Self-organising Motorway Traffic Management

Niall O'Hara
B.A.(Mod.) Computer Science
Final Year Project April 2011
Supervisor: Prof. Vinny Cahill

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted as an exercise for a degree at any other university.

_________________________ April 11, 2011
Niall O’Hara
Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

_________________________________________  April 11, 2011
Niall O'Hara
Acknowledgements

First of all, I would like to thank my supervisor Prof. Vinny Cahill for the advice and guidance he has provided throughout this project. I would also like to thank all my fellow classmates for making the past few years so enjoyable. Finally, special thanks to Joanne Ahern for her constant support and the enthusiasm she has for everything I do.

Niall O’Hara

University of Dublin, Trinity College
April 2011
Abstract

Traffic congestion is a major problem across the world - it costs businesses and individuals in terms of time and money, and also impacts on the environment. A solution to the problem, which would provide accurate and cost effective journey times, has long been sought. But what if computer-controlled vehicles, which could sense their environment and communicate with each other, could solve the problem? Such intelligent transport systems are currently a large area of research.

This research proposes a novel motorway traffic management system, providing a time and thus cost-effective answer to road congestion through insights gained from the study of self organisation and their application to autonomous vehicles.

As part of this study, a state-of-the-art review was carried out on current and previous works in the areas of self organisation and motorway traffic management. Two algorithms are presented that allow autonomous vehicles to self organise themselves in such a manner that it mitigates the effects of a reduction in capacity on a motorway.

The algorithms were developed and evaluated within VISSIM, a microscopic traffic flow simulator, through the creation of a self-organising driver model. The evaluation was carried out against a human driver model, Weidemann 99, at increasing levels of traffic volumes on a simulated scenario where a three lane motorway merges into two lanes.

Results from the evaluation indicated that a self-organising traffic management system has the potential to reduce delay times, improve travel times and hence increase efficiency in situations where congestion occurs when compared against the human driver model.

Finally miniature robots were designed and built to serve as a cost-effective platform to investigate the practicalities of implementing the algorithms, and to physically study self-organised driving. Insights gained from this physical implementation will serve to improve future research.

This report documents each stage of the progress throughout the project.
# Contents

Acknowledgements iii  
Abstract iv  
List of Figures viii  

## Chapter 1 Introduction
1.1 Motivation ................................................................. 1  
1.2 Objectives ................................................................. 2  
1.3 Contribution ............................................................... 3  
1.4 Road map ................................................................. 3  
1.5 Summary ................................................................. 4  

## Chapter 2 Background and Related Work
2.1 Traffic Congestion ....................................................... 5  
2.1.1 Consequences of Motorway Traffic Congestion ................. 5  
2.1.2 Economic Theories .................................................. 6  
2.2 Intelligent Transportation Systems ................................... 6  
2.2.1 Existing Technologies .............................................. 7  
2.2.2 Autonomous Vehicles: Driverless Car ......................... 7  
2.2.3 Platooning ......................................................... 8  
2.2.4 Automated Highways ............................................. 9  
2.2.5 Slot Based Driving ............................................... 9  
2.3 Self Organisation .................................................... 10  
2.3.1 Examples of Self Organisation .................................. 10  
2.4 Summary ............................................................... 10  

## Chapter 3 Algorithm Design
3.1 Lane Closure Scenario ................................................. 11  
3.2 Vehicle Requirements ................................................ 12  
3.3 Virtual Grids .......................................................... 12  
3.4 Self-organising Algorithm .......................................... 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>49</td>
</tr>
<tr>
<td>Appendix A Abbreviations</td>
<td>52</td>
</tr>
<tr>
<td>Appendix B Contents of the CD</td>
<td>53</td>
</tr>
<tr>
<td>Appendix C Robot Building Instr.</td>
<td>54</td>
</tr>
</tbody>
</table>
List of Figures

3.1 The ideal self organised formation .................................. 13
3.2 Free space ahead in lane 1 ............................................ 13
3.3 Acceleration to fill the free space in lane 1 ......................... 14
3.4 Self organisation to fill the free space in lane 1 complete ......... 14
3.5 Lane 1 and lane 3 accelerating into position ....................... 14
3.6 Car travelling in lane 1 merging into free space in lane 2 ...... 15
3.7 Merge from lane 1 to lane 2 complete .............................. 15
3.8 Self organisation after the merge into the free space in lane 2 complete .......... 15
3.9 Acceleration to fill the free space in lane 2 ....................... 16
3.10 Acceleration to fill the free space in lane 2 complete ........... 16
3.11 Car travelling in lane 2 merging into free space in lane 3 ....... 16
3.12 Merge from lane 2 to lane 3 complete ............................. 17
3.13 Free space ahead in lane 3 .......................................... 17
3.14 Acceleration to fill the free space in lane 3 ....................... 17
3.15 Acceleration to fill the free space in lane 3 complete .......... 18
3.16 First phase of the merging algorithm .............................. 20
3.17 Second phase of the merging algorithm ........................... 20
3.18 Final phase of the merging algorithm .............................. 20

4.1 The scenario under investigation in VISSIM ......................... 24
4.2 Simulation of 5,000 vehicles per hour - self-organising algorithm .. 25
4.3 Simulation of 5,000 vehicles per hour - human drivers ............ 26
4.4 Average delay time per vehicle .................................... 26
4.5 Travel times - 5,000 vehicles ...................................... 26
4.6 Travel times - 2,000 vehicles ..................................... 27
4.7 Average travel time per vehicle .................................... 27

5.1 Ultrasonic sensor ..................................................... 31
5.2 NXTSumoEyes sensor ................................................ 32
5.3 NXTSumoEyes view zone ........................................... 32
5.4 NXTLineLeader sensor ............................................... 33
5.5 Lego differential gears ............................................. 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>Ackermann Steering</td>
<td>38</td>
</tr>
<tr>
<td>5.7</td>
<td>Design of robot in MLCad</td>
<td>39</td>
</tr>
<tr>
<td>5.8</td>
<td>Ray traced image of the final design</td>
<td>40</td>
</tr>
<tr>
<td>5.9</td>
<td>One lane test track</td>
<td>41</td>
</tr>
<tr>
<td>5.10</td>
<td>Two lane test track</td>
<td>41</td>
</tr>
<tr>
<td>5.11</td>
<td>The four robots involved in testing on the nine meter test track</td>
<td>43</td>
</tr>
<tr>
<td>5.12</td>
<td>‘Wings’ that had to be placed on the robots</td>
<td>43</td>
</tr>
<tr>
<td>5.13</td>
<td>Robot changing lanes to avoid an obstruction</td>
<td>44</td>
</tr>
<tr>
<td>5.14</td>
<td>Test of robots set to follow each other at 9 GSD</td>
<td>44</td>
</tr>
<tr>
<td>5.15</td>
<td>Robots self-organising as described in self-org algorithm</td>
<td>45</td>
</tr>
<tr>
<td>5.16</td>
<td>Alternate view of the robots self-organising</td>
<td>45</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This dissertation presents a motorway traffic management system (TMS) based on the concept of self organisation. The intent is to show that by designing and evaluating the system through simulation and then by constructing, and implementing on, miniature robots that a self-organising TMS has the potential to increase the efficiency of motorway driving. It also serves to show that by creating a cost-effective physical testing platform that insights can be gained to improve future research.

This chapter describes the motivation behind this work, outlines the objectives and contributions and provides a road map outlining the content of the following chapters. Finally a summary of this chapter is presented.

1.1 Motivation

Traffic congestion is a very costly affair for businesses. The European Commission believes it costs Europe about 1% of GDP each year, which equates to about 100 billion Euro [1]. Business models such as just-in-time production and distribution systems require reliable road transport solutions. Freight transport is predicted to increase by about 80% by 2050 compared with 2005 levels, according to figures from the European Commission [2]. Meanwhile, the same report predicts passenger traffic to increase by 34 per cent by 2030 and by 51 per cent by 2050. Transportation reliability, which was defined by the OECD/ITF as ”the ability of the transport system to provide the expected level of service quality, upon which users have organised their activities” [3], is vital for both business and personal efficiencies.

Traffic congestion affects everyone in one way or another - be it in time lost travelling, late deliveries and higher cost of fuel for businesses or higher prices for consumers to cover the cost of transport, so it is in everyone’s interest to solve the problem. Authorities are now seriously looking to the idea of intelligent transport systems (ITS) for a solution.
Much research has already been done in this field, possibly the most well known are Google’s self driving car [4] and the platooning experiments of the EU-funded Sartre project [5] - both of these will be discussed in detail in Chapter 2. Automated cars are considered to be the ultimate solution to motorway traffic congestion, while also cutting the number of accidents and being better for the environment. Demonstrated solutions such as platooning are expected to enhance safety and comfort, enable increased road capacity and efficiency, and reduce congestion, energy consumption and pollution [6].

While there is no shortage of research into this wide field, one work of particular influence on this dissertation is a paper by Marinescu et al. which proposes a slot-based TMS that combines virtual slots with semi autonomous driving to “reduce highway congestion by increasing traffic efficiency [7]”. They build upon previous works that propose that congestion-free travel can be guaranteed by a system which involves vehicles being assigned slots in which to position themselves [8]. These works rely on information provided by an infrastructure component that is communicated to the vehicles, generally referred to as vehicle-to-infrastructure (V2I) communications.

This dissertation proposes an alternative approach by introducing the concept of self-organising vehicles which solve the problem themselves, without such an external controller.

1.2 Objectives

While many traffic congestion solutions are currently in the research stage, a comprehensive solution to the problem would still seem to be some time in the future. This work seeks to learn from and improve upon other ITS research projects and to propose a novel solution to the issue of traffic congestion using self organisation.

To this end, this research intends to evaluate the efficiencies that can be gained by using a self-organising TMS. In the absence of an infrastructure component it is intend to demonstrate that increases in efficiency can be gained through the self organisation of autonomous vehicles. The hypothesis is that increased efficiencies can be gained through the co-ordination of autonomous vehicles.

The first objective is to design algorithms that allow autonomous vehicles to self organise to mitigate the effects of a reduction in capacity of a motorway due to a lane closure. This will be done by modelling a system which relies only on vehicles equipped with advanced sensors and vehicle-to-vehicle (V2V) communication. The algorithms’ intent is to allow the vehicles to self organise, providing their own solution to the problem.
The second objective is to simulate and evaluate the proposed algorithms in order to validate the hypothesis under investigation. This intends to identify the efficiencies that can be gained through the co-ordination of self-organising autonomous vehicles.

When researching autonomous vehicles conducting physical experiments to test proposed algorithms is an important step that needs to be undertaken. However, this is not always practical because of the unavailability of multiple fully sized automated vehicles. This was addressed by Sheng et al. by building an experimental platform based on miniature robots which served as "a cost-effective testbed to study vehicle-to-vehicle communication-based cooperative driving in intelligent transportation systems [9]". This forms the basis for the third and final objective of this research: To design and build autonomous robots and investigate the practicalities of implementing the proposed self-organising TMS on such a platform.

1.3 Contribution

This dissertation presents a novel TMS based on the self-organising behaviour of autonomous vehicles. The principal contribution is two algorithms, evaluated through simulation, which indicate that such a system has the potential to increase the efficiency of motorway travel. The secondary contribution is the design and build of a cost-effective platform for physically testing the deployment of TMS on miniature autonomous vehicles.

1.4 Road map

The remainder of this report has been structured as follows. Chapter 2 introduces the fundamentals necessary to understand this dissertation and provides a review of previous and related work in the fields of self organisation and motorway traffic management. It also evaluates their influence on this work. Chapter 3 introduces the specific scenario under investigation, discusses the influences of self organisation on the algorithm design and introduces the two algorithms developed that comprise the TMS. Chapter 4 outlines the methodologies used and a series of experiments carried out to evaluate the algorithm. Chapter 5 catalogues the design and building of the robots and the practicalities of implementing the algorithm on them. Chapter 6 summarises the contributions to the field and also discusses plans for future work.
1.5 Summary

This chapter outlined the goals of this dissertation, essentially the design of a self-organising TMS and creation of a physical testing platform for autonomous vehicles. The chapter began by outlining the motivations behind this work, the need for increased efficiencies on motorways. It then introduced the field of ITS and autonomous vehicles. The main contributions were then detailed and it concluded with a brief description of the following chapters.
Chapter 2

Background and Related Work

As outlined in the introduction, this dissertation seeks to examine the potential of a self organisation system to deal with the issue of motorway traffic congestion. Over-capacity motorways are a big problem across the globe. Previous responses to the issue were simply to build new roads. However, taking into account the fiscal, social and environmental costs, authorities are now looking at less destructive methods, namely smart technology. Many ITS solutions are currently under development. This chapter looks at the causes and consequences of traffic congestion in Section 2.1, outlines related ITS research in Section 2.2 and introduces the concept of self organisation and its relevance to the field of study in Section 2.3. Finally chapter is summarised in Section 2.4.

2.1 Traffic Congestion

Traffic congestion basically occurs when there are too many cars for a particular stretch of road to handle with any efficiency. A classic example of this is when a busy dual carriageway suddenly becomes a single carriageway i.e. two lanes of traffic suddenly merge to share one lane - the volume of traffic remains the same but the road space available halves, thus creating a traffic jam. Road accidents and road works can also lead to congestion.

2.1.1 Consequences of Motorway Traffic Congestion

Motorway traffic congestion is the bane of many a driver’s life. To individuals and businesses alike it’s a costly affair, in terms of time lost, fuel wasted and deadlines missed. Because of the nature of a motorway’s structure - exits at specific intervals, a driver may be sitting on a congested motorway for long periods of time simply because they had the misfortune to have just passed an exit when they encountered the blockage. The impact on emergency vehicles also has to be considered - congested roads make it more difficult, and sometimes impossible, for them to reach an emergency fast.
2.1.2 Economic Theories

Traffic congestion costs both in terms of time and of money. However inconvenient it is for the average commuter, it also impacts greatly on businesses that rely on the road network. It makes employees and deliveries late and it affects just-in-time supply chains. As previously mentioned, the European Commission estimates traffic congestion costs businesses 1 per cent of GDP annually - that’s about 1bn Euro. In the past, the simple answer was to build more roads however, now some smarter thinking is required. Some of the economic solutions put forward to counter traffic jams are congestion charges for city centre areas and road tolls. Many economists, such as American Anthony Downs, see tolls as the answer to congestion issues which hit during peak hours only, rather than widening roadways or increasing flow pressure via automated highway systems [10]. Indeed, the Commission on Taxation in Ireland has described tolls and congestion charges as "incentives to road users to use road capacity more efficiently" [11]. While these charges may well be incentives they push the problem elsewhere, on overloaded and inflexible public transport system for instance, rather than actually solving it.

2.2 Intelligent Transportation Systems

The Australian Department of Infrastructure and Transport defines ITS as "the application of informed technology to transport operations in order to reduce operating costs, improve safety and maximise the capacity of existing infrastructure". The aim is to establish a system that enables commuters to arrive at their destinations safely, quickly, and comfortably. ITS innovation is also expected to push down distribution costs, by allowing motorists to reduce fuel consumption by driving more efficiently, which would also be better for the environment.

Many early ITS smart technologies concerned themselves primarily with driver assistance systems, such as adaptive cruise control (ACC) [12], automatic reverse parking [13] and driver information systems [14]. However, the ITS industry is now evolving swiftly, gaining traction commercially with the motor industry and while driver assistance systems are still of major importance, futuristic predictions, such as self driving cars, have become more of a reality. In 2010 VisLab ran a 13,000 km test run of autonomous vehicles from Italy to China [15]. Google’s test fleet of autonomous vehicles [4] has, as of October 2010, driven more than 140,000 miles without any incidents.

Interestingly, in the United States the ITS field is closely associated with, and often funded by, the Department of Homeland Security [16]. That government department has an interest in the technology because it involves surveillance of the roadways. It can also help in the quick mass evacuation of people in urban centres after large casualty events such as a natural disaster or

2.2.1 Existing Technologies

Everyday mass produced cars are getting smarter by the generation. Many of the smart technologies previously demonstrated are now available in everyday affordable cars. Many motor companies, such as Volvo, Ford and Audi, offer technologies such as adaptive cruise control, lane departure warning and active park assist as options in their mid-priced cars. They are very generally marketed as driver aids to improve safety. While these features certainly do that, ultimately they may combine to achieve an affordable, mass produced semi-autonomous car. More recently, some work has focused on fully autonomous vehicles.

2.2.2 Autonomous Vehicles: Driverless Car

Although driverless cars seem to have been around on TV and in the movies forever, Knight Rider’s KITT dates back to 1982, in actuality they are still only in their infancy in terms of development. A driverless car is one which is totally automated, requiring no input from a human driver. Equipped with an autopilot system, the vehicle is capable of driving from point A to point B totally on its own. The benefits of driverless cars are numerous and include transporting loads in unstable areas, such as battlefields or disaster zones, and allowing human ‘drivers’ to concentrate on other matters while travelling in their private car. Driverless cars could impact on traffic congestion by managing traffic flow and accidents could be avoided.

The concept of self driving cars is nothing new, indeed from the time the first car rolled off the Ford production line around 1913 the self driving car has been an aspiration. While the average motorist may not realise it, each new technological advance may be bringing us one step closer to an all round self driving car. Drivers are becoming ever more familiar with technologies such as cruise control, speed sign recognition, lane departure warning, low speed collision prevention technology, and active park assist - each one of these technologies is employed in the self-driving car.

Scientists and engineers actually began working on autonomous cars in the 1960s but mass production of such cars isn’t expected before 2020 [17]. For some proposed solutions to work, infrastructure needs to be embedded in or near the roads however, the more advanced research tries to mimic human perception and decision-making during driving, using advanced computer software linked to a number of sensors, including cameras, radars and GPS. The TMS presented in this dissertation is an example of such a system.

Google’s self driving cars, a fleet comprised of six modified Toyota Prii (that’s the decreed plural
of Prius \(^2\)) and an Audi TT, are a perfect example of the autonomous vehicle. The technology has already, with human supervision, completed more than 140,000 miles of real life testing. The test cars have also each driven 1,000 miles without human intervention. One car has even tackled San Francisco’s Lombard Street, a prime example of the City by the Bay’s trademark steep and curvy streets, without incident. The only accident reported in the 140,000 mile test run was when a car was rear-ended at traffic lights [17]. Google aims for its technology to reduce road traffic and to cut the number of road accidents. In fact, it believes its self driving car has the potential to cut by half the 1.2 million road deaths recorded by the World Health Organisation each year. Reducing the likelihood of a car crash would mean cars could be made lighter, thus reducing the cost of running them.

### 2.2.3 Platooning

While the Google project concentrates on individual cars, a European Commission supported scheme is looking at a road-train type (platooning) scenario. The Safe Road Trains for the Environment (Sartre) project is more in keeping with this research. Car giant Volvo is also involved in the scheme [18].

Platooning involves a convoy of vehicles with a professional driver at the head of the platoon leading the others - the rest of the drivers can sit back and relax while in the platoon. Each vehicle notes the distance, speed and direction and adjusts to the car in front. Platooning has been described as a 'car train' but with one notable difference from a regular train - as each vehicle is detached, any in the platoon can leave at any time. Because the human reaction factor is removed, vehicles could travel closer together than human driven vehicles would allow, thus easing congestion.

Tom Robinson, who is the road-train project coordinator for Ricardo UK, one of the seven companies working on the project, said: "Platooning offers the prospect of improved road safety, better road-space utilisation, improved driver comfort on long journeys and reduced fuel consumption and hence CO2 emissions. [19]"

While early platooning projects were based on the modification of road infrastructure, much current research, such as the Sartre project, does not. Road tested for the first time earlier this year, it is perhaps the most widely known current platoon scheme in operation.

The self-organising TMS that this research proposes could be considered an extension of platooning.

\(^2\)http://www.autoblog.com/2011/02/21/toyota-says-plural-of-prius-is-prii
2.2.4 Automated Highways

Automated highways have also been touted as a possible answer to the world’s growing traffic congestion problem but with investment now going more towards smart cars than smart roads, they seem to have had their day. Automated highway systems (AHS) involve equipping special lanes on existing motorways with magnets and other infrastructure to allow vehicles to stay in the centre of the lane, while communicating with other vehicles to manage traffic and to avoid collisions. Each specially equipped vehicle can remain independent and just use the special routes to get to where they’re going quicker and return to ‘normal roads’ when it suits them. Some AHSs, such as California’s PATH project [20], use radar to avoid pile ups and to co-ordinate speeds.

The PATH project, a prototype automated highway system, was tested in San Diego, California in 1997. The Interstate 15 test was sponsored by the US government and co-ordinated with the State of California and Carnegie Mellon University.

2.2.5 Slot Based Driving

In a slot based TMS an infrastructure component assigns slots to the motorway. A slot represents a time-space corridor negotiated among vehicles. Marinescu et al. extend on the previous notion of a slot moving constantly to a system in which slots behave in a more flexible manner, change lanes, accelerating and decelerating.

They presented results that indicated that their slot based approach could both solve the problem of traffic congestion and provide guaranteed travel time estimations regardless of the volume of traffic on the motorway. This research expands further on the concept of a slot by removing the centralised controller co-ordinating traffic and replacing it with a self organising system relying on virtual grid spaces which the individual vehicles assign to themselves.

2.3 Self Organisation

Self organisation can be described as the ability of a system to spontaneously, and without the help of an external controller, arrange its elements in a focused (non random) manner in response to a specific situation. This research explores self organisation as a means of eliminating motorway traffic congestion.
2.3.1 Examples of Self Organisation

There are many examples of self organisation all around us, some are very visible to the naked eye, others evade our consciousness. Many natural systems such as cells, chemical compounds and planets display this property. It is also quite inherent to the everyday lives of humans and animals.

Looking at the human aspect of self organisation, and combining it with biological systems, how we move, develop, and learn are all examples of self organisation. In human society, out of the ordinary events, such as an emergency evacuation, can be linked to self organisation. Research has shown that it takes just five per cent of informed individuals (those who know where the exit is) to guide a group without verbal communication or gestures [21]. Each informed individual simply communicates with the people nearest to them and this is enough to guide the group.

Similarly, birds flying in a V formation, and a school of dolphins self-organising through echo location are also examples of the phenomenon. A combination of these examples formed the basis for the self-organising TMS developed as part of this research.

Self organisation isn’t confined to nature. It is also used to produce emergent behaviour in swarm robotics [22]. Then there’s the ‘sexy’ artificial self organisation - namely robot football. Robot football has its genesis in South Korea in 1995 and an international contest has been held every year since. The competition offers numerous categories, from simulation to humanoid robots, on a specially constructed ‘pitch’. The aim is to create a robot team capable of beating a human one by 2050. This year’s competition will be held in Istanbul in July.

BeeJamA [23] is an example of self-organising ITS proposed by Wedde et al. They present a routing system for vehicular traffic derived from honey bee behaviour. Their research showed that by borrowing the main ideas from swarm intelligence as detected in honey bee communication, efficient routing algorithms could be constructed that established a “somewhat stable yet fluent traffic” situation.

2.4 Summary

This chapter began by providing an insight into the causes and effects of traffic congestion. The next section introduced the field of ITS and described both demonstrated and deployed technologies. The concept of self organisation was then introduced and examples provided. The next chapter outlines the design of the TMS this dissertation proposes.

[3]http://www.youtube.com/watch?v=cP03SM_w82m
Chapter 3

Algorithm Design

Designing a TMS is a very challenging undertaking. As outlined in Chapter 2 there are many technologies involved and variables to be taken into consideration. For this reason this work simplifies the problem by assuming ideal conditions where necessary. The first section introduces the scenario under investigation. Section 3.2 describes the vehicle requirements and details the assumptions that were made. In Section 3.3 the concept of virtual grids, upon which the algorithms depend, is introduced. Sections 3.4 and 3.5 detail both algorithms used in the self-organising TMS and explain the scenarios in which they apply. The chapter is summarised in Section 3.6.

3.1 Lane Closure Scenario

The main premise under investigation is that efficiencies can be gained through a self-organising system. This premise was tested on a scenario where a rush-hour accident has led to a three-lane motorway merging into two lanes. With peak traffic volumes, and a third of the capacity of the road off limits, a bottleneck situation emerges which has the potential to lead to congestion.

Human driving in the main is competitive rather than co-operative. Marinesco et al. proposed that another cause for congestion was this driving behaviour. They ran experiments which showed that human drivers tackle merging lanes in an inefficient manner which causes congestion. This results in a longer than average travel time.

In order to address this, the self-organising algorithms are designed to position the vehicles in such a way that they can easily merge from three to two lanes in the event of a lane closure.
3.2 Vehicle Requirements

Each vehicle had to adhere to basic rules and requirements. The principal rules were that each vehicle should always try to get into a faster lane and that it should always keep a pre-determined safe distance from neighbouring cars.

Each vehicle was also required to be aware of its own speed, acceleration, road and lane positioning, as well as the presence of other vehicles ahead.

The final requirement was that each vehicle should have the ability to communicate with other vehicles. In order to simplify the design process, an ideal form of communication between the vehicles was assumed in which no transmissions were lost or delayed.

3.3 Virtual Grids

Marinesco et al. described the concept of slot based driving where a centralised controller coordinates the traffic by assigning slots on the motorway and each vehicle had to position itself into these slots. Removing the centralised controller, this work introduces the concept of virtual grids. In this research, each vehicle is aware of its own grid, upon which it bases its decisions.

Each vehicle is aware of the concept of grid spaces. A vehicle sits in the centre of its own virtual grid, a 5 meter long by 1 lane wide space. The vehicle measures distances with regard to a grid space distance (GSD). So if a vehicle wishes to remain 5 GSD from another that translates to a distance of 25 meters.

3.4 Self-organising Algorithm

The central concept behind the self-organising TMS is to maintain throughput on the motorway without slowing down. In order to achieve this, as with any self-organising system, a number of base rules have to be followed, as detailed below under the vehicle requirements section.

There are many examples of self organisation in nature. Of particular interest to this project were the V formation in which birds fly and the ability of dolphins to navigate as a group. These formed the basis for this algorithm. The algorithms were developed and modelled in VISSIM, a microscopic traffic simulator, which will be detailed in the next chapter.

In the beginning many algorithms were attempted and ultimately rejected through a trial and error process. At each stage, the performance of the algorithm was noted and the results recorded.
After each run, the results were compared to previous runs, lessons were learned and the algorithm was modified. This trial and error approach, based on results and educated examinations resulted in the following scenarios and ultimately the self-organising algorithm. The self-organising algorithm applies to vehicles that are at least 150 meters upstream from any lane closure.

3.4.1 Ideal self organised formation

Figure 3.1 represents the ideal self organised formation. The rules being applied here form the basis for the algorithm.

![Figure 3.1: The ideal self organised formation](image)

Cars in odd numbered lanes are trying to stay 6 GSD behind a car in the same lane and 3 GSD behind a car in the lane directly to its right. The green car is applying these rules in respect of the blue and red cars. The orange car is applying this rule to the yellow car alone as there is no right lane available to it.

Cars in even numbered lanes are trying to stay 6 GSD behind the car in the same lane and 3 GSD behind the cars travelling in the lanes to their left and right. The red car is applying these rules.

3.4.2 Scenarios

**Scenario A - Lane 1 accelerating to free space ahead in same lane**

![Figure 3.2: Free space ahead in lane 1](image)
Figure 3.2 represents a situation where the green car in lane 1 can sense that there is no car preventing it from accelerating into a further position in lane 1, it does this by sensing how far ahead the next car is in lane 1 and determining if it can accelerate to a position where it can still conform to the base set of rules previously described.

This allows the green car to overtake the red car as the desire to maintain position behind a vehicle in the same lane outweighs the desire to maintain position behind a vehicle in a different lane.

![Figure 3.3: Acceleration to fill the free space in lane 1](image)

In Figure 3.3 the green car continues to move into its new position. The pink car represents a vehicle that has followed the same rule as the green car and is now accelerating into the space that the green car left vacant.

![Figure 3.4: Self organisation to fill the free space in lane 1 complete](image)

Figure 3.4 shows the formation after the cars have completed their goals.

**Scenario B - Lane 1 merging to free space in lane 2**

![Figure 3.5: Lane 1 and lane 3 accelerating into position](image)
In the scenario shown in Figure 3.5 the orange and green cars are getting into position behind the blue and yellow cars in lanes 1 and 3 at a distance of 6 GSD.

![Figure 3.6: Car travelling in lane 1 merging into free space in lane 2](image)

In Figure 3.6 the green car has calculated that it has the required distance to get into a faster lane, lane 2 in this case. It announces its intent to all other cars that it is going to move into this lane and if it does not get a denial it proceeds. This prevents the red car from trying to accelerate as it received the lane change notification before it had made any announcements itself. The announcements that are sent include GPS co-ordinates and a timestamp that allows the cars to determine if they apply to them.

![Figure 3.7: Merge from lane 1 to lane 2 complete](image)

Figure 3.7 illustrates that the green car has now moved into its new position. The pink car in lane 1 now accelerates to fill the space that has been made in lane 1.

![Figure 3.8: Self organisation after the merge into the free space in lane 2 complete](image)

Figure 3.8 shows the formation after the cars have completed their goals.
**Scenario C - Lane 2 accelerating to free space ahead in same lane**

In Figure 3.9 we have a similar situation to that shown in Figure 3.6, the difference this time is that the red car has not received an announcement from the green car that it intends to change lane.

![Figure 3.9: Acceleration to fill the free space in lane 2](image)

This prevents the green car from changing lane as the red car responds to any lane change announcement with a denial.

![Figure 3.10: Acceleration to fill the free space in lane 2 complete](image)

Figure 3.10 shows the formation after the cars have completed their goals.

**Scenario D - Lane 2 merging to free space in lane 3**

In Figure 3.11 there is a free space in lane 3, a similar situation to that which occurs when a car is changing from lane 1 to 2. The red car calculates if there is a safe distance to get into lane 3 and announces its intention to move into that lane, preventing the orange car from accelerating.

![Figure 3.11: Car travelling in lane 2 merging into free space in lane 3](image)
In Figure 3.12 the formation is exactly the same as occurred in Figure 3.5. The algorithm now continues as it did there.

![Figure 3.12: Merge from lane 2 to lane 3 complete](image)

**Scenario E - Lane 3 accelerating to free space ahead in same lane**

In Figure 3.13 the orange car has not received any broadcasts that would deny it accelerating to fill the space ahead. It accelerates to fill the space available and denies any requests from the red car to change lane.

![Figure 3.13: Free space ahead in lane 3](image)

![Figure 3.14: Acceleration to fill the free space in lane 3](image)

As can be seen in Figure 3.14, the orange car continues to move into its new position. The pink car represents a vehicle that has followed the same rule as the orange car and is now accelerating into the space that the orange car made. Figure 3.15 shows the cars have returned to the ideal formation after carrying out their manoeuvres.

Figure 3.15 shows the formation after the cars have completed their goals.
3.4.3 Algorithm: SELF-ORGANISE

The scenarios previously described outline all the rules the vehicles should be following, a combination of these rules now from the self-organise algorithm of the TMS.

begin SELF-ORGANISE
    set merging to false (SELF-MERGE);
    get current lane number (LN);
    while not SELF-MERGE do
        set vehicle to accelerate to max speed limit and maintain speed;
        respond to any lane change announcements;
        if lane closure detected do
            set SELF-MERGE to true;
            broadcast closure co-ordinates and current time;
        end
        if valid broadcast of lane closure received do
            set SELF-MERGE to true;
            re-broadcast closure co-ordinates and time;
        end
        if LN is odd then
            if sensed vehicle ahead in the current lane then
                save distance to vehicle minus 6x grid space distance (DVA);
            end
            if right lane exists and sensed vehicle ahead in the right lane then
                save distance to vehicle minus 3x grid space distance (DVR);
            end
            if DVR is greater than 9x grid space distance and right lane exists then
                broadcast intention to change lane;
                change to right lane;
            else if DVR is greater than DVA then
                set speed to maintain 6x grid space distance to vehicle ahead;
            else
                set speed to maintain 3x grid space distance to vehicle ahead right;
        end
    end
end
end
else if LN is even then
   if left lane exists and sensed vehicle ahead in the left lane then
      save distance to vehicle minus 3x grid space distance (DVL);
   end
   if sensed vehicle ahead in the current lane then
      save distance to vehicle minus 6x grid space distance (DVA);
   end
   if sensed vehicle ahead in the right lane then
      save distance to vehicle minus 3x grid space distance (DVR);
   end
   if DVR is greater than 9x grid space distance then
      broadcast intention to change lane;
      change to right lane;
   else if DVL is less than DVA then
      set speed to maintain 3x grid space distance to vehicle ahead left;
   else if DVR is greater than DVA then
      set speed to maintain 6x grid space distance to vehicle ahead;
   else
      set speed to maintain 3x grid space distance to vehicle ahead right;
   end
end
GOTO SELF-MERGE
end SELF-ORGANISE

3.5 Self-merging Algorithm

The self-organise algorithm only needs minor modification to become the self-merging algorithm. This is because most of the initial alignment has already been done through the application of the self-organising algorithm. The self-merging algorithm applies when the vehicle is within 150 meters of the lane closure. The new set of base rules are as follows.

Cars in odd numbered lanes are trying to stay 4 grid space distances (GSD) behind a car in the same lane and continually wish to change to the the lane directly to the right, if one exists. If no right lane exists they accelerate to the maximum speed limit.

Cars in even numbered lanes are trying to stay 4 GSD behind the car in the same lane, 3 GSD behind cars in the lane to its left and 2 GSD behind cars in the lane to its right.
3.5.1 Merging Phases

Figure 3.16: First phase of the merging algorithm

In Figure 3.16 the blue car has detected or received an announcement of an upcoming lane closure, it immediately starts the process of merging to lane 2. The blue car also announces this lane closure along with a timestamp and GPS co-ordinate. The green and red cars now both accelerate to change their position. The cars travelling in lane 3 increase their speed.

Figure 3.17: Second phase of the merging algorithm

In Figure 3.17 the blue car has now successfully moved into lane 2. The self-merging formation is now shown. Cars in even numbered lanes are trying to remain 4 GSD behind the car in front, hence the pink car is accelerating. They will also change to the right lane if there are at least 3 GSD spaces available, otherwise they will follow the car to their right. Cars in odd numbered lanes are trying to remain 2 GSD behind cars to their left, if applicable, otherwise they will change to the right lane.

Figure 3.18: Final phase of the merging algorithm

We are now close to the lane closure, the remaining cars in lane 1 will now merge into lane 2.
Any cars in lane 2 that have space available will merge into lane 3 and the cars in lane 3 continue to close the gap to the car ahead to be 4 GSD.

### 3.5.2 Algorithm: SELF-MERGE

The phases just described the actions required to efficiently merge three lanes of self-organised traffic into two lanes. A combination of these phases now forms the self-merge algorithmic component of the TMS.

begin SELF-MERGE
    set merging over to false (MERGE-OVER);
    get current lane number (LN);
    get lane closure co-ordinates (LCC):
    while not MERGE-OVER do
        if LN is odd then
            if right lane exists and LCC is less than 6x grid space distance then
                change to right lane;
                set MERGE-OVER to true;
            else
                set speed to maintain 4x grid space distance to vehicle ahead;
            end
        else if LN is even then
            if left lane exists and sensed vehicle ahead in the left lane then
                save distance to vehicle minus 3x grid space distance (DVL);
            end
            if sensed vehicle ahead in the current lane then
                save distance to vehicle minus 4x grid space distance (DVA);
            end
            if sensed vehicle ahead in the right lane then
                save distance to vehicle minus 2x grid space distance (DVR);
            end
            if DVR is greater than 3x grid space distance and right lane exists then
                change to right lane;
                set MERGE-OVER to true;
            else if DVR is greater than DVA then
                set speed to maintain 4x grid space distance to vehicle ahead;
            else
                set speed to maintain 2x grid space distance to vehicle ahead right;
            end
        end
    end
end
if current co-ordinates are past LCC then
    set MERGE-OVER to true;
end
end
GOTO SELF-ORG
end SELF-MERGE

3.6 Summary

This chapter began by outlining the main scenario under investigation. The requirements for the vehicles was introduced and the concept of virtual grids described. Finally the two algorithms that the self-organising TMS is based upon were described in detail. The next chapter describes the environment in which the algorithms were designed and evaluated and presents the results from the evaluation.
Chapter 4

Simulation and Evaluation

This dissertation presents a self-organising TMS comprising of two algorithms that allow vehicles to efficiently merge in the case of a reduction in capacity on a motorway. This chapter first introduces the simulation software used to design and test the algorithms. Section 4.2 describes the experimental configuration. In Section 4.3 the evaluations of the TMS are presented. The chapter concludes with a summary.

4.1 VISSIM

VISSIM [24] is a microscopic traffic flow software system developed by the German company PTV AG. It is regarded as a market leader in traffic simulation and is used by industry giants such as Volkswagen \(^1\) and extensively in ITS research around the world.

4.1.1 COM Interface

VISSIM offers an additional module which provides COM (component object model) functionality for use with external programming environments [27]. This way it is possible to automate certain tasks in VISSIM by executing COM commands from an external program or DLL (dynamic link library).

4.1.2 Driver Model

The traffic flow model in VISSIM is based on the psycho-physical movement of vehicles theory of computer scientist Rainer Weidemann. The Weidemann model [25] was first developed in 1974 but the Weidemann 99 [26] version is more applicable to motorway traffic and was used in this research. It provides the vehicle with information about itself as well as about neighbouring vehicles. The vehicle can tell its own speed, the lane it is travelling in, acceleration and steering

\(^1\)http://www.comesafety.org/fileadmin/user_upload/Events/simulationWorkshop/SW_Wewetzer.pdf
as well as the relative speed and distance of nearby vehicles.

4.1.3 External Driver Model

An external driver model can be used to compute and return to VISSIM variables such as the desired speed and lane. This functionality was perfect for this project. PTV also provides a C++ template to program a DLL replacement to the standard driver model which controls the behaviour of the individual vehicles. DSG has written a wrapper for the VISSIM COM interface which allows external driver models to be written in C#. This API wrapper was used in the project to as a way to code the self-organising driver model in C#.

4.2 Simulation Configuration

In order to test the algorithms described in the previous chapter, a three to two lane merging scenario was modeled in VISSIM. For testing and evaluation purposes a 2.5km stretch of motorway was designed with three lanes merging into two lanes after 2.2km.

![Figure 4.1: The scenario under investigation in VISSIM](image)

4.3 Evaluation

The purpose was to evaluate the difference between the self-organising TMS and the human driver.
The human driver was evaluated using the Weidemann 99 model. Two measurements were taken for each vehicle in the network - the time it took to travel the road and the amount of time spent delayed. VISSIM defines delay time measurement as "a determination, compared to the ideal travel time, of the mean time delay calculated from all vehicles observed" [28]. The readings were taken over a period of one hour and the average was calculated. The maximum speed was set at 120kph to simulate a real motorway. Acceleration and braking were both set at +/- 4meters/second, and minimum safe distance was set at 1.5meters.

A number of simulations were run at different volumes of traffic, namely 2,000, 3,000, 4,000, 5,000 and 6,000 vehicles per hour. Each test was run 12 times and the average of these test runs was taken.

For traffic volumes up to 4,000 vehicles per hour there was minimal difference between the human drivers and the self-organising TMS. However, it was observed that the self-organising algorithm produced marginally lower travel times at these volumes. This was most likely due to the fact that they were always trying to achieve their maximum speed and there was always enough space for them to do so.

At traffic volumes of 5,000 vehicles per hour a dramatic difference can be seen between the human and self-organising drivers. Travel times for human drivers greatly increased. If the Weidemann 99 model can be assumed to accurately simulate human driving, one could say this increase is due to the competitive rather than co-operative nature of human driving. Indeed, the human drivers were observed during simulation to constantly switch lanes to get into a lane that was always moving.

When the self-organising system was applied to volumes of 5,000 vehicles per hour and above, travel times continued to rise slightly but nowhere near the level of increase the human drivers experienced. As previously suggested, this indicates that the self organised manner of driving provides far more efficient merging than that of human drivers.

For volumes of vehicles above 6000 vehicles per hour the performance of self-organising TMS began to degrade due to the fact that the volume of traffic on the motorway was in excess of the physical space to allow the distances required by the system. Vehicles began to start queuing at the entrance to the motorway and the delay time spent at that point increased.

Figure 4.2: Simulation of 5,000 vehicles per hour - self-organising algorithm
Figure 4.3: Simulation of 5,000 vehicles per hour - human drivers

Figure 4.4: Average delay time per vehicle

Figure 4.5: Travel times - 5,000 vehicles
Figure 4.6: Travel times - 2,000 vehicles

Figure 4.7: Average travel time per vehicle
4.4 Summary

This chapter introduced VISSIM and the steps taken to develop and evaluate the self-organising TMS algorithms by creating a replacement driver model. The simulation configuration was then introduced. Finally the evaluation of the TMS was described and results presented. The next chapter details the design, build, testing and evaluation of the robots specifically created for this dissertation.
Chapter 5

Robot Design and Implementation

This chapter covers the design and implementation of the robots created to investigate the practicalities of using the proposed algorithms on autonomous vehicles. The platform used to create the robots will firstly be introduced along with the sensors that were used in their construction. Section 5.2 details the specification for what the robots should be able to do and the supporting requirements to achieve these goals are listed in Section 5.3. In Section 5.4 the control system for the robots is described. The physical design process is outlined in 5.5 and descriptions of the testing, evaluation and difficulties encountered follows in Sections 5.6, 5.7 and 5.8. The chapter concludes with a summary.

5.1 Lego Mindstorms NXT

The Lego Mindstorms NXT system may look like a toy but it’s really a full blown robotics development environment. It is based on work done at the MIT Media Lab by learning researchers Seymour Papert and Mitchel Resnic 1. The heart of the NXT system is the NXT programmable brick. It is a full 16-bit microprocessor with inputs for up to four sensors and outputs for up to three motors. The current generation is known as Lego Mindstorms NXT 2.0. The Distributed Systems Group (DSG) in Trinity College Dublin (TCD) provided four kits for use in this project.

5.1.1 Choice of OS & Programming Environment

Lego provides an open source firmware development kit which allows for a number of environments in which to develop applications to run on the platform. A review was carried out on each of these as part of the evaluation of the NXT system, a summary of which follows.

NXT-G 2

---

2http://www.ortop.org/NXT_Tutorial
• No Custom Firmware required.
• Default graphical language for programming LEGO’s NXT.
• Adequate for basic programming, such as driving motors, incorporating sensor inputs, doing calculations, and learning simplified programming structures and flow control.

Microsoft Robotics Developer Studio - R3

• No Custom Firmware required.
• Communicates to the NXT via bluetooth.
• Restful service oriented environment - Decentralized Software Services (DSS) built on top of Concurrency and Coordination Runtime (CCR). Each sensor or actuator (motor) is represented as a separate service.

RobotC

• Custom Firmware required.
• much faster than NXT-G.
• C-Based Programming Language with an IDE.
• Similar to NBC / NXC / BricxCC but license required.
• Support for Mindsensors sensors.

LeJOS

• Custom Firmware required.
• much faster than NXT-G.
• Replacement firmware for the NXT that includes a Java Virtual Machine.
• Supports NXT to NXT Bluetooth and RS485 communications.
• Plugins for both Eclipse and Netbeans.
• Support for Mindsensors sensors.

NBC / NXC / BricxCC

• Custom firmware required for advanced support.
• Next Byte Codes (NBC) - a simple language with an assembly language syntax that can be used to program LEGO’s NXT.
• Not eXactly C (NXC) - a high level language, similar to C, built on top of the NBC compiler.
• BricxCC Command Center (BricxCC) - IDE that supports programming the LEGO Mindstorms NXT brick using NXC and NBC.
• Much faster than NXT-G.
• Supports NXT to NXT Bluetooth and RS485 communications.

---

3http://www.microsoft.com/robotics
4http://www.robotc.net/support/nxt
5http://lejos.sourceforge.net
6http://bricxcc.sourceforge.net/nbc
• Support for Mindsensors sensors.

Ultimately it was decided to develop the controlling software for the robots in NXC using the BricxCC IDE. During evaluation tests it provided the perfect balance between required speed and capabilities.

5.1.2 Sensors

The base Mindstorms NXT kit includes a number of sensors for detecting sound, light, touch and distance via an ultrasonic sensor. There are also a number of third party sensors available from various manufacturers including Mindsensors\(^7\). It was decided that the main requirements for the sensors were that they had to allow the robots to follow a line, provide the ability to detect the distance to obstacles and also the ability to detect obstructions and other vehicles to their left and right. The sensors that were used in the construction of the robot are detailed below.

Ultrasonic Sensor

Senses distance and recognises objects and movement. The Ultrasonic Sensor is able to detect an object and measure its proximity in inches or centimeters.

![Ultrasonic sensor](image)

Figure 5.1: Ultrasonic sensor

---

\(^7\)http://www.mindsensors.com/
**NXTSumoEyes**
Dual Range, Triple Zone Infrared Obstacle Detector for NXT that supports two ranges. The short range makes the sensor insensitive to objects beyond 15cm whilst the long range makes it sensitive to objects up to 30cm.

![NXTSumoEyes sensor](image)

**Figure 5.2: NXTSumoEyes sensor**

**NXTLineLeader**
This is an array of 8 sensors with controlled light source, returning you values of the sensor readings. Allows you to write line follower programs with your own decision making and develop your own PID control algorithms.

![NXTSumoEyes view zone](image)

**Figure 5.3: NXTSumoEyes view zone**
5.2 Specification

The constructed robots will implement both the algorithms as outlined in Chapter 3 as their prime directives but in order to achieve these behaviours a number of lower level controls are required. What follows is a specification of how these behaviours were implemented.

As laid out in the algorithms, the main requirements for the robots are to be able to sense distances to other robots in the environment and to maintain a required distance from these robots. They must also be able to detect an obstruction, avoid the obstruction, notify other robots of the obstruction and notify other robots of their intention to change lane.

5.3 Requirements

5.3.1 Follow line

Prerequisites
A vehicle must be provided with the speed limitations of the current environment.

Required behaviour
A vehicle must be able to drive unaided along a 'road' using only a combination of locally attached sensors. This road will be represented as a line that the vehicle will follow.

Actions required to achieve behaviour

- take reading from line detection sensor
- increase speed
- decrease speed
- steer left
- steer right
5.3.2 Change line to follow

Prerequisites
A vehicle must be aware of the current line in which it is following.

Required behaviour
A vehicle must be able to leave the line that it is currently following and switch to follow a different line.

Actions required to achieve behaviour

- take reading from line detection sensor
- increase speed
- decrease speed
- steer left
- steer right

5.3.3 Detect entities in the environment

Required behaviour
A vehicle must be able to detect entities in its environment. These can be either stationary or moving (another vehicle).

Actions required to achieve required behaviour

- take reading from entity detection sensor(s)
- calculate distance to entity
- calculate relative position of entity
- determine if entity is moving

Resultant actions that may need to be taken

- increase speed
- decrease speed
- change line to follow
- communicate with other vehicles

5.3.4 Communicate with other vehicles

Required behaviour
A vehicle must be able to both send/receive messages to/from other vehicles.

Actions required to achieve required behaviour

- listen for message
• broadcast message

**Resultant actions that may need to be taken**

• change line to follow

### 5.3.5 Follow vehicle

**Prerequisites**

A vehicle must continue to follow a line

**Required behaviour**

A vehicle must be able to maintain a required distance to another vehicle

**Actions required to achieve required behaviour**

• follow line
• detect entities in the environment
• increase speed
• decrease speed

### 5.4 Modelling Behaviours

From the requirements in the previous section it can be seen that really there is only one top level behaviour required, that is to drive along the road. All other behaviours link back to that one directive. A robot can detect, or be informed by another robot, that there is an obstruction ahead and it will have to change the line that it is following. Similarly a robot can modify its speed to follow another vehicle in its path, but as before it still has to maintain the line it is following.

A hierarchical control system was chosen to implement this control structure. In this design paradigm behaviour is modelled into nodes, each one representing a different kind of behaviour.

It takes the shape of a tree in which each node operates independently, performing tasks from its superior node, commanding tasks of its subordinate nodes, sending abstracted sensations to its superior node, and receiving sensations from its subordinate nodes. Leaf nodes are sensors or actuators. As NXC supports the concept of tasks which can run in parallel this is the perfect environment for implementing a hierarchical control system.

Each node can have input parameters that slightly modify its behaviour. There are two kinds of input parameters. The first tells the node how to behave while executing. Speed and steering angle are examples of this. The second kind of input tells a node when to finish executing,
for example after driving for fifty milliseconds. Circumstances can prevent nodes from finishing their task. One such circumstance is when the robot has detected an obstacle. When a node finishes executing it returns to the parent node whether it finished on a stop condition or due to circumstances. This information is used by the parent node to select new behaviour.

To make it easier to follow the structure of the commands the robots follow, here is an outline of that structure presented in pseudo code. Some of the helper functions have been omitted for clarity.

```pseudo
// Top level node for behaviour
// follow line until notified of obstacle
// if a vehicle is detected, follow that vehicle
// if an obstruction is detected, notify other vehicles and change followed line
// repeat forever
DriveAlongRoad()

    ReturnParameter = InitialiseSteering() // centers the steering on the robot on startup
    start loop forever
        If ReturnParameter = success then
            ReturnParameter = FollowLine(until detected or notified of obstacle)
        If ReturnParameter = vehicle then
            ReturnParameter = FollowVehicle(until detected or notified of obstacle)
        If ReturnParameter = obstruction then
            notifyOtherVehicles(obstruction)
            ReturnParameter = ChangeLane(lineNumber)
    end loop forever

// Follow a line
// halt function when stop condition is met (detection or notification of an obstacle)
FollowLine(stopCondition)

    setSteeringAngle(getFeedBackFromLineSensor())
    setSpeed(MAX_SPEED)
    loop until stopCondition
        If CheckForObstacle() == true then
            return getObstacleType()
        end loop

// Follow a vehicle by keeping a certain distance from that vehicle
// halt function when stop condition is met (detection or notification of an obstacle)
FollowVehicle(stopCondition)

    setSteeringAngle(getFeedBackFromLineSensor())
```
setSpeed(determineFollowSpeed())
loop until stopCondition
    If CheckForObstacle() == true then
        return getObstacleType()
    end loop

// Steer the robot onto a new line
// repeat this sequence until a new line is being followed
SteerIntoNewLane(stopCondition, parameters, speed)
    setSpeed(speed)
    loop until stopCondition
        SteeringLaneChange(parameters)
        updateParameters() // adjusts the steering angle and speed
    end loop
    return success

// Change line
// gather required information in order for robot to change line
ChangeLane(lineNumber)
    steering_actions = getSteeringActions(lineNumber)
    loop forever
        ReturnParameter = SteerIntoNewLane(until following new line, steering_actions, LANE_CHANGE_SPEED)
        If ReturnParameter = success then
            return success
        end loop

5.5 Robot Design

The robots were designed to be as realistic as possible under the given circumstances. To simplify the complexity of the design it was decided to give them rear-wheel-drive and to use a differential which can split or combine motion. In a rear-wheel-drive car, the differential takes the motion from the motor and shares it between the two back wheels. In this scenario, the engine would be driving the differential itself (the dark gray piece in Figure 5.5) on one of the sides where it has gear teeth.

It was also decided to design the robot with Ackermann steering. The principle of Ackerman steering is the relationship between the front inside tyre and front outside tyre in a corner or curve. It is the term used to define the steering geometry where the inside tyre needs to turn
tighter than the outside tyre. This allows both tyres to roll around a common point in a corner or curve. This design allows for more accurate steering and a control system that works on degree to turn left or right.

Figure 5.5: Lego differential gears

Figure 5.6: Ackermann Steering

5.5.1 Modeling in MLCad

MLCad \(^8\) (Mike’s Lego CAD) is a powerful CAD system specifically designed to create building instructions for Lego models. MLCad reads and writes LDraw \(^9\) (a program of James Jessiman) compatible files but in an extensive window based environment. The program helps to create building instructions, which show step by step how to build a model. It also allows models to be virtually constructed when not all parts are physically available. This ability to test concepts in a virtual environment was used extensively when designing the robots.

As there are currently no LDraw parts available for the Mindsensors components they were omitted from the 3d model.

\(^8\)http://mlcad.lm-software.com/e_default.htm
\(^9\)http://www.ldraw.org/
Figure 5.7: Design of robot in MLCad
5.5.2 Completed Vehicle

A ray-traced image of the completed model is shown in Figure 5.8.

![Completed Vehicle](image)

Figure 5.8: Ray traced image of the final design

5.6 Testing

Initially the robot was tested on a small test track, two meters long and one lane wide, this was sufficient to test the line following behaviour. Next the robot was tested on a larger track, two meters long and two lanes wide, this allowed testing and verification of the lane changing behaviour and also testing of multiple vehicles.
Figure 5.9: One lane test track

Figure 5.10: Two lane test track
Finally a nine meter long and one meter wide track containing two lanes to follow was printed for full testing and demonstration purposes. This track can be seen in 5.11.

5.7 Evaluation

Due to complications with the ultrasonic sensors, which are detailed in the problems encountered section of this chapter, the evaluation scenarios had to be limited.

The evaluations that were run successfully were as follows:

1. Robot following a line.
1a. Robot following a line and changing lane in the event of an obstruction.
2. Four robots self organising into pattern described in the self-organising algorithm.
3. Robot notifying another robot of an obstruction in the lane.
3a. Notified robot changing lane to avoid obstruction before it could sense the obstruction itself.

Videos are available on the accompanying CD showing the evaluations.

5.8 Difficulties Encountered

Ultrasonic interference was far greater than had originally been anticipated. Multiple attempts were made to fix the problem, including putting a cover around the ultrasonic to limit its range. Other solutions such as programmatically changing the setting of the ultrasonic to use the event capture command rather than continuous measurement. When executing the event capture command, according to the NXT developers kit the sensor will measure if any other ultrasonic sensors are in the vicinity. With this information a program can evaluate when it is best to make a new measurement which will not conflict with other ultrasonic sensors. While this improved the readings, unreliable readings still caused problems with distance measurement when multiple robots were in use.

Other issues arose when the same software ran differently on apparently identical robots, the slight differences in the current power level of the battery had to be taken into account. Also, each robot had to be specifically tuned to the motors that it was assembled with, presumably due to slight variations in the manufacturing process of the motors.

Problems were also encountered when the final road, used for testing and evaluation purposes, was printed at the wrong scale. This led to lanes being much further apart than originally envisaged and because of this the infra-red sensors for detecting objects to the left and right

---

were now just beyond their range limits. The solution was to extend the width of the robots with the addition of paper 'wings' to their rear.

Figure 5.11: The four robots involved in testing on the nine meter test track

Figure 5.12: 'Wings' that had to be placed on the robots
Figure 5.13: Robot changing lanes to avoid an obstruction

Figure 5.14: Test of robots set to follow each other at 9 GSD
Figure 5.15: Robots self-organising as described in self-org algorithm

Figure 5.16: Alternate view of the robots self-organising
### 5.9 Summary

This chapter catalogued the process of creating a physical testing platform to allow the study of the TMS on robots. Details on the software and hardware utilised were provided. The specification and requirements for the robots were then described. Detailed information on the design of the software developed for controlling the robots was provided and the physical design process was explained. Testing procedures were then outlined. Finally the evaluation scenarios were listed and difficulties encountered were discussed. The next, and final, chapter concludes this report and proposes possible future work.
Chapter 6

Conclusions and Future Work

This chapter draws conclusions from all the previous chapters and then the details of possible future work are outlined.

6.1 Conclusions

The goal of this project was to investigate the efficiencies which could be applied to motorway traffic management through self organisation. This was to be done by creating and evaluating a self-organising TMS and then investigating the practicalities of implementing such a system on robots. This goal was successfully reached.

All the objectives listed in Chapter 1 were successfully achieved and the results showed that the presented self-organising TMS has indeed the potential to reduce travel times on a motorway, especially when the volume of traffic is reaching capacity.

The implementation on robots, while slightly restricted due to the problems previously described, did provide valuable insights into the application of intelligent systems on autonomous vehicles. In particular it showed that the data from sensors can sometimes be quite unreliable either due to interference or by the nature of their design. The slight manufacturing differences that occur in components also have to be taken into consideration. Neither of these insights would have been gained through simulation alone.

6.2 Future Work

Future work would involve adding more layers of realism to the problem. This would include introducing more complex wireless communication methods and adding different classifications of vehicles such as HGVs and emergency services vehicles.
Further sensors could also be added to the robots. The addition of GPS, accelerometer and compass sensors would greatly improve the ability of the robots to both navigate their environments and accurately represent their location.

The simulation and robot platforms could also be more closely integrated. One possibility could be to develop a system which interprets .NET intermediate language instructions on the Mindstorm’s microprocessor.

More robots could also be constructed and the scale of the real world testing increased to include a full track rather than just a limited stretch of lanes.

The algorithms could also be tested on larger scale robots such as the DSG golf buggies.
Bibliography


# Appendix A

## Abbreviations

<table>
<thead>
<tr>
<th>Short Term</th>
<th>Expanded Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS</td>
<td>Traffic Management System</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>V2I</td>
<td>Vehicle-to-infrastructure</td>
</tr>
<tr>
<td>V2V</td>
<td>Vehicle-to-vehicle</td>
</tr>
<tr>
<td>GSD</td>
<td>Grid Space Distance</td>
</tr>
<tr>
<td>DSG</td>
<td>Distributed Systems Group</td>
</tr>
<tr>
<td>TCD</td>
<td>Trinity College Dublin</td>
</tr>
</tbody>
</table>
## Appendix B

### Contents of the CD

<table>
<thead>
<tr>
<th>Folder</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>/DriverModel-SELFORG</td>
<td>Visual Studio 2008 Project. Creates a replacement driver model DLL for use in VISSIM.</td>
</tr>
<tr>
<td>/media</td>
<td>Collection of images and video from throughout the project.</td>
</tr>
<tr>
<td>/MLCAD</td>
<td>3D Model of the robot car, building instructions in sub-dir html.</td>
</tr>
<tr>
<td>/nxc</td>
<td>Software written to control the robot car.</td>
</tr>
<tr>
<td>/niallohara-fyp.pdf</td>
<td>This document.</td>
</tr>
</tbody>
</table>
Appendix C

Robot Building Instructions