defined as shown by Eq. 3.7.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Eq. 3.7: Observation matrix

For each iteration n, the Kalman filter gain matrix, H_n, contained in Eq. 3.3, is calculated according to Eq. 3.8. Covariance matrix $S^*_{n,n-1}$ contains estimates of the variances (uncertainty) associated with each filtered state parameter (diagonal elements), as well as estimated covariances between parameters (off-diagonal elements). At each iteration n, $S^*_{n,n-1}$ is calculated according to Eq. 3.9. In this equation, covariance matrix Q_n contains the estimated variance of a random-walk component that occurs from iteration n to iteration (n+1). In general, the inclusion of Q_n keeps the Kalman filter stable (Brookner, 1998). Here, the purpose of Q_n is to account for an unexpected change of acceleration, $\dot{s}$, of a vehicle, or an unexpected change in the rate at which the vehicle's heading is changing with respect to time, $\dot{\theta}$. To reflect this, the diagonal elements of Q_n corresponding to $\dot{s}$ and $\dot{\theta}$ are made non-zero. The magnitude of each diagonal element is set to match the magnitude of the unexpected change in that element: in general, the higher each value is, the faster the Kalman filter will adapt to the unexpected change. The drawback of increasing the magnitude is that the steady-state predictive error of the filter also increases (Kim and Oh, 2000).

Covariance matrix R_n in Eq. 3.8 contains the estimated accuracy of each measurement defined by Y_n, represented as variances. Thus, here R_n describes the estimated accuracy of a vehicle's current position, speed and heading as reported by the GPS receiver.

$$H_n = S^*_{n,n-1} M^T \left[R_n + M S^*_{n,n-1} M^T \right]^{-1}$$

Eq. 3.8: Calculation of Kalman gain matrix used in Eq 3.3 (Brookner, 1998, p74)

$$S^*_{n,n-1} = \Phi . S^*_{n-1,n-1} \Phi^T + Q_n$$

Eq. 3.9: Calculation of $S^*_{n,n-1}$ on which Kalman gain matrix H_n is based (Brookner, 1998, p74)

3.5.3) Handling Non-Linearity

The predictor covariance matrix equation shown in Eq. 3.9 is intended for a linear Kalman filter. Here the filter is non-linear, and the consequence is that Eq. 3.9 is only valid if we assume that the variances of the filtered estimates, $\theta^*_{n,n}$ and $\dot{\theta}^*_{n,n}$, are virtually zero. This can never be true, even if it was possible to measure heading θ_n with perfect accuracy. In the real world, GPS measurements always incur some non-zero error, and the result is that Eq. 3.9 no longer exactly predicts the covariances, $S^*_{n,n-1}$, of the filtered estimate, $X^*_{n,n-1}$. The resulting filter is an approximated Kalman filter. The reason is that the inclusion of non-linear sine and cosine terms in Eq. 3.4 and 3.6 makes the resulting probability density functions (PDFs) of positional coordinates $x^*_{n,n-1}$ and $z^*_{n,n-1}$ asymmetric. In other words, if the true PDF of these coordinates is Gaussian, which is the best assumption we can make, the filtered PDF is non-Gaussian and skewed to an extent that increases as the variances of $\theta^*_{n,n}$ and $\dot{\theta}^*_{n,n}$ increase. Without elaboration, it is easy to show why a Taylor series truncation of the sine and cosine functions cannot avoid the need for an approximation. The only certainty is that the true variance magnitudes are greater than those predicted by the first two diagonal elements of $S^*_{n,n-1}$ calculated by Eq. 3.9. This can be approximately catered for by making the two elements, corresponding to (x, z) position in the covariance matrix Q_n, non-zero.

Details of all actual filter parameters used and the results of the Kalman filter described here for the VRS prototype are presented in Chapter 7.

3.6) Summary

Accurate positioning and tracking is important in any VRS system to ensure that the correct set of virtual signs are displayed at the correct time. NavStar GPS was chosen as a positioning device for the VRS prototype. By performing a simple data-logging experiment, the accuracy of a GPS receiver was deemed acceptable for this application, although the experimental conditions were ideal. It was not possible to measure the accuracy of GPS in conditions more representative of what might be expected during the operation of a real VRS system (for example, in a vehicle moving between buildings). The prototype uses the NMEA protocol to transfer data between a GPS receiver and the onboard processor (a laptop PC). A Kalman filter accommodates the changing accuracy of GPS measurements and predicts vehicle data during times when it is not reported by the GPS receiver. In the next chapter, we will see how a virtual world containing virtual signs may be represented for efficient processing. We then see how the vehicle data measured by the positioning and tracking subsystem is used to determine the set of virtual signs that should be displayed to a driver at a given time.

CHAPTER 4
Data Subsystem: Virtual Worlds

This chapter presents and explains design decisions regarding low-level data management of a VRS virtual world. In this chapter, low-level data management is concerned with the logical composition of the virtual worlds that contain virtual signs: how a world can be represented and stored efficiently, how good scalability of the world can be achieved and how the world is processed at a given vehicle position to determine the corresponding set of virtual signs that must be displayed. Details of the VRS prototype are presented in later sections.

4.1) Sign Regions

A sign is defined inside a virtual world by placing it within a small localising area known as a *sign region*. Without sign regions, signs would be defined with respect to the entire virtual world, which would make sign processing more complex and extremely inefficient for large worlds. Sign regions can be defined in such a way that they can be efficiently stored and processed. It also makes sense to use sign regions to localise signs because, as is often the case in the real world, signs are grouped in clusters such that the driver observes many signs at the same location. In order to specify that a sign is to appear at a given position, a sign region is first created and the sign is defined with respect to that region. Each region determines how far away a sign first appears in the distance, the duration for which it is visible, and the direction in which a sign is visible, as well as a sign's world position. Each region can contain multiple signs.

Figure 4.1 (p. 4-2) illustrates how a sign region might be defined to display a single Stop sign at the end of a T-junction intersection. The position of a vehicle is insufficient by itself to determine whether the sign(s) within a sign region should be displayed, because different signs apply when a vehicle travels in different directions (for example, on alternate lanes on a two-lane road). Therefore, the set of signs observed by a driver is dictated by the overall motion of the vehicle. A sign appears in view whenever two conditions are satisfied for a given sign region, SR:

1) The vehicle position lies anywhere the area defined by SR, and
2) The vehicle is approaching one predefined side of SR, known as the *major sign axis*.

If these two conditions are satisfied, SR is described as *active*. Sign regions which do meet these conditions are described as *inactive*.

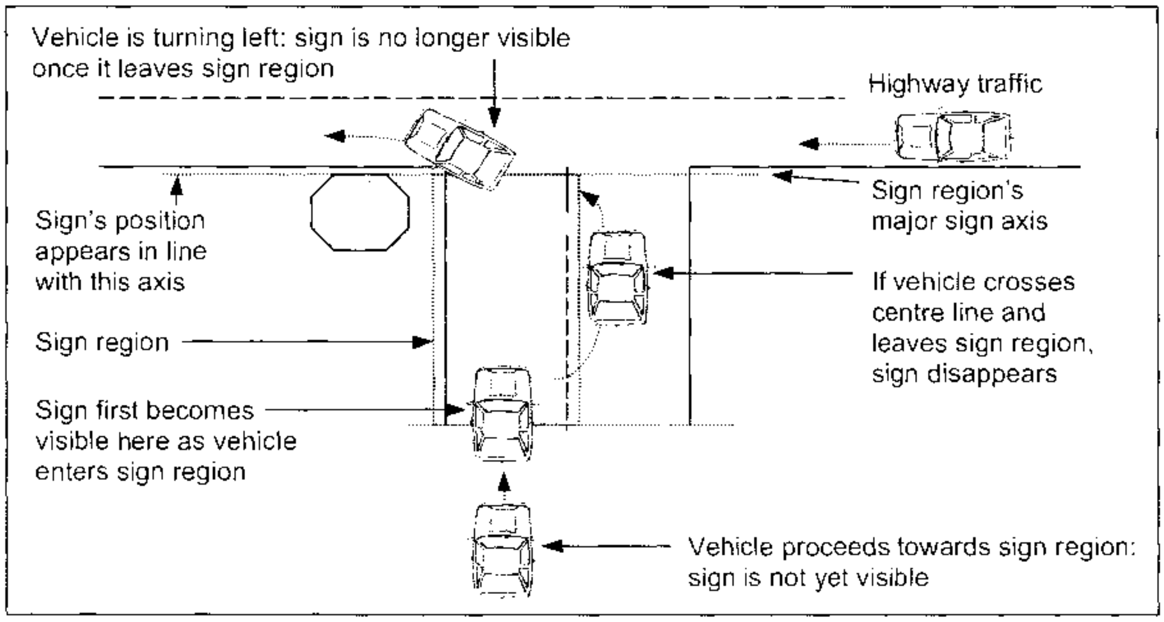

Vehicle is turning left: sign is no longer visible once it leaves sign region
Highway traffic
Sign region's major sign axis
Sign's position appears in line with this axis
If vehicle crosses centre line and leaves sign region, sign disappears
Sign region
Sign first becomes visible here as vehicle enters sign region
Vehicle proceeds towards sign region: sign is not yet visible

Figure 4.1: An example of a rectangle sign region, illustrating the timing of visibility and perceived position of one sign, for a vehicle approaching a T-junction intersection.

The following points apply:

1) Major sign axes must always lie on the perimeter edge of a sign region. If this restriction was relaxed, we would need to add a third condition to the two conditions listed above. In order for a sign to be displayed, the position of a vehicle would have to lie on a predefined *viewing side* of the major axis, and the two conditions listed above would have to be satisfied. Consider the case if this third condition was not imposed. If the major axis bisected a sign region, for example, then a vehicle approaching from either end of the region would see the sign(s) – in most cases, a particular set of signs should be visible from one direction only.

2) The vehicle is assumed to be driving in a forward direction. On its own, a vehicle's heading has no influence on the set of signs displayed at any one time, so the correct set of signs will indeed be generated for a reversing vehicle (those signs located behind the vehicle rather than in front). However, whether a driver observes the signs or not will depend on the hardware employed to display signs. Here it is assumed that a HUD will augment the forward view only. A driver looks through the rear window of the vehicle when reversing, and thus would not observe any virtual signs during this time.

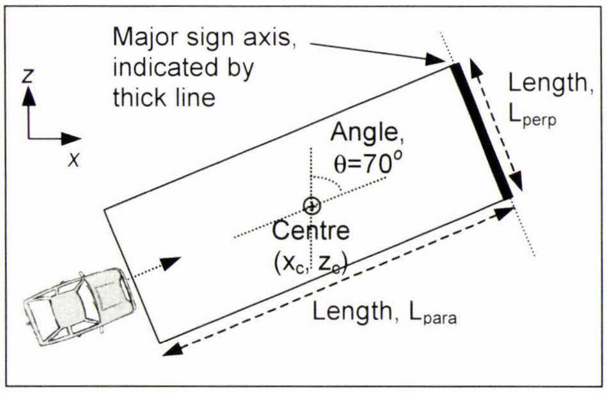
The major sign axis has a second important role: to define the line along which signs will appear to be situated in the world. The distance between a sign and vehicle determines the size of the sign when it is displayed. In this sense, the relative size of the sign face provides a visual cue of the distance between a vehicle and the feature of interest, as for conventional signs. The major axis defines the position of all signs associated with the sign region as any point along the linear axis. That is, the exact three-dimensional world coordinates of each sign are not accommodated. Chapter 5 describes sign position as it relates to presentation in more detail.

4.2) Designing Sign Regions

4.2.1) Rectangle and Arc Sign Regions

A general VRS system must support at least two types of sign region to be useable. Here, rectangles are chosen to cater for straight stretches of road and arcs are chosen to cater for curved road segments. Rectangle sign regions are defined in terms of their centre coordinates, parallel and perpendicular length, and angle (see Figure 4.2.a). The angle defines which one of the four rectangle sides is the region's major axis, and must lie in a range of 0-360 degrees. By convention, all angles are defined as clockwise with zero degrees representing North.

Arc regions are defined in terms of their centre coordinates, inner and outer radii, and two angles (see Figure 4.2.b). To visualise how the angles define the arc, consider a clock whose large hand is rotating clockwise from 0-360 degrees. The first angle is the start angle (known as the *anticlockwise* angle) where the large hand starts sweeping out the arc. The second angle is the end angle (known as the *clockwise* angle) where the large hands stops sweeping out the arc. An arc's major axis must coincide with one of the two straight butting ends of the region.



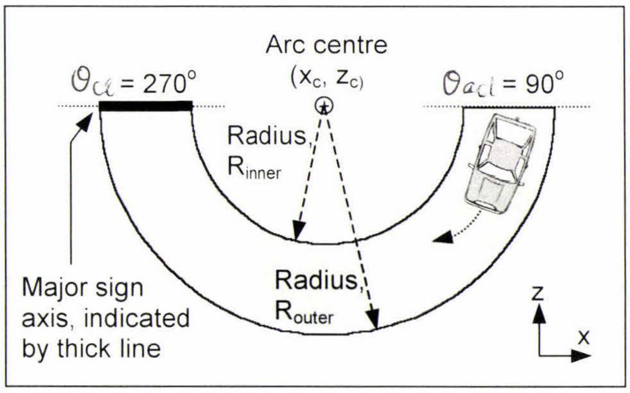


Figure 4.2.a: Rectangle sign region definition. **Figure 4.2.b: Arc sign region definition.**

Rectangle and arc sign regions should be suitable for placing virtual signs in most areas of a virtual world. Each region is identified in the world by its type (rectangle/ arc) and a key number that uniquely identifies it.

4.2.2) Composite Sign Regions

It is not difficult to conceive of situations where the association of signs with single sign regions is unsatisfactory for the purpose of conveying information to drivers. For example, consider the commonly occurring road configuration shown in Figure 4.3 in which a minor road leads onto a major road. A virtual sign is placed at the intersection of the two roads to alert a driver approaching from the east on the minor road to give way to main-road traffic. The sign cannot be defined with respect to long rectangle region SR_{rect1} because the end of SR_{rect1} does not lie at the intersection; if so defined, the sign would appear at the right time but disappear early. Neither can the sign be defined with respect to short rectangle region SR_{rect2}, because then the driver would observe the sign too late and fail to stop in time. The sign certainly cannot be defined with respect to short arc SR_{arc1}, because then the driver observes the sign for a short duration sometime after the driver should first be alerted, and the sign disappears before the intersection is reached. How can this problem be overcome?

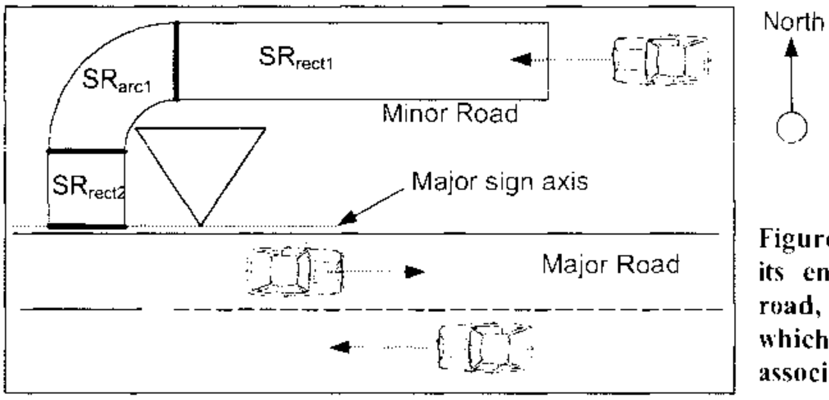


Figure 4.3: A minor road turns at its end to intersect with a major road, signalled by a Give Way sign: which sign region should the sign be associated with?

One solution is to define a third L-shaped region type that encapsulates all three regions. However, there will always be some roads that need to be decomposed into even more rectangle and arc regions (for example, an S-shaped road over a railway crossing) – it would be impractical to try and cater for all possible combinations. A better and more general solution is to allow rectangle and arc sign regions to be linked together to form *composite* regions. To distinguish rectangle and arc regions from composite regions, the former shall be known as *primitive* regions. In Figure 4.3, one composite sign region replaces primitive regions SR_{rect1}, SR_{arc1} and SR_{rect2}, and the Give Way sign is then associated with this composite region. Then when the effective sign distance is to be calculated, the calculated distance is measured correctly and the sign appears in the correct world position.

The primitive regions that comprise composite regions are still associated with their own set of signs. For example, in Figure 4.3, it will be sensible to display a sign at the left-hand end of region SR_{rect1} to indicate a comfortable speed that drivers should slow down to prior to travelling around the curve. Given that such a primitive region SR is active, the set of signs to be displayed includes those signs associated with SR and those signs associated with the composite region that SR is a part of.

4.2.3) Region Length

In a generic virtual world, the length of a given sign region will be largely influenced by:

1) The expected speed of vehicles travelling through the sign region. For example, sign regions located on rural highways should have a greater maximum length to account for the higher vehicle speeds encountered, compared to sign regions in an urban area.
2) The importance of the sign(s) associated with the sign region. A higher priority sign may be associated with a longer sign region to give a driver earlier warning of the sign's message.

4.3) World Entity Mappings

Preparation of an *entity relationship diagram* is a standard technique for describing the mappings between data structures of a software application (Whitten, Bentley and Barlow, 1994). Figure 4.4 illustrates the important data mappings between VRS world entities.

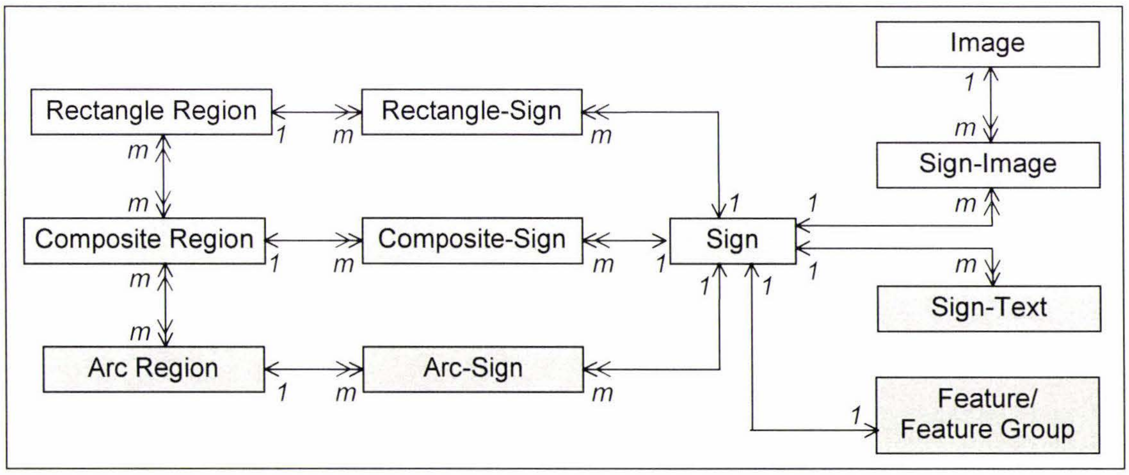


Figure 4.4: Entity relationship diagram of world entities. 1:1 arrows indicate one-to-one relationships, 1:m arrows indicate one-to-many relationships, m:m arrows indicate many-to-many relationships.

Each feature or related group of features is associated with one sign. Multiple signs can appear in a given sign region, and multiple sign regions can contain a given sign. Hyphenated entity names represent many-to-many mappings that have been decomposed into two one-to-many mappings. For example, the *Rectangle-Sign* table stores the set of signs associated with each rectangle sign region. Multiple images can appear on a given sign, and multiple signs can display a given image. Multiple lines of text can appear on a given sign in any specified language. The topic of virtual sign design is discussed in Chapter 5.

4.4) World Compilation and Representation

4.4.1) Introduction

A *source world* database stores data describing all virtual signs and sign regions in a virtual world. Let us assume that it is stored and maintained by a central authority. It is very likely that the source world database will not be a suitable entity for distribution to end-users. The purpose of world compilation is to process the appropriate data in a source world database and reassemble it in an alternate format in a separate *compiled world* database, ready for distribution to an end-user.

The primary reason for world compilation is to cater to the distinct requirements of data providers and data consumers. The source world database may be physically distributed across multiple data stores, may be serving any number of end-users, and may be offline or online depending on the frequency of maintenance. In contrast, the end-user has a very high real-time processing requirement - virtual signs must appear at the correct time – and thus the end-user data store must be online. This is true regardless of whether only an onboard database is stored in a vehicle, or whether an in-vehicle wireless communication system receives data from an offboard source. A given end-user is only interested in signs in their local area and may or may not be granted the ability to edit sign data. World compilation provides an opportunity to:

1) Convert world data into an efficient end-user form.

Primitive and composite sign regions are stored in *region definition* tables in the source world database. Because the database provides the interface via which a world designer enters world data, its tables may be intentionally designed to aid data entry by representing sign regions in their most intuitive form. This form is often not very efficient in terms of space (memory) or time (processing) requirements. As a simple example, here a rectangle sign region is defined in a source world database table in terms of its centre coordinates, angle and parallel and perpendicular lengths, whereas in practice, storing the coordinates of the four corners is more efficient from an algorithmic point of view.

2) Localise the compiled world database depending on a vehicle's location.

For an onboard data philosophy, the entire source world database may be too large to distribute to end-users. For an offboard data philosophy, it may be impractical to continually broadcast data to end-users in real-time. The best compromise may be to update compiled world databases periodically with information relevant to a vehicle's location.

3) Verify the virtual world prior to run-time.

It is much less costly to detect conflicting signs and incorrect sign region definitions before the data is distributed to end-users. In practice, only an extremely low error rate in the data processed by end-users will be tolerable, in the interests of safety and liability.

4.4.2) Non-Recursive Data Structures

Many different options exist for world representation, each varying in terms of efficiency and its space-time trade-off. The simplest possible technique is to store all sign regions in one large list. To search for the set of sign regions associated with a particular point P in the world, the same list is always processed regardless of the coordinates of P, which is very inefficient. Clearly a better data structure is desired to decompose a world of reasonable size into smaller areas to make it manageable and afford a degree of search locality. Let the entire area of a virtual world be represented as a simple rectangular polygon. A very simple data structure is a collection of single-layered lists. Here the world is represented as a two-dimensional space of known measurements, divided into non-overlapping, rectangular spaces (blocks) in grid fashion that completely fill the world space; that is, every point in the world is contained within one and only one block. Blocks may be of varying sizes. Samet (1990) calls this representation a *fixed-grid (*or *cell)* structure. Each block stores a *region list*, which is a linked list of pointers to sign regions associated with the block, in addition to up to eight pointers to its eight neighbouring blocks.

To determine the list of sign regions to be processed for given vehicle coordinates (x, z), the world space is searched in both dimensions using a fast binary search to identify the unique block that contains (x, z). This search is required only for the first iteration. On successive iterations, only the neighbouring blocks of the current block (identified by their pointers) need to be searched to determine whether the new vehicle coordinates (x, z) still lie inside the same block or have entered another block. This assumes of course that the smallest block size is large enough to ensure that the vehicle is incapable of travelling more than the length of a block between iterations. The space-time trade-off is biased in favour of time: the search algorithm executes very quickly, but the world space is not space-efficient. For example, each block stores eight pointers regardless of whether it stores any sign regions in its region list. Also, references to a large sign region spanning multiple blocks are duplicated across all such blocks.

4.4.3) Recursive Data Structures and Decomposition Techniques

A better solution is to use a recursive decomposition data structure that supports blocks of many different areas. A *BSP tree* is one very general technique for dividing a space up into regions (Samet, 1990). At each subdivision stage, the space is divided into two regions of arbitrary size with no restriction on the set of division lines being either orthogonal or parallel (see Figure 4.5.a). The space is thus decomposed into a set of arbitrarily-shaped convex polygons. A *bintree* is a more ordered structure: at each subdivision stage, the space is divided into two-equal sized parts. Starting at iteration one, spaces are divided by a horizontal separation line for odd iterations and a vertical separation line for even iterations (see Figure 4.5.b). A variation of this technique, *adaptive hierarchical coding*, proposed by Cohen, Landy and Pavel (cited by Samet, 1990), relaxes the condition of alternating horizontal and vertical divisions. The decision as to which dimension to divide on then depends on the shape of the complete space or the contents of the space itself. For a *region quadtree*, spaces are divided into four equal-sized regions at each subdivision stage by dividing over both dimensions at once (see Figure 4.5.c). A quadtree is characterised as being able to represent data at multiple levels of resolution. Development of the quadtree structure was motivated by the desire to reduce the amount of data required to represent a space by capitalising on the similarity within local regions of the space (Samet, 1990).

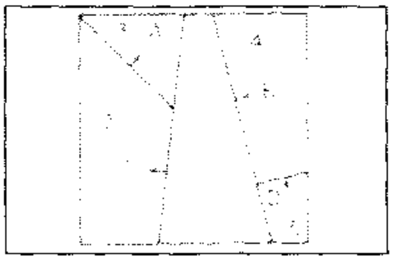


Figure 4.5.a: Example of block decomposition for a BSP tree (Samet, 1990, p25)

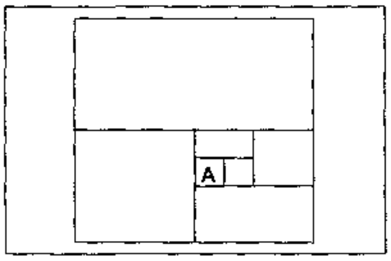


Figure 4.5.b: Example of block decomposition for a bintree

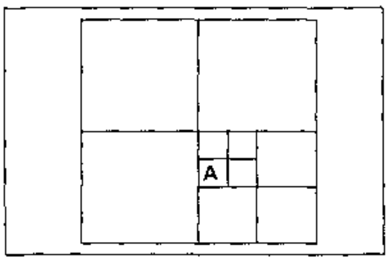


Figure 4.5.c: Example of block decomposition for a region quadtree

One method of characterising specific variations of the standard quadtree is noting whether they are designed to represent point data or areas, and whether the decomposition is regular or non-regular. A *regular* decomposition divides a space into regions according to a consistent rule; for example, that the space will be divided into two equal-sized regions at each point (such as for a bintree). Sign regions are represented as areas by their nature, but it is conceivable that they could be stored in a data structure as points. For example, the centre point of each region could be stored in a quadtree structure, and the parameters that describe the shape of the region relative to its centre could be stored separately. One example of a regular point-based data structure suitable for this task is the *MX quadtree* (Samet, 1990). This subdivides a space into quadrants such that each point in the space lies at the lower-left corner (by convention) of one quadrant (see Figure 4.6.a). An MX quadtree offers very efficient data storage, but it is limited by the fact that the set of points must exist in a discrete coordinate system. In the real world, sign region coordinates may be measured to an accuracy of a few metres, so the resulting system may be approximated as being discrete. However, continuous measurements are less restrictive. A *PR quadtree* is an alternative adaptation of a region quadtree to point data (Samet, 1990): decomposition proceeds until each quadrant contains at most one point (see Figure 4.6.b). In order to represent points efficiently, the coordinates of the point contained in each quadrant Q may be expressed in the local coordinate system of Q.

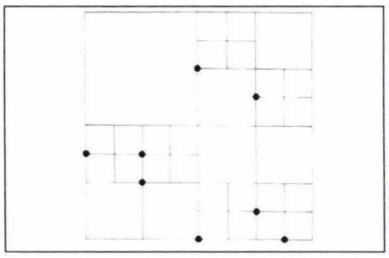


Figure 4.6.a: Example of block decomposition for a MX quadtree (Samet, 1990, p86)

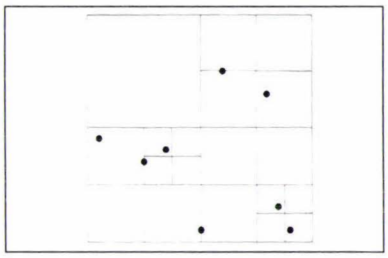


Figure 4.6.b: Example of block decomposition for a PR quadtree (Samet, 1990, p93)

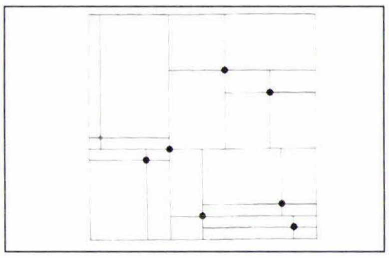


Figure 4.6.c: Example of block decomposition for a point quadtree (Samet, 1990, p49)

Non-regular trees divide the space at each subdivision stage based on the data to be represented by the quadtree. A *point tree* places one point in the space at the centre of each division, as shown in Figure 4.6.c. A less restricted variation is the *k-d tree*, which chooses to divide a space one dimension at a time based on some decision (Samet, 1990).

A more intuitive choice for the representation of sign regions is to choose an *area-based* tree structure; that is, one which treats each region as occupying a finite area rather than as an infinitesimal point. The MX-CIF quadtree is one such structure. It associates each sign region SR with the smallest quadtree block that contains SR in its entirety (Samet, 1990), as shown in Figure 4.7. To determine the set of sign regions corresponding to a world vehicle position (x_i, z_i), the sign region list associated with each quadrant must be searched, for multiple quadrants. Processing begins for the region list of the highest-level quadrant (the largest quadrant that contains all other quadrants) and traverses down one path in the tree until the smallest quadrant containing (x_i, z_i) is reached. Still other area-based options exist, such as the *expanded MX-CIF quadtree* and *R-tree* (Samet, 1990). These structures are useful when one is interested in knowing the set of regions contained within a specific area of the entire space known as a *query window*.

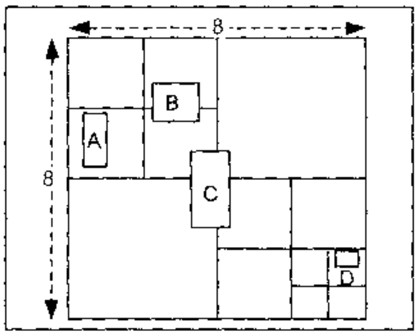


Figure 4.7: Example of block decomposition for a MX-CIF quadtree. Region C's enclosing block is the outer 8x8 border, region B's block is the top-left 4x4 block, region A's block is a 2x2 block, and region D's block is a 1x1 block.

4.4.4) Decomposition Techniques using a Quadtree Data Structure

For a general VRS system, one is interested in knowing the set of sign regions that correspond not to an area but to a single point. This point represents a moving vehicle in a virtual world of sign regions. Here a standard region quadtree is employed to represent the world. This is amenable to an efficient pointer representation and search algorithm. Recall that a quadtree can decompose an area into quadrants at variable resolution, such that in some localised areas the quadrants are smaller than in other areas. Each quadrant of a quadtree is schematically represented as a *node*. A full decomposition creates quadrants at many different resolutions (known henceforth as depths) and thus many different nodes linked together in a tree structure. The top node (*root* node) at a depth of one represents the entire world space, whose rectangular bounds are defined by a world database. Each node at depth d_i stores four pointers to its four decomposed *child* nodes at depth (d_i+1). A leaf node is a node with no children.

Having chosen a suitable data structure for world representation, a decision rule must be defined to direct the actual sequence of decomposition that occurs during world compilation. World compilation assigns each sign region to one or more specific quadrants of a quadtree. Each node in the quadtree stores a region list of pointers to sign regions associated with that node. From an algorithmic point of view, a good decision rule is one that assigns an approximately equal number of sign regions to each node (a balanced tree). That is, the rule should evenly distribute sign regions across nodes in the quadtree even when the distribution of sign regions in the virtual world is far from uniform. To achieve this, decomposition must proceed to a greater depth in areas where sign regions are densely populated compared to where sign regions are sparse. Such a rule is described as *adaptive* because it is dependent on the geometry of the set of sign regions processed. An important measure of a decision rule's efficiency is the mean number of assignments (pointers) to quadrants per sign region. Smaller mean values mean a lower pointer overhead of the quadtree and a higher storage efficiency.

The decision rule is defined by specifying how the assignment of a given sign region SR to quadrants Q_1, Q_2....Q_n takes places. Three options are:

1) SR must be associated with exactly one quadrant Q. One example is to define Q as the smallest quadrant that completely encloses SR. This is the rule imposed by the MX-CIF quadtree described earlier. Although the pointer overhead is minimised, this method is inefficient for regions that straddle high-level quadrant boundaries because such regions will be assigned to nodes very high in the tree. As a result, higher nodes may be associated with many more sign regions than lower nodes, yielding an imbalanced tree.
2) SR can be associated with more than one quadrant, but every such quadrant must be represented by a leaf node. The fact that only leaf nodes are associated with sign regions simplifies processing. The downside is that the mean number of pointers to quadrants per sign region may be very high if a poor choice of leaf quadrant size is made (see Figure 4.8.a for one example).

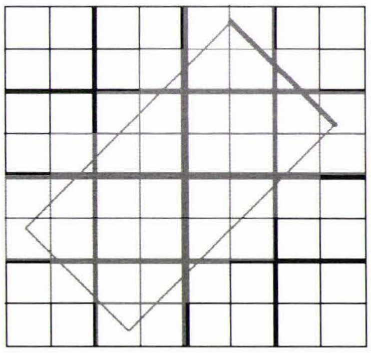


Figure 4.8.a: For option (2), poor choice of leaf node quadrant size associates too many quadrants per region (shaded gray).

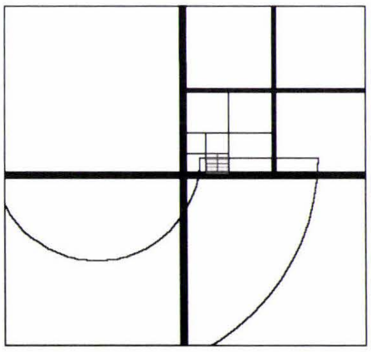


Figure 4.8.b: For option (3), deep decomposition is required (shown here for part of an arc region, shaded dark gray) where a region only just straddles a high-level quadrant.

3) SR must be associated with exactly k sibling quadrants (at the same depth). A sensible choice for k is two or four. Like (1), sign regions can be associated with nodes at multiple depths, and like (2), a sign region may be pointed to by more than one quadrant. One problem is that regions that only just straddle a high-level quadrant

boundary will necessitate a very deep decomposition to satisfy the rule for the small overhanging region area (see Figure 4.8.b for one example). To avoid excessive decomposition, it makes sense to stop decomposition at a minimum quadrant size.

Here, option (2) is chosen for its simplicity and versatility.

4.4.5) Defining Association between Nodes and Sign Regions

For this option, we choose what is probably the simplest and most intuitive association rule. For a sign region SR to be associated with a node quadrant Q, any part of SR must be contained within Q. Given that SR is associated with Q, one of three cases must be true:

1) Q completely encloses SR (see Figure 4.9.a for one example).
2) Q incompletely encloses SR (see Figure 4.9.b for one example).
3) Q is completely enclosed by SR (see Figure 4.9.c for one example).

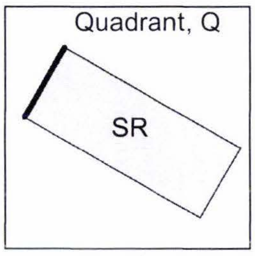


Figure 4.9.a: Q completely encloses SR

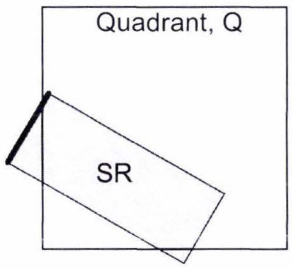


Figure 4.9.b: Q incompletely encloses SR

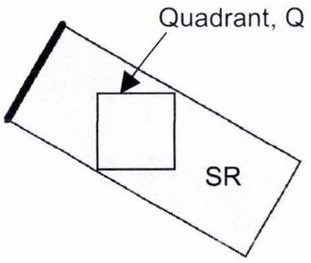


Figure 4.9.c: Q is completely enclosed by SR

How do we formally define the rules that state whether a given sign region SR is associated with a given node quadrant Q? One method that correctly determines the association for both rectangle and arc regions is described in Box 4.1. Examples are given for these region types by Figures 4.10.a and 4.10.b respectively.

1) Split region SR into its four sides.
2) For each side S, determine which horizontal and vertical lines of quadrant boundaries intersect with S, and calculate the set of intersection points P_{set}.
3) For each point P_i in P_{set}, the two adjacent quadrants must be associated with SR.

Box 4.1: Defining association between a sign region SR and node quadrants

The method described in Box 4.1 is in fact robust for any closed shape in general. However, it is only effective if at least one intersection point P_i in P_{set} exists. If no intersections points exist, we must check whether any point enclosed by SR is also enclosed by a quadrant Q. If

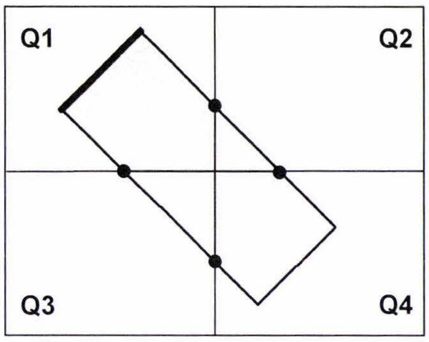


Figure 4.10.a: Rectangle sign region, showing intersection points between quadrant boundaries and the region sides, and quadrant numbering scheme.

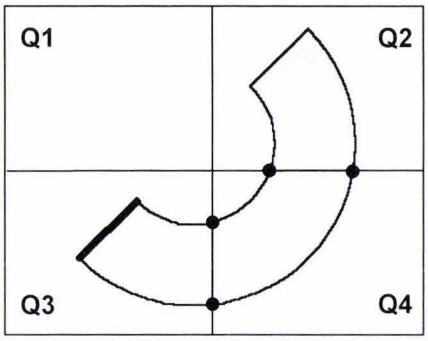


Figure 4.10.b: Arc sign region, showing intersection points between quadrant boundaries and the region sides, and quadrant numbering scheme.

so, Q must completely enclose SR. The node of quadrant Q is found in the quadtree by searching for the coordinates of any point in the region (such as one corner or the centre point). Also, this method requires an extra step to identify those node quadrants completely enclosed by a region, even if steps (1)-(3) of Box 4.1 identify quadrants associated with SR. For each quadrant Q, we check whether any point enclosed by Q (such as its centre point) is enclosed by SR. The task of determining whether a point is enclosed by a region is performed by noting whether a set of inequalities are all true, depending on the region type. These inequalities are given in (C.1) and (C.2) of Appendix C.

In practice, it is sensible to specify the minimum leaf node quadrant size. In general, the size of the smallest permissible quadrant should be of the order of the longest dimension of a typical sign region, or probably larger. Smaller leaf node quadrants significantly increase the mean number of pointers to quadrants per sign region, needlessly reducing storage efficiency. In other words, the case of a node quadrant being completely enclosed by a sign region should occur very rarely.

4.4.6) Methods of Compilation using a Quadtree Data Structure

Two possible methods of world compilation are:

1) Bottom-Up Adaptive Compilation

Compilation proceeds in two stages: decomposition, followed by tree pruning. Decomposition creates all leaf nodes at the same depth. The leaf node quadrant size is pre-selected, which is equivalent to specifying a maximum tree depth, d_{max}. For each sign region SR, the leaf nodes associated with SR are determined and SR is added to the region list of each node. Once complete, the tree is pruned from the bottom upwards where there is a high degree of region commonality between sibling leaf nodes. For example, if four sibling leaf nodes have identical region lists, the four nodes can be deleted and the common list transferred to the parent node, which becomes a leaf node. If three of four sibling leaf nodes have identical region lists, it may also be worthwhile merging the four lists into one and assigning the result to the parent. To improve the merge efficiency, sign regions must be processed in a strictly ascending order of region type and region key number; that is, all rectangle regions must be processed in one sublist first followed by all arc sign regions and then composite regions. The final region list per node is a combination of the sublists of all region types. An example of bottom-up adaptive compilation is shown in Figure 4.11.a.

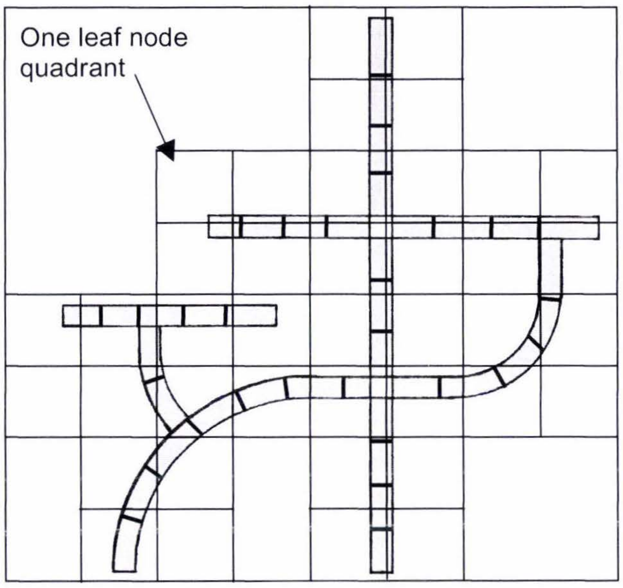


Figure 4.11.a: Bottom-up adaptive decomposition of one world. Sign regions are shaded gray. In this case, no pruning was necessary.

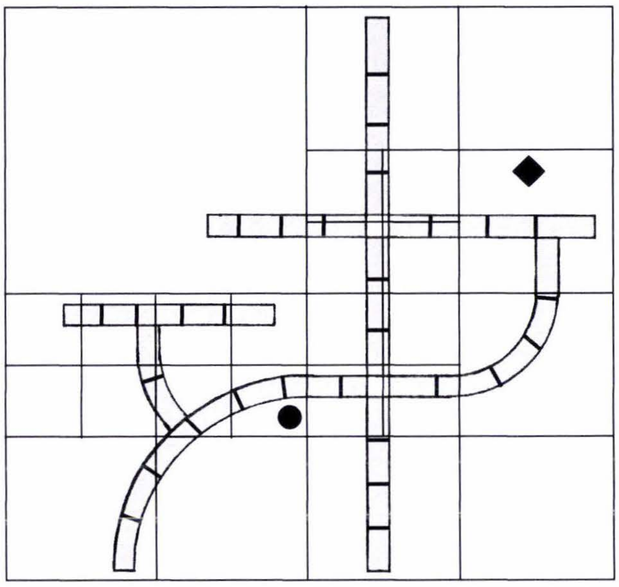


Figure 4.11.b: Top-down adaptive decomposition of the same world, $n_{max\ SR} = 4$. Sign regions are shaded gray.

2) Top-Down Adaptive Compilation

Top-down compilation does not predefine the depth at which leaf nodes reside; rather, a maximum number of sign regions associated per leaf node, $n_{max\ SR}$, is predefined. The process begins with a single leaf node (root node), representing the entire world space. For each sign region SR, a search is undertaken down the tree for the node(s) that is associated with SR and currently a leaf node, starting at the root node. When a region is to be added to a current leaf node N whose region list already contains the maximum number permitted, up to four children are created and become new leaf nodes. The region list of node N is then split amongst its new children. An example of top-down adaptive compilation is shown in Figure 4.11.b.

Of the two options, bottom-up compilation is simpler because the first process of decomposition is not adaptive, whereas the second tree pruning stage removes redundant nodes adaptively. Also, bottom-up compilation is potentially less computationally intensive in practice for large numbers of sign regions because the decomposition stage takes place at the same tree level, and ordered region lists can be merged very efficiently during tree pruning. In comparison, top-down compilation may end up determining the quadrants associated with a given sign region multiple times (at different tree depths) if the region lists of many high-level nodes need to be split (for large numbers of sign regions). However, top-down compilation is favoured here as the preferred option, for the following reasons:

1) The condition that each leaf node stores up to a maximum number of sign regions is very useful from an algorithmic point of view, because the memory and processing requirement per node is bounded and set in advance.
2) Memory allocation occurs only as needed when creating new nodes. In contrast, bottom-up compilation may require much more memory during decomposition than is required to store the final tree structure.
3) The maximum leaf node depth does not need to be specified in advance.
4) Top-down compilation is the only truly adaptive technique because adaptation occurs on-the-fly. In rural world areas, sign regions are likely to be large and few in number, yielding large leaf node quadrants. In inner city areas, sign regions are naturally likely to be more densely populated and smaller to suit the confined spaces and lower vehicle speeds, yielding smaller leaf node quadrants 1-2 levels deeper in the tree.

The compiled quadtree created by top-down adaptive decomposition of the world shown in Figure 4.11.b is given by Figure 4.12. At each level, each tree node represents one quadrant, Q, and has up to four child nodes, each representing one child quadrant contained by Q. The node numbering scheme of child quadrants is the same as that given by Figures 4.10.a and 4.10.b in (4.4.5). Consider the tree traversal involved in locating the leaf node of the two black shapes in Figure 4.11.b. Starting from the root node, a search for the small black diamond traverses the node sequence 1, 2, 4, whereas a search for the small black circle traverses the sequence 1, 3, 2, 4.

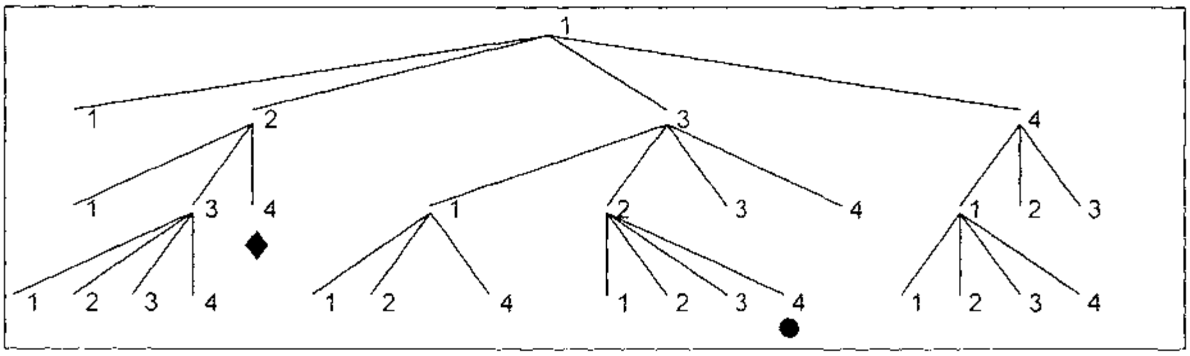


Figure 4.12: Compiled quadtree for top-down adaptive decomposition of world shown in Figure 4.11.b

4.4.7) Logical Database Structures of Compiled Worlds

As mentioned in (4.4.1), the compiled world database generated as a result of world compilation is stored separately. Here, non-leaf nodes in the compiled quadtree are stored in a separate database table to leaf nodes. Each node is represented by one record in its corresponding table. First, all non-leaf nodes are assigned unique key numbers to identify each node. These numbers act as the *primary key* in the database table because they permit unique identification of each record in the table. The root node is assigned a number of one by convention, and all further nodes are assigned positive integers in ascending order. Next, all leaf nodes are similarly assigned unique key numbers. It is convenient to assign negative integers to leaf nodes to distinguish them from the positive-numbered non-leaf nodes. Records in the leaf node table describe the set of sign regions associated with each node quadrant. Figure 4.13 shows an example of node numbering.

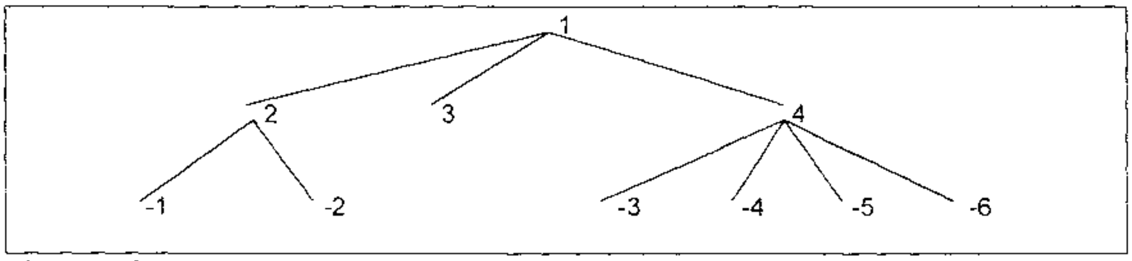


Figure 4.13: Example of primary key number allocation to quadtree nodes

The structure of each record in the non-leaf database table is shown by Figure 4.14.a. Each record specifies the node's key number, plus up to four key numbers of the node's children. If the leaf child of a node contains zero sign regions in its region list, there is clearly no need to include the child node in the database. A simple way of specifying that a child node of a given node does not exist is to set the node's child number to zero. The structure of each record in the leaf database table is shown in Figure 4.14.b. One record exists in the table per sign region associated per leaf node. Each record specifies the node's key number, a second key number that permits sign regions associated with the same node to be distinguished, and the region type and region key number of one region associated with this node.

Node key #	Child 1 key #	Child 2 key #	Child 3 key #	Child 4 key #

Example:

1	2	3	0	4

Figure 4.14.a: Structure of one non-leaf node record in compiled world database, with an example for node 1 in Figure 4.13

Node key #	Index	Sign region type	Sign region key #

Example:

-4	1	2	5

Figure 4.14.b: Structure of one leaf node record in compiled world database, with an example for node –4 in Figure 4.13

Figure 4.15 shows the logical data structure of the information pointed to by one leaf record. Each sign region in the region list of the leaf node is represented by an ellipse. Each sign region points to a set of location data that defines the region geometry (as defined in [4.2]) plus the major sign axis of the region. In addition, each sign region points to a set of sign data that defines which signs shall be displayed when this region becomes active.

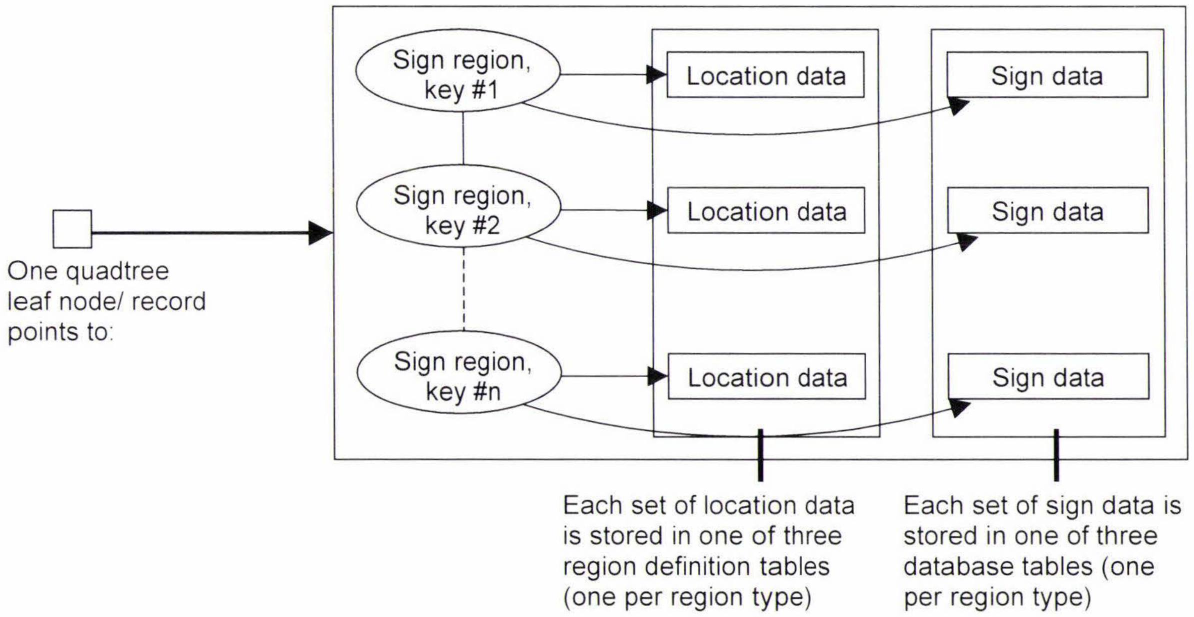

Sign region, key #1
Sign region, key #2
Sign region, key #n
Location data
Location data
Location data
Sign data
Sign data
Sign data
One quadtree leaf node/ record points to:
Each set of location data is stored in one of three region definition tables (one per region type)
Each set of sign data is stored in one of three database tables (one per region type)

Figure 4.15: Logical structure of data pointed to by each leaf node in compiled world database

4.4.8) Alternative World Compilation, Using a Nine-Tree

A more complicated world compilation algorithm uses both a quadtree and a derivation of a quadtree, which shall be known as a *9-tree*, to represent the world. This is described in detail in Appendix D. A 9-tree divides a two-dimensional space into its nine overlapping blocks, as shown in Figure 4.16.a. Whereas the quadtree method of world representation described up to this point yields good memory space and time efficiency, the 9-tree algorithm makes still better use of memory. Therefore, this algorithm may be preferred when local memory is scarce and/or particularly large virtual worlds are to be compiled. For the previously described quadtree method, it is likely that a given sign region will be contained in up to four region lists of four separate leaf nodes. If regions are frequently contained in multiple region lists, excessive pointer duplication results and the storage efficiency of the quadtree suffers.

A 9-tree solves this problem by ensuring that a given sign region SR is associated with exactly one block B only. By convention, B is chosen as the smallest block that completely encloses SR. This philosophy is the same as for the MX-CIF quadtree described earlier. However, it was mentioned in (4.4.4) that one potential problem with this quadtree is the tendency to assign too many regions to high-level nodes due to the regions spanning high-level quadrant boundaries. To visualise this, consider four small rectangular sign regions in a world that are spread across one or more quadrants, as shown in Figure 4.16.b. Even though each region is very small, the smallest enclosing quadrant Q for each is very large.

As a result, the time taken to process the region list for higher-level quadrants may be long. When it is considered that quadrants are recursed at multiple resolution levels (scales), this effect can become significant. In other words, the slight gain in storage efficiency afforded by an MX-CIF quadtree may be well and truly offset by the increased processing time required. A 9-tree substantially reduces the extent of this problem by intentionally overlapping its blocks, with the result that B can often be much smaller than would be possible with a quadtree.

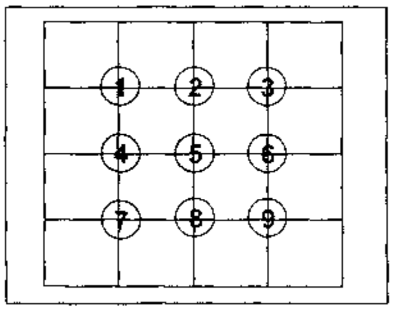


Figure 4.16.a: 9-tree decomposition of a space. Each node number is shown at the centre of its block, where one block contains four squares. Node 4's block is shaded gray as an example.

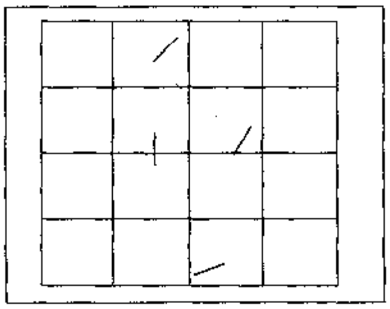
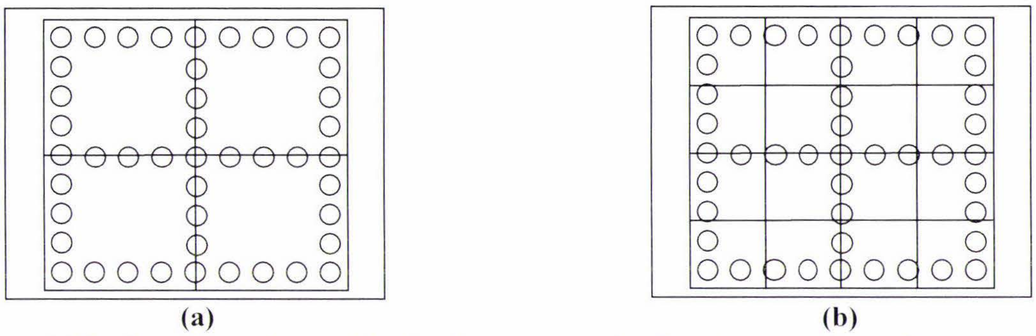


Figure 4.16.b: Four rectangle sign regions are spread across multiple quadrants of a quadtree. The smallest quadrant Q that encloses each region is the highest level quadrant.

To illustrate the potential improvement in storage efficiency effected by a 9-tree, consider the world shown in Figure 4.17.a-b comprised of 45 small circular regions. Such a world is not representative of a typical virtual world and circular regions are not supported by the VRS prototype, but the world is a good example in this case. Consider the case when a standard region quadtree, as described in (4.4.4) to (4.4.6), is utilised to store all sign regions in leaf node region lists at depth two. Four quadrants point to 16 sign regions each in their region list, which means a total of 64 pointers to sign regions must be stored at a depth of two. If a 9-tree is utilised, only 45 pointers to regions must be stored (one pointer per region), and every region is completely enclosed by a smallest enclosing block B at a depth of two in the tree. Blocks 1, 3, 7 and 9 contain seven pointers each, blocks 2, 4, 6, 8 contain four pointers each, and block 5 contains one pointer to a region, giving a mean number of regions to process per region list of five. Thus the 9-tree saves 30% space and 69% processing time compared to a quadtree of the same depth. These results cannot be taken literally, however, because there are other small overheads associated with a 9-tree. In practice, for real worlds in which sign regions are commonly spread across multiple quadrants, a 9-tree often yields good gains in storage efficiency at the expense of slightly increased processing time.


(a) (b)

Figures 4.17.a-b: Example world. (a) shows a standard region quadtree decomposition to a depth of two. (b) shows a 9-tree decomposition to a depth of two. In both figures, the quadrants/ blocks at depth two are the same size; however, they overlap for the 9-tree decomposition, as indicated by Figure 4.16.a.

Due to the greater complexity of its implementation, a 9-tree was not investigated further for the representation of virtual worlds. The remainder of this chapter assumes that a standard region quadtree is used for this purpose, as described in (4.4.4).

4.5) Determination of Active Sign Regions

4.5.1) Overview

Box 4.2 summarises the process of determining which sign regions are active when a vehicle is located at a given world position. The output of this process is $L_{\text{active SR}}$, a list of active sign regions; that is, those regions which currently contain the vehicle and whose major sign axes are being approached by the vehicle.

On application start-up:

1) When the first vehicle position (x_0, z_0) is processed, the quadtree leaf node that corresponds to this position is unknown. Therefore, search the quadtree stored in the compiled world database for the leaf node whose quadrant contains (x_0, z_0). Once found, this leaf node is known as the *current* leaf node. Define $L_{active\ SR}$ as an empty set.

2) For all sign regions in the current leaf node's region list, test whether each region SR is active or not in a sequential fashion:
   a) If it is active, add SR to list $L_{active\ SR}$.
   b) If it is inactive, do nothing.

3) Return list $L_{active\ SR}$.

After application start-up:

4) When a new vehicle position (x_i, z_i) is to be processed, check whether it is contained within the quadrant of the current leaf node, where the latter is remembered from the previous iteration:
   a) If it is not, search the quadtree stored in the compiled world database for the new leaf node whose quadrant contains the current vehicle position.
   b) If it is, no new quadtree search is needed.

5) Define $L_{active\ SR}$ as an empty set. Return a new list of active regions $L_{active\ SR}$ according to steps (2)-(3) above.

Box 4.2: Summary of active sign region determination for a given world position

After a search for the first vehicle position (x_0, z_0) has been completed, the quadtree does not need to be searched from the topmost root quadtree node for each new vehicle position (x_i, z_i). The interval between measurements will be short (1-2 seconds for unfiltered data received from a GPS receiver), and during this time a vehicle is unlikely to travel any significant distance in comparison to the size of the virtual world. Therefore, on this time scale, each successive position (x_i, z_i) is likely to lie in the same leaf node quadrant as the previous position (x_{i-1}, z_{i-1}). On occasions when it does not, it is usually sufficient to traverse back up the tree only 1-2 levels and back down again to find the corresponding new leaf node. This technique saves processing time if the virtual world is very large because the tree does not need to be searched from the topmost root node every time.

4.5.2) Memory Caching Strategy

Caching is an important technique for improving the real-time performance of software algorithms (Silberschatz and Galvin, 1994). As information is to be processed, it is copied from slower storage to a faster storage area, the cache, on a temporary basis. When data is to be processed, a check is first made if it is stored in the cache. If not, the data is retrieved from slow storage and stored in the cache on the assumption that there is a high probability it will be needed again soon. If it is, the data is retrieved directly from cache. Cached information can be retrieved much faster from the cache than it can from slow storage.

In this case, the slow storage is represented by a secondary storage device on which the compiled world database resides. One can argue that the access time of modern hard drives is fast enough (typically ten milliseconds or less) to negate the need for another cache. However, the total time to query a very large compiled world database on secondary storage may be much larger, in the order of seconds. The real-time nature of a VRS system demands that delays of this length are avoided where possible. Also, the type of storage device cannot be pre-supposed, and it may well be that a specific VRS implementation will opt for a secondary storage device with a much slower access time, such as a CD-ROM drive. Regardless, it is good software engineering practice to minimise secondary storage access and process a localised working data set in fast solid-state memory. This is particularly true in a moving vehicle where secondary storage has to contend with the harsh operating environment. Solid-state memory contains no moving parts and is therefore less affected by vibration and shock.

Two caches are described here. The *quadtree cache* stores the current quadtree state. As the search for a new leaf node proceeds in steps (1) and (4a) in Box 4.2, each non-leaf node record at each level in the search path is found in the database and its data retrieved. As the search progresses down the tree, the data stored in each non-leaf record is cached by rebuilding that branch of the quadtree in memory. Traversal to a deeper non-leaf node triggers creation of all its child nodes in memory. When the search concludes and a leaf record is reached, one complete branch of the original quadtree built during world compilation has been built in cache memory. An example is shown in Figures 4.18.a and 4.18.b, using the compiled quadtree shown in Figure 4.12 earlier. When a new quadtree node is to be found in step (4a) of Box 4.2, no database access is required to traverse back up the quadtree because the branch is currently cached. Only when the search traverses down a different branch will the database once again need to be accessed.

When the search for a node concludes and a leaf record is reached, the information pointed to by the leaf record is retrieved from the database and stored in a second *region cache*. The logical memory structure of the region cache matches the database structure presented in Figure 4.15, as shown in Figure 4.19. Each sign region associated with the leaf node is stored as a sign region object in memory. The location data associated with each sign region is cached at the time that the region object is created. The sign data associated with each sign region remains uncached for the time being. In step (2) of Box 4.2, each sign region is processed in turn to determine whether it is currently active or not. This is done by processing the cached location data of each region object in turn. If a region is found to be inactive, no

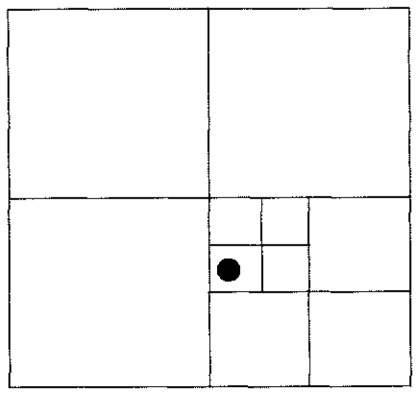


Figure 4.18.a: Vehicle's current position is represented by the black dot

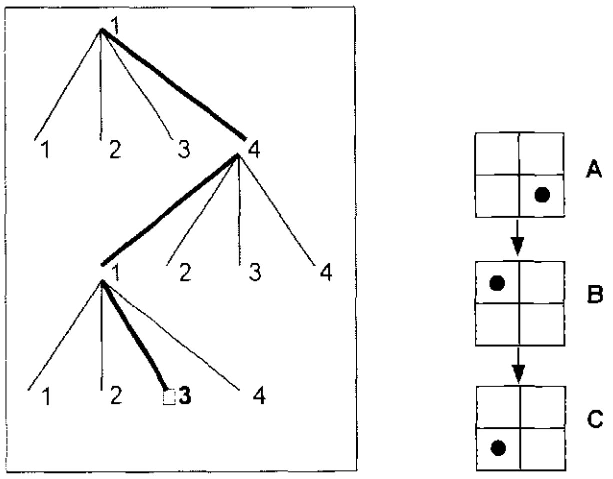


Figure 4.18.b: Quadtree structure created in memory as the search in the database for the current vehicle position progresses (left); recursive search for each child quadrant (right).

The search begins at the root node. At each step, the child quadrant that contains the current vehicle position is selected, beginning with the bottom-right child quadrant (represented by the black dot at step A). Within this quadrant, the vehicle lies within its top-left child quadrant, represented by the black dot at step B. Within this quadrant, the vehicle lies within its bottom-left child quadrant, represented by the black dot at step C. At this point, a leaf node is reached (shaded gray), and the search concludes.

action is taken and the next region is processed. If, however, the region is found to be active, the signs associated with this region are about to be displayed. Only at this time is it necessary to retrieve the sign data corresponding to this region from the compiled world database and cache it. Specifically, the useful information cached is the set of signs defined for this region, plus definition of the region's major sign axis. The latter information is used to calculate the distance between a vehicle and the region's signs.

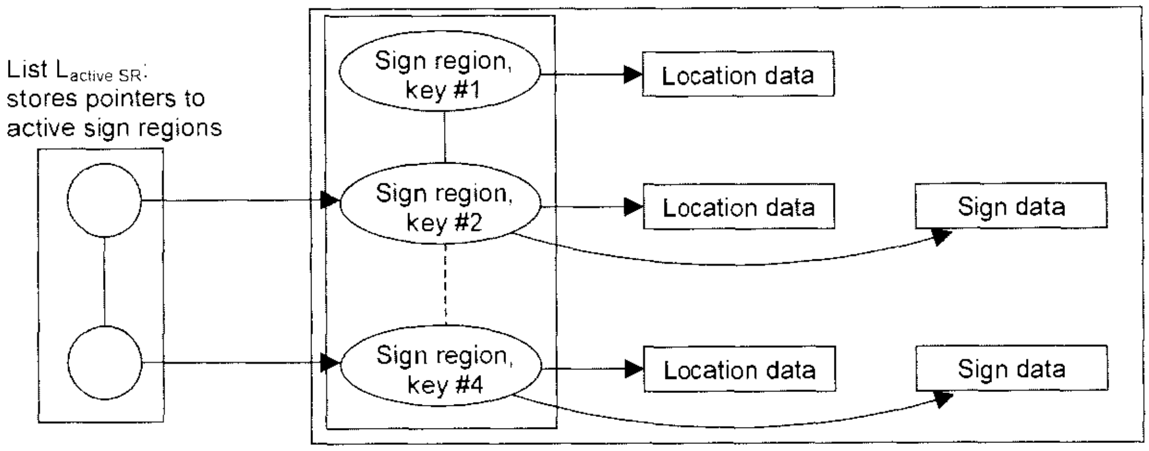


Figure 4.19: Logical memory structure of region cache. Sign data has not been cached for region 1 yet because this region has not yet become active since it was stored in the cache. Sign data has been cached for regions 2 and 4, as they are currently active.

By retrieving the region list of the current leaf node from the compiled world database only once, a lengthy database query is avoided for all further instances when a new vehicle position is processed, until the current leaf node changes. For each iteration i, the list of active sign regions generated, $L_{active\ SR}$, is remembered. When a new list of active sign regions is generated at iteration (i+1), the new list contents are compared with the list remembered from the previous iteration. If the two lists are found to be identical, no active list is returned; rather, a flag is set to indicate that the list of active regions has not changed since the previous iteration. This simplifies subsequent processing, as is described in Chapter 6.

When the vehicle crosses from one leaf quadrant to another, a new leaf node becomes current. One option is to simply discard all region data (region object, location data, and possibly sign data) in the cache corresponding to regions associated with the outgoing leaf node to make way for the region data of the incoming node. A better option is to discard only the data for sign regions which are not currently active at the time of leaf node transition. This is because if a given sign region is active immediately before the leaf node changes, it is very likely that the region spans across the quadrant boundary into the new quadrant, and thus it is very likely that the region will still be active when the next vehicle position is processed.

In this case, one is interested in making the best use of local memory allocated to the cache. Rather than removing data from the cache to coincide with a change in leaf node, the most sensible option is to remove it on a per-region basis when all available cache memory has been exhausted. New region data for incoming leaf nodes is still loaded from the database when the leaf node changes. The cache management strategy is thus generalised and a replacement rule must be defined to identify which region data will be removed from the cache when the cache fills up completely. It is worth noting here that because the cache is stored in local memory, its contents will be lost between power-downs of a VRS system. Let us define the time between power-downs when a VRS system is in use by a driver as one *session*. The best cache replacement rule depends on the statistics of sign regions, such as the mean number of times a driver visits each sign region per session and the elapsed time since each region was visited. Of the set of sign regions visited by a vehicle in one session, it is most likely that each region will be visited once or twice. A driver travelling a non-return journey will visit a number of sign regions during that session, in which case the number of visits per visited region will be one. Another likely trip is a return journey in which sign regions are visited a second time on the return leg of the journey. If the forward leg of the journey exactly matches the return leg, the number of visits per visited region will be two in most cases. Providing that a driver is not hopelessly lost, it is unlikely that a region will be visited more than twice for any given session. A suitable cache replacement algorithm for these circumstances is a simple *first-in, first-out (FIFO)* algorithm. For this algorithm, the region data replaced in the cache is always that data stored in the cache the longest time ago (that is, the oldest data). This assumes that given that a sign region SR is visited, it is no more likely to be visited again in future than any other region. Furthermore, we assume that this is true even if SR's data happens to be stored in the cache when SR is visited.

4.5.3) An Example of Memory Caching

Let us assume for the purpose of an example that a specific VRS hardware implementation has enough onboard cache memory to store data for only four sign regions at one time (in practice, the memory capacity will be much greater). Consider the small virtual world modelled on a roundabout, as shown in Figure 4.20. In this world, all sign regions are primitive (non-composite).

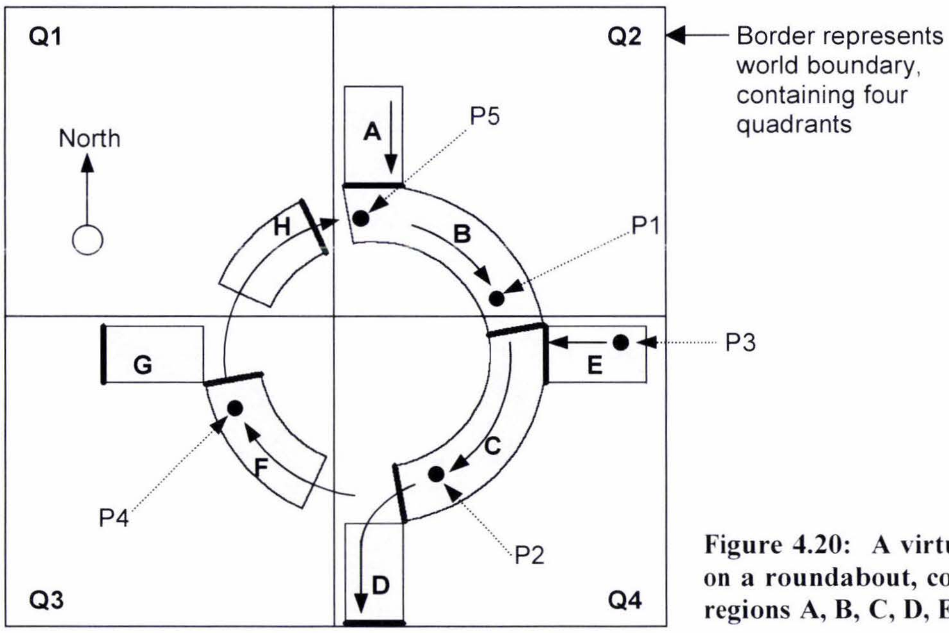
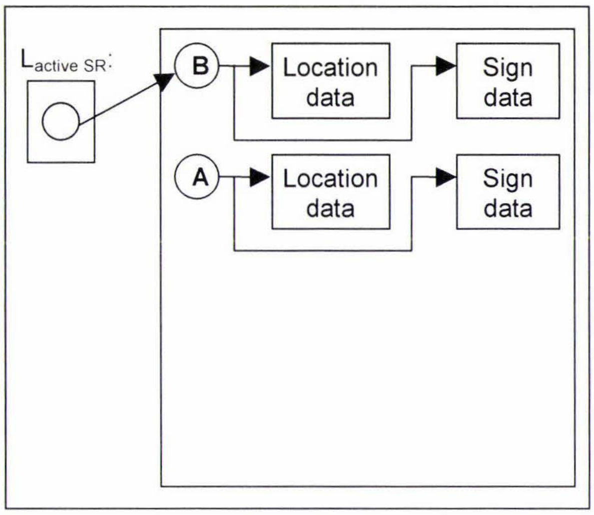


Figure 4.20: A virtual world modelled on a roundabout, containing eight sign regions A, B, C, D, E, F, G, H

A vehicle is in the process of moving from North to South via regions A, B, C and D. At present, it has proceeded through sign region A onto the roundabout and entered region B (region B is currently active). The cache memory state when the vehicle is at point P1 is shown in Figure 4.21.a. The vehicle is about to leave quadrant Q2 and enter Q4. Both location data and sign data for regions A and B are loaded in the cache. Consider the vehicle at a later point P2. The cache memory state at this point is shown in Figure 4.21.b.



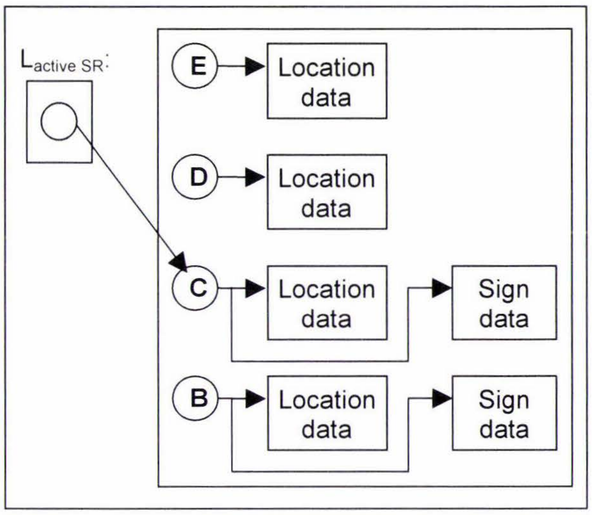


Figure 4.21.a: Memory state of region cache when vehicle reaches point P1

Figure 4.21.b: Memory state of region cache when vehicle reaches point P2

When the current node changed from Q2 to Q4, location data for regions C and D was first loaded from the compiled world database because data for these regions was not already in the cache. The cache was then full, but data for region E was yet to be loaded. To make space, the oldest region data, that for region A, was removed from the cache. Sign data for region C is cached because region C is currently active; conversely, no sign data has yet been loaded for regions D or E because they have not yet been active since their data was stored in the cache. The vehicle exits the roundabout and proceeds south through region D.

Now consider the vehicle re-emerging from the east at point P3, travelling through region E (region E is currently active). The cache memory state at this point is shown in Figure 4.22.a. Sign data is loaded for all four regions B, C, D and E associated with current leaf node Q4, as all four regions have been active at some time during their lifetime in the cache. As the vehicle enters the roundabout, it visits region C a second time. Because data for region C is still in the cache, no database look-up is necessary. Consider the vehicle at a later point P4. Here the vehicle is about to move from current leaf quadrant Q3 into Q1, and region F is currently active. The cache memory state at this point is shown in Figure 4.22.b. Data for regions F and G has displaced data for regions B and C in the cache, even though region C was recently visited. Sign data for region G is never loaded because this region is not visited by the vehicle during its journey. Finally, consider the vehicle at a later point P5, before it exits the roundabout to proceed North. Data for regions A and B was not found in the cache so was reloaded from database, displacing data for oldest regions E and F (not shown). Whereas both location and sign data is cached for region B, only location data is cached for region A because this region has not yet been active since its data was stored in the cache.

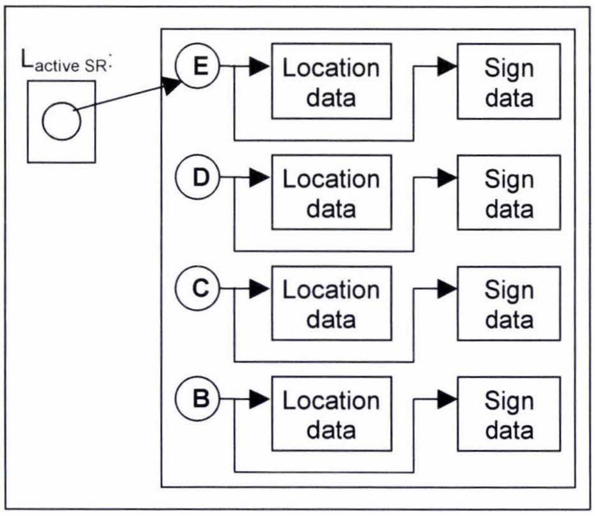


Figure 4.22.a: Memory state of region cache when vehicle reaches point P3

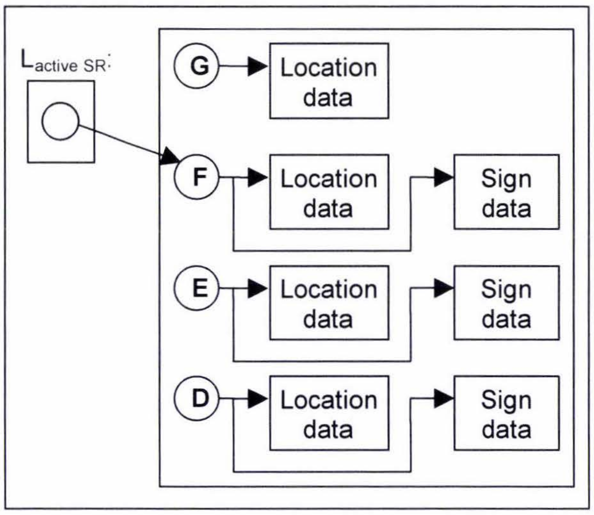


Figure 4.22.b: Memory state of region cache when vehicle reaches point P4

4.6) Prototype Data Subsystem: Low-Level Data Management

4.6.1) World Compilation

The VRS prototype employs the top-down compilation algorithm described in (4.4.6) for a standard region quadtree. Implementation of a 9-tree world representation was not considered due to its relative complexity and the fact that the VRS prototype was designed to test very small virtual worlds (the benefits of a 9-tree are only realised for large worlds). The sequence of steps performed in the top-down compilation algorithm is described in Box 4.3.

Each quadtree node stores a count value. A value of zero indicates that a node is a leaf and no sign regions have yet been assigned to it. A positive value indicates that a node is a leaf, and the value specifies the number of sign regions that have been assigned to it. A value of –1 indicates that a node is non-leaf (that is, it is not associated with any sign regions). $n_{max\ SR}$ defines the maximum permissible number of sign regions that may be assigned to a leaf node whose quadrant is not of minimum allowed size. Leaf nodes whose quadrants are of minimum allowed size can have any positive count value; that is, more than $n_{max\ SR}$ sign regions may be assigned to these nodes.

For each sign region SR in the world, call recursive function Store(SR, root node).

Definition of function Store(SR, node N):

```
Check count value of node N:

a)  If count = -1:
        For each child quadrant N_i of node N's quadrant:
            Check if SR is associated with quadrant N_i. If so, call Store(SR, N_i).

b)  If count < n_max SR or count > n_max SR:
        Add SR to node N's region list, increase count by one.

c)  If count = n_max SR:
    i)  If node's quadrant is of minimum allowed size, add SR to node N's region list, increase
        count by one.

    ii) If node's quadrant is not of minimum allowed size:
        Create four children of node N.

        Reassign regions currently associated with node N to child nodes:
        For each sign region SR_j in node N's region list:
            For each child quadrant N_i of node N's quadrant:
                Check if SR_j is associated with quadrant N_i. If so, call Store(SR_j, N_i).

            Empty region list of node N and set its region count = -1.

            For each child quadrant N_i of node N's quadrant:
                Check if SR is associated with quadrant N_i. If so, call Store(SR, N_i).
```

Box 4.3: World compilation algorithm of VRS prototype

4.6.2) Memory Caching Strategy

The caching strategy described in (4.5.2) is implemented in the VRS prototype using a binary tree and a separate node list. A binary tree is equivalent to a quadtree except that each node has up to two children rather than four (Azmoodeh, 1990). The memory available to store location data and sign data for cached regions is represented as two arrays. The number of elements in each array equals n_{cache}, the maximum number of sign regions that can be stored in the cache at one time. The four data structures are shown in Figure 4.23 for a cache state in which data for six sign regions is currently stored in the cache.

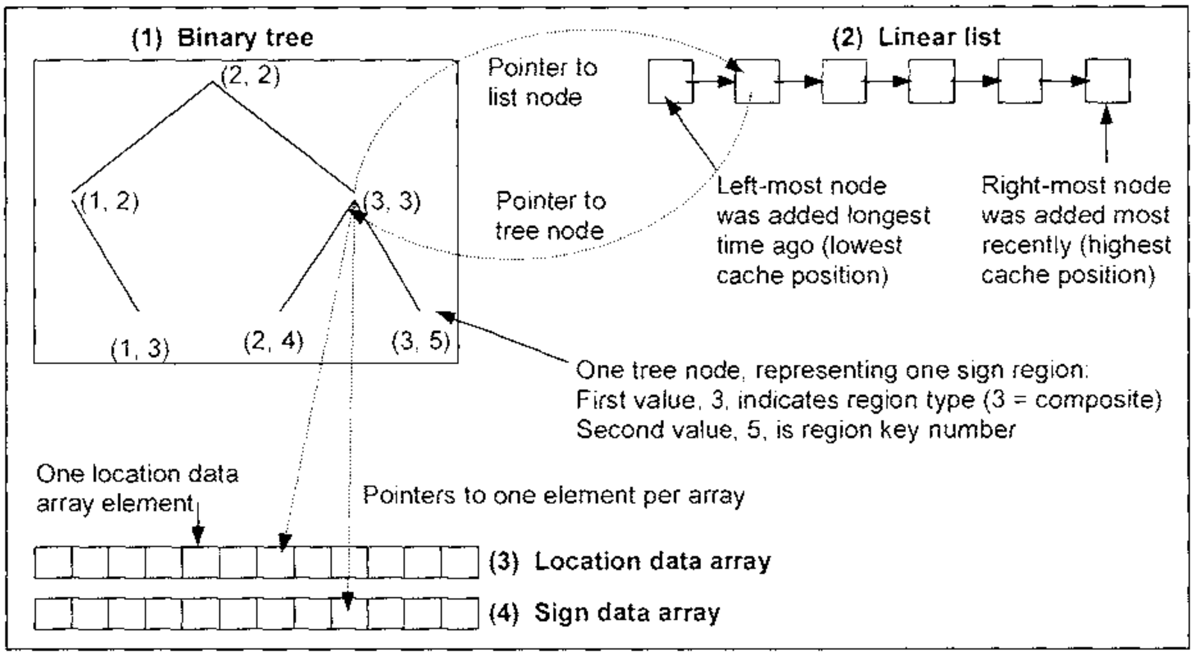

(1) Binary tree
(2, 2)
(1, 2)
(3, 3)
(1, 3)
(2, 4)
(3, 5)
Pointer to list node
Pointer to tree node
(2) Linear list
Left-most node was added longest time ago (lowest cache position)
Right-most node was added most recently (highest cache position)
One tree node, representing one sign region: First value, 3, indicates region type (3 = composite) Second value, 5, is region key number
One location data array element
Pointers to one element per array
(3) Location data array
(4) Sign data array

Figure 4.23: VRS prototype cache data structures. For clarity, only the set of pointers for one binary tree node, representing sign region (3, 3), is shown.

Each node in the binary tree represents one cached sign region. The purpose of the binary tree is to efficiently determine whether a sign region is currently stored in the cache, and if so, where its node is located in the tree. Once found, the node data stored in the two arrays can be processed as described in (4.5.2). Each tree node is uniquely identified by the type (rectangle/ arc/ composite) and key number of one sign region. For convenience, the three region types are enumerated: rectangle regions are assigned type one, arc regions are assigned type two and composite regions are assigned type three. Each tree node stores:

1) One pointer to its parent node (except for the root node, which has no parent).
2) Up to two pointers to its child nodes.
3) One pointer to its corresponding node in the list.
4) One pointer to an element in the array where location data is stored.
5) One pointer to an element in the array where sign data is stored.

A *subtree* is itself a tree whose root node is the child of another tree. Let one paired combination of region type-key [RT_1, K_1] be greater than another paired combination [RT_2, K_2] if and only if $RT_1 = RT_2$ and $K_1 > K_2$, or $RT_1 > RT_2$. During one session in which the contents of the cache continually change, the tree nodes are always ordered according to the following rule: for each non-empty subtree T, all the nodes in the right subtree of T have paired combinations of region type-key number greater than the root node of T, and all the nodes in the left subtree of T have paired combinations of region type-key number less than the root node of T (Azmoodeh, 1990). It can be readily confirmed that the tree shown in Figure 4.23 satisfies this rule. Arranging a binary tree in this manner allows a specific region, as identified by a specific paired combination of region type-key number, to be found in the tree using an efficient binary search.

The purpose of the list is to efficiently identify the next sign region whose data should be removed from the cache when data for a new region is to be added to the cache. Each list node stores one pointer to its corresponding node in the binary tree and one pointer to its neighbouring right-hand node next in the list (except for the last node in the list, which has no next node). At any one time, the left-most list node represents the sign region that should next be replaced in the cache when a new sign region is to be added. The right-most list node represents the sign region that has been most recently added to the cache.

The algorithm listed in Box 4.4 describes how the cache is updated when a new quadtree leaf node quadrant in the compiled world database becomes current:

```
For each sign region SR associated with new quadtree leaf node quadrant Q:

1)  Rebalance the binary tree if the height of the left and right subtrees of any tree node differs by
    >= a certain number, k. Periodic rebalancing of the tree is required to prevent the tree
    degenerating towards a linear list, which cannot be searched nearly as efficiently as a tree.

2)  Search the binary tree for SR:

    a)  If found, SR is already stored in the cache, and thus its data does not need to be
        retrieved from the compiled world database.
    b)  If not found:
        i)  If cache is not full:
                Find where the new node should be added to the binary tree to preserve the
                ordering, and add the new node. Assign the next available elements in the
                location data array and sign data array to this node. Add a new node to the
                right-most end of the list.

        ii) If cache is full:
                Delete the left-most node in the list, and delete the corresponding node in
                the binary tree. Remember the element assigned to this node in the location
                data array and sign data array.

                Find where the new node should be added to the binary tree to preserve the
                ordering, and add the new node. Assign the elements in the location data
                array and sign data array previously assigned to the deleted node to this
                new node. Add a new node to the right-most end of the list.
```

Box 4.4: Sign region cache algorithm

When a new session is begun, the cache is empty – no nodes exist in the binary tree or list. When a new sign region is processed, a new node is added to the tree in the correct location to preserve the ordering, and a corresponding node is added to the list. While the tree and list contain fewer than n_{cache} nodes each, the cache is not full and both data structures grow as new regions are processed. Once the tree and list contain exactly n_{cache} nodes, addition of a new region to the cache removes the oldest node in both the tree and list, and adds a new node to both data structures to represent the newly added region. Thus, once the cache fills up, the amount of memory it consumes does not change. When a sign region SR becomes active, the element in the sign data array referenced by the node corresponding to this region is processed. If no sign data is stored by the array element yet, it is retrieved from the database and stored in the array element.

4.8) Summary

A sign region is a convenient entity for grouping signs together and defining how and when those signs will be made visible to a driver. Composite sign regions allow greater flexibility when designing sign regions by joining primitive sign regions together. There are a number of different methods available to represent sign regions in a virtual world - only a handful have been described here. Each method varies in the way it trades space efficiency for processing efficiency. A recursive quadtree data structure is utilised for the VRS prototype, owing to its simplicity and good scalability. Adaptive world compilation is an important technique for adjusting the quadtree structure to suit specific areas of a virtual world. Adaptive compilation of a quadtree ensures that the task of determining the set of active sign regions for a given vehicle position and velocity is simple and efficient. By storing data in cache memory as it is retrieved from secondary storage, the performance of a VRS system is improved.

Before the low-level database design of the prototype can be examined, it is necessary to consider the important design decisions of the main objects of interest contained in a virtual world – virtual signs. This topic is covered in the next chapter.

CHAPTER 5
Data Subsystem: Virtual Signs

In this chapter, the design of virtual signs is discussed, followed by a detailed description of how virtual signs may be processed efficiently in a VRS system. The specific design and method of processing implemented for the VRS prototype are presented. Finally, the chapter concludes by summarising the low-level database design of the prototype. The databases described in this latter section collectively store all data contained in a virtual world, including virtual signs, and thus encompass all of the material covered in Chapter 4 and this chapter.

5.1) Sign Design

5.1.1) Presentation Style

What is the best-suited presentation style for virtual road signs? For virtual road signs to be effective, they should appear to approximately mimic the behaviour of real physical signs. That is, they should appear long before a vehicle reaches the feature they represent, convey the illusion of fluid motion as a vehicle approaches, dispose of their image cleanly and promptly, and be as readable as physical signs. Should the exact sign position be reproduced in three dimensions, or does an approximation suffice? Will a sign react to small movements of a vehicle, or will it cater for the general direction of motion only? Should an attempt be made to simulate peripheral vision depending on a vehicle's approach angle? These are probably the most important design decisions regarding sign design.

The layout of information on virtual signs should be as simple as possible to avoid placing unrealistic demands on the driver's visual processes, especially when the traffic situation is complex or the driver is unskilled. Each sign should serve one purpose only. In a laboratory-based study, Parkes and Coleman (cited by Ashby and Parkes, 1993) found that the use of directional symbols such as arrows consistently ranked higher in terms of favourability than printed text amongst those drivers surveyed. Schraagen (1993) suggests that only one navigation-oriented instruction should be presented at a time to prevent overload. In one experiment, navigation instructions in the form of arrows or verbal commands seemed to be equally effective. Almost half of those drivers studied preferred arrows over text-based road signs, and a quarter suggested a dual approach in which arrows are used in cities and textual signs are used on highways. Popp and Farber (1991) also concluded that the symbolic presentation of navigation information is superior to other presentation modes such as text explanations.

The ability of a sign to attract attention is determined by its conspicuity (Spijkers, 1991). Virtual road signs can be expected to accommodate the same visual content as conventional signs. The use of colour, design of graphics, symbols and text, the layout of visual elements and the amount of information are likely to be similar. However, because virtual signs are computer-generated, there are new opportunities to animate their content. One method of helping drivers avoid information overload is to assign a priority to each virtual sign. The priority indicates the relative importance of each sign's message, ranging from least-important to indication-of-emergency. With virtual road signs, the priority of signs can be dynamically changed to prioritise some messages, such as those that indicate immediate danger, over others, depending on the total information available at a given time. For example, flashing or scrolling text could be used for this purpose. Such techniques are useful for grabbing a driver's attention, but care must be taken to ensure that the effect is not overdone to avoid excessive driver distraction. The level of conspicuity cannot exclusively be attributed to properties of the sign itself, as other factors such as the sign's surrounding environment, driver expectations, rate of sign occurrence and type of information also have an effect (Spijkers, 1991). The weather and time of day have a significant impact on conspicuity as noted by Summala and Hietamaeki (cited by Spijkers, 1991), who found that the effect of a warning flasher which accompanied several road signs disappeared in sunny weather.

The positioning and display subsystems of a VRS system will largely influence the choice of presentation style. The choice of display device dictates the field of view through which a sign can appear. Whereas a conventional HUD may have a narrow field of view, the field of view of a head-mounted display may be much larger and be able to accommodate peripheral vision. Another issue is the degree with which the displayed information appears to 'blend in' with the real-world view. For example, Todoriki *et al.* (1994) found that by overlaying a large guiding arrow on the road in the correct position as a vehicle entered an intersection, the average recognition time was 60% shorter than when a plan view of the intersection (which appeared out of place in the real world) was displayed. Perhaps the most effective way of making a sign appear to blend in with the real-world view is to attempt to reproduce the exact three-dimensional coordinates of a sign relative to a vehicle. However, here this is considered impractical for reasons listed below:

1) It is unnecessary to reproduce the exact sign coordinates in order to convey a sign's message effectively.
2) It would be exceedingly complex to try and reproduce virtual signs exactly in relation to a vehicle; for example, in response to every dip in the road and sideways vehicle movement. The amount of computation required would be excessive.
3) The chosen vehicle positioning device may not offer the precision and sensitivity required to render the exact coordinates of a sign relative to a vehicle. Certainly, no such positioning system was available for the VRS prototype. GPS inaccuracy, coupled with the relatively long period between measurements, simply does not offer the level of precision and sensitivity demanded. As mentioned in (3.5), Kalman filtering mitigates this problem but does not solve it. At a display rate of 10-25 fps, a filtered GPS tracking system can only make a best-guess prediction as to the location of a vehicle relative to a virtual sign. Kim *et al.* (1998) found that even using differential GPS (a more accurate version of standard GPS), the positional accuracy of one outdoors AR system was too poor to yield acceptable registration between the real-world view and superimposed images.

Fortunately, it is sufficient to faithfully reproduce only the distance between a virtual sign and a vehicle to achieve a realistic effect. As mentioned in (4.1), the world position of a given sign is defined as any point on the major axis of the sign region that contains the sign. In other words, a sign's position is defined with respect to one dimension only; each sign is invariant to a vehicle swerving or dips in the road. If anything, this is actually an advantageous trade-off, as the sign stability improves readability at the expense of only slightly impaired realism. Perspective transformation is not only useful for permitting the driver to instantly estimate the distance between his/her vehicle and a sign – it is also very effective at producing the illusion of motion. When a sign first appears, the distance at which it appears is calculated as the distance between the vehicle's current position and the sign's major axis. This calculation applies while the sign remains in view.

For a rectangle sign region, the distance is calculated as straight-line distance, as shown in Figure 5.1.a. Providing a driver continues to proceed towards the major sign axis, the apparent positions of signs associated with a rectangle sign region do not change. Unlike a rectangle region, the apparent position of signs associated with an arc region changes as the vehicle proceeds around the region. That is, each sign does not appear to be anchored to a fixed point on the roadside. When a vehicle initially enters an arc region, virtual signs appear to be

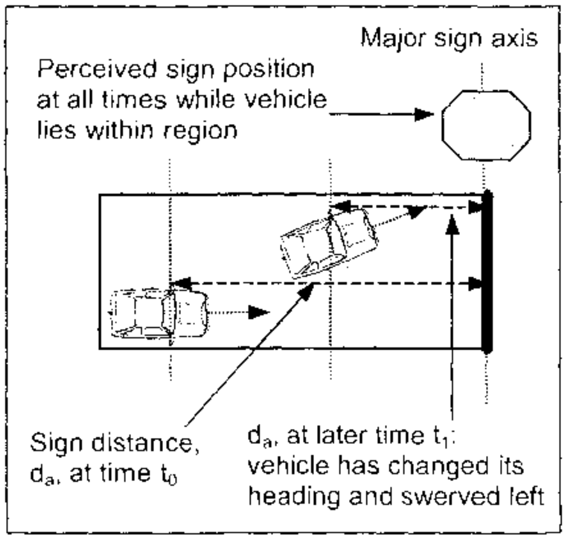

Major sign axis
Perceived sign position at all times while vehicle lies within region
Sign distance, d_a, at time t_0
d_a, at later time t_1: vehicle has changed its heading and swerved left

Figure 5.1.a: Perceived sign position for a driver travelling through a rectangle sign region. Straight-line distance measurement is invariant to yaw, pitch and roll angle of vehicle, as well as any motion in axis parallel to major sign axis.

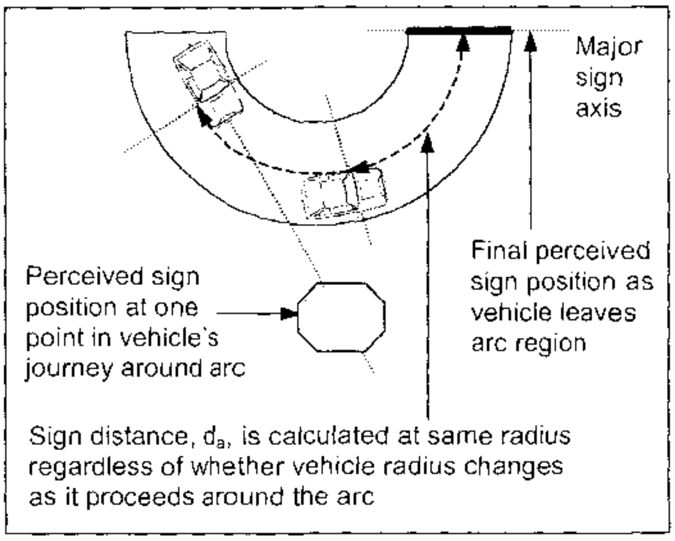

Major sign axis
Final perceived sign position as vehicle leaves arc region
Perceived sign position at one point in vehicle's journey around arc
Sign distance, d_a, is calculated at same radius regardless of whether vehicle radius changes as it proceeds around the arc

Figure 5.1.b: Perceived sign position for a driver travelling through an arc sign region. In the real world, arc width will typically be smaller relative to arc radius than shown.

located straight ahead of the vehicle. As the vehicle continues towards the major axis, signs appear closer but exhibit no lateral movement relative to the vehicle. Near to when the vehicle is about to leave the sign region and the signs appear nearly full-size, the apparent sign position lies approximately in line with the major axis. This behaviour is illustrated in Figure 5.1.b. Sign distance is calculated as the curved distance around the arc region, from the vehicle's current position to the major sign axis. One option is to assume that a vehicle travelling through an arc sign region is most likely to travel at a constant radius with respect to the arc's centre. However, one undesirable consequence of this technique is that if the vehicle's radius appreciably increases as it rounds an arc region, the sign distance will increase and signs will appear to move away from the vehicle. A better and simpler option is to calculate the sign distance purely as a function of the angle made by the vehicle relative to the arc centre. It is easiest to calculate the actual curved distance as if the vehicle is travelling at a standard radius through the arc at all times (see Figure 5.1.b). Then if the vehicle's radius does actually change, the sign distance continues to decrease but the rate at which the vehicle appears to approach the sign (with respect to time) slows down.

The advantage of this display technique for arc sign regions on very tight road segments (for example, on a roundabout) is that the position at which a virtual sign is last observed (at its largest size) can lie much nearer to the feature of interest (for example, a roundabout off-road), while still giving a driver ample warning of the impending sign. To be truly effective and avoid confusing the driver, he/she must be familiar with this effect. In particular, the driver must be aware of when the effect will occur (in short-radius arc regions), and when it will not occur (in rectangle regions).

When the vehicle leaves a sign region, virtual signs associated with that region disappear instantly, although leaving the signs visible for a short duration at their largest size is one option that would present a driver with the best opportunity to interpret the signs when they are most readable.

5.1.2) Sign Face Design

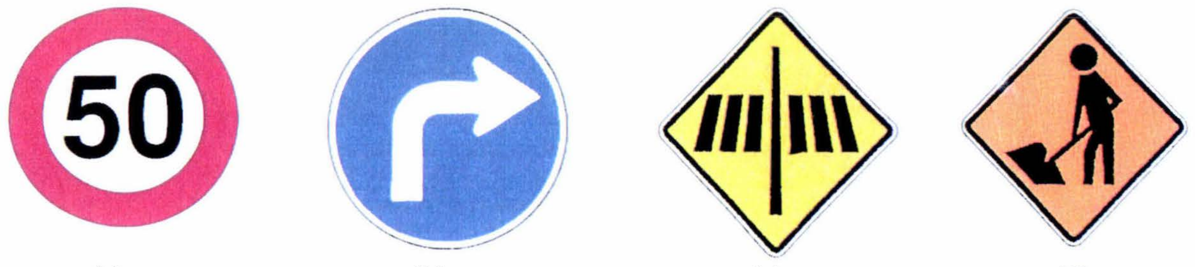
In general, the same principles are important when designing virtual road signs as they are when designing conventional road signs. Three main design issues are considered important when designing virtual road signs:

1) Readability.

The visual layout of signs should be simple, with an aim to convey a sign's message as early as possible. With conventional road signs, the readability of a sign in ideal visibility conditions is limited by the resolving ability of the human visual system. With virtual road signs, the resolution and refresh rate of the display device may be limiting factors. Therefore, a good sign design should offer incremental recognition, providing information to a driver gradually. For example, the colour of a sign's backboard can generally be recognised earliest, long before its textual message is readable. Also, colour can be assimilated to some extent by peripheral vision. The backboard shape is recognised later, followed by symbolic images, and finally text.

2) Consistency.

Consistent sign layout improves readability and reduces the recognition workload of drivers. Consistent use of backboard colours and shapes permits drivers to associate these visual cues with information regarding the type of message conveyed by a sign. For example, internationally standard road signs are divided into three categories (New Zealand Ministry of Transport, 1992). Regulatory signs are divided into two subcategories. Prohibitory regulatory signs tell drivers what they must not do, and are always circular with red and white face colours. Mandatory regulatory signs tell drivers what they must do, and are always circular with blue and white face colours. Permanent hazard warning signs are diamond-shaped with yellow and black face colours; temporary hazard warning signs are diamond-shaped with orange and black face colours. Examples of conventional signs of these types are shown in Figure 5.2.a-d. It makes sense that virtual road signs should adhere to these conventions, in addition to introducing their own specific conventions to cater for the wider range of information they may provide.



(a) (b) (c) (d)

Figures 5.2.a-d: Examples of standard road signs: (a) prohibitory regulatory sign: driver must not exceed 50 kilometres per hour; (b) mandatory regulatory sign: driver must turn right; (c) permanent hazard warning sign: pedestrian crossing ahead; (d) temporary hazard warning sign: roadworks ahead.

3) Flexibility.

In order to accommodate any imaginable content on a sign face, the design must be made entirely flexible in terms of its elementary visual content. A virtual sign backboard may be transparent (no backboard) or one of many predefined colours. Its backboard may be triangular, rectangular, hexagonal, circular, or one of many other standard shapes. It may be rotated to display at any angle. Each sign will support images and lines of text to convey the sign's message. Unlike text, for which there is usually only enough room on a sign for one or two languages, images are intended to be universally recognisable even if the viewer cannot read. Also, images may be recognised faster than the associated text on a sign.

5.1.3) Other Design Issues of Virtual Signs

Timing

Like conventional signs, drivers need time to attend to the messages on virtual signs, understand them, make a decision as to whether action is required, and finally carry out the action. A virtual road sign must be displayed well before it is intended for recognition. The use of sign regions, as described in Chapter 4, ensures that drivers receive ample advance warning of a virtual sign's existence. A warning time of fifteen to twenty seconds is not unreasonable for tasks such as turning corners (Brown and Hook, 1993).

Obviously, timing of sign display must coincide with road intersections and the location of facilities which may be of interest to a driver. Too early sign presentation places an extra burden on the driver's memory capacity and increases his/her mental workload. Too late presentation increases the temporal demands upon the driver, as demonstrated by Ashby, Fairclough and Parkes (cited by Alm, 1993). Each sign has a minimum time over which it must be displayed to be effective, and a maximum time before the message must be delivered (Brown and Hook, 1993). Increasing the viewing time improves recognition performance; short viewing times are particularly detrimental when signs are arranged in a complex configuration (Spijkers, 1991). In a complex configuration, a viewing time of 350 milliseconds led to the same mean recognition performance as a 200 millisecond viewing time in a simple sign configuration, in one experiment.

Multiple signs in close temporal proximity compete with one another for the driver's attention (Claus and Claus, 1974). Brown and Hook (1993) argue that such competing messages contribute to driver workload and should be avoided. Furthermore, a competing message increases the time required to read and react to a message and increases the likelihood of failure to correctly interpret the message. Therefore, it is sensible to limit the number of virtual road signs that can be simultaneously displayed. One way of reducing the problems associated with competing messages is to assign a higher priority to some signs, as noted in (5.1.1). Brown and Hook (1993) also note that sequences of messages become troublesome at or near complex intersections or highway interchanges. In these cases, because the driver must make a sequence of manoeuvres in a brief period of time, it is necessary to communicate the complete sequence of instructions to the driver before the manoeuvres begin.

Position

The position of a virtual road sign is dictated by the position of the major sign axis of a sign region. A near-corner site, where a sign is located before the crossing of a conventional right-angle intersection, is advantageous owing to the fact that it is the first sign perceived by a driver as he/she approaches the intersection, and thus likely to be read first and remembered (Claus and Claus, 1974). The disadvantage is the possibility of reduced visibility due to buildings and other structures (such as trees) located adjacent to it. Far-corner sites are situated across an intersection from oncoming traffic. They tend to be visible from a wider angle and greater distance, due to the open space of the intersection, but at the same time, they are less readable and more susceptible to other visual distractions located around an intersection such as traffic signals and advertising messages. Non-traffic oriented signs, such as signs indicating the presence of a nearby facility (such as a service station), should be placed 1-2 miles from the turn-off to the advertised service when possible (Claus and Claus, 1974). Distances greater than three miles limit the effectiveness of signs whose message tends to the short-term needs of road users, such as a toilet stop. However, for services that tend to long-term needs such as motels and petrol stations, a distance of 20 miles is not excessive, especially when such services are sparse (for example, when driving through a desert).

5.2) Sign Compilation

5.2.1) Overview

The visual content of each road sign will accompany the sign region definitions in the compiled world database. When a sign is to be rendered on-screen, the simplest option is to retrieve this data from the relevant database tables and draw the visual elements (sign backboard, text and images) sequentially. However, this is inefficient for two reasons:

1) The database look-up is relatively slow because parameters defining the sign face content must be retrieved from secondary storage every time the sign is to be drawn.
2) Redrawing all visual elements every time the screen is to be refreshed is clumsy. It is more convenient and potentially more efficient to build (compile) a sign once to form a single object, rather than continue to represent a sign in terms of its individual elements.

The solution to both (1) and (2) is to store all visual sign information in cache memory as it is retrieved from the database.

5.2.2) Memory Caching

In the same way that fast local memory was used as a cache in (4.5.2), a memory cache is again used here to improve the efficiency of data access, by avoiding the time penalty associated with slow secondary storage access after the first retrieval. Before discussing caching strategies for the storage of visual sign content, it is necessary to describe the composition of the sign data that was cached in Chapter 4 (see Figures 4.21.a-b and 4.22.a-b). Each virtual sign is identified by two fields in the database:

1) A major key field. This stores an integer that represents the major sign category. For example, possible sign categories may be weather hazard signs, parking signs and accommodation signs (pointing to hotels or motels, for example).
2) A minor key field. This stores an integer that uniquely identifies a specific sign within each major sign category.

The provision of major and minor key fields, rather than a single key field, permits signs to be organised into convenient groups; it is possible that further key fields may be nested. Together here, the major and minor key fields uniquely identify a virtual sign in the compiled world database.

Recall that sign data is loaded into cache memory for each sign region when it first becomes active, if it does not already exist in the cache. This data is arranged as an array, $A_{\text{sign data}}$. Each element of the array corresponds to one location on-screen where a sign may request to be rendered. As the sign data is retrieved from the database, each sign is assigned to the element in array $A_{\text{sign data}}$ that corresponds to its requested screen location. Compilation of each sign does not need to occur at this time and can be delayed until the sign has been allocated a screen location, immediately before the sign is to be rendered (allocation of signs to screen locations is described in Chapter 6). Here, compilation of one sign incurs three database queries: one query to retrieve the set of images associated with the sign, one query to retrieve the text associated with the sign, and one query to retrieve general data for the sign, such as its shape and colour. In Figure 4.4 of (4.3), these operations query tables *Sign-Image*, *Sign-Text* and *Sign* respectively. The time taken to perform these three queries, combined with the fact that signs are displayed multiple times in short succession after compilation, makes signs ideal candidates for caching.

In addition, the images displayed on signs in cache are candidates for caching. The amount of data required to store one image is typically much greater than the amount needed to represent the rest of the sign. In practice, the transfer rate of modern secondary storage devices permits even large images (hundreds of kilobytes) to be retrieved very quickly (in fractions of a second). If sign images can be compiled as part of a sign's definition, there would appear to be little need for image caching, as the image would be cached as part of the overall sign. However, many signs that serve a slightly different purpose, but follow the same theme as other signs, often share one or more common images; for example, conventional parking signs. Because a single image may be shared across signs, it makes little sense to duplicate the image across signs as part of each compiled sign definition. Thus, the logical decision is to store images in cache memory separate to the memory used to cache compiled sign definitions. Each cached sign then points to one or more cached images.

This memory structure is illustrated for one example in Figure 5.3. Signs and images are stored in separate caches. Signs and their images are also likely to be shared across sign regions. Therefore, the cached set of signs and images is made accessible to all sign regions. The add/delete behaviour of the sign and image caches is the same as for the sign region cache described in (4.5.2). Each cache is empty when a new session begins. While a cache is not full, new entities are added to the cache as necessary. When the cache fills completely, addition of a new entity necessitates the deletion of the entity in the lowest cache position, such that the total size of the cache does not change.

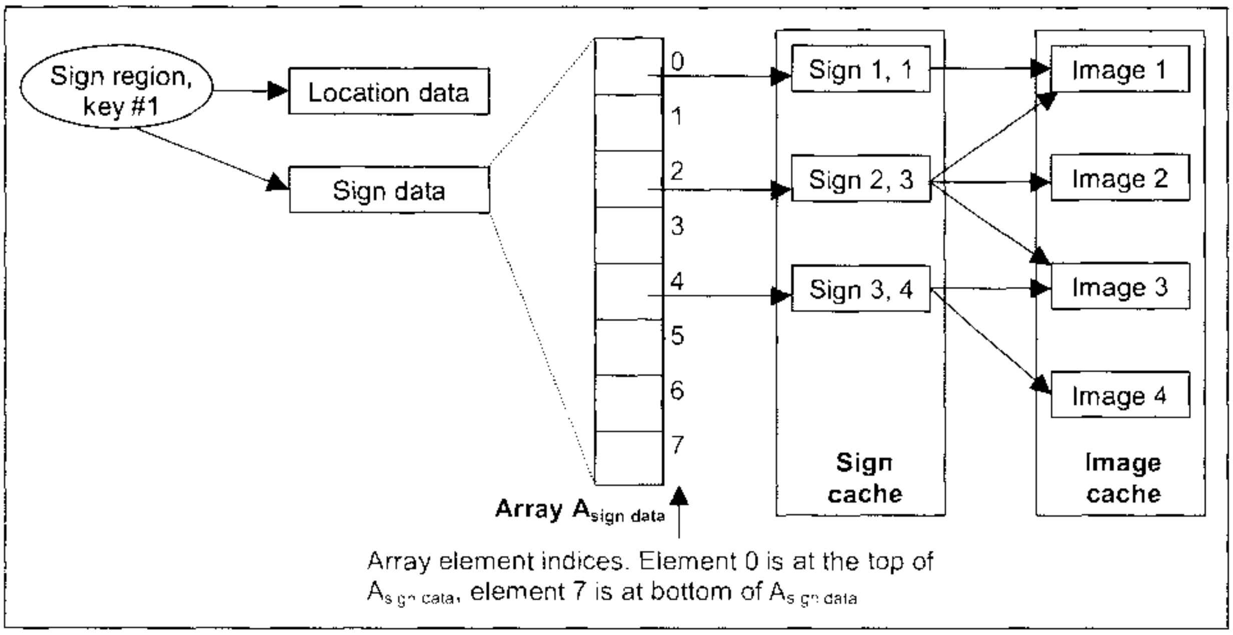


Figure 5.3: Example of sign and image cache memory structure.
The sign data of one sign region is stored in an array of pointers, $A_{sign\ data}$, to its associated signs stored in the sign cache. The index of each element in $A_{sign\ data}$ corresponds to the requested on-screen position of each sign. Here, three signs are pointed to by $A_{sign\ data}$ with requested positions of 0, 2 and 4 respectively. Each sign in the sign cache points to zero or more images in the image cache. Here, images 1 and 3 are displayed by two signs (shared), whereas images 2 and 4 are displayed on one sign only.

5.2.3) Cache Algorithm

In (4.5.2), a FIFO caching algorithm was chosen to decide which sign region data should be removed from the cache as new sign regions are added. This is not a suitable algorithm for signs and images - if a given sign or image is displayed many times in quick succession, it is likely that the sign or image will continue to be displayed frequently in future. For example, some signs are naturally more likely to be encountered by a driver more frequently during a journey, such as lane direction signs, no-parking signs and give-way signs. Other signs may be associated with only one sign region, such as a sign directing road users to a specific destination. A suitable cache algorithm should accommodate this by moving signs/images that occur frequently higher in the cache, such that they are less likely than other entities to be removed from the cache later when the cache fills up and new entities are added. Those signs/images that occur infrequently (or perhaps only once) do not get moved higher in the cache, and therefore are more likely to be removed from the cache as new entities are added.

An optimal cache entity replacement algorithm is simply stated as "remove the entity (sign/image) that will not be accessed for the longest period of time". This result is generalised from Silberschatz and Galvin (1994, p319), who quote the optimal algorithm in the context of replacing virtual memory pages, a topic of computer operating systems. Silberschatz and Galvin (1994) note that the optimal replacement algorithm is difficult to implement because it requires future knowledge of the occurrence of entities that will be stored in the cache. However, they describe approximations to the optimal algorithm that are generalised here to the case of virtual signs and their images. As in (4.5.2), the lifetime scope of a cache will be defined as one *session*, which is the length of time between power-downs when a VRS system is in use. The following cache replacement algorithms all identify the cache entity that is considered least likely to be accessed again this session; all algorithms use the past history of a cache entity as an indication of what is likely to happen in the near future.

1) Least frequently used (LFU) algorithm (Silberschatz and Galvin, 1994, p325).
This replaces the sign/image that has been displayed the fewest number of times this session. One drawback is that signs or images are not aged as time passes. This means an entity that is displayed very frequently early in a session and then not displayed again remains high in the cache. Also, consider what happens if the cache happens to fill up with entities which have all been displayed more than once in a given session. If an entity is then added to the cache that has not yet been displayed this session, it will be immediately removed the next time an entity is due to be added. Because the cache never shrinks during a session, no such new entities will survive longer than the time between successive additions.

2) Least recently used (LRU) algorithm (Silberschatz and Galvin, 1994, p319).
This replaces the sign/image whose elapsed time since last being displayed is greatest. This ages cache entities, but it does not consider the display history (such as the number of times displayed) prior to when the sign/image was last displayed.

3) Highest mean time between displays algorithm
This replaces the sign/image whose average time between display events is greatest.

Here, option (3) is chosen. The average time between display events is considered to be a better summary statistic (in terms of predicting the future) than either the number of times it has been displayed in the past, or the elapsed time since it was last displayed.

During its lifetime in the cache, each entity remembers:

1) The time when it was first added to the cache this session, t_{first}.
2) The number of times it has been displayed this session, n_{disp}.
3) Its mean time between display events, t_{cache} (if $n_{disp} > 1$).

One complication is dealing with entities whose data has been cached but who have not yet been displayed ($n_{disp}=0$), or have been displayed only once ($n_{disp}=1$). In these cases, the average time between display events cannot be calculated or even estimated. In the former case, the entity must not be displaced by another entity for this cache update iteration because it is yet to be displayed (it would not have been loaded into the cache if it was not destined to be displayed at least once). Clearly, an entity that has been displayed only once this session should reside lower in the cache (more likely to be replaced soon) than entities which have been displayed more than once. To indicate this, a flag f_{cache} is set for each entity in the cache to augment t_{cache}. The position of an entity in the cache is then a function of f_{cache} and t_{cache}, as shown in Table 5.1.

Value of flag, f_{cache}	Value of t_{cache}
1	> 0 Value indicates that entity was first displayed some time ago and has only been displayed once this session. Value represents time since entity's data was first stored in cache.
1	= 0 Value indicates that entity has just been added for the first time this session and has not been displayed yet. This state is transitionary and brief, as the fact that the entity has been cached means that a successive display event is imminent.
0	> 0 Value indicates that entity has been displayed at least twice this session. Value represents mean time between display events.

Table 5.1: Possible combinations of values f_{cache} and t_{cache} for a cached entity (sign/image). Combinations are listed in increasing cache position (decreasing likelihood of being replaced any time soon).

According to Table 5.1, an entity that has not yet been displayed for the current session (and is about to be displayed for the first time) is placed very low in the cache when added, with $f_{cache} = 1$ and $t_{cache} = 0$. As time progresses and the entity is not displayed again, f_{cache} remains equal to one and its value of t_{cache} increases, pushing it towards the lowest end of the cache. If the entity is displayed again during the current session, its value of f_{cache} changes to 0 and t_{cache} is recalculated, pushing the entity higher up in the cache. That is, the display event reduces the chance of the entity being removed from the cache any time soon, as it should.

Because a sign or image may be removed from the cache and loaded back into the cache at a later time, the display history of each entity must be stored permanently outside the cache during each session. A sensible decision is to record this information (t_{first} and n_{disp}) in the record corresponding to the cached sign or image in the compiled world database. At the start of a new session, the value of t_{first} is cleared and n_{disp} is set to zero for all records. When an entity is first added to the cache for a given session, the current time is stored as t_{first} in its database record. When the entity is removed from the cache, the number of times it has been displayed this session is copied back out to the entity's database record. On subsequent retrievals from the database, the stored parameters t_{first} and n_{disp} are used to calculate f_{cache} and t_{cache} for each new cache entity, according to Table 5.1. If f_{cache} is zero, t_{cache} is calculated according to Eq. 5.1:

$$t_{cache} = \frac{(t - t_{first})}{(n_{disp} - 1)}$$

Eq. 5.1: Calculation of mean time between display events. t is current time.

For a very large compiled world database, it may take too long to clear the value of t_{first} and set n_{disp} to zero for all records at the start of each session. A more scalable solution is to note the time that a new session begins, $t_{session}$. When an entity is added to the cache, $t_{session}$ is compared with t_{first} for the database record corresponding to this entity. If t_{first} is older than $t_{session}$, the value of t_{first} is set equal to the current time, t, and n_{disp} is set to zero for this record.

There are two further complications associated with the sign and image caches. Sign data must not be removed from the cache while that sign is being displayed. Similarly, an image must not be removed from the cache while it is being displayed on any currently visible signs. Failure to observe these rules would result in an attempt being made to display a sign or image for which there is no data currently loaded, meaning that nothing would be rendered. It makes sense to impose a more restrictive rule for sign images: a sign image must not be removed from the cache while it is associated with any sign currently cached. This ensures that the lifetime of a given image in the cache is at least as long as the longest lifetime of all signs in the cache that display this image. This avoids the possibility of having to reload images for a sign that is currently cached.

5.2.4) An Example of Memory Caching

Let us assume for the purpose of an example that a specific VRS hardware implementation has enough onboard cache memory to store data for only four signs and four images at one time (in practice, the memory capacity will be much greater). This example involves five signs and starts at time t = 1000. At this time, three of the five signs have already been displayed during the current session – their statistics at this time are given in Box 5.1. In each figure presented on p. 5-15 and 5-16, the set of signs and images cached in each case are shown at their correct cache position. The conventions used to indicate whether each sign/image is displayed at a given time are shown in Figures 5.4.a and 5.4.b. For simplicity, the statistics of the images displayed on each sign are not considered in this example.

Sign	Value of t_{first}	Current value of n_{disp}
A	400	3
B	500	2
C	600	1

Box 5.1: Statistics of three signs A, B and C that have already been displayed this session, prior to the cache state shown in Figure 5.5.a. Signs D and E have not been displayed yet this session.

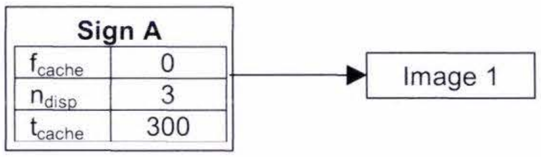


Figure 5.4.a: Conventions indicating that a sign/image is not currently displayed

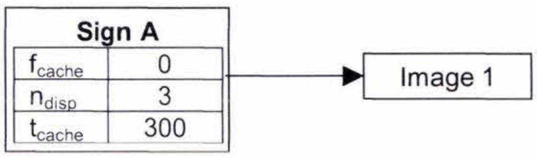


Figure 5.4.b: Conventions indicating that a sign/image is currently displayed (grayed)

Figure 5.5.a shows the state of the cache at time t = 1000. Three signs A, B and C are initially cached, as are the three images displayed on them. Sign A displays image 1, sign B displays image 2, and sign C displays images 1 and 3. Immediately prior to time t = 1100, signs A and B are currently visible. A new sign D which contains one new image 4 is about to be added to the cache, and sign C is to be re-displayed. Because neither the sign or image cache are full, deletion from either cache is not necessary as the new sign and image are added. Because sign C and image 3 are already cached, their data is not loaded again. The memory state at time t = 1100 is shown in Figure 5.5.b. The value of n_{disp} for sign C has been incremented by one and its value of f_{cache} has been changed from one to zero. As a result of the re-display event, sign C now resides higher in the cache than sign B, and the position of image 3 has been moved higher in the image cache.

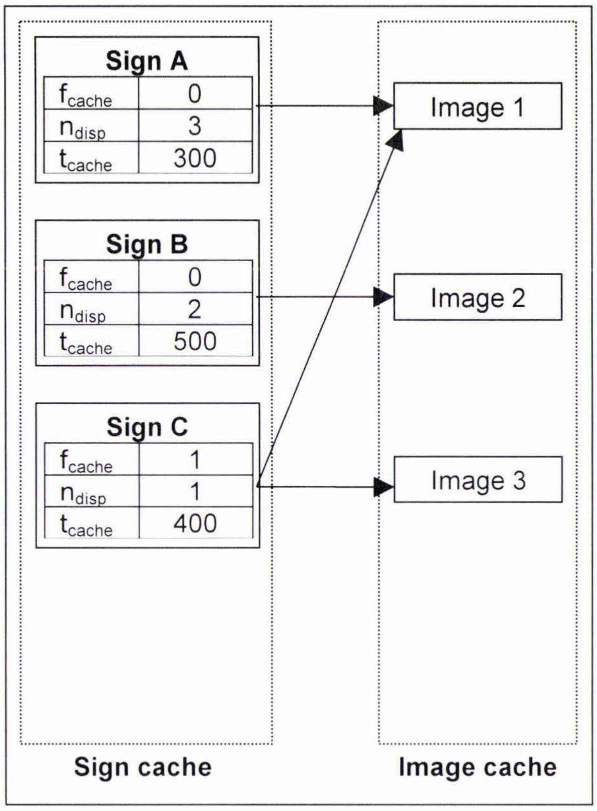


Figure 5.5.a: Cache state at time t = 1000. Signs A and B are currently visible.

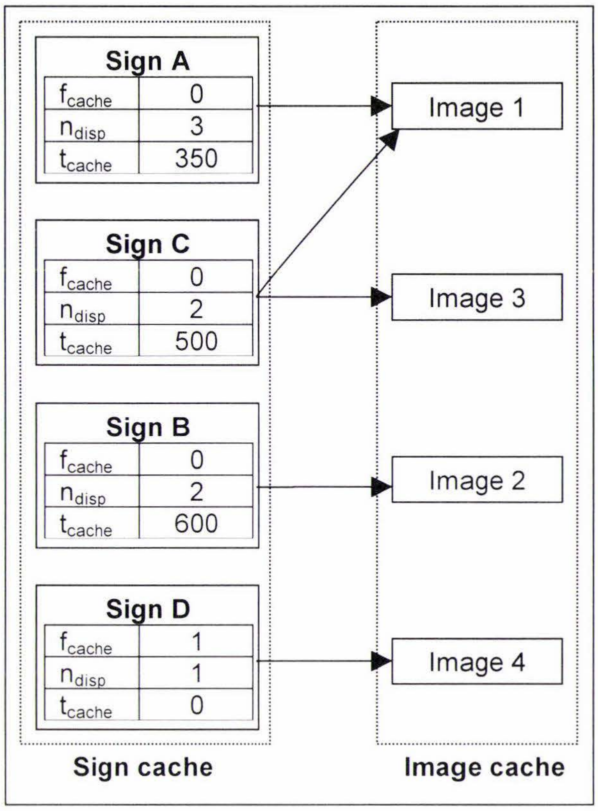
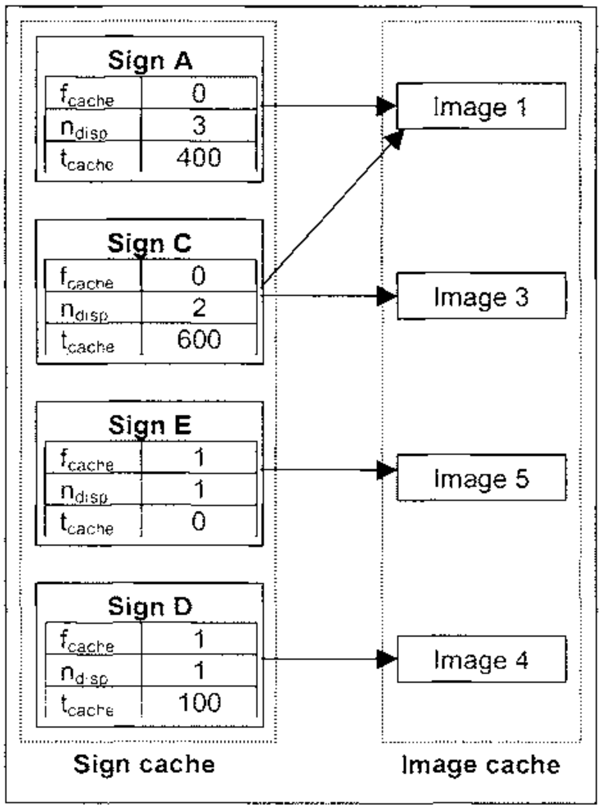


Figure 5.5.b: Cache state at time t = 1100. Sign D has been added. Signs C and D are currently visible.

Immediately prior to time t = 1200, signs C and D are currently visible. A new sign E which contains one new image 5 is about to be added to the cache. Because both the sign and image cache are full, one existing sign and image in the caches must first be deleted before new sign E and its image can be added. Sign D is at the lowest position in the sign cache, but it cannot be deleted because it is currently displayed. Similarly, image 4 is at the lowest position in the cache, but it cannot be deleted because it is associated (and currently displayed) on cached sign D. Therefore, sign B is removed from the cache followed by its image 2 to make way for new sign E and image 5.

The memory state at time t = 1200 is shown in Figure 5.5.c. Although signs D and E both have the same values of f_{cache} and n_{disp}, sign E is at a higher cache position because it has been added to the cache more recently. Immediately prior to time t = 1300, signs C and E are currently visible. Sign B and its image 2 are about to be re-added to the cache, having been previously displaced. Sign D is at the lowest position in the sign cache, and this can be deleted because it is not currently visible. In the image cache, image 4 is at the lowest position, and this can also be deleted immediately after the only sign that contains it (sign D) is deleted from the cache. The memory state at time t = 1300 is shown in Figure 5.5.d. At this time, only sign B is currently visible. The value of n_{disp} for sign B has been incremented by one. As a result, it now has the lowest value of t_{cache} for all cached signs and thus resides at the highest position in the cache.



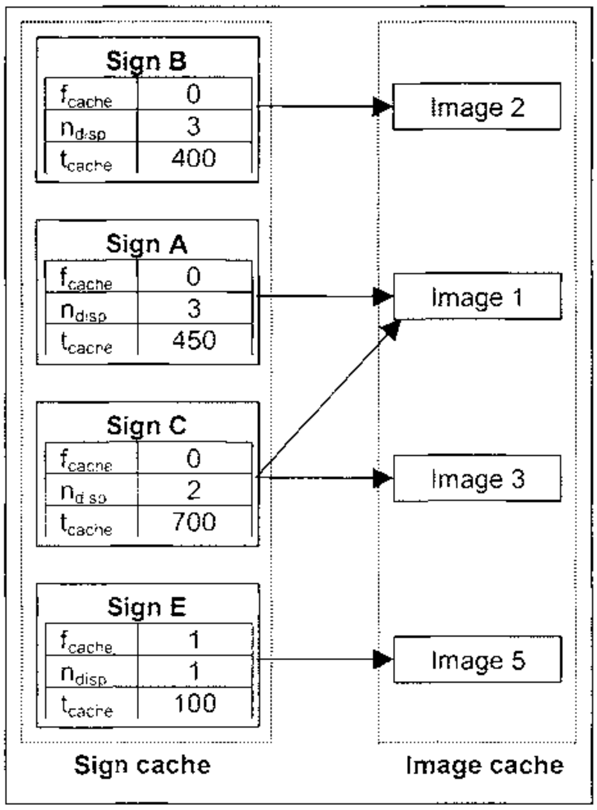


Figure 5.5.c: Cache state at time t = 1200. Sign E has been added, displacing sign B and its image 2 from the caches. Signs C and E are are currently visible.

Figure 5.5.d: Cache state at time t – 1300. Sign B has been re-added, displacing sign D and its image 4 from the caches. Sign B is currently visible.

5.3) Prototype Sign Design

<u>5.3.1) Sign Face Design</u>

A specific sign design was chosen for the purpose of prototype testing. The design was chosen to maximise the readability of signs displayed on a computer monitor in the lab. Each sign adheres to a consistent format and set of conventions, with support for up to four graphical images and four lines of text. This allows a conventional four-panel sign, as is commonly used on interstate highways (Smailus, Bullock and Besly, 1996), to be reproduced as one virtual sign. The backboard of each sign is shape-coded and colour-coded to permit drivers to instantly deduce the type of information conveyed by a sign (for example, road names versus real-time traffic information) when it is first displayed, long before its message may be readable.

Pre-designed clipart was found to be an ideal source of sign images. An enormous range of clipart was available as part of the CorelDraw desktop publishing software (Corel Corporation, 1993; Corel Corporation, 1997). Clipart images are typically highly contrasting, vibrant and not overly detailed. One alternative to clipart is symbolic icons, as are often displayed on conventional road signs. Such icons are simpler, less detailed and even more contrasting than clipart (often black on one other background colour). In this case, only generic icons were available, and clipart was judged to convey a better graphical representation.

It was noted in (5.1.1) that many drivers prefer the use of arrows for indicating direction on signs rather than text. In this case, there was insufficient space on each virtual sign for an arrow that is large enough to be adequately recognisable from some distance. It is well known that the human visual system is more sensitive to some colours than others and finds horizontal and vertical lines visually appealing (Russ, 1995). Thus, a thin, bright red, vertical band was placed on the left-hand or right-hand edge of each sign to indicate left or right direction respectively. Consistent use of this band allows the referring direction of a sign to be instantly assimilated when a sign first appears. Of course, this relies on a driver being familiar with the use of the vertical band to indicate which side of the road a feature is located.

Here, a normalised coordinate system (x, y) of each sign face is defined. The origin and scale are predefined for several pre-defined, standard shapes. By convention, positive x is right, positive y is up, and the origin lies at the centre of a rectangle, $R_{enclosing}$, that minimally encloses the sign after any rotation. Signs may be rotated around this origin. Rectangle $R_{enclosing}$ defines the range of positions that can be addressed on the sign face:

- For all unrotated standard shapes that can be minimally enclosed by a square, a 1 x 1 coordinate system applies. That is, $R_{enclosing}$ is a 1 x 1 square that completely and minimally encloses the shape, giving an addressable coordinate range of [x, y] = [-0.5:0.5, -0.5:0.5]. Figure 5.6.a shows one example.
- For all other unrotated standard shapes, $R_{enclosing}$ is defined on a per-shape basis (see [E.2.8]). Figure 5.6.b shows one example.
- For all rotated standard shapes, $R_{enclosing}$ is calculated on a per-shape basis after rotation. Figure 5.6.c shows one example.

This design allows additional standard shapes to be easily added in future. Because the backboard of virtually all road signs conforms to one of a small set of shapes, user-defined shapes are not supported. Images and text can be placed anywhere within the addressable region. It should also be noted that because virtual signs do not require supporting legs, they are not included in the sign's definition.

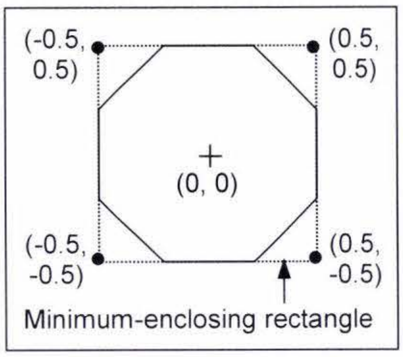


Figure 5.6.a: Coordinate system of unrotated standard octagon. $R_{enclosing}$ is a 1 x 1 square.

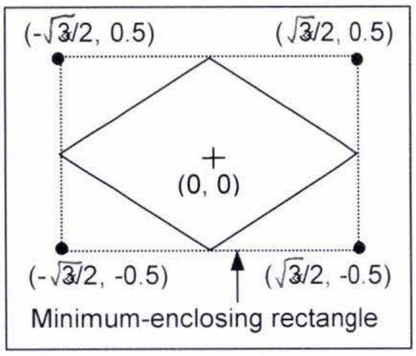


Figure 5.6.b: Coordinate system of unrotated standard rhombus. $R_{enclosing}$ is a ($\sqrt{3}$ x 1) rectangle.

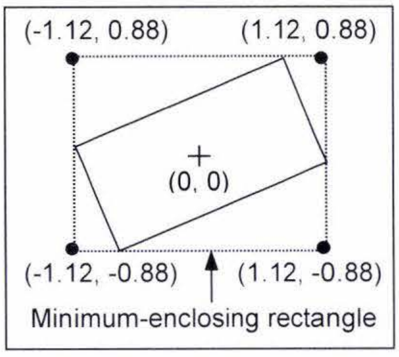


Figure 5.6.c: Coordinate system of standard 2 x 1 rectangle rotated 25° anti-clockwise from horizontal. $R_{enclosing}$ is a (2.24 x 1.75) rectangle.

One example of a virtual sign that indicates a feature on the left side of the road, and which adheres to the design discussed above, is shown in Figure 5.7. This sign was in fact created during prototype testing, which is discussed in Chapter 7.

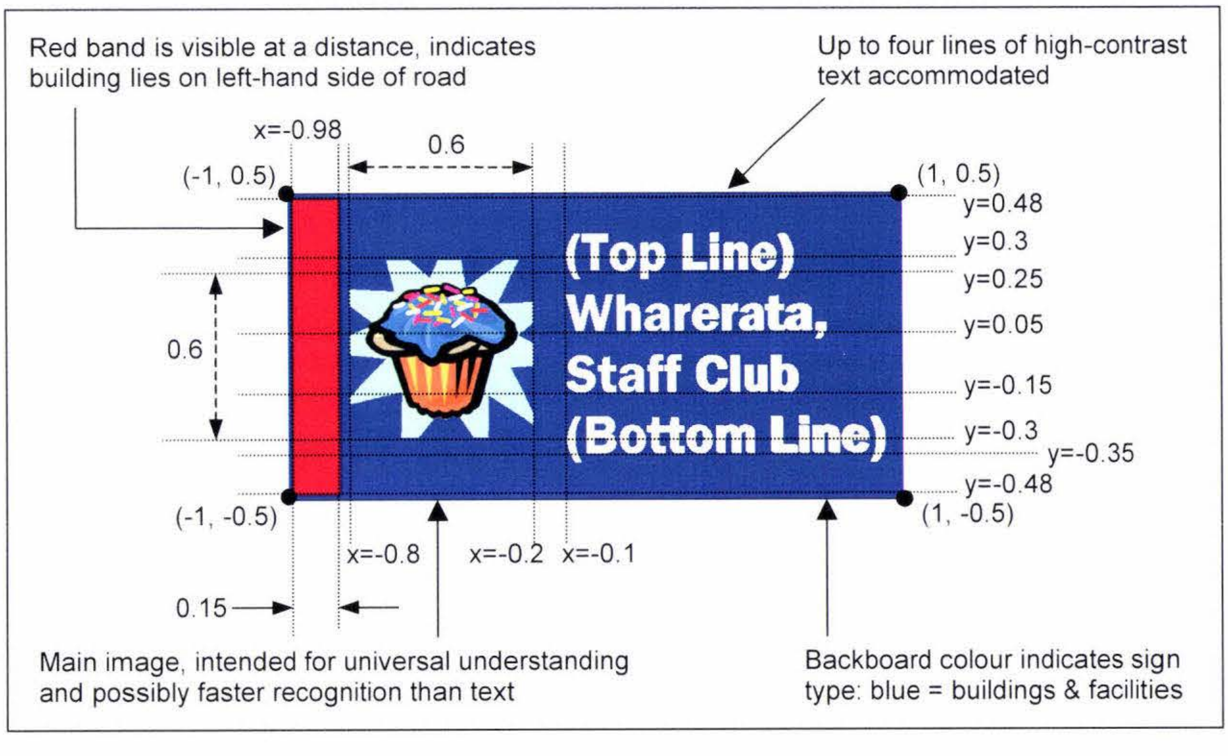

Red band is visible at a distance, indicates building lies on left-hand side of road
Up to four lines of high-contrast text accommodated
x=-0.98
0.6
(-1, 0.5)
(1, 0.5)
y=0.48
y=0.3
y=0.25
y=0.05
y=-0.15
y=-0.3
y=-0.35
y=-0.48
(-1, -0.5)
(1, -0.5)
0.6
(Top Line)
Wharerata,
Staff Club
(Bottom Line)
x=-0.8
x=-0.2
x=-0.1
0.15
Main image, intended for universal understanding and possibly faster recognition than text
Backboard colour indicates sign type: blue = buildings & facilities

Figure 5.7: Example of a test virtual road sign, showing main visual elements and coordinate system.

5.3.2) Memory Caching Strategy

OpenGL provides two data structures that are used in the VRS prototype to cache compiled signs and sign images. The first structure is known as a *display list*, which is intentionally designed to accelerate rendering of static, frequently drawn graphics. A display list stores a compiled version of a sign's visual appearance in memory in a form that permits faster rendering than if the individual elements are redrawn sequentially every time (Warhol, 2001). By using display lists, a sign's geometry needs only to be defined once; once compiled, a display list can be executed multiple times to render the sign. A total of $n_{max\ signs}$ display lists are created during prototype initialisation to represent up to $n_{max\ signs}$ compiled signs that can be cached at one time. Display lists are also used here to store images of all font characters that are used to render text on each sign. One display list is used to render each character of a Truetype font. Each display list is identified by a key number.

Each sign image is stored on secondary storage as a bitmap file that accompanies the compiled world database. Each sign image is represented in memory as a second type of OpenGL structure known as a *texture* (Warhol, 2001). Like display lists, textures are stored in local memory (or possibly video memory if it is available on the local display hardware) in 32-bit RGBA format. The four RGBA fields define the red, green, blue and alpha component of each pixel respectively, where alpha describes a pixel's degree of transparency. Each texture is identified by a key number known as a *texture name*. A total of $n_{max\ images}$ texture names are created during prototype initialisation to represent up to $n_{max\ images}$ images that can be cached at one time.

The caching strategy described in (5.2.3) is implemented in the VRS prototype using the same data structures as described in (4.6.2). In this case, a binary tree and a separate node list is utilised for both the sign cache and image cache. The data structures of both caches are shown in Figure 5.8. For the sign cache, each element in the cache array, A_{sign}, stores the number of one display list that will store one compiled sign. Each node in the binary tree, T_{sign}, is identified by the major and minor key values of the compiled sign it represents, and points to one element in array A_{sign}. Each tree node stores the height and width of the sign's minimum enclosing rectangle (as described in [5.3.1]) plus up to four pointers to nodes in the image cache binary tree, T_{image}, that represent the images displayed on this sign (see Figure 5.8). For the image cache, each element in the cache array, A_{image}, stores the texture name of one texture that will store one sign image. Each node in the binary tree, T_{image}, is identified by the key number of the image it represents in the compiled world database, and points to one

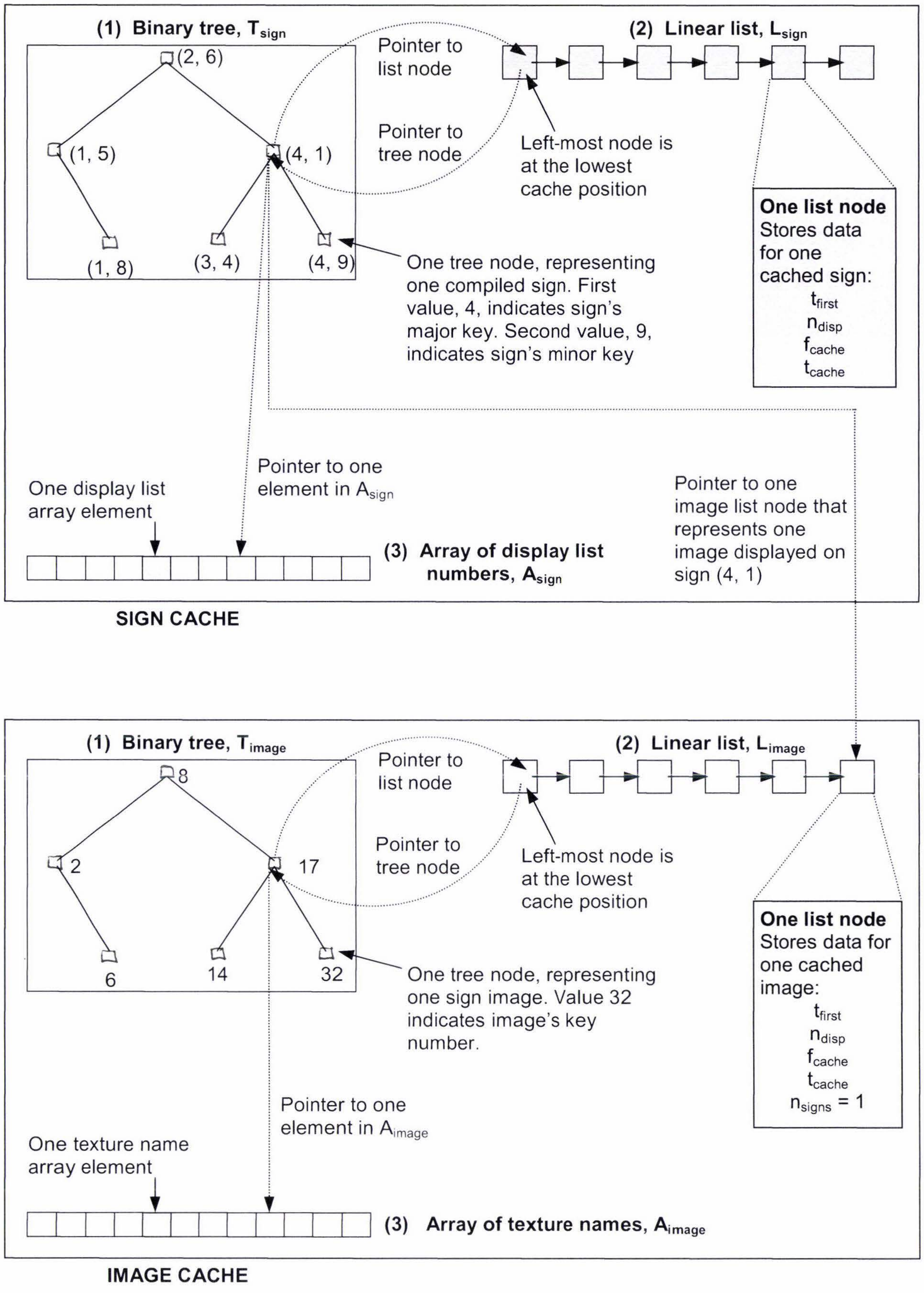

(1) Binary tree, T_{sign}
(2, 6)
(1, 5)
(4, 1)
(1, 8)
(3, 4)
(4, 9)
Pointer to list node
Pointer to tree node
(2) Linear list, L_{sign}
Left-most node is at the lowest cache position
One tree node, representing one compiled sign. First value, 4, indicates sign's major key. Second value, 9, indicates sign's minor key
One list node
Stores data for one cached sign:
t_{first}
n_{disp}
f_{cache}
t_{cache}
Pointer to one element in A_{sign}
One display list array element
Pointer to one image list node that represents one image displayed on sign (4, 1)
(3) Array of display list numbers, A_{sign}
SIGN CACHE
(1) Binary tree, T_{image}
8
2
17
6
14
32
Pointer to list node
Pointer to tree node
(2) Linear list, L_{image}
Left-most node is at the lowest cache position
One tree node, representing one sign image. Value 32 indicates image's key number.
One list node
Stores data for one cached image:
t_{first}
n_{disp}
f_{cache}
t_{cache}
$n_{signs} = 1$
Pointer to one element in A_{image}
One texture name array element
(3) Array of texture names, A_{image}
IMAGE CACHE

Figure 5.8: Data structures of sign and image cache. For clarity, only the set of pointers for one binary tree node is shown in each cache (nodes (4, 1) and 17 respectively)

element in array A_{image}. As in (4.6.2), each node in binary trees T_{sign} and T_{image} corresponds to a node in list L_{sign} and L_{image} respectively. Here, each node of both lists stores the time the sign/image was first loaded into the cache (t_{first}), the number of times the sign/image has been displayed this session (n_{disp}), plus the current values of f_{cache} and t_{cache}. In addition, each node in list L_{image} stores a value n_{signs}, which is the number of signs currently stored in the sign cache that display the image represented by this list node.

In the VRS prototype, signs and images are cached as part of the sign compilation process. For a sign not presently cached, the sign is not compiled until immediately before it is to be assigned a screen location where it will be rendered on-screen. If the sign is presently cached, sign compilation has already been performed and is not required again. Screen locations are assigned when a new set of signs is to be displayed, which occurs when a new set of active sign regions is to be processed. Determination of the current set of active sign regions occurs in response to a change in vehicle position or velocity, as described in (4.5.1). Therefore, signs and images are cached in response to a new vehicle position or velocity, but the process of caching a specific sign or image does not occur until it is absolutely necessary (just before the sign/image is to be displayed on-screen). Let the total set of signs/images that are processed in response to a new vehicle position or velocity be termed one *batch*. The algorithm listed in Box 5.2 describes how the cache is processed and updated when new sign or image data is to be loaded from the compiled world database during one batch.

```
For each sign SR/image M that is about to be displayed:

1)  Rebalance the binary tree $T_{sign}/T_{image}$ if the height of the left and right subtrees of any tree node differs by >= a certain number, k. Periodic rebalancing of the tree is required to prevent the tree degenerating towards a linear list, which cannot be searched nearly as efficiently as a tree.

2)  Search the binary tree for SR/M:

    a)  If found, SR/M is already stored in the cache. Thus, for a sign, its data does not need to be retrieved from the compiled world database and sign compilation is not required. For an image, its data does not need to be retrieved from secondary storage.

    b)  If not found:
        i)  If cache is not full:
            1)  Find where the new node should be added to the binary tree to preserve the ordering, and add the new node.

            2)  For a sign:
                a)  Assign the next available element in the cache array of display list key numbers to this node.
                b)  Compile the sign into the assigned display list.
                c)  Load values of $t_{first}$ and $n_{disp}$ from the database for this sign record, and calculate corresponding values for $f_{cache}$ and $t_{cache}$.
                d)  Check whether all images for this sign are currently loaded in the image cache, and if not, cache them. That is, execute this algorithm for each sign image. Set sign's pointers to each image node in list $L_{image}$.
                e)  Increment the value of $n_{signs}$ for each image on this sign by one.
                f)  Add a new node to the right-most end of list $L_{sign}$.

            3)  For an image:
                a)  Assign the next available element in the cache array of texture names to this node.
                b)  Load the image and store it under the assigned texture name.
                c)  Load values of $t_{first}$ and $n_{disp}$ from the database for this image record, and calculate corresponding values for $f_{cache}$ and $t_{cache}$.
                d)  Add a new node to the right-most end of list $L_{image}$.

        ii) If cache is full:

            1)  If list $L_{sign}/L_{image}$ has not been re-ordered yet for this batch:
                a)  Recalculate $f_{cache}$ and $t_{cache}$ for all nodes in the list.
                b)  Re-construct the list in ascending order of cache position.

            2)  Find the left-most node in list $L_{sign}/L_{image}$ that is eligible for deletion. Delete this node and the corresponding node in binary tree $T_{sign}/T_{image}$. For a deleted sign, decrement the value of $n_{signs}$ for each image on this sign by one. Remember the element assigned to this node in the cache array $A_{sign}/A_{image}$.

            3)  Add a new node to binary tree $T_{sign}/T_{image}$ and list $L_{sign}/L_{image}$ according to steps (2bi, 1-3). For a sign, at step (2bi, 2a), re-use the display list at the array element previously assigned to the deleted node to compile the sign. For an image, at step (2bi, 3a), assign the texture name at the array element previously assigned to the deleted node to this new node.
```

Box 5.2: Sign/image cache algorithm

In step (2bii), recall the rules that must be satisfied for a node to be deleted: a sign must not be removed from the cache while that sign is being displayed, and an image must not be removed from the cache while a sign that displays it exists in the cache. If this latter rule was not enforced, the texture name of an image, M, displayed on a currently-cached sign would be assigned differently when that image was re-added to the cache after being removed, compared to the texture name originally assigned to M when the sign was compiled. This would require the entire sign to be recompiled, defeating the purpose of the sign cache. To quickly determine whether a given image, M, satisfies this rule, the value of n_{signs} stored for M's node in list L_{image} is checked. If this value is zero, then no signs that display M are currently stored in the cache, so M can be removed from the cache. If n_{signs} is non-zero, M is not eligible for deletion and so another image must be found to delete.

The algorithm listed in Box 5.2 is similar to that listed in Box 4.4 of (4.6.2). The main difference here is the addition of the re-ordering step in (2bii, 1) which is necessary to update all values of f_{cache} and t_{cache} in list L_{sign}/L_{image} immediately prior to when the first sign/ image is to be deleted from the cache for this batch. Due to the fact that each batch is processed in its entirety before the display subsystem does its work (as discussed in Chapter 6), and that the re-ordering step can occur midway through processing of a given batch, any nodes found in the cache during batch processing must not be deleted while the remainder of the batch is processed. We could try to re-order the cache lists every time the next entity is processed in a batch, in an attempt to ensure that any such node never reaches the lowest position in the cache (and thus face imminent deletion). A more efficient and simpler technique is to perform the re-ordering step no more than once per batch (this is sufficient, as the time taken to process each batch is considered to be negligible with respect to the age of a session), and simply flag ineligible nodes for deletion as they are found in the cache.

For a given batch, one of three possible scenarios applies for the sign and image cache:

1) The cache is not full before processing the batch, and remains not full after all relevant nodes have been added to the binary tree and list. Processing is simple: node deletion is not necessary, and thus re-ordering of cache positions in list L_{sign}/L_{image} is not necessary.

2) Cache is not full before processing the batch, but fills completely during processing. Re-ordering of cache positions occurs when the cache becomes full, and node deletion is necessary for every new node added thereafter. Any nodes added for this batch must not be deleted for the remainder of this batch as new nodes are added.

3) Cache is full before processing the batch, and remains full after processing. Node deletion is immediately necessary in order to cache new signs/images, so re-ordering of cache positions occurs immediately. There is no need to check that nodes added during this batch are not deleted this batch, as this is guaranteed not to happen providing the cache is larger than a minimum size, k. For the sign cache, k is at least equal to the maximum number of signs that can be visible at any one time ("at least" because more signs may be loaded into the cache than can be displayed concurrently). For the image cache, k is equal to the maximum number of images that can be stored on cached signs at any given time. These are theoretical values of k for the two caches: in practice, k must be greater than those given above due to practical overheads.

For (2) and (3), we would like to know the full set of signs and images that will be processed for this batch before the re-ordering step occurs. This avoids the possibility of a sign/image very low in the cache being deleted before it is processed, only to be re-added again for this same batch. For the VRS prototype, it is difficult to determine the full set of signs and images that will be processed in advance, before batch processing begins. However, this is of minor consequence because the probability of deleting a sign/image that is yet to be processed for the current batch is very low.

5.4) Prototype Database Design

5.4.1) Database Structure

For the VRS prototype, three Microsoft Access 97 databases are utilised:

1) Feature database, DB1.

This stores the spatial coordinates of all features to be represented by signs, plus information that allows sign regions to be computer-generated to match the features. It is only accessed during creation of databases DB2 and DB3, as discussed in Chapter 7, and serves no purpose during prototype execution.

2) Source world database, DB2.

This is one of the two databases described in (2.3.3) and (4.4.1). It defines all world entities as shown in Figure 4.4 of (4.3): all sign regions, the visual layout of all signs, sign elements such as text and images, mappings of which signs appear in which regions, as well as parameters describing the dimensions of the virtual world and conversions between coordinate systems.

3) <u>Compiled world database, DB3.</u>

This is the second of two databases described in (2.3.3) and (4.4.1). It stores the logical quadtree structure created during world compilation. In a real VRS system, this database would also store the localised set of sign region, sign and image definitions in that part of a large virtual world that is of interest to an end-user. Here, for the purpose of prototype testing, it is unnecessary to copy this latter data out of the source world database into the compiled world database. Thus, during prototype execution, both databases DB2 and DB3 are continually accessed: DB3 provides the list of sign regions per quadtree node quadrant, and the actual sign region and sign definitions are then retrieved from DB2.

The process of database creation required before the prototype can be executed is illustrated in Figure 5.9.

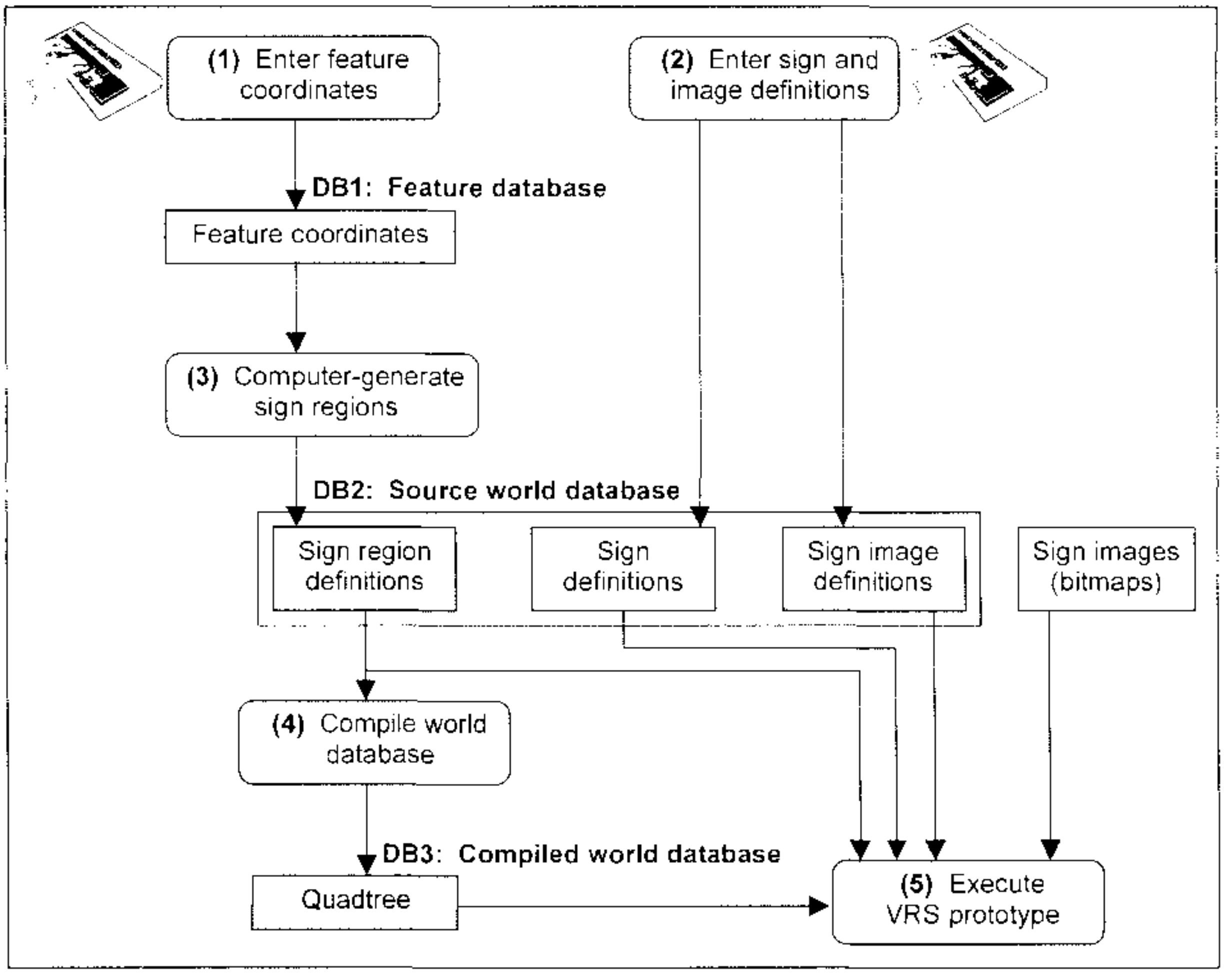

(1) Enter feature coordinates
(2) Enter sign and image definitions
DB1: Feature database
Feature coordinates
(3) Computer-generate sign regions
DB2: Source world database
Sign region definitions
Sign definitions
Sign image definitions
Sign images (bitmaps)
(4) Compile world database
DB3: Compiled world database
Quadtree
(5) Execute VRS prototype

Figure 5.9: Database creation process for VRS prototype

The database design of the source world database, DB2, can be well summarised by a diagram of relationships between the implemented database tables. Figure 5.10 shows the table definitions and relationships between ten of the thirteen tables in DB2. Database normalisation is a technique that simplifies entities and removes redundancy to improve data storage efficiency and reduce the possibility of data invalidation (Whitten *et al.*, 1994). In Figure 5.10, all tables have been normalised for efficiency, and each many-to-many mapping has been decomposed into two one-to-many mappings according to standard practice. Comparison of Figure 5.10 with Figure 4.4 of (4.3) shows that the table mappings of Figure 5.10 correspond exactly with the entity mappings of Figure 4.4. The one exception is sign features, which are stored separately in database DB1.

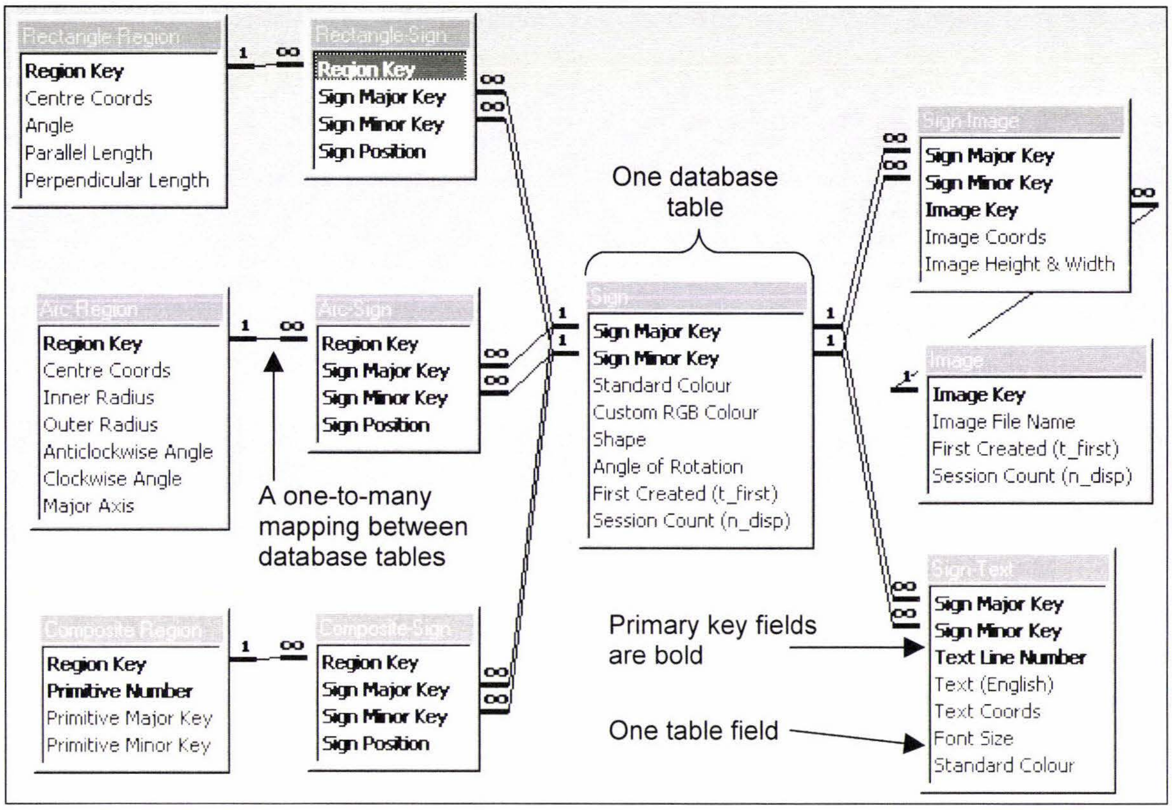


Figure 5.10: Table definitions and relationships between tables, for source world database DB2.

5.4.2) Table Definitions

Table 5.2 gives a brief description of each table in database DB2 shown in Figure 5.10; full details of the table fields in all three prototype databases may be found in Appendix E. Recall that the *primary key* fields of a database table permit unique identification of each record in the table. In Figure 5.10, the primary key fields of each table are highlighted in bold type at the top of each table.

Table Name(s)	Primary Key Field(s)	Table Description
Rectangle Region Arc Region	Region Key	Define all rectangle and arc sign regions respectively. Each region type is stored in its own table.
Composite Region	Region Key Primitive Number	Defines all composite sign regions. Each composite region points to multiple rectangle/arc regions.
Rectangle-Sign, Arc-Sign, Composite-Sign	Region Key Sign Major Key, Sign Minor Key, Sign Position	Defines many-to-many mapping between sign regions and signs; that is, they define which signs appear in which sign regions and in which locations on screen. Multiple signs can appear in a given sign region, and multiple sign regions can contain a given sign. The *Sign Position* field defines the requested on-screen position of a sign, which is described fully in Chapter 6.

Table Name(s)	Primary Key Field(s)	Table Description
Sign	Sign Major Key Sign Minor Key	Defines colour, shape and angle of rotation of all signs. During prototype execution, each sign record also stores the time when the sign was first created in the sign cache, plus the number of times the sign has been displayed for the current session.
Image	Image Key	Defines the filenames of all sign images. During prototype execution, each image record also stores the time when the image was first created in the image cache, plus the number of times the image has been displayed for the current session.
Sign-Image	Sign Major Key, Sign Minor Key, Image Key	Defines many-to-many mapping between signs and sign images; that is, defines which images appear on which signs. Multiple images can appear on a given sign, and multiple signs can display a given image. For each image, table stores (x, y) coordinates of lower left corner, height and width, for up to four images per sign.
Sign-Text	Sign Major Key, Sign Minor Key, Text Line Number	Defines all sign text. For each text string, table stores (x, y) coordinates of lower left corner, font point size, colour, and text in any language.

Table 5.2: Description of tables in source world database DB2.

Table *Sign-Text* supports a user-defined set of non-primary key fields to describe the text displayed on a sign. Each field represents the same text in a different language. This allows multiple languages to be easily implemented for a specific database: the user then needs only to specify their desired user language at application run-time. At present, only one text field for the English language is defined (*Text (English)* in table *Sign-Text* in Figure 5.10).

5.5) Summary

Good design and presentation style of virtual signs is crucial if they are to be effective as a means of disseminating information to drivers. The same design goals of conventional road signs also apply to virtual road signs, such as good readability, consistency, flexibility, time of presentation and location. Sign compilation retrieves sign information from secondary storage on a just-in-time basis. Like sign regions, virtual signs and their visual elements are cached in local memory to improve the real-time performance of a VRS system. The cache algorithm attempts to keep that data in local memory that is likely to be accessed again soon. The database design of the prototype is simple and flexible. A third database was used by the prototype to store the GPS coordinates of all features in the test world, to accompany the two other databases introduced in (2.3.3). In the next chapter, we investigate how virtual signs are displayed on-screen to present their messages to a driver.

CHAPTER 6
Display Subsystem

The display subsystem is responsible for accepting a list of virtual road signs output by the data subsystem and rendering each sign in a graphical form on the VRS display device. The latter is the in-vehicle visual display that a driver observes in order to view and recognise all virtual road signs.

A variety of display devices may be capable of rendering virtual road signs. For this reason, it is unwise to assume the capabilities and limitations of a specific target display device. Even a specific implementation will need to account for improvements in the chosen display technology. Therefore, the design of the display subsystem in a generic VRS system must be made flexible. Like the data subsystem discussed in previous chapters, one would like to consider the display subsystem in a general sense. Here, it is easier to describe the display subsystem adopted for the VRS prototype and show how this can be generalised, rather than approach the topic from a general point of view.

Prototype Display Subsystem

6.1) Overview

As described in (1.8), a HUD is the desired projection device for sign rendering. It offers the least intrusion and inconvenience for the driver: as the HUD would be built into a vehicle's interior construction, the driver would not need to wear any new device or radically adjust their driving behaviour. Unfortunately, a HUD was simply not available to use as a prototype display device at a reasonable price. With current display technology, the most suitable and affordable compromise was deemed to be an in-vehicle flat-panel liquid-crystal display (LCD). For example, one option was to install a laptop into the test vehicle which would execute the prototype and simultaneously render virtual signs in real-time on its built-in LCD. Such a display is essentially the same device commonly used in in-vehicle HMIs, and thus it would not mitigate the problems of driver distraction and continual diversion of attention that virtual road signs set out to address. This is an inevitable outcome of this project – suitable projection technology is currently too expensive and relatively scarce, even for research purposes.

For the purpose of this research, it was difficult even to locate a laptop that was sufficiently modern to handle the real-time display requirements of the VRS prototype. Therefore, the decision was eventually made to utilise a standard CRT computer monitor as the primary display device. The bulky size, fragility and high power consumption of a CRT monitor makes it unsuitable to be used as a display device inside a vehicle. To solve this problem, the prototype allowed a log file of real-world data to be captured on a laptop PC, and 'replayed' on another PC later to display the virtual signs corresponding to the logged data. A driver could then see the set of virtual road signs that would have been observed as they travelled around the road, had a suitable in-vehicle display device been found. This is an important facility because it permits the two processes of (1) data collection, and (2) world processing and sign rendering, to be handled as two completely separate tasks. This decoupling yields major advantages for the purposes of research and development:

a) A low-powered laptop satisfies the conditions of small size, battery power source and portability for in-vehicle data collection, while a much more capable (and bulky) PC performs all subsequent processing back at the lab. Similarly, a bulky CRT monitor that is not at all suited to the confines of a vehicle is easily accommodated back at the lab.

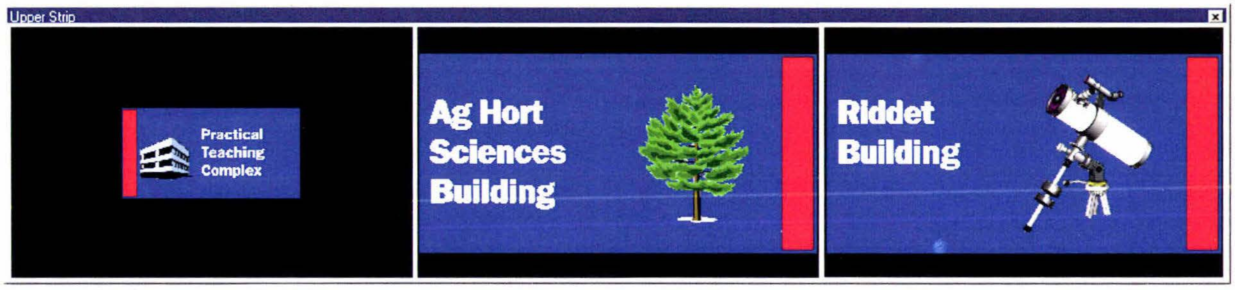
b) Repeatable application demonstration to a large audience is made feasible, because the saved data file can be replayed as often as desired. Without this file, the audience would be limited to a handful of people who could fit inside a vehicle at one time.

6.2) Prototype Graphical User Interface (GUI) Design

The prototype GUI was designed with an emphasis on sign readability. This meant that each sign must be completely unobscured at every point when it is visible, and signs must appear fully opaque rather than transparent or translucent. To permit more than one sign to be displayed simultaneously, the screen is divided into three discrete sections:

6.2.1) Upper Form Strip

A *form strip* is simply any window in which signs may be displayed. The upper form strip is located at the top of the screen. Three equal-sized adjacent regions known as *strip rectangles* span horizontally across the strip, as shown in Figure 6.1. Each strip rectangle is assigned a number, as shown in Figure 6.1, and can display no sign or one sign at any time. By convention, this upper strip displays signs whose road-side position is not critical. For example, speed limit signs would be displayed in this strip because the message conveyed is not dependent on the side of the road on which the sign appears. Therefore, when a new sign is to be displayed in this form strip, it may be displayed in any available strip rectangle.



Strip rectangle #5 **Strip rectangle #6** **Strip rectangle #7**

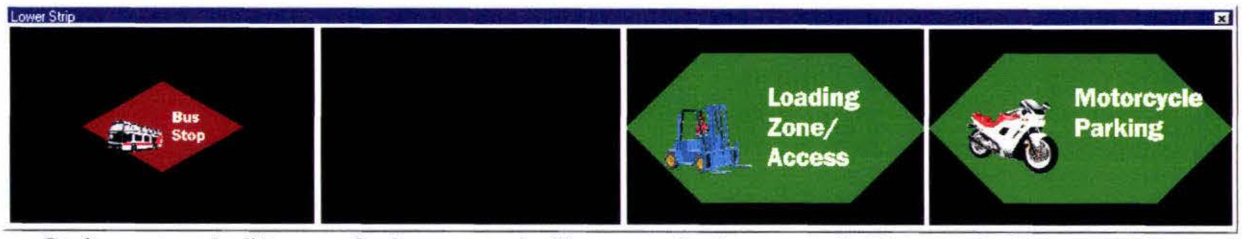
Figure 6.1: Upper form strip, showing strip rectangles. Three signs are allocated: the sign in strip rectangle #5 is some distance from the vehicle, whereas signs in #6 and #7 are at their maximum display size. Strip rectangles #1-4 are contained by the lower form strip (described in [6.2.2]).

In (5.2.2), it was mentioned that each virtual sign nominates an on-screen rendering position. The number assigned to each strip rectangle, as shown in Figure 6.1, corresponds to this requested position. For example, a sign that requests a position of five indicates that it would like to be rendered in the left-most strip rectangle. Signs in the upper form strip are not guaranteed to receive the position they nominate, however, due to the competing requests of other signs that are currently displayed. In addition, a position of zero is reserved to indicate that a sign has no preferred position. The topic of sign allocation to screen locations is discussed later in (6.5).

An alternative idea for sign display in the upper strip is to have every sign always appear first in the left-most strip rectangle available. As each sign displayed in a strip rectangle disappears (as the vehicle passes a virtual sign), signs are smoothly scrolled left to fill the gap. This matches the analogy of reading from left to right with our eyes. An advantage is that a driver only ever needs to scan the far left strip rectangle rather than the entire strip, although they can receive advance warning of more distant signs by reading right if they wish. However, shifting signs may be distracting and reduce the effective time during which a sign is readable, as moving signs cannot be read as easily.

<u>6.2.2) Lower Form Strip</u>

The lower form strip is located at the bottom of the screen. Up to four smaller signs can be displayed at once on this form in four strip rectangles, as shown in Figure 6.2. By convention, this lower strip is reserved to display signs whose road-side position is critical, such as lane arrow markings normally painted on a road – the assigned strip rectangles are then guaranteed to match the lane markings. For example, on a four-lane highway, the far left strip rectangle might mark a left-turning lane, the two central strip rectangles might mark straight-ahead lanes, and the far right strip rectangle might mark a right-turning lane. For signs that indicate structures beside the road (such as buildings), the left two strip rectangles are reserved for left-reference signs (signs that refer to a feature located on the left side of the road) and the right two strip rectangles are reserved for right-reference signs, by convention. Unlike the upper form strip, signs do not fight for a common pool of strip rectangles; rather, each sign nominates its required position. The required position is specified as the number of one strip rectangle in this strip, as shown in Figure 6.2.



Strip rectangle #1 **Strip rectangle #2** **Strip rectangle #3** **Strip rectangle #4**

Figure 6.2: Lower form strip, showing strip rectangles. Three signs are allocated: the vehicle is some distance from sign in strip rectangle #1, whereas signs in #3 and #4 are at their maximum display size. Strip rectangle #2 is empty (no sign allocated).

6.2.3) Controller Form

The controller form is located in the centre of the screen. It displays a plan map view of a VRS virtual world including current vehicle position and currently active sign regions, a list of received GPS data, a simple command button interface for starting and stopping the simulation, and a grid that displays information about current signs. This latter grid was useful for debugging purposes during prototype testing (as discussed in Chapter 7). A screen shot of the controller form is shown in Figure 6.3 (p. 6-5).

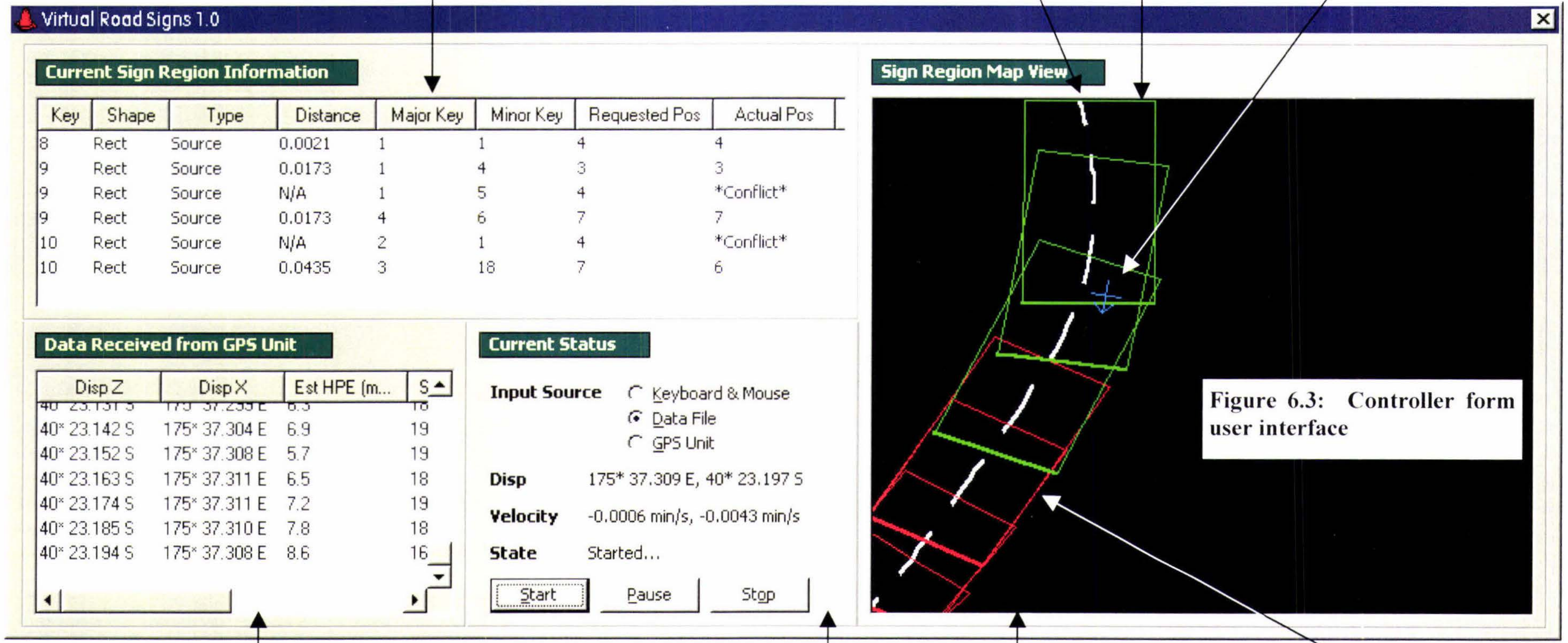
The controller form provides useful information to the user for the purpose of prototype testing and demonstration. In a real VRS system, the controller form would not exist, as this area of the screen is where a driver would see the real world. Also, the border outline of the upper and lower form strips would not be made visible. A driver would only observe the signs displayed in the upper and lower form strips.

Current sign information. Each grid row describes one sign associated with a currently active sign region. Each row lists region key number, region type (rectangle/ arc/ composite), distance to major axis (sign distance), major and minor sign key values, requested strip rectangle # and actual strip rectangle # allocated.

A sign region glows **green** when it is active (see [4.1])

Blue pointer indicates **current vehicle position** (at cross centre) and **velocity** (direction and length of arrow).

Road centre line



Current Sign Region Information

Key	Shape	Type	Distance	Major Key	Minor Key	Requested Pos	Actual Pos
8	Rect	Source	0.0021	1	1	4	4
9	Rect	Source	0.0173	1	4	3	3
9	Rect	Source	N/A	1	5	4	*Conflict*
9	Rect	Source	0.0173	4	6	7	7
10	Rect	Source	N/A	2	1	4	*Conflict*
10	Rect	Source	0.0435	3	18	7	6

Data Received from GPS Unit

Disp Z	Disp X	Est HPE (m...	S
40* 23.142 S	175* 37.304 E	6.9	19
40* 23.152 S	175* 37.308 E	5.7	19
40* 23.163 S	175* 37.311 E	6.5	18
40* 23.174 S	175* 37.311 E	7.2	19
40* 23.185 S	175* 37.310 E	7.8	18
40* 23.194 S	175* 37.308 E	8.6	16

Current Status

Input Source Keyboard & Mouse [✓ Data File] GPS Unit

Disp 175* 37.309 E, 40* 23.197 S

Velocity -0.0006 min/s, -0.0043 min/s

State Started...

Start | Pause | Stop

Figure 6.3: Controller form user interface

Received GPS data (one row added every two seconds). Each row shows current vehicle latitude and longitude, estimated HPE (metres), and speed and heading of vehicle (not visible) reported by GPS unit.

Status, showing input source, filtered/ predicted vehicle position ('Disp'), filtered/ predicted vehicle velocity, application state (started, stopped, pause – controll-ed by three command buttons).

Map view of one area of virtual world, corresponding to area of a quadrant of one node in quadtree. A number of rectangle sign regions can be seen. The view can be zoomed in or out as desired.

A sign region glows **red** when it is **inactive** (see [4.1]).

6.2.4) Data Sources

Three tracking data sources are supported in the Status pane:

1) A real GPS receiver. This option is selected when the prototype is running inside a vehicle and data is being received from a GPS receiver, in real-time.
2) A log file of real GPS data. The log file is prerecorded by connecting a GPS receiver to an in-vehicle laptop PC and navigating around a route where virtual road signs are to be displayed. Once complete, the log file is transferred to the lab and 'replayed' by setting this option and starting the prototype, thereby effecting a delayed simulation of virtual road signs.
3) Keyboard and mouse. This option allows the current vehicle position and velocity to be simulated in the lab by moving around the virtual world using keystrokes and a mouse. This greatly simplified prototype debugging and testing by allowing custom test case data to be quickly developed and simulated as the prototype evolved. In this manner, the time-consuming collection of real-world data was avoided until the prototype was ready to be rigorously tested using a real test world. Chapter 7 describes one such test world.

6.2.5) Application States

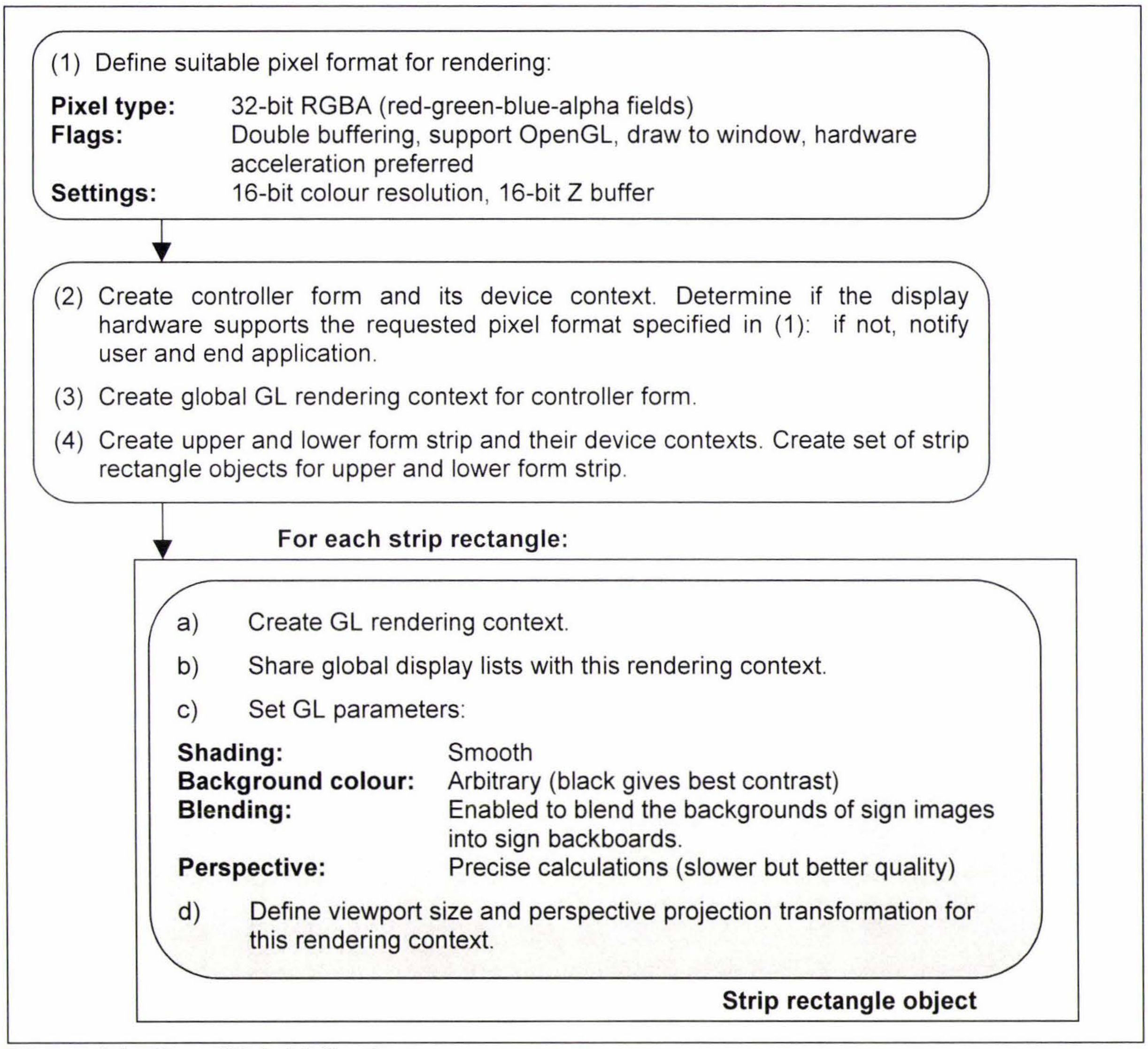
The prototype can be in one of three different states at any given time (see Figure 6.3):

1) Started. The vehicle position and velocity are directly controlled by the selected data source (data received from a GPS unit, data sourced from a logged data file, keystrokes or mouse clicks). Virtual signs are only updated in this state.
2) Paused. If the selected data source is the keyboard and mouse, attempts to change the vehicle position and velocity are ignored. If the data source is a logged data file, reading of the file is temporarily suspended. In both cases, current virtual signs remain visible but are not updated. If the data source is a GPS receiver, the option to pause the application is unavailable. If this option was supported, it is possible that the reported GPS data would significantly change between the time of pausing and the time when the prototype was 'started' again, which would invalidate predictions produced by the Kalman filter. By forcing the user to stop processing rather than pause it, all filtered and prediction data is flushed out and the filter state is re-initialised when processing recommences.
3) Stopped. The vehicle position and velocity cannot be set by any means and remains undefined. No virtual signs are visible.

6.3) Sign Rendering using OpenGL

6.3.1) Overview

OpenGL is a comprehensive 3D graphics API and several good texts have been written on its design and implementation. A full treatment is presented by Warhol (2001). This section summarises the important concepts of OpenGL initialisation relevant to the VRS prototype. The OpenGL initialisation process occurs at application start-up and sets up the necessary structures ready for sign rendering. It also defines those parameters that affect how virtual signs appear on-screen when rendered. Figure 6.4 illustrates the process.



(1) Define suitable pixel format for rendering:

Pixel type: 32-bit RGBA (red-green-blue-alpha fields)
Flags: Double buffering, support OpenGL, draw to window, hardware acceleration preferred
Settings: 16-bit colour resolution, 16-bit Z buffer

(2) Create controller form and its device context. Determine if the display hardware supports the requested pixel format specified in (1): if not, notify user and end application.

(3) Create global GL rendering context for controller form.

(4) Create upper and lower form strip and their device contexts. Create set of strip rectangle objects for upper and lower form strip.

For each strip rectangle:

a) Create GL rendering context.

b) Share global display lists with this rendering context.

c) Set GL parameters:

Shading: Smooth
Background colour: Arbitrary (black gives best contrast)
Blending: Enabled to blend the backgrounds of sign images into sign backboards.
Perspective: Precise calculations (slower but better quality)

d) Define viewport size and perspective projection transformation for this rendering context.

Strip rectangle object

Figure 6.4: OpenGL initialisation process

A *device context* (DC) is "a structure that defines a set of graphic objects and their associated attributes, and the graphic modes that affect output" (Microsoft Corporation, 1995). In step (2) in Figure 6.4, a *form device context* is the inner region of a graphics window where graphics can be drawn. In step (1), the desired pixel format for the controller form's DC is specified. *Double buffering* employs two frame buffers to store a rendered image prior to its display on-screen: while one buffer is currently displayed, the second buffer is written 'behind the scenes' and swapped back when complete to refresh the screen cleanly (Molofee, 2001). This avoids the visual artefacts associated with *single buffering* whereby a single frame buffer is refreshed while its contents are visible, such that for a short time period, the buffer contains part of the new image and old image. The form DC is notified that it will be used for OpenGL rendering and hardware acceleration is preferred; if supported, the PC video card will render OpenGL primitives (such as polygons) in hardware, relieving the system CPU of this task and yielding much faster rendering performance.

Once a form DC has been created, an OpenGL rendering region known as a *GL rendering context* (GLRC) is created on top of the form DC in step (3). Unlike a form DC which is completely generic, a GLRC is designed to render OpenGL primitives only. The upper and lower form strips are created, each with its own device context. Each strip rectangle on the two form strips is then initialised. A GLRC is created for each strip rectangle, and the set of global display lists (which store all compiled signs and font characters) is made accessible to the GLRC of each strip rectangle by 'sharing'. If these display lists were not shared, a complete set of display lists would have to be independently defined with respect to every GLRC, which would be impractical. This explains why a GLRC is created for the controller form in step (3) even though the form is not the target of OpenGL rendering: it provides the global GLRC with which all display lists are defined with respect to, and which can be shared amongst strip rectangles. For each strip rectangle, its display parameters and viewport are defined. The viewport is the subject of the next section.

6.3.2) Viewports and Viewing Frustums

The *viewport* of a strip rectangle is simply the inner region of the GLRC where OpenGL primitives can be rendered (Warhol, 2001). A viewport's *projection transformation* defines a *viewing volume*, which is used in two ways. The volume determines how a three-dimensional object is projected onto a two-dimensional screen, and it defines which objects or portions of objects are clipped out of the final image (Warhol, 2001). A perspective projection yields the familiar characteristic of foreshortening: the farther an object is from the viewpoint, the smaller it appears; in contrast, an orthographic projection preserves the size of a rendered object regardless of its distance away from the viewpoint. Here a perspective projection is the logical choice to give a realistic impression of a distant sign when a vehicle first enters its sign region, and the realistic illusion of movement as the sign approaches. The perspective viewing volume takes the shape of a truncated pyramid whose top has been cut off by a plane parallel to its base, as shown in Figure 6.5.a. This volume is known as a *viewing frustum*.

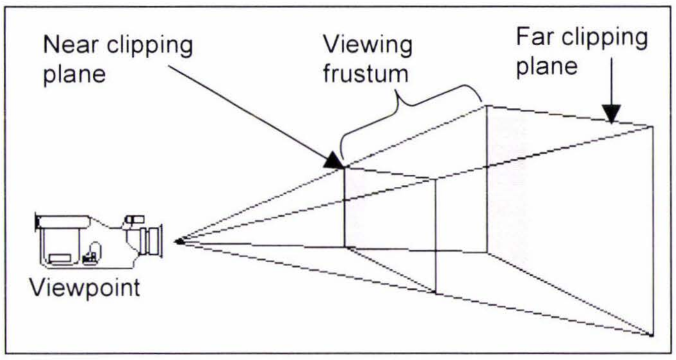


Figure 6.5.a: Viewing frustum (Microsoft Corporation, 1999).

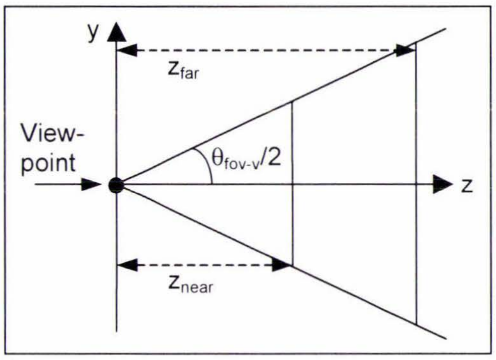


Figure 6.5.b: Side view of viewing frustum.

The shape of the viewing frustum is defined by four parameters (refer Figure 6.5.b):

1) Vertical angular field of view, θ_{fov_v}. This is directly related to the focal length of a camera lens: a wide field of view corresponds to a short focal length and produces a 'zoomed-out' or *wide-angle* image, whereas a narrow field of view corresponds to a long focal length and produces a 'zoomed-in' or *telescopic* image.
2) *Aspect ratio*, R_{aspect}, which defines the ratio of width to height of a vertical plane taken from the volume of the viewing frustum; for a virtual sign, this is equal to a strip rectangle's width divided by its height, in screen pixels.
3) Z coordinates, z_{near} and z_{far}, of the *near* and *far clipping planes* respectively. These are vertical planes that represent the near and far limits of visibility of rendered objects, as shown in Figures 6.5.a-b. Objects in front of the near clipping plane are not rendered, nor are objects behind the far clipping plane. In a virtual world in which the viewpoint is specified by a vehicle's current position and sensitive to all movements of the vehicle, and signs are specified in exact three-dimensional coordinates, these clipping planes would be useful in dictating when a virtual sign first becomes visible and is last visible as the vehicle passes it. However, this behaviour is not true by design for the VRS prototype (see [5.1.1]). Whether or not a sign is visible is determined by whether or not its sign region is active, and thus the clipping planes serve no purpose.

6.4) Rendered Sign Distance versus Actual Sign Distance

The maximum face area of a real road sign that is observed by a driver as he/she passes it can be estimated as soon as the sign becomes visible, when it may still be some distance away. A large sign becomes readable early to give a driver advance warning, whereas a small sign is not readable until much later. If a large sign and small sign are equally distant, the large sign appears to have greater area than the small sign. This intuitive behaviour is not favoured for virtual road signs, however. To maximise readability, a virtual sign should always occupy the maximum area of its strip rectangle when displayed at its largest size, regardless of its defined horizontal and vertical lengths. To achieve this, the apparent distance at which a sign is rendered in its strip rectangle (*rendered sign distance*) must be calculated to accommodate:

1) The actual distance between current vehicle position and sign position in the virtual world.
2) The width and height of the minimum enclosing rectangle $R_{enclosing}$ (see [5.3.1]) of the sign to be displayed. Let these parameters be represented as (L_x, L_y) respectively.
3) The current width and height of the strip rectangle in which the sign is displayed. For the purpose of 3D viewing, it is more convenient to represent the area of each strip rectangle using a camera analogy, rather than specifying the dimensions of the rectangle measurements in pixels.

To give an example of this concept, consider two signs whose minimum enclosing rectangles are of different sizes. One sign is of dimensions $(L_x, L_y) = (1, 1)$ and the other sign is of dimensions $(L_x, L_y) = (2, 1)$. Let us assume that the two strip rectangles which will display these signs are square and of equal size. This means that the horizontal field of view of both strip rectangles equals their vertical field of view; that is, the aspect ratio is one. In the real world, the 2 x 1 sign would appear to be as tall and twice as wide as the 1 x 1 sign when the two signs are viewed from the same distance. Let this common viewing distance be the shortest distance that permits both signs to just fit inside the square viewing area (strip rectangle), as shown in Figure 6.6.

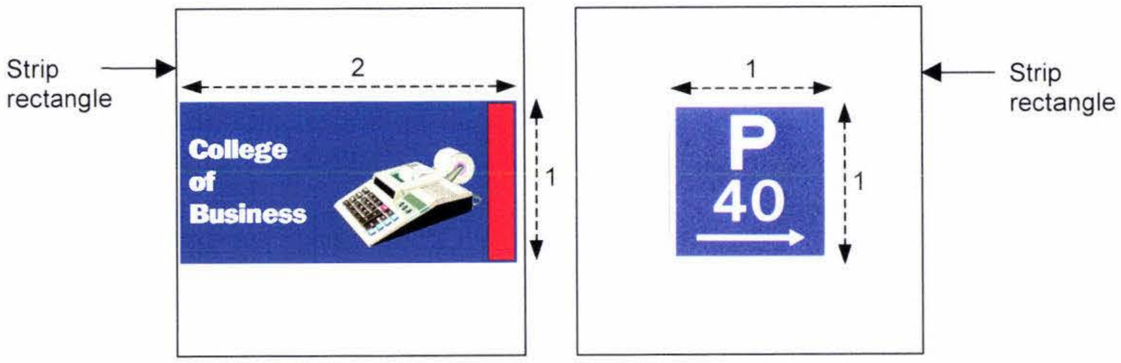


Figure 6.6: Two signs of different sizes viewed from a common distance, such that both signs are contained within their strip rectangle

The 2 x 1 sign cannot be made any larger because doing so would mean part of the sign would no longer fit inside the strip rectangle. However, the 1 x 1 sign can be made twice as large to completely fill the strip rectangle. To increase the display size of the 1 x 1 sign without changing its size, the rendered sign distance of the 1 x 1 sign is intentionally made less than the rendered sign distance of the 2 x 1 sign. In this case, the rendered sign distance of the 1 x 1 sign is set equal to half that of the 2 x 1 sign at all times. Figure 6.7 shows the situation when both signs appear at their largest size. At this time, both signs completely fill the horizontal field of view of their strip rectangles; however, only the 1 x 1 sign completely fills the vertical field of view of its strip rectangle as well, as shown in Figure 6.8.

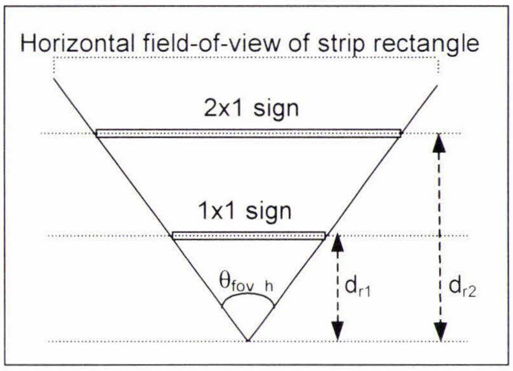


Figure 6.7: Plan view of world, showing different rendered distances, d_{r1} and d_{r2}, of two signs.

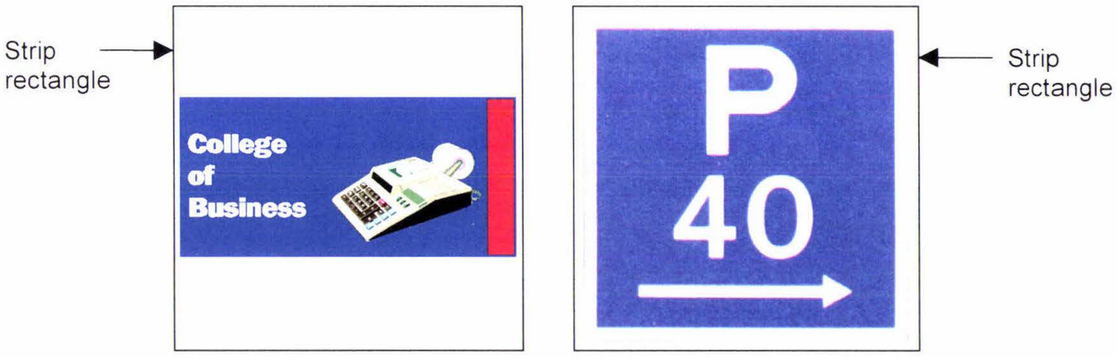


Figure 6.8: Two signs of different sizes viewed from different rendered distances d_{r1} and d_{r2} (see Figure 6.7), such that each sign is displayed at its largest possible size

Box 6.1 describes the process of calculating the rendered sign distance, d_r, given the sign's actual distance, d_a.

Immediately prior to when a virtual sign S first becomes visible in a strip rectangle R:

1) Calculate distance d_{r0}, which is that rendered distance that causes sign S to be displayed at its largest size:
   a) Calculate $d_{r0\ horizontal}$, the rendered distance that causes sign S to completely fill the horizontal field of view of strip rectangle R, according to Eq. 6.1.a.
   b) Calculate $d_{r0\text{-}vertical}$, the rendered distance that causes sign S to completely fill the vertical field of view of strip rectangle R, according to Eq. 6.1.b.
   c) Calculate d_{r0} as the greater of the two values, $d_{r0\text{-}horizontal}$ and $d_{r0\text{-}vertical}$. This ensures that none of the sign's area is obscured in either field of view at its maximum display size.
2) Calculate $d_r = d_{r0}.(1 + c.d_a)$, where c is a constant that dictates the display size of a sign when first displayed for a given sign region length, horizontal and vertical field of view. A small value of c causes a sign to initially appear large and grow only slightly as a vehicle approaches it, whereas a large value of c causes a sign to appear to approach from a far distance.

Box 6.1: Calculation of rendered sign distance

$$d_{r0-horizontal} = \frac{L_x \sin\theta_{fov-x}}{2R_{aspect}(1-\cos\theta_{fov-x})}$$

Eq. 6.1.a: Calculation of $d_{r0\text{-}horizontal}$ as described in Box 6.1.

$$d_{r0-vertical} = \frac{L_y \sin\theta_{fov-y}}{2(1-\cos\theta_{fov-y})}$$

Eq. 6.1.b: Calculation of $d_{r0\text{-}vertical}$ as described in Box 6.1

6.5) Allocation of Signs to Screen Locations

6.5.1) Overview

From a driver's perspective, the lifetime of every virtual sign begins when it first appears on-screen in its strip rectangle, and ends when the vehicle departs from the sign region containing that sign. The task of deciding which strip rectangle shall contain which sign is the responsibility of a sign allocation scheme. Such a scheme is necessary because the area available for sign rendering on any arbitrary display device is limited, and it is likely that in world areas densely populated with sign regions, there may be times when more signs are vying for display than there is space available. The simplest solution to this 'sign conflict' is to force the world designer to ensure that no such conflicts can possibly occur in a virtual world for any vehicle position. While feasible in a simple world with few densely-populated regions, proving the non-existence of conflicts is not a trivial exercise for a world of any reasonable size. In addition, the existence of conflicts is directly dependent on the available display area of the chosen display device: thus it is only possible to remove all conflicts for a given display device. Even though the prototype display device was known in advance, it is still sensible to make as few assumptions as possible about the target display device. Here, the VRS prototype employs a sign allocation scheme that is tailored to suit the display area of a CRT monitor. However, the allocation scheme is sufficiently flexible that it can be easily adapted to accommodate the available display area of any display device.

Sign allocation occurs when a new list of sign region objects, $L_{\text{active SR}}$, is generated by the data subsystem (see [4.5.1]), corresponding to a new vehicle position. Of these sign regions and their contained signs, three categories exist:

1) Category C1: those regions entered by a vehicle since the display was last refreshed, whose signs are yet to be allocated to a strip rectangle.
2) Category C2: sign regions which remain active from previous iterations, whose signs retain their strip rectangle initially allocated.
3) Category C3: sign regions that are no longer active, whose signs have reached the end of their display lifetime and whose strip rectangles are to be deallocated, ready to display new signs.

The allocation scheme described here classifies each sign contained in the sign region list, $L_{\text{active SR}}$, as belonging to one of these three categories by comparing the contents of list $L_{\text{active SR}}$ with the current status of all strip rectangles. To conclude that a sign is of category C2 and not C1, the sign must be matched with one sign currently allocated to a strip rectangle. However, care must be taken in defining what constitutes a matching sign. Two signs with the same visual content but which are contained by different sign regions, or which request different screen positions, are two independent signs, and thus they cannot be described as 'matching'. Therefore, the enforced rule is that for two signs in a virtual world to match, all of the following parameters must be the same for both signs:

1) Sign major and minor key.
2) Requested strip rectangle position.
3) Sign region with which the two signs are associated.

6.5.2) Upper Form Strip Sign Allocation

Sign allocation in the upper form strip assigns a strip rectangle to a sign based on its preferred position (as explained in [6.2.1]). Rather than opt for a cooperative scheme, strip rectangles are assigned on a first-come, first-served basis. Once a strip rectangle has been allocated to display a sign S, that strip rectangle does not become available (empty) to display another sign until S has disappeared from view. A *sign conflict* is reported if more than three signs request display in the upper strip concurrently. Box 6.2 describes the algorithm that allocates signs to strip rectangles in this form strip. This algorithm makes reference to the array, $A_{\text{sign data}}$, described in (5.2.2) that contains pointers to all signs associated with a sign region. An example of sign allocation in the upper form strip follows an explanation of sign allocation in the lower form strip.

(1) **Merge signs to be displayed in the upper form strip together:**

Each active sign region specifies its own array, $A_{sign\ data}$, that describes the set of signs to be displayed. Indices 5-7 correspond to signs with a requested left, centre, right position respectively and take priority over index 0, which corresponds to a sign with no preferred position.

Merge the contents of array $A_{sign\ data}$ at indices (5, 6, 7, 0) across all active sign regions into a single array, A_{merged}. A_{merged} contains four elements at indices 1-4. The merge operation copies each sign pointed to by each $A_{sign\ data}$ array at indices (5, 6, 7, 0) into an element of A_{merged}, starting at index 1. The number of signs merged is n.

Merging speeds up sign allocation because only the single array A_{merged} must be searched in steps (2) and (3) below, rather than each array $A_{sign\ data}$ of all active sign regions. It is initially assumed that each sign stored in array A_{merged} is of category C1; that is, new and waiting to be allocated to a strip rectangle. This is indicated by setting a flag, SignIsNew(i), true for elements at index $i \in [1\ldots n]$ in A_{merged}.

(2) **Empty all strip rectangles in upper strip whose sign is out-of-date:**

```
For strip rectangle R = 5 to 7:
    If R has a sign already allocated, $S_{allocated}$ (sign is currently visible):
        Search array $A_{merged}$ for sign $S_{allocated}$:

        If found (at index i in $A_{merged}$), sign $S_{allocated}$ is of category C2 – it is current but has
        already been allocated a strip rectangle, so no sign allocation is necessary: set
        SignIsNew(i) false

        If not found, sign $S_{allocated}$ is of category C3 – it is no longer current, so deallocate
        (empty) strip rectangle R, ready for another new sign.
```

(3) **Allocate all signs to strip rectangles:**

```
For i = 1 to n, the number of signs stored in array $A_{merged}$:
    If sign at index i in $A_{merged}$, S, still requires allocation to a strip rectangle (that is, if
    SignIsNew(i) is true):

        Search for an empty strip rectangle:
    a)  If the requested position of sign S is left, or no position is requested, scan strip
        rectangles in order from left to right.

    b)  If the requested position of sign S is centre, scan strip rectangles in order: centre,
        right, left.

    c)  If the requested position of sign S is right, scan strip rectangles in order: right,
        centre, left.

        If an empty strip rectangle was found in (a), (b) or (c), allocate sign S to it. If not,
        flag sign S as conflicting.
```

Box 6.2: Upper form strip sign allocation algorithm

6.5.3) Lower Form Strip Sign Allocation

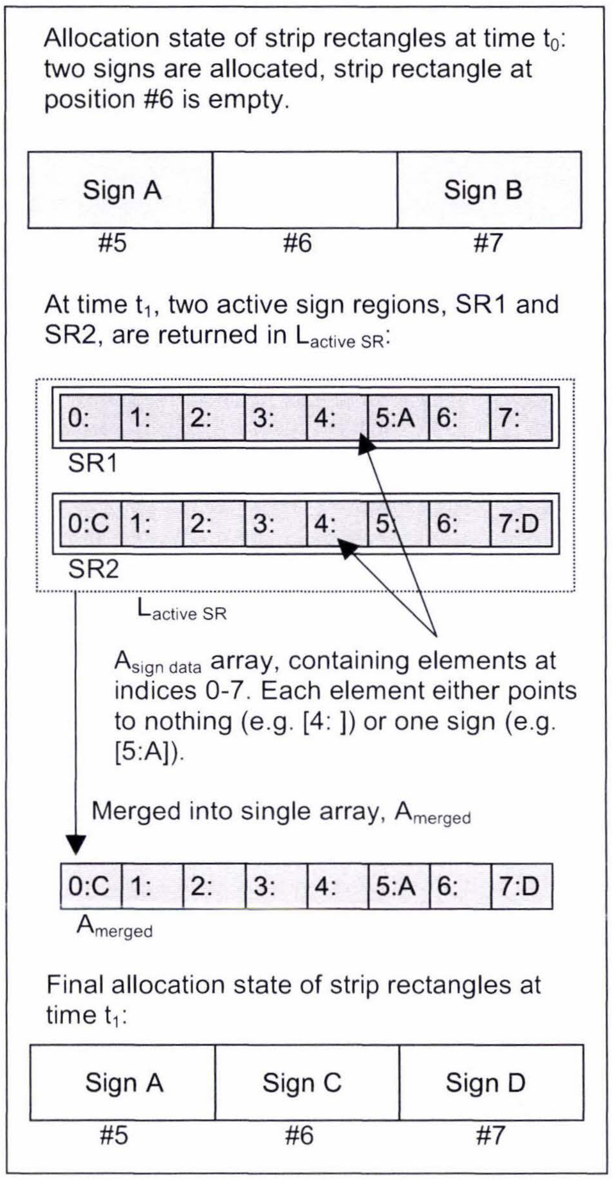
Sign allocation in the lower form strip is simplified by the fact that signs which appear in this strip request an exact position. Instead of processing strip rectangles in a group as is done for the upper strip, each strip rectangle can be treated independently. Like the upper form strip, once a strip rectangle has been allocated to display a sign S, that strip rectangle does not become available (empty) to display another sign until S has disappeared from view. If a sign is currently allocated to the requested strip rectangle, a conflict is reported and no attempt is made to allocate the sign to another strip rectangle. Box 6.3 describes the algorithm that allocates signs to strip rectangles in this form strip. An example of sign allocation in the lower form strip follows Box 6.3.

```
For strip rectangle R = 1 to 4:

Assume a new sign is waiting to be allocated to this strip rectangle (category C1). This is indicated by
setting a flag, SignIsNew(R), true.

(1)   Empty strip rectangle R if its sign is out-of-date:

      If R has a sign already allocated, $S_{allocated}$ (sign is currently visible):
            Search element at index R in array $A_{sign\ data}$ of all active sign regions for sign $S_{allocated}$:

                  If found, sign $S_{allocated}$ is of category C2 – it is current but has already been allocat-
                  ed a strip rectangle, so no sign allocation is necessary: set SignIsNew(R) false.

                  If not found, sign $S_{allocated}$ is of category C3 – it is no longer current, so deallocate
                  (empty) strip rectangle R, ready for another new sign.

(2)   Allocate a sign to strip rectangle R:

      If strip rectangle R is still eligible for sign allocation (that is, if SignIsNew(R) is true):
            Search element at index R in array $A_{sign\ data}$ of all active sign regions for any sign, S. If
            found:
                  If strip rectangle R is empty (no sign currently allocated), allocate sign to R, else
                  flag this sign S as conflicting.
```

Box 6.3: Lower form strip sign allocation algorithm

6.5.4) Examples of Sign Allocation

Figures 6.9.a and 6.9.b present simple sign allocation examples for both form strips:


Allocation state of strip rectangles at time t_0: two signs are allocated, strip rectangle at position #6 is empty.
Sign A | | Sign B
#5 #6 #7
At time t_1, two active sign regions, SR1 and SR2, are returned in $L_{active\ SR}$:
0: 1: 2: 3: 4: 5:A 6: 7:
SR1
0:C 1: 2: 3: 4: 5: 6: 7:D
SR2
$L_{active\ SR}$
$A_{sign\ data}$ array, containing elements at indices 0-7. Each element either points to nothing (e.g. [4:]) or one sign (e.g. [5:A]).
Merged into single array, A_{merged}
0:C 1: 2: 3: 4: 5:A 6: 7:D
A_{merged}
Final allocation state of strip rectangles at time t_1:
Sign A | Sign C | Sign D
#5 #6 #7

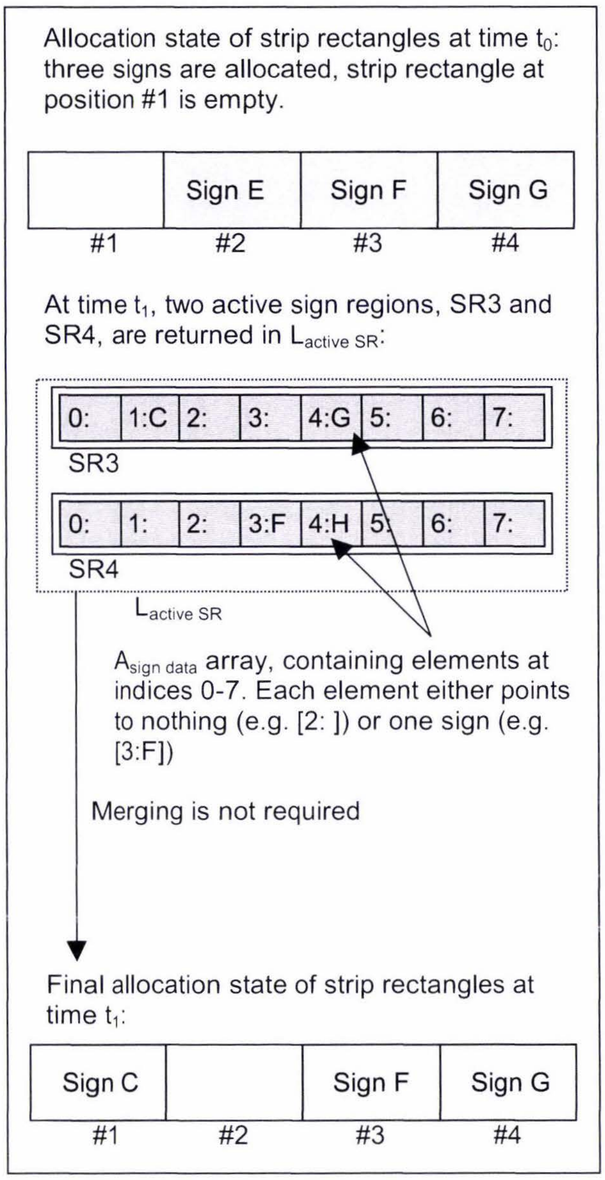

Allocation state of strip rectangles at time t_0: three signs are allocated, strip rectangle at position #1 is empty.
| Sign E | Sign F | Sign G
#1 #2 #3 #4
At time t_1, two active sign regions, SR3 and SR4, are returned in $L_{active\ SR}$:
0: 1:C 2: 3: 4:G 5: 6: 7:
SR3
0: 1: 2: 3:F 4:H 5: 6: 7:
SR4
$L_{active\ SR}$
$A_{sign\ data}$ array, containing elements at indices 0-7. Each element either points to nothing (e.g. [2:]) or one sign (e.g. [3:F])
Merging is not required
Final allocation state of strip rectangles at time t_1:
Sign C | | Sign F | Sign G
#1 #2 #3 #4

Figure 6.9.a: Example sign allocation in upper form strip: sign A is of category C2 (current), sign B is of category C3 (old), signs C and D are of category C1 (new).

Figure 6.9.b: Example sign allocation in lower form strip: signs C and H are of category C1 (new), sign E is of category C3 (old), signs F and G are of category C2 (current). Sign H conflicts with sign G and is not allocated.

It is noted in (4.5.2) that a new list of active sign regions, $L_{active\ SR}$, is not passed out if the list is found to be identical to the list passed out in the previous iteration. In this case, the same set of signs remain visible as for the previous iteration. Thus, there is no need to execute the allocation algorithm for either form strip at all.

For each sign region object, the world designer is responsible for ensuring that signs corresponding to each sign region are distributed in such a way that ensure signs have the best chance of being displayed. For example, two signs within the same region cannot both request the left-hand strip rectangle #5 in the upper form strip, or request the far-left position #1 in the lower form strip. Also, it makes no sense to associate four signs with four respective positions 5, 6, 7 and 0 because only three strip rectangles are available for sign display in the upper form strip.

It is worth noting that this sign allocation scheme makes no attempt to rank signs in terms of their importance; that is, no sign has a priority higher than any other sign in terms of whether it should be displayed or not. For the VRS prototype, it was necessary to restrict the set of sign categories in the main test world to a manageable yet effective size; the resulting categories were all considered to be of equal importance. However, a general VRS system should support signs of different priority, which is a straightforward extension to the current prototype. The decision as to which signs qualify for a higher priority is then the choice of the world designer. For example, if road safety is deemed most important to the driver of a vehicle, then it makes sense that a virtual sign alerting the driver to a crash scene ahead should be given higher priority than a sign directing the driver to the nearest Post Office. Higher priority signs would be processed first during initial sign allocation, whereas highest priority signs might be displayed pre-emptively at any time, possibly displacing currently allocated signs.

6.5.5) An Example of Conflicting Signs

As described earlier, a sign conflict is generated when a sign's requested position cannot be honoured. An example of conflicting signs is shown in the 'current sign information' grid of Figure 6.3 in (6.2.3). In this figure, six signs are currently scheduled to be displayed based on the current set of active sign regions; however, due to space constraints (that is, a shortage of available strip rectangles), only four signs were allocated to screen locations. Signs described by rows 1, 2 and 4 of the grid were allocated the strip rectangle position they requested. The sign described by row 6 was allocated strip rectangle #6 (centre, upper form strip) because its requested position #7 was unavailable. Rows 3 and 5 indicate that the respective sign in each case could not be displayed because the requested strip rectangle #4 was already allocated to sign (1, 1) (listed on the top row).

6.6) Summary

The VRS prototype displays virtual signs in two areas on-screen. one above a driver's line of sight and one below. A third area is reserved for the purpose of prototype testing and debugging, and would not be displayed in a real VRS system. The prototype allows one to collect the positional data of a vehicle separately from the task of displaying virtual signs. This was an important facility in this case. because it was not possible to find a suitable display device that could render virtual signs to a driver in real-time. In addition, it allowed the prototype to be demonstrated away from the confines of a vehicle. OpenGL is used to render virtual signs on-screen at their optimal size. Two simple algorithms allocate virtual signs to screen locations where they are displayed on a real-time basis. This is also necessary for a general VRS system, because one cannot guarantee that there will be enough screen space to display all the virtual signs intended to be seen by a driver at any given time. Consistent adherence to conventions regarding the on-screen display position of signs simplifies the user interface. In the next chapter, we discuss how a test world was created to test the VRS prototype, and the important results of testing.

CHAPTER 7
Prototype Testing

7.1) Introduction

One of the main objectives of the VRS prototype was to demonstrate one application of a virtual road sign system operating in the real world. The intention was to illustrate how a real VRS system might be used to provide travel information to a specific target user. In this case, the target user was chosen to be a new visitor to Massey University, travelling by car. A loop road encircles the major university buildings and provides the main access to most campus areas. Although the real name of this road is University Ave, it is colloquially referred to as the Ring Road, and this name will be used in this thesis. The Ring Road was chosen as an ideal road on which to set up a VRS world. It is suitably small for testing purposes, yet provides ample opportunities to erect a number of different signs around it, pointing out various campus facilities to the target user. Virtually all visitors to the university must traverse the Ring Road at some point in their travels, so the virtual Ring Road signs would be observed and of use to a potentially large number of road users.

Construction of a comprehensive Ring Road test world began by identifying four categories of information that may be of interest to the target user. These were:

1) Parking zones. The Ring Road contains seven types of parking: general, motorcycle, bicycle, reserved (for example, for university staff), time restricted, loading zones, and mobility parking. It is important that the target user knows where they can and cannot park their vehicle and for how long, depending on their intended length of stay.
2) Bus stops. As the target user has their own means of transport, they may have no need to wait for public transport into the city. In this case, bus stop signs indicate that the road space is unavailable for general car parking. Regardless, it is good practice to let drivers of private vehicles know that such public transport is available.
3) Buildings and facilities. It is likely that the target user will be seeking to visit one or more specific campus facilities once they have parked their vehicle. Signs displaying building names such as the Veterinary Sciences Tower and Residential Services can indicate the location of these facilities.
4) Road names. The Ring Road branches off onto a number of minor roads that provide

access to other campus resources and services. Road names provide the target user with an alternate means of navigating their vehicle and finding their intended destination.

Chapter 1 notes that virtual road signs intend to provide a driver with two categories of information. In this sense, categories (1) to (3) can be considered as providing en-route trip information to the target user, while category (4) can be loosely considered as providing navigation information. Virtual road name signs are intended as a very simple navigational aid and cannot claim to aid the target user for the purpose of route guidance.

Sixty-seven features were identified around the Ring Road in these four categories. The latitudinal and longitudinal coordinates of each feature were first measured by walking around the Ring Road with a handheld GPS receiver. The road centre line provided a common reference datum. At the exact location of each feature, the GPS receiver was placed adjacent to the feature on the road centre line and its coordinates were recorded. In many cases, because some features were in such close proximity to each other, a group of features was taken as lying at the same coordinates. This meant that only 34 distinct GPS coordinate pairs were recorded. A table of all feature coordinates may be found in (G.1).

7.2) Ring Road Sign Design

A total of 67 virtual signs were designed, one sign per feature (details are given in [G.1]). An example of one virtual sign that displays the name of a building (category 3 above) was given in Figure 5.7 of (5.3.1). Figures 7.1.a-c give examples of signs for the other three information categories.

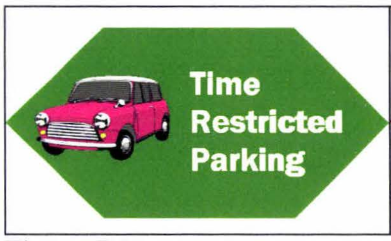


Figure 7.1.a: Example of a virtual parking sign

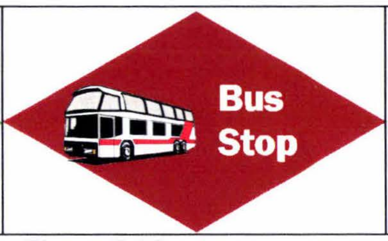


Figure 7.1.b: Example of a virtual bus stop sign

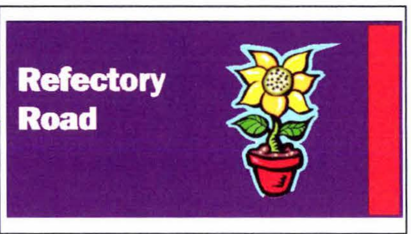


Figure 7.1.c: Example of a virtual road name sign

As described in (5.3.1), the sign backboard is shape-coded and colour-coded to permit the target driver to quickly deduce which of the four information categories a sign belongs to. Each type of parking sign displays a different-coloured vehicle to distinguish the parking types at some distance. The conventions adopted are listed in (G.2). In addition, the GUI form strip in which signs can appear is restricted based on their category. Parking zone and bus stop signs can only appear in the lower form strip, and building name and road name signs can only appear in the upper form strip. It makes sense to display the latter two sign categories in the upper strip where each strip rectangle is larger (and thus the maximum display size of a sign is larger) because these signs typically contain more textual information. Recall that the strip rectangle position of a sign displayed in the lower GUI form strip is important because it indicates the sign's referring direction (see [6.2.2]). Therefore, the use of red vertical bands to indicate direction (see [5.3.1]) is only necessary on signs displayed in the upper form strip. This saves valuable space on signs displayed in the lower form strip.

7.3) Calculation of Coordinate System Transformation Parameters

7.3.1) Overview

Once a feature set had been measured and the virtual signs describing each feature had been designed, the task of defining all Ring Road sign regions was addressed. Manual calculation of all sign region parameters would have been extremely tedious. Therefore, a separate software application was designed and coded to generate these regions automatically. The major advantage was that region parameters such as minimum/maximum region length and width could be changed and a completely new set of regions created instantly to reflect the change. This was indeed done several times before the final set of regions was arrived at.

Knowledge of feature coordinates is inadequate to generate sign regions on their own: each coordinate pair defines only where a given feature is located, and says nothing about the shape of the road leading up to that feature. To address this problem, a scale A3 plot of the Ring Road was obtained and a road centre line drawn onto the plot by hand and eye in red ink (a copy of this plot is given in [G.3]). The plot was scanned and all road outlines removed by image processing, leaving only the red centre line. This image was then downsampled to 75 pixels per inch, corresponding to an approximate real-world distance of 0.67 metres per pixel, and skeletonised to leave a line of single pixel width. The paper plot's coordinate system is the third main coordinate system utilised by the VRS prototype; the two other coordinate systems were previously introduced in (2.4.1) (see Appendix A for a description of all three coordinate systems). The idea was that if the transformation parameters that convert between the GPS coordinates of each feature (in latitudinal and longitudinal minutes) and the paper plot (in pixels) could be calculated, the plotted road centre line could provide enough information in conjunction with the feature coordinates to generate the set of sign regions. These transformation parameters are listed in Table 7.1:

Parameter	Description of Parameter
$[x\ z]_{\text{origin pixel, GPS}}$	(x, z) coordinates of paper plot's origin when represented in GPS coordinate system.
$[x]_{\text{Scale GPS --> pixel}}$	Number of pixels in paper plot's coordinate system per x unit in GPS coordinate system; that is, per longitudinal minute.
$[z]_{\text{Scale GPS --> pixel}}$	Number of pixels in paper plot's coordinate system per z unit in GPS coordinate system; that is, per latitudinal minute.
$\theta_{\text{GPS --> pixel}}$	Clockwise angle of rotation that maps the axes of the GPS coordinate system onto the axes of the paper plot's coordinate system, such that the axes of both systems coincide.

Table 7.1: Transformation parameters that convert between GPS and paper plot coordinates

Appendix A describes how these parameters are used to transform between GPS coordinate system (1), in which all Ring Road features are measured, and system (3), in which the paper plot is represented. As mentioned in (2.4.1), the full GPS coordinates of degrees and minutes are not processed. For the Ring Road, all features lie at the same integer value of 40 degrees South (latitude) and 175 degrees East (longitude). Therefore, the value of degrees can be safely ignored. This is equivalent to redefining the true origin of GPS coordinate system (1) by subtracting a constant offset of (40 degrees South, 175 degrees East) from all feature coordinates.

7.3.2) Minimisation Algorithm

Values of the transformation parameters listed in Table 7.1 were calculated by performing multiple iterations of a least-squares minimisation algorithm. The objective of the minimisation algorithm was to gradually converge towards the optimal set of transformation parameters that yielded a global minimum sum-of-squared-error. At each iteration, the algorithm employed a brute-force search to find this optimal parameter set. For the first iteration, an estimate of the optimal parameter set was made and used as a central starting parameter set. Prior to algorithm execution, it was unknown exactly how well-conditioned the six-dimensional parameter space was, as is often the case with optimisation problems. Thus an attempt was made to estimate values of the central starting parameter set as close as possible to the optimal set (details of how these estimates were arrived at are given in [G.4]). The algorithm then searched all combinations of parameters in the local neighbourhood of the central starting set at a coarse resolution; that is, the range of values tested per parameter was intentionally made large. At the end of the first iteration, the algorithm reported the optimal parameter set (that yielded minimum SSE) found at this coarse resolution. As the algorithm was repeated iteratively, the central starting parameter set was chosen as the optimal parameter set found in the previous iteration, and the search resolution was halved. This process was continued until the minimum practical resolution of each parameter was reached.

The minimisation algorithm is summarised in Box 7.1. This summary refers to a simple example shown in Figures 7.2.a-b and Figures 7.3.a-c.

1) Starting from one point on the Ring Road centre line, place points around the centre line in a clockwise direction at a short uniform spacing, as shown in Figure 7.2.a. The resulting set of points will be known as the set of *minor* points.

2) Starting from the same initial point on the Ring Road centre line as in (1), identify every kth minor point moving around the centre line in a clockwise direction. Define every kth point as one *major* point, as shown in Figure 7.2.b.

3) For every set of transformation parameters to be tested this iteration:
   a) For every pair of GPS feature coordinates (x_G, z_G) measured around the Ring Road, process the set of major and minor points in a hierarchical fashion to find the nearest pixel on the Ring Road centre line to (x_G, z_G), as shown in Figures 7.3.a-c. The distance between (x_G, z_G) and the nearest pixel is then the error, E, and the squared error is SE.

   b) Record the total error, SSE, as the sum of SE taken over all measured GPS coordinates.

4) Report the optimal parameter set as that set of transformation parameters that yields the minimum SSE value.

Box 7.1: Summary of minimisation algorithm

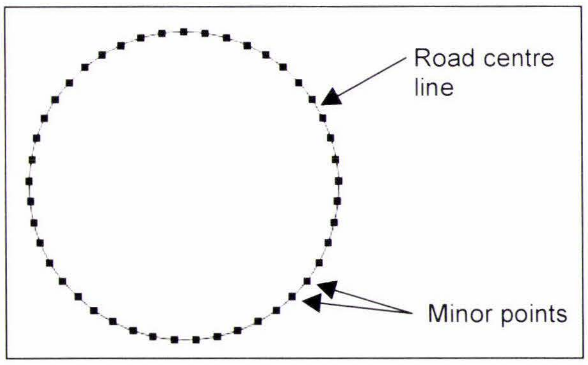


Figure 7.2.a: Example of minor points spaced around a circular road centre line, created in step (1) of Box 7.1.

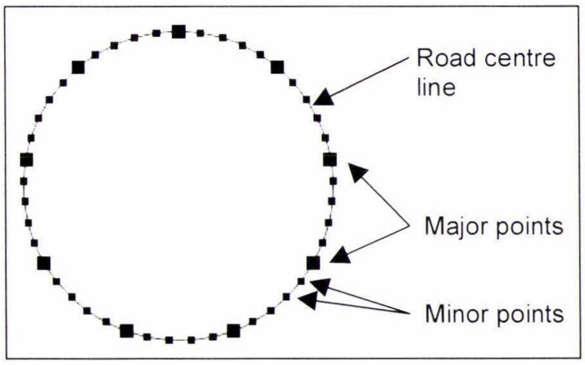
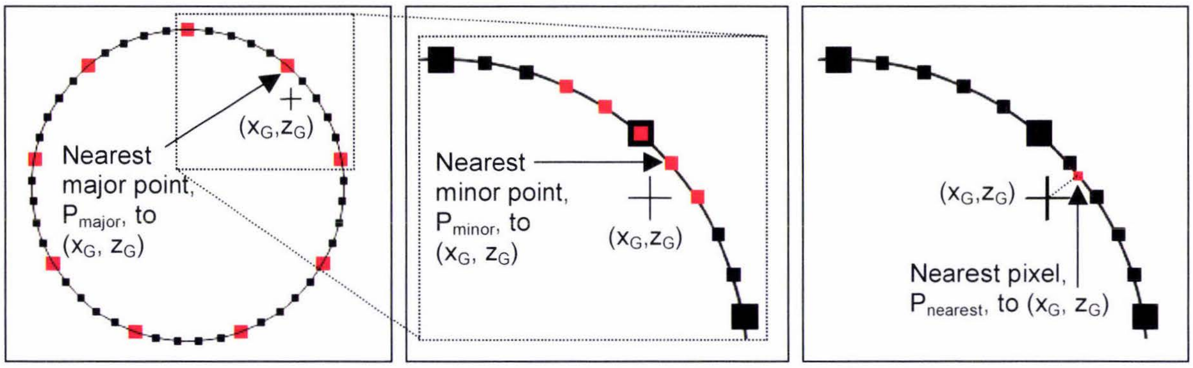


Figure 7.2.b: Example of minor and major points spaced around a circular road centre line, created in steps (1)-(2) of Box 7.1.

In Box 7.1, the use of major and minor points in a hierarchical fashion dramatically reduces the amount of time taken to find the minimum distance between each GPS coordinate pair and the nearest centre line pixel. This optimisation is possible because the Ring Road centre line is a fairly simple shape. If major and minor points were not utilised, it would be necessary to compute the distance between each GPS coordinate pair and every centre line pixel, which would be extremely slow.



Nearest major point, P_{major}, to (x_G, z_G)
(x_G, z_G)
Nearest minor point, P_{minor}, to (x_G, z_G)
(x_G, z_G)
(x_G, z_G)
Nearest pixel, $P_{nearest}$, to (x_G, z_G)

Figure 7.3.a: Search the set of major points (red) for that major point P_{major}, that lies nearest GPS feature coordinates (x_G, z_G)

Figure 7.3.b: Determine which minor point (red), P_{minor}, in the immediate vicinity of point P_{major}, is nearest (x_G, z_G)

Figure 7.3.c: Determine which centre line pixel (red), $P_{nearest}$, in immediate vicinity of point P_{minor}, is nearest (x_G, z_G)

The final value of minimum SSE found is a measure of the degree of fit of all Ring Road feature coordinates as a whole around the Ring Road centre line. An optimisation method is required for this calculation, as opposed to a direct solution, to account for two sources of uncertainty:

1) The error incurred when measuring the coordinates of Ring Road features using GPS.
2) The estimated position of the centre line drawn on the scale Ring Road plot by hand.

7.3.3) Results of Minimisation Algorithm

If the GPS coordinates of the Ring Road features were measured with no error and the red skeletonised centre line was perfectly drawn, we would expect a global minimum SSE value of zero. In other words, we would expect that some combination of parameters would yield a perfect transformation between the GPS coordinate system and paper plot coordinate system. In this case, these conditions were known to be false and thus the brute-force optimisation method was justified. Figure 7.4 shows the actual minimum SSE values found by the algorithm over successive iterations.

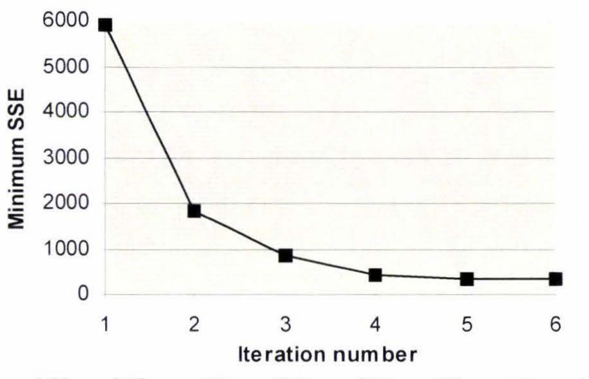


Figure 7.4: Graph of minimum SSE found per iteration

As can be seen, the minimum SSE values decreased rapidly as the algorithm converged quickly to an optimal solution for the parameter set. Details of the range of parameter values tested per iteration, and the final optimal parameter values, are given in (G.6). The final minimum value of SSE = 361 corresponds to an RMS error of only 3.26 pixels per GPS coordinate pair. This corresponds to an approximate real-world error of 2.2 metres, a surprisingly small error. This is somewhat less than the RMS error of about five metres measured in (3.2). The set of 34 GPS feature coordinates were measured over a 90 minute period, so it is possible that due to the high temporal coherence of GPS positional error, the positional errors of the measured feature coordinates were more correlated than expected. This would explain the very low RMS error calculated here. (G.3) contains a full-page scale plot of the skeletonised Ring Road centre line, all measured GPS feature coordinates fitted using the optimal set of transformation parameters found, and all major points.

7.4) Ring Road Sign Region Generation

7.4.1) Overview

Once an optimum set of transformation parameters had been calculated, sign regions could be generated. Virtual signs were to be made visible to vehicles travelling around the Ring Road in a clockwise direction only. The centre of each sign region was aligned with the road centre line of the Ring Road to ensure that the region was properly moulded to the shape of the road. By centring each sign region on the road centre line, each region covered both road lanes, which ensured that a clockwise-travelling vehicle would continue to observe the correct set of virtual signs even if it periodically crossed the centre line (for example, to pass another vehicle). The width of all sign regions was fixed and set equal to twice the nominal lane width, L_{lane}. If GPS positional coordinates were measured with perfect accuracy and no drift, L_{lane} would represent the true width of one road lane. Because the former statement is false, L_{lane} was intentionally made much larger than the actual width of a road lane, as described in (3.1.2). The major axis of all signs was located a short distance $L_{precedence}$ in front of (a small anticlockwise angle preceding) its corresponding feature. This was necessary to give the target user a chance to react after the sign appeared large enough to read.

In a general world, it is sensible to specify two further parameters to ensure that regions of unreasonable size are not generated. For practicality, the curved length of road contained within each sign region should lie within an interval defined by its lower bound, L_{min}, and upper bound, L_{max}. This rule was adopted for the Ring Road sign regions. Figure 7.5 shows how these parameters are defined for an arbitrary rectangular sign region. Those minor points defined around the centre line by the algorithm described in Box 7.1 earlier (section [7.3.2]) are particularly useful entities for measuring curved length along the centre line.

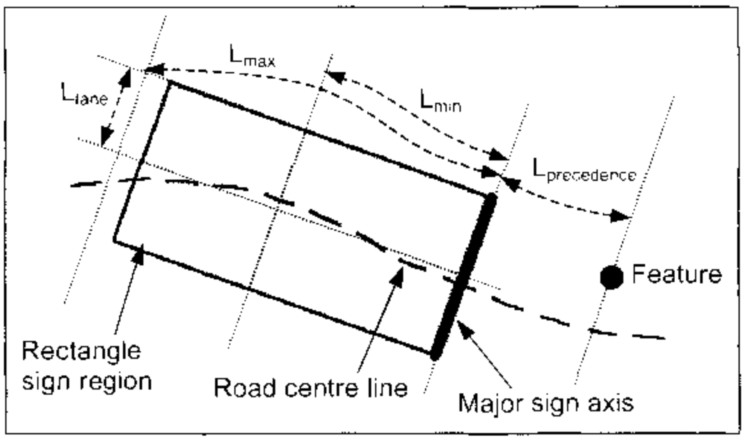


Figure 7.5: Important parameters $L_{precedence}$, L_{min}, L_{max} and L_{lane} of fitted sign regions.

Both rectangle and arc sign regions were suitable candidates for placement around the Ring Road. An algorithm was designed to select the most suitable combination of region types for a given segment of the Ring Road (the mechanics of this algorithm are discussed later). Clearly, the most suitable combination of regions depends on the shape of the centre line where the regions are to be 'fitted'. Due to the different shape of rectangle and arc regions, a different technique of fitting each region type to the centre line was necessary for each.

For each feature at one pair of GPS coordinates (x_G, z_G), the centre line segment to be processed was identified by finding the nearest centre line point $P_{nearest}$ to (x_G, z_G), using major and minor points as described in (7.3.2) to improve search efficiency. The length $L_{precedence}$ was then measured in front of $P_{nearest}$ to arrive at one endpoint of the centre line segment to be processed, $P_{centreline\ end2}$. A further distance was measured in front of $P_{centreline\ end2}$ up to a maximum length L_{max} to arrive at the other segment endpoint, $P_{centreline\ end1}$. The line segment to be processed was then defined as a point set $S_{centreline}$ containing all centre line pixels lying between endpoints $P_{centreline\ end1}$ and $P_{centreline\ end2}$. This process is now illustrated for the two region types supported by the VRS prototype.

7.4.2) Rectangle Sign Regions

To find the best-fitting rectangle region, a linear-perpendicular regression was used to find the straight line that fits the centre line segment with minimum SSE. This regression is not the same as a standard linear regression in which the error is measured as vertical distance between a point P and the regression line. This is unsuitable in this case because the regression would fail completely when the best-fit line is vertical. Instead, the error is measured as the distance between a point P and the regression line, measured perpendicular to the regression line, as shown in Figure 7.6.

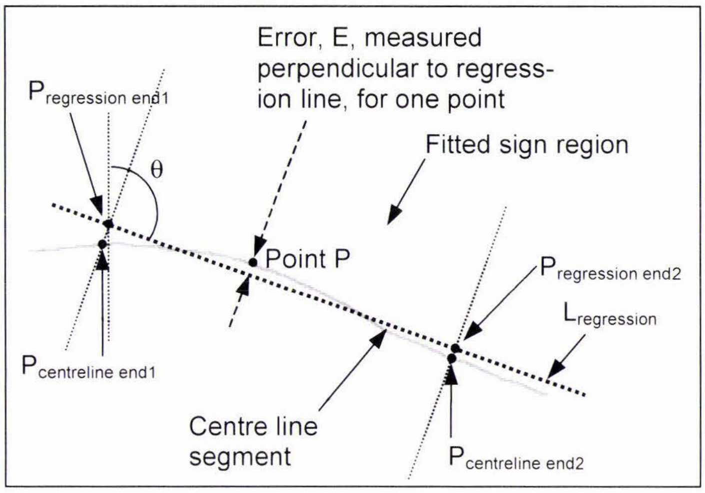

Error, E, measured perpendicular to regress-ion line, for one point
$P_{\text{regression end1}}$
θ
Fitted sign region
Point P
$P_{\text{regression end2}}$
$L_{\text{regression}}$
$P_{\text{centreline end1}}$
Centre line segment
$P_{\text{centreline end2}}$

Figure 7.6: Fitting a computer-generated rectangle sign region to a Ring Road centre line segment

A simple closed-form derivation of this regression is given in (C.3). An infinite two-dimensional line $L_{\text{regression}}$ is represented in terms of the shortest signed distance between it and the origin, k, and angle, θ. Once optimum values of k and θ were found by regression, the line was made finite by locating two endpoints, $P_{\text{regression end1}}$ and $P_{\text{regression end2}}$, which were the closest points on line $L_{\text{regression}}$ to the centre line segment endpoints, $P_{\text{centreline end1}}$ and $P_{\text{centreline end2}}$, respectively, as shown in Figure 7.6. It was then a simple matter of calculating the required sign region parameters for the source world database (as described in [4.2.1]).

7.4.3) Arc Sign Regions

To find the best-fitting arc region, we would like to adapt the linear-perpendicular regression technique so that instead of measuring all errors perpendicular to a straight line oriented at a fixed angle, the error is measured in line with each point P and the arc centre (x_c, z_c). A circular arc is defined in terms of its centre coordinates (x_c, z_c) and radius R (as described in [4.2.1]). In equation form, a finite arc lies on the perimeter of a circle which is defined with respect to parametric variable θ by two equations (see Eq. C.4.1 in [C.4]).

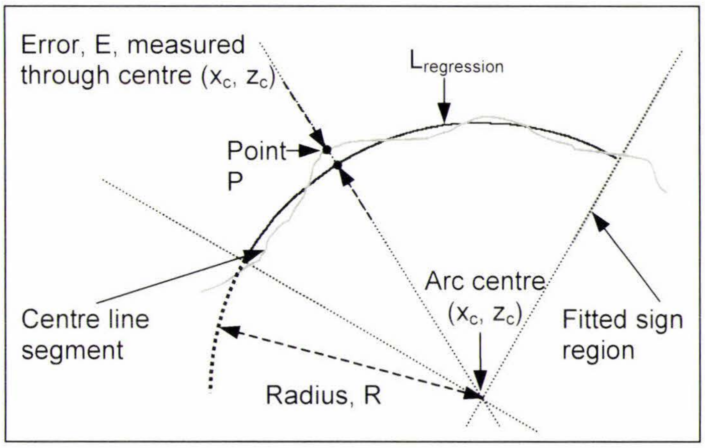


Figure 7.7: Fitting a computer-generated arc sign region to a Ring Road centre line segment

The optimum solution (x_c, z_c, R) found by regression defines that arc that best fits the centre line segment; in other words, that yields the global minimal SSE (see Figure 7.7). Unfortunately, a closed-form solution to the desired regression does not exist. (C.4) contains the non-linear equations resulting from partial differentiation which must be solved simultaneously to yield the optimum solution (x_c, z_c, R) (see Eq. C.4.4-6 in [C.4]). There are various techniques that can be used in such circumstances to try and find a near-optimal solution. One option is Taylor series expansion, whereby the three non-linear equations to solve are approximated as a power series and the contribution of all higher-order terms in the series is taken to be negligible. This permits an approximate closed-form solution to be calculated. Another option is a numerical technique such as the Newton-Raphson algorithm, which can be coded and applied iteratively to find an approximate solution (Page, 1996).

Fortunately, the structure of the four-dimensional space (x_c, z_c, R, SSE) to search permits a fast and robust numerical technique to be applied if the angle spanned by the arc is less than 180 degrees. As shown in (C.4), the three non-linear equations to solve can be re-arranged to solve for R as a function of x_c and z_c. This reduces the dimensionality of the search space to three variables (x_c, z_c, SSE), an important simplification because the resulting three-dimensional space can now be visualised in its entirety. Figure 7.8.a shows a set of data points for which a best-fit arc is sought: their coordinates were calculated by taking a point set that forms a perfect circular arc and adding low-amplitude random noise. Figure 7.8.b shows the corresponding three-dimensional rendered plot of the space (x_c, z_c, SSE).

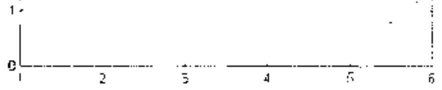


Figure 7.8.a: Point set for which a best-fit arc is sought. Best centre (x_c, z_c) = (0.88, 0.74), radius R≈ 5

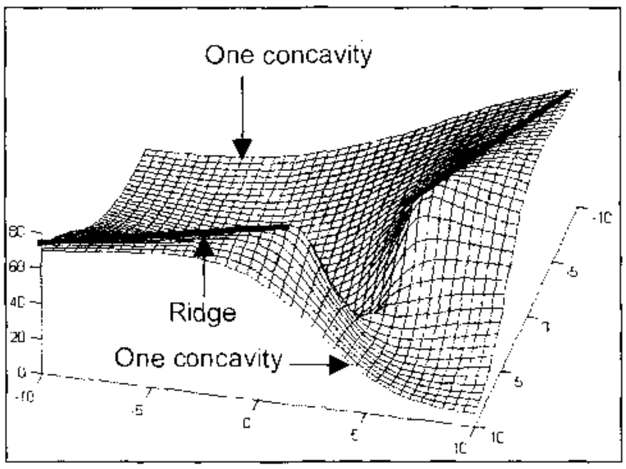


Figure 7.8.b: Corresponding space (x_c, z_c, SSE) for data, showing centre ridge (partly shown as black thickened line) and two neighbouring concavities

If the set of points to regress is sufficiently well-conditioned, such that the optimum arc fits the points with a reasonable error, then the function surface in (x_c, z_c, SSE) is approximately divided into two large concavities separated by a well-defined ridge. At the lowermost point of one concavity lies a local minimum value, which represents the true global minimum SSE value. Given an initial starting point that lies somewhere within one of the two concavities, a steepest-descent iterative method can be used to rapidly converge to the local minimum. Because the primary development environment for the VRS prototype (Microsoft Visual Basic) does not provide built-in optimisation functions, a simple minimisation algorithm was designed and implemented to find the minimum point. The concavity that contains the true global minimum is often well-conditioned in the local neighbourhood of the minimum point; in this region, the algorithm converges with a very high success rate and is largely invariant to the initial starting point. However, in the other convexity, the function may not contain any minimum point; in this region, the algorithm rapidly diverges, failing to converge at all. Details of the arc regression algorithm are given in (G.7).

The question remains of how to choose suitable starting coordinates for the independent variables (x_{c_start}, z_{c_start}) such that the correct concavity is tested. One option is to apply the steepest-descent algorithm in two separate passes, each pass starting in an alternate concavity. In many instances, the ridge separating the two concavities lies along a line that is approximately linear in the x-z plane. That is, if the z value that corresponds to the highest point on the ridge is plotted for each x value, the resulting plot is often nearly linear. Moreover, the parameters k and θ for this line can be robustly estimated by applying a linear-perpendicular regression on the point set to be regressed. This is the same regression technique applied to fit rectangle sign regions, so the algorithm is conveniently reused. Then we simply choose a starting point lying a short distance on one side of the regression line and execute the algorithm. If it is found to be diverging, the algorithm is restarted on the opposite side of the regression line. In fact, in many instances a crude check can tell us on which side of the best-fit line the centre coordinates must lie, avoiding the need to test both concavities.

7.4.4) Region-Fitting Algorithm

Appendix G (G.5) describes a simple algorithm that determines a suitable sign region configuration for each GPS feature coordinate pair. This algorithm chooses either one primitive region (rectangle/arc) per coordinate pair, or two adjacent primitive regions per coordinate pair arranged as one composite region. If a single region can be fitted to the centre line segment adequately, the process stops, otherwise various permutations of rectangle and arc regions are tested until an adequate fit is found. Whether a fit is adequate or not is determined by comparing the RMS regression error to a predefined threshold $E_{threshold}$ at each step. The value of $E_{threshold}$ defines how bad the fit can be before examination of a better region permutation is justified.

If it is found that two regions are justified for a given GPS feature coordinate pair, a binary search is utilised to quickly attempt to find the optimum length of each region. The idea is that there exists some length pair that produces the best combined fit of the two regions and thus minimises the combined RMS error. For simple centre line segments taken around the Ring Road, this is likely to be true. A binary search tries to find where the derivative of the combined RMS error function (slope) is zero by adjusting the lengths of both sign regions while preserving the combined length. It proceeds in large steps at first and gradually decreases as the search is refined, until it converges to an optimal RMS error solution.

7.4.5) Final Set of Ring Road Sign Regions Generated

Figure 7.9 shows the final set of sign regions created for the Ring Road virtual world. These regions were generated for a minimum region length of 0.01 equatorial minutes (≈18 metres), a maximum region length of 0.05 equatorial minutes (≈93 metres), a lane width of 0.0125 equatorial minutes (≈23 metres) and a sign precedence of 0.01 equatorial minutes. As it turned out, the degree of curvature of the Ring Road was small enough to permit a single rectangle sign region to be adequately fitted per Ring Road feature.

In Figure 7.9, all sign regions except the left-most region are of maximum allowed length. The intended methodology was to start with this configuration, execute the VRS prototype, note where conflicts for screen space were generated (as discussed in [6.5]) and reduce the lengths of specific sign regions as necessary to resolve the conflicts. In practice, the conflicts were never actually resolved.

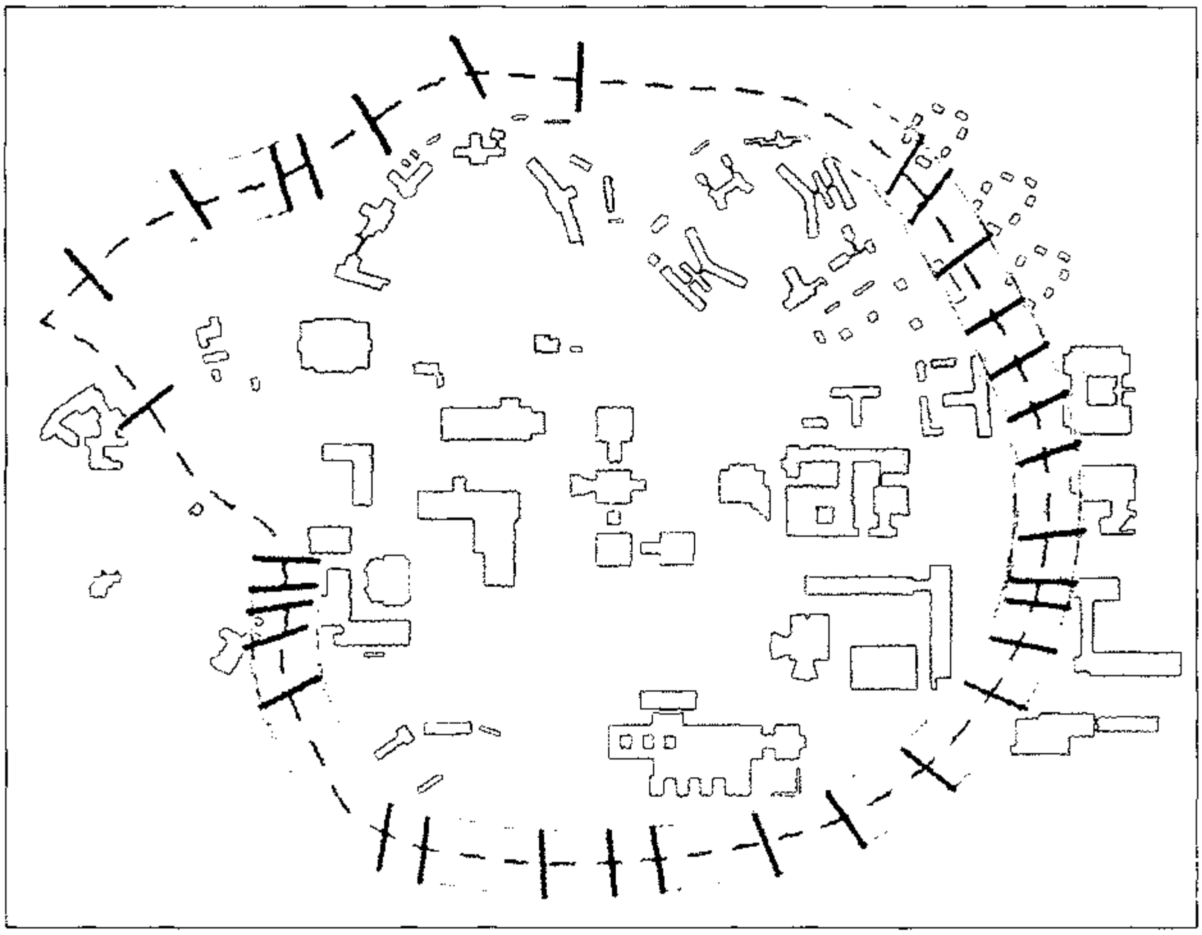


Figure 7.9: Final set of computer-generated sign regions fitted to the Ring Road centre line. Major university buildings are also shown.

7.5) Testing Methodology

GPS data describing the motion of a test vehicle was measured in a separate data collection experiment. A GPS receiver was placed under the windscreen of the test vehicle (to observe the skyward satellites) and connected to a laptop PC. A small data-logging application recorded all GPS parameters as the test vehicle was driven in a clockwise direction around the Ring Road, completing two laps in total. The file was then transferred to the development desktop PC and verified for integrity. Kalman filter parameters were adjusted slightly to refine initial estimates until the RMS predictive error (difference between predicted and measured data) was acceptably small. The log file was then opened in the main VRS prototype and 'replayed' to demonstrate the virtual road signs that would be observed by the driver as he/she drove around the Ring Road. Figure 7.10 shows a pictorial representation of the prototype test architecture:

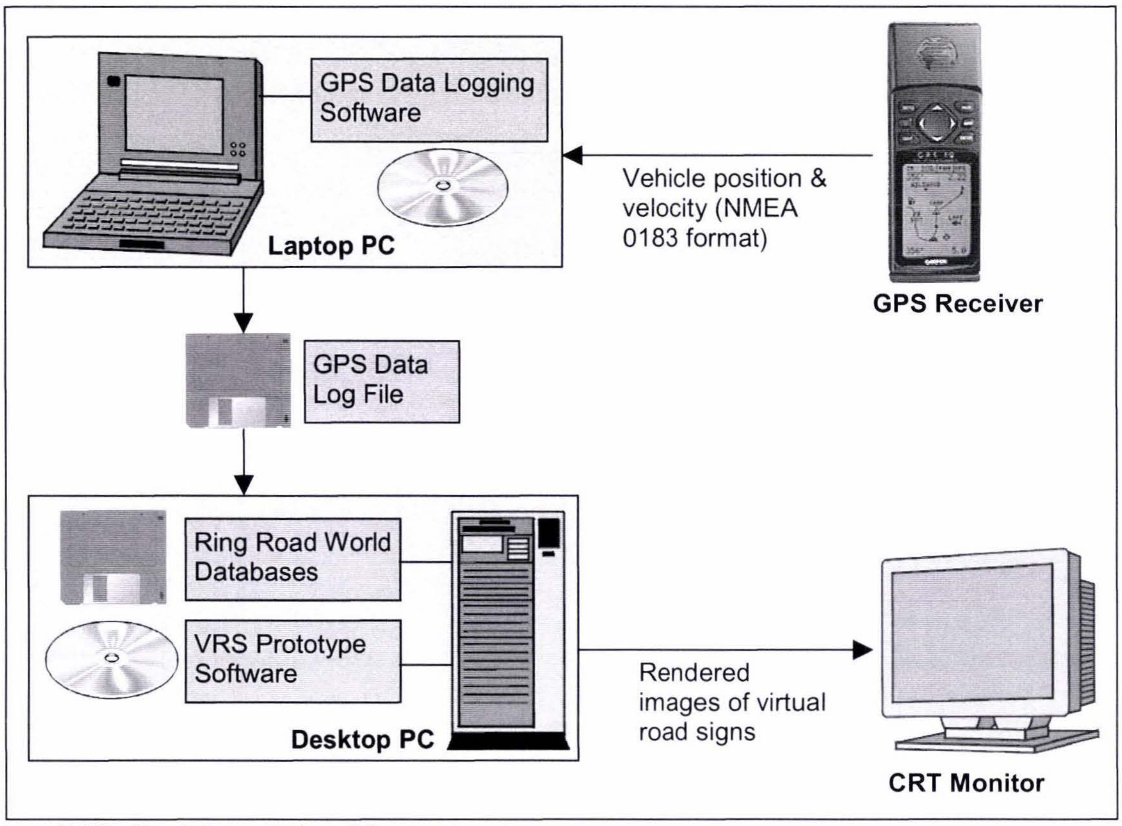

GPS Data Logging Software
Laptop PC
Vehicle position & velocity (NMEA 0183 format)
GPS Receiver
GPS Data Log File
Ring Road World Databases
VRS Prototype Software
Desktop PC
Rendered images of virtual road signs
CRT Monitor

Figure 7.10: Prototype test architecture

The test experiment described in the previous paragraph (experiment 1) was repeated some weeks later under identical test conditions (experiment 2). The purpose of repeating the experiment was to follow up issues of GPS positional accuracy, as shall be discussed in (7.7).

7.6) Prototype Results

7.6.1) Kalman Filter Parameters

For both experiments 1 and 2, the same set of Kalman filter parameters were specified. Eq. 7.1 defines the covariance matrix of measurement vector Y_n. Recall that this matrix describes the accuracy of measured data reported by the GPS receiver.

$$R_n = \begin{bmatrix} \left(\frac{0.85\sigma^*_{HPE,n}}{1850}\right)^2 & 0 & 0 & 0 \\ 0 & \left(\frac{0.85\sigma^*_{HPE,n}}{1850}\right)^2 & 0 & 0 \\ 0 & 0 & \left(\frac{0.2}{3600}\right)^2 & 0 \\ 0 & 0 & 0 & 1^2 \end{bmatrix}$$

Eq. 7.1: Covariance matrix of measurement vector Y_n.
The first two diagonal elements (from top-left) represent the estimated horizontal positional accuracy of longitude and latitude reported by the GPS receiver (in equatorial GPS minutes) respectively, as variances. $\sigma^*_{HPE,n}$ is the estimated HPE value reported by the GPS receiver (in metres) at iteration n. This is a CEP value, as discussed in (3.1.2), whereas matrix R_n expects an RMS value in one dimension only (latitude/ longitude). To convert a CEP HPE value to an RMS one-dimensional value, the CEP value is multiplied by 0.85 (Wilson, 2001 [b]). The third and fourth diagonal elements are estimated RMS accuracy of speed and heading values reported by the GPS receiver, represented as variances. Literature quotes RMS speed accuracy as 0.1 knots (Garmin Corporation, 1999): here the value is doubled as a conservative measure. The value in knots is divided by 3600 to convert it to minutes per second. The RMS accuracy of heading was estimated as one degree.

Eq. 7.2 defines the random system dynamics matrix, Q_n, of the Kalman filter. Recall that each diagonal element in Q_n is sized to cater for unexpected changes in each filtered state.

$$Q_n = \begin{bmatrix} \left(\frac{2}{1850}\right)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \left(\frac{2}{1850}\right)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{2}{1850}\right)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3^2 \end{bmatrix}$$

Eq. 7.2: Covariance matrix of random system dynamics.
The first two diagonal elements (from top-left) are intentionally set non-zero here to account for the non-linear approximation described in (3.5.3). They represent an additional random uncertainty in the reported GPS longitude and latitude of two metres RMS (stated in equatorial GPS minutes). The fourth diagonal element represents a random change in $\dot{s}$ (vehicle acceleration) that may occur between iteration n and (n+1). Here, this is set to 2 ms^{-2} RMS to account for sudden vehicle accelerations in response to judder bars, pedestrian crossings, etc around the Ring Road. The sixth diagonal element represents a random change in $\dot{\theta}$ (rate of change of vehicle heading) between iterations, estimated at three degrees/second RMS to account for small vehicle manoeuvres.

Eq. 7.3 and 7.4 define the starting state of the Kalman filter; that is, the filter state before any vehicle GPS data had been logged. For both experiments 1 and 2, data logging was commenced when the vehicle was parked stationary, and, similarly, ceased when the vehicle had returned to the same parking spot after having completed two laps of the Ring Road. Matrix $X^*_{0,-1}$ contains the estimated starting values of the filtered states, and $S^*_{0,-1}$ contains the estimated variances of these starting values.

$$X^*_{0,-1} = [x_0 \quad z_0 \quad 0 \quad 0 \quad \theta_0 \quad 0]^T, \; S^*_{0,-1} = \begin{bmatrix} \left(\frac{0.85\sigma^*_{HPE,0}}{1850}\right)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \left(\frac{0.85\sigma^*_{HPE,0}}{1850}\right)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Eq. 7.3: Starting filter state matrix at iteration n = 0.

Eq. 7.4: Starting predictor covariance matrix at iteration n = 0.

In Eq. 7.3, the first two elements are set to the first reported longitude and latitude of the vehicle when parked. Because the vehicle was known to be stationary at this time, the speed, change of speed and change of heading in the third, fourth and six elements respectively are set to zero. In Eq. 7.4, the estimated variances of longitude and latitude are based on the HPE values reported by the GPS receiver at iteration n (converted to equatorial GPS minutes), in agreement with the first two diagonal elements of R_n (Eq. 7.1). Similarly, the fifth element is set equal to the estimated variance of heading in agreement with the fourth diagonal element of R_n.

7.6.2) Filtered/Predicted Positional Coordinates

Figure 7.11.a shows the set of GPS positional coordinates logged on the first lap of the Ring Road for the first experiment, and the corresponding filtered and predictive results. Figure 7.11.b shows a magnified version of Figure 7.11.a for one part of the test vehicle's journey. This latter figure highlights examples of how the Kalman filter behaved when the test vehicle accelerated and made sudden unexpected turns. In addition, it shows the effect that bad data recorded at one instance in the vehicle's journey had on the filtered and predicted data.

For comparison, the same data shown in Figure 7.11.b is presented here for an alternative version of Q_n for which the first two diagonal elements are set to zero (Eq. 7.5), shown by Figure 7.12. The resulting Kalman filter does not attempt to correct for the fact that the estimated predicted covariances of coordinates (x, z) specified by $S^*_{n,n-1}$ are underestimated, due to the non-linear sine and cosine terms included in the predictor equations (see [3.5.3]). It can be seen that although the predicted values between measured GPS coordinates visually appear to be better than the predictions made in Figure 7.11.b, the distance between filtered and measured coordinates is unacceptably large.

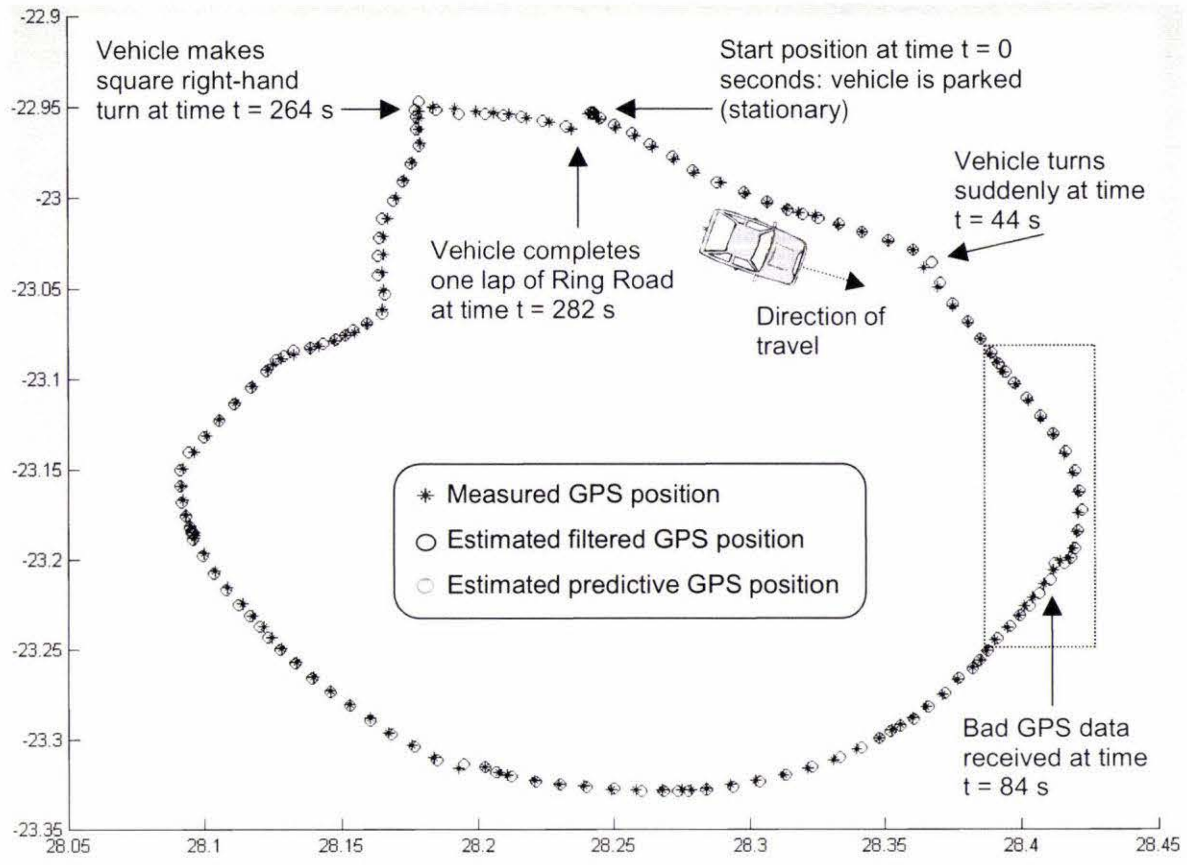


Figure 7.11.a: Measured, Kalman filtered and predictive vehicle GPS coordinates on Ring Road

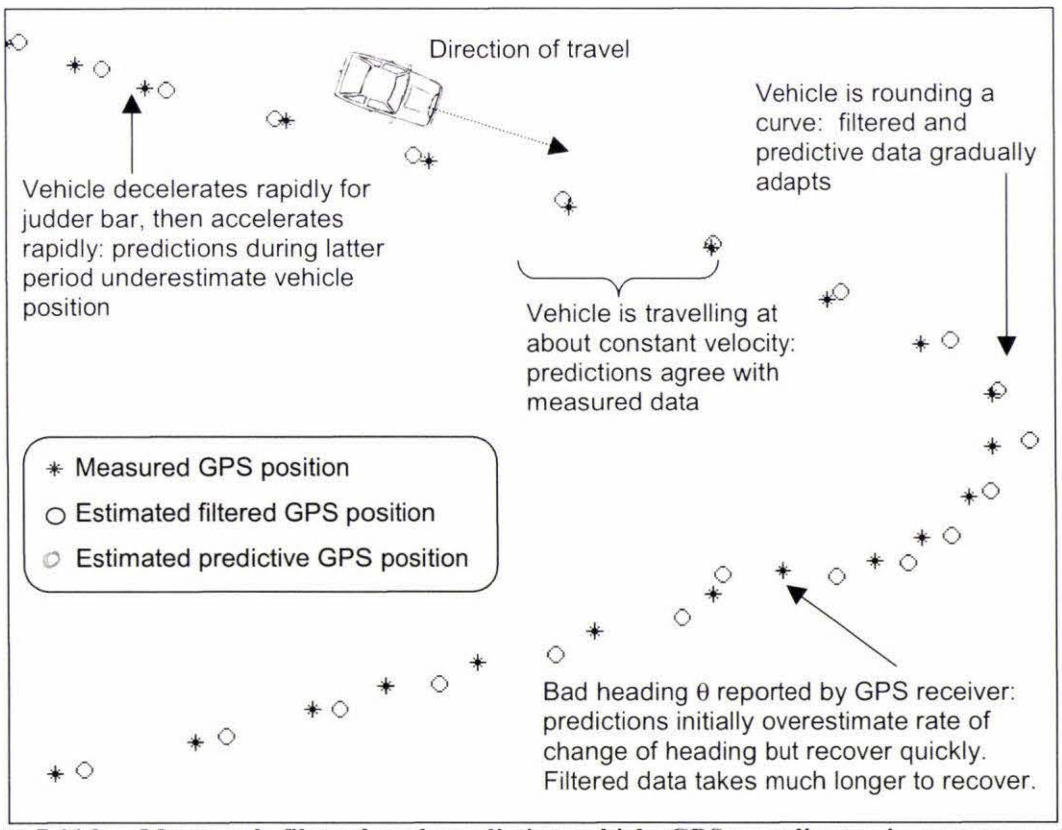


Figure 7.11.b: Measured, filtered and predictive vehicle GPS coordinates in one area of test world (represented by dotted outline in Figure 7.11.a), for matrix Q_n quoted in Eq. 7.2.

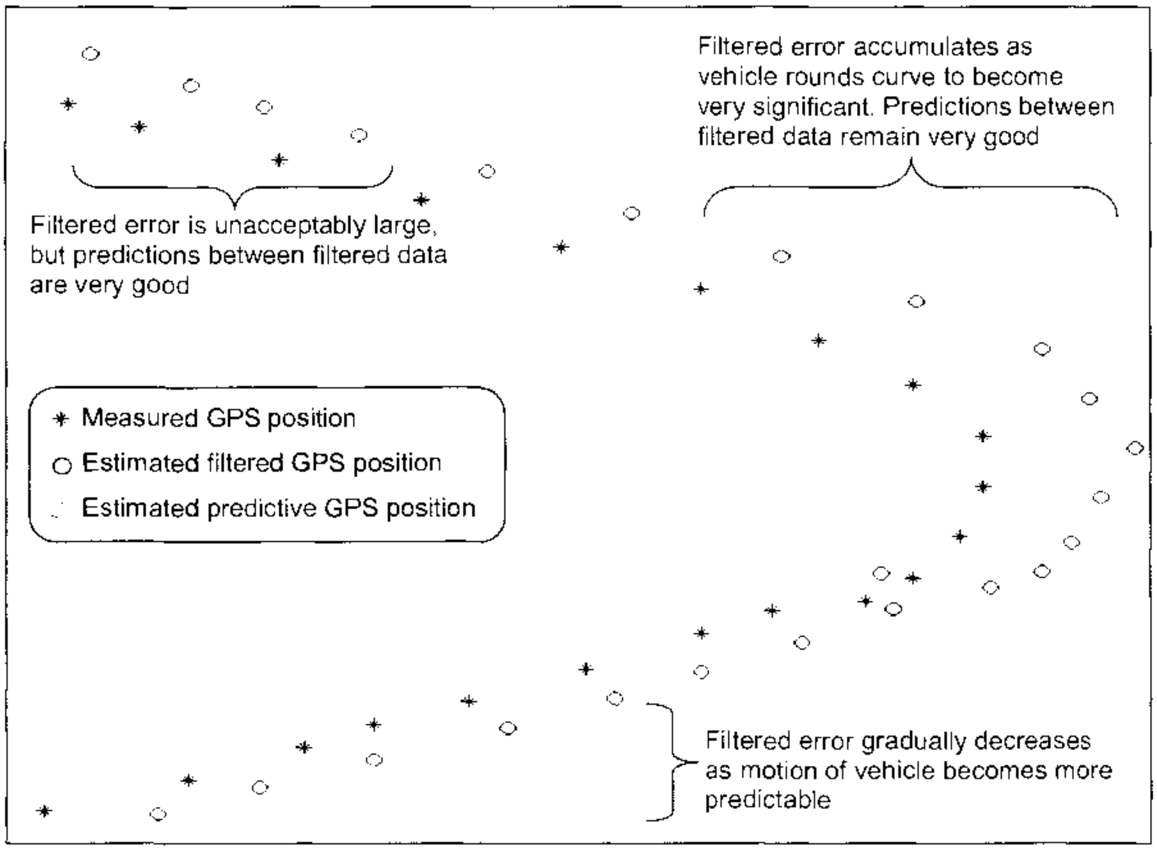


Figure 7.12: Measured, Kalman filtered and predictive vehicle GPS coordinates in same area of test world (represented by dotted outline in Figure 7.11.a, not to scale), for alternative matrix Q_n described by Eq. 7.5 (below). Magnitude of filtered and predictive error (distance between measured data and filtered and predictive data respectively) is unacceptably large in almost all cases.

$$Q_n = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{2}{1850}\right)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3^2 \end{bmatrix}$$

Eq. 7.5: Alternative covariance matrix of random system dynamics.
The only difference between this matrix and that given by Eq. 7.2 is that the first two diagonal elements are zero.

7.6.3) Predictive Errors

Figure 7.13.a shows a graph of predictive error with respect to time for the first lap of experiment 1, where error describes the difference between predicted positional coordinates and measured positional coordinates of the test vehicle. Similarly, Figures 7.13.b-c show graphs of predictive error with respect to time for vehicle speed and heading, respectively.

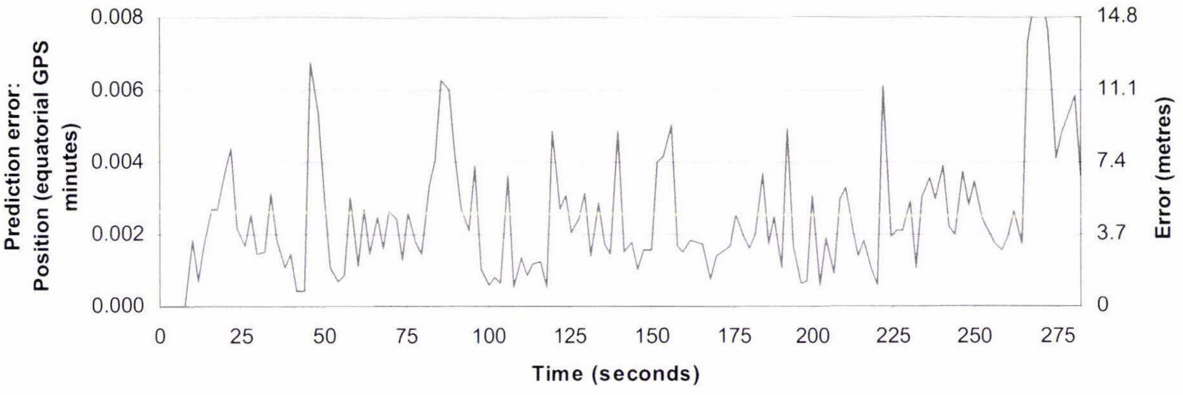


Figure 7.13.a: Graph of predictive error (difference between measurement and prediction) versus time for lap 1 of Ring Road, exp. 1: Position. RMS error = 0.00298 minutes ≈ 5.5 metres.

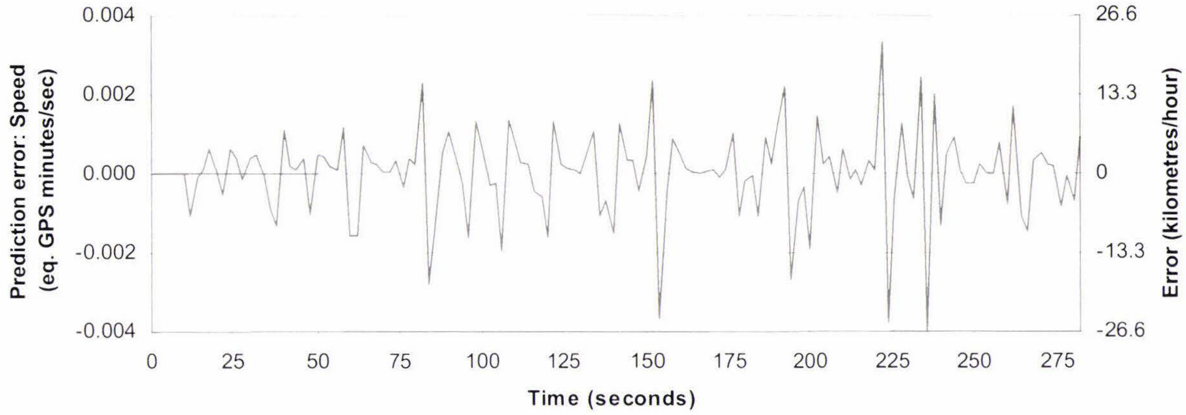


Figure 7.13.b: Graph of predictive error (difference between measurement and prediction) versus time for lap one, exp. 1: Speed. RMS error = 0.00109 minutes/ second ≈ 7.3 kilometres/hour.

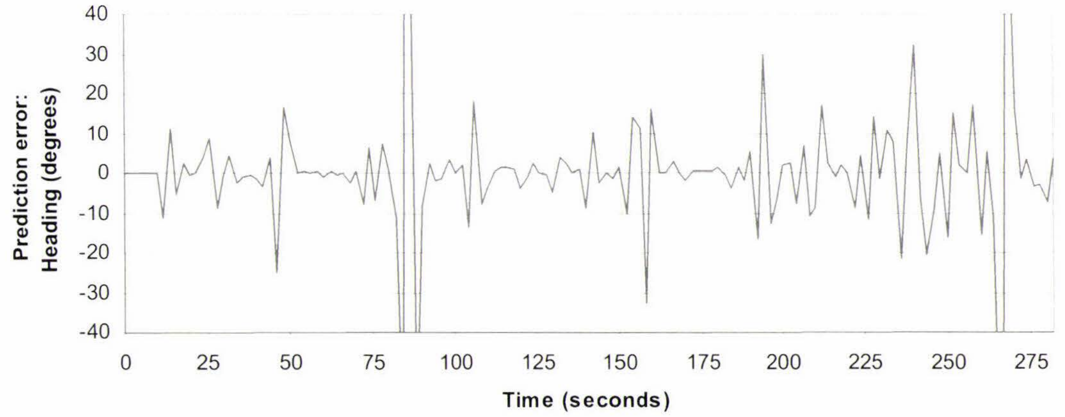


Figure 7.13.c: Graph of predictive error (difference between measurement and prediction) versus time for lap one, exp. 1: Heading. RMS error = 19.5 degrees. When bad data at time = 84 seconds is removed, RMS error = 13.6 degrees. When bad data at time = 84 seconds is removed and vehicle's right-angle turn at time t = 264 seconds is removed, RMS error = 8.5 degrees.

7.6.4) Estimates of Positional Error: Centre Line Error and HPE

One estimator of the error associated with each filtered/ predicted position of the test vehicle is the shortest straight-line distance between the estimated vehicle position (generated by the Kalman filter) and the Ring Road centre line. This measure shall be referred to as the *centre line error*. This should be a valid error estimator method in this case because the precise coordinates of the Ring Road centre line can be calculated from the scale plot mentioned in (7.3.1) and presented in (G.3). Furthermore, the very low minimum RMS error value found in calculating the optimal set of coordinate system transformation parameters in (7.3.3) confirms that the plotted centre line on paper matches the actual real-world centre line configuration very closely.

Figure 7.14.a is a graph of the centre line error at each point in the test vehicle's journey, for both laps of experiment 1. For comparison, Figure 7.14.b shows the values of estimated horizontal positional error (HPE) reported by the GPS receiver for the same two laps. In both figures, the horizontal time scale has been adjusted for the second lap to account for the slightly different time taken to travel the two laps and allow the two lines to be compared. As a result, the errors measured for both laps at a given time approximately correspond to the same location on the Ring Road. Figures 7.15.a and 7.15.b are presented for the data collected during experiment 2, as shown on p. 7-22.

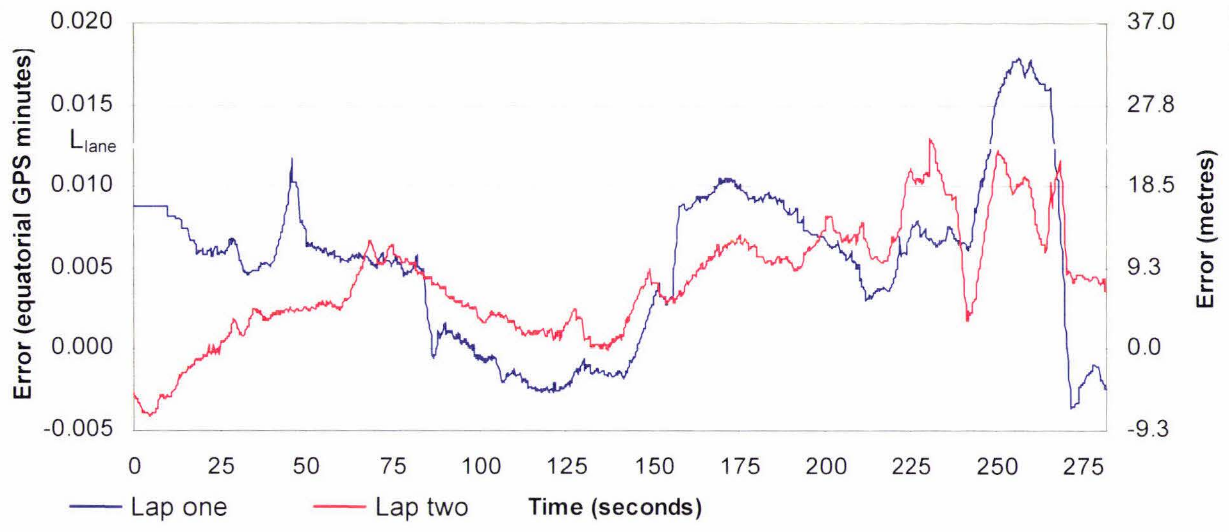


Figure 7.14.a: Graph of positional centre line error (shortest distance between filtered/predicted coordinates and road centre line) versus time, for two consecutive laps of Ring Road, experiment 1. Negative values indicate that the vehicle lay inside the Ring Road centre line; positive values indicate that the vehicle lay outside the centre line.
RMS error for lap 1 = 0.00723 minutes (≈13.4 metres), lap 2 = 0.00541 minutes (≈ 10.0 metres).

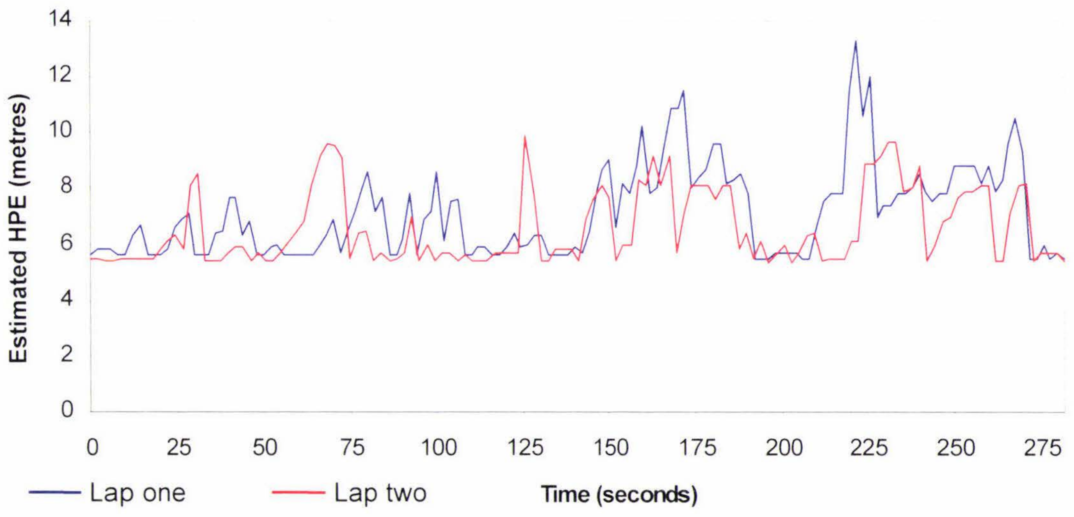


Figure 7.14.b: Graph of estimated HPE reported by GPS receiver versus time, for two consecutive laps of Ring Road, experiment 1.
Lap 1: Mean CEP HPE = 7.1 metres (equivalent RMS HPE = 8.5 metres)
Lap 2: Mean CEP HPE = 6.6 metres (equivalent RMS HPE = 7.9 metres)

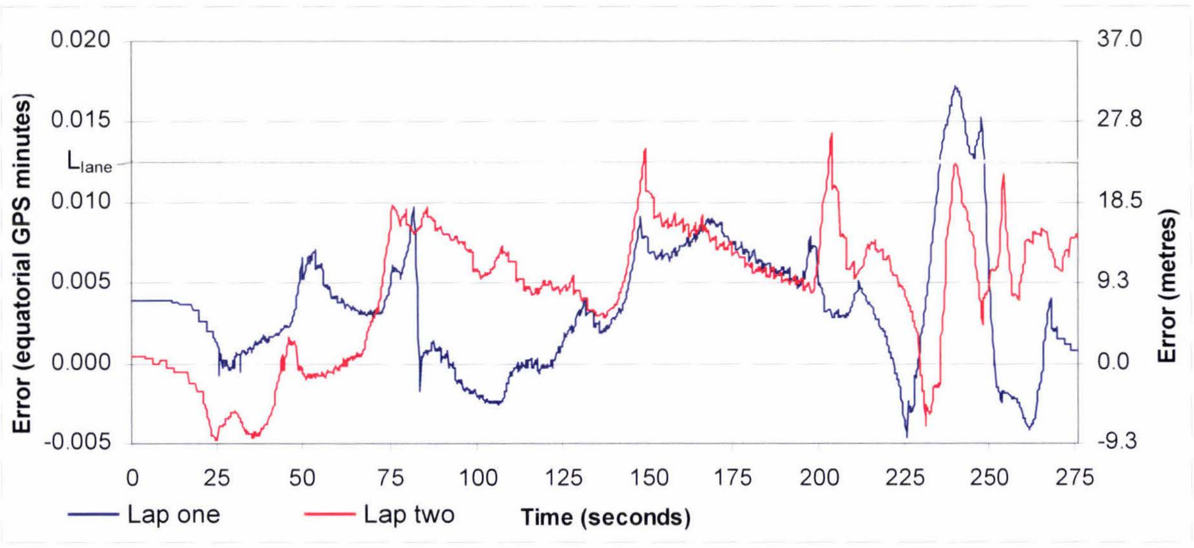


Figure 7.15.a: Graph of positional centre line error (shortest distance between filtered/predicted coordinates and road centre line) versus time for two consecutive laps of Ring Road, experiment 2. Negative values indicate that the vehicle lay inside the Ring Road centre line; positive values indicate that the vehicle lay outside the centre line.
RMS error for lap 1 = 0.00538 minutes (≈ 10.0 metres), lap 2 = 0.00610 minutes (≈11.3 metres).

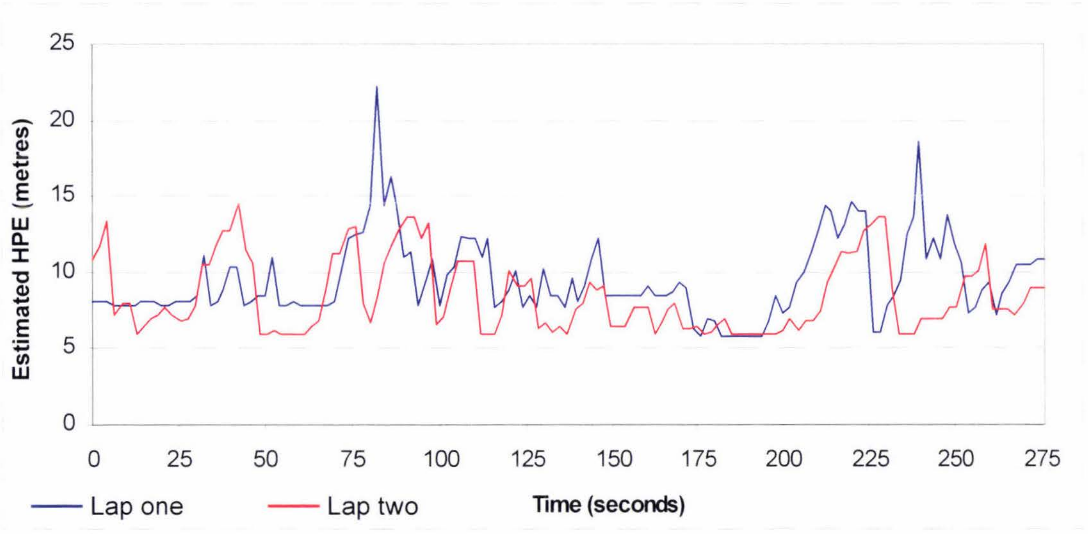
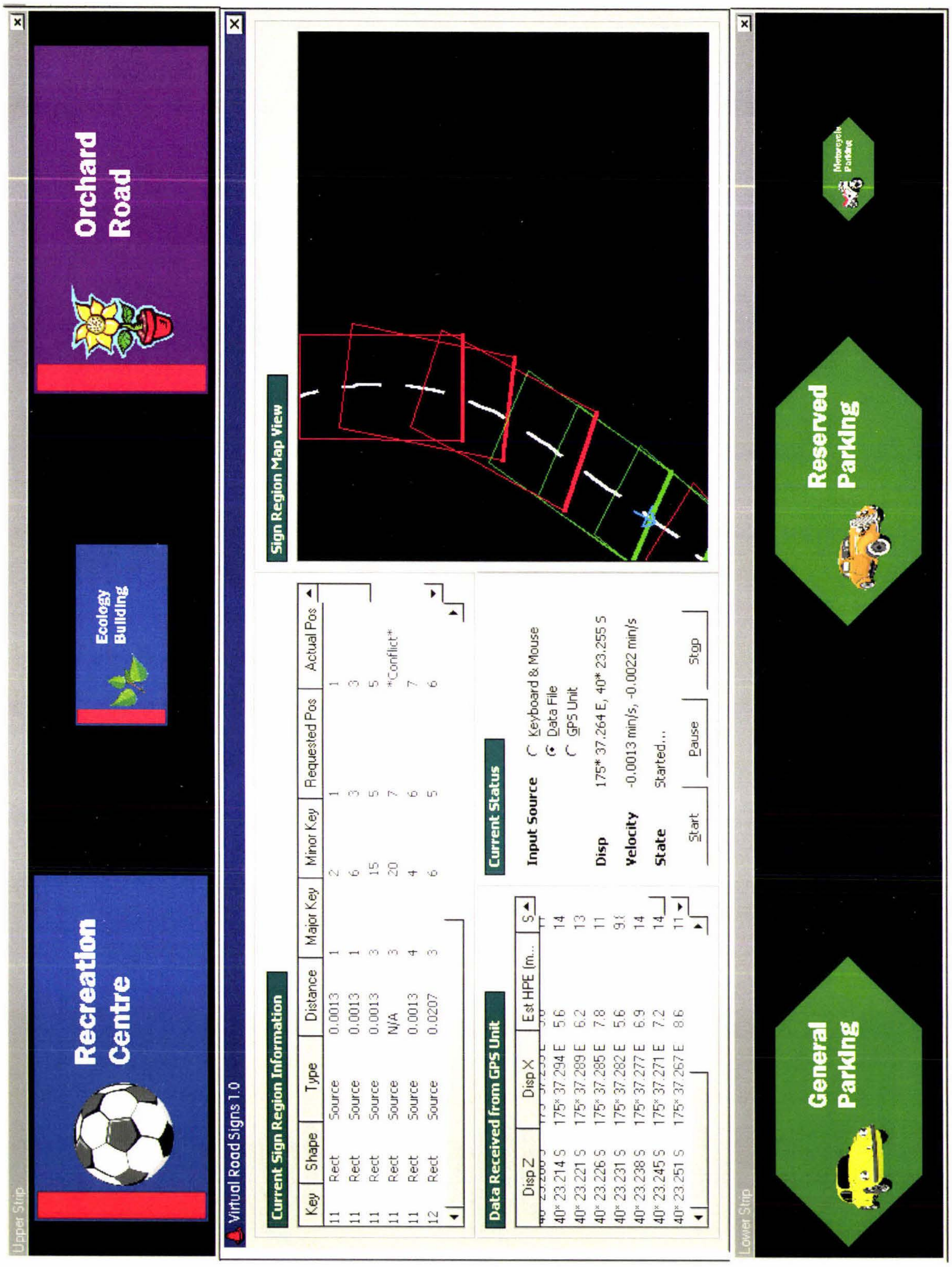


Figure 7.15.b: Graph of estimated HPE reported by GPS receiver versus time, for two consecutive laps of Ring Road, experiment 2.
Lap 1: Mean CEP HPE = 9.5 metres (equivalent RMS HPE = 11.4 metres)
Lap 2: Mean CEP HPE = 8.4 metres (equivalent RMS HPE = 10.1 metres)

Prototype Screen Dump

A screen dump of the prototype in operation is shown in Figure 7.16:



Current Sign Region Information

Key	Shape	Type	Distance	Major Key	Minor Key	Requested Pos	Actual Pos
11	Rect	Source	0.0013	1	2	1	1
11	Rect	Source	0.0013	1	6	3	3
11	Rect	Source	0.0013	3	15	5	5
11	Rect	Source	N/A	3	20	7	*Conflict*
11	Rect	Source	0.0013	4	4	6	7
12	Rect	Source	0.0207	3	6	5	6

Sign Region Map View

Data Received from GPS Unit

Disp Z	Disp X	Est HPE (m...	S
40* 23.214 S	175* 37.294 E	5.6	14
40* 23.221 S	175* 37.289 E	6.2	13
40* 23.226 S	175* 37.285 E	7.8	11
40* 23.231 S	175* 37.282 E	5.6	9.[illegible]
40* 23.238 S	175* 37.277 E	6.9	14
40* 23.245 S	175* 37.271 E	7.2	14
40* 23.251 S	175* 37.267 E	8.6	11

Current Status

Input Source	Keyboard & Mouse [✓ Data File] GPS Unit
Disp	175* 37.264 E, 40* 23.255 S
Velocity	-0.0013 min/s, -0.0022 min/s
State	Started...

Start | Pause | Stop

Figure 7.16: Screen dump of VRS prototype in operation. See (6.2) for a description of each user interface element.

7.7) Discussion of Results

7.7.1) Filtered/Predicted Positional Coordinates

The constant-acceleration Kalman filter proved to be very effective for the task of predicting vehicle position during the vacant two second intervals when vehicle data was not reported by the GPS receiver. In Figure 7.11.b, the adaptive behaviour of the filter is well illustrated. As the test vehicle accelerated after slowing for a judder bar, predicted vehicle positions were underestimated until the next set of GPS data was reported. When the vehicle was travelling at a constant speed and heading, the filter predicted vehicle position with virtually no error, as expected. The filter then adapted gradually as the test vehicle rounded a curve in the road.

The effect of bad data is also clearly illustrated in Figure 7.11.b. At the point shown, a vehicle heading of 279 degrees was reported by the GPS receiver: the previous heading reported two seconds earlier was 200 degrees and the next heading reported two seconds later was 203 degrees. It was not possible that the true test vehicle heading had changed so quickly in this space of time, so the heading of 279 degrees was labelled as bad data. Figure 7.11.b shows nothing untoward in the set of positional predictions immediately after the bad data was received. However, when the next correct heading of 203 degrees was received, the filter interpreted this result as indicating that the vehicle was undergoing a rapid change of heading, as shown by the very tight curve of predicted positions. It was not until the next heading was reported by the GPS receiver that the predicted positions recovered. In contrast, the filtered positional coordinates do not appear to fully recover from the effect of the bad heading until sometime later.

By intentionally setting the first two elements of the random system dynamics matrix Q_n non-zero, the non-linearity of the implemented filter was handled satisfactorily as an approximation. However, comparison of Figures 7.11.b and 7.12 shows that a trade-off exists between the level of damping and the predictive performance of the filter. For the case when the first two elements of Q_n were set to zero, as shown by Eq. 7.5, the observed over-damping seen in Figure 7.12 was anticipated. This is due to the fact that the distance between filtered and measured coordinates tends to accumulate as filtering progresses. In contrast, when the same elements of Q_n were set non-zero (Eq. 7.2), the slight reduction in predictive performance was not anticipated. In practice, it is much more important that the former over-damping is reduced because it is the most significant systematic error. Therefore, the use of non-zero elements in the first two elements of Q_n was well justified.

7.7.2) Predictive Errors

Figures 7.13.a-c illustrate the quantitative predictive performance of the constant-acceleration Kalman filter. In Figure 7.13.a, the frequent spikes in the prediction error can be attributed to the frequent changes in speed of the test vehicle as it navigated around the Ring Road (mainly in response to judder bars and pedestrian crossings), its frequent changes in vehicle heading, and the random measurement error associated with the positional coordinates reported by the GPS receiver. The RMS positional prediction error of 5.5 metres is acceptable: it is well below the quoted accuracy of 15 metres RMS (Garmin Corporation, 1999) and closely agrees with the value found experimentally in (3.2.3). The contribution of random GPS measurement error to the positional prediction error is likely to be small because it is known that the GPS positional error exhibits high temporal coherence, and the Kalman filter adapts much more rapidly than the rate at which the GPS error drifts.

In Figure 7.13.b, the RMS predictive speed error of 7.3 kilometres per hour is deemed acceptable. During the test period, the speed of the test vehicle ranged from approximately 15 kilometres per hour (as it crossed each judder bar) to a consistent top speed of 40 kilometres per hour between judder bars and pedestrian crossings. The changes in vehicle speed in response to these features were sudden and brief, lasting only a few seconds each time. The onset of each deceleration was manifested as a random event to the Kalman filter, so an RMS error of 7.3 kilometres per hour is reasonable. The filter had very little chance to adapt during decelerations, but would have had time to adapt to subsequent vehicle accelerations (which occurred more gradually than decelerations).

In Figure 7.13.c, each spike of magnitude more than about 10 degrees represents a sudden change in vehicle heading as the vehicle navigated around the Ring Road. The most dominant features are the spikes that occurred between 75 and 100 seconds and between 250 and 275 seconds. The former spike represents the filter response to the bad heading reported at time t = 84 seconds, whereas the latter spike represents the filter response to the sudden right-angle turn made by the vehicle near the end of its first lap of the Ring Road (see Figure 7.11.a). In (G.3), the latter event occurs at point 34 marked on Figure G.2. These two spikes were expected for the constant-acceleration Kalman filter. However, the number of smaller spikes and their magnitude is slightly greater than what was expected, given the small degree of curvature of most sections of the Ring Road. If the error spikes at time t = 84 seconds and 264 seconds are not considered, the resulting RMS predictive error is 8.5 degrees, which is still greater than expected. One proposed explanation was that the filter's response to changes of vehicle heading was underdamped; that is, the filter over-reacted to small changes in vehicle heading. This corresponds to the sixth diagonal element of Q_n being too high in Eq. 7.2. Further experimentation showed that the optimum value for this element was near one degree/second (down from three degrees/second); however, the reduction in RMS predictive error of heading was very minor (about one degree) when this value was used. Fortunately, the higher-than-expected RMS error was of no real consequence for the VRS prototype. Because the time between GPS data measurements was short (two seconds) and the maximum vehicle speed was low, even fairly large errors (up to 20 degrees) had a very minor effect on the predicted vehicle position, which is by far the most important predicted data.

7.7.3) Estimates of Positional Error: Centre Line Error and HPE

The graph shown in Figure 7.14.a was not entirely expected. For both laps of the Ring Road, the distance between filtered/ predicted coordinates and the road centre line appears to change fairly slowly. This is in agreement with the high temporal coherence of GPS positional error (Kim and Oh, 2000). Also, most error values are positive, indicating that most filtered/ predicted coordinates lay outside the Ring Road centre line. One might expect this to be so, due to the finite response of the Kalman filter and navigation of the test vehicle around the Ring Road in the outside lane. However, the RMS centre line error of 13.4 metres for lap one is greater than expected, and certainly greater than the RMS error of 4.8 metres found in (3.2). In lap one, the centre line error was less than 17 metres at most times prior to about time t = 250 seconds. Between time t = 250 seconds and the end of the first lap at time t = 282 seconds, the centre line error increased beyond 0.015 minutes ($\approx$27.8 metres). This is a substantial error. In addition, because it exceeds the lane width of each Ring Road (0.0125 minutes), the test vehicle was reported as lying outside the correct sign region during this time. That is, the driver of the test vehicle would not have been presented with the correct set of virtual road signs during this time, had it been possible to render them in real-time.

Comparison of Figure 7.14.a with Figure 7.14.b shows very little correlation between the HPE values estimated by the GPS receiver and the centre line error for lap one. The HPE graph does increase markedly at t = 225 seconds but quickly decreases, so it is not clear that the large centre line error shown in Figure 7.14.a after t = 250 seconds is due to reduced accuracy of positional coordinates reported by the GPS receiver. The mean RMS HPE value of 8.5 metres is 37% less than the RMS centre line error of 13.4 metres quoted above. Having said this, most HPE values lie in range 5-10 metres (CEP) which roughly agrees with the range of most centre line errors noted above (<17 metres) for a 95% confidence interval.

Comparison of the plotted lines for the two laps in Figure 7.14.b suggests no clear correlation in HPE values between laps before t = 150 seconds, but does show a possible correlation after this time. This supports the possibility that the increased HPE values recorded between t = 150 seconds and 200 seconds, and between t = 220 seconds and 275 seconds could be due to environmental elements at two locations around the Ring Road. The plotted line for lap two in Figure 7.14.a supports this theory, with the greatest centre line errors once again reported from time t = 225 seconds onwards. The two lines also show good correlation between times t = 80 seconds and t = 150 seconds when the centre line error decreased for both laps. However, the two plotted lines show little correlation at all other times. The RMS centre line error for lap two was lower than that for lap one at 10.0 metres, but still slightly higher than expected.

In Figure 7.15.a for lap one, experiment 2, the centre line error is very similar to that measured for lap one, experiment 1, in three cases:

1) The dip in error between time t = 80 seconds and 150 seconds.
2) The rise in error between time t = 150 seconds and 200 seconds.
3) The peak in error above 0.015 minutes after time t = 225 seconds.

Thus, the two laps of the different experiments show slight correlation. For both laps of experiment 2, the magnitude of most errors is less than 18 metres. The RMS centre line errors for the two laps are very similar to those values recorded in experiment 1 (10.0 metres and 11.3 metres respectively). The mean RMS HPE values for the two laps are of the same order of magnitude as for experiment 1 and give good agreement with the RMS centre line errors. Also, most HPE values lie in range 5-13 metres (CEP), which roughly agrees with the range of most centre line errors noted above (<18 metres) for a 95% confidence interval. Once again, the estimated HPE values do not appear to be correlated with the centre line error except after time t = 225 seconds, as for experiment 1. However, the centre line errors for the two laps show good correlation from time t = 0 to 80 seconds, and from t = 130 seconds through to near the end of each lap.

Overall, it is unclear with so few experimental results as to why some centre line errors in Figures 7.14.a and 7.15.a were greater than expected. When Figures 7.14.a and 7.15.a were later replotted using only the measured vehicle positions reported by the GPS receiver (not shown), the centre line error magnitudes were virtually unchanged. Thus, the Kalman filter cannot be blamed for the higher-than-expected RMS error (supposing that a poor choice of filter parameters was made). The data logging experiment in (3.2) suggested that one could expect an RMS centre line error of about 5-7 metres for a stationary GPS receiver. The most likely causes for this discrepancy are:

1) Environmental elements situated around the Ring Road, such as nearby trees and buildings, reduced the GPS receiver's positional accuracy. The receiver's view of lower satellites closer to the horizon was obstructed for most of the time as the test vehicle travelled around the Ring Road, forcing the receiver to acquire its signal from other satellites nearly directly overhead. This degraded *satellite-receiver geometry* can be expected to reduce the level of positional accuracy obtained (Drane and Rizos, 1998). Also, nearby buildings can reflect the satellite signal, causing a *multipath* error that further degrades accuracy (Hofmann-Wellenhof *et al.*, 1994; Wormley, 2000).
2) The fact that the GPS receiver was not stationary, and located behind the glass windscreen of the test vehicle.
3) The possibility that the optimal set of coordinate system transformation parameters calculated in (7.3) were optimal only for that set of 34 measured feature coordinates, due to the high temporal coherence of GPS positional error (as suggested in [7.3.3]).

Within each experiment, the high temporal coherence of the GPS positional error makes it impossible to suggest how much of the correlation between errors for the two laps, particularly for experiment 2, was due to this temporal coherence or causes (1)-(3). If the error was purely due to random measurement error of the GPS receiver, we would expect the errors observed between experiments to have been quite different, which was not entirely the case. If (1) or (2) were the dominant causes, we would expect the estimated HPE values reported by the GPS receiver to have been more correlated with the measured centre line error. It is possible that the HPE values simply underestimated the true positional error when the centre line error was particularly high. In particular, cause (1) could certainly explain why the centre line error was significantly elevated near the end of each lap (from t = 225 seconds onwards). The view of the sky was minimised during this time, due to trees and buildings located very close to the Ring Road and, later, dense tree cover. During this time, the receiver's view of lower satellites closer to the horizon was completely obstructed and the view of overhead satellites was poor. One would expect that the direction of the GPS positional error would be random for the same Ring Road locations on two different days. According to this expectation, the measured centre line error should have been highly negative on at least one of the four laps. The fact that the actual centre line error was highly positive on all four laps of the two experiments does not necessarily rule out reduced GPS positional accuracy as a possible contributing factor, as the tree cover could have biased the sign of the error in the same way on all four laps.

It is not likely that the accuracy of the scale paper plot of the Ring Road was responsible for the elevated centre line errors. Industrial surveying equipment was used to survey the Ring Road some years ago, and the process of surveying new areas is ongoing. The measurement error of this equipment was estimated by the provider of the paper plot as less than one metre. The processes of estimating the position of the Ring Road centre line on the paper plot and scanning the paper plot into a PC may have contributed a small, insignificant amount of error. The approximation that a minute of longitude was of a fixed length, as utilised by the equatorial GPS coordinate system, would have contributed a tiny negligible error, as the size of the test world was very small. It is worth noting that the four RMS centre line errors are all less than the value of 15 metres RMS quoted for the receiver used (Garmin Corporation, 1999). Therefore, although the centre line RMS error magnitudes were greater than expected based on the results of (3.2) and (7.3), they were not excessive overall.

7.8) Summary

A small test world was constructed around the main road of the university to test the VRS prototype. A specific user was targeted for this test, and a set of virtual signs were designed to suit them. The set of sign regions was generated with the aid of a computer and a scale plot of the test world area. Before this could be done, it was necessary to calculate the transformation parameters that converted between the three prototype coordinate systems. Techniques for fitting rectangle and arc sign regions were developed, as was an algorithm to fit composite sign regions. However, the final test world contains only rectangle sign regions. The prototype was tested by traversing a pre-chosen test route on two separate occasions. Results show that the Kalman filter performed acceptably in filtering and predicting vehicle data. However, the accuracy of the positional coordinates reported by the GPS receiver was brought into question. The measured positional error was greater than expected in some instances, based on a simple error estimator, and several possible explanations were proposed. Fortunately, the actual errors observed were of no consequence for the purpose of prototype testing. However, in a real VRS system, errors in excess of 15 metres would be considered unacceptable. In the next chapter, the overall design of the VRS prototype is critically reviewed, and issues of implementation for a real VRS system are discussed.

CHAPTER 8
General Discussion

All salient aspects of the VRS prototype are critically reviewed in this chapter, and several issues that would require further attention in a general, larger VRS system are discussed. Some potential applications of virtual road signs are described. The chapter concludes with a description of some opportunities for future research.

8.1) Positioning & Tracking Subsystem

8.1.1) GPS Receiver

A GPS receiver proved to be well suited to the task of measuring vehicle data (position, speed and heading) for the purpose of prototype testing. Its small size meant that it could easily be tucked away at the apex of where the windscreen of the test vehicle met the dashboard. In fact, this position was one of the few possible positions where the GPS satellite signal was of sufficient magnitude to give an acceptable position fix (the signal strength was not sufficient directly beneath the roof at the centre of the test vehicle). At times when the signal strength was weaker, GPS performance gracefully degraded by increasing the numerical value of HPE reported (corresponding to a decreasing accuracy). The fact that vehicle measurements could still be made at low signal strength was not only convenient, but vital around the Ring Road test world due to numerous instances where viewing conditions of the skyward satellites were far from ideal.

8.1.2) Positional Accuracy of GPS

The accuracy and drift of standard GPS with selective availability disabled appear to be marginally satisfactory for this application, in most instances (95%) under reasonable viewing conditions of the sky. Local relative latitudinal and longitudinal accuracy (over short distances) is considerably better than the quoted absolute accuracy of 15 metres RMS, due to relatively high temporal coherence of positional errors. However, the absolute accuracy is a more important measure than short-term accuracy because it dictates the set of sign regions that are associated with a given world position. It is essential that on every occasion, the error is small enough that the same correct set of sign regions is reported. Given then that the correct signs are displayed, the correct observed positions of signs, where 'correct' is defined as lying within specified limits, is a secondary concern. The true absolute RMS positional error appeared to be lower than the quoted value of 15 metres RMS for the data logging experiment described in (3.2) and the main prototype test described in Chapter 7. However, in the absence of a more accurate positioning system, this could not be confirmed.

The gradual drift of positional error over a period of minutes and hours caused the reported position of a stationary receiver to vary considerably (under ideal viewing conditions), with very inaccurate data reported on rare occasions (<5%) (see [3.2.3]). For the real-world Ring Road test, the mean magnitudes of the logged CEP HPE values (shown in Figures 7.14.b and 7.15.b of [7.6.4]) were slightly but not significantly greater (1-4 metres) than those found experimentally for the stationary GPS receiver. The difference between RMS centre line error for the test world and the RMS positional error for the stationary GPS receiver was greater (5-8.5 metres). Thus, a slight-to-moderate level of degradation of mean positional accuracy was observed under real-world test conditions, as predicted. From (7.7.3), there is also evidence to suggest that GPS accuracy will be inadequate when the satellite view is obstructed beyond a certain extent, which can be expected to occur frequently in dense urban environments (Ding, 1999). It was noted in (3.1.2) and (7.4.1) that poor accuracy can be compensated for in most instances in areas where sign regions are not in close proximity to each other, by increasing the width and length of sign regions. However, in areas where higher accuracy is required (for example, to distinguish road lanes, or in areas where sign regions are tightly packed together), a more accurate positioning system will be needed.

8.1.3) Improving Positional Accuracy

Modelling GPS Error

Because the positional error of GPS changes so slowly, it can be modelled as a first- or second-order Gauss-Markov process. A common technique is to use a Kalman filter to estimate the parameters of the Gauss-Markov process and then use knowledge of the error at a given time to improve the positional accuracy. Cooper and Durrant-Whyte (1994) developed a complex filter model for this purpose using GPS fused with inertial dead-reckoning sensors (see below). Kim, Jee and Lee (2000) developed a simpler filter model for GPS fused with a map-matching algorithm to reduce the along-track error that is often unaffected by conventional map-matching. The models were quite effective for improving positional accuracy prior to the disablement of selective availability. However, now that selective availability is no longer imposed, the remaining errors are more random (Cooper and Durrant-Whyte, 1994), and it is unknown whether comparative models can be formulated to improve positional accuracy any further.

GPS Fused with Dead Reckoning

One of the simplest options for improving the positional accuracy of standalone GPS is to augment it with one or more dead-reckoning sensors (see [3.1.3]). A fused GPS-dead reckoning system uses multiple measurement sources to calculate a more accurate position than would be possible using any source by itself (Mattos, 1994). Dead reckoning sensors provide short-term positional data during times when the GPS signal cannot be received, and the GPS signal is used to periodically recalibrate the dead reckoning sensors. *Tightly coupled* systems combine the GPS receiver, dead reckoning sensors and processing capability into one unit.

A magnetic pickup is a cheap, readily available sensor that could be adhered to the chassis of the test vehicle with little effort. This measures the revolution rate of a small magnet embedded in one wheel of the vehicle to yield an accurate distance measurement over short distances. The fused positioning system of GPS and wheel sensor may be able to achieve a five metre short-term accuracy under normal operating conditions. However, the accuracy of a wheel sensor is limited by factors such as changes of tyre pressure and load. Also, it is only effective providing the rotary motion of the sensor directly corresponds to the linear motion of the vehicle over ground. For example, this is not true if the wheel spins up under heavy acceleration or the wheel locks under heavy braking. This effect is aggravated by the fact that the accuracy requirement of a VRS system is highest during the times when wheel spin-up or locking is likely to occur (at intersections). A single-axis gyroscope can sense a change in vehicle heading at right-angle turns (Ding, 1999) which would be important for a VRS system. Overall though, dead reckoning can be expected to reduce the along-track positional error (measured parallel to the vehicle's forward direction of motion), but cannot be expected to reduce the cross-track error (measured perpendicular to the vehicle's motion). Thus, it may still be inadequate to distinguish lanes of a highway.

Differential GPS

Ideally a fixed accuracy of around 1-3 metres is desired for a VRS system to be able to discern individual lanes (or lane groups) of a highway. A standalone GPS receiver is precise enough to report such a distance, but its accuracy is not adequate to discern which lane a vehicle is in with sufficient confidence. One option that could facilitate this is *differential* GPS (DGPS). DGPS utilises a set of land-based reference stations whose spatial coordinates on the earth's surface have been accurately pre-measured (Bretz, 2000). During use of a standard GPS receiver, one or more nearby stations record their apparent coordinates measured by standard GPS. By comparing this data to their known higher-accuracy coordinates, a correction is calculated and broadcast to all GPS receivers equipped to receive DGPS transmissions (Cosentino and Diggle, 1996). The two major drawbacks of DGPS are that it requires an expensive RF radio receiver module to receive the corrections, and its availability is dictated by the presence of nearby DGPS stations (a distance of 100 kilometres is considered to be the limit [Drane and Rizos, 1998]). This is due to the fact that the spatial coherence of positional error must be high between the GPS receiver and transmitting DGPS station to be effective (spatial coherence decreases as this distance increases). In this case, a separate backpack DGPS receiver was found late in the project's development which could have been used for this purpose. However, there was little point seen in doing this, especially considering that it would have been necessary to set up a temporary DGPS station.

To use DGPS as a positioning technique in a VRS system with a wide area of coverage, a substantial number of reference stations would need to be deployed. The high accuracy required to distinguish lanes of a highway is often necessary only in densely populated areas. Thus, one practical alternative may be to deploy one DGPS station per city (or suburb, depending on the coverage area). Each station should be cheaper to maintain than the significant number of conventional signs that would otherwise exist. This solution would be less practical in country areas and towns where a main road passes through, because the low population density may not justify the cost of maintaining a DGPS station. In these cases, a more cost-effective solution may be *wide-area* DGPS, which requires very few reference stations to cover a large area. For example, ten sparsely located stations might be enough to cover the entire United States (Zhao, 1997). Sophisticated wide-area DGPS implementations achieve their accuracy by providing orbital corrections for each satellite and modelling the error due to atmospheric refraction (Cosentino and Diggle, 1996).

Map-Matching

Another option that can be used in conjunction with DGPS or dead-reckoning to improve accuracy is *map-matching*. This technique works on the principle that a vehicle must be on a road (rather than traversing cross-country), and therefore its position can be calculated by reference to its most likely position in the virtual world (Camus and Fortin, 1995). When the vehicle coordinates lie on a road segment as defined by the world map, map matching may have no effect (Drane and Rizos, 1998). In cases when this is not true, however, a map-matching algorithm correlates the shape of the recent path travelled by a vehicle with the shape of nearby road segments and 'snaps' the vehicle path onto the best-fit road segment

(Kim *et al.*, 2000). Assumptions such as the vehicle travelling on the correct side of the road can further refine the 'fitted' coordinates.

High positional accuracy is essential to minimise incorrect road segment selections and thus avoid actually degrading the accuracy of reported positional coordinates. For a given accuracy, configuration of the road network has a strong bearing on the success of map-matching. For rural roads and highways separated by relatively long distances, map-matching will be very effective even when positional accuracy is quite low. For closely-spaced urban streets, a much higher accuracy may be needed for map-matching to be effective. One limitation of map-matching is that although it can reduce cross-track positional error, it cannot significantly reduce the along-track error on long straight roads (Kim *et al.*, 2000). Another drawback is that erroneous coordinates may be reported when a vehicle strays from the set of road segments defined by the world map, perhaps when pulling in to park or entering a minor road. Thus any robust map-matching algorithm must be able to detect when its assumptions are no longer valid. In practice, this limits the realistic use of map-matching in low accuracy conditions to highway use only.

8.1.4) Accuracy of Other GPS Measurements

Speed and heading data reported by the GPS receiver were sufficiently accurate for this application. The speed accuracy of 0.1 knots RMS (Garmin Corporation, 1999) should be more than adequate for any VRS system. In contrast, altitude measurements are considerably less accurate, in the order of 20-30 metres. Fortunately, altitude information was not required for the VRS prototype. Would knowledge of altitude be necessary in a larger, more complex VRS world? Most likely. For example, in large cities near motorways, it is common to find overbridges 'stacked' on top of each other such that for given 2D position coordinates (x, z), a vehicle could lie on any one of a set of roads, each with its own set of sign regions and signs. The high error associated with GPS altitude measurements would possibly require the installation of a separate, more accurate altimeter sensor to distinguish these roads, such as an electronic barometer.

8.1.5) Kalman Filter

The limitation of receiving a set of GPS measurements only once every two seconds was successfully overcome. The constant-acceleration Kalman filter was very effective in predicting GPS data during each two second interval when no updated information was available. The fact that the GPS receiver estimated its own error, in the form of HPE, was very convenient because the reported values were simply converted from CEP to RMS (see [3.2]) and supplied to the filter directly as variances (squared RMS). Not only did this mean that no pre-determined estimation was required, but it also meant that the filter could account for the changing accuracy of the positional fix at different points around the Ring Road.

In steady-state when the test vehicle was maintaining a constant velocity around the Ring Road, the degree of error between filtered data values and real data values was negligible. At times when the test vehicle was rounding a curve, the Kalman filter elegantly adapted in synchronisation, yielding an error well within acceptable limits. The error only really became noticeable and significant when the test vehicle braked or accelerated suddenly, which occurred frequently around the Ring Road in response to judder bars and pedestrian crossings (see [7.6.2-3]). In this case, the filter performed sufficiently well that it was not felt necessary to consider an improved filter. However, investigation of a better filter will be justified for a real VRS system. The most needed improvement is a model that can handle sudden non-linear changes in direction of a vehicle (such as at intersections) to yield an acceptable predictive error. Kim and Oh (2000) describe one such system that uses a detection algorithm to detect changes in vehicle motion and a sliding-mode technique to estimate the magnitude of accelerations. They show that this system significantly improves the tracking performance of a Kalman filter when a vehicle makes sudden right-angle turns.

If the rate at which a GPS receiver reports data could be increased, perhaps to 5 Hz or greater, then one could argue that there would be no need for a Kalman filter to predict vehicle position. Unfortunately, increasing the refresh rate is not possible with inexpensive handheld GPS receivers like the one used. Even if it was possible, GPS accuracy and sensitivity would then become the two limiting factors, so it is doubtful whether much improvement would be observed. It is likely that only a larger, more expensive DGPS unit may be capable of offering a significant improvement. Assuming that this is the case, then in a real VRS system, a Kalman filter would still be useful to account for the continuously-varying HPE values estimated by the GPS receiver.

<u>8.1.6) Recommended Improvements</u>

For the VRS prototype, map-matching is considered to be the most worthy improvement in terms of the (small) cost and time involved in its implementation. In (7.7.3), a region of the Ring Road was noted where the correct signs were not displayed at all because the filtered/ predicted vehicle coordinates were reported as lying outside the corresponding sign regions.

In these circumstances, a simple software map-matching algorithm could correctly assume that the vehicle lies in the centre of the left-hand lane at all times.

In a real VRS system, the most cost-effective positioning and tracking subsystem is recommended as a fused combination of DGPS, dead reckoning, map-matching and a Kalman filter. During times when an adequate satellite signal is available, dead reckoning would not be necessary. At times when an adequate position fix cannot be attained, the combination of dead reckoning and map-matching would take over to ensure 100% availability. Over short distances, dead reckoning would reduce the along-track positional error while map-matching would reduce the cross-track error. In other words, the two techniques would compliment each other to yield improved overall positional accuracy. Of course, the success of map-matching relies on certain knowledge that the map to be used for matching is highly accurate, at least as accurate as the positioning technique it augments. Digital road maps are now commonly produced with an accuracy of 3-5 metres or less (Kim *et al.*, 2000), so it should not be difficult to find suitable data for map-matching. Map-matching should also solve the problem of correctly determining the road on which a vehicle is located where two overbridges are stacked vertically, because typically the two overbridges will be oriented perpendicular to each other. During normal conditions, dead reckoning and map-matching would provide a degree of redundancy in those rare instances when bad data is reported by the DGPS receiver.

8.2) Data Subsystem

8.2.1) Low-Level Data Management

Real-time performance and memory requirements of the prototype data subsystem scale very well as the size of the virtual world increases. A region quadtree is an ideal data structure for processing and representing worlds ranging from very small to very large size. On one hand, the fact that it can efficiently represent an entire world in a single structure is advantageous, as node searching and modification tasks are simplified. On the other hand, a quadtree leaf node is an excellent structure for localising sign regions at a low level of detail. Its corresponding quadrant contains sign regions in a natural neighbourhood without excessive pointer duplication.

Top-down adaptive compilation is a simple but elegant technique for representing a virtual world in an efficient form using a quadtree data structure. It automatically chooses the optimal quadrant size to suit the scale and population density of sign regions in specific areas of a world. The only parameter that needs to be specified is the maximum number of sign regions permitted per node quadrant, $n_{\max\ SR}$, before decomposition of that quadrant is justified. The value of $n_{\max\ SR}$ chosen directly determines the space-time trade-off of the quadtree. For a given world, a very small value of $n_{\max\ SR}$ will tend to create a deep quadtree containing nodes at many different levels. This promotes lower space efficiency because the quadtree will necessarily be larger, but higher time efficiency because the number of sign regions to process for the current leaf quadrant (associated with the current vehicle position) is smaller. Conversely, a very high value of $n_{\max\ SR}$ promotes higher space efficiency but lower time efficiency. In practice, the actual value of $n_{\max\ SR}$ chosen is not critical, given modern high-speed processors and abundant secondary storage capacities. By building the quadtree in secondary storage and selectively caching only the search path leading to the current leaf node, only a very small amount of local cache memory is needed.

Table 8.1 gives estimates of the storage requirements for virtual worlds of various sizes; the figures are intended as a rough guide only. This table estimates the number of conventional permanent signs in a real world of each size, and considers the case in which each sign is replaced by a virtual road sign. In a real world, Table 8.1 accounts for multilingual virtual signs displayed to drivers travelling in both directions along highways with potentially many lanes, unlike in the prototype. In a real VRS system, data for each world would be stored in one or more compiled world databases. Many more signs and sign regions may be designed to take advantage of the benefits that virtual road signs bring, such as signs tailored to suit various driver and vehicle categories, and temporary signs according to current driving conditions.

The estimated storage requirements in Table 8.1 are fairly low by modern standards. Modern hard drives offering storage capacities of tens of gigabytes are inexpensive and readily available, so the storage requirements pose no problem even for very large virtual worlds. The byte values quoted in Table 8.1 are conservative estimates and probably do not fully take into account all database overheads. In addition, other information must be stored in a world database that is not considered in Table 8.1. The most notable example is specification of the mappings between signs and sign regions; that is, specification of which signs are associated with which sign regions.

	World				
	Ring Road Test World	**Large City**	**Province/ Small State**	**Small Country (e.g. New Zealand)**	**Large Country (e.g. USA)**
Purpose	Prototype testing	End-user	End-user	End-user	End-user
Approximate area (square kilometres)	0.5-1.0	2,000	25,000	250,000	10 million
Number of sign regions	33	80,000	200,000	1 million	40 million
Number of signs displayed	67	160,000	400,000	2 million	80 million
Number of unique signs	43	80,000	200,000	1 million	40 million
Number of unique images	29	4,000	10,000	100,000	100,000
Approximate space required to represent sign regions (assuming an average 128 bytes per region)	4.1 KB	9.8 MB	24 MB	122 MB	4.8 GB
Approximate space required to represent all unique signs, excluding sign images (assuming an average 512 bytes per sign*)	12.6 KB	39 MB	98 MB	488 MB	19 GB
Approximate space required to store unique sign images (assuming an average size of 50,000 bytes per image)	1.4 MB	191 MB	477 MB	4.7 GB	4.7 GB
Total approximate space required to store world entities	1.4 MB	Hundreds of megabytes	Hundreds of mega-bytes, or gigabytes	Several gigabytes	Tens of gigabytes

Table 8.1: Estimated storage requirements of major world entities, for various world sizes. KB = kilobyte = 1,024 bytes. MB = megabyte = 1024 kilobytes. GB = gigabyte = 1024 megabytes. * An average amount of 512 bytes may be sufficient to store sign text in four or more languages.

The number of images stored in a world database is the most influential factor in the amount of storage required; a maximum of 100,000 unique images was assumed in Table 8.1. If the decision is made to design images for categories of virtual road sign, then perhaps only a few hundred unique images may be needed. If, however, the decision is made to support one or more custom images per sign (such as for individual facilities, perhaps showing a company logo), then there may be many more than 100,000 unique images in large virtual worlds. For example, many transportation authorities lease space on road signs to businesses adjacent to highways to allow them to advertise their logo to motorists (Smailus *et al.*, 1996). It should be noted that in the VRS prototype, the choice of graphic type (icon, clipart, photograph) per sign image had no effect on the amount of secondary storage required to store it. Every sign image was stored as a fixed-size bitmap, simply because this was the input format accepted by OpenGL for texturing. In a large world containing thousands of signs, a more efficient and resolution-independent format such as a vector image might be desirable to reduce the amount of storage required.

The decision to compile a world database separate to the main world database is sensible. Designing a virtual world of any reasonable size and entering its data into a database is demanding enough already without the prospect of having to convert it into a cryptic, more efficient form first. In a real VRS system, the compiled world database would be stored on in-vehicle secondary storage with a very low real-time processing overhead. By storing only a localised set of data per vehicle, the cost of system hardware is reduced. The caching algorithms described in (4.5.2) and (5.2.3) then provide a lower level of data management that attempts to make optimal use of local memory to store recently/ frequently used data, as the vehicle proceeds along its route. Caching is an important mechanism for relieving the burden of frequent accesses to secondary storage and improving the real-time performance of a VRS system. The caching algorithms are completely flexible, requiring only that the amount of memory available for caching be specified in advance. An in-vehicle processor could reasonably be equipped with many megabytes of random-access memory (RAM), sufficient to cache sign regions, signs and sign images over a range of many square kilometres.

One of the most time-consuming tasks performed in the VRS prototype is sign compilation. This occurs every time a sign whose data is not currently stored in the sign cache is to be displayed. By convention, the prototype does not compile signs until the latest possible time, immediately before they are due to be displayed. For a very large compiled world database of a real VRS system, a more effective sign compilation strategy might be to compile signs in a separate thread of execution. For example, when the quadtree leaf node quadrant corresponding to the current vehicle position changes, a new thread would be created to compile all signs for all sign regions associated with the new leaf node quadrant in the background. The advantage of this *predictive compilation* is that signs are instantly ready for display as necessary, once the compilation thread completes. The disadvantage is that the cache may be loaded with sign data that is never displayed, which is wasteful.

<u>8.2.2) High-Level Data Management</u>

As the size of a virtual world increases, the design decisions associated with high-level data management become more important. For a small world (see Table 8.1), a road authority under the auspices of a city council may be responsible for storing and maintaining one or more source world databases, describing the local virtual world and all sign regions in that area. In a medium-sized world, this responsibility might fall on a provincial ruling body, whereas in a large world such as the land area of the country of New Zealand, the national land transport authority may take on this role. Is it practical to let others have an input into the information conveyed by virtual road signs? Take, for example, the opening scenario of this report in Chapter 1. Is there room for informal messages (such as dinner being served imminently) to be conveyed to individual vehicles, as well as traditionally formal messages? One of the design objectives of a VRS system is to support information dissemination on a per-vehicle basis, but there is no restriction stating that all messages must be strictly road-oriented. In a distributed offboard database architecture, it is conceivable that a set of network access points could be accessible by an entire community. As soon as non-road-oriented information is considered for publication via augmented reality, then the VRS system becomes a general communication medium in its own right. An interesting project that envisages augmented reality as a public communication medium of all types of information is the WorldBoard project (Spohrer, 1999).

One of the major attractions of a VRS system is that it can display dynamic road information in real-time or near real-time. The question is thus: which bodies will be responsible for collecting this dynamic information, and how will this information be consolidated into a useable form? This is an ongoing topic of current ATIS systems. New systems are requiring increasingly sophisticated distributed networks and control systems to collect and relay the increasing volume of real-time information they provide to drivers. Private service providers are increasingly contributing to the collection, evaluation and distribution of road information, in addition to public providers (Behrendt, 2000).

For relatively static road information that changes slowly or is essentially permanent, the absence of real-time requirements somewhat relaxes the data collection process. In the past, communicating dynamic information to the driver using physical signs has been the real problem, regardless of whether the information was known or with what accuracy. There are major time constraints during which time the information is useable, and these constraints must be met if a sign's message is to be effective. For example, there is no point attempting to erect a conventional sign "Animals Crossing" on a rural road if the event of animals crossing takes ten minutes and the event of erecting the sign takes thirty minutes. Typically, information which must be disseminated to drivers very quickly has been broadcast by radio in limited quantities. With a VRS system, the problem of real-time visual communication is addressed for those drivers whose cars are equipped to display virtual signs. The new question to address is: how can a wide range of dynamic information be measured with sufficient accuracy within the required time constraints? A good example is hazard reporting. The Ring Road test case did not report weather hazards because suitable beacons (to measure road surface friction coefficient, temperature, visibility or moisture, for example) were unavailable. Many automated highways currently employ networks of weather beacons to automatically indicate the presence of ice or flooding, with no need for human interaction (American City & County, 1998; Communications News, 2000). The signals are typically collected in a hazard-reporting database, transmitted to vehicles and conveyed as warnings to vehicles approaching the danger zone. One problem is that weather information and notification of other hazards are only disseminated on designated roadways, with little or no support on minor roads. In some current information systems, vehicles receive warnings from each other via a central database (Catling, 1994). For example, if oncoming drivers observe an accident, they can report it via special hazard-reporting roadside beacons to the central database. All vehicles approaching the site then receive the appropriate warning. One challenge with permitting the public to contribute to the dynamic information pool is preventing misuse of the service; that is, ensuring that false alarms and illegitimate information are not posted.

Another important data management issue to address is the method by which information is relayed between those sensors and authorities that collect it, and every moving vehicle. The Ring Road test world is the simplest approach because the source and compiled world databases are completely self-contained and static. That is, there is no real-time offboard communication requirement. However, nothing stays constant forever of course. To be truly effective, a static database stored by a real VRS system would still need to be periodically synchronised with an offboard data registry that provides updated sign and region information. An ideal opportunity for performing this task is when a vehicle is inspected for its Warrant of Fitness, for example. This activity occurs on a regular basis, and the database could be updated in a manner completely transparent to the driver. The drawback is that the length of time between updates would be fairly long (6-12 months). Still, this may be considered adequate for such a static database, and it is certainly more practical than assigning the responsibility of maintaining the onboard world database to drivers themselves. An analogy exists in the PC world in which PC users are reminded to regularly update their antivirus definitions, at least once every three months. In commercial settings, system administrators perform this task; in the automotive world, the analogous person may be head mechanic of a vehicle fleet. In the home environment, no such analogous person exists, and few home PCs can claim to having their antivirus definitions updated at the recommended frequency. The same 'lazy' updates could be expected of VRS world databases if this approach to database maintenance was taken.

Obviously, a static database cannot disseminate real-time dynamic information to drivers, such as notification of a nearby vehicle accident. The true benefits of a VRS system are not enjoyed unless such dynamic information can be communicated, and thus every end-user system will support this capability. Therefore, unlike the Ring Road test world, every end-user system will require every vehicle to be equipped with an onboard communication system, as described in (1.6). If one or more offboard databases are utilised, decisions must be made as to how available bandwidth will be allocated. For example, there should be at least one broadcast channel that is received by all vehicles, and it would be sensible to implement an emergency channel.

The communication channel provides a convenient means of updating onboard databases. This may be done continuously during system operation. Another option is for the onboard processor to transmit a request for the latest version of data available every time the system is initialised (perhaps as frequently as every time the vehicle is started). If the idea of deploying one DGPS station per city described in (8.1.3) was implemented, it might be possible to interleave transmission of real-time information updates with the transmission of DGPS positional corrections. Then each city would be responsible for the upkeep of its positioning subsystem and data subsystem. A simple technique would be to periodically broadcast real-time updates repetitively, much like how teletext is broadcast and received by televisions. The updated information would be received by all VRS-enabled vehicles in the operating area, and each vehicle would update its sign database as appropriate to its needs. Naturally, more dynamic information (such as current traffic conditions) would be transmitted more frequently than less dynamic information. It may still be most cost-effective to update the least dynamic and least critical data in a vehicle's onboard database at the time of receiving its Warrant of Fitness, as described earlier. For example, virtual signs for newly constructed roads might be added to the vehicle's sign database in this manner.

How can the individual needs of all vehicles be catered for? This question continues to be addressed in the design of new ATIS systems. Given that a communication channel exists between vehicles and land-based data, the opportunity exists to distribute the compiled world database between each vehicle and a more centralised data repository to varying degrees. For example, it may be considered more feasible to transmit all database information on-the-fly and create virtual signs as and when they are required, rather than storing a precompiled world and set of virtual signs. This would tend to trade less onboard storage for a higher real-time processing requirement. Such an extreme implementation is perhaps unrealistic. Regardless of the final trade-off decided though, it is likely that some onboard non-volatile storage will always be necessary, if only to cache frequently used or permanent data. This would make efficient use of the scarce total bandwidth available.

8.3) Display Subsystem

8.3.1) Prototype Testing

The only major limitation of the VRS prototype is the lack of an inexpensive projection device to complete the human-machine interface. A laptop PC was an ideal in-vehicle device for data collection and storage because it was small, portable and self-contained (battery-powered). However, the performance of most laptop PCs does not approach that of desktop PCs of the same age. The laptop PC used to collect Ring Road test data was over three years old, considered obsolete in the PC world. By today's standards, the two main limitations were its slow system CPU, and more importantly, lack of 3D video acceleration hardware that could optimise OpenGL rendering. Together, the consequence was that the laptop was simply not fast enough to display virtual road signs in real-time at the required frame rate. Fortunately, it was not difficult to find a modern desktop PC with the required performance at a cost significantly less than any laptop with equivalent specifications. In practice, there was no other reasonable alternative available.

8.3.2) Use of OpenGL as a 3D API

OpenGL proved to be an excellent API for the VRS prototype, even though only a basic subset of its functionality was utilised. It was surprisingly simple to learn, and the World Wide Web was invaluable in providing an abundance of tutorials and reference information. Rendering performance was never going to be a problem for the VRS prototype on the development PC, and the final frame rate of 5-10 fps conveyed sufficiently fluid motion. The alternative option to using an existing 3D API was to code the prototype in terms of low-level two-dimensional primitives supported by the chosen programming language. Given the simple task of rendering virtual signs, this may have been feasible, but in practice there was little point in "reinventing the wheel". In addition, utilising an API like OpenGL provides a layer of hardware abstraction to cater for a variety of display devices. This would be important for a general VRS system, because it is very unlikely that a single common standard for the VRS display subsystem would emerge (see [8.4.6]). In a real VRS system, the choice of rendering API is insignificant, providing the three-dimensional nature of virtual signs can be reproduced and an acceptable frame rate can be achieved. It is likely that dedicated hardware will be utilised to accelerate sign rendering to a frame rate of 10-25 fps, assuming that the display device is capable of such frame rates.

8.3.3) The Search for a Suitable VRS Display Device

As can be expected, a CRT monitor is no substitute for an AR display device, even for demonstration purposes in the lab. Because the image is flat and not presented to the driver in real-time (synchronised with a vehicle's motion), all sense of realism is lost. The signs of the VRS prototype observed on a monitor do not even come close to the effect that was hoped would be possible with a VRS system. A small flat image simply cannot immerse the viewer in the same manner as a true AR display device. Having said this, the lack of a suitable projection device was known when the project began, so the aim was never to try and reproduce the imagined effect of augmented sign images superimposed onto a real-world backdrop. One problem observed with the VRS prototype was that signs did not become readable until quite late, partly due to inadequate screen resolution, but mostly due to the small screen region size allocated to each sign. A true AR display device would need to render signs at a larger maximum size to be practical.

Even if sufficient funds had been available and a suitable HUD found, there are plenty of unanswered questions concerning HUDs that need to be addressed before such a device could be deemed appropriate as a primary VRS display device. Is the reflectivity of a vehicle windscreen adequate to reproduce virtual signs with adequate vibrancy? Current HUDs typically display simple visual information such as a set of digits, straight lines and simple single-colour icons. As such, they may not be capable of providing sufficient contrast, colour depth and resolution to display virtual signs with the required clarity. One would hope that it is simply a matter of time before these deficiencies are overcome with ongoing development, if indeed they do exist, and if a driving force such as virtual road signs could push forward the need for improvements. The other possibility is that these deficiencies are a fundamental limitation of HUD technology, in which case an alternative VRS display device will need to be identified. How readable will signs be, and from what perceived distance? A VRS display device must be robust in this sense to ensure that virtual signs really are visible to a driver even when conventional road signs are not.

A recent development in head-mounted display technology is AR spectacles. These continue the trend of a gradual reduction in size of AR display devices. For example, MicroOptical has developed a pair of spectacles that uses a tiny projection device to reflect a computer-generated image off the eyeglass lens into the eye of the wearer (MicroOptical Corporation, 2001) (see Figure 8.1). The device is lightweight and compact enough to be worn comfortably for hours at a time (Ditlea, 2000). For those people who already wear ordinary spectacles as a vision aid, a clip-on version of the projection device is available. Unlike a HUD, AR spectacles are *visually coupled* displays, which means that the wearer observes the same image regardless of whether they turn their head (Pimentel and Teixeira, 1995). AR spectacles are only slightly larger than conventional spectacles, and in the absence of an augmented projected image, permit the wearer to see the surrounding real world with no apparent distortion. A similar recent development is Microvision's "Virtual Retinal Display" technology which projects virtual images directly onto the human retina (Spohrer, 1999). However, both AR spectacles and this technology are still experimental and prohibitively expensive at present.

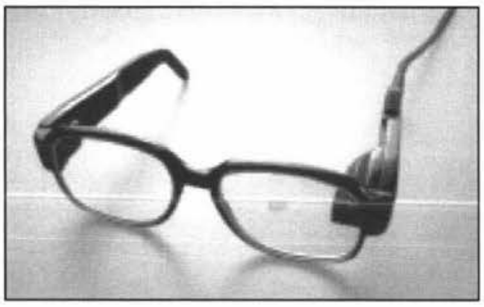


Figure 8.1: AR spectacles developed by MicroOptical Corporation. The projection device that generates the augmented image can be clearly seen.

Another display technology that may be a suitable candidate for virtual sign display in the future is 3D holography (Groves, 1999). Unlike HUDs or AR spectacles, 3D holography would project a true three-dimensional image just in front of a driver, rather than projecting a two-dimensional image directly into the driver's vision. By relying on natural reflectance of the 3D image into the driver's eyes, visual interference may be reduced, and hence holography may be deemed a safer, less immersive option.

8.4) Systems Acceptability

8.4.1) Overview

Public acceptance of a VRS system is crucial. It is not sufficient to thrust new technology at drivers and expect them to use it if it does not meet their needs. If new technology is not accepted by drivers, then market penetration will never occur and the expected benefits will never be reaped. This is particularly true for virtual road signs, where there is already an infrastructure of conventional signs in place that has proved its worth over many decades. The fact that virtual road signs would need to be rolled out in a gradual transition, rather than all at once, and that they would not replace conventional signs completely, may make it very difficult to convince the majority of drivers that virtual road signs are worthwhile. Even if drivers agree that the intended benefits of virtual signs (listed in [1.4]) are realistic on paper, there is a strong temptation on the part of drivers to "stick with what they trust and know works". This effect may be aggravated in two ways:

1) Many drivers will see virtual signs as yet another instance where technology is "taking over" for the worse.
2) The unproven nature of virtual signs, combined with the importance of sign messages being correct, may make drivers uncomfortable about trusting virtual signs.

Providing choice is seen by ITS suppliers as an appropriate mechanism for minimising resistance to the installation of new ITS technology (Catling, 1994). Two activities are necessary: first, suppliers must persuade the relevant authorities to develop the central facilities necessary for service provision to individual drivers. Once this has been done, drivers must be convinced that there are substantial benefits to be had, in terms of their information needs as well as safety, by using virtual road signs.

There are a number of aspects concerning acceptability of a VRS system that can and should be investigated fully before the real-world use of virtual road signs can be seriously contemplated. Most of these aspects apply to the design and operation of ATIS systems in general. A selection is described below.

8.4.2) Safety & Human Factors

The problem of driver distraction and inattentiveness caused by in-vehicle HMIs has been previously discussed. Further research is needed into the use of augmented reality as a primary vehicle HMI display technique, to produce and refine guidelines, rules and recommendations. ECMT (1995) presents a detailed set of such guidelines describing ergonomics and safety as they relate to general in-vehicle information systems. They recommend that at least some form of surveillance, if not regulation, by the governments of participating countries be implemented to ensure safety compliance in the design of in-vehicle HMIs. In addition, the behaviour and reactions of all categories of driver must be evaluated. This is particularly true for older drivers. Gish *et al.* (1999) suggest that HUDs may not be an effective aid for older drivers (older than 55 years) due to their reduced peripheral contrast sensitivity.

Do virtual road signs cause an acceptable level of visual interference to a driver? Is it possible to strike a balance between the level of driver distraction caused by virtual road signs rendered on a HUD and the level of concentration required for safe driving? The VRS prototype displays a lot of visual information at once. In a real VRS system, it may be practical to display a maximum of 3-4 signs at once. In such a system, symbolic, highly contrasting icons will probably be more legible than clipart graphics. Text may need to be made larger relative to each graphic (compared to the VRS prototype) to be effective. It will be sensible to display only the outline of each sign backboard, rather than a solid shape, to prevent excessive obscurement of the driver's forward view. Also, the lower form strip may be reserved entirely for lane marker arrows, which are traditionally painted on the road.

It may well be that a future AR display device will be a more feasible option for displaying virtual road signs inside a vehicle than a HUD. One human problem is the imposition of AR spectacles on drivers. Today, nobody takes the idea of having to wear a pair of custom-designed AR spectacles when driving seriously. It is difficult enough to remind people to wear their seatbelt when driving, let alone wear a pair of spectacles which many drivers may likely find unacceptably intrusive to their vision. Without a major publicity campaign or legal enforcement, such spectacles could end up unused simply because the minor inconvenience of wearing them is thought to make the task of driving unnecessarily difficult. Another potential problem is system malfunction. Erratic behaviour of an LCD in-vehicle display has no immediate effect on a driver. In contrast, if a VRS system starts rendering garbage into a driver's vision, the effect will be extremely distracting and may make it impossible to continue to drive safely. Therefore, a method of safely switching off virtual road signs should be accessible and known to drivers.

A psychological problem associated with a VRS system that may affect drivers is *risk compensation*. If a driver is presented with so much warning information about the situation in front of a vehicle, the sense of danger may be reduced, and the driver may speed up to compensate. In the extreme case, a driver perceives that their information needs are being addressed so effectively that they are willing to place their complete faith in the system. If a driver knows that they will always see a virtual road sign warning them of ice on the road if the road really is icy, then they may sustain a higher speed in normal conditions until they observe such a sign. In the absence of so much information, there is always some element of uncertainty about the road conditions ahead which, hopefully, prevents drivers from travelling faster than they actually do. Unfortunately, there will always be an element of uncertainty that no information system can hope to address; for example, cattle to jump the fence, pedestrians to cross the road, vehicles overtaking on the wrong side of the road. Thus this false sense of security is likely to be dangerous. Research is needed to ascertain the degree to which virtual road signs contribute to this problem.

Because the VRS prototype could not display virtual signs in real-time to the driver of the test vehicle, in the intended manner (superimposed onto the forward view observed by the driver), no conclusion can be reached about the safety of virtual road signs.

8.4.3) Liability

Who is accountable in case of VRS system failure? In this case, failure does not mean a complete inability of the system to operate, but rather the provision of erroneous information or failing to provide information when there is an obligation to provide it (OECD Scientific Experts Group, 1988). The degree of liability attached to such a failure depends on the potential harm to vehicle and the driver. For example, if a hazardous warning sign is displayed to indicate an oil spill on the road ahead when no such spill exists, then the driver will be merely inconvenienced. If a sign indicating roadworks is not displayed where it should be, the potential for real harm to vehicle and driver exists. If a Stop sign is missing at a busy intersection, the potential for harm is life-threatening. For new, unproven technology like virtual road signs, it is even more important to minimise liability in case of failure. This need is probably the single most important factor in limiting the types of information a general VRS system should reasonably be allowed to convey to drivers. For example, it may be considered too risky to replace physical Stop signs and Give Way signs at intersections with virtual signs, because the potential for harm is too great if, for some reason, these signs are ever not displayed. Even taking into account the low rate of theft and destruction of physical signs, most people would argue that physical signs are more reliable at conveying their message than virtual signs. Moreover, physical sign failure (by theft, for example) never conveys an incorrect message, at least not without considerable malicious human interference. In contrast, a VRS system is a complex network of interconnected modules in which there are plenty of opportunities for "things to go wrong". Three research areas minimise the effects of system failure (Catling, 1994, p194):

1) Systems dependability, which aims to develop methods of assessing system reliability and safety checklists.
2) Fault-tolerant systems, which identify safety-critical system components and implement fail-safe and redundant components in the event of failure.
3) Software dependability, which studies failure modes and reliability of software.

The potential for virtual sign failure supports the idea that virtual road signs should only supplement conventional road signs in many instances, and not entirely replace them. In the event that virtual signs cannot be displayed properly and fault-tolerant systems are deemed too expensive, full automatic system shutdown is probably safer than any attempt at graceful degradation.

8.4.4) Private Costs

Only the purchase cost of a VRS system is likely to be significant. Operating and maintenance costs may apply to reimburse dynamic information providers, but would need to be low to convince drivers that the benefits of virtual road signs are worthwhile. An additional maintenance cost may need to be taken into account to periodically update onboard databases. OECD Scientific Experts Group (1988, p52) states that four main factors affecting cost of a general ITS to individuals are:

a) How much of the system intelligence must be built into a vehicle. Cost increases as this proportion increases.
b) The amount of data to be exchanged between onboard and remote equipment. Operating cost will increase as the amount of data increases.
c) The operational safety standard of the system.
d) The luxury rating of the system.

Prototype Costs

For the VRS prototype, the cost of the positioning and tracking subsystem was approximately NZ$500. This cost represents the purchase of the GPS receiver and a PC data interface cable. No dead reckoning sensors were used for the prototype, and the Kalman filter cost nothing to code. The cost of the data subsystem cannot be directly measured because it was part of the two PCs used for testing and development (see [7.5]). Similarly, the cost of the display subsystem cannot be directly measured. The present-day financial worth of the two PCs, including the monitor, was estimated at NZ$4000.

Costs of a General VRS System

Table 8.2 lists the estimated costs of a VRS system. The left-most column lists the value of the various components of the VRS prototype. The middle column lists the estimated component costs of a recommended VRS system in today's currency. The right-most column lists the upper limits of estimated component costs that are considered realistic if the VRS system recommended in the middle column is to be a viable proposition to vehicle owners (on a per-vehicle basis). A real VRS system would not use an in-vehicle laptop PC, but rather utilise dedicated embedded hardware to perform all in-vehicle real-time processing.

Estimated system value for VRS prototype (NZ dollars)*		**Estimated present-day costs of a VRS system (NZ dollars)***		**Estimated costs required for commercial viability (NZ dollars)***
GPS receiver and cable	500	DGPS-enabled receiver	1000	100
		Dead reckoning sensors	Unknown	25
Laptop PC, desktop PC, including monitor	4,000	Hard drive	200	75
		HUD	10,000+	300
		Other e.g. embedded hardware and packing	400	100
Total	4,500	Total	>10,000	600

Table 8.2: Value of VRS prototype components, current costs and viable costs of a general VRS system. * At the time of writing, one New Zealand dollar was worth 0.44 US dollars.

It is clearly evident that the present-day cost of a VRS system is exorbitant, mainly due to the very high cost of a HUD or similar display device. There are good indications that GPS receivers will continue to decrease in price as new applications for GPS continue to be found, especially now that selective availability has been disabled. Secondary storage with tens of gigabytes of storage can now be bought for not much more than the figure of $75 listed in Table 8.2. It is very likely that the cost of the display subsystem will be the biggest hurdle in making the total system cost acceptable to consumers, simply because it is the most state-of-the-art component. No consumer versions of HUDs were found for the purposes of this study; in comparison, consumer GPS receivers have been readily available for years, and data storage and signal processing components have been evolving in consumer electronics for decades. One way of helping to fund VRS system deployment might be to lease space on virtual signs for sponsorship and general advertising.

8.4.5) Personal Privacy

A VRS system, like any ATIS system, can provide services which are increasingly tailored to individual requirements. The position of a vehicle must be measured accurately as part of normal operation. However, the fact that this means vehicle movements can be tracked and monitored may seriously infringe upon personal privacy. The possibility of using vehicle data for surveillance, enforcement (for example, of speed) and road pricing is an issue with any ITS system. With a VRS system, it may be necessary to uniquely identify vehicles and transmit and receive specific data relevant to the individual circumstances of each vehicle and driver, in order to tailor signs adequately. However, there should be no need to know the identity of users or more personal user information. Users should also be able to trust that information describing their personal driving behaviour (such as travelled routes and services sought), which may be stored by onboard or offboard databases, will remain confidential at all times. Bodies such as marketing groups could stand to gain from knowing this information, and the in-vehicle communication system would make it very easy to transmit in a silent manner completely oblivious to the driver.

A VRS system may have implications in accident investigation from both a legal and scientific point of view (OECD Scientific Experts Group, 1988). If a driver sustains an accident, should certain authorities such as the Police have access to VRS log files? If so, the information provided by a VRS system might be argued as legally binding. For example, if it is proven that a driver observed a specific virtual sign, is he/she more liable than a driver who, in different circumstances, saw no such sign?

8.4.6) Standards and System Compatibility

Car buyers will pay a premium for a VRS-equipped vehicle of the future. As for current ATIS systems, vehicle owners will want to buy interoperable equipment that will function properly regardless of the state or city in which the vehicle is driven (Whelan, 1995). Manufacturers may be unable to produce such equipment economically unless a set of standards is prepared and complied with. The modular structure of a VRS system in terms of its three distinct subsystems increases the importance of system components, perhaps manufactured by a range of vendors, being compatible with each other. This will require standard algorithms and component interfaces to be defined. While a VRS system will require preparation of its own set of standards, primarily concerning the safety and effectiveness of the display subsystem, it must also attempt to maximise compatibility with existing ITS systems by adopting existing standards where applicable. In order not to block future developments, standards should be modular in design, allowing new modules or solutions to be added as development continues (Catling, 1994). Standard design practices for safety, especially for certification procedures regarding conformance to standards, quality assurance and reliability, are needed. The principal standards group for human-machine interfaces of in-vehicle systems is ISO/TC22/SC13 WG8, which has active members in nine countries and addresses the standardisation of topics such as the definition of 'acceptable' readability of in-vehicle HMIs (Stevens, 2000).

Camus and Fortin (1995) recommend that the following technical elements be standardised for a general ITS:

a) Vehicle tracking system. GPS is an ideal candidate because it is its own standard.
b) Data dictionary and message structure; for example, format of digital map databases.
c) Ground-vehicle communication protocols and frequency allocation.

ATIS information providers must be capable of communicating with each other to guarantee continuity of service (Camus and Fortin, 1995). Data collectors, operators and users need to know the extent to which those who compile navigation and en-route information retain some control over the form in which it is disseminated. It would make sense that in some circumstances, such as in the event of an emergency, control of data reverts back to public authorities so that they have full responsibility for the information that is disclosed.

<u>8.4.7) Acceptability to Traffic Authorities, Governing Bodies and Information Providers</u>

The final decision as to whether the benefits of VRS systems outweigh the costs will be made by the responsible authorities concerned, not the drivers who will use such systems. Several factors will have a major influence on their decision to introduce a VRS system (OECD Scientific Experts Group, 1988, p48):

a) Feasibility of staged introduction, staggered in space and time.
b) Compatibility with existing infrastructure, and upward compatibility.
c) Ability to positively affect traffic; for example, to reduce congestion. In comparison, traffic authorities may have little interest in the VRS provision of en-route trip information.
d) Possibility of using the system in cases of emergency. Virtual road signs are well suited to emergency notification (for example, of a head-on collision between vehicles) and

even civil emergencies (such as after a major earthquake), assuming that the underlying infrastructure remains intact.

e) Accessibility to the majority of road users. System cost will be one factor that restricts accessibility to road users. Subsidisation by authorities may help.

f) Reliability and ability of alternative navigation and information systems, or even a competitive transport mode (for example, rail).

g) Ability to regulate the type of information conveyed to drivers. The decision to support more general VRS content such as advertising must be carefully considered.

8.5) Potential VRS Applications

The most ambitious application of virtual road signs is supplementation of the existing erected sign infrastructure. In practice, however, large-scale rollout is impractical to expect within ten years. A more realistic aim at this stage is to identify specific sectors that may benefit from the application of virtual road signs:

8.5.1) Bilingual Signage

Recently, the topic of bilingual signage has received much attention in New Zealand (The Dominion, 1999; The Dominion, 2000 [a-b]; The Press, 2000). In one extreme, people have called for all English placenames on existing road and information signs to be supplemented with their Maori equivalents. Virtual road signs would seem an ideal way of achieving this in practice when they become technically feasible. The VRS prototype specifically recognises that signs should be displayed in one of many languages depending on the user. In a multi-language VRS system, the user would select their preferred language(s) at run-time, and only signs in the chosen language(s) would be displayed.

8.5.2) Rental Cars and Tourism

Automated navigation systems that target tourists are increasing in popularity. One current system plays a cassette tape in a tape deck, a common car audio component, to describe local landmarks of interest as the vehicle passes by. In the same way, a set of virtual road signs could be tailored toward the needs of tourists. Obviously the composition of such signs would be quite different to the set of signs designed for local use. In addition to navigation information, signs may point out local attractions such as theme parks, hotels, shopping malls and museums. For overseas tourists, signs could easily be made readable in their own language in the same way described as in (8.5.1) above. The logical site to install such a system is in rental cars, as many overseas tourists who want their own independent means of travel hire private cars. In a simple system, no real-time communication facility would be required as all local tourist attractions could be stored in the onboard database of a vehicle. This database would be updated as necessary when the rental cars were returned to base.

At present, rental car companies charge their fees based on the distance travelled and number of days a vehicle is hired. A more advanced charging system may also base its calculation on the location and time of day where and when the vehicle is driven. The positioning and tracking subsystem would measure the vehicle coordinates and log them to a data file that can be downloaded when the vehicle returns to base. This is similar to a new insurance payment scheme recently unveiled (Bretz, 2000) that calculates insurance premiums based on where and when a vehicle is driven. The system is arguably fairer because it is inevitable that certain geographical areas and times of day present a greater insurance risk than others, so premiums should reflect this difference.

8.5.3) Intelligent Driving Signs

Correct timing of virtual sign presentation is important to maximise sign readability. In one possible VRS system, the software could 'learn' a driver's style of driving by measuring how they accelerate, brake and corner to create a full behavioural profile. Virtual signs could then be adapted to display at an optimum time that suits the driver. In addition, signs could be adapted in real-time if a driver appears to deviate from the stored profile. For example, a much larger Stop sign may be displayed if the driver appears to be braking too late when approaching an intersection. Similarly, if the driver enters an intersection much faster than normal, the Stop sign might appear earlier than usual and/or be replicated.

8.5.4) Rally Car Navigation

One potential application of virtual road signs in a competitive context is rally car driving. Virtual road signs are a specific case of abstract virtual objects. As the method of presenting information is the real novel factor, it would be fairly straightforward to adapt a VRS system to display any virtual objects rather than road signs in particular. One possibility is to project most rally navigation information directly into the driver's field of view, thereby largely relieving the co-driver of navigation duties. In this sense, virtual signs would largely or completely replace the co-driver's paper navigation notes. GPS is already used in rally driving to measure and record the coordinates of cars on the track, so the positioning and tracking subsystem is already implemented. By integrating it with the other two VRS subsystems, navigation details can be displayed at the correct time. The criticality of correct timing is much greater in rally driving than highway driving due to the higher vehicle speeds and tighter routes travelled.

There is some question as to whether visual navigation information would be as effective as the aural information which is currently provided by the co-driver. Also, rally car drivers are already taxed enough by the intense concentration required to keep the vehicle on the track; expecting the driver to read navigation information as well might not be reasonable nor safe. A better option may be to display virtual signs to the co-driver instead who then relays the navigation information to the driver aurally.

8.5.5) Skiing and Mountain Biking

Even greater opportunities exist for virtual signs where there are currently few or no physical signs utilised. Ski runs are frequently subject to extremely poor visibility and dangers such as cliffs. A basic portable virtual sign system could indicate dangers and suggested routes to a skier in all weather conditions. A more advanced related system could even use short-range imaging to draw the track ahead in whiteout conditions, dramatically improving skiing safety. Virtual sign navigation could keep mountain bikers on track without the need for a cumbersome paper map.

8.5.6) Alternate Modes of Transport

This report describes the use of augmented reality to present virtual signs in the context of road vehicles. However, there is no reason why the idea of virtual signs cannot be extended to other modes of transport. For example, airports could use virtual signs to direct pilots to the correct runway or terminal. Rockwell Collins Flight Dynamics is developing a synthetic vision upgrade to direct aircraft around airports in poor visibility, using a projected image of taxiways and ramp married to a GPS-derived aircraft location (Donoghue and Watson, 2000). The same benefits as for road signs can be expected, such as perfect visibility in all conditions. In fact, airports are a prime candidate for virtual signs:

a) Pilots often land at airports they have never visited before, and thus the airport layout is unfamiliar.
b) Conventional signs cannot be erected on the tarmac where aircraft taxi. Even if they could, the signs would have to be very large to be readable from far away.
c) The number of aircraft converging at very large world airports is so great now that the abilities of air traffic controllers are being stretched to coordinate all aircraft landings.

By augmenting aural commands with virtual signs, time savings could be achieved.

Another potential application of virtual signs is in trains. Many sections of train track are remote and it is difficult or impractical to maintain conventional signs at these locations. Virtual signs would be observed by a train driver in the same way as for road users.

8.6) Future Research

The objectives of this research described in (1.5) were met. The technical implementation of two of the three major subsystems of a VRS system (the positioning and data subsystems) is feasible using current technology. Only implementation of the display subsystem at a reasonable cost is infeasible at the present time.

This project has only investigated specific technical elements of a generic virtual road sign system. Further large-scale technical design decisions have yet to be evaluated, particularly with respect to high-level data management. The next step in development of a VRS prototype is to develop much larger test worlds than the Ring Road test case to fully study some of the ideas coded but not yet tested. For example, arc sign regions were not tested in a real-world test, nor were composite sign regions (see [4.2]). An in-vehicle communication system must be installed to permit dynamic information to be transmitted from a remote data store (on a very small scale for development purposes) and displayed on virtual road signs in real-time. A more accurate, low-cost positioning system needs to be developed, such as that described in (8.1.6), to achieve the accuracy that is realistically required for a real VRS system. A more advanced Kalman filter, like that described in (8.1.5) by Kim and Oh (2000), should be implemented.

A more fundamental topic that needs to be addressed is the provision of dynamic information that will be displayed on virtual signs. How will it be collected in sufficient volume, who will be responsible for collection and how will the combined efforts be networked together into a useable information source?

To be fair, the technical design decisions of a VRS system are straightforward and only of minor concern in comparison to the social, economic and political implications. The potential range of applications for virtual signs as a communication medium is huge; it is conceivable that this more natural user interface could one day replace the HMI currently used in many existing ATIS navigation systems. If virtual road signs are to progress beyond the lab and into real-world use, however, then they must be accepted by the public and authorities. Therefore, being proceeding further, issues of systems acceptability such as those discussed in (8.4) should be researched to gauge whether the idea of virtual road signs is acceptable and worthy of further development.

More research needs to be conducted to measure the safety and effectiveness of augmented reality as a communication medium for in-vehicle information presentation. One fundamental question that is yet to be answered is whether the visual interference caused by virtual road signs is acceptable to drivers. Before this can be done, however, an AR display device needs to be found at reasonable cost. The display device is the most critical component of a VRS system that will determine whether virtual road signs are feasible or not. Therefore, there seems little point in continuing the research if a suitable display device cannot be found and subsequently tested in a prototype similar to that described in this report.

The next step would be full public simulation testing of a VRS system to determine how effective virtual road signs really are for reducing driver distraction and inattention. Real drivers must be surveyed for their views. Do they find virtual road signs to be better at conveying information while driving than current systems, such as in-vehicle LCDs? How useful is it in practice to observe signs tailored to the needs of individual drivers and their vehicles? How do virtual road signs affect their driving habits? One problem that may be associated with virtual road signs is *inconsistency*. Virtual road signs take the idea of personalised dynamic information one step further than many current ATIS systems by attempting to cater for the individual needs of drivers. But is this really what drivers want? Conventional signs consistently display the same information all the time. A good example is recommended speeds around curves. If these values are dynamically adapted to the current road conditions, how will drivers react? At the very least, it may make it more difficult for a driver to remember a suitable speed to approach the corner at. Another question is whether the public and authorities feel it is wise to try and adapt as much information as possible to the specific driver-vehicle pairing. Are variable speed limits a good idea? Or is a fixed speed limit for all vehicles safer and less confusing? The success of such adaptive information relies on the acceptance by drivers that the system adapts the information optimally to suit their own circumstances. This is in stark contrast to the current practice, whereby drivers receive static information that applies to all road users and then make their own judgement.

CHAPTER 9
Conclusion

Conventional roadside-erected signs are subject to a number of problems inherent in their design, such as high costs of construction and maintenance, poor visibility and increasing vandalism. In addition, most signs are unsuitable for conveying truly dynamic information. Modern advanced traveller information systems (ATIS) address these problems to an extent by displaying the information traditionally communicated by road signs on a dynamic display inside a vehicle. However, there are still a number of improvements that can and should be made to the in-vehicle human-machine interfaces that present this data. Many current in-vehicle displays force drivers to frequently divert their attention away from the immediate road situation for unacceptably long time periods to read the display. They can also be an annoying distraction.

Virtual road signs may be capable of improving driver safety by presenting information that is more naturally readable. The presentation style of virtual road signs is quite different to current ATIS systems. Because they do not actually exist on the roadside, virtual road signs are, in theory, capable of solving many of the inadequacies of conventional signs. When coupled with the infrastructure of existing navigation systems, virtual road signs are ideal candidates for displaying navigation and other en-route trip information that may be of interest to drivers. Moreover, they introduce many new exciting opportunities for meeting the information needs of drivers, such as tailoring sign messages to specific categories of drivers and their vehicles.

Development of a software prototype was very successful in achieving three of the project objectives. The prototype permitted a general software architecture to be designed and implemented that is applicable to a wide range of possible VRS system designs. At the same time, the prototype served as an ideal vehicle to test and evaluate a number of different design alternatives and optimisations to discover what was and was not practical, or even possible. In addition, the prototype made effective use of the limited resources available to demonstrate how one VRS system might operate.

Development of the VRS prototype was simplified by dividing a general VRS system into three distinct subsystems. For the positioning and tracking subsystem, a GPS receiver was undoubtably the best choice of positioning device. No other positioning system can boast the same combination of light weight, portability, robustness, continuous global coverage, measurement of spatial coordinates in three dimensions and velocity. High positional accuracy is one of the highest priorities of any vehicle navigation system. In this case, high accuracy is vital to ensure that the correct virtual signs are observed by a driver at the correct time. Even though the accuracy of GPS improved considerably in the year 2000, the current accuracy of GPS is only marginally satisfactory for the purpose of measuring vehicle position in a VRS system. Temporal drift of positional measurements over a period of minutes and hours caused the reported position of a stationary receiver to vary considerably. The positional error was acceptable in most cases (95%) under ideal viewing conditions of the sky, but was excessive on rare occasions. Positional accuracy was degraded during real-world vehicle navigation, as predicted, but the observed error exceeded expectations in some instances. Although the observed error appeared to be less than the quoted error of 15 metres (RMS), it is not sufficient to distinguish lanes of a highway by itself. In urban areas, signal blockage and reduced accuracy due to signal obstruction will almost certainly justify a fused, more accurate positioning subsystem.

In addition, consumer-oriented GPS receivers are essentially incapable of reporting vehicle measurements at a sufficiently high frequency to know where a vehicle is located on a sub-second basis. In practice, a Kalman filter is able to reduce the severity of this problem admirably, providing a vehicle is not attempting a particularly intricate manoeuvre. However, even an optimum filter like the Kalman filter is no substitute for a more continuous data source. For a real VRS system, a more accurate and continuous positioning system is desired. One candidate that shows particular promise, albeit at extra expense, is DGPS, augmented with one or more dead-reckoning sensors that provide high short-distance accuracy. A less expensive and more immediately attainable improvement is the addition of a map-matching algorithm that makes likely assumptions about the vehicle's motion to refine its calculated position. For a real VRS system, the recommended positioning and tracking subsystem combines a DGPS receiver with dead-reckoning sensors, a map-matching algorithm and a Kalman filter to provide continuous measurements with an accuracy of 1-3 metres. The fusion of these components would also provide a degree of redundancy in those instances when the GPS signal is blocked (a relatively frequent occurrence in urban environments) and when very inaccurate data is reported by the DGPS receiver.

The database subsystem of the prototype was designed to support good scalability and simple maintenance. Without actually creating further test worlds, the final design shows that these requirements were met. The data architecture is capable of efficiently managing virtual worlds containing extremely large quantities of sign regions and signs with very few modifications required. Due to time constraints, it was not possible to test the actual performance and data storage requirements of very large worlds, so this scalability remains to be proved.

Compilation of a virtual world prior to its real-time processing converts data stored in a human-intuitive form to a more efficient internal form for processing, and yields several other benefits. Several data structures are suitable for storing sign regions and their corresponding signs efficiently, each located at a different position on the space-time requirement continuum. A quadtree is an ideal data structure for organising sign regions and storing them efficiently with minimal duplication in most circumstances. It is flexible to suit the region layout of particular worlds, scales extremely well and is amenable to a very efficient pointer representation. By caching frequently-used data in local memory, the real-time performance of a VRS system is improved.

An ongoing challenge of current ATIS systems is the collection, consolidation and communication of dynamic information that will be provided to drivers. This topic has only been very briefly discussed in this thesis. For a very large VRS operating area (such as the size of a small country), the most manageable high-level data management philosophy is considered to be a distributed system in which local body authorities maintain their own information databases. This is essentially the same approach taken as for conventional road signs. The most cost-effective database option is considered to be an onboard-offboard hybrid system. A distributed network of DGPS reference stations, with one station per city or major suburb, would provide enhanced positional information to all vehicles within its operational area. Dynamic information, such as the notification of road closures, would be transmitted in real-time over the same area serviced by each DGPS station. Static information, such as the location of nearby service facilities, would be retrieved from an onboard database stored in each vehicle. The latter could be conveniently updated at the same time as when a vehicle undergoes its routine service.

The Ring Road test world provided an excellent opportunity to exercise virtual road signs using real-world data. In particular, the decision to utilise computer-generated sign regions was crucial to refine the Ring Road world in the available time. The Ring Road world permitted the key components of the VRS prototype to be tested. However, it was very small, static and simple, and thus was not representative of a virtual world that might be utilised for a real VRS system. Future VRS prototypes should employ an onboard in-vehicle communication system to relay dynamic information between a vehicle and an offboard data source. In addition, much larger and more complex virtual worlds must be designed and tested.

The main limitation of the VRS prototype, and indeed any VRS system, concerns the display subsystem. Current, inexpensive, flat-panel display technology such as LCDs and CRTs are completely unsuitable as AR display devices. Projection devices capable of projecting the image of virtual signs into a driver's field of view are available but exceedingly expensive for research purposes. For this reason, there was no choice for the VRS prototype but to compromise by prerecording a data log file inside a vehicle and replaying it using a CRT monitor back in the lab. The result was that no augmented image appeared in a driver's field of view and real-time sign display synchronised with the vehicle's motion did not exist. Thus the environment in which virtual signs were displayed was highly unrealistic in comparison to the HMI that was sought, and the major novel element of the VRS system was never realised. This meant that no in-vehicle testing of virtual signs could be performed, and thus no conclusions can be made regarding the effectiveness of virtual signs as a medium, their safety in terms of visual interference or their ability to improve driver safety by reducing distraction and inattention. However, the lack of a suitable in-vehicle projection device was known at the project outset so the results were not disappointing. While the CRT sign images of the VRS prototype were not immersive or at all representative of what a real VRS system would present to drivers, the CRT monitor was suitable for research purposes.

A real VRS system would certainly need to address this lack of display device, as this is the single most significant factor holding back further development of a VRS system. It is still unclear as to whether any current AR display device such as a HUD or AR spectacles is suitable for displaying virtual road signs to drivers. In particular, it is unclear whether it is possible to strike a balance between the improved readability of virtual signs and the level of visual interference, and possibly attention tunnelling, caused by virtual signs. Because it was not possible to acquire a HUD, or indeed any AR device, for research purposes, a substantial amount of testing is still required to ascertain the suitability of these devices for virtual sign display or otherwise.

Much research has yet to be done on the economic, social and political implications of the rollout of a VRS system. The prototype could not and did not attempt to investigate the acceptability of virtual road signs as a method of conveying information to drivers in any depth. Further research into this non-technical aspect is necessary to ascertain whether road users, authorities and general public are ready to embrace the VRS concept. Issues of safety of a VRS system, liability in case of system failure and private costs are the three most significant factors likely to determine whether a VRS system is deemed acceptable or not.

The large scale public roll-out of virtual road signs is not expected to be feasible within the next ten years. Therefore, virtual road signs would appear to be more suitable for supplementing conventional road signs rather than replacing them altogether. To minimise disruption to the existing sign infrastructure, virtual road signs could be implemented in a transitionary manner over a period of many years.

A number of specific applications exist for virtual road signs. In New Zealand, bilingual signage would appear to be a prime contender. Rental car companies could tailor a set of virtual road signs to suit visitors from overseas countries. In a competitive setting, rally car drivers could specify their own personalised set of virtual road signs for navigation purposes. The idea of virtual signs as an information source can in fact be generalised to any environment in which real-time visual information is sought by a user in motion, who cannot afford to divert their attention away from the task at hand for long periods of time. One example in a non-road context is skiing.

In summing up, the pairing of the prototype and Ring Road test world has proved that the concept of virtual road signs is potentially technically feasible for research purposes. Two of the three main VRS subsystems can be implemented using current technology at an acceptable cost. However, several shortcomings associated with current display technology dismiss the notion of a VRS system becoming viable in any useable form beyond the lab for several years yet. System implementation is not practical at the present time due to the lack of inexpensive AR display technology. A number of potential applications of the technology have been proposed. However, they will remain speculative until a device becomes available that can display virtual signs with the required combination of high clarity, low cost, high durability that can withstand the confined environment inside a vehicle, and which is acceptable to all parties concerned. When such a device becomes available, virtual road signs may become a commercially viable proposition.

CHAPTER 10
References

Akin, A. (1998). Microsoft and 3D Graphics: A Case Study in Suppressing Innovation and Competition.
http://www.vcnet.com/bms/features/3d.html (last visited February 2001)

Alm, H. (1993). Route navigation. Deciding driving information needs. In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 187-192).* London: Taylor & Francis Ltd.

Alpine (2001). NVE-N851A DVD Vehicle Navigation System.
http://www.alpine1.com/html/D2_n_1_n_n.html (last visited February 2001)

American City & County (1998). Chameleon signs tailor message to the weather. *Vol. 113, No. 4, p. 76, April 1998.*

Arnholt, M. (2000). Paving a Telematics Future. *Ward's Auto World, Detroit, Oct 2000, Vol. 36, Issue 10, pp. 87-90.*

Ashby, M. C. & Parkes, A. M. (1993). Interface design for navigation and guidance. In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 295-310).* London: Taylor & Francis Ltd.

Azmoodeh, M. (1990). *Abstract Data Types and Algorithms* (2nd ed.). London: Macmillan Education Ltd.

Behrendt, J. (2000). Traffic operations and maintenance: Operations innovations in Germany. *Institute of Transportation Engineers Journal, Dec 2000, Vol. 70, Issue 12, pp. 26-29.*

Bennett, P. (2000). The NMEA FAQ.
http://vancouver-webpages.com/peter/nmeafaq.txt (last visited February 2001)

Bretz, E. A. (2000). X Marks the Spot, Maybe. *IEEE Spectrum, April 2000, pp. 26-36.*

Brookner, E. (1998). *Tracking and Kalman Filtering Made Easy.* New York: John Wiley & Sons, Inc.

Brown, C. G. & Hook, K. (1993). A human view of dialogue and dialogue management in the automobile. In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 333-346).* London: Taylor & Francis Ltd.

Bursa, M. (2000). Hertz fits navigation system into rental fleet. *ISATA Magazine, Dec/Jan 2000, Issue no. 8, p. 26.*

Camus, J. & Fortin, M. (1995). *Road Transport Informatics: Institutional and Legal Issues.* European Conference of Ministers of Transport (ECMT). Paris: OECD Publications Service.

Catling, I. (1994). *Advanced Technology for Road Transport: IVHS and ATT.* Norwood, Boston, MA 02062: Artech House, Inc.

Chislenko, A. (1997). Intelligent Information Filters and Enhanced Reality. *http://www.lucifer.com/~sasha/EnhancedReality.html* (last visited February 2001)

Chui, C. K. (1992). *An Introduction to Wavelets.* San Diego: Academic Press, Inc.

Claus, K. E. & Claus, R. J. (1974). *Visual Communication Through Signage: Volume 2, Sign Evaluation.* Cincinnati: Signs of the Times Pub. Co.

Communications News (2000). Wireless infrastructure makes inroads. *Oct 2000, Vol. 37, Issue 10, p. 90.*

Cooper, S. & Durrant-Whyte, H. (1994). A Kalman Filter Model for GPS Navigation of Land Vehicles. *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Vol. 1, pp. 157-163.*

Corel Corporation (1993). CorelDraw 3.0 clipart. *CorelDraw 3.0 computer software.*

Corel Corporation (1997). CorelDraw 8.0 clipart. *CorelDraw 8.0 computer software (CD-ROM).*

Cosentino, R. J. & Diggle, D. W. (1996). Differential GPS. In E. D. Kaplan (Ed.), *Understanding GPS Principles and Applications (pp. 321-383).* Norwood, Boston, MA 02062: Artech House, Inc.

DeVaul, R. W., Rhodes, B. & Schwartz, S. (2001). Augmented Reality (part of MIT Wearable Computing Web Page).
http://wearables.www.media.mit.edu/projects/wearables/augmented-reality.html (last visited February 2001)

Ding, Z. (1999). Kalman Filtering for Enhanced Global Positioning Accuracy. *Proceedings of DESIGNCON 99, Vol. 3, pp. 23-30.*

Ditlea, S. (2000). The PC Goes Ready-To-Wear. *IEEE Spectrum, October 2000, pp. 34-39.*

Donoghue, J. A. & Watson, R. R. (2000). Heads up for safety. *Air Transport World, Vol. 37, Issue 11, pp. 63-72.*

Donoho, D., Johnstone, I., Buckheit, J., Chen, S. & Scargle, J. (1996). WaveLab .701 wavelet analysis library.
http://playfair.stanford.edu/~wavelab (last visited July 1998)

Drane, C. & Rizos, C. (1998). *Positioning Systems in Intelligent Transportation Systems.* Norwood, Boston, MA 02062: Artech House, Inc.

Eby, D. W. (1999). An on-the-road comparison of in-vehicle navigation assistance systems. *Human Factors, Jun 1999, Vol. 41, Issue 2, pp. 295-311.*

ECMT (1995). *New Information Technologies in the Road Transport Sector: Policy Issues, Ergonomics and Safety.* European Conference of Ministers of Transport. Paris: OECD Publications Service.

Electronic Times (2000). Internet on wheels. *Oct 9, 2000, p. 58.*

Elliott, S. D. & Dailey, D. J. (1995). *Wireless Communications for Intelligent Transportation Systems.* Norwood, Boston, MA 02062: Artech House, Inc.

Enderby, C. & Wood, S. (1992). Head-Up Display in Automotive/Aircraft Applications. SAE Technical Paper 920740. SAE, Warrendale, Pa.

Foyle, D. C., McCann, R. S., Sanford, B. D. & Schwirzke, M. F. J. (1993). Attentional Effects with Superimposed Symbology: Implications for Head-Up Displays (HUD). *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting, Vol. 2, pp. 1340-1344.*

Frenzel, L. (2001). An evolving ITS paves the way for intelligent highways. *Electronic Design, Jan 8, 2001, Vol. 49, Issue 1, pp. 102-109.*

Garmin Corporation (1999). *GPS 12 Owner's Manual.* Olathe, Kansas: Garmin International, Inc.

Garmin Corporation (2000 [a]). What is GPS?
http://www.garmin.com/aboutGPS/ (last visited February 2001)

Garmin Corporation (2000 [b]). Decision to Discontinue SA Announced, GPS Position Accuracy Increased.
http://www.garmin.com/whatsNew/announcements2000/announcements01.html (last visited February 2001)

Garmin Corporation (2001). Mobile Electronics: StreetPilot ColorMap: Screen Examples.
http://www.garmin.com/products/colorMap/screen.html (last visited February 2001)

Gish, K. W., Staplin, L., Stewart, J. & Perel, M. (1999). Sensory and Cognitive Factors Affecting Automotive Head-Up Display Effectiveness. *Transportation Research Record, No. 1694, pp. 10-19.*

GPNN (2000 [a]). In-Car Nav: The Next Generation. *Global Positioning & Navigation News, Vol. 10, Issue 11, p. 1.*

GPNN (2000 [b]). Offboard Navigation: The New Paradigm? *Global Positioning & Navigation News, Vol. 10, Issue 25, p. 1.*

GPNN (2000 [c]). Visteon, Clarion Push Real-Time Traffic Icons, Rerouting To In-Car Navigation Systems. *Global Positioning & Navigation News, Vol. 10, Issue 10, p. 1.*

GPNN (2000 [d]). In-Vehicle Internet With Location Content Coming Soon? *Global Positioning & Navigation News, Vol. 9, Issue 22, p. 1.*

Groves, M. K. (1999). Networked Augmented Reality (AR) Future. *http://www.matra.com.au/~matt/ar.html* (last visited February 2001)

Haller, R. (1991). Experimental Investigation of Display Reading Tasks in Vehicles and Consequences for Instrument Panel Design. In A.G. Gale (Ed.), *Vision in Vehicles – III (pp. 197-203).* New York: Elsevier Science Publishing Company, Inc.

Harrison, B. L., Ishii, H., Vicente, K. J. & Buxton, W. A. S. (1995). Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention.
http://www.acm.org/sigchi/chi95/proceedings/papers/blh_bdy.htm (last visited February 2001)

Hertz (2001). NeverLost: Le Système de Navigation embarqué vous guide dans chacun de vos déplacements.
http://fr.hertz.com/serv/us/prod_lost.html (last visited February 2001)

Hofmann-Wellenhof, B., Lichtenegger, H. & Collins, J. (1994). *GPS: Theory and Practice* (3rd ed.). Wien, Austria: Springer-Verlag Wien.

Huang, B. & Lin, H. (1999). GeoVR: a Web-Based Tool for Virtual Reality Presentation from 2D GIS Data. *Computers & Geosciences, Vol. 25, No.10, pp. 1167-1175.*

Humerfelt, S. (2001). The Earth according to WGS 84.
http://home.online.no/~sigurdhu/Grid_1deg.htm (last visited February 2001)

Kim, J., Kim, H., Jang, B., Kim, J. & Kim, D. (1998). Augmented Reality Using GPS. *Proceedings of SPIE – the International Society for Optical Engineering, Vol. 3295, pp. 421-428.*

Kim, J. H. & Oh, J. H. (2000). A Land Vehicle Tracking Algorithm Using Stand-Alone GPS. *Control Engineering Practice, Vol. 8, No. 10, pp. 1189-1196.*

Kim, W., Jee, G. & Lee, J. (2000). Efficient Use of Digital Road Map in Various Positioning for ITS. *IEEE Position Location and Navigation Symposium, pp. 170-176.*

Kujawa, M. (1999). Navigating the market for ITS. *Telecommunications, Feb 1999, Vol. 33, Issue 2, International Edition, pp. 51-54.*

Lewis, J., DeMeis, R., Mehri, D. & Russelburg, K. (2000). Hot new auto technologies for 2001. *Design News, Oct 2, 2000.*

M2 Presswire (1999). RAM MOBILE DATA: Govt scheme beats traffic delays with real-time wireless info on road conditions ahead. *Apr 22, 1999, p. 1.*

M2 Presswire (2000). FANTASTIC: Fantastic – Broadband Multimedia and Internet into the Car; Broadband Internet to the Car will be Reality. *Feb 17, 2000, p. 1.*

Mattos, P. G. (1994). Integrated GPS and Dead Reckoning for Low-Cost Vehicle Navigation and Tracking. *Proceedings of VNIS'94 – 1994 Vehicle Navigation and Information Systems Conference Proceedings, IEEE Vehicular Technology Society, pp. 569-574.*

McHenry, R. (Ed.) (1992 [a]). Latitude and longitude. In *Encyclopaedia Britannica (15th ed.), Micropaedia, Vol. 7, p. 184.* Chicago: Encyclopaedia Britannica, Inc.

McHenry, R. (Ed.) (1992 [b]). Knot. In *Encyclopaedia Britannica (15th ed.), Micropaedia, Vol. 6, p. 918.* Chicago: Encyclopaedia Britannica, Inc.

Mehaffey, J. & Yeazel, J. (2001). Joe Mehaffey and Jack Yeazel's GPS Information Website. *http://joe.mehaffey.com/* (last visited February 2001)

Mehaffey, J. (2001). Error Measures.
http://joe.mehaffey.com/errors.htm (last visited February 2001)

MicroOptical Corporation (2001) WWW page. Miniature display systems. *http://www.microopticalcorp.com/* (last visited February 2001)

Microsoft Corporation (1995). Device Contexts Overview. *Win32 Programmer's Reference. ftp://ftpc.inprise.com/pub/delphi/techpubs/delphi2/win32.zip* (last visited February 2001)

Microsoft Corporation (1999). DirectX 7.0 Programmer's Reference: Direct3D. Available as part of the Microsoft DirectX 7.0 SDK (now superseded). Latest SDK version is available at *http://www.microsoft.com/directx/* (last visited February 2001)

Molofee, J. (2001). Jeff Molofee's OpenGL Windows Tutorials. *http://nehe.gamedev.net/opengl.asp* (last visited February 2001)

Murray, C. J. (2001). In-car PCs eye speed bump as interfaces shift. *Electronic Engineering Times, Jan 15, 2001, Issue 1149, p. 1.*

Nash, T. (2000). Council slams sign thieves. *Manawatu Evening Standard newspaper, July 21, 2000, front page.* Palmerston North.

New Zealand Ministry of Transport (1992). *The Road Code*. Wellington: GP Publications Limited.

Nissan (2001). Innovation: Technologies: BirdView Navigator. *http://www.nissan.co.uk/* (last visited February 2001)

NZPA. (2000 [a]). Call for variable speed limits. *Manawatu Evening Standard newspaper, October 4, 2000.* Palmerston North.

NZPA. (2000 [b]). Flexible speed limit preferred. *Manawatu Evening Standard newspaper, October 6, 2000.* Palmerston North.

OECD Scientific Experts Group (1988). *Route Guidance and In-Car Communication Systems*. Paris: OECD Publications Service.

Okabayashi, S. & Sakata, M. (1991). How an Automotive Head-Up Display Affects a Driver's Ability to Recognise the Forward View. *Proceedings of the Society for Information Display, Vol. 32/1, pp. 31-37.*

Okabayashi, S., Sugie, N. & Hatada, T. (1999). Visual Perception of HUD (Head-Up Display) Images in Practical Automobiles. *Vision In Vehicles, No. 7, pp. 169-176.*

Page, W. (1996). *43.203: Technological Mathematics: Numerical Methods Course Notes.* Palmerston North: The Department of Production Technology, Massey University.

Peacock, C. (2000). Beyond Logic: Serial / RS-232 Interfacing.
http://www.beyondlogic.org/ (last visited February 2001)

Pierce, A. (1999). Auto night vision. *Tech Directions, Feb 1999, Vol. 58, Issue 7, p. 9.*

Pimentel, K. & Teixeira, K. (1995). *Virtual Reality: Through the New Looking Glass* (2nd ed.). New York: McGraw-Hill, Inc.

Polikar, R. (2001). The Wavelet Tutorial.
http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html (last visited February 2001)

Popp, M. P. & Farber, B. (1991). Advanced Display Technologies, Route Guidance Systems, and the Position of Displays in Cars. In A.G. Gale (Ed.), *Vision in Vehicles – III (pp. 219-225).* New York: Elsevier Science Publishing Company, Inc.

Pressman, R. S. (1997). *Software Engineering: A Practitioner's Approach* (4th international ed.). New York: The McGraw-Hill Companies, Inc.

Russ, J. C. (1995). *The Image Processing Handbook* (2nd ed.). Boca Raton: CRC Press, Inc.

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures.* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Schneider, D. K. & Martin-Michiellot, S. (1998). VRML Primer and Tutorial. *http://tecfa.unige.ch/guides/vrml/vrmlman/vrmlman.html* (last visited February 2001)

Schraagen, J. M. C. (1993). Information presentation in in-car navigation systems. In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 171-185).* London: Taylor & Francis Ltd.

Silberschatz, A. & Galvin, P. B. (1994). *Operating System Concepts.* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Silicon Graphics Inc. (2001). OpenGL – High Performance 2D/3D Graphics. *http://www.opengl.org/* (last visited February 2001)

Smailus, T., Bullock, D. & Besly, D. (1996). Implementation of a Multimedia Highway Sign Database. *Institute of Transportation Engineers Journal, Vol. 66, No. 9, pp. 28-32.*

Spijkers, W. (1991). The Recognition of Traffic Signs under "Natural" Conditions. In A.G. Gale (Ed.), *Vision in Vehicles – III (pp. 317-323).* New York: Elsevier Science Publishing Company, Inc.

Spohrer, J. C. (1999). Information in Places. *IBM Systems Journal, Vol. 38, Issue 4, pp. 602-628.*

Steinfeld, A. & Green, P. (1998). Driver Responses to Navigation Information on Full-Windshield, Head-Up Displays. *International Journal of Vehicle Design, Vol. 19, No. 2, pp. 135-149.*

Stevens, A. (2000). Safety of Driver Interaction with In-Vehicle Information Systems. *Proceedings of the Institution of Mechanical Engineers, Vol. 214, Part D6, pp. 639-644.*

Stone, B. (1997). What does Garmin mean by EPE? *http://joe.mehaffey.com/epenew.txt* (last visited February 2001)

Sviden, O. (1993). MMI scenarios for the future road service informatics. In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 29-38).* London: Taylor & Francis Ltd.

The Dominion (1999). Maori road signs, place names proposed (road signs in Maori & English idea proposed by Maori Language Commission interim manager Holden Honaia). *August 18, 1999, p3.*

The Dominion (2000 [a]). Maori names 'not substitutes' (Maori names for North & South Island road signs may come into official use, but should not replace English names – Prime Minister Helen Clark). *March 6, 2000, p3.*

The Dominion (2000 [b]). Minister wants more bilingual road signs (New Zealand urged to consider using bilingual road signs using Maori & Pakeha place names – Associate Maori Affairs Minister Sandra Lee). *May 6, 2000, p7.*

The Press (2000). Road signs to be in Maori (bilingual signs to be erected in the South Island after decision by Transit NZ). *March 3, 2000, p4.*

Todoriki, T., Fukano, J., Okabayashi, S., Sakata, M. & Tsuda, H. (1994). Application of Head-Up Displays for In-Vehicle Navigation/Route Guidance. *Vehicle Navigation and Information Systems Conference Proceedings (Cat. No.94CH35703), IEEE, pp. 479-484.*

Trimble, P. S. (2000). Smart roads and smarter vehicles. *Federal Computer Week, Aug 7, 2000, Vol. 14, Issue 27, pp. 38-39.*

Unknown author A (2001). Graph of instantaneous error before and after disablement of GPS selective availability.
http://jennifer.mehaffey.com/sa_trans.gif (last visited February 2001)

Unknown author B (2001). Programming OpenGL with Visual Basic.
http://is6.pacific.net.hk/~edx/contents.htm (last visited February 2001)

Unknown author C (2000). Direct3D vs. OpenGL: A Comparison.
http://www.xmission.com/%7Elegalize/d3d-vs-opengl.html (last visited February 2001)

Vallino, J. R. (2000). Introduction to Augmented Reality.
http://www.cs.rit.edu/~jrv/research/ar/introduction.html (last visited February 2001)

Verwey, W. B. (1993). How can we prevent overload of the driver? In A. M. Parkes & S. Franzen (Eds.), *Driving Future Vehicles (pp. 235-244).* London: Taylor & Francis Ltd.

Ward, N. J., Parkes, A. & Crone, P. R. (1995). Effect of Background Scene Complexity and Field Dependence on the Legibility of Head-Up Displays for Automotive Applications. *Human Factors and Ergonomics Society, Vol. 37, No. 4, pp. 735-745.*

Warhol, A. (2001). OpenGL Programming Guide (Addison-Wesley Publishing Company) *http://heron.cc.ukans.edu/ebt-bin/nph-dweb/dynaweb/SGI_Developer/OpenGL_PG/ @Generic__BookView* (last visited February 2001)

Whelan, R. (1995). *Smart Highways, Smart Cars.* Norwood, Boston, MA 02062: Artech House, Inc.

Whitten, J. L., Bentley, L. D. & Barlow, V. M. (1994). *System Analysis and Design Methods* (3rd ed.). Burr Ridge, Illinois: Richard D. Irwin, Inc.

Wierwille, W. W., Hulse, M. C., Fischer, T. J. & Dingus, T. A. (1991). Visual Adaptation of the Driver to High-Demand Driving Situations While Navigating with an In-Car Navigation System. In A.G. Gale (Ed.), *Vision in Vehicles – III (pp. 79-87).* New York: Elsevier Science Publishing Company, Inc.

Williams, M. & Green, P. (1992). *Development and Testing of Driver Interfaces for Navigational Displays,* (Technical Report UMTRI 92-21). Ann Arbor, MI: The University of Michigan Transportation Research Institute.

Wilson, D. L. (2001 [a]). Horizontal error distribution with and without GPS selective availability.
http://users.erols.com/dlwilson/gpscmpsa.htm (last visited February 2001)

Wilson, D. L. (2001 [b]). GPS Horizontal Position Accuracy.
http://users.erols.com/dlwilson/gpsacc.htm (last visited February 2001)

Wilson, H. W. (1999). Hazards ahead. *Electronics Now, May 1999, Vol. 70, Issue 5, p. 14.*

Wormley, S. J. (2000). GPS Errors & Estimating Your Receiver's Accuracy. *http://www.cnde.iastate.edu/staff/swormley/gps/check_sa.html* (last visited February 2001)

Zhao, Y. (1997). *Vehicle Location and Navigation Systems.* Norwood, Boston, MA 02062: Artech House, Inc.

APPENDIX A
Coordinate Systems and Transformations

This appendix describes the three main coordinate systems of the VRS prototype (summarised in [2.4.1]) and the transformations that convert between them. The transformations are completely defined in terms of six parameters: $[x\ z]_{\text{origin pixel, GPS}}$, $[x\ z]_{\text{Scale GPS --> pixel}}$, $\theta_{\text{GPS --> pixel}}$, and $[z]_{\text{point, GPS}}$ at world centre. The first five of these parameters are described in Table 7.1 of (7.3.1). The sixth parameter, $[z]_{\text{point, GPS}}$ at world centre, is the latitude of the centre point in a virtual world. This parameter is used to calculate equatorial GPS coordinates.

Coordinate system (1): GPS Latitude and Longitude (WGS-84)

This is an absolute standard coordinate system for navigation on the surface of the Earth. It is a non-linear latitude-longitude system because the length of one longitudinal minute changes with latitude.

Origin:
0 minutes latitude, 0 minutes longitude = point on equator in line with the Greenwich meridian.

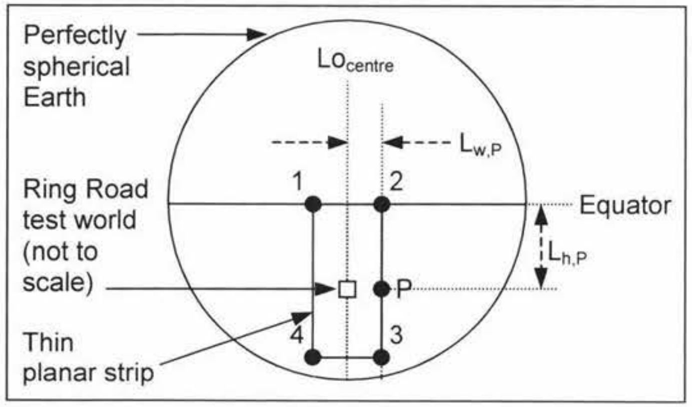
Units:
Longitude is taken as x dimension, in minutes.
Latitude is z dimension, in minutes.

Sign conventions:
Negative longitude is West, positive longitude is East.
Negative latitude is South, positive latitude is North.

Conversions:

In general, conversion from system (1) to linear systems of (2) and (3) is complex. Hofmann-Wellenhof *et al.* (1994) describe the Transverse Mercurator (Gauss-Kruger) projection that allows planar coordinates (corresponding to system 2) to be calculated given a pair of ellipsoid coordinates (corresponding to system 1). This takes the form of two series expansions.

A simpler approximation is given here. Consider a thin planar rectangular strip projected onto a perfect sphere of radius R metres that represents the Earth, as shown in Figure A.1. The height and width of this strip are L_h and L_w respectively. The top margin of the strip, defined by points 1 and 2, lie at zero GPS latitude and the strip is centred at a GPS longitude Lo_{centre}. The bottom margin is defined by points 3 and 4. Because the top and bottom margins lie at different latitudes, the longitudinal angle between points 1 and 2 and points 3 and 4 is different. Take a point P lying on the perimeter of the thin strip. Point P lies a curved distance $L_{w,P}$ measured from centre longitude Lo_{centre} parallel to the equator, and a curved distance $L_{h,P}$ measured from the equatorial parallel line perpendicular to it (see Figure A.1). Both $L_{w,P}$ and $L_{h,P}$ are measured in metres. The latitude and longitude of point P (measured in radians) are then approximately given by Eq. A.1 and Eq. A.2. If calculations are made for all points on the perimeter of the thin rectangular strip and the latitude and longitude are plotted on linear axes, the shape shown in Figure A.2 is generated.



Perfectly spherical Earth
Lo_centre
L_w,P
1
2
Equator
L_h,P
Ring Road test world (not to scale)
P
Thin planar strip
4
3

Figure A.1: Thin rectangular strip projected onto a perfect sphere. $L_{h,P}$ is an unsigned positive distance, $L_{w,P}$ is a signed distance. $L_{h,P}$ and $L_{w,P}$ are measured in metres.

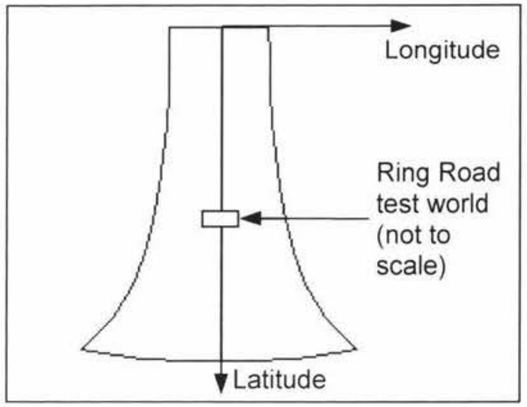


Figure A.2: Generated shape when calculated latitude and longitude of all points on perimeter of thin strip are plotted on linear axes.

$$\mathrm{La}_{\mathrm{point\ P}} \approx -\frac{\pi}{2} + \frac{\sqrt{L_{w,P}{}^2 \sin^2(L_{h,P}/R) + (\pi R/2 - L_{h,P})^2}}{R}$$

Eq. A.1: Approximate GPS latitude of a point P, in radians.

$$\mathrm{Lo}_{\mathrm{point\ P}} \approx \mathrm{Lo}_{\mathrm{centre}} + \frac{L_{w,P}}{R\cos(\mathrm{La}_{\mathrm{point\ P}})}$$

Eq. A.2: Approximate GPS longitude of a point P, in radians.

For the VRS prototype, the Ring Road test world is very small and corresponds to a very small thin rectangle (shown in Figure A.1) relative to the size of the Earth. The corresponding shape of the test world in Figure A.2 is approximately rectangular. Then:

Conversion of coordinates from system (1), $[x\ z]_{\text{point, GPS}}$, to system (2), $[x\ z]_{\text{point, equator GPS}}$:

$$[x]_{\text{point, equator GPS}} = [x]_{\text{point, GPS}}.* \cos([z]_{\text{point, GPS}} \text{ at world centre}) = [x]_{\text{point, GPS}}.* \text{ constant}$$

$$[z]_{\text{point, equator GPS}} = [z]_{\text{point, GPS}}$$

Conversion of coordinates from system (1), $[x\ z]_{\text{point, GPS}}$, to system (3), $[x\ z]_{\text{point, pixel}}$:

$$[x\ z]_{\text{point, pixel}} = ([x\ z]_{\text{point, GPS}} - [x\ z]_{\text{origin pixel, GPS}}).* [x\ z]_{\text{Scale GPS --> pixel}} * \begin{bmatrix} \cos\theta_{\text{GPS-->pixel}} & -\sin\theta_{\text{GPS-->pixel}} \\ -\sin\theta_{\text{GPS-->pixel}} & -\cos\theta_{\text{GPS-->pixel}} \end{bmatrix}$$

Coordinate system (2): "Equatorial" GPS Latitude and Longitude

This system is identical to system (1) with one major exception: one minute of longitude is of fixed length, defined as the length of one minute of longitude measured at the equator. This is approximately equal to the length of one minute of latitude. This is a linear latitude-longitude system because the length of one minute of longitude is fixed and invariant to latitude. To keep the mapping between system (1) and this system approximately linear, this system is only defined for a small localised region corresponding to the small Ring Road test world shown in Figure A.2.

Origin:
0 minutes latitude, 0 minutes longitude = point on equator in line with the Greenwich meridian.

Units:
Equatorial longitude is taken as x dimension, in minutes.
Equatorial latitude is z dimension, in minutes.

Sign conventions:
Negative longitude is West, positive longitude is East.
Negative latitude is South, positive latitude is North.

Conversions:

Conversion of coordinates from system (2), $[x\ z]_{\text{point, equator GPS}}$, to system (1), $[x\ z]_{\text{point, GPS}}$, if conversion region is small and localised:

$$[x]_{\text{point, GPS}} = [x]_{\text{point, equator GPS}}./\cos([z]_{\text{point, GPS}} \text{ at world centre}) = [x]_{\text{point, equator GPS}}./\text{ constant}$$

$$[z]_{\text{point, GPS}} = [z]_{\text{point, equator GPS}}$$

Conversion of coordinates from system (2), $[x\ z]_{\text{point, equator GPS}}$, to system (3), $[x\ z]_{\text{point, pixel}}$:

$$[x\ z]_{\text{point, pixel}} = \left(\left[\frac{x}{\cos([z]_{\text{point,GPS}} \text{ at world centre})} \quad z\right]_{\text{point, equator GPS}} - [x\ z]_{\text{origin pixel, GPS}}\right).*\ [x\ z]_{\text{Scale GPS --> pixel}} * \begin{bmatrix} \cos\theta_{\text{GPS-->pixel}} & -\sin\theta_{\text{GPS-->pixel}} \\ -\sin\theta_{\text{GPS-->pixel}} & -\cos\theta_{\text{GPS-->pixel}} \end{bmatrix}$$

Coordinate system (3): Paper Plot Pixels

This system matches the standard coordinate system used to represent bitmap images. Here the paper plot refers to the scale plot of the Ring Road used for prototype testing, described in (7.3) and shown in Figure G.1 of (G.3).

Origin:
(0, 0) = top left hand corner of image.

Units:
x coordinate is in horizontal dimension (see Figure G.1 of [G.3]), in pixels.
z coordinate is in vertical dimension (see Figure G.1 of [G.3]), in pixels.

Sign conventions:
Positive x is right, positive z is down.

Conversions:

Conversion of coordinates from system (3), $[x\ z]_{\text{point, pixel}}$, to system (1), $[x\ z]_{\text{point, GPS}}$, if conversion region is small and localised:

$$[x\ z]_{\text{point, GPS}} = \left(\frac{[x\ z]_{\text{point, pixel}} * \begin{bmatrix} \cos\theta_{\text{GPS-->pixel}} & -\sin\theta_{\text{GPS-->pixel}} \\ -\sin\theta_{\text{GPS-->pixel}} & -\cos\theta_{\text{GPS-->pixel}} \end{bmatrix}}{[x\ z]_{\text{Scale GPS-->pixel}}} \right) + [x\ z]_{\text{origin pixel, GPS}}$$

Conversion of coordinates from system (3), $[x\ z]_{\text{point, pixel}}$, to system (2), $[x\ z]_{\text{point, equator GPS}}$:

$$[\mathrm{x}]_{\text{point, equator GPS}} = \left(\frac{[\mathrm{x\ z}]_{\text{point, pixel}} * \begin{bmatrix} \cos\theta_{\text{GPS--->pixel}} \\ -\sin\theta_{\text{GPS--->pixel}} \end{bmatrix}}{[\mathrm{x}]_{\text{Scale GPS--->pixel}}} + [\mathrm{x}]_{\text{origin pixel,GPS}} \right) .* \cos([\mathrm{z}]_{\text{point, GPS}} \text{ of centre of world})$$

$$[\mathrm{z}]_{\text{point, equator GPS}} = \frac{[\mathrm{x\ z}]_{\text{point, pixel}} * \begin{bmatrix} -\sin\theta_{\text{GPS--->pixel}} \\ -\cos\theta_{\text{GPS--->pixel}} \end{bmatrix}}{[\mathrm{z}]_{\text{Scale GPS--->pixel}}} + [\mathrm{z}]_{\text{origin pixel, GPS}}$$

APPENDIX B NMEA Protocol Sentences

This appendix describes the NMEA-0183 v2.0 protocol introduced in (3.3) in more detail. The VRS prototype utilises the NMEA protocol to transfer all navigation data between the GPS receiver and laptop PC. The material in this appendix is quoted directly from (Bennett, 2000), except for the example NMEA data burst in (B.1) which was captured experimentally.

B.1) Example of Full NMEA Data Burst

Box B.1 shows the sentences contained within one full NMEA data burst received from the Garmin GPS-12 unit inside a stationary vehicle.

```
$GPRMC,011946,A,4012.853,S,17533.060,E,000.0,360.0,220600,021.4,E*68
$GPRMB,A,,,,,,,,,,,,V*71
$GPGGA,011946,4012.853,S,17533.060,E,1,04,2.3,89.6,M,19.8,M,,*6A
$GPGSA,A,3,01,,,,,,,18,19,,,31,2.3,2.3,1.0*31
$GPGSV,3,1,12,01,40,006,39,03,43,115,30,08,27,240,00,10,06,223,00*7C
$GPGSV,3,2,12,13,28,227,00,15,03,078,00,16,03,327,00,18,62,302,43*75
$GPGSV,3,3,12,19,79,161,41,22,17,119,00,27,48,232,00,31,58,053,49*7E
$PGRME,16.6,M,38.9,M,42.4,M*1F
$GPGLL,4012.852,S,17533.060,E,011947,A*30
$PGRMZ,299,f,3*19
$PGRMM,WGS 84*06
$GPBOD,,T,,M,,*47
$GPRTE,1,1,c,0*07
```

Box B.1: Example of full NMEA data burst

B.2) Description of Sentence IDs Sent by GPS-12 Receiver

This section describes those NMEA-0183 sentence IDs sent by the Garmin GPS-12 receiver not previously described in (3.4). The standard talker ID for GPS receivers is "GP". Thus, for each standard sentence ID given in Table B.1, the ID is preceded by "$GP" in each sentence, as shown in Box B.1.

Sentence ID	Description by Example	
BOD	Bearing – Origin to destination waypoint Example: BOD,045.,T,023.,M,DEST,START	
	045.,T	Bearing 045 True from "START" to "DEST"
	023.,M	Bearing 023 Magnetic from "START" to "DEST"
	DEST	Destination waypoint ID
	START	Origin waypoint ID
GGA	Global Positioning System fix data Example: GGA,123519,4807.038,N,01131.324,E,1,08,0.9,545.4,M,46.9,M, , *42	
	123519	Fix taken at 12:35:19 UTC
	4807.038,N	Latitude 48 deg 07.038' N
	01131.324,E	Longitude 11 deg 31.324' E
	1	Fix quality: 0 = invalid, 1 = GPS fix, 2 = DGPS fix
	08	Number of satellites being tracked
	0.9	Horizontal dilution of position
	545.4,M	Altitude, Metres, above mean sea level
	46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
	(empty field)	Time in seconds since last DGPS update
	(empty field)	DGPS station ID num
GLL	Geographic position, latitude and longitude Example: GLL,4916.45,N,12311.12,W,225444,A	
	4916.46,N	Latitude 49 deg. 16.45 min. North
	12311.12,W	Longitude 123 deg. 11.12 min. West
	225444	Fix taken at 22:54:44 UTC
	A	Data is valid
GSA	GPS DOP and active satellites Example: GSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39	
	A	Auto selection of 2D or 3D fix (M = manual)
	3	3D fix
	04,05...	Pseudorandom noise numbers (PRNs) of satellites used for fix
	2.5	(space for 12)
	1.3	PDOP (dilution of precision)
	2.1	Horizontal dilution of precision (HDOP)
		Vertical dilution of precision (VDOP)
		DOP is an indication of the effect of satellite geometry on the accuracy of the fix.
GSV	Satellites in view Example: GSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75	
	2	Number of sentences for full data
	1	Sentence 1 of 2
	08	Number of satellites in view
	01	Satellite PRN number
	40	Elevation, degrees
	083	Azimuth, degrees
	46	Signal strength – higher is better
		<repeat for up to 4 satellites per sentence>
		There may be up to three GSV sentences in a data packet.

RMB	Recommended minimum navigation information (sent by GPS receiver when a destination waypoint is active) Example: RMB,A,0.66,L,003,004,4917.24,N,12309.57,W,001.3,052.5,000.5,V*0B	
	A	Data status A = OK, V = warning
	0.66,L	Cross-track error (nautical miles, 9.9 max.), steer Left to correct (or R = right)
	003	Origin waypoint ID
	004	Destination waypoint ID
	4917.24,N	Destination waypoint latitude 49 deg. 17.24 min. N
	12309.57,W	Destination waypoint longitude 123 deg. 09.57 min. W
	001.3	Range to destination, nautical miles
	052.5	True bearing to destination
	000.5	Velocity towards destination, knots
	V	Arrival alarm: A = arrived, V = not arrived
	*0B	Mandatory checksum
RTE	Waypoints in active route Example: RTE,2,1,c,0,W3IWI,DRIVWY,32CEDR,32-29,32BKLD,32-I95,32-US1,BW-32,BW-198*69	
	2	Two sentences for full data
	1	This is sentence 1 of 2
	c	c = complete list of waypoints in this route w = first listed waypoint is start of current leg
	0	Route identifier
	W3IWI...	Waypoint identifiers

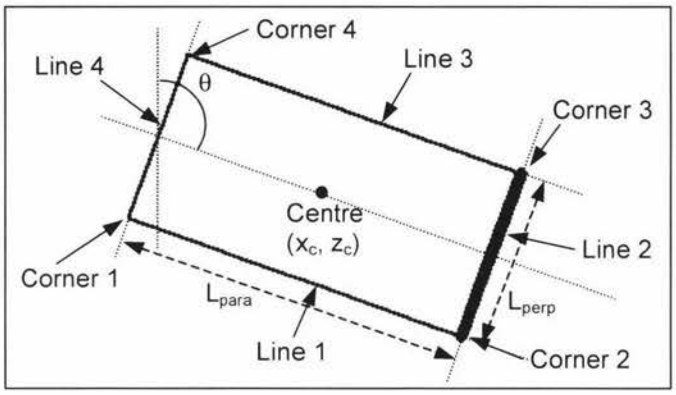
Table B.1: Explanation of standard sentence IDs sent by Garmin GPS-12 receiver

The NMEA-0183 standard allows individual manufacturers to define proprietary sentence formats. These sentences start with "$P", then a three-letter manufacturer ID, followed by whatever data the manufacturer wishes to specify. The two proprietary Garmin NMEA-0183 tags sent by the Garmin GPS-12 receiver not previously described in (3.4) are:

1) $PGRMM. This describes the currently active horizontal datum. For this application, the horizontal datum was defined by the WGS-84 coordinate system (see [2.4.1]).
2) $PGRMZ. This describes the receiver altitude. The VRS prototype did not make use of the altitude measurement reported by the GPS receiver.

APPENDIX C
Supplementary Equations

C.1) Area of a Rectangle Sign Region



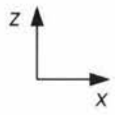


Figure C.1: Rectangle sign region definition

With reference to Figure C.1:

In equatorial GPS coordinate system (2), a rectangle region is defined by that set of (x, z) that satisfy $k_3 \le (x \cos\theta - z \sin\theta) \le k_1$ and $k_4 \le (x \sin\theta + z \cos\theta) \le k_2$, where the equations of the four lines representing the sides of the rectangle are given below.

Line 1: $x_{corner2} \cos\theta - z_{corner2} \sin\theta = k_1$, where:

$$x_{corner2} = x_c + 0.5\, L_{para} \sin\theta + 0.5\, L_{perp} \cos\theta,$$

$$z_{corner2} = z_c + 0.5\, L_{para} \cos\theta - 0.5\, L_{perp} \sin\theta$$

Line 2: $x_{corner2} \sin\theta + z_{corner2} \cos\theta = k_2$, where $x_{corner2}$ and $z_{corner2}$ are defined as for line 1.

Line 3: $x_{corner4} \cos\theta - z_{corner4} \sin\theta = k_3$, where:

$$x_{corner4} = x_c - 0.5\, L_{para} \sin\theta - 0.5\, L_{perp} \cos\theta,$$

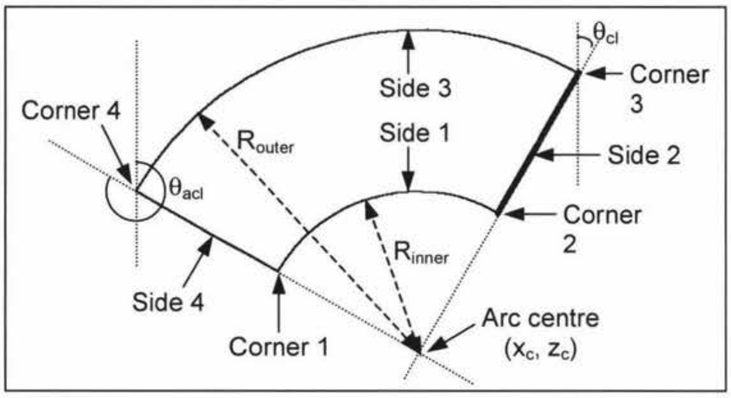
$$z_{corner4} = z_c - 0.5\, L_{para} \cos\theta + 0.5\, L_{perp} \sin\theta$$

Line 4: $x_{corner4} \sin\theta + z_{corner4} \cos\theta = k_4$, where $x_{corner4}$ and $z_{corner4}$ are defined as for line 3.

These equations are used in two instances for the VRS prototype:

1) To determine whether a vehicle position is contained by a rectangle sign region SR, as part of the check as to whether SR is currently active or not (see [4.1]).
2) To determine whether any point contained by a quadtree quadrant is also contained by a rectangle sign region during world compilation (see [4.4.5]).

C.2) Area of an Arc Sign Region



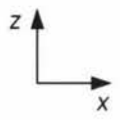


Figure C.2: Arc sign region definition

With reference to Figure C.2:

In equatorial GPS coordinate system (2), a point (x, z) lies inside an arc region if the conditions listed below are satisfied:

1) $R_{inner} \leq \sqrt{(x - x_c)^2 + (z - z_c)^2} \leq R_{outer}$, and

2) Angle ϕ made by (x, z) relative to (x_c, z_c) lies within the angular range swept out by the arc.

These conditions are used for the same two purposes in the VRS prototype as for rectangle sign regions (see [C.1]).

Calculation of ϕ:

```
If $(x = x_c)$ then
    If $(x > x_c)$ then $\phi = (\pi / 2)$ otherwise $\phi = (3\pi / 2)$.
Otherwise
    $\phi = \text{arctangent}((x - x_c) / (z - z_c))$.
    If $(z < z_c)$ then add $\pi$ to $\phi$.
    If $(x < x_c)$ and $(z > z_c)$ then add $2\pi$ to $\phi$.
```

If $(\theta_{acl} <= \theta_{cl})$, then angle ϕ lies within arc's angular range if $(\theta_{acl} <= \phi)$ and $(\phi <= \theta_{cl})$.
If $(\theta_{acl} > \theta_{cl})$, then angle ϕ lies within arc's angular range if $(\theta_{acl} <= \phi)$ or $(\phi <= \theta_{cl})$.

C.3) Linear-Perpendicular Regression

This is used to generate best-fit rectangle sign regions in (7.4.2) and calculate suitable starting coordinates when generating best-fit arc sign regions (see [7.4.3] and [G.7]). The set of N points to regress is (x_{pi}, z_{pi}) for i ε [1:N]. In equatorial GPS coordinate system (2), the equation of the best-fit regression line is [$x \cos \theta - z \sin \theta = k$]. The aim of regression is to find values of k and θ that minimise sum of squared error SSE over all N points, where for a given point p_i, E is the distance between the best-fit regression line and p_i, measured perpendicular to the regression line.

$$E = (x_{pi} \cos\theta - z_{pi} \sin\theta - k) \tag{C.3.1}$$

$$SSE = \sum_{i=1}^{N} (x_{pi} \cos\theta - z_{pi} \sin\theta - k)^2 \tag{C.3.2}$$

$$\frac{\partial SSE}{\partial k} = -2\sum_{i=1}^{N} (x_{pi} \cos\theta - z_{pi} \sin\theta - k) = 0 \text{ at minimum SSE point.} \tag{C.3.3}$$

$$\therefore \text{ Best value for } k = \frac{1}{N}\sum_{i=1}^{N} (x_{pi} \cos\theta - z_{pi} \sin\theta) \tag{C.3.4}$$

$$\frac{\partial SSE}{\partial \theta} = 2\left[\sqrt{\left[C - \frac{(B^2 - A^2)}{2N}\right]^2 + \left[D - \frac{AB}{N}\right]^2} \cos(2\theta - \phi)\right] = 0 \text{ at minimum SSE point, where}$$

$$\phi = \tan^{-1}\left[\frac{\left[C - \frac{(B^2 - A^2)}{2N}\right]}{-\left[D - \frac{AB}{N}\right]}\right],$$

$$A = \sum_{i=1}^{N} x_{pi}, \; B = \sum_{i=1}^{N} z_{pi}, \; C = \sum_{i=1}^{N}\left(\frac{z_{pi}^{\;2} - x_{pi}^{\;2}}{2}\right) \text{ and } D = \sum_{i=1}^{N} x_{pi} z_{pi} \tag{C.3.5}$$

Best value of θ is found directly by solving $[2\theta_1 - \phi = \pi/2]$ and $[2\theta_2 - \phi = 3\pi/2]$ for θ_1 and θ_2 respectively, and noting that value of θ_1 or θ_2 that minimises SSE when substituted into Eq. (C.3.2) as θ.

C.4) Arc Regression

This is the desired method of generating best-fit arc sign regions in (7.4.3). The relevant workings are given here, even though no closed-form solution of the three simultaneous equations (C.4.4-6) exists.

The set of N points to regression is (x_{pi}, z_{pi}) for i ε [1:N]. In equatorial GPS coordinate system (2), parametric equations that define a circle centred at (x_c, z_c) with radius R are:

$x = x_c + R \sin\theta$, $z = z_c + R \cos\theta$, where θ is angle. (C.4.1)

The aim of arc regression is to find values of x_c, z_c and R that minimise the sum of squared error SSE over all N points, where for a given point p_i, E is the straight-line distance between the best-fit arc and p_i measured perpendicular to the arc.

$$E = \sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2} - R \qquad (C.4.2)$$

$$SSE = \sum_{i=1}^{N} \left[(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2 - 2R\sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2} + R^2 \right] \qquad (C.4.3)$$

$$\frac{\partial SSE}{\partial x_c} = \sum_{i=1}^{N} -2(x_{pi} - x_c) \left[1 - \frac{R}{\sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2}} \right] = 0 \text{ at minimum SSE point} \qquad (C.4.4)$$

$$\frac{\partial SSE}{\partial z_c} = \sum_{i=1}^{N} -2(z_{pi} - z_c) \left[1 - \frac{R}{\sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2}} \right] = 0 \text{ at minimum SSE point} \qquad (C.4.5)$$

$$\frac{\partial SSE}{\partial R} = \left[\sum_{i=1}^{N} -2\sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2} + 2RN \right] = 0 \text{ at minimum SSE point} \qquad (C.4.6)$$

$$\therefore \text{ Best value for } R = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2} \qquad (C.4.7)$$

Substituting best value for R into SSE:

$$SSE = \sum_{i=1}^{N} \left[(x_{pi} - x_c)^2 + (z_{pi} - z_c)^2 \right] - NR^2 \qquad (C.4.8)$$

APPENDIX D
World Compilation Using a Nine-Tree

D.1) Overview

The quadtree data structure utilised to decompose a virtual world into manageable blocks (see [4.4]) is simple and effective for this application. One problem noted in (4.4.8) is sign region duplication, where one sign region SR is referenced by multiple quadtree leaf nodes. The MX-CIF quadtree mentioned in (4.4.3) and (4.4.8) overcomes this problem; however, it is ineffective for sign regions that span high-level boundary lines running between quadrants.

To improve the spread of sign regions across nodes for the case of spanning sign regions described above, an alternate world compilation algorithm is presented in (D.2). This utilises the 9-tree data structure introduced in (4.4.8) that contains nine overlapping blocks at its first decomposition level. Figure 4.16.a in (4.4.8) illustrates this decomposed structure and the numbering convention that associates blocks with nodes. Because the blocks overlap, blocks at the third decomposition level and deeper in the tree can potentially be described as belonging to more than one parent node. To resolve this ambiguity, a rule is enforced to classify each block. Given a block B1 represented by a leaf node, a list of nodes describes the path taken down the tree from the root node to reach B1, as is done for a standard quadtree. At each level, the block B2 in the tree path that potentially leads to B1 is reduced in size, as for a standard quadtree. At each step, if B1 is completely enclosed by block B = 1, 3, 7 or 9, according to the numbering convention of Figure 4.16.a, then block B2 is taken to be B. If not, then blocks B = 2, 4, 6 and 8 are examined, and if B1 is completely enclosed by one of these blocks, then B2 is taken to be B. Finally, if B2 has still not been identified at this stage, B2 must be block 5. Thus the search order of blocks is 1, 3, 7, 9, 2, 4, 6, 8, 5, and this avoids the possibility of associating two or more nodes with a given block. As an example, a fully decomposed world space to three levels is shown in Figure D.1.a, and the corresponding 9-tree structure is shown in Figure D.1.b.

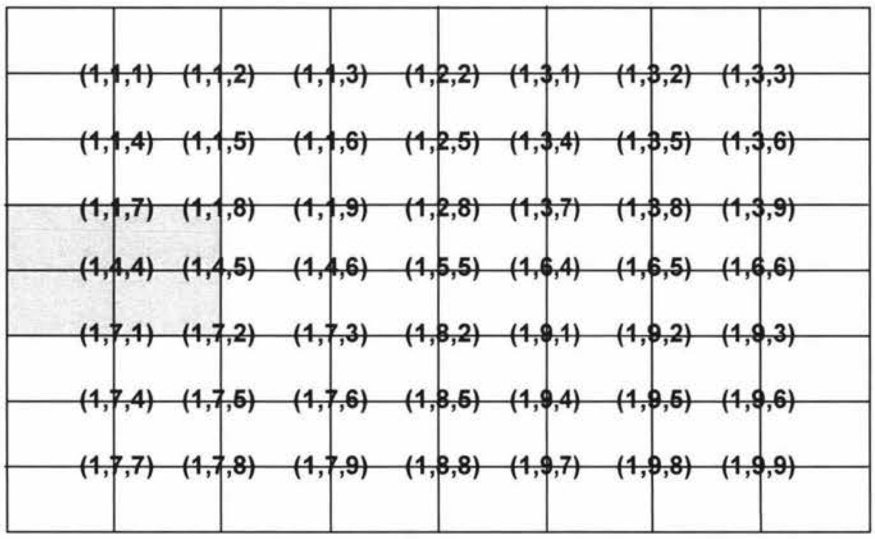


Figure D.1.a: World space decomposed to three levels using a 9-tree, showing the numbers of nodes traversed when searching for each leaf node in the 9-tree. The block corresponding to search path (1, 4, 4) is shaded gray as an example.

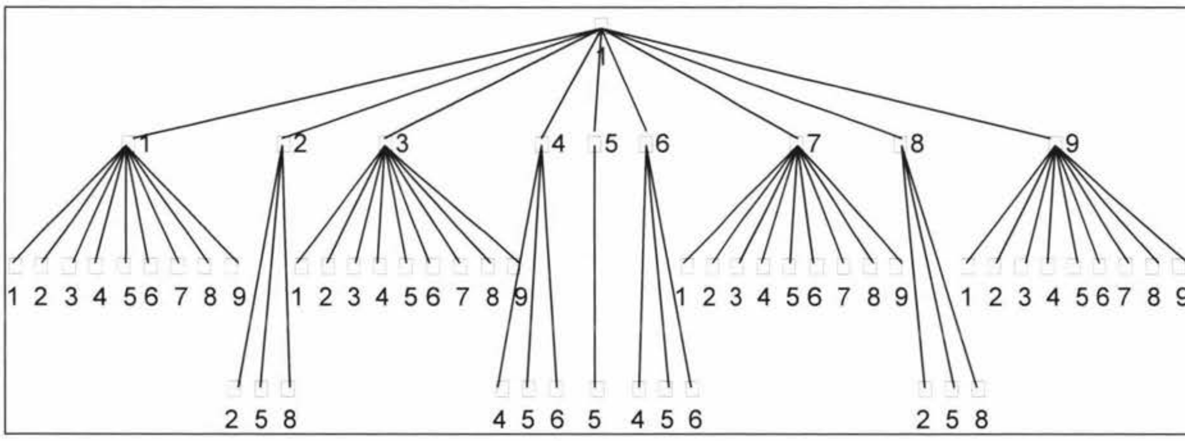


Figure D.1.b: 9-tree structure corresponding to the decomposed world shown in Figure D.1.a

The algorithm that would compile a virtual world for the VRS prototype using a 9-tree is described in Stage 1 of (D.2). Both a standard quadtree and a 9-tree are utilised. The quadtree is used to associate a given vehicle position (x_i, z_i) with a node quadrant Q, as per usual. Once Q is known, the set of sign regions associated with Q is stored by one or more nodes in the 9-tree. For simplicity, the algorithm proceeds in a bottom-up, non-adaptive manner (see [4.4.6]). A more advanced algorithm would proceed in a top-down, adaptive manner similar to that described in (4.4.6) to yield even greater efficiency. The steps that return the set of sign regions associated with a given vehicle position (x_i, z_i) are described in Stage 2 of (D.2).

D.2) World Compilation and Sign Region Determination

Choose a depth of decomposition for a quadtree, D_{max}. Leaf node quadrants will exist at depth D_{max}.

Each leaf node QN in the quadtree initially stores four properties describing the smallest rectangular space that must be searched to correctly determine which sign regions are associated with this quadtree leaf node. This is described as the *merged (minimal) search space* of QN. The four properties describe the top left corner coordinates, height and width of the rectangle. The starting value of height and width for each leaf node is zero. That is, the merged (minimal) search space of each leaf node begins with an area of zero.

Three possibilities exist: leaf node QN represents a quadrant containing:

a) 0 completely enclosed regions, 0 incompletely enclosed regions.
This quadrant contains no sign regions.

b) >0 completely enclosed regions, 0 incompletely enclosed regions.
In this case, all sign regions associated with QN are contained within QN's quadrant, so QN's merged (minimal) search space must be completely enclosed by QN's quadrant. Figure D.2.a shows one example.

c) >0 incompletely enclosed regions
In this case, one or more sign regions extend outside the perimeter of the quadrant represented by QN. The merged (minimal) search space of QN is the smallest rectangle that can be fitted around all sign regions associated with QN's quadrant, such that all regions are completely enclosed by the rectangle. Figure D.2.b shows one example.

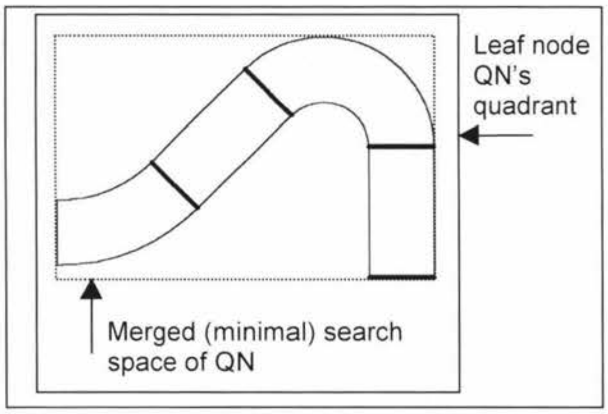


Figure D.2.a: QN's quadrant contains only completely enclosed sign regions

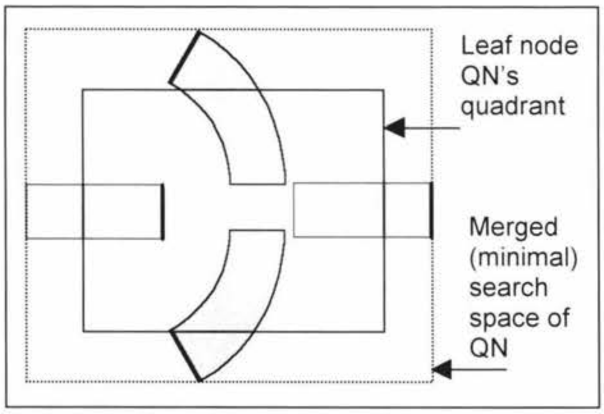


Figure D.2.b: QN's quadrant contains sign regions that extend beyond its perimeter

For each leaf node QN, the *merged (enlarged) search space* is the smallest block represented by a node in the 9-tree that completely encloses the *merged (minimal) search space* of QN. Each leaf node QN in the quadtree stores a pointer to the 9-tree node that corresponds to the merged (enlarged) search space of QN. Each node in the 9-tree defines a list of sign regions that are completely enclosed by that node's block. Each sign region SR in a virtual world is associated with exactly one 9-tree node, that being the smallest block that completely encloses SR.

Stage 1: Compile World Data Structures

A) Determine merged (minimal) search space of each sign region:

For each sign region SR in source world database DB2:

1) Determine the leaf node quadrants in the quadtree that contain SR as an incomplete region or complete region. For each quadrant QN:

a) Find QN in the quadtree. Create quadtree nodes as necessary until leaf node QN is found.

b) Compare the merged (minimal) search space of node QN with the smallest rectangle that can be fitted around sign region SR. If the latter is not completely enclosed within the former, determine the merged (minimal) search space that completely encloses both the existing merged search space and SR, and set new merged (minimal) search space of node QN equal to this area.

B) Form list of sign regions associated with each 9-tree node:

For each sign region SR in source world database DB2:

2) Determine the smallest block that can be represented by a 9-tree node that completely encloses region SR. Find this node P in the 9-tree, creating new nodes in the 9-tree as necessary, and add region SR to node P's region list.

C) Determine mappings between quadtree nodes and 9-tree nodes:

For each quadtree leaf node QN in existence:

3) Determine the smallest rectangle that can be represented by a node in the 9-tree that completely encloses the merged (minimal) search space of QN. This rectangle is the merged (enlarged) search space of QN. Set QN's pointer to the corresponding 9-tree node.

Stage 2: Use Compiled Data Structures to Determine Which Signs are Visible

On application start-up:

1) The quadtree leaf node that corresponds to the first vehicle position is unknown. Therefore, when the first vehicle position (x_0, z_0) is received, search the quadtree for the leaf node QN whose quadrant contains position (x_0, z_0). Once found, this leaf node is known as the *current* leaf node. Follow node QN's pointer into the 9-tree at node P. Define $L_{active\ SR}$ as an empty set.

2) For all sign regions in node P's region list, test whether each region SR is active or not (see [4.1]) in a sequential fashion:
   a) If it is active, add SR to list $L_{active\ SR}$.
   b) If it is inactive, do nothing.

3) Select the four nodes P1, P2, P3 and P4 at the next deeper level in the 9-tree whose blocks contain current vehicle position (x_i, z_i). For all sign regions in the region lists of nodes P1-P4, test whether each region SR is active or not in the same manner as step (2).

4) Repeat step (3) in a recursive manner until no further recursion is possible (a leaf node in the 9-tree is reached).
5) Return list $L_{active\ SR}$.

After application start-up:

6) When a new vehicle position (x_i, z_i) is to be processed, check whether it is contained within the quadrant of the current leaf node, where the latter is remembered from the previous iteration:

   a) If it is not, search the quadtree for the new leaf node whose quadrant contains the current vehicle position. It is likely to be more efficient to traverse back up the quadtree from the current leaf node and down again, rather than search the whole quadtree from its root node.
   b) If it is, no new quadtree search is needed.

7) Define $L_{active\ SR}$ as an empty set. Return a new list of active regions $L_{active\ SR}$ according to steps (2)-(5) above.

APPENDIX E
Database Field Definitions

This appendix provides definitions of all fields in databases DB1, DB2 and DB3 of the VRS prototype (see [5.4]). Three-letter prefixes of each field name define the data type of that field, as listed in Table E.1. Implementations of the databases may be found on the CD-R disc contained in the back pocket of this thesis.

Prefix	Data Type
byt	Byte
dtm	Date and time
int	16-bit integer
lng	32-bit long integer
sng	32-bit single precision floating-point number
dbl	64-bit double precision floating-point number
str	Variable-length string

Table E.1: Field data types and their prefixes

E.1) Field Definitions for Feature Database (Database DB1)

This database stores the raw GPS coordinates of features measured around the Ring Road virtual world, sign region parameters such as width and maximum length (see [7.4]), and all calculated transformation parameters that convert between coordinate systems (see Appendix A). The region generator application described in (7.3) and (7.4) takes this database as an input and writes data directly into tables *tbdRegionRect* and *tbdRegionArc* of the source world database DB2 (see [E.2]).

E.1.1) Table tbdFeature

This table stores the GPS coordinates of all Ring Road features. All coordinates are defined in the GPS coordinate system 1 (see Appendix A).

Field Name	Definition
intFeatureKey0	Primary key field of feature.
dblCoordX0	X coordinate of feature, relative to GPS offset.
dblCoordZ0	Z coordinate of feature, relative to GPS offset.

E.1.2) Table tbdMisc1

This table stores the range and resolution of values of the transformation parameters that were tested per iteration of the minimisation algorithm described in (7.3.2). The transformation parameters convert between GPS coordinates and pixel coordinates of the Ring Road paper plot. Data is stored in two text fields, *strFieldName1* and *strFieldValue1*, as described below.

strFieldName1	strFieldValue1
dblMinOriginPinGX, dblMaxOriginPinGX, dblStepOriginPinGX	Lower bound, upper bound and resolution of values tested for $[x]_{origin\ pixel,\ GPS}$ in (7.3.2).
dblMinOriginPinGZ, dblMaxOriginPinGZ, dblStepOriginPinGZ	Lower bound, upper bound and resolution of values tested for $[z]_{origin\ pixel,\ GPS}$ in (7.3.2).
sngMinRotGtoP, sngMaxRotGtoP, sngStepRotGtoP	Lower bound, upper bound and resolution of values tested for $\theta_{GPS \dashrightarrow pixel}$ in (7.3.2).
sngMinScaleGtoPX, sngMaxScaleGtoPX, sngStepScaleGtoPX	Lower bound, upper bound and resolution of values tested for $[x]_{Scale\ GPS \dashrightarrow pixel}$ in (7.3.2).
sngMinScaleGtoPZ, sngMaxScaleGtoPZ, sngStepScaleGtoPZ	Lower bound, upper bound and resolution of values tested for $[z]_{Scale\ GPS \dashrightarrow pixel}$ in (7.3.2).

E.1.3) Table tbdMisc2

This table stores final optimal transformation parameters that convert between GPS coordinates and pixel coordinates of the Ring Road paper plot.

Field Name	Definition
strMapPath	Absolute file path specifying the location of the Ring Road image file that represents the centre line, on secondary storage.
strMapFile	Filename of Ring Road centre line image.
dblOriginPinGX, dblOriginPinGZ	(x, z) coordinates of paper plot's origin when represented in GPS coordinate system, relative to GPS offset. These fields represent $[x\ z]_{origin\ pixel,\ GPS}$ in (7.3.1).
intFirstPointPX, intFirstPointPZ	(x, z) coordinates of paper plot's starting pixel, in pixel coordinates, when defining minor and major points around the centre line (see [7.3.2]).
intLineMinorLengthP	Distance between minor points on centre line, in pixels.
intNumMajorPoints	Number of major points created for the purpose of speeding up search for the nearest centre line pixel to a given feature.
sngGPSCentreGZ	Centre latitude of virtual world, in GPS coordinate system 1.
sngGPSOffsetGX, sngGPSOffsetGZ	Coordinates of a longitudinal/ latitudinal offset utilised to reduce magnitude of coordinates and localise the coordinate system. Specified in GPS coordinate system 1 here and converted to equatorial GPS coordinates in code, based on the value of sngGPSCentreGZ.
sngLaneWidthE	Half the width of a Ring Road sign region, in equatorial GPS units. This field represents L_{lane} in (7.4.1).
sngMaxRegionLengthE	Maximum sign region length, L_{max}, in equatorial GPS units.
sngMinRegionLengthE	Minimum sign region length, L_{min}, in equatorial GPS units.

sngRotGtoP	Clockwise angle of rotation that maps the axes of the GPS coordinate system onto the axes of the paper plot's coordinate system, such that the axes of both systems coincide. This field represents $\theta_{GPS \rightarrow pixel}$ in (7.3.1).
sngScaleGtoPX sngScaleGtoPZ	Number of pixels in paper plot's coordinate system per x/z unit in GPS coordinate system; that is, per longitudinal/ latitudinal minute. These fields represent $[x]_{Scale\ GPS \rightarrow pixel}$ and $[z]_{Scale\ GPS \rightarrow pixel}$ in (7.3.1), respectively.
sngSignPrecedenceE	Distance measured along centre line preceding a feature where the corresponding sign region ends (where signs are displayed). This field represents $L_{precedence}$ in (7.4.1).

E.2) Field Definitions for Source World Database (Database DB2)

The implementation of source world database DB2 for the VRS prototype was shown by Figure 5.10 in (5.4.1). The table and field names shown in this figure were simplified to make them more readable. The actual names of tables and fields implemented for the VRS prototype are given in this section.

E.2.1) Table tbdRegionMisc

This table stores miscellaneous information about a virtual world and sign regions in records consisting of two text fields, *strFieldName2* and *strFieldValue2*, as described below. Description of the coordinate systems may be found in Appendix A.

strFieldName2	strFieldValue2
	The four values below define the entire rectangular space of a given virtual world. Specified in equatorial GPS coordinates relative to an equatorial GPS offset:
dblWorldMinX dblWorldMinZ dblWorldMaxX dblWorldMaxZ	Lower longitudinal bound of virtual world. Lower latitudinal bound of virtual world. Upper longitudinal bound of virtual world. Upper latitudinal bound of virtual world.
sngGPSCentreGZ	Centre latitude of virtual world, in GPS coordinate system 1.
sngGPSOffsetGX, sngGPSOffsetGZ	Coordinates of a longitudinal/ latitudinal offset utilised to reduce the magnitude of coordinates (such as sign region centre coordinates) and localise the coordinate system. Specified in GPS coordinate system 1 here and converted to equatorial GPS coordinates in code, based on the value of sngGPSCentreGZ.

E.2.2) Table tbdRegionRect

This table corresponds to table *Rectangle Region* in Figure 5.10 of (5.4.1). It stores definitions of all rectangle sign regions (refer to Figure 4.2.a of [4.2.1]). All coordinates and lengths are defined in the equatorial GPS coordinate system (see Appendix A).

Field Name	Definition
lngRegionKey3	Primary key field of sign region.
dblCentreX3	X coordinate of rectangle centre, relative to equatorial GPS offset X.
dblCentreZ3	Z coordinate of rectangle centre, relative to equatorial GPS offset Z.
sngTheta3	Angle of rotation of rectangle, range 0-360 degrees, major axis is always line 2 (see [C.1]).
sngLengthPara3	Length of two parallel sides.
sngLengthPerp3	Length of two perpendicular sides.

E.2.3) Table tbdRegionArc

This table corresponds to table *Arc Region* in Figure 5.10 of (5.4.1). It stores definitions of all arc sign regions (refer to Figure 4.2.b of [4.2.1]). All coordinates are defined in the equatorial GPS coordinate system (see Appendix A).

Field Name	Definition
lngRegionKey0	Primary key field of sign region.
dblCentreX0	X coordinate of arc centre, relative to equatorial GPS offset X.
dblCentreZ0	Z coordinate of arc centre, relative to equatorial GPS offset Z.
sngRadius10	Inner radial length of arc.
sngRadius20	Outer radial length of arc.
sngThetaACL0	Start (anticlockwise) angle of arc region.
sngThetaCL0	End (clockwise) angle of arc region.
bytMajorAxis0	Specifies which end of the arc defines the major sign axis. 1 = anticlockwise end, 2 = clockwise end.

E.2.4) Table tbdRegionComp

This table corresponds to table *Composite Region* in Figure 5.10 of (5.4.1). It stores definitions of all composite sign regions. A composite region is defined by specifying the region key and region type of primitive rectangle and arc regions (stored in tables tbdRegionRect and tbdRegionArc respectively) that comprise the composite region.

Field Name	Definition
lngRegionKey11	Primary key field of composite sign region.
bytRegionNum11	Unique key number of a primitive sign region that makes up this composite sign region.
bytRegionKeyMajor11	Major primary key field value of a primitive region (region type).
lngRegionKeyMinor11	Minor primary key field value of a primitive region (region key).

E.2.5) Table tbdRegionRectSign

This table corresponds to table *Rectangle-Sign* in Figure 5.10 of (5.4.1). It defines which signs appear in which rectangle sign regions.

Field Name	Definition
lngRegionKey4	Primary key field of one sign region in table tbdRegionRect.
intSignKeyMajor4	Major primary key field value of a sign in table tbdSign to be associated with this sign region.
intSignKeyMinor4	Minor primary key field value of a sign in table tbdSign to be associated with this sign region.
bytPosition4	Requested display position of sign on-screen in a strip rectangle: See (6.2) for a description of strip rectangle positions. A position of 0 indicates that a sign has no preferred display position.

E.2.6) Table tbdRegionArcSign

This table corresponds to table *Arc-Sign* in Figure 5.10 of (5.4.1). It defines which signs appear in which arc sign regions.

Field Name	Definition
lngRegionKey1	Primary key field of one sign region in table tbdRegionArc.
intSignKeyMajor1	Major primary key field value of a sign in table tbdSign to be associated with this sign region.
intSignKeyMinor1	Minor primary key field value of a sign in table tbdSign to be associated with this sign region.
bytPosition1	Requested display position of sign on-screen in a strip rectangle, as for table tbdRegionRectSign.

E.2.7) Table tbdRegionCompSign

This table corresponds to table *Composite-Sign* in Figure 5.10 of (5.4.1). It defines which signs appear in which composite sign regions.

Field Name	Definition
lngRegionKey12	Primary key field of one sign region in table tbdRegionComp.
intSignKeyMajor12	Major primary key field value of a sign in table tbdSign to be associated with this sign region.
intSignKeyMinor12	Minor primary key field value of a sign in table tbdSign to be associated with this sign region.
bytPosition12	Requested display position of sign on-screen in a strip rectangle, as for table tbdRegionRectSign.

E.2.8) Table tbdSign

This table corresponds to table *Sign* in Figure 5.10 of (5.4.1). It defines the colour, shape and angle of rotation of all virtual road signs, plus data used for memory caching.

Field Name	Definition
intSignKeyMajor5	Major primary key field value of sign.
intSignKeyMinor5	Minor primary key field value of sign.
bytColourStd5	Standard colour of sign backboard: 0 = use RGB colour as specified by RGB fields, 1 = transparent (no colour), 2 = red, 3 = green, 4 = blue, 5 = black, 6 = white.
bytColourRed5 bytColourGreen5 bytColourBlue5	If bytColourStd5 = 0, these fields define the red, green and blue component of a custom RGB colour of the sign backboard.
bytShape5	Standard shape of sign backboard: 0 = same shape as rectangular screen region in which it is displayed (this shape has a maximum possible size) 1 = 1.5 x 1 rectangle 2...9 = 2...9 x 1 rectangle 10 = equilateral triangle (size = $2/\sqrt{3}$x 1) 11 = isosceles triangle 20 = 1 x 1 square 21 = rhombus (major axis = horizontal axis, size = $\sqrt{3}$x 1) 30 = hexagon (size = 2 x 1) 40 = octagon 50 = circle 61-69 = 1.5 x 1, 2-9 x 1 rectangle with tapered left-hand side (pointing left). 71-79 = 1.5 x 1, 2-9 x 1 rectangle with tapered right-hand side (pointing right).
intRotation5	Angle of rotation away from the vertical, in degrees, that sign is displayed at.
dtmFirstCreated5	Date and time when this sign is first stored in cache memory for a given session. This field corresponds to t_{first} in (5.2.3).
intSessionCount5	Number of times this sign has been displayed for a given session. This field corresponds to n_{disp} in (5.2.3).

E.2.9) Table tbdSignImage

This table corresponds to table *Sign-Image* in Figure 5.10 of (5.4.1). All coordinates and lengths are defined in the sign's local coordinate system.

Field Name	Definition
intSignKeyMajor6	Major primary key field value of sign.
intSignKeyMinor6	Minor primary key field value of sign.
lngImageKey6	Primary key field value of one image in table tbdImage to be displayed on this sign.
sngCoordX6	Horizontal (X) coordinate of bottom left corner of image.
sngCoordY6	Vertical (Y) coordinate of bottom left corner of image.
sngWidth6	Width of image when displayed on sign.
sngHeight6	Height of image when displayed on sign.

E.2.10) Table tbdImage

This table corresponds to table *Image* in Figure 5.10 of (5.4.1). It defines the filename of each sign image plus data used for memory caching.

Field Name	Definition
lngImageKey9	Primary key field of sign image.
strFile9	Filename of sign image, defined relative to path stored by strRelImagePath field value stored in table tbdSignMisc (see below).
dtmFirstCreated9	Date and time when this image is first stored in cache memory for a given session. This field corresponds to t_{first} in (5.2.3).
intSessionCount9	Number of times this image has been displayed for a given session. This field corresponds to n_{disp} in (5.2.3).

E.2.11) Table tbdSignText

This table corresponds to table *Sign-Text* in Figure 5.10 of (5.4.1). It stores the textual message of all signs. All coordinates and lengths are defined in the sign's local coordinate system.

Field Name	Definition
intSignKeyMajor8	Major primary key field value of sign.
intSignKeyMinor8	Minor primary key field value of sign.
bytTextNum8	Primary key field describing text line number: range 1-4.
strTextEnglish8	Sign text, in English.
strText*8	Sign text, in an alternate language specified by *.
sngCoordX8	Horizontal (X) coordinate of lower left corner of text.
sngCoordY8	Vertical (Y) coordinate of lower left corner of text.
bytPointSize8	Point size of text font (font name is hard-coded in the VRS prototype).
bytColourStd8	Standard colour of text, as for sign backboard (see table tbdSign).

E.2.12) Table tbdSignMisc

This table stores miscellaneous sign information in one record consisting of two text fields, *strFieldName7* and *strFieldValue7*, as described below.

strFieldName7	strFieldValue7
strRelImagePath	Path relative to current path where all sign image files are located on secondary storage.

E.2.13) Table tbdMapMisc

This table stores miscellaneous information about the Ring Road centre line paper plot that is plotted in the map view on the prototype controller form (see [6.2.3]), plus coordinate system transformation parameters. It stores data in records consisting of two text fields, *strFieldName10* and *strFieldValue10*. All of the fields in this table are also contained in table tbdMisc2 in feature database DB1. To ensure consistency, the field values were manually copied from database DB1 to the identical fields in this table (where applicable).

E.3) Field Definitions for Compiled World Database (Database DB3)

The logical data structure of records in compiled world database DB3 was shown in (4.4.7). This section describes the implementation of this logical structure for the VRS prototype.

E.3.1) Table tbdNonLeafNode

This table stores all non-leaf nodes in the compiled quadtree and the mappings between them.

Field Name	Definition
lngNodeKey2	Primary key field of non-leaf quadtree node.
lngChild1Num2, lngChild2Num2, lngChild3Num2, lngChild4Num2	Primary key field values of all children of this non-leaf node. A positive number indicates that a child is a non-leaf node, a negative number indicates that a child is a leaf node, and a value of zero indicates that a child does not exist.

E.3.2) Table tbdLeafNode

This table stores all leaf nodes in the compiled quadtree and the list of sign regions associated with each node.

Field Name	Definition
lngNodeKey1	Primary key field of leaf quadtree node.
intRegionNum1	Unique key number of a sign region that exists in this node's region list.
bytRegionKeyMajor1	Major primary key field value of a sign region (region type).
lngRegionKeyMinor1	Minor primary key field value of a sign region (region key).

APPENDIX F
Prototype Implementation

This appendix describes specific issues of implementation concerning how the VRS software was designed and created. The development platform is first described, and a summary of the class modules follows. This summary is intended to show how the various components of the positioning and tracking subsystem, data subsystem and display subsystem were combined to form the prototype. It is intended as a possible starting point for those who are contemplating the implementation of their own VRS system.

F.1) Hardware & Software

A Pentium-II PC running the Microsoft Windows 95 operating system was the primary prototype development platform. This was equipped with a 3D accelerator card to optimise OpenGL rendering in hardware. An older Pentium laptop PC was used for in-vehicle data collection, as described in Chapter 7.

Microsoft Visual Basic 5.0 was chosen as the primary programming environment because it supports *rapid application development* (RAD paradigm), and its visual design tools made it easy to design and develop the prototype GUI. Moreover, the author's knowledge and experience of Visual Basic far outweighed any other available development software. The modular nature of the prototype within its subsystems meant object-oriented programming (OOP) was well-suited, yielding better decoupling of modules, scope control, and a more natural design than a non-OOP implementation. A single *dynamic-linked library* (DLL) file provided support for the OpenGL API in Visual Basic and was retrieved free of charge from the World Wide Web (Unknown author B, 2001).

Two other primary programming environments were identified as possible options for development at the project outset: Microsoft Visual C++ and Borland Delphi. However, given the author's limited knowledge and experience of these environments, they were never seriously considered. Had the prototype been developed using one of these alternative environments, the total time required for completion would probably have exceeded twice that of the actual time taken in practice, and thus the prototype would never have been completed on time. Mathwork's Matlab 5.2 was chosen as a secondary programming environment owing to its extensive, specialised math libraries (toolboxes) and matrix support. Matlab was used for experimental testing purposes of specific prototype modules during the design phase. The Matlab software routines were converted to Visual Basic code as the prototype progressed.

F.2) Class Summary & Integration

F.2.1) Overview

This section briefly describes the set of class modules that comprise the VRS prototype. Figure F.2 shows the prototype class hierarchy. This illustrates how the class modules are integrated together to form the VRS prototype. Figure F.1.a describes the notation used in this figure, which adheres to the standard notation described by Pressman (1997). Figure F.1.b describes one example of this notation.

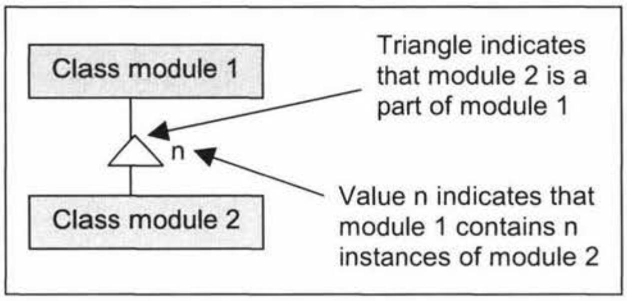

Class module 1
Triangle indicates that module 2 is a part of module 1
n
Value n indicates that module 1 contains n instances of module 2
Class module 2

Figure F.1.a: Explanation of notation used in Figure F.2

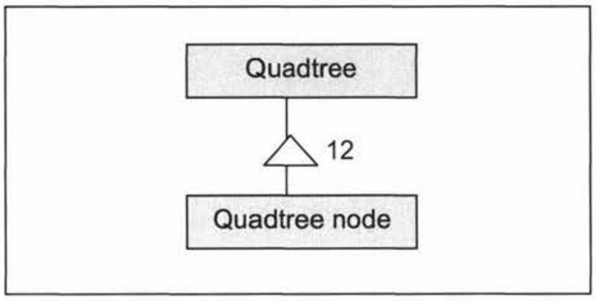


Figure F.1.b: Example of module notation. Here a quadtree is composed of 12 quadtree nodes.

A brief description of each class module is given below. The modules are organised into four groups: each group approximately corresponds to one subsystem described in Chapters 3-6 respectively (in practice the boundaries are not sharply defined). Where applicable, references to relevant sections of this thesis are included in the description of each module. Code listings of all class modules may be found on the CD-R disc contained in the back pocket of this thesis.

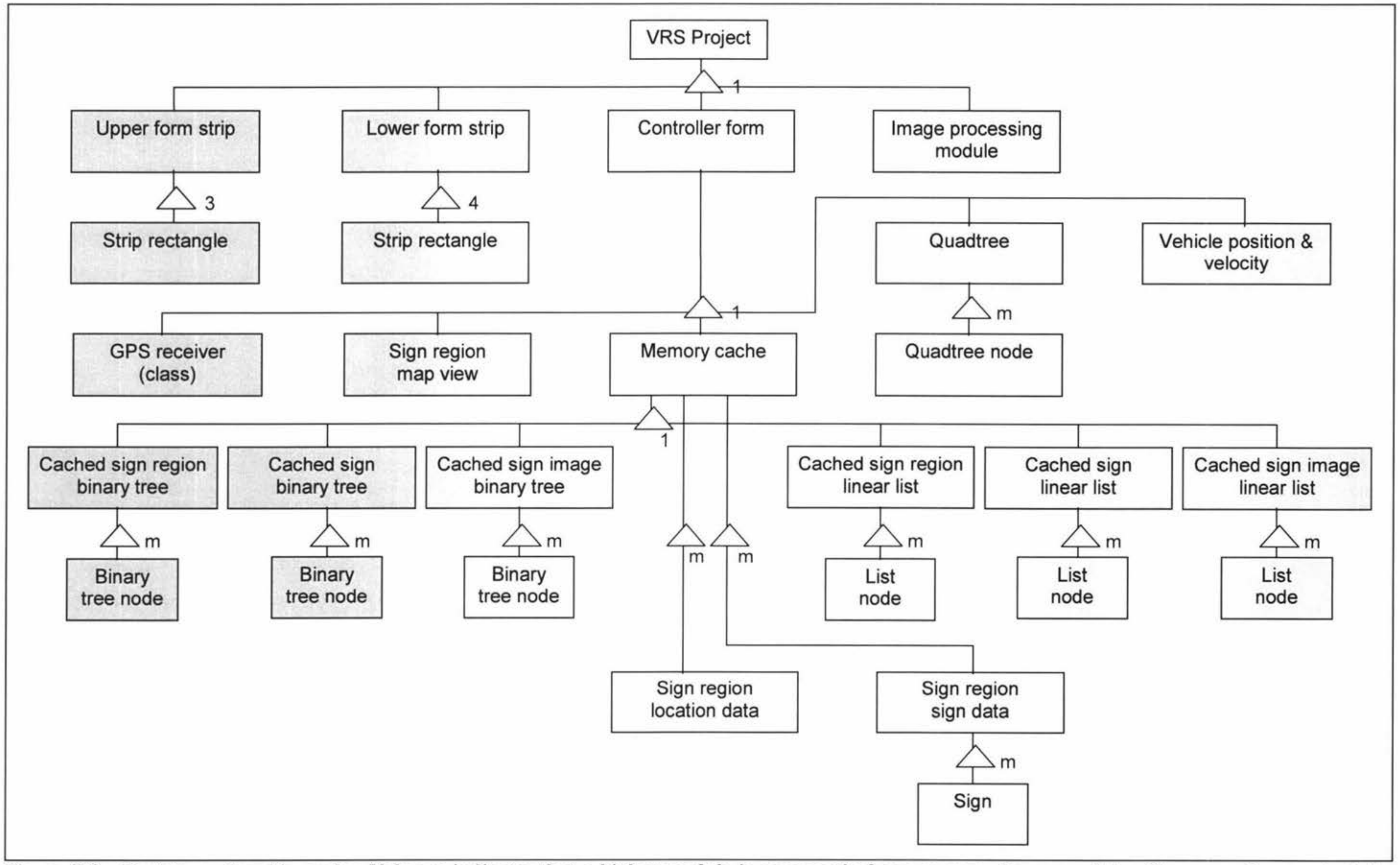


Figure F.2: Prototype class hierarchy. Value m indicates that a higher module is composed of one or many lower modules. For example, a cached sign linear list is composed of one or many list nodes.

F.2.2) Positioning & Tracking Subsystem

Table F.1 summarises the two class modules for this subsystem:

Module Name	Primary Tasks	Reference to Section
GPS receiver	Processes (parses) NMEA data.	(3.4)
	Decodes NMEA data, and provides useful information (vehicle position and velocity) to other modules.	(3.4)
Vehicle position and velocity	Stores current position and velocity of vehicle.	
	Converts between GPS coordinate system and equatorial GPS coordinate system.	Appendix A
	Filters received GPS data and predicts future vehicle position and velocity, using Kalman filter.	(3.5)
	Displays data received from GPS unit and filtered data on controller form.	(6.2)
	High resolution timing (to drive screen updates).	(2.3.4)

Table F.1: Summary of classes in positioning and tracking subsystem

F.2.3) Data Subsystem: Virtual Worlds

Table F.2 summarises the eight class modules for this part of the data subsystem:

Module Name	Primary Tasks	Reference to Section
Quadtree	Caches the sequence of nodes stored in compiled world database leading to the leaf quadrant that contains the current vehicle position.	(4.5.2)
	Stores boundaries of virtual world.	
	Plots all visible sign regions in map view on controller form.	(6.2)
Quadtree node	Primitive element of quadtree data structure.	(4.4.4)
Memory cache	Creates sign region cache, sign cache and sign image cache in local memory.	(4.5.2), (5.2.2)
	Returns the active list of sign regions $L_{active\ SR}$ for the current vehicle position and velocity.	(4.5.1)
	Displays sign region information for all active sign regions on controller form.	(6.2)
Sign region location data	Stores geometrical parameters of one sign region.	(4.2)
	Determines whether this sign region is currently active or not.	(4.1)
	Plots sign region in map view on controller form as this region changes state (active/ inactive).	(6.2)
Cached sign region binary tree	Stores set of sign regions currently cached in local memory. Tree structure supports an efficient search for any given sign region.	(4.6.2)
Binary tree node	Primitive element of binary tree data structure. In this context, each node represents one cached sign region.	(4.6.2)

Cached sign region linear list	Tracks the age of sign regions stored in the cache; that is, determines which sign region should be next removed from the cache, according to a first-in, first-out (FIFO) replacement algorithm.	(4.6.2)
List node	Primitive element of linear list data structure. In this context, each node represents one cached sign region.	(4.6.2)

Table F.2: Summary of classes in data subsystem concerned with virtual world management

F.2.4) Data Subsystem: Virtual Signs

Table F.3 summarises the eight class modules for this part of the data subsystem:

Module Name	Primary Tasks	Reference to Section
Sign region sign data	Defines set of signs associated with one sign region. Calculates distance between each sign and current vehicle position.	(5.2.2) (5.1.1)
Sign	Compiles set of elements (backboard, images and text) of one sign into one image, stored as a display list. Renders sign on-screen in a strip rectangle.	(5.2) (6.2)-(6.4)
Cached sign binary tree	Stores set of signs currently cached in local memory. Tree structure supports an efficient search for any given sign.	(5.3.2)
Cached sign image binary tree	Stores set of sign images currently cached in local memory. Tree structure supports an efficient search for any given sign image.	(5.3.2)
Binary tree node	Primitive element of binary tree data structure. In this context, each node in the sign binary tree represents one cached sign, and each node in the sign image binary tree represents one cached sign image.	(5.3.2)
Cached sign linear list	Records the position of signs stored in the cache. A sign at a high cache position is less likely to be removed from the cache anytime soon than a sign at a low cache position. A sign at a position high in the cache has a low mean time between display events, whereas a sign low in the cache has a high mean time between display events. When a sign is to be removed from cache, the list identifies which sign (at the lowest cache position) should be removed from the cache.	(5.3.2)
Cached sign image linear list	Records the position of sign images stored in the cache, as for cached signs. When a sign image is to be removed from cache, the list identifies which image (at the lowest cache position) should be removed from the cache.	(5.3.2)
List node	Primitive element of linear list data structure. In this context, each node in the sign list represents one cached sign, and each node in the sign image list represents one cached sign image.	(5.3.2)

Table F.3: Summary of classes in data subsystem concerned with virtual sign management

F.2.5) Display Subsystem

Table F.4 summarises the six class modules for this subsystem:

Module Name	Primary Tasks	Reference to Section
Controller form	Application initialisation and termination.	
	Allocates signs to strip rectangles.	(6.5)
	Presents current state of application to the user, such as information describing currently active sign regions, data received from GPS unit, current position and velocity of vehicle, and current source of input (keyboard and mouse/ data log file/ real GPS receiver).	(6.2)
	Processes directions given by user. For example, if the input source is keyboard and mouse, the user dictates the current position and velocity of the simulated vehicle.	(6.2)
Upper form strip	OpenGL initialisation and termination for three strip rectangles on this form.	(6.3)
	Presentation of up to three signs (concurrently) on this form.	(6.2)
Lower form strip	OpenGL initialisation and termination for four strip rectangles on this form.	(6.3)
	Presentation of up to four signs (concurrently) on this form.	(6.2)
Strip rectangle	OpenGL initialisation of the screen area where one sign will be displayed.	(6.3)
	Renders one sign contained within the screen area defined by this strip rectangle.	(6.2)-(6.4)
Sign region map view	Presents a plan view of all sign regions visible in the current view (as chosen by the user), as well as the current vehicle position and velocity. Road centre lines may also be plotted.	(6.2)
	Converts between equatorial GPS coordinate system and screen coordinate system.	Appendix A
Image processing module	Reads in an image file that represents the centre lines of one or more roads. Centre lines are shown in the map view on the controller form.	

Table F.4: Summary of classes in display subsystem

APPENDIX G
Supplementary Ring Road Results

This appendix presents additional results and workings for the Massey University Ring Road test case described in Chapter 7.

G.1) Collected GPS Coordinates of Ring Road Features

	Raw GPS Coordinates*		Side of road on which feature exists	Features Corresponding to (x, z) Coordinates
	x Coord: Longitude	z Coord: Latitude		
1	37.060	-22.956	Left	General parking
			Right	Bicycle parking
2	37.113	-22.983	Right	Reserved parking
3	37.150	-23.006	Right	Bus stop
4	37.160	-23.011	Right	College of Business, loading zone/ access
5	37.197	-23.021	Right	Motorcycle parking
6	37.237	-23.049	Right	Time restricted parking
7	37.263	-23.085	Right	Refectory building, Pink building, Refectory road, reserved & mobility parking
8	37.295	-23.209	Right	Bicycle parking
9	37.288	-23.224	Right	Rehab road, mobility & motorcycle parking
10	37.272	-23.242	Right	Residential services, bus stop
11	37.257	-23.265	Left	Recreation centre, Orchard road, general parking
			Right	Science towers, reserved parking
12	37.241	-23.283	Left	Ecology building
			Right	Facilities Management Unit
13	37.223	-23.296	Right	Loading zone/ access, motorcycle parking
14	37.208	-23.307	Left	Wool building, bus stop
15	37.172	-23.320	Right	Riddet, Ag Hort Sciences building, Riddet road, motorcycle & mobility parking
16	37.150	-23.327	Right	Landcare Research
17	37.137	-23.330	Left	Practical Teaching Complex
18	37.114	-23.330	Left	Albany drive
19	37.087	-23.330	Left	Printery
20	37.039	-23.318	Left	Boiler house, reserved parking
21	37.000	-23.299	Right	Ag Engineering building, loading zone/ access, reserved parking
22	36.970	-23.277	Right	Vet building
23	36.939	-23.244	Right	Time restricted parking
24	36.924	-23.229	Right	Bus stop
25	36.907	-23.207	Right	Library, Social Science Lecture Block, Library road, time restricted & reserved parking
26	36.886	-23.165	Left	Monro Hill exit (South only)
27	36.885	-23.148	Left	Motorcycle parking
28	36.934	-23.095	Left	Wharewata, loading zone/ access, reserved parking
29	36.956	-23.082	Right	Social Science Tower
30	36.970	-23.078	Right	Loading zone/ access

31	36.977	-23.073	Right	Geography building
32	36.984	-23.064	Right	Registry & information centre, Turitea road, time restricted parking
33	37.013	-22.990	Left	International Students Centre, Commercial Complex, bicycle & time restricted
34	37.011	-22.951	N/A	Fork in Ring Road

Table G.1: Recorded GPS coordinates of 67 features around Massey University Ring Road.
* Relative to an offset of latitude 40° South, longitude 175° East. For example, true global GPS coordinates of feature 1 are latitude 40° 22.956 minutes South, longitude 175° 37.060 minutes East.

Figure G.2 (see [G.3] below) shows a scale plot of these feature coordinates in relation to the Ring Road centre line.

G.2) Design Conventions of Ring Road Virtual Signs

Sign text font:	Franklin Gothic Heavy
Sign text colour:	RGB = 249, 243, 218 = standard colour 20

Sign Category	Sign Major Key	Backboard Colour	Form strip display position	Shape
Parking	1	Deep green (RGB = 5, 102, 8) = Standard colour 30	Lower strip	Hexagon
Bus stop	2	Deep red (RGB = 127, 6, 6) = Standard colour 31	Lower strip	Rhombus
Buildings and facilities	3	Deep blue (RGB = 6, 56, 135) = Standard colour 32	Upper strip	2 x 1 rectangle
Road names	4	Deep purple (RGB = 93, 6, 127) = Standard colour 33	Upper strip	2 x 1 rectangle

Table G.2: Design conventions of four categories of virtual road signs erected around Massey University Ring Road.

G.3) Scale Ring Road Plot and Fitted Feature Coordinate Set

Figure G.1 (p. G-3) is a scale diagram of the Ring Road, originally drawn by a plotter. The marked Ring Road centre line was drawn on the plot by hand. The diagram was then scanned into a PC and used to find the best fit of all features around the Ring Road, as described in (7.3). Figure G.2 (p. G-4) shows the skeletonised centre line of the Ring Road and measured GPS coordinates of all features fitted to the centre line (using the best parameter set found by the brute-force search described in [7.3.2]). In addition, the set of major points utilised to speed up the search for the nearest point on the centre line to each feature (see [7.3.2]) is shown.

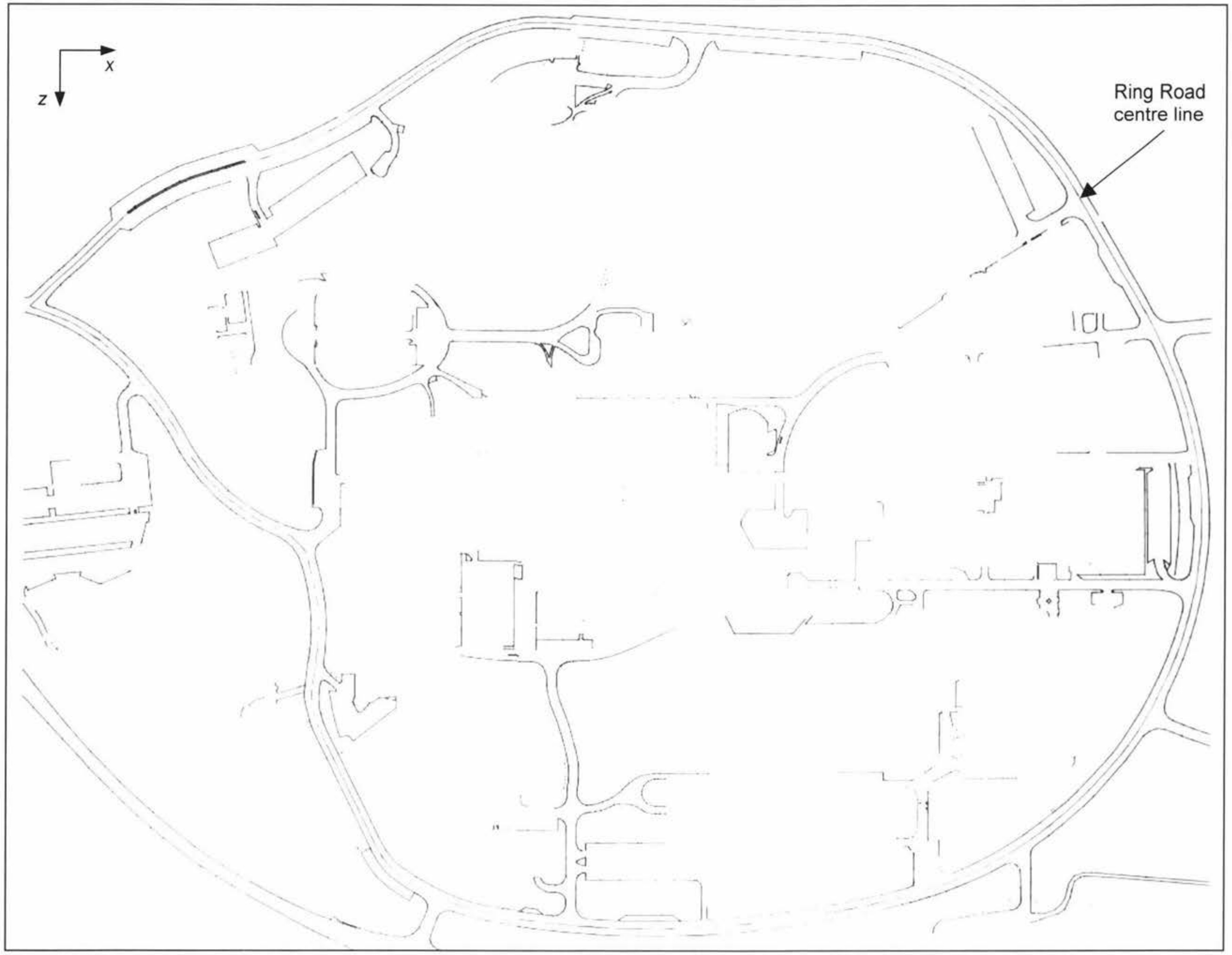


Figure G.1: **Scale plot of Massey University Ring Road and other minor roads, showing road outline and red Ring Road centre line (latter was added manually).**

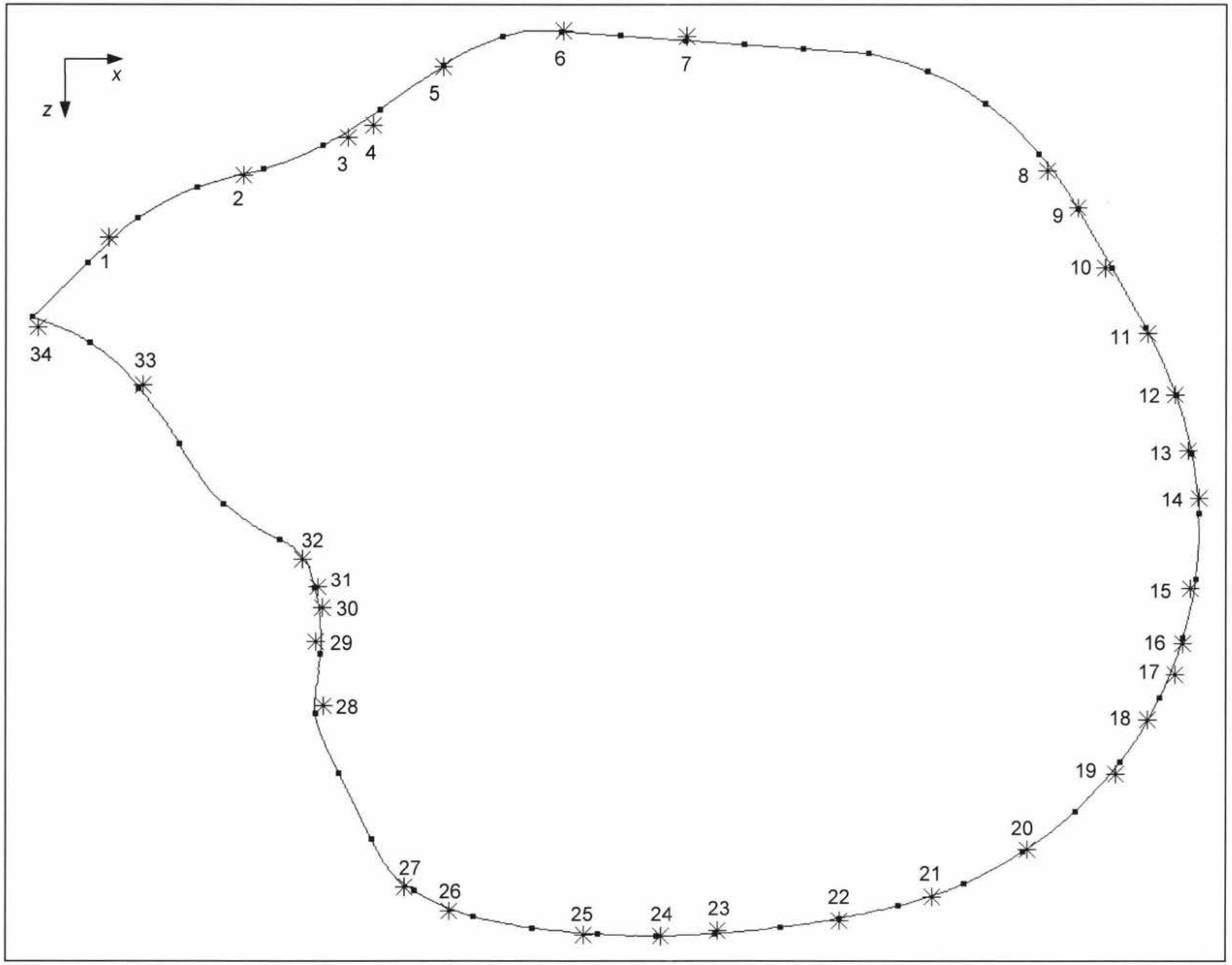


Figure G.2: Scale plot of skeletonised Ring Road centre line, measured GPS coordinates of all features (see Table G.1) and major points.

✳ Measured feature

▪ Major point

G.4) Estimation of Starting Parameters for Minimisation Algorithm

In (7.3.2), the optimum values of five parameters that completely define the transformation between coordinate systems 1 and 3 (corresponding to GPS and pixel coordinates of the Ring Road paper plot respectively, see Appendix A) were found by a brute-force search. The optimum transformation parameters fitted the measured set of Ring Road features (in GPS coordinates) to the Ring Road centre line (in pixel coordinates) such that the sum of squared error was minimised. This section explains how the set of parameters used to start the brute-force search was estimated.

One simple method of estimating the parameter set is to choose two or more feature coordinates that are separated by a large distance in both GPS and paper plot pixel coordinates. The pixel coordinates of each feature can be estimated by eye by looking at another scale map of the Massey University campus (such as an aerial photograph) and correlating it with the map shown in Figure G.1. Obviously, the features must be clearly visible on the former scale map. Starting values of the five parameters can then be solved by finding those values that minimise the sum of squared error, SSE, over the small set of chosen features. The error per feature is the distance between the estimated coordinates and calculated coordinates of the feature, in pixel coordinates.

For the purpose of estimation, a non-iterative technique is described here. Eqs. G.1 and G.2.a-b permit estimation of $\theta_{GPS \dashrightarrow pixel}$, given two points P1 and P2 representing two features:

$$\theta_{GPS\text{-->}pixel} = \theta_{GPS} - \theta_{pixel}$$

Eq. G.1: Estimation of $\theta_{GPS \text{-->} pixel}$, where:

$$\theta_{GPS} = \tan^{-1}\left(\frac{\Delta x_{GPS}.[x]_{Scale\ GPS\text{-->}pixel}}{\Delta z_{GPS}.[z]_{Scale\ GPS\text{-->}pixel}}\right) + \theta_{k,GPS}$$

Eq. G.2.a: Definition of θ_{GPS} in Eq. G.1 where:

$$\theta_{pixel} = \tan^{-1}\left(\frac{\Delta x_{pixel}}{-\Delta z_{pixel}}\right) + \theta_{k,pixel}$$

Eq. G.2.b: Definition of θ_{pixel} in Eq. G.1, where:

θ_{GPS}	=	angle made by P2 relative to P1 in GPS coordinate system 1, where an angle of 0^o is aligned with the z axis of this system.
θ_{pixel}	=	angle made by P2 relative to P1 in pixel coordinate system 3, where an angle of 0^o is aligned with the z axis of this system.

$\theta_{k,GPS}$, $\theta_{k,pixel}$ =	constants calculated to accommodate the limited range of (–90° to 90°) returned by the arctangent function.

$\Delta x_{GPS} = (x_{GPS,P2} - x_{GPS,P1})$, $\Delta z_{GPS} = (z_{GPS,P2} - z_{GPS,P1})$,

$\Delta x_{pixel} = (x_{pixel,P2} - x_{pixel,P1})$, $\Delta z_{pixel} = (z_{pixel,P2} - z_{pixel,P1})$, where:

$(x_{GPS,P1}, z_{GPS,P1})$ =	GPS coordinates of point P1.
$(x_{pixel,P1}, z_{pixel,P1})$ =	pixel coordinates of point P1.
$(x_{GPS,P2}, z_{GPS,P2})$ =	GPS coordinates of point P2.
$(x_{pixel,P2}, z_{pixel,P2})$ =	pixel coordinates of point P2.

Note that the dimensions of the numerator and denominator in the brackets of Eqs. G.2.a-b are pixel units because in transforming between coordinate systems 1 and 3, the rotation operation is performed on a point defined in pixel coordinate system 3.

In all but two cases, Eq. G.2.a shows that $\theta_{GPS\text{-->}pixel}$ cannot be calculated without estimation of $[x\ z]_{Scale\ GPS\text{-->}pixel}$. However, if we choose two points P1 and P2 such that $\Delta x_{GPS} = 0$ or $\Delta z_{GPS} = 0$, then $\theta_{GPS} = 0^{\circ}$, 90°, 180° or 270° and thus knowledge of $[x\ z]_{Scale\ GPS\text{-->}pixel}$ is not necessary. For the Ring Road, two measured points that satisfy $\Delta x_{GPS} = 0$ are P1 = feature (3) and P2 = feature (16) (see Table G.1 of [G.1] and Figure G.2 of [G.3]). For these points, the arctangent function in Eq. G.2.a returns 0°. P2 has a more negative latitude than P1 so $\theta_{GPS} = 180^{0}$ and thus $\theta_{k,GPS} = 180^{\circ}$. Δx_{pixel} and Δz_{pixel} were measured by estimating the position of points P1 and P2 on the original paper plot and noting the horizontal and vertical distance between points respectively. The measured distances (in centimetres) were converted to pixels by noting that the original paper plot was scanned at a resolution of 75 dots per inch (dpi), and the conversion of one inch equals 2.54 centimetres. The calculated value of θ_{pixel} is shown by Eq. G.3:

$$\theta_{pixel} = \tan^{-1}\left(\frac{25.5\ \text{cm}.75\ \text{dpi}/(2.54\ \text{cm}/\text{inch})}{-15.4\ \text{cm}.75\ \text{dpi}/(2.54\ \text{cm}/\text{inch})}\right) + \theta_{k,pixel} = -58.9^{\circ} + \theta_{k,pixel}$$

Eq. G.3: Calculation of θ_{pixel} for points P1 and P2, using Eq. G.2.b

From Figure G.2 of (G.3), it can be seen that the angle made by P2 relative to P1 in pixel coordinate system 3 lies between 90 and 180 degrees. Thus $\theta_{k,pixel} = 180^{\circ}$ and $\theta_{pixel} = 121.1^{\circ}$.

Combining the values of θ_{GPS} and θ_{pixel} found using Eq. G.1 gives $\theta_{GPS \dashrightarrow pixel} = 180^o - 121.1^o \approx 59^o$.

To estimate $[x\ z]_{Scale\ GPS \dashrightarrow pixel}$, the approximate x and z axes of GPS coordinate system 1 were drawn by hand on the original paper plot at an angle of 59^o (estimated $\theta_{GPS \dashrightarrow pixel}$) anticlockwise from the x and z axes of the paper plot. The position of the origin point, P_{origin}, was obviously known for the pixel coordinate system ([x, z] = [0, 0]); this point was very near the top-left corner of the paper plot (the plotted diagram did not extend to the exact edge of the page). The approximate GPS axes were correctly drawn so as to intersect at this origin point. For two points P1 and P2, Eqs. G.4.a-b yield the [x z] scale conversion factors:

$$[x]_{Scale\ GPS\dashrightarrow pixel} = \frac{\Delta x_{pixel,parallel}}{\Delta x_{GPS}}$$

Eq. G.4.a: Calculation of $[x]_{Scale\ GPS\dashrightarrow pixel}$, where:

$$[z]_{Scale\ GPS\dashrightarrow pixel} = \frac{\Delta z_{pixel,parallel}}{\Delta z_{GPS}}$$

Eq. G.4.b: Calculation of $[z]_{Scale\ GPS\dashrightarrow pixel}$, where:

$\Delta x_{GPS} = (x_{GPS,\ P2} - x_{GPS,\ P1})$, $\Delta z_{GPS} = (z_{GPS,\ P2} - z_{GPS,\ P1})$, as above.

$\Delta x_{pixel,\ parallel} =$	signed distance between points P1 and P2, measured parallel to x axis of GPS coordinate system 1, in pixels.
$\Delta z_{pixel,\ parallel} =$	signed distance between points P1 and P2, measured parallel to z axis of GPS coordinate system 1, in pixels.

As for the earlier estimation of $\theta_{GPS \dashrightarrow pixel}$, $\Delta x_{pixel,\ parallel}$ and $\Delta z_{pixel,\ parallel}$ were measured in centimetres and converted to pixels. Actual scale calculations are shown by Eqs. G.5.a-b for P1 = feature (6) and P2 = feature (25) (see Figure G.2 of [G.3]):

$$[x]_{Scale\ GPS\dashrightarrow pixel} = \frac{-23.6\text{cm. } 75\text{dpi} / (2.54\text{ cm/inch})}{-0.330} = 2112 \text{ pixels/ GPS minute.}$$

$$[z]_{Scale\ GPS\dashrightarrow pixel} = \frac{-14.5\text{cm. } 75\text{dpi} / (2.54\text{ cm/inch})}{-0.158} = 2710 \text{ pixels/ GPS minute.}$$

Eq. G.5.a & G.5.b: Calculation of scale factors using Eqs. G.4.a-b

Calculation of $[x\ z]_{origin\ pixel,\ GPS}$ was done according to Eq. G.6.a-b and Eq. G.7.a-b:

$[x]_{\text{origin pixel, GPS}} = x_{\text{GPS, P1}} - \Delta x_{\text{GPS}}$

Eq. G.6.a: Calculation of $[x]_{\text{origin pixel, GPS}}$, where:

$[z]_{\text{origin pixel, GPS}} = z_{\text{GPS, P1}} - \Delta z_{\text{GPS}}$

Eq. G.6.b: Calculation of $[z]_{\text{origin pixel, GPS}}$, where:

$$\Delta x_{\text{GPS}} = \frac{\Delta x_{\text{pixel,parallel}}}{[x]_{\text{Scale GPS-->pixel}}}$$

Eq. G.7.a: Calculation of Δx_{GPS} in Eq. G.6.a, where:

$$\Delta z_{\text{GPS}} = \frac{\Delta z_{\text{pixel,parallel}}}{[z]_{\text{Scale GPS-->pixel}}}$$

Eq. G.7.b: Calculation of Δz_{GPS} in Eq. G.6.b, where:

$(x_{\text{GPS, P1}}, z_{\text{GPS, P1}})$ =	GPS coordinates of point P1, as above.
$\Delta x_{\text{pixel, parallel}}$ =	signed distance between point P1 and origin point P_{origin}, measured parallel to x axis of GPS coordinate system 1, in pixels.
$\Delta z_{\text{pixel, parallel}}$ =	signed distance between point P1 and origin point P_{origin},, measured parallel to z axis of GPS coordinate system 1, in pixels.

Note that Eqs. G.7.a-b are simple rearrangements of Eqs. G.4.a-b. Actual origin point estimations are shown by Eqs. G.8.a-b for P1 = feature (16) and the estimated scale factor values calculated in Eqs. G.5.a-b:

$$[x]_{\text{origin pixel, GPS}} = 37.150 - \frac{1.8\text{ cm}.75\text{ dpi}/(2.54\text{ cm}/\text{inch})}{2112\text{ pixels}/\text{GPS minute}} = 37.125$$

$$[z]_{\text{origin pixel, GPS}} = -23.327 - \frac{-39.8\text{ cm}.75\text{ dpi}/(2.54\text{ cm}/\text{inch})}{2710\text{ pixels}/\text{GPS minute}} = -22.893$$

Eqs. G.8.a-b: Calculation of $[x\ z]_{\text{origin pixel, GPS}}$ using Eqs. G.6.a-b

Comparison of the [five parameters estimated here as starting values] with the final optimal values found in Table G.3 of (G.6) shows that the two parameter sets are in very close proximity.

G.5) Region-Fitting Algorithm

Box G.1 describes one algorithm that determines a suitable sign region configuration for each GPS feature coordinate pair, as summarised in (7.4.4).

1) Convert all points in set $S_{centreline}$ from paper plot coordinate system (3) to equatorial GPS coordinate system (2).

2) **Test whether a single sign region is sufficient**:

   a) Test a single rectangle region
   Fit best rectangle region to point set $S_{centreline}$, using least-squares perpendicular regression. If RMS regression error < $E_{threshold}$, stop, else:

   b) Test a single arc region
   Fit best arc region to point set $S_{centreline}$, using least-squares arc regression. If RMS regression error < $E_{threshold}$, stop, else:

3) **Test whether two adjacent sign regions are sufficient**:

   Break segment $S_{centreline1}$ into two segments to form two point sets, $S_{centreline1}$ and $S_{centreline2}$. In each case (a)-(c), find best combined fit of two regions using an iterative binary search to find where combined RMS error is minimised.

   a) Test a rectangle region followed by an adjacent arc region
   Fit best rectangle region to point set $S_{centreline1}$ and best arc region to point set $S_{centreline2}$ separately. If combined RMS regression error < $E_{threshold}$, stop, else:

   b) Test an arc region followed by an adjacent rectangle region
   Fit best arc region to point set $S_{centreline1}$ and best rectangle region to point set $S_{centreline2}$ separately. If combined RMS regression error < $E_{threshold}$, stop, else:

   c) Test two adjacent arc regions
   Fit best arc region to point set $S_{centreline1}$ and best arc region to point set $S_{centreline2}$ separately. If combined RMS regression error < $E_{threshold}$, stop, else:

4) Flag the centre line segment represented by point set $S_{centreline}$ as unable to be decomposed into one or two regions with an acceptable degree of fit.

Box G.1: Algorithm that determines a suitable sign region configuration for given GPS coordinates of one Ring Road feature.

It must be noted that paper plot coordinate system (3), the system utilised by set $S_{centreline}$, is not suitable for regression. The VRS prototype processes a virtual world in equatorial GPS coordinates, and therefore all points in $S_{centreline}$ must be converted from coordinate system (3) to system (2) prior to regression (step 1 in Box G.1). If this conversion was performed after regression, the different x and z scaling factors of the two coordinate systems would cause all arc regressions to be invalidated. For example, consider a short centre line segment that can be regressed with an RMS error of zero in coordinate system (3); that is, a perfect circular arc can be fitted to the point set $S_{centreline}$. When this point set is converted to system (2), the points are stretched more in one direction than the other, such that the segment appears as an elliptic arc rather than a circular arc. Elliptic arc sign regions are not supported, so the resulting sign region will not fit at all well to the point set.

G.6) Results of Search for Optimal Transformation Parameters

Parameter	**Value Type**	**Iteration number**					
		1	**2**	**3**	**4**	**5**	**6**
$[x]_{\text{origin pixel, GPS}}$ (minutes)	Minimum	36.9	37.04	37.07	37.105	37.120	37.123
	Maximum	37.3	37.24	37.17	37.155	37.140	37.133
	Resolution	0.04	0.02	0.01	0.005	0.002	0.001
	Optimal value found	37.14	37.12	37.13	37.130	37.128	37.128
$[z]_{\text{origin pixel, GPS}}$ (minutes)	Minimum	-23.1	-23	-22.93	-22.915	-22.895	-22.890
	Maximum	-22.7	-22.8	-22.83	-22.865	-22.875	-22.880
	Resolution	0.04	0.02	0.01	0.005	0.002	0.001
	Optimal value found	-22.9	-22.88	-22.89	-22.885	-22.885	-22.885
$[x]_{\text{Scale GPS --> pixel}}$ (pixels per minute)	Minimum	2000	1850	2000	2050	2075	2090
	Maximum	3000	2350	2200	2150	2125	2110
	Resolution	100	50	20	10	5	2
	Optimal value found	2100	2100	2100	2100	2100	2100
$[z]_{\text{Scale GPS --> pixel}}$ (pixels per minute)	Minimum	2000	2650	2650	2760	2755	2770
	Maximum	3000	3150	2850	2860	2805	2790
	Resolution	100	50	20	10	5	2
	Optimal value found	2900	2750	2810	2780	2780	2778
$\theta_{\text{GPS --> pixel}}$ (degrees)	Minimum	40	50	53	56.5	57.75	57.75
	Maximum	60	70	63	61.5	60.25	60.25
	Resolution	2	2	1	0.5	0.25	0.25
	Optimal value found	60	58	59	59.0	58.75	58.75
Minimum SSE		5936	1814	834	405	361	354

Table G.3: Results of brute-force search for minimum SSE. Final optimal values found are shaded gray.

Here, the number of pixels between minor points, L_{minor}, was six pixels, yielding 465 minor points, and 50 major points were created (see [7.3]). Each iteration took approximately 45 minutes to complete on the development machine.

G.7) Steepest-Descent Algorithm for Arc Regression

This section describes how the parameters (x_c, z_c, R) can be found when fitting arc sign regions to a point set. The algorithm presented in Box G.2 was summarised in (7.4.3).

1) Perform least-squares linear-perpendicular regression on point set $S_{centreline}$. This returns parameters defining the best-fit line: θ, k, SSE, and the coordinates of two endpoints $P_{regression\ end1}$ and $P_{regression\ end2}$ on the line (see Figure 7.6 in [7.4.2]).

2) Determine on which side of the best-fit line the arc centre should lie. For example, if the amount of noise in the point set is not too large and the angle spanned by the best-fit arc is not too small, then the centre coordinates of the arc will lie on the same side of the best-fit line as the two end points of the point set to be regressed, as shown in Figure G.3.

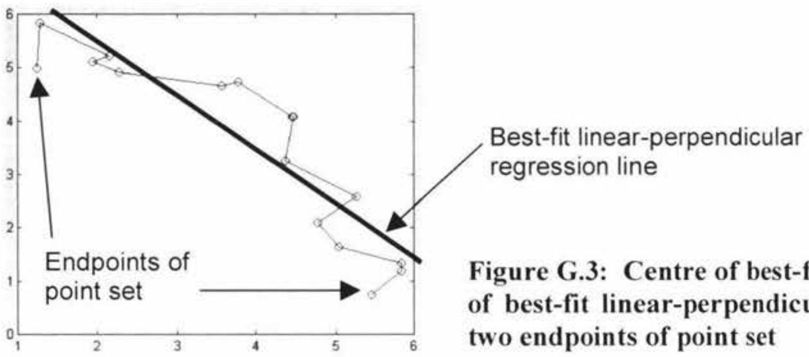


Figure G.3: Centre of best-fit arc lies on same side of best-fit linear-perpendicular regression line as two endpoints of point set

3) Calculate starting coordinates (x_0, z_0) by measuring a distance L_{perp_start} from the best-fit line on the side chosen in step (2). This distance is measured perpendicular to angle θ.

4) Find local minimum point using steepest-descent method:
For each iteration i at current point (x_i, z_i):

   a) Calculate SSE_i, radius R_i and first partial derivatives of SSE with respect to x and z at (x_i, z_i).

   b) Calculate optimal resolution, L_{res}:

      i) If this is the first iteration (i = 0), set L_{res} equal to a small value e.g. 0.01. If this is not the first iteration, set L_{res} based on the optimal resolution found in the previous iteration.

      ii) Traverse a short distance, L_{res}, from current point in direction of steepest descent to point (x_k, z_k).

      iii) Calculate SSE_k, radius R_k and first partial derivatives of SSE with respect to x and z at (x_k, z_k).

      iv) Compare SSE_k with minimum value of SSE found for this iteration i, SSE_{min}. If $SSE_k < SSE_{min}$, we have not reached the minimum point for this iteration yet, so multiply L_{res} by two and repeat steps (ii)-(iv).

   c) Check whether solution has converged by measuring the absolute distance D traversed this iteration:

      i) If D < threshold $T_{converged}$, convergence to a local minimum point has occurred. The centre coordinates of the best-fit arc are located at the values of (x_i, z_i) that yielded the lowest overall value of SSE calculated in step (4b,iii). The radius of the best-fit arc is calculated according to Eq. C.4.7 in (C.4).

      ii) If D >= threshold $T_{converged}$, iterate steps 4 (a-c) again.

Box G.2: Arc regression algorithm that finds best-fit arc for a point set