NTNU – Trondheim
Norwegian University of
Science and Technology

Flocking for Road Traffic Efficiency Improvement

A Concept Study

Sindre Ilebekk Johansen
Andreas Sløgedal Løvland

Master of Science in Computer Science
Submission date: June 2015
Supervisor: Helge Langseth, IDI
Co-supervisor: Jo Skjermo, IDI
Tomas Levin, Statens Vegvesen

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis presents a concept study into the viability of flocking as a future road traffic management system for automatic vehicles. The idea is to remove the road lanes and let the vehicles themselves decide on the most effective road usage by dynamically allocating road space for either driving direction. The dynamic allocation achieved through flocking is more efficient than, e.g., lane barriers or traffic lights for changing direction of dynamic lanes.

Flocking requires each vehicle to follow a set of steering behaviors that, when followed by all vehicles, will result in emergent behavior.

In this study, we have implemented a road segment in Unreal Engine 4, along with multiple different vehicles to test the performance of flocking on diversified and homogeneous asymmetric road traffic. We have also performed some experiments to test flocking on homogeneous symmetric road traffic.

Our results show that flocking is a viable option for future road traffic management, and the emergent behavior is able to dynamically allocate road space as needed.

Further, the results show throughput on a 20m wide road can be increased by 72% with diversified asymmetric traffic, and 56% with homogeneous symmetric traffic. We believe some of the throughput increase is due to automatic vehicles being more effective than human drivers and not solely the achievement of emergent behavior.

Sammendrag

Denne tesen presenterer en konseptstudie av mulighetene for å bruke flokking til fremtidig styring av automatiske kjøretøy. Flokking går ut på å fjerne felt fra veien og la bilene selv bestemme hvordan veien kan brukes mest mulig effektivt ved å fordele veibredden mellom kjøreretningene. Den dynamiske fordelingen som kan oppnås med flokking er mer effektiv enn for eksempel fysiske barrierer eller trafikklys som kan brukes for bytte retning på dynamiske felt.

Flokking krever at alle kjøretøy følger et sett med *steering behaviors* (styringsregler) som, når alle kjøretøy følger dem, vil resultere i emergent atferd.

Vi har implementert et veistykke i Unreal Engine 4 sammen med flere forskjellige typer kjøretøy for å undersøke hvor bra flokking fungerer med diversifisert og homogen asymmetrisk trafikk. Vi har også gjennomført eksperimenter for a undersøke flokking for homogen symmetrisk trafikk.

Resultatene våre viser at flokking er et reelt alternativ for fremtidig styring av automatiske kjøretøy og den emergente atferden klarer å dynamisk fordele veibredde etter behov.

Videre viser resultatene at biler-per-time på en 20m bred vei kan øke med opptil 72% for diversifisert asymmetrisk trafikk, og øke med opptil 56% for homogen symmetrisk trafikk. Vi vil tilskrive deler av effektivititesøkningen til automatiske biler, siden de er mer effektive enn menneskelige sjåfører, og ikke kun til den emergente atferden alene.

Preface

This thesis is the final result of a master's project by Sindre Ilebekk Johansen and Andreas Sløgedal Løvland, and has been a joint effort of the Norwegian University of Science and Technology (NTNU) and The Norwegian Public Roads Administration (NPRA). We would like to thank our supervisor at NTNU, *Helge Langseth* for his support and feedback. We would also like to thank *Jo Skjermo* at NTNU for the project idea and continual support. His knowledge made this project possible.

At the NPRA, we would like to thank *Tomas Levin* for feedback on intelligent traffic systems and support in matters concerning future regulations of road traffic.

We would like to thank the hardware office at the NTNU Department of Computer and Information Science for providing the computers needed for the project.

Our sincerest gratefulness goes to *Anne Elster* and *Rolf Harald Dahl* for providing us with the computer cluster needed for running our experiments and setting up the machines.

Sindre Ilebekk Johansen, Andreas Sløgedal Løvland
Trondheim, June 11, 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Norwegian public roads will have to cope with more and more vehicles while funding keeps lagging. In this chapter, we present the major problems that need to be addressed and argue why our method is both economical and efficient.

1.1 Background

The number of cars in Norway is increasing. Figure 1.1 shows how the number of cars has increased over the last twelve years [SSB, 2015b]. It is reasonable to assume that there will be even more cars in the future. It is however not only the number of cars that is increasing; the car density is also increasing. Figure 1.2 shows how the car density per 1000 inhabitants have developed [SSB, 2015b]. The increasing number of cars demonstrate the future need for more roads and upgrades of the current road infrastructure. However, building new roads is expensive. The price per meter can vary from 60.000 NOK to 230.000 NOK [Garathun, 2014]. The price typically depends on the geological characteristics and the width of the road. A road between 12 and 22 meters wide will usually cost between 140.000 NOK and 230.00 NOK per meter. Besides the costs of building new roads, huge investments are needed to maintain the current road infrastructure. A report published by the RIF (Rådgivende Ingeniørers Forening) concluded that 1.6 trillion NOK is necessary to maintain the Norwegian public roads network and upgrade it to 2015 standards [RIF, 2015]. The Norwegian Transport Minister has also confirmed the dire state of the public roads and the previous Transport Minister states that the lag has been building up over the last 30-40 years [Engan et al., 2015]. In comparison, the entire 2015 Norwegian national budget is only 1.2 trillion NOK [Regjeringen, 2014]. The situation of increasing road traffic and lagging public investments gives good cause to search

for alternative ways of organizing road traffic. New ways could increase the road throughput and efficiency without requiring the same level of funding.

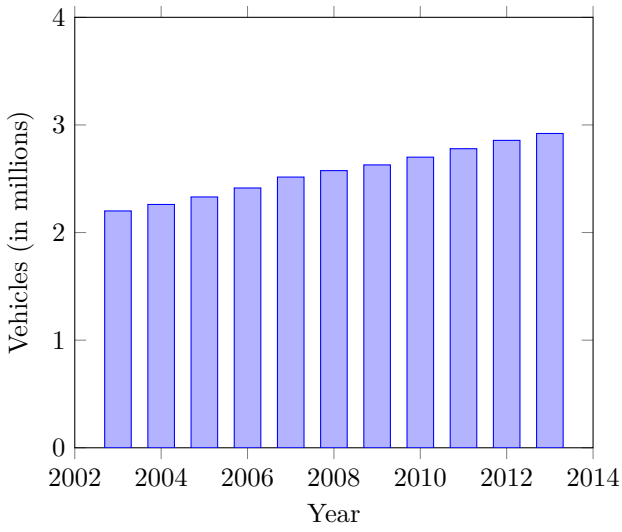


Figure 1.1: *The number of registered vehicles in millions per year in Norway. In 2003, there were 2.2 million vehicles in Norway and by 2013 the number has increased to 2.9 million.*

The next leap in road traffic efficiency will probably come from automatic vehicles, i.e. vehicles that can drive by themselves without requiring any interaction from the driver. There is a thin line between calling a vehicle automatic and calling it autonomous. An autonomous vehicle is a self-driving vehicle that can handle all road traffic scenarios on its own. It does not need to communicate with other vehicles and can cope with high degrees of uncertainty over an extended period (see Section 2.2.1). If a vehicle is automatic, it can communicate with other traffic entities (typically other vehicles or roadside infrastructure) to resolve issues or increase efficiency. Automatic vehicles already exist and are being test driven to ensure safety [Volvo, 2014], but legal issues are currently keeping them from the public (see Section 2.2.2).

Automatic vehicles may remove the causes of inefficiency due to human characteristics such as reacting slowly (e.g., when driving at a green light), or overreaction (e.g., a driver braking more than necessary to keep a comfortable safety margin), or simply driving slower than necessary. It could also be possible to have vehicles driving closer to each other due to faster reaction time from automatic vehicles that would enable them to stop in a shorter distance. Vehicles

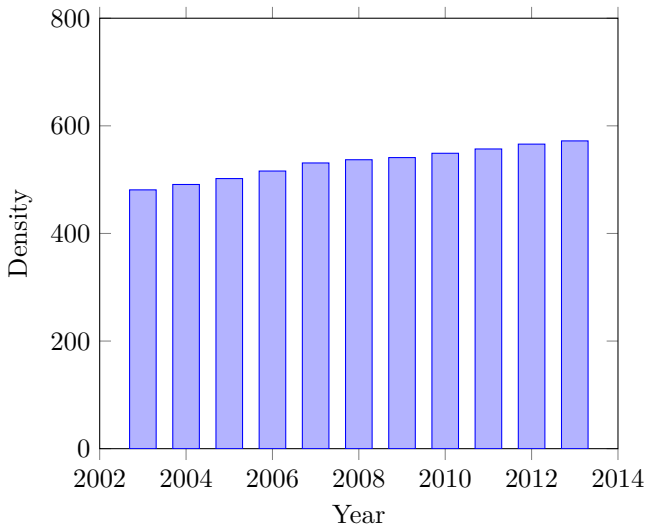


Figure 1.2: *The number of registered vehicles per 1000 inhabitants in Norway from 2003 to 2013. By the end of 2003, there were 481 vehicles per 100 inhabitants and by the end of 2013 there were 572.*

driving closer would mean increased road capacity.

No matter how effective automatic vehicles will be, they will still be limited to follow the lanes of the road. Particularly in the morning or afternoon rush the traffic is often jammed in one direction while there are almost no vehicles in the other direction. Dynamically assigning lanes to the direction that has the highest traffic load could improve efficiency ever further. Currently, solutions exist for shifting concrete lane barriers on the highway. A four-lane highway could, e.g., be divided into three lanes heading into the city in the morning and one lane out. In the afternoon, the barriers could be shifted back to allow three lanes out and one in [QMB, 2013; Cohen, 2015]. A lane where the driving direction can vary is called a reversible lane. Reversible lanes increase road throughput during rush hour but also introduces other issues such as: purchase of a ”zipper” vehicle to move the barrier when needed, the concrete barrier itself, and employment of labor to handle the lane management. A cheaper solution would be to change the direction of a lane automatically and dynamically as needed. Dynamically changing lane direction can be done by traffic lights signaling which direction a lane is used, but this solution increases the number of accidents [Sørensen, 2008] and is hence not a preferred solution.

When all vehicles drive automatically, there will no longer be a problem with

the driver forgetting the driving direction of a lane, and hence no increase in accidents. Automatic vehicles enable lanes to be dynamically assigned a driving direction without having to use lane barriers. We believe automatic vehicles can increase road traffic efficiency even further by removing the lanes and letting the vehicles use their sensors in combination with vehicle-to-vehicle communications to determine where on the road to locate themselves. Removing lanes will enable the whole road to be divided more dynamically between vehicles driving either direction and require no traffic lights or lane markings. Hence, removing lanes will decrease the cost of building roads while at the same time increase road efficiency and throughput.

1.2 Goals and Research Questions

In this research project, we want to investigate the feasibility of removing lanes from the road and letting the vehicles themselves determine road usage. The idea is that letting the vehicles themselves determine where on the road to drive will allow for a more dynamic allocation of road space in the direction of highest traffic load. There are some fundamental issues that need to be solved before vehicles can be let loose on the road: How do we ensure that vehicles do not collide? How do we merge traffic from entrance ramps? How do vehicles avoid oncoming traffic? How do we prioritize emergency vehicles? If we look towards the animal kingdom, we can find the inspiration we need.

An excellent example of animals traveling at high speed very close to each other while avoiding collisions is a flock of birds. These flocks appear to fly in unison without external guidance, they avoid all obstacles, can merge with other bird flocks and separate into multiple smaller flocks, they also avoid crashing with oncoming birds. Another example of animals mastering the lane-less mode of transport is a school of fish. How birds and fish can steer and avoid collisions was presented in [Reynolds, 1987]. The overall idea is that each individual flock member follows a set of rules that, when followed by all flock members, causes the *emergent behavior* of a flock. Emergent behavior occurs when multiple simple individuals interact and form more complex behavior as a collective. *Flocking* as a behavior is the behavior exhibited by a group of birds when they are foraging, in flight, or during murmuration. The rules followed by each individual flock member is what enables the individual to steer and avoid collisions. The rules of a flock individual are hence called *steering behaviors* or simply *behaviors*. We believe that if all vehicles had a common set of steering behaviors they would no longer need lanes, and road space could be allocated dynamically as necessary without overhead. The goal of this project is to investigate if automatic vehicles in combination with steering behaviors is a viable option for improving road traffic efficiency, hence the following research goal:

Research Goal *Investigate the viability of automatic vehicles in combination with steering behaviors to increase road traffic efficiency.*

Efficiency in this context means both that we aim to improve the overall throughput of road traffic, i.e., how many cars can pass a point or the road per hour, but also reduce the costs of building new roads as the lane markings and lane barriers would no longer be needed. Additionally we want to remove the rush-hour traffic jams that occur when one driving direction has a higher traffic load than the other.

Steering behaviors will control the movements of the automatic vehicles and must ensure that vehicles are driving in the same, or opposite direction do not collide. Hence our first research question:

Research Question 1 *Can steering behaviors ensure that vehicles driving in the same or opposite direction do not collide?*

No collisions could be achieved by having large safety margins around each vehicle. However, that would reduce the throughput of the road as each vehicle would require more space and less of the road would be used for actual driving. We would like the vehicles to be relatively close to each other while still maintaining a safety margin.

Ensuring that there are no collisions is possibly the most important concern for applying steering behaviors to automatic vehicles. However, there are other issues that a lane-less traffic system would need to manage. One such issue is entrance ramps where vehicles can enter the highway. Our next research question is concerned with managing vehicles merging into traffic from entrance ramps.

Research Question 2 *Can steering behaviors effectively manage the merging of traffic from entrance ramps into the highway road traffic?*

Research Question 1 and 2 are concerned with normal road traffic management, but there are special scenarios that require extra care such as emergency vehicles. An ambulance needs to arrive quickly to save lives, and currently the road traffic has to yield to let emergency vehicles pass. Can vehicles be made to yield by using steering behaviors as well? Research question three is concerned with prioritizing emergency vehicles over regular traffic.

The Norwegian government intends to increase funding of sustainable transport [Avinor and Transportetatene, 2013] and plans to be carbon neutral by 2030 [Klima- og miljødepartementet, 2014]. Part of the government's plan to become carbon neutral is to increase funding of railroads and buses. Currently, buses have their own lanes that can be used only by buses, taxis, motorcycles, and mopeds. Buses have their own lanes to ensure that the buses are not delayed by traffic congestion. To help promote public transport, it would be beneficial if

buses could be prioritized over regular traffic. In practice, prioritizing of buses could done by, e.g., letting buses drive slightly faster than the regular traffic and having the regular traffic yield to let buses pass. Research question three is concerned with how particular types of transport could be prioritized.

Research Question 3 *Can steering behaviors be used to prioritize emergency vehicles and public transport (buses) over regular traffic?*

Letting vehicles utilize steering behaviors to achieve emergent behavior could reduce the funding needed to build new roads. However, for the idea of flocking to be accepted as a way of managing traffic, we believe that the vehicle throughput needs to be considerably higher than what is currently observed (see Figure 1.3).

Research Question 4 *How does the vehicle throughput of road traffic systems utilizing flocking compare to regular traffic systems using lanes?*

We plan on using automatic vehicles for our experiments in accordance with Assumption 1 (see Section 1.4), and the vehicles will need some form of communication. The planned future communication for automatic vehicles is cooperative awareness messages (CAMs) (see Section 2.1.2). An inherent part of this project will be to determine if the planned future communication standard of CAMs will satisfy the requirements of self-driving automatic vehicles. The vehicles need enough information to avoid collisions with other vehicles and at the same time drive as efficiently as possible. Hence Research Question 5.

Research Question 5 *Will the future communication standard of CAMs [ETSI, 2014] be able to satisfy the requirements of safely guiding automatic vehicles without reducing the overall efficiency of the road traffic?*

Through answering our research questions, we plan to investigate whether flocking is a viable option for road traffic management. We plan on performing experiments and create simulations demonstrating the feasibility of each research question.

1.3 Motivation

We have summarized the main motivation of this project into the following list:

Increase Efficiency of Existing Road Infrastructure
We want the current road infrastructure to be able to handle a higher traffic load than today. Enabling vehicles to utilize steering behaviors can achieve a more dynamical allocation of road space and increase vehicle throughput.

Reduce Costs of Building new Road Infrastructure

The new road infrastructure would need no additional elements, e.g., barriers, lane markings, or traffic lights, which the current reversible lanes do.

Managing the road infrastructure will likely cost less as there is no need to, e.g., repaint lane markings, move the barriers of reversible lanes, or manage the traffic lights of reversible lanes.

A bonus effect of flocking is that people will arrive quicker because of higher road throughput. Less time will be wasted waiting in traffic jams, and hence the overall productivity of the drivers can be increased.

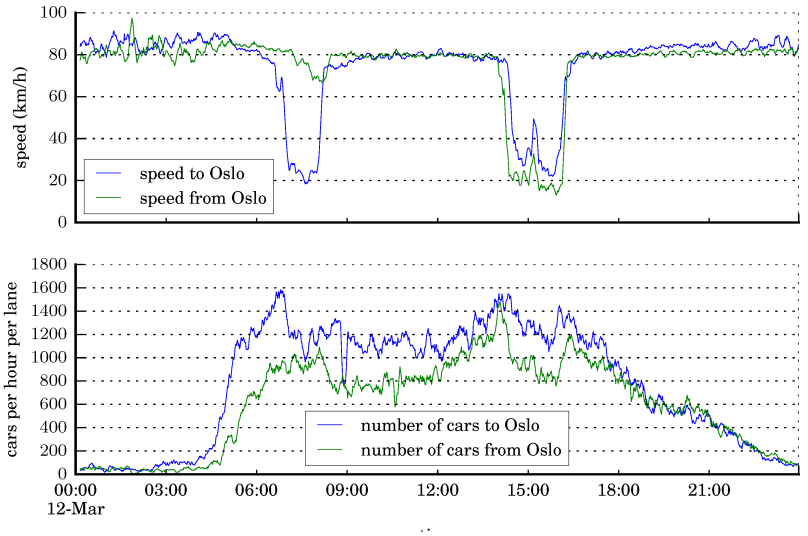


Figure 1.3: *Road traffic measurements of the Lysaker highway, Thursday the 12th of March 2015 [Levin, 2015].*

Figure 1.3 shows the amount of cars each hour, traveling to and from Oslo a Thursday in March 2015. The graph shows that there are fewer cars traveling to, than from, Oslo in the morning. Since the road must be able to handle the evening rush from Oslo, part of the road will be empty in the morning and the vehicles should be able to use this space more efficiently.

The graph also clearly shows that the road can only manage a fixed amount of cars per hour. When the number of vehicles reaches a certain limit, a traffic jam forms, and the number of cars per hour and the speed of the cars plunges.

Note that this limit is about 1600 cars per hour in Figure 1.3. When looking at graphs for different dates, it is clear that the limit is somewhere between 1400 and 1600 cars per hour for the Lysaker Highway.

1.4 Research Method

To answer our research questions, we will model a four-lane highway after Norwegian standards, an entrance ramp, and vehicles that will constitute the road traffic. Further, we will create multiple different steering behaviors for the vehicles that, in combination, will form their final steering behavior. The idea is to perform multiple experiments to investigate the feasibility of flocking and steering behaviors as a basis for traffic management. We are creating a computer model rather than running real-life highway experiments because a real life experiment would be significantly more costly.

An alternative to creating a computer model could have been to perform an entirely theoretical analysis of the concept of flocking as a road traffic management system, and determine if it would work or not. However, multiple species are already applying flocking in their everyday life, and hence we know that flocking is a viable concept and in itself possible. A flock of birds has no collisions and allows individual birds to enter or leave the flock as they see fit. Hence flocking for road traffic should behave similarly, but with some constraints: (1) vehicles can not move upwards or downwards, only forwards, backwards, and to the sides, (2) birds exercise what is called *free flocking*, i.e., flocking with no space limits, vehicles on a road will exercise *constrained flocking* because they will have to remain within the area of the road. None of the constraints should significantly impede the performance of flocking as a road traffic management system. Hence, a computer model that visually demonstrates whether flocking could be a proper future choice for road traffic management is much more interesting than the theoretical analysis.

To answer Research Question 1, 2, 3, and 4 we will create models and simulations that will demonstrate the feasibility of the concepts, or, if the concepts are not feasible, demonstrate the problems and reasons why. Research Question 4 will be answered by measuring throughput in every experiment and compare to throughput currently observed in traffic systems. We plan to measure the throughput of vehicles per hour as that is the Norwegian industry standard for measuring road traffic throughput [Levin, 2015].

1.5 Assumptions

In our experiments, we plan to have each vehicle follow a common set of rules. However, having every vehicle follow a certain set of rules is very unlikely to happen while we still have human drivers. Due to the limitation of human drivers we envision that steering behaviors for vehicles will be possible only after introducing automatic vehicles. Hence the following assumption:

Assumption 1 *We assume all vehicles in this project to be capable of what the European Road Transport Research Advisory Council have termed level 5 automation or full automation [ERTRAC, 2015].*

Level 5 automation is defined as: *"the full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver."* [ERTRAC, 2015]. ERTRAC has the following narrative explanation for level 5 automation: *"The fully automated vehicle should be able to handle all driving from point A to B, without any input from the passenger. The driver can at all-time override or switch off the system. Note: no realistic time estimation exists on this system."* [ERTRAC, 2015]. As the explanation states, the vehicles will handle all aspects of driving, and can not expect the driver to be available for input. The U.S. National Highway Traffic Safety Administration (NHTSA) has also defined levels of autonomy for automatic vehicles. What ERTRAC have termed level 5, the U.S. NHTSA have termed *Level 4* automation [NHTSA, 2013]. The U.S. NHTSA have the following definition for level 4 automation: *"The vehicle is designed to perform all safety-critical driving functions and monitor roadway conditions for an entire trip. Such a design anticipates that the driver will provide destination or navigation input, but is not expected to be available for control at any time during the trip. This includes both occupied and unoccupied vehicles."* [NHTSA, 2013].

The vehicles simulated in this project will transmit messages between themselves with information about important attributes that the vehicles will need to operate safely. We have the following assumption for the message transmittance:

Assumption 2 *We assume reliable communication between all vehicles.*

Reliable communication means that messages are guaranteed to arrive at their destination complete, uncorrupted, and in the order they were sent. Messages can in other words not be lost, contain corrupted information, or arrive out-of-order [Howe, 2015].

The messages transmitted by the reliable communication will include coordinates for the position of the transmitting vehicle (see Section 3.2.5). Hence, the vehicles will need to be capable of determining their position with high accuracy.

Assumption 3 *We assume all vehicles to be capable of positioning with negligible error.*

Assumption 3 is reasonable as it is possible to improve the GPS signals to provide locations with an error less than 10 centimeters with 95% confidence [ARINC Incorporated, 2008], see Section 3.2.5.

1.6 Thesis Structure

The thesis is structured into chapters with sections and subsections. The outline of the thesis is as follows:

- **Chapter 1: Introduction** contains an introduction to the problems road traffic is facing, the research goal of this project, and the research questions we will try to answer. We also present the motivation for performing this research.
- **Chapter 2: Background Theory** contains background theory about the road traffic domain, automatic vehicles, and flocking.
- **Chapter 3: Architecture/Model** contains the architecture of our road traffic model, vehicle model, steering behavior model, and our reasoning for designing the models as we did.
- **Chapter 4: Experiments and Results** describes our experiments, experimental setup, and the results of our experiments.
- **Chapter 5: Discussion** is where we discuss the results, incidents that occurred, and implications for future road traffic.
- **Chapter 6: Conclusion** presents the conclusions of our work with respect to the research questions, we also present a future work section with interesting topics for future projects.

Chapter 2

Background Theory

In this chapter, we present relevant background information for understanding our research goal and for understanding what earlier research our solution builds on. We explain what the future with automatic vehicles could look like and how automatic vehicles would work in combination with steering behaviors. We will explain why we have chosen Unreal Engine as our modeling tool, what future scenario we envision for automatic and autonomous vehicles, and what flocking is and how the research area of flocking came into existence.

2.1 The Road Traffic Domain

This section focuses on the aspects of road traffic that are relevant for our experiments. We focus on how the future road traffic may be governed and what standards have been developed to achieve a more intelligent road traffic system. The most central part of our experiments is the cooperative awareness message (CAM) for vehicle communication. We will explain what cooperative awareness is and how our vehicles are practicing it. Future road traffic will most likely adopt the cooperative awareness standards. Since we assume level 5 autonomous vehicles (Assumption 1) we should also adapt our experiments to standards that will exist in the future and affect future road traffic.

2.1.1 The Modern Road Traffic

Road Traffic typically takes place on public roads for the purpose of moving passengers and goods. Organized road traffic have many jurisdictions, e.g., lanes, intersections, and traffic lights. Road traffic is often categorized based on the size of the vehicles, *heavy-duty vehicles*(HDV) typically consists of trucks and trailers,

light-duty vehicles(LDV) consists of passenger cars and light trucks, *two-wheelers* consists of mopeds and motorcycles, and *soft modes* are typically bikes or people walking, but may also be kids on skateboards or wheelchair users. Generally soft modes are any transport that does not have a motor.

The number of cars incorporated into the road traffic system is steadily increasing, see Figure 1.1. More vehicles generate more pollution, especially when moving slowly in a traffic jam, and occupies significant amounts of space for roads and parking, hence, alternative ways of controlling traffic flow may soon prove useful.

In Norway, it is the Norwegian Public Roads Administration (NPRA) that plan, build, operate, and maintain national and county roads [NPRA, 2014a]. The NPRA has the primary responsibility for spreading knowledge and experience about *intelligent transportation systems* (ITS) [NPRA, 2014b]. ITS can help prevent traffic accidents, but also increase the efficiency of the driver. ITS is typically implemented using new information technology, enabling the driver to make more informed decisions. An example can be the usage of GPS in cars. The driver no longer has to be uncertain which direction to choose; the GPS will display both the geographic position of the vehicle as well as the direction to continue driving. Other examples include parking guidance, weather information, open/closed roads, and congestion warnings with alternate route recommendations [TOMTOM, 2015]. In principle, ITS does not refer to any particular single mode of transportation, but EU Directive 2010/40/EU (July 2010) [EU Parliament and Counsil, 2010] has the following definition for ITS:

> *"Intelligent Transport Systems (ITS) are advanced applications which without embodying intelligence as such aim to provide innovative services relating to different modes of transport and road traffic management and enable various users to be better informed and make safer, more coordinated and 'smarter' use of transport networks."*

It is clear from the definition that road traffic management is a part of ITS. Drivers should be supported to operate in a more informed manner and make more efficient usage of road infrastructure. For automatic vehicles to be able to collect information they somehow need to gather information about the surrounding environment. Wireless communication may be used to transmit information to automatic vehicles. A classic example is the GPS signals from satellites in space. However, more short-range wireless communications are also possible like the 802.11 standard [IEEE, 2012]. The abbreviation used for communication within ITS systems is ITSC (ITS Communications) or C-ITS (cooperative ITS).

2.1.2 ETSI Standards and Cooperative Awareness

The European Telecommunications Standards Institute (ETSI) [ETSI, 1988] have produced a standard for vehicular communications [ETSI, 2014]. The idea with vehicular communications is *cooperative awareness*. Cooperative awareness means that other road users and roadside infrastructure are informed about each other's position, speed, heading, and other attributes. Road users in this context mean all road users even bicycles or pedestrians. Roadside infrastructure is, e.g., traffic lights, barriers, or traffic signs. The central part of cooperative awareness is a *cooperative awareness message* (CAM). CAMs are constructed, managed, and processed by a *cooperative awareness basic service* (CA basic service). There is another ETSI standard that defines a communication architecture for ITS systems [ETSI, 2010]. This architecture standard defines how the communication of CAMs should be executed. The standard follows the principles of the OSI model [ISO, 1994] for layered communication protocols. The reason for designing a layered architecture is that the ITS architecture now constitutes a basis on which multiple other ITS applications can be built. In the ITS communications architecture, the CA basic service is located in the *facilities* layer that is the second layer in the architecture, below the application layer.

Cooperative awareness is used in other industries such as the automatic identification system (AIS) used by ships and vessel traffic services to identify, locate, and exchange information about vessels [NCA, 2011]. AIS allows vessels to be aware of each other's actions and increases safety. For the Norwegian Coastal Administration, AIS also improves marine traffic monitoring and marine traffic services. The AIS system enables vessels to transmit messages that contain their position, speed, course, identification, vessel type, dimensions, destination, cargo, estimated time of arrival, and draft. The information transmitted in AIS messages is similar to the information transmitted in CAMs and AIS serves as a practical example of a cooperative awareness system that is in use in everyday vessel traffic.

The CA basic service is a mandatory facility for all ITS-Stations (ITS-S) that are involved in road traffic. Vehicles will have vehicle ITS-S, and people will have personal ITS-S. The ETSI standard [ETSI, 2014] focuses on CAM messages sent from vehicles to other traffic participants. CAM messages can be transmitted from vehicles to multiple other entities, e.g., other vehicles (V2V), roadside infrastructure (V2I), or any other entity with an ITS-S (V2X).

CAMs are messages transmitted periodically between ITS-S in the ITS network to maintain awareness of each other and to support cooperative performance. There are two types of fields in a CAM message, (1) mandatory fields that each message must contain, and (2) optional fields that can be included if desirable. A description of relevant fields in a CAM can be found in Appendix C. The most relevant fields for our experiments are the heading, speed, driving

direction, vehicle width, vehicle length, and longitudinal acceleration.

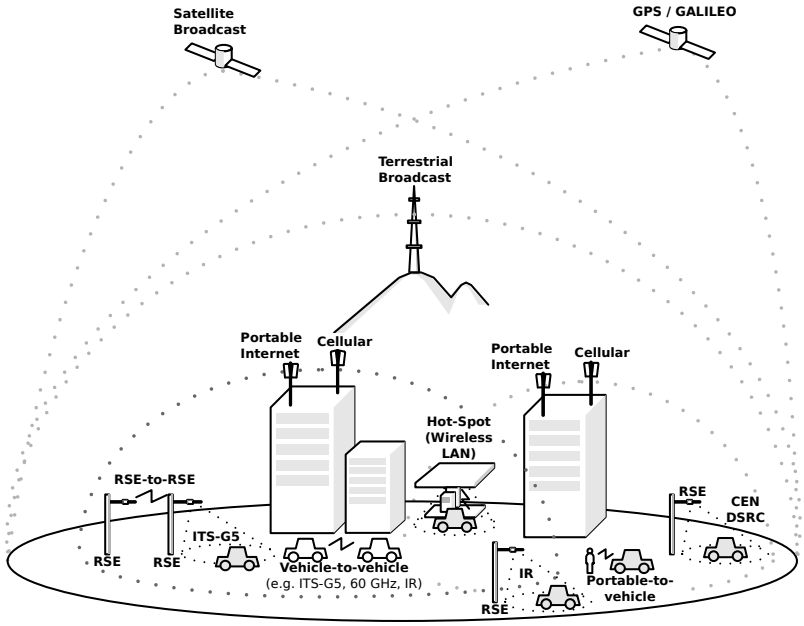


Figure 2.1: *An illustration showing how the various forms ITSC is intended to operate. (Figure source: [ETSI, 2010].)*

Figure 2.1 shows a draft from the ITS standard [ETSI, 2010] that outlines how communication between various road traffic infrastructure can be realized. GPS can be used for location on a global scale, terrestrial radio towers will be used for area transmissions, and then various forms of short range communications will be used on a street-level scale. Roadside equipment (RSE) will be able to communicate with other RSE units or vehicles using, e.g., infra-red (IR), the IEEE 802.11p based ITS-G5 protocol [ETSI, 2011], or CEN-DSRC (Commission Européen de Normalisation - Dedicated Short Range Communication [ISO, 2013]). However, due to the lag in infrastructure funding it could prove challenging to realize the ITS visions of ETSI. An alternative could be to use the already existing 4G mobile network [Valle, 2009; Telenor, 2015].

2.1.3 Overlap Phase Between Road Traffic Systems

The current road infrastructure is adapted to human drivers. When automatic vehicles start to fill the streets, simplifications to the current road infrastructure can be made. For most western countries, the change in infrastructure will be

to start building roads without lanes or central dividers. Where central dividers have already been built, they would need to be removed. However, some countries around the world do not have lanes or central reservations in the first place. These countries could have an easier transition period. The best example of current road traffic without lanes is probably in India where there are no lanes, but the drivers cope quite well nonetheless. Autonomous vehicles in the absence of speed lanes have been researched for the purpose of developing a steering behavior adapted to the absence of lanes [Kala and Warwick, 2012].

When automatic vehicles become available, traffic will be a mix of automatic and manually driven vehicles. It is not only the streets that could be rearranged to increase efficiency; intersections can also be adapted. Managed intersections have been researched [Dresner and Stone, 2005], that could exchange information by e.g. CAMs. The challenge is that people can not receive CAMs and respond to them by themselves, they would need extra devices installed in their vehicles. A possibility for this overlap phase is to use unmanaged intersections [VanMiddlesworth et al., 2008]. Unmanaged intersections may even be used permanently in more rural areas where it is not socially economically sane to invest in a fully managed intersection as they are expensive.

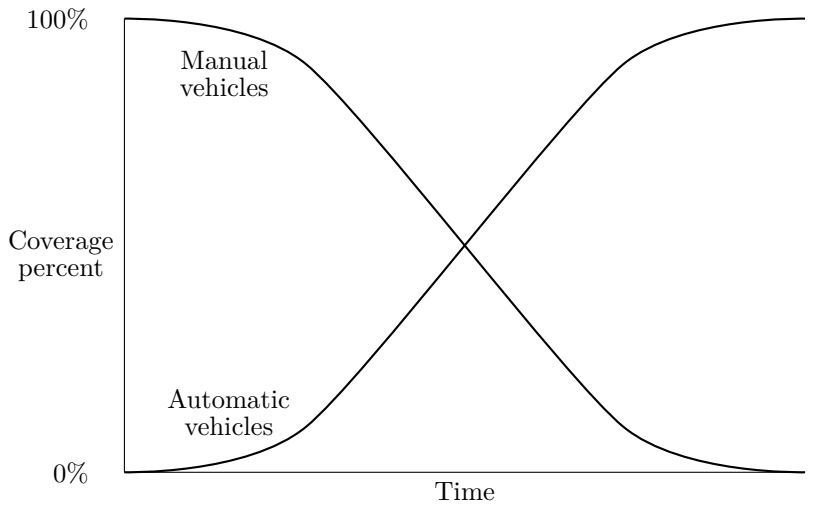


Figure 2.2: *A typical S-curve graph that depicts the transition from manually driven vehicles to automatically driven vehicles. In the beginning, 100% of all vehicles will be manual, and then more and more people will acquire an automatic vehicle, and in the end all vehicles will be automatic.*

Figure 2.2 shows how the transition for manually driven vehicles to automat-

ically driven vehicles may occur. One possible way of proceeding with infrastructure adaptions is to wait till all vehicles are automatic and then start adapting roads to the automatic vehicles. However, it is not necessary to wait till all vehicles are automatic. Once a certain percentage of the vehicles are automatic, roads could be classified into two categories, one where only automatic vehicles are allowed, and one where both automatic and manual vehicles are allowed. Such a classification may also increase the rate of people acquiring automatic vehicles, as it will enable them to drive on all roads and not only the ones categorized as allowed for manual drivers.

The switch to automatic vehicles will probably not happen overnight. There will be an overlap phase where vehicles driven by humans and automatic vehicles will coexist. An average Norwegian personnel car is 18.5 years before it is scrapped [SSB, 2015a], and the average age of a Norwegian personal car is 10.5 years [SSB, 2015c]. Hence, it is reasonable to assume that the overlap phase will last for at least ten years before all vehicles are automatic.

One approach to handling the overlap phase is to build infrastructure that is capable of handling both automatic vehicles as well as human drivers. Research is being done with respect to having automatic cars drive with the current infrastructure that includes traffic light [Kaneko and Shimamura, 2000], even though automatic vehicles will probably not need traffic lights as there are much more efficient ways to manage an intersection [Dresner and Stone, 2005]. However, the need for intersections capable of handling both automatic and manual drivers is recognized and have been researched [Dresner and Stone, 2006; VanMiddlesworth et al., 2008].

2.2 Automatic and Autonomous Vehicles

Automatic vehicles, i.e., level 5 automatic vehicles by Assumption 1, constitutes the basis for our experiments. Hence, we view it as important to have an overview of how automatic vehicles would work and how far into the future automatic vehicles are. In this section, we present the basics of the technology used to create an automatic vehicle.

2.2.1 The Automatic Technology

In a modern car, several of the functions a human driver used to have are now handled by the car itself. When the driver brakes, the car uses an anti-lock braking system (ABS) [Douglas and Schafer, 1971] to avoid the wheels locking. Wheels that do not lock have higher friction against the road surface and a shorter stop distance, and hence greater chance to avoid accidents. If the road is icy and the car starts to lose traction, it is possible for an electronic stability

program (ESP) [Liebemann et al., 2004] to apply braking to individual wheels to help steer the vehicle. It is even possible for the car to avoid skidding in the first place by utilizing a traction control system (TCS) [Farr, 1990]. The TCS detects mismatches between throttle input and engine torque, and the road surface conditions. The TCS can then apply countermeasures as, e.g., reducing engine power output.

The peak of automation in current vehicles is the adaptive cruise control (ACC) [Bhatia, 2003] (sometimes also called autonomous cruise control). ACC will determine the distance to the vehicle in front and maintain a certain distance to it. Earlier versions of ACC did only brake through reducing the power output of the engine, but today's ACCs will actively use the brake if necessary [Toyota, 2000]. ACC typically uses either a laser or radar for determining the distance to the vehicle in front [Mitsubishi, 1992; Jaguar, 1998]. The distance to objects in front of the vehicle can be measured even though the cruise control is not active. Hence, ACC can warn the driver if a person, animal, or general object is in front of the vehicle and the driver needs to stop. A system that warns the driver of possible obstacles that needs to be avoided is called a collision avoidance system. The mentioned technologies combined can handle all of the tasks a normal driver would have to perform except turning the steering wheel. Here is where automatic vehicles enter the picture. An automatic vehicle could sense the environment and determine the correct steering behavior by itself the entire trip without expecting any intervention from a human driver at all.

Autonomous means to have the freedom to govern itself or control its own affairs [Press, 2015]. However, this does not mean that any system that is able to perform its own actions is autonomous; then almost all systems would be autonomous. Autonomous also means being able to cope with a high degree of uncertainty during an extended period of time and compensate for failures without external intervention [Antsaklis et al., 1990]. Thus, if a vehicle communicates with other vehicles to determine the situation, it must be regarded an automatic vehicle rather than an autonomous vehicle. To sense its surroundings and gather information an automatic vehicle uses technologies like radar [Clark and Durrant-Whyte, 1998], lidar [Levinson et al., 2011], GPS [Cui and Ge, 2003], and computer vision [Chapuis et al., 2002; Konolige et al., 2008]. The information is used to, e.g., model a 3D map of the surroundings, detect obstacles, pedestrians, other vehicles, or determine the correct direction to steer. An interesting and exciting feature of using lidar, radar, or GPS for navigation is that these techniques work just as well regardless of whether it is night or day. Human vision is dependant on daylight to function optimally and has reduced functionality during night hours. Though automatic vehicles indicate performance that will surpass what can be expected from humans, automatic vehicles still have their problems. Snow and heavy rain can disturb the sensors and as of August 2014 the Google self-driving

car has, due to security reasons, yet to be tested in either [Gomes, 2014].

Some types of road traffic have priority over others. A car should not move into the path of a police, fire, or emergency medical vehicle displaying a flashing blue light [Samferdselsdepartementet, 1986]. It is possible that vehicles could detect blue light using a computer vision system [Ortiz and Neogi, 2006], however, a more reliable approach might be a message broadcast that can be signed with digital signatures as suggested in [VanMiddlesworth et al., 2008]. Digital signatures will not only allow detection of emergency vehicles, but also improve security. With digital signatures, all vehicles can verify that an emergency vehicle is a legit emergency vehicle and not some impostor vehicle trying to impersonate an emergency vehicle.

2.2.2 Legal Issues

From the smallest automatic windshield wiper to the fully autonomous car, there are many levels of automation. ERTRAC has split automation into two groups based on who is monitoring the driving environment [ERTRAC, 2015]. The first group consists of level 0-2 and is characterized by the driver monitoring the driving environment. The second group consists of level 3-5 and is characterized by the vehicle monitoring the environment. In the US, the National Highway Traffic Safety Administration has divided automatic vehicles into four levels based on how automatic the vehicle is [NHTSA, 2013]. The level of autonomy is important because as long as a human driver is controlling the vehicle, the driver may be liable for any damages that occur. However, when the driver is not exerting any control over the vehicle, the vehicle itself is responsible, but it does not make sense to hold a car responsible for a crash. Should the car then go to jail? An autonomous vehicle is supposed to drive by itself without crashing or creating any damage. If the autonomous vehicle still crashes or creates damage, it must be regarded as a defect product and hence the producer should be held liable [Gasser, 2012].

According to ERTRAC, one of the main concerns for introducing highly autonomous vehicles is the legal framework [ERTRAC, 2015]. The majority of road traffic in Europe is governed by the 1968 Vienna Convention on Road Traffic [UN, 2015]. This convention ensures a level of conformity between the road traffic systems of signing countries and eases the driving experience for a driver who does not have to learn a new set of traffic rules for every country. An issue with the Vienna Convention, concerning automatic vehicles, is that it defines a driver as a person [UNECE, 1968]; hence automatic vehicles are consequently not legal in most European countries. Automatic vehicles not being legal, is an issue for Germany, France, and Italy whose high-end car manufacturers have started to develop automatic vehicles and need a change in legislation to be able to sell

them. Hence, Germany, France, and Italy have pushed amendments to the 1968 Vienna Convention that allows automatic driver systems as long as the system can be switched off or overridden by the human driver [Safe Car News, 2014; ECE, 2014].

Google has been pushing the development of automatic cars and has been granted patents for their technology [BBC, 2011]. The patent is, however, not a fully functioning autonomous vehicle, but concerned with a method of switching an autonomous vehicle between modes of being fully autonomous and being driven by a human driver. Autonomous vehicles will contain large quantities of technology that will probably be patented. With smartphones, all the new and innovative technology caused the origin of the *patent wars* [Duhigg and Lohr, 2012]. We can only hope this does not happen in the automobile industry as well as it could delay the introduction of automatic vehicles.

There are many legal issues that need to be solved before autonomous vehicles can enter the streets. The first issue of any driver is to get a license. This is also the case for the autonomous vehicles. In Nevada, the state has approved the first self-driven US vehicle license [BBC, 2012; Nevada Senate, 2011], later California [California Senate, 2012], and Florida [Florida Senate, 2012a,b] have also allowed autonomous vehicles. In Europe the Economic Commission for Europe (ECE) has made amendments to the 1968 Convention of Road Traffic to allow autonomous vehicles, as long as *"…such systems can be overridden or switched off by the driver."* [ECE, 2014]. This will require ECE member countries to incorporate the new rules into their respective legislation. Some European countries have already adopted the law change and allowed driverless vehicles, e.g., UK where the government has announced that driverless cars will be allowed on public roads from January 2015 [BBC, 2014], or Sweden where Volvo is preparing to put 100 production-ready self-driving vehicles on the road by 2017 [Larsson, 2014]. In Norway, the 1968 Vienna Convention on road traffic still applies, which means that a driver is still defined as *"…any person who drives a motor vehicle, …"* [UNECE, 1968]. Hence, autonomous vehicles are not yet legal in Norway.

All producers of autonomous cars are using very advanced technology. There are vulnerabilities to this technology, particularly the GPS signals and the network communication. What if a car is hacked? Would it then be possible for an attacker to crash the car and kill the passengers? Could the car be used as a driving projectile, killing pedestrians? The US Federal Bureau of Investigation (FBI) is worried about the consequences of autonomous vehicles [Harris, 2014]. If the criminals no longer have to be driving the car, then a scenario where suspects shooting at pursuers from getaway cars that are driving themselves becomes possible. Criminals no longer have to dedicate both hands and vision to driving which opens up possibilities for usage of advanced weaponry while driving. It will also be possible to program an explosive-packed car to drive a specific route

and then detonate like a self-driving bomb. The FBI does not view autonomous vehicles purely as bad news though, they believe surveillance will be easier and more effective, the autonomous car can perform awkward maneuvers that might otherwise delay response time, and the number of accidents involving emergency vehicles may be reduced. In the US, in 2012, the number of persons killed in crashes involving emergency vehicles were 83 [NHTSA, 2013].

For the engineers who work on the autonomous vehicles, there are ethical issues with implementing the steering behavior. Some of the issues are well demonstrated by the Open Roboethics Initiative (ORi) [Moon et al., 2012]. If e.g. a death by an autonomous vehicle is unavoidable, should the vehicle run over a child, or should the vehicle crash into the mountainside and kill the driver? This dilemma [ORi, 2014] and other dilemmas are posted online, and visitors can answer polls to indicate the popular opinion. The reason for the initiative is to help legislative authorities and policymakers to create thoroughly discussed laws and policies, and foster active discussions of ethical, legal, and societal issues of robotics. Regardless of what the answer to a dilemma is, it is up to the engineer to implement the behavior of the autonomous vehicle and to have a thoroughly discussed set of policies can be of great help.

2.2.3 Current State of Automatic Vehicles

In this section, we present what a selection of car manufacturer are working on and give insight into what levels of autonomy have been achieved so far.

Mercedes have built their F 015 concept car that represents what Mercedes imagines their future cars might look like [Davies, 2015]. However, the car is far from production-ready, and only represents a peek into what the future may look like.

Audi has driven an A7 Sportsback from Silicon Valley to Las Vegas, a distance of about 860km [Rügheimer, 2015]. The car was automatic in the sense that it could handle lane changes, adjust speed, and overtake other vehicles. The speed range covered by the car was 0km/h to 112km/h. Sensors used were multiple radars in both the front and rear as well as laser scanners to provide redundant information about dynamic or static objects in the vicinity. Some sensors are aimed at the left and right to provide a 360° view. Multiple cameras are also used, a 3D camera in front and four smaller additional cameras mounted in the front and rear. The system can drive by itself on highways, but requires the driver to resume control in city environments. The driver is notified by lights and acoustic warnings that interaction in needed, and should the driver choose to ignore the warnings the vehicles will be brought to a halt.

Audi has also created a self-driving racing car that has driven around the Hockenheim circuit in Germany [Ramey, 2014]. The car had 560 horse powers

(≈418 kilowatts) and reached a speed of 220km/h.

Tesla is developing an autopilot for a car that will be in their Model S [Ziegler, 2014]. The autopilot has been created by incremental development and is currently able to take control on highways, park itself in a garage, and intervene when the car believes a crash is imminent. In 2014, Tesla's CEO, Elon Musk, said that completely autonomous cars are still 5 to 6 years away [Lowensohn, 2014].

Volvo is also participating in the race towards self-driving cars. Their ambition is to have 100 fully autonomous cars, driving regular people, on the road in Gothenburg by 2017 [Larsson, 2014]. The cars are currently on the road but are only driving scientists and engineers. The cars can drive by themselves on highways, perform parking, and pick up passengers. The technology used is much the same as the other manufacturers; radars, cameras and GPS. Volvo has also been involved in the SARTRE project [SARTRE, 2014], which is aimed at reducing fuel consumption and emissions of toxic gasses by combining vehicles into *road trains* [Davila et al., 2013]. A lead vehicle will drive in front of a column of vehicles. The concept is the same as a vehicle in front with a trailer behind, only the tow bar is virtual. Hence, multiple trailers can be attached. However, the trailing vehicles are not simply trailers; they are individual autonomous vehicles that have their own engine, sensors, and safety systems.

Toyota has displayed a research vehicle that they will use to develop a higher level of autonomy in their cars [Simonite, 2013]. However, Toyota has a slightly different approach to autonomous cars, as Vice President at Toyota's Technical Administration Planning Office said: *"The human being is the ultimate in sensor fusion. We have the visual, audible advantage, all the different inputs to make the best judgments moving forward"* [Atiyeh, 2014]. Toyota means to introduce more semi-automatic security features while still keeping the driver in control. However, they are still conducting research on autonomous cars as an R&D project and have teamed up with Nissan and Honda as well [The Rakyat Post, 2015].

BMW is also researching autonomous cars, and their i3 model can automatically park itself and come pick passengers up when they are ready [Tilley, 2015]. The car has four laser scanners on each side to get readings of the environment, but the car also still needs a map of the parking space to be able to find parking locations.

Google is not a company that is usually associated with cars, but that might change soon. The Google self-driving car project is aimed at creating a vehicle that can handle all aspects of driving [Urmson, 2014]. Google is pushing hard to realize self-driving and has a lead on its competitors. The Google self-driving car can drive on highways, perform parking, and drive around in the city. Driving in the city is one of the big obstacles that needs to be overcome before fully autonomous cars can be a reality. It is difficult for an autonomous car to drive in

the city due to the complex traffic situation. For the Google Car to drive around in the city, Google uses the same trick that BMW uses for their i3 model to park itself. The trick is to have a map of the area with machine readable information. Google is currently collecting massive amounts of data and has created a *virtual track* of Mountain View in California which contains information about where the curbs are, signs, traffic lights, and even how high the lights are off the ground [Madrigal, 2014]. When the Google Self-driving car drives around Silicon Valley, it also drives around the virtual world created by the Google engineers. The data for the route is pre-loaded into the car before the trip starts, this lets the car know what to expect. Other car manufacturers like Volvo, Tesla, or Audi lets the car process the entire scene from scratch, which is a much harder problem. Making a virtual map of the road converts the problem of interpreting the real-world environment, to detecting differences between the map and the environment. There is effort associated with creating a virtual-reality map that the Google car can utilize, as of 15 of May 2014 Google has mapped 3200 km of road in the US of a total of 6.4 Million km [ARTBA, 2014]. Researchers at Google have great confidence in huge amounts of data for solving complex problems [Halevy et al., 2009], and the Google Self-driving car has been described as working so well *"...it's boring."* [Markoff, 2014]. Google co-founder Sergey Brin has said the technology for autonomous cars will be available from 2017, but it may not be commercially available for the general public before 2020 [Markoff, 2014].

We believe that it is only fair also to mention Apple's endeavors into automated driving, even though they are very secretive about it. According to industry sources, Apple is researching every aspect of self-driving cars [Taylor and Oreskovic, 2015]. Apple is gathering information about how to build a self-driving car from the bottom up. It could be that Apple will produce their very own car at some point in the future.

ERTRAC is worried about security issues as insecure communication may open the road traffic system for abuse, criminal, or terroristic attacks [ERTRAC, 2015]. Other researchers are also worried about the security risks of autonomous vehicles. Petit and Shladover have published work where they investigate the feasibility, damage, and mitigation techniques of various attacks against autonomous vehicles [Petit and Shladover, 2014]. Their conclusion is that there is a need for considerably more redundancy than many have been expecting. The redundancy will be used to construct overlapping security measures that would allow a vehicle to detect whether it is attack by, e.g., a GPS spoofing attack or forged network messages.

After reviewing the work of multiple car manufacturers and also the renowned Google car project it is clear that fully autonomous cars for the public will not be commercially available until 2020. Level 3 automation, as defined by both ERTRAC and NHTSA, is what is currently being developed by most car man-

ufacturers. Level 3 requires the vehicle to be able to cede full control of all safety-critical functions to the driver. The driver is still expected to be available for occasional control, but there needs to be a sufficiently large transition time. When level 4 automation arrives, also as defined by both both ERTRAC and NHTSA, the driver will no longer be expected to be available for control at any time, and this includes both occupied and unoccupied vehicles.

2.3 Background Theory on Flocking

Flocks of birds are delightful to watch and gives the impression of a very synchronized system. However, every bird is simply following a set of steering behaviors and pays attention to a few close neighbors. Flocking constitutes the foundation for creating the steering behaviors for our automatic vehicles. Hence, we believe it is important to understand the background of flocking to better understand the possibilities and limitations of what can be achieved by steering behaviors. This section presents an analysis of what flocking is, how flocking works, the complexity needed for flocking to occur, and also discusses individual behaviors as a basis for emergent behavior. We also discuss different types of steering behaviors, how steering behaviors can be represented by vectors, and how they could be modeled and implemented in a computer program.

This section builds upon a structured literature review that can be found in Appendix B.

2.3.1 Flock Complexity and Emergent Behavior

Up until 1987 computer simulations of flocks had mainly been created by scripting the path of every individual flock member. Scripting the behavior of every single flock individual, required a significant amount of manual labor. Specifically, the work required to simulate a flock was $\mathcal{O}(n)$ where n is the number of individuals in the flock. A linear increase in workload may not appear as much, but considering it has to be done manually it becomes a very limiting constraint. What Reynolds did in 1987 was to describe a flock as a particle system [Reeves, 1983] where each individual flock member was implemented as an individual particle [Reynolds, 1987]. Reynolds used birds as an example to describe his flocking algorithm. A flock of birds consists of multiple discrete birds that all have their own mind and follow the rules of flocking based on their observations. Yet, the flock still appears to be moving as if there was some central control. All evidence, however, indicates that every single bird is moving based on its own local perception of the world [Reynolds, 1987]. The approach of Reynolds assumes that the behavior of a flock is the result of interaction between individual behaviors of flock members.

This type of flock behavior without central control is an example of *emergent behavior*.

Reynolds introduced the word *boid* for a bird-like object. This word is also used to represent the individual member of a flock, even when the creature in question is not a bird, e.g. a school of fish.

Even though Reynolds used particle systems to model a flock of birds there are some fundamental differences between the two. First off, a boid has geometrical shape, a difference that mostly comes into play for visual purposes where a particle is visualized as a dot and a boid covers a volume. The boid's volume is important for collision detection. Second, boids have a geometrical state: direction. Particles are represented as a point and can move in a direction, but the direction is not part of their state. Thirdly, particles do not need to interact with each other. Albeit this is not ruled out by definition, e.g., it is entirely possible for particles to interact. However, birds and boids must interact to achieve the correct flocking behavior. In other words, the behavior of a boid is dependent on both its *internal* and *external state*.

An interesting aspect of natural flocks is that there does not appear to be any upper boundary on flock size. Flocks do not become full or overloaded. When the herring travels towards their mating grounds, they travel in schools that may contain millions of fish [Shaw, 1970]. It would seem impossible that a single fish could be paying attention to the actions of every other fish. Hence, a fish can only be paying attention to the fish that are closest to it. Flocks or herds on land appear to be behaving the same way, each individual observes the actions of the closest flockmates and acts in a manner consistent with the observations. A bird might be aware of three categories: itself, its two or three nearest neighbors, and the rest of the flock [Partridge, 1982]. The feature of only paying attention to a few of the neighbors in a flock is that it lowers the *computational complexity* of participating in the flock. In computer science terms the birds are using a *constant time algorithm*, i.e. there is a fixed amount of work for each bird participating in the flock and the complexity can be described as $\mathcal{O}(1)$. Flocking complexity analysis is important because it dictates what sizes of flocks that can be expected, both in real nature and in simulations. A constant time algorithm for flocking means that there is no "maximum size" for a flock in real life, but the maximum size is rather dependant on how many birds are available. For computer simulations, however, there is a limit as each boid will have to be processed at least once per iteration. The complexity of iterating over every boid is $\mathcal{O}(n)$, and for each boid we have to update its state with respect to its nearest neighbors which is a constant time operation: $\mathcal{O}(1)$. In total the running time is $\mathcal{O}(n * 1) = \mathcal{O}(n)$, which is the same as in 1987, only now it can be done by computers and is hence much faster.

2.3.2 The Different Steering Behaviors

For natural birds there seem to be two balanced opposing behaviors that allow flocking and emergent behavior to occur: (1) A desire to stay close to the flock, and (2) a desire to avoid collisions [Shaw, 1975]. The desire to avoid collisions is easy enough to understand from an evolutionary perspective. Birds colliding mid-air may have fatal consequences. The urge to flock in the first place can be related to several evolutionary factors, e.g., protection from predators; a higher chance of survival for the shared gene pool, searching for food more efficiently by a larger search pattern, and social and mating activities [Shaw, 1970]. In a boid these natural urges are simulated by a set of *behaviors*. Reynolds have defined three such behaviors:

1. Collision Avoidance: avoid collisions with nearby flockmates.
2. Velocity Matching: attempt to match velocity with nearby flockmates.
3. Flock Centering: attempt to stay close to nearby flockmates.

Collision Avoidance is the behavior that reflects a boid's desire to avoid collisions. A boid will observe its surrounding flockmates and determine if the distance between them is appropriate. If the distance is too small, the collision avoidance behavior will return a steering vector that points in a direction away from the flockmates to increase the distance. The collision avoidance may also incorporate the velocity of the surrounding flockmates to avoid a future collision.

Velocity Matching is a behavior that is more directly focused on generating emergent behavior. A boid will seek to match speed with its flock neighbors and then maintain that speed. Velocity matching also helps to avoid collisions because a boid will not try to move any faster or slower than the neighboring boids and hence status quo will be maintained. A dilemma with Velocity matching is that it is a behavior created by Reynolds to make flocks more attractive to watch. However, having visually appealing flocks of vehicles may not be a major concern for conducting experiments. The main reason for vehicles to flock is reduced drag and hence flocking behavior that produces practical advantages should be prioritized.

Flock Centering is the behavior that keeps a flock together and also determines the compactness of the flock. Flock centering is the behavior that urges a boid to stay close to the center of the flock. A boid will not incorporate the entire flock in its calculations of where the center of the flock is, but rather calculate the *local center* of its neighboring flockmates and seek towards that center. If all boids follow this behavior, they will seek together and remain in closed formation. The boids in the center of the flock will have an approximately homogeneous population density in every direction and the flock centering urge will be small.

On the other hand, if the boid is on the outskirts of the flock all the other boids will be on one side, and the centering in that direction will be strong.

2.3.3 Summation of Steering Vectors

The resulting behavior of a boid is typically a sum of all the individual behaviors. Each behavior will use information about the environment and determine to either remain where it is or move in a direction. When a behavior determines that movement is necessary it will return a vector that will be accumulated with the other behavior vectors into the resulting movement direction. The vector returned from each behavior is called a *steering vector*. A steering vector has both a direction and a magnitude. The direction indicates the direction the behavior wants to steer, and the magnitude indicates the desired speed. A typical way of accumulating the vectors from the different behaviors is to use a weighted sum.

Collision avoidance is a behavior that typically will not return any vector if there is nothing nearby that can be collided with. Hence this behavior will only occasionally return a vector with a preferred movement direction, but when a vector is returned it is imperative that this vector be given priority (otherwise a collision will probably occur). Giving certain behaviors priority over others is typically done by adjusting the weights of the behavior. Weight adjustments ensure that critical behaviors are given priority while still accommodating the behaviors of lesser importance in some degree. The result is a natural motion that appears smooth and fluid. An alternative to using a weighted sum is to use a *prioritized acceleration allocation* (PAA). This is the approach taken by Reynolds in his 1987 paper [Reynolds, 1987]. PAA accumulates the steering vectors up to a certain threshold and simply ignores any additional behavior input. This approach can be optimized by not computing steering vectors for additional behaviors if the threshold have already been reached. For PAA to work the behaviors needs to be considered in prioritized order. The collision avoidance will typically be considered first, and any resulting steering vectors will be added to the sum. If the threshold has not yet been reached, the velocity matching behavior can now be considered, and any resulting steering vectors also added to the sum. PAA ensures that critical safety behaviors is prioritized when necessary but does not incorporate the smoothness of a weighted sum as the steering vectors of additional behaviors will probably be ignored as they may be above the threshold.

2.3.4 Sensory Simulation and World Perspective

For a bird to behave correctly in a flock, it needs to have some sort of sensory input. The real birds have e.g. eyes that they use to gather information about their surroundings. A virtual boid does not have any natural sensors and needs to

have its sensory input simulated. The idea of simulating sensory input is to make available the same information that a real-life animal would have at the end of its perceptual and cognitive process. The simulated birds in Reynolds's experiments had access to a database where they could look up the position, orientation, and heading of nearby boids. It is actually very important that the perspective of a boid be limited to its nearest neighbors, as a global perspective would lead all flock members to simultaneously converge towards the flocks centroid. The vital part of information here being that the emergent behavior we recognize as flocking is actually dependant on a localized view of the world.

2.3.5 Modeling Steering Behavior

The 1987 paper by Reynolds [Reynolds, 1987] introduced three basic steering behaviors, these behaviors were later elaborated on in his 1999 paper [Reynolds, 1999]. In the latter paper, he presented solutions for more life-like movements for autonomous characters in animations. He also presented a basic vehicle model with the following attributes and data types:

Simple Vehicle Model:
- mass scalar
- position vector
- velocity vector
- max_force scalar
- max_speed scalar
- orientation N basis vectors

In the above vehicle model, the mass scalar is the vehicle mass, position is the vehicle position, and velocity is the vehicle velocity. Not so self-explaining is the max_force, which is the maximum torque produced by the engine of the vehicle. max_speed is the maximum speed of the vehicle and orientation is the heading of the vehicle.

The vehicle attributes described above can be used to implement basic vehicle behaviors like *seek*, *flee*, *pursue*, and *evade*. Seek is the behavior that aims to steer a vehicle by radially aligning its velocity towards a target. Vectors are used for the calculations of the correct heading, and the speed to travel at will be obtained from the vehicle. All vehicles will have a preferred speed that will typically be the maximum allowed speed. Reynolds's method of obtaining the desired velocity for the seek behavior is to normalize the vector between the current position and the target position. Equation 2.1, 2.2, and 2.3 shows how this method can be implemented. The steering direction ($\mathbf{v}_{dir}$), can be obtained by subtracting the target position (p_{target}) from the current position ($p_{current}$). Equation 2.1 gives us the direction to steer, but not the desired speed. Equation 2.2 shows how the

desired velocity can be obtained by normalizing the steering vector ($\mathbf{v}_{dir}$) and multiplying by the *max_speed*. Here we assume that *max_speed* is the desired speed, but *max_speed* could be an arbitrarily speed dependant on the application. The resulting velocity vector ($\mathbf{v}_{des}$) holds both the desired direction and speed (vector magnitude). Finally the steering vector ($\mathbf{v}_{steer}$) can be obtained as shown in Equation 2.3 by subtracting the current velocity ($\mathbf{v}_{cur}$) from the desired velocity ($\mathbf{v}_{des}$).

$$\mathbf{v}_{dir} = p_{current} - p_{target} \tag{2.1}$$

$$\mathbf{v}_{des} = \frac{\mathbf{v}_{dir}}{\|\mathbf{v}_{dir}\|} * max\_speed \tag{2.2}$$

$$\mathbf{v}_{steer} = \mathbf{v}_{des} - \mathbf{v}_{cur} \tag{2.3}$$

Figure 2.3 shows how the steering of a seek behavior would look. The current velocity is represented by a green arrow, the desired velocity for seek behavior by a gray arrow, and the steering vector for seek behavior by a blue arrow. The blue seek path shows the path the vehicle would take. Figure 2.3 also shows another steering behavior called flee. The flee behavior is mathematically very similar to seek, but instead of approaching the target it acts to steer the vehicle radially aligned away from the target.

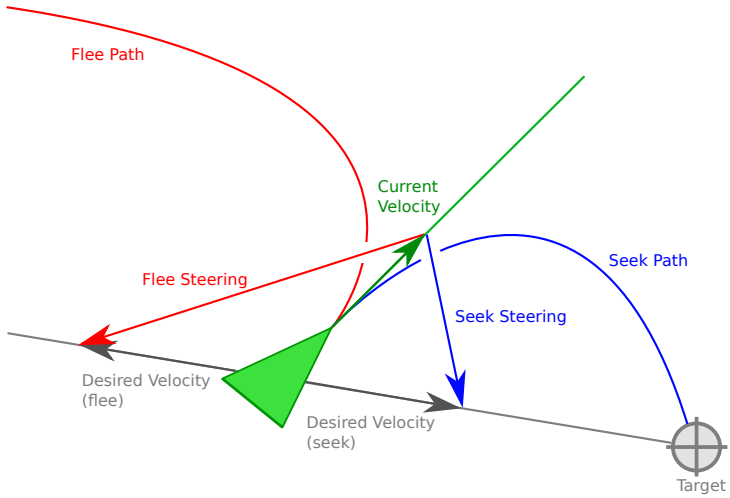


Figure 2.3: *This figure shows the overall idea of the seek and flee steering behavior. (Adopted from: Reynolds [1999])*

A computer simulated boid may have the ability to turn while standing still, this is not the case for real-life vehicles. The cars on the road have to be moving to turn. This element is also reflected in Figure 2.3 where we can see that the seek path (blue) is curving, as opposed to strait.

The computer simulated environment is not continuous as the real world but instead discrete. Updates to the steering vector happen incrementally at intervals, usually multiple times per second. If a vehicle is travelling at 80km/h, then it moves approximately 22.2 m/s ($\frac{80km/h}{3600s/h} \approx 22.2m/s$), hence updating the steering vector more than once per second is reasonable to avoid a jagged steering path.

The steering vectors can be represented with as many dimensions as desirable. However, for our experiment we will be using 2D vectors as the vehicles are moving around in the same plane and hence only two dimensions are needed. If we were modeling fish in water or birds flying, we would need three dimensions.

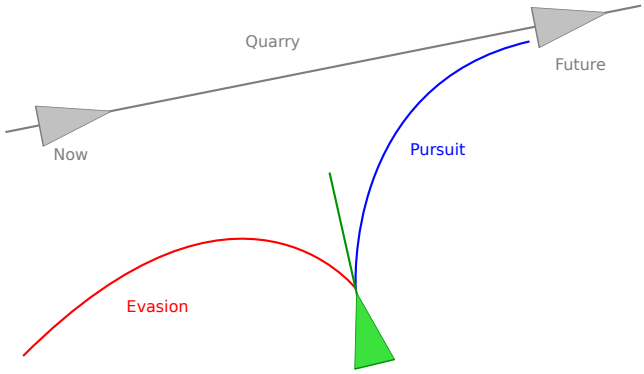


Figure 2.4: *This figure shows the overall idea of the pursue and evasion steering behavior. The key is to predict the future position of a quarry (target). (Adopted from: Reynolds [1999])*

Seek and flee are only two out of multiple different behaviors described by Reynolds in [Reynolds, 1999]. Figure 2.4 shows two more of the basic behaviors: *Pursue* and *evasion*. Pursuit is much like seek, only now the target is dynamic and can change position. The vehicle must predict the future position of the target and adjust its own steering vector accordingly. Since simulations proceed in incremental manners, updates to the steering vectors also need to happen incrementally and hence so must the predictions as well. The key to implementing

a sound pursue behavior can often be to correctly estimate the prediction interval T. Ideally the predictions should happen as often as possible ($T = 0$). However, to ensure reasonable performance it could be an idea to have a larger interval when the vehicle is far from the target and smaller when close.

As flee is the inverse of seek, evasion is the inverse of pursue. Evasion intends to flee from a moving target by using the flee behavior from a predicted target position. The evasion technique described by Reynolds is a lightweight behavior and non-optimal. Optimal behavior in the field of control theory can be found in [Isaacs, 1999]. However, an evasive maneuver is often intentionally non-optimal to foil predictive pursuit strategies [Cliff and Miller, 1996].

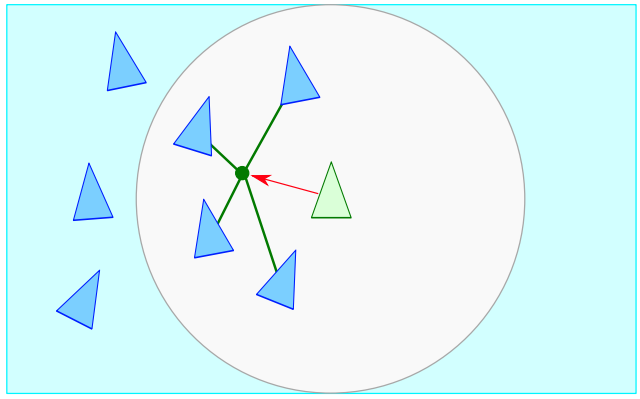


Figure 2.5: *This figure shows the overall idea of the cohesion steering behavior. (Adopted from: Reynolds [1999])*

There are many other behaviors described in [Reynolds, 1999], e.g., *pursuit with offset*, *arrival*, *obstacle avoidance*, and *path following*. We will not describe them all here but return to behaviors such as obstacle avoidance and path following in Section 3.6. There are however two more behaviors that we would like to mention; *cohesion* and *alignment*. Cohesion is the behavior that urges a boid to remain in formation with the rest of the flock. The boid has a simulated field of vision in which the boid can observe the position of other nearby boids. The average position P_{cntr} is then calculated by averaging all the boids observed in the simulated field of vision. By subtracting the boid's position from P_{cntr} it can be used as a basis for the seek behavior. Applying seek behavior only makes sense if P_{cntr} is stationary. Otherwise, we would use P_{cntr} as a basis for pursue behavior. Figure 2.5 shows an example of cohesion behavior. A boid (green) calculates the average position of all the other boids within its field of vision. The desired velocity would then be directed towards the resulting average location (marked by a green dot).

The final behavior that we present here is alignment. Alignment is the behavior that allows a boid to align itself with other boids in a flock. Alignment in this context means both direction and speed. A possibility for achieving alignment is to average the current velocity vector of all the nearby boids within the field of view. This averaging will produce the desired velocity vector for our boid, and then the steering vector will be the difference between the current velocity and the desired velocity. Note that averaging the velocities of the nearby boids may produce unsuitable results if the nearby boids are not sufficiently aligned themselves. Figure 2.6 demonstrates the idea of how a boid (green) will observe the nearby boids (blue) within its field of vision and their velocities. The boid will then calculate the average velocity that will be the boid's desired velocity (the blue line from the green boid). The steering vector will be obtained by subtracting the current velocity from the desired velocity and applied to align the boid with the rest of the flock.

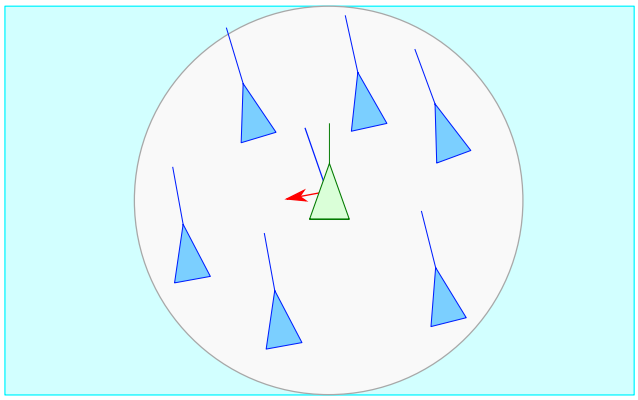


Figure 2.6: *This figure shows the overall idea of the Alignment steering behavior. (Adopted from: Reynolds [1999])*

2.3.6 Combining Behaviors

All the steering behaviors mentioned in Section 2.3.5 can be combined to achieve a more complex behavior. More complex behaviors are typically various utility maximizing behaviors that need multiple steering behaviors to effectively execute a movement strategy. Combinations of behaviors can be done in two ways: the behaviors may be sequentially switched between as circumstances change, or they may be combined in parallel. An example of switching behavior could be an employee running late for a meeting. The employee would like to spend his time watching movies, however while running to the meeting the employee is not going

to stop and watch a movie. Two behaviors that can be combined while running late for a meeting is the target seeking and obstacle avoidance behaviors. The employee needs to run towards the meeting room while avoiding crashing into co-workers, desks, or other obstacles.

2.3.7 Branching of Flocking into Other Domains

The basis for the current flocking algorithms was described by Reynolds in 1987 [Reynolds, 1987]. The idea is that the emergent behavior of a flock is the result of interactions between the behavior of individual flock members. This view of flock behavior is closely related to particle systems. The thoughts of Reynolds have then been built on by multiple authors e.g. [Olfati-Saber, 2006], but applications of flocking have also branched into multiple other domains. We will now describe some of the domains where the ideas of flocking have contributed. Some of the ideas are also presented in the literature review (Appendix B.3).

Robotics

In the robotics domain, the idea of first having layered behaviors was introduced by Brooks in [Brooks, 1986]. Since then Arkin introduced the idea of having multiple behaviors that could sum up to a single action [Arkin, 1987, 1989, 1992]. The work done by Arkin is similar to the work done by Reynolds, but Arkin used artificial potential fields (AFPs, see Appendix B.3.1) instead of procedural approaches. Path planning was added to the steering behaviors in [Raibert and Hodgins, 1991; Hodgins and Wooten, 1998]. Later robots started to gain speed and steering strategies accounting for momentum was introduced [Zapata et al., 1993]. Robots with emergent behavior have been researched by Mataric and fundamental for these emergent properties are the steering behaviors [Mataric, 1995]. Concerning swarm intelligence and cooperative behavior, flocks of robots are heavily researched [Mohan and Ponnambalam, 2009; Jain et al., 2010]. Multiple robots can achieve tasks that a single robot can not, and multiple robots enables another robot to take over in the event of a failure.

Aritifical Intelligence

The emergent behavior of flocking is in itself is a field of research for artificial intelligence, but the steering behaviors of Reynolds have also inspired animations of intelligent characters [Kahn, 1979]. Later Zeltzer popularized the idea of an abstract ”task level” specification of motion [Zeltzer, 1986, 1991]. Then more complex behaviors started to evolve, e.g. Ridsdale developed characters capable of getting from a start position to a goal position while avoiding collisions with static obstacles or other characters [Ridsdale, 1987]. Strassmann developed

handling of properties and emotional portrayal [Strassmann, 1991]. Costa et al. created characters capable of navigating a house while avoiding obstacles [Costa et al., 1990]. Improvisational, dramatic characters which touch on steering behavior have been developed [Bates et al., 1994; Hayes-Roth and Van Gent, 1996a]. The work by Hayes-Roth and Van Gent can be viewed at The Virtual Theater Project [Hayes-Roth and Van Gent, 1996b].

Artifical Life (and others)

The boids model that Reynolds introduced in 1987 was ideal for creating simulations of flocks, herds, and schools [Reynolds, 1987]. The complex emergent behavior was decomposed into steering behaviors at the individual level that enabled implementation of flocks by letting each individual be a separate object. At the 1987 Artificial Life Workshop Resnick presented work on autonomous vehicles implemented in Lego [Resnick, 1987], and Travers demonstrated his AGAR Animal Construction Kit [Travers, 1987]. The next year Reynolds presented steering behavior for obstacle avoidance [Reynolds, 1988]. As mentioned earlier there is a difference between using AFPs to achieve flocking and procedural approaches. Bruderlin and Calvert used a procedural approach to achieve animations of goal-directed human walking [Bruderlin and Calvert, 1989]. Beer created an artificial cockroach that is noteworthy for the depth and complexity of its neuroethological model [Beer, 1990]. The implementation uses tropisms that are direct analogs of steering behavior. The steering behavior of vehicle-like characters was investigated in [Wilhelms and Skinner, 1990]. Simulated sensing is important to allow animated characters to move realistically. Simulated vision was introduced in [Renault et al., 1990]. The simulated vision allowed animated characters to navigate corridors and around obstacles. State-space search has been used to implement controllers for tasks like parallel parking of an autonomous vehicle [van de Panne et al., 1990]. Large human crowds have been modeled using a model of the steering behavior for each individual [Still, 1994]. Karl Sims used a modified genetic algorithm [Mitchell, 1998] to simultaneously evolve brains and bodies for artificial creatures for various types of locomotion and goal seeking [Sims, 1994]. Cliff and Miller co-evolved pursue and evasive behaviors for predator and prey agents [Cliff and Miller, 1996]. A very realistic model of the locomotion, biomechanics, perception, and behavior was created by Tu and Terzopoulos in [Tu and Terzopoulos, 1994; Tu, 1999]. The locomotion and steering behaviors were based on physical attributes of the environment, and the action selection was based on ethiological attributes of the fish. A complex mechanism for action selection was described in [Blumberg, 1994], and in [Blumberg and Galyean, 1995] a virtual reality character, capable of both autonomous improvisation and response to external direction, was discussed. An application of the characters created was the ALIVE system [Maes et al., 1995]. Perlin and Goldberg has made a sys-

tem called Improv, which covers the gamut from locomotion to action selection [Perlin and Goldberg, 1996]. The techniques used are behavioral scripting and Perlin's procedural synthesis of textures applied to motion [Perlin, 1985]. Cremer et al. created autonomous drivers for ambient traffic in interactive automobile driving simulators [Cremer et al., 1996]. The Virtual Fishtank installation at The Computer Museum created by teams from MIT's Media Lab and Nearlife used steering behaviors [Resnick et al., 1998]. Pottinger has developed a system for autonomous characters used in Dungeon Keeper 2 [Pottinger, 1999a] which was inspired by an early draft of [Reynolds, 1999]. He also provided a detailed discussion of steering and coordination for groups of characters in games, as well as details of implementation [Pottinger, 1999b].

2.4 Unreal Engine 4

In this Section, we explain what Unreal Engine is and why we will be using it. We also present some alternatives to Unreal Engine and compare them. Lastly, we present the advantages of using a game engine in the first place as an alternative to e.g. pixel art.

2.4.1 What Is Unreal Engine 4?

Unreal Engine (UE) [Epic Games, 2015a] is a game engine developed by Epic Games [Epic Games, 1991] to create video games more efficiently. Epic games is a US video game development company that is famous for producing games such as Gears of War [Epic Games, 2006] and Unreal [Epic Games, 1998]. The Unreal Engine have been used to develop both of these games, although in an earlier version than Unreal Engine 4. The first Unreal game was released in 1998 and built using the first version of Unreal Engine. Gears of War was released in 2006 and build using Unreal Engine 3. The current newest release as of March 2015 is Unreal Engine 4.

UE is a professional game engine that can be used to create 3D games of all categories. However, due to UE's physics engine it can also be used to build realistic simulations of real-life scenarios, e.g., simulating visual impairment [Lewis et al., 2011] for medical purposes, simulating robots for research or education [Carpin et al., 2007], or surgical training [Marks et al., 2007]. UE can also be used to model a highway with vehicles and buses to simulate and experiment with road traffic. The usage of computer games for research, education, or something *"…more than entertaining"* is called *serious gaming* [SGS, 2014]. The antonym of serious gaming is *leisure gaming*.

2.4.2 Why Did We Choose Unreal Engine?

There are many game engines available, and we had to choose between multiple different products. For an extensive list of game engines see [Mod DB, 2002], here we will only present what we considered to be our best options:

CryEngine [CryEngine, 2015]
CryEngine can either be subscribed to at $9.90 per month or purchased as a commercial license where the company will enter a partnership with CryEngine. CryEngine is a very powerful game engine. However, there does not appear to be a free version.

id Tech 4 [id Software, 2011]
id Tech 4 is a somewhat different game engine than the others presented here. This engine is based on OpenGL [OpenGL, 1992], and is released under the GNU GPL license [Free Software Foundation, 2007; Sneddon, 2011] meaning anyone can access, modify, and distribute the code under the same license. Naturally this means that the engine is open source and freely available. id Tech is now released in a new version (id Tech 5), but it is proprietary and not freely available [Graft, 2010].

Unity 5 [Unity Technologies, 2015b]
Unity is a game engine created by Unity Technologies. The fifth and newest release was released on March 3, 2015. Unity includes scripting with C# (not C++), is available for multiple platforms, extensive documentation, free for personal use, and can import 3D objects created by other programs. Unity is a widely-used game engine with a claimed 45% market share [Unity Technologies, 2015a]. Unity is free for personal use and allows earnings of $100,000 in gross revenue before a purchase of the professional edition is needed. The professional edition can either be bought for $1500 or subscribed for $75 a month.

Unreal Engine 4 [Epic Games, 2015a]
Unreal engine is available without charge for academic purposes. During this project, it was also released free of charge for any project that does not create a $3000 revenue or more per product per quarter. Unreal Engine also has extensive documentation and examples along with open source code on GitHub [GitHub, 2008; Epic Games, 2015b]. It integrates with Visual Studio to allow direct manipulation of C++ code, can import 3D objects created in other programs, and supports Oculus Rift for virtual reality experiences. Unreal Engine also has a visual scripting system that allows for rapid development without the need of C++ code and includes live debugging instead of having to wait for the C++ code to compile. As

a final bonus, UE has many pre-made models that can be downloaded with the engine, e.g. a car model.

Though CryEngine is one of the most powerful game engines, it is not freely available. It was important for us that the software was freely available hence CryEngine was considered not to be an option. id Tech 4 was a viable option, but it does not provide the ease of use or the extensive documentation as both Unity and Unreal Engine do. Hence, the real competition was between Unity and Unreal Engine. We landed on Unreal Engine due to: (1) its superior graphical capabilities [Mayden, 2014], (2) UE has a profiler that lets you determine what parts of your project consumes the most time and resources and hence can be of great help in real-time simulations with many vehicles, (3) UE having better processing performance with many autonomous agents [Méndez, 2014], (4) UE having its own visual scripting language for quick implementation of light functionality, and (5) previous experience using C++ which is the other programming language used in UE.

There are other options than using a game engine for simulations in the first place, e.g. creating pixel art [Silva et al., 2013]. Pixel art is probably the most simplistic approach that would require the minimal amount of work. However using a game engine has some advantages over pixel art: (1) realistic looking simulations are much more visually appealing, (2) some content is already premade e.g. vehicles, (3) game engines already have physic engines that manages elements such as gravity, friction, sunlight, and atmosphere, (4) UE also lets you film the simulations from multiple angles to create a movie for distribution.

Chapter 3

Architecture/Model

In this chapter, we will present the architecture used for our system, the model used for the different behaviors, and our vehicle model. We also present reasoning for why the behavior and vehicle models were constructed as they were.

The complete source code used for the experiments is attached and can be viewed online[1]. The source code is described in Appendix A.

3.1 Overview

The goal of this project is to determine if simple flocking, as introduced in the previous chapter, can be used to steer vehicles in road traffic. More specifically, to steer vehicles running both directions on a highway, with an entrance ramp, and multiple vehicle models. To determine if flocking is a viable concept, we will create a simulation of a segment of highway road traffic. Vehicles will be spawned in both ends of this segment, driving in opposite directions.

The simulation will be created using Unreal Engine (UE) which will ensure that the project have a visually high-quality representation of the simulation. UE also have an integrated physics library that are using NVIDIA PhysX [NVIDIA, 2015]. Having PhysX means physics is accelerated on the GPU if the machine running the simulation have a relatively recent NVIDIA GPU. The physics framework also has an integrated vehicle object, which we will be using. Using an already existing vehicle simulation means that we only have to define a few parameters, including width, height, weight and torque curve. The vehicle object is given the desired throttle and steering as an input for each time-step. The engine then calculates the locomotion of the vehicle for each step.

[1]Source code can be viewed at: `https://github.com/HighwayFlocking/HighwayFlocking`

The calculations associated with simulating realistic vehicles are extensive, which means that the machine running the simulation needs a rather high-end graphics card and processor. In addition to simulating vehicles, we need to build a highway segment to simulate the road traffic. We will be spawning and removing vehicles on both sides of the highway, to create similar conditions to a real-world highway system. The size of the road is a matter of optimization, the length of the road segment should be large enough to show whether flocking works. However not too large as that would cause unnecessary load and increase the time needed for running simulations.

The simulation can be thought of as a 2.5D model. Meaning that the simulation have three dimensions, x, y and z. Every object extends in a 3D space. However, the simulation is constrained to the road, which is a surface, and therefore have only two dimensions. Hence, all the calculations are done as if the simulation was two-dimensional, using the road as the plane of the simulation. The road could have been the plane even if it had slopes, but the calculations would be more complicated. However, in our simulation the road is completely flat, and the simulation is therefore constrained to the xy-plane.

3.2 The System Architecture

The system architecture is heavily based on how Unreal Engine is structured. Two languages are used to create applications using Unreal Engine, C++, and Blueprint. C++ is a system language that have existed since the 1980s and is used for many applications. Blueprint is a language created by Unreal. Coding in Blueprint is not done by writing code, but by drag and drop. The benefits of using Blueprint is that the editor is tightly integrated into the Unreal Editor, and it is easy to edit the code quickly for experimenting. The benefit of using C++ is that the resulting code is faster, and the language is more powerful than Blueprint. Luckily both of the languages can be combined in the same Unreal Engine application, a Blueprint class can even inherit from a C++ class (the opposite, however, is not possible). This simulation will therefore use both languages to exploit the benefits of both.

The architecture of Unreal Engine is heavily class-based, using both inheritance, and composition. For example, each actor in the simulation (e.g., the vehicles, or the road itself) is an instance of a class that is a descendant of the Actor class. Each Actor can have some components, which are classes descendant of the Component class. Since we are using Unreal Engine, our architecture will also be heavily based on classes.

The most interesting class hierarchy in our architecture is the Vehicles. Unreal Engine exposes a class called a *WheeledVehicle*, which have all the attributes and physics of a regular vehicle. In our architecture, we define two subclasses of

the WheeledVehicle class, a *FlockingVehicle* and a *DriveableVehicle*. The FlockingVehicle is the ancestor of all the vehicles that are autonomous and defines the flocking algorithm that each one is running. The DriveableVehicle is the ancestor of the vehicles that can be driven directly by the user of the simulation. The FlockingVehicle class consist of two classes, one created using C++, and a child of this class, created using Blueprint. The DriveableVehicle class is pure Blueprint. Both of these classes have a component called the Cam Component, which handles sending out CAM packages to all nearby vehicles. The CAMs ensures that a FlockingVehicle knows about both other FlockingVehicles and vehicles driven by the user. FlockingVehicle and DriveableVehicle have only one layer of classes below them. This bottom-layer of classes consists of classes for each vehicle type in our simulations. The vehicle types are different models of vehicles, like a bus or a car. We will define various types of vehicles to have somewhat different properties on the highway (e.g., emergency vehicles will drive faster). Variation in vehicle attributes is also important to test how the flocking behaves with different vehicle sizes and variations of velocities. We will also change some of the visual properties to give the road traffic a more natural look.

3.2.1 A Layered Behavior Model

The algorithm that governs the movement of each vehicle uses a layered behavior model, which is illustrated in Figure 3.1. This layered model is running in each of the vehicles in the simulation for each timestep. The layered model makes the system easier to reason about, given that each layer have its specified responsibilities.

The highest layer is the flocking layer that handles almost pure flocking, as defined in the previous chapter. The flocking layer's primary responsibility is to output a Desired Velocity for the vehicle. The next layer is the Autonomous Vehicle Layer, or the steering layer. This layer receives the desired velocity from the flocking layer, and uses the velocity in combination with feedback from the current velocity and heading, to give steering input to the locomotion layer. The locomotion layer is not created by us, but are implemented as part of the physics library in Unreal Engine.

3.2.2 Locomotion Layer

We have no control over the Locomotion layer except for some attributes that are exposed to us, such as throttle curve or weight of the vehicle. The limited exposure of the locomotion layer ensures that bugs can not introduce behaviors that gives a better outcome from the simulation, but could not exist in the real world. An example of an unrealistic vehicle behavior could be a vehicle having sideways acceleration without steering to the side. Realistic simulations

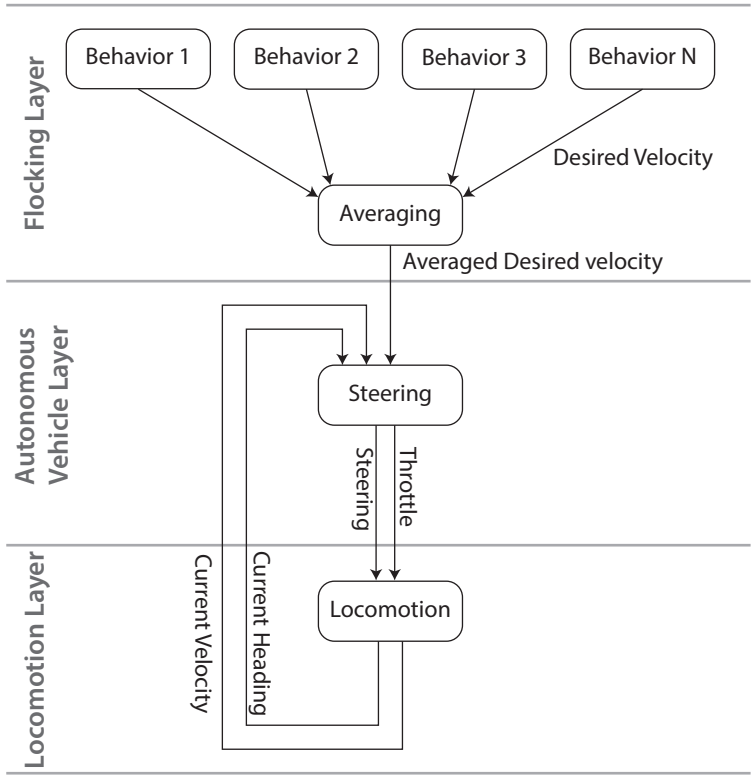


Figure 3.1: *An overview of the layered model that controls a vehicle. The Desired Velocity and the Averaged Desired Velocity are vectors.*

are necessary because we want to create a simulation of the real world, and in the real world a vehicle has certain limitations in its movements. The input to the locomotion layer is the steering and the throttle, the same as what we would have to output if we wanted to control a real-life vehicle. This layer is also the same as what exist in the vehicle controlled by the user of the simulation. However in that case the throttle and steering would have been given by the user, using a keyboard or a game controller.

3.2.3 Autonomous Vehicle Layer

The most critical responsibility of the Autonomous Vehicle Layer is to be a bridge between the flocking layer and the locomotion layer. In the Flocking layer, the world looks easy, and the output can be any velocity. The task of the Autonomous

Vehicle Layer is to find a steering input to the Locomotion layer that will take the velocity of the vehicle in the future closer to the Desired Velocity from the Flocking Layer. This layer could also be used to implement collision avoidance and other security measures in cases where just blindly following the output of the flocking layer could be undesirable.

3.2.4 Flocking Layer

The Flocking layer is adopted from the model given in [Reynolds, 1987] and more refined in [Reynolds, 1999]. This layer consists of a number of behaviors, where each one is as simple as possible and independent of the other behaviors. The goal is to have many simple behaviors, rather than a few advanced. Each behavior outputs a vector that gives the desired velocity in the direction the behavior wants the vehicle to head. The desired velocities from all behaviors are combined by averaging them, using an average function. The combined desired velocity is then output as the desired velocity from the flocking layer.

After some experimenting, we have chosen to use a weighted sum as the averaging function. Reynolds proposes in [Reynolds, 1987] to use a more complex averaging function, called *prioritized acceleration allocation*. However, in [Reynolds, 1999] he changed his mind and stated that a simple linear combination is often sufficient. He also proposes using priorities for the behaviors.

We have tried three different averaging schemes. We started by using a weighted average, where each vector greater than zero is multiplied by some weight; the vectors are summed, and the result is divided by the sum of the weights. However, this gave poor results when combining some types of behaviors. For example, one behavior can output a small vector, meaning that it is less important. However, this can have a too great an influence on the averaging function since the dividing by the weights does not care about the size of the vectors.

Equation 3.1 shows the averaging function we decided on. The function multiplies each vector ($\mathbf{v}_b$) with a weight (w_b) and sums the result. B is the set of all behaviors. The resulting vector is clamped to a length of 1.0, and multiplied with the max speed in km/h to get the desired ($\mathbf{v}_{des}$) velocity in km/h.

$$\mathbf{v}_{des} = \sum_{b \in B} \mathbf{v}_b * w_b \tag{3.1}$$

We have also experimented with adding priorities to this scheme. The current implementation contains a priority scheme, where each behavior contains a priority and a flag. The flag indicates when the behavior is turned on and the priority indicates when behaviors with lower priorities should be ignored. However, for the behaviors used in the results we have not used the priority scheme.

Another scheme that often is used, is to have multiple sets of behaviors. We have used this scheme since the vehicles on the entrance ramp needs a unique set of behaviors to be able to guide themselves along the ramp. When the vehicles pass a virtual gate in the end of the entrance ramp, they change to using the default road behavior set. A third behavior set is used when vehicles are spawned on the entrance ramp, where the only behavior in the set makes them wait for the highway entrance point to be free of prioritized vehicles. The vehicles then switch to the entrance ramp behavior set when the waiting-behavior observes that the entrance point will be free of cars when the vehicles reach it.

3.2.5 Cooperative Awareness Message (CAM) Simulation

To produce simulations as accurate as possible, we will base the communication between the Vehicles in the simulation on how communication is predicted to work in the future real-world. As mentioned in Section 2.1.2, communication between vehicles will be based on CAMs. One of the results of the experiments should be if the CAMs have enough information for guiding automatic vehicles, as stated in Research Question 5, or if additional information is needed.

Note that we will not use the actual ITS communications CAM exchange protocol for the communication, but instead we will exchange C++ objects that have the same information as can be found in CAMs. We will also not always use the same units for the information, for instance, we will use XYZ-coordinates instead of lat/lon/height. Lat/lon/height is what is used to define a point on the earth, but XYZ is what is used to represent a point in the Unreal Engine 3D space. However, even though the representation is different, the information is the same. All the information that we pass back and forth should be possible to convert directly to the CAM protocol, however we do not believe that actually spending computational capacity converting between units in the simulation is worth the extra complexity.

In a real-life scenario, there will be some vehicles that should have higher priority than others, e.g., emergency vehicles. Exactly how the higher priority should be implemented is outside the scope of this thesis, however, note that the CAM standard have a possibility to define the role of a vehicle and the role can be set to one of multiple different types, including *PublicTransport* and *Emergency*. We have decided to create a simple implementation, adding a field called PriorityLevel, where a higher value refers to a higher prioritized vehicle. In our implementation, priority 0 is the default, priority 1 is buses and priority 2 is emergency vehicles.

Listing 3.1 shows the actual C++ struct we will use to pass information in the simulation. The comments refer to which fields in the CAM standard the defined variable corresponds to (see Appendix C for the CAM standard). In listing 3.1

Listing 3.1: *The C++ struct used for CAM in the simulation*

```
typedef struct {
    // ItsPduHeader::StationID
    int32 StationID;
    // BasicContainer::ReferencePosition
    FVector Location;
    // BasicVehicleContainerHighFrequency::Heading
    FVector ForwardVector;
    // BasicVehicleContainerHighFrequency::Speed
    float speed; // cm/s
    // Not in the ETSI standard
    float GoalSpeed;
    //CoopAwarness::GenerationDeltaTime
    float timestamp;
    // BasicVehicleContainerHighFrequency::VehicleWidth
    float Width;
    // BasicVehicleContainerHighFrequency::VehicleLength
    float Length;
    //BasicVehicleContainerLowFrequency::VehicleRole
    uint32 PriorityLevel;
} FCamPacket;
```

we have omitted some fields that can be found in our implementation, but do not need to be transferred over the air in an actual real life implementation. An example of such a field is the spawn time of the vehicle, real vehicles are not spawned and do not have spawn times.

Note that the only field not in the CAM standard is GoalSpeed. GoalSpeed in our implementation means the speed the flocking algorithm would like to move at. GoalSpeed would update faster than the actual speed since the actual speed is delayed by the acceleration or brake power of the vehicle. This means that GoalSpeed is a good indicator of what the speed will be shortly. GoalSpeed is not part of the CAM standard, since the CAM standard is also intended to be usable by human drivers and it is not possible to know how fast a human driver would like to drive (excluding tools like cruise control). However in the future there might be amendments to the standard, adding fields solely used by automatic vehicles.

In the real world, the coordinates passed using CAMs could be determined using the GPS system, which would imply some level of inaccuracy. Observations show that GPS users in 2015 will have an accuracy error of less than 4.7 meters in 95% of the cases [William J. Hughes Technical Center, 2014]. A position within a

circle with radius 4.7 meters is not accurate enough for driving a car. The current automatic vehicles use GPS in combination with, e.g., cameras, lasers, or radar to determine the exact position of the car. However, techniques to improve on GPS accuracy is currently under development, e.g., the high accuracy nationwide differential GPS system (HA-NDGPS) will be able to provide locations with an error less than 10 centimeters with a 95% confidence [ARINC Incorporated, 2008]. In our experiments the vehicle objects have access to their exact x, y, and z coordinates, and we will not implement simulation of GPS or other positioning tools, but rather assume that the vehicles are using a form of future positioning tool in accordance with Assumption 3.

3.3 The Road Model

Since UE is a 3D game engine, we will need to create the asset that represents the road. Unreal Engine has a spline tool, which we are using to represent the path of the road. A 3D mesh gives the texture and the road markings. The road will then be extruded using the mesh and the path. We have chosen to model the road curvature after a 500 meter long path of E18 west of Oslo, to get a relatively realistic road.

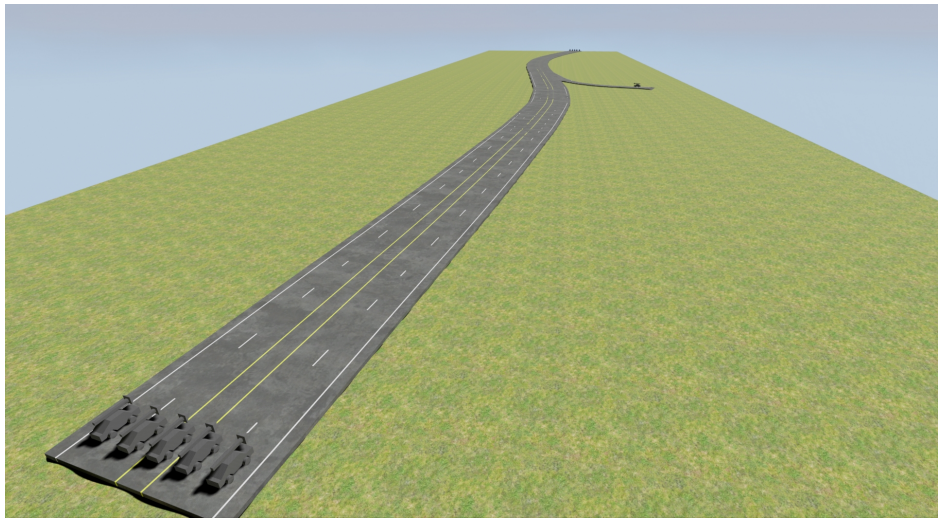


Figure 3.2: *The road model.*

On the road mesh, 3D vehicle objects will be driving. The mesh and vehicle objects are the visual representation used to generate the image the user will

be watching and to produce the physics that will be utilized in the simulation. However, the vehicles will also use the path for their representation of the world. The vehicles will have a roadmap, generated from the path that is used to produce the physical road. The roadmap consists of two parts, a path that represent the middle of the road, and a road radius. Combining path and radius, will give a 2D roadmap that the vehicles can query about where the road is heading.

For the width of the road, we will be using the road widths from NPRA standards, using four lanes. The lanes are 3.5 meter across, with 3 meter in the middle, between the two directions. The road shoulders are 1.5 meter. That gives a total width of 20 meter, see Figure 3.2. We will be creating a texture showing the road markings as it would be in a real-world road, however, this texture will mostly not be active. The textures can be toggled on or off and can be seen in Figure 3.2. Since we are using flocking, the vehicles do not know about the lanes. Hence, the road markings are only a visual feature for the users that are watching the simulation, and not the vehicles in the simulation. The lanes will be ignored by the vehicles and hence not showing the markings may make the simulation more trustworthy.

3.4 Implementation of the Locomotion Layer

The vehicle implementation in Unreal Engine is a wrapper around the vehicle implementation in NVIDIA PhysX. We need to create the gearing behavior as part of our implementation, since we have found the auto gear implementation in PhysX too simple (it appears unstable for constant velocity).

The PhysX vehicle implementation simulates a motor with gearbox and clutch, and even the suspensions of the vehicle. It is created to give a realistic vehicle experience for the user of games with vehicles. We do not need a vehicle model this complex, however, the extra vehicle complexity does not introduce intolerable amounts of extra load.

When creating a vehicle class in Unreal Engine, multiple parameters can be tuned. These parameters include, but are not limited to:

- Wheel radius and width
- Wheel mass
- Wheel damping rate
- Braking behavior
- Steer angle (including which wheels can steer)
- Tire friction

- Tire stiffness
- Max brake torque
- Vehicle Mass
- Drag Coefficient
- Chassis width and height (used to calculate drag force)
- Torque curve (torque for different motor RPM)
- Engine Setup (things like Max RPM, moment of inertia, damping of different parts)
- Differential setup (front or rear drive, or both, how the force is split between the wheels)
- Transmission setup (Gear ratios and number of gear)
- Steering curve

The parameters will need some tweaking to determine what works for us. Some parameters have more impact on the simulations than others, e.g., the torque parameters which determines how fast a vehicle can accelerate. Acceleration is necessary to avoid collisions and thus needs to be modeled carefully, however, we can make it less complicate by having a flat torque curve (resembling an electric car). We would like the parameters to be as similar as possible to a real-world car, but it is not necessary to have the torque curve perfectly resembling a real world car.

3.5 The Vehicles

The vehicles are critical assets to our experiments and in this section we will explain how they are modeled, what characterizes each vehicle, and what purpose the vehicle serves. UE exposes the parameters of the PhysX vehicles for tuning. Consequently all of our vehicles are instances of the PhysX vehicle, but tuned differently, e.g., a bus is large with a high torque output, and a personal car is small with lower torque output.

3.5.1 The Personal Vehicles

All our vehicles are modeled to resemble real-world vehicles, i.e., the dimensions, torque curves, and visual appearance is what one would typically expect from a regular vehicle. The *sedan* car, see Figure 3.3, comes with UE and is pre-modeled and ready for usage. Our other personal car is modeled to resemble a Volkswagen Golf, see Figure 3.4. We chose Golf as it is the most sold vehicle in Norway from January to June 2014 [Larsen-Vonstett, 2014] and the most common car in Norway in general [Moberg, 2014].

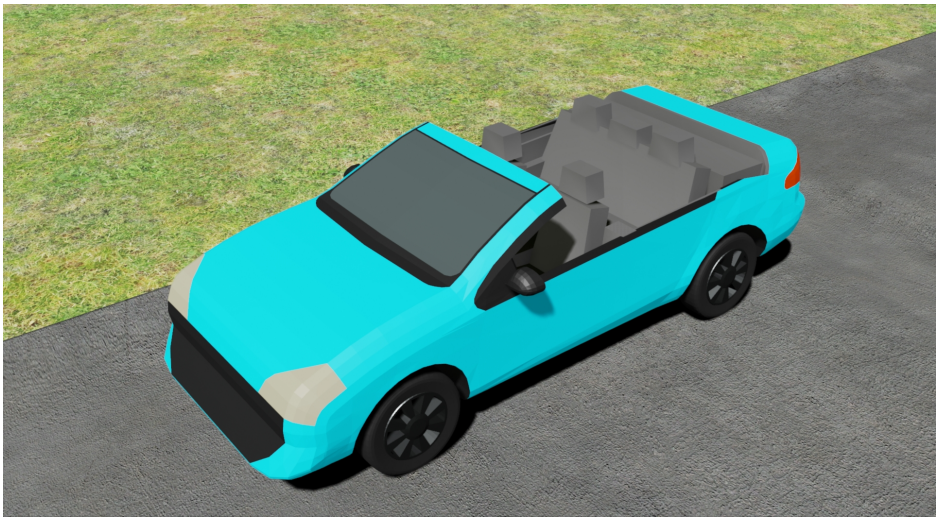


Figure 3.3: *An image illustrating a blue sedan vehicle model. The sedan is already implemented in Unreal engine as an example vehicle and based on the NVIDIA PhysX vehicle.*

Both the sedan and golf is modeled to weigh 1500kg, be 180cm wide, 140cm high, and 452cm long. The torque curves of both vehicles are very flat, resembling an electric vehicle, starting at 400Nm with a slight peak at 1800 rpm with 500Nm, and then decreasing to 400Nm again at 5700rpm. The torque is powerful enough to enable the vehicles to do 0 to 100km/h in about 7 seconds.

The desired speed of a personal vehicle is set to be 90% of 70km/h, i.e. 63km/h. The reason for setting the speed to 90% of something, as opposed to simply setting the desired speed to 63km/h, is that it creates a 10% buffer that the vehicle can use for avoidance purposes. In a scenario where a vehicle v_i is on the road and blocking a vehicle v_j from entering the road from an entrance ramp, v_i would be able to increase its speed to allow v_j to enter. See Appendix

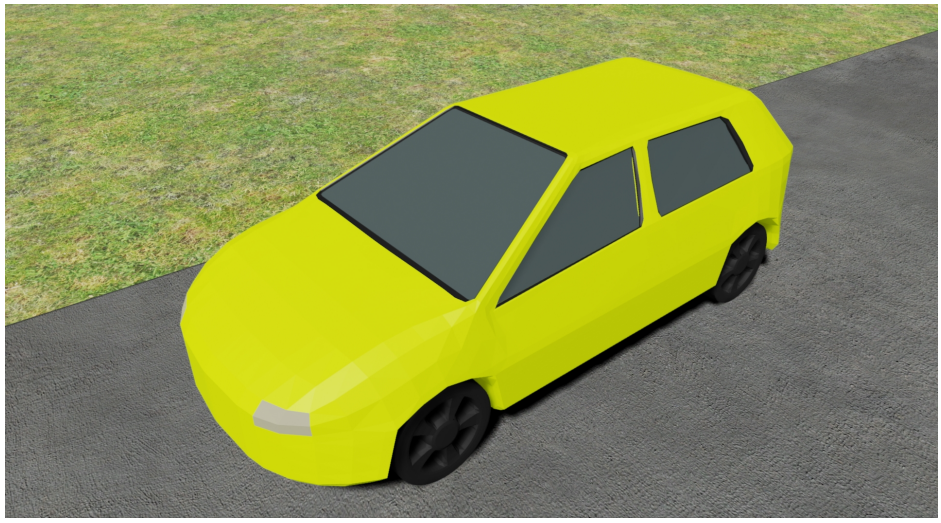


Figure 3.4: *An image illustrating one of our yellow Volkswagen Golf vehicles. This vehicle is built upon the NVIDIA PhysX vehicle and has the same physical attributes,e.g., height, torque, width, as the sedan vehicles.*

D for an illustration of this scenario. The reason for setting the desired speed of a personal vehicle to 63 km/h is that the stable state CO2 emissions are at the lowest between 48km/h and 64km/h [Barth and Boriboonsomsin, 2008]. We would like the vehicles to arrive as fast as possible to save time for the commuters, but we still not produce any more CO2 emissions than necessary. We could have set the speed of the personal vehicles to 64km/h, but there are some variations in the velocity of the PhysX vehicle hence the need for a PID controller to manage the velocity (see Section 3.5.4). Because of the variation in velocity from the PhysX vehicle the speed of our personal car may deviate from 63km/h with a few km/h. We rather want our vehicles to vary speed within the minimum-emission interval than outside of it, and hence chose to have a 1km/h margin.

The sedan and golf vehicles come in multiple colors, the color is irrelevant for the vehicle performance and is purely a visual feature for the simulation audience. There is, however, one color that has a special meaning; the red color. The red vehicles are emergency vehicles (see Section 3.5.3).

3.5.2 The Bus

The bus is modeled after a Volvo 8700 [Volvo Group, 2011] and is 247cm wide, 261cm high, and 12,52m long. Since we are using the NVIDIA PhysX vehicle to

model the locomotion of our vehicles we have to tune the PhysX vehicle properties to resemble a bus. The physical properties of a bus engine proved difficult to acquire and hence the bus engine is modeled as a heavy car with more torque. The weight is 5000kg, and the torque starts at 1600Nm, peaks at 1890rpm with 1700Nm and then decreases to 1600Nm at 5729rpm. The torque curve is unrealistic, a real-life bus will only have a small rpm range with max torque. However, we have found a torque curve and weight that works for our usage. The purpose of tuning the torque curve is to achieve vehicles with different acceleration characteristics. The bus can do 0km/h to 100km/h in about 26 seconds. Even though the bus is modeled significantly lighter than a normal bus (a real-world Volvo 8700 weighs between 18 and 24 tons), the weight only matters in accelerations and collisions. The acceleration is tuned to be lower than the acceleration for the cars. In terms of collisions, we are trying to model a steady-state road traffic scenario with as few collisions as possible. Thus, the weight is not essential for creating realistic simulations. If a collision still happens we will log it and then remove the vehicles from the simulation, see Section 4.2.

Figure 3.5: *The bus model used in the simulation. The buses are larger, heavier, and have slower acceleration than the cars and the sedans.*

The bus is set to drive at 90% of 80km/h, i.e. 72km/h. We have chosen to have the bus drive faster than the personal vehicles in to demonstrate how a public transport vehicle could be prioritized over a regular vehicle, see Research Question 3. The buses will be driving faster than the ordinary traffic as it, in a future road traffic system, could encourage people to take advantage of the bus

rather than drive their personal vehicles.

There is one particular steering behavior than only applies to the bus and no other vehicle; the *Keep Right Behavior* (KR). Buses have to stop to let people on and off, hence they will often have to approach the side of the road as we assume people will be waiting for the bus on the *side* of the road and not *in* the road. We have further assumed that it would be a good idea to have buses stop on the right side of the road as we have defined that yielding to oncoming road traffic should happen to the right (see Section 3.6.5). Buses stopping on the right would ensure that we get two major flows of traffic, one in each direction, and both of them on their respective right-hand side of the road. If buses were to stop on the left side of the road, in driving direction, then we would potentially have four directional flows of road traffic: from the left it would be the buses, and then the oncoming vehicles, and than the vehicles going the same direction as the left-side bus, and then the buses going the same direction as the oncoming vehicles. Having multiple directional flows of vehicles is possible, but increases complexity unnecessarily. The major point is to have buses drive on the same side of the road that vehicles move to when yielding to oncoming traffic.

3.5.3 The Emergency Vehicle

We have included an emergency vehicle in our simulations; it can be recognized as being the only red vehicle and is shown in Figure 3.6. The reason for creating an emergency vehicle is to make an attempt at answering Research Question 3. The vehicle model is much the same as the Volkswagen Golf model used for the personal vehicles with a few changes. (1) The emergency vehicle is the only red vehicle, the color has no impact on the performance of the vehicle but is purely a visual feature. (2) The emergency vehicle's default speed is 90% of 100km/h, i.e., 90 km/h, as opposed to the personal car's 63km/h. The reason for having the emergency vehicle drive significantly faster than regular traffic is that an emergency vehicle is considered more important than ordinary vehicles and should hence be prioritized over regular vehicles.

While the buses will always try to keep to the right, the emergency vehicles have no such limitations and only seek to maintain a max speed at 90km/h.

3.5.4 PID controller for self correction

Keeping an exact velocity can be challenging. The locomotion layer with the PhysX vehicle has some variation in its speed and steering, hence the steering behaviors can not simply tell the vehicle to drive at e.g. 60km/h. A number between 0 and 1 has to be given as an input parameter for how much throttle should be applied. The system can be compared with a human driver, driving a regular real-world car. The driver can not tell the vehicle to drive at 60km/h, but

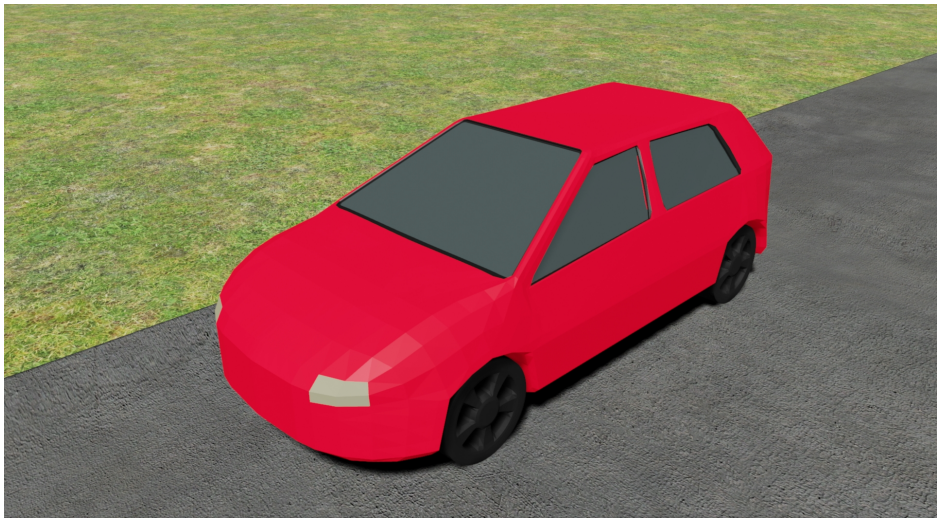


Figure 3.6: *An emergency vehicle. The vehicle model is the same as the one used for personal vehicles, but the default speed is 90km/h as opposed to 63km/h. The emergency vehicle can easily be recognized by its red color.*

will instead push the gas pedal down as much as the driver estimates is necessary. The driver will then keep adjusting the pressure on the gas pedal till the desired speed is achieved. In our simulations the *proportional-integral-derivative* (PID) controller [Bennett, 1984] can be compared with a driver. The PID controller will accept a desired speed as input and output a decimal number between 0 and 1 reflecting how far down the PID controller wants to push the gas pedal. The PID controller will then observe the resulting speed of the vehicle and adjust the throttle output, observe the new speed and adjust the throttle again. The PID controller will keep iterating and always try to adjust the speed of the vehicle to the desired speed. We use two PID controllers in our implementation of the *steering* mechanism in the *Autonomous Vehicle Layer*, see Figure 3.1. The first PID controller is used for throttle and the second is used for controlling the steering wheel of the vehicle.

A PID controller is composed of three major parts: a proportional part, a derivative part, and an integral part. The proportional part looks at the current error in speed. If we are driving at 50km/h and want to be driving 60km/h, then the current error is 60-50=10km/h. The derivative part considers the rate of change in the error. If only the proportional part is used the car will increase its speed at a constant rate till 60 is reached and then stop accelerating. However, the constant acceleration up till 60 will probably cause the car to overshoot 60

by a few km/h, and will then have to use constant acceleration to brake and will again miss 60km/h by a few km/h. The PID controller will keep iterating and will eventually reach the desired speed, however, we will be simulating a very dynamic road traffic environment and need more precise throttle control to avoid collisions. The derivative part will reduce the acceleration as the error in speed decreases and hence help not overshooting the target speed. The Integral part accounts for how the error has evolved in the past. The error may have settled out as too low or too high and the acceleration to reach the target speed may need to be higher or lower than what the proportional and derivative parts would indicate.

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, \mathrm{d}\tau + K_d\, \frac{\mathrm{d}}{\mathrm{d}t}\, e(t) \tag{3.2}$$

Equation 3.2 shows how a PID controller is composed, $u(t)$ is the output that in our case will be a number reflecting how much throttle should be given. K_p is a tuning parameter that reflects how important the proportional part is for the result, K_i is the tuning parameter for the integral part, and K_d is the tuning parameter for the derivative part. The purpose of the tuning parameters is to adjust how much one of the parts should be weighted compared to the other parts. The integral part could prove inaccurate and be weighted lower than the two other parts to achieve a more precise result. $\int_0^t e(\tau)\, \mathrm{d}\tau$ is the integral part that accounts for past magnitude and duration of the error. $\frac{\mathrm{d}}{\mathrm{d}t}\, e(t)$ is the derivative part that accounts for the rate of change in the error.

In our implementation, we have added a limit on the total output and the integral, keeping it from becoming too large. If the error is large for an extended period, e.g. in an accelerating car, the integral part can grow large enough to overshoot the target speed. The limit on the integral is given by the limit on the output. The integral can not grow more than what is needed to reach the limit of the output of the equation, i.e., when the integral is large enough to reach the output limit by itself.

3.6 The Steering Behaviors

The final behavior of the vehicle is composed out of multiple different behaviors, each focusing on an important attribute of road driving. We will now describe the different behaviors we have implemented, why they are necessary, what they do, and how they do it.

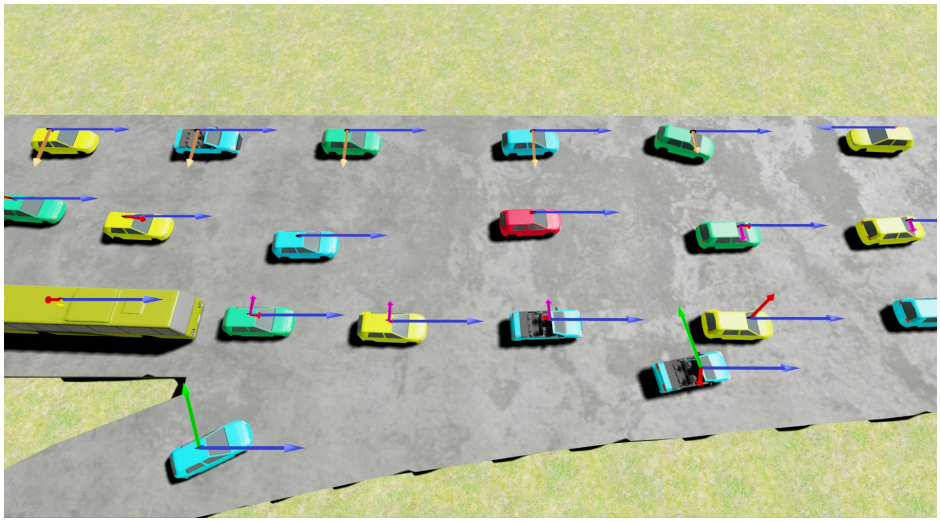


Figure 3.7: *Multiple vehicles with their steering behaviors indicated by colored arrows. The final behavior of a vehicle will be a sum of the individual steering behaviors.*

3.6.1 What is a Steering Behavior

One steering behavior can be thought of as representing a single desire of an entity. The entity can be, e.g., a bird, fish, or a vehicle. When using steering behaviors the final behavior of the entity will often be composed of multiple individual steering behaviors that each account for some specific situation where steering is needed. E.g., for a bird the *avoid collision* behavior will ensure that birds do not collide, and the *cohesion* behavior will ensure that birds flock together. The resulting behavior will be birds who flock together while avoiding collisions. For vehicles, we need steering behaviors to account for the many scenarios that may arise during road traffic, e.g., for avoiding collisions, yielding to prioritized vehicles, or merging with oncoming traffic. When a vehicle is determining the desired speed and heading, each steering behavior can be thought of as an individual voice telling the vehicle where it should steer and with what speed. The vehicle will then make a decision for speed and heading based on all of the individual steering behaviors. The steering behaviors may not agree on a velocity (heading and speed). In fact, the steering behaviors may voice opposite desires for velocity, and it will be up to the vehicle to decide the final velocity. To account for disagreements between behaviors, a weighting scheme is applied. The most important behaviors will have the highest weight, and less important behaviors will have a lower weight.

There are multiple example implementations of steering behaviors available online, e.g., [Shiffman, 2012a,b], which provides detailed explanations and illustrations of steering behaviors for many scenarios.

3.6.2 Road Tangent

Road Tangent is the behavior that ensures that the car is following the road. The behavior computes a vector that will point in the direction of the road and have the length corresponding to 90 % of the maximum speed, where the maximum speed is different for the different vehicle classes.

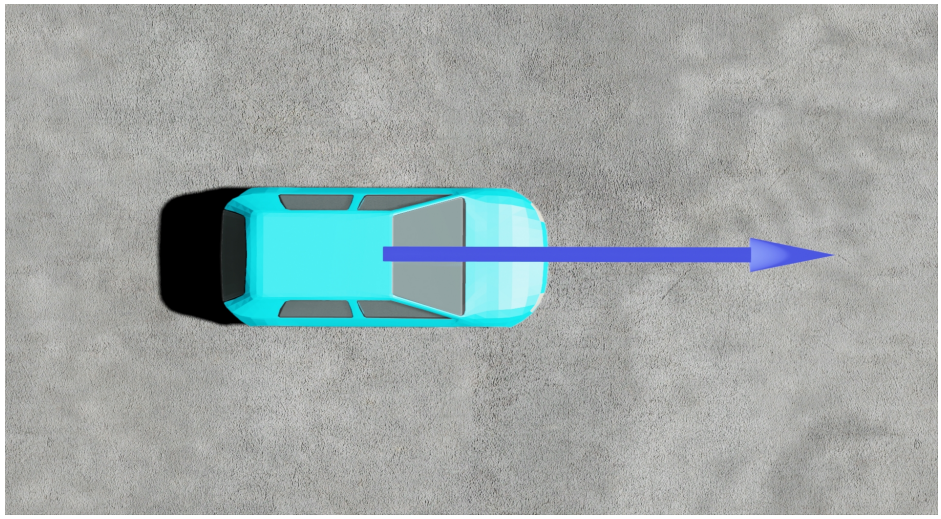


Figure 3.8: *This figure illustrates the steering vector generated from the Road Tangent Behavior. This behavior urges the vehicle to follow the road. The steering vector is here displayed as a blue arrow pointing forward from the center of the car. The direction of the arrow indicates the desired driving direction, and the length indicates the desired speed.*

The Road Tangent Behavior is one of the most fundamental behaviors that ensures that the car is moving forward. Figure 3.8 shows an illustration of a vehicle and its Road Tangent vector pointing in the road direction. In real-world scenarios, this behavior could be generated by vehicles having sensors that could determine where the road is, by GPS, or by roadside infrastructure transmitting messages to the vehicles. In Unreal Engine, the road follows a path that can provide us with a tangent vector to the road direction. This vector is used as the desired velocity vector ($\mathbf{v}_{des}$) for the vehicle.

Since we will have road traffic in both directions, a check is necessary to ensure that $\mathbf{v}_{des}$ is pointing in the correct direction. If the $\mathbf{v}_{des}$ were used directly for all vehicles, all vehicles would be driving the same direction. Every vehicle in UE has a vector that points in the vehicle's forward direction. This vector is referred to as the vehicle's *ForwardVector*. If the dot product between the ForwardVector and $\mathbf{v}_{des}$ is less than zero, the $\mathbf{v}_{des}$ can be inverted by multiplying by -1 to achieve the correct direction.

Road Tangent is inspired by the *flow field following* steering behavior in [Reynolds, 1999]. Flow field following steering behavior directs the motion of characters based on their position in an environment. Flow field following does not require beforehand programming to specify what the character will do at all times. Instead, the character will steer to align its motion with a local tangent of a flow field. (*Flow fields* are also often called *force fields* or *vector fields*).

3.6.3 Avoid

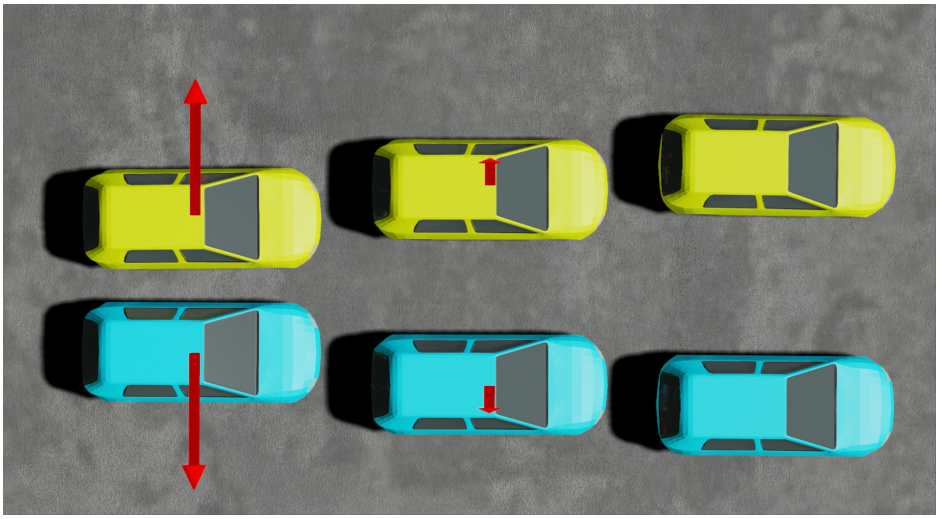


Figure 3.9: *The Avoid Behavior urges vehicles to keep a minimum distance between themselves. If two vehicles come too close, the Avoid Behavior will generate a steering vector that points directly away from the other vehicle. The generated steering behavior is here illustrated as a red arrow starting from the vehicle center and pointing away from the other vehicle.*

Avoid is the behavior aimed at preventing vehicles from crashing. This behavior ensures that vehicles will try to stay at least a minimum distance away

from each other. The behavior has a *start distance* and a *min distance*, with *min distance* being smaller than *start distance*. The range between *start distance* and *min distance* is used to determine the force of the Avoid Behavior. If two vehicles are closer than *min distance* the Avoid Behavior will urge them to steer away from each other with maximum force. If the two vehicles are barely closer than *start distance* the Avoid Behavior will urge the vehicles to steer away from each other with a much smaller force. The magnitude of the steering vector from the Avoid Behavior is, in other words, inversely proportional to the distance between the vehicles. Figure 3.9 shows an illustration of the steering vector generated from the Avoid Behavior.

The Avoid Behavior is similar to the *collision avoidance* steering behavior in [Reynolds, 1987]. The idea of collision avoidance is to prevent individual birds from crashing into flockmates.

A vehicle v_i could have multiple other vehicles that are too close, Figure 3.10 shows an example of this scenario. v_i keeps track of the other vehicles by listening to CAMs. Each vehicle maintains a list of the most recently received CAM from each vehicle within range. To determine whether v_i needs to keep a larger distance to any other vehicles, the Avoid Behavior calculates the estimated current position to all the vehicles within CAM range. Equation 3.3 shows how the position of a neighboring vehicle v_j is calculated. $\mathbf{p}_j$ is the resulting position estimate of a neighboring vehicle v_j, $\mathbf{p}_{CAM}$ is the most up-to-date position received in CAMs from vehicle v_j. $\mathbf{v}_f$ is a vector that points in the direction the neighbor is driving (the neighbor's ForwardVector), *speed* is the neighbor speed, and Δ_t is the time since the CAM was received.

$$\mathbf{p}_j = \mathbf{p}_{CAM} + (\mathbf{v}_f * \text{speed} * \Delta_t) \tag{3.3}$$

$\mathbf{p}_j$ now holds the position of the neighbor vehicle v_j if it kept traveling with the speed and direction it had when it transmitted the CAM. The estimate is very dependent on the Δ_t, as a car traveling at 80 km/h will move slightly more than 22m per second. Hence, a delta of one second is very much. To keep Δ_t low the CAMs are transmitted multiple times per second.

To determine whether a vehicle v_i and a vehicle v_j is too close, the distance between them is calculated. This is done by treating each vehicle as a rectangle and calculating the distances between the two rectangles. This ensures that the distance between the vehicle is the same if the corners are 5 cm from each other, as if the doors are 5 cm from each other. d is the resulting distance between the two vehicles.

The size of d will control the magnitude of the avoid vector produced. However, the direction is controlled by the vector pointing from v_j to v_i. Equation 3.4 shows how the direction is calculated. $\mathbf{p}_i$ is the position of v_i and $\mathbf{p}_j$ is the

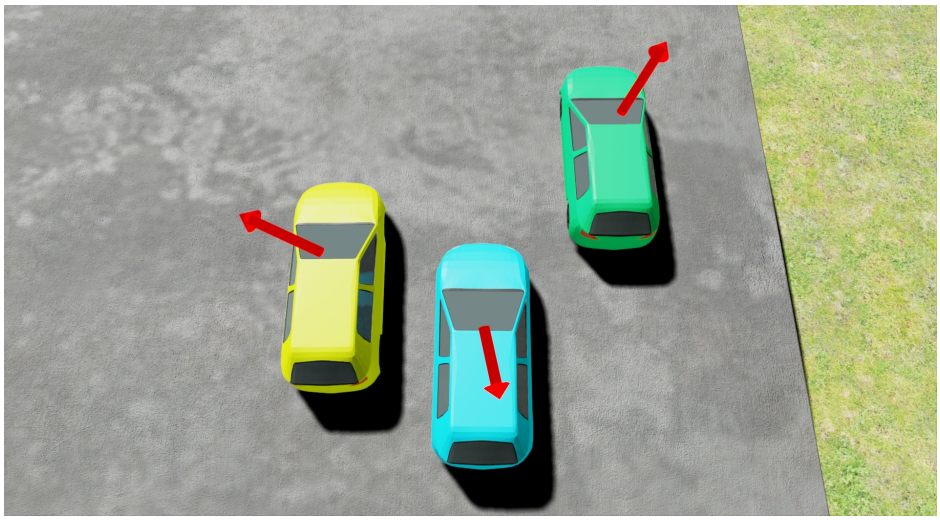


Figure 3.10: *A vehicle could have multiple other vehicles that are too close. An avoidance vector will be computed for each vehicle that are too close and summed up to form the final avoidance steering vector.*

position of v_j. $\mathbf{d}_{ij}$ is the vector pointing from v_j to v_i.

$$\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i \tag{3.4}$$

If d is less than *start distance* the Avoid Behavior will urge v_i to move away from v_j. Note here that both v_i and v_j will perform corresponding calculations. Hence if v_i determines that v_j is too close, v_j will determine that v_i is too close to v_j as well. The Avoid Behavior will then create a steering vector urging v_j to move away from v_i. $\mathbf{d}_{ij}$ is used to determine the direction to avoid v_j and d is used to determine the desired speed used to avoid. If there are multiple vehicles that are too close, v_i will produce an avoidance vector for each vehicle and sum them all into one final avoidance steering vector. The final vector is then divided by the number of vectors to make certain the magnitude remains within limits to ensure the car does not drive beyond the speed limit.

3.6.4 Keep Inside Road

Keep Inside Road (KIR) is the behavior that ensures that the car remains on the road. Road Tangent ensures the car follows the road in the correct direction but does not dictate how far away from the road the car may be allowed to travel. KIR creates a steering vector that points away from the edge of the road and

keeps the car at a safe distance from the edge. KIR is similar to the *path following* steering behavior in [Reynolds, 1999], where the actor inspects a predicted future position and determines whether the position is inside the path or not. Figure 3.11 illustrates the KIR Behavior.

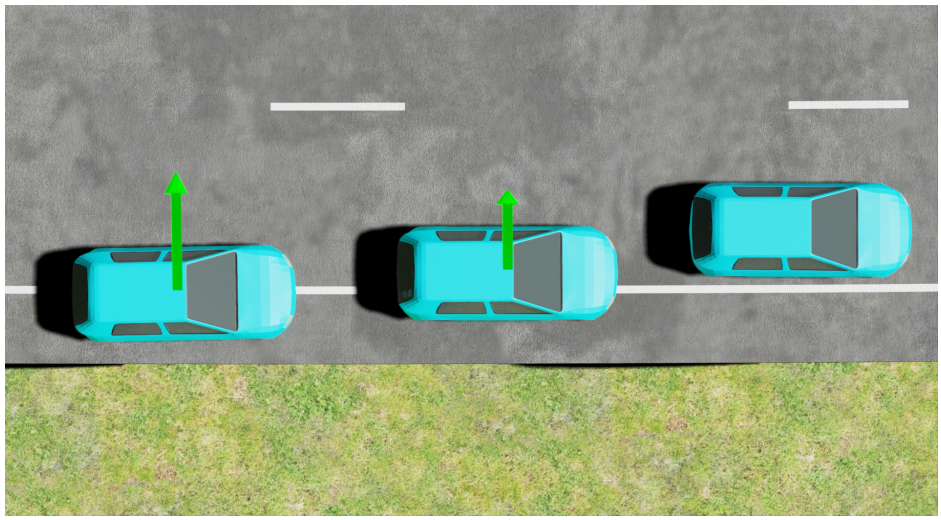


Figure 3.11: *The KIR Behavior ensures that a vehicle remains on the road. The behavior is dynamical in the sense that it increases in strength as a vehicle approaches the edge of the road. In this figure, the green arrow illustrates the KIR Behavior as it urges the blue vehicles to steer to the left to avoid the edge of the road. We can see that the green arrow decreases in magnitude as the vehicle increases its distance to the edge.*

Specifically the KIR Behavior works by looking ahead a configurable number of seconds and determines if the vehicle would still be on the road with its current heading and speed. More specifically the vehicles determine if a point, at a certain distance, in front of the vehicle is within the acceptable bounds of the road or not. Equation 3.5 shows how this point is calculated for a vehicle v_i. $\mathbf{p}_i$ is the position of vehicle v_i, $\mathbf{v}_f$ is the normalized direction of v_i (the ForwardVector), *speed* is the speed of v_i, and t_l is the number of seconds we look ahead to determine if the vehicle would still be inside the road. $\mathbf{p}_f$ is the resulting predicted future position that will be checked for being on the road or not.

$$\mathbf{p}_f = \mathbf{p}_i + (\mathbf{v}_f * \text{speed} * t_l) \tag{3.5}$$

A high t_l would cause the vehicle to drive smoothly since it is looking far ahead and performs no sharp turns. However, a large t_l would prevent the vehicle from utilizing the full capacity of the road since it can not turn too much. Turning too

much would cause $\mathbf{p}_f$ to be off the road. Sometimes sharper turns are necessary to perform evasive maneuvers. Figure 3.12 illustrates the difference between having a low or high t_l. If the vehicle in Figure 3.12 were to avoid an obstacle, a sharp turn might be necessary. A turn as sharp as the one in figure 3.12 would not be possible with a high t_l as $\mathbf{p}_f$ then would be outside the road, as illustrated by the point $\mathbf{p1}_f$. $\mathbf{p2}_f$ illustrates $\mathbf{p}_f$ where the t_l is low, in this scenario $\mathbf{p}_f$ is inside of the road and the maneuver is possible.

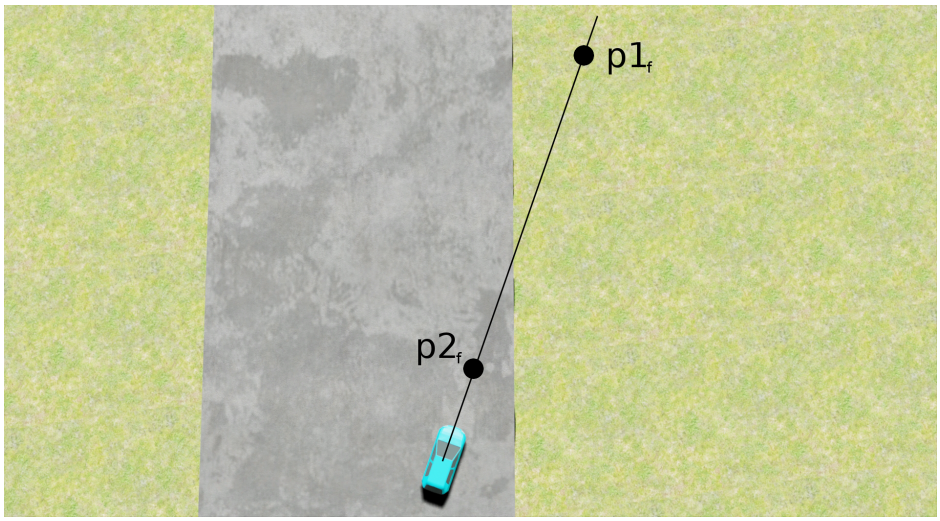


Figure 3.12: *This figure illustrates the concept of looking ahead a certain distance to determine if a vehicle will be on the road in the near future. Looking far ahead, as illustrated by* $\mathbf{p1}_f$*, will cause the vehicle to determine that it will be outside of the road, and hence such a sharp turn would not be possible. On the other hand, if the vehicle only looks ahead a short distance, as illustrated by* $\mathbf{p2}_f$*, the sharp turn would be possible. However, only looking ahead a short distance causes jagged driving, and hence how far to look ahead is a trade-off between stable driving and maneuverability.*

Ideally we want $\mathbf{p}_f$ to be as close to the vehicle as possible as it allows for the greatest maneuverability. However, having $\mathbf{p}_f$ very close to the vehicle causes jagged driving when the vehicle has to avoid other vehicles. Hence, how far ahead to position $\mathbf{p}_f$ is a trade-off between maneuverability and stable driving.

The KIR Behavior comes into play especially when vehicles are entering the road and need to merge with the existing traffic already on the road. Figure 3.13 shows an example of a blue vehicle entering from the right, trying to merge into the existing traffic already on the road. The KIR Behavior urges the blue car to drive to the left, as it is currently outside of the road. However, as the

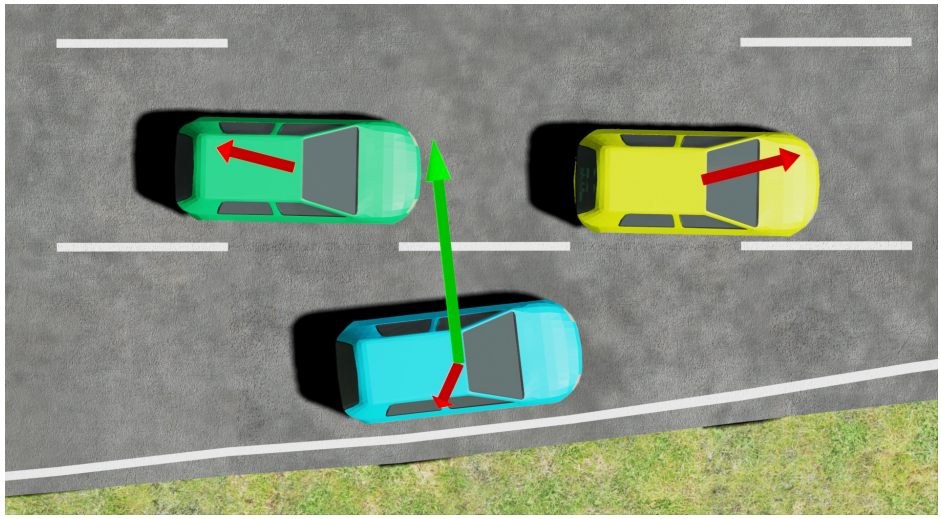


Figure 3.13: *A blue vehicles entering the road. The KIR Behavior (green arrow) urges the blue car to steer onto the road. However, the Avoid Behavior (red arrows) urges the blue car to keep at least a minimum distance towards the existing traffic. The KIR Behavior needs to be weighted higher than Avoid for the blue car to be able to keep approaching the existing traffic and cause the other cars to steer away. The yellow and green car are already on the road and will be urged away, by the Avoid Behavior, from the blue car as it keeps approaching.*

blue car approaches the existing traffic (the yellow and green car), the Avoid Behavior prevents the blue car from driving too close. The Avoid Behavior is illustrated as red arrows. In this scenario, the individual weighting of behaviors become important as the Avoid Behavior would have prevented the blue car from entering the road if Avoid and KIR were weighted equally. KIR needs to be weighted higher than Avoid. Otherwise the blue car would be urged to the left by KIR, but exactly equally urged to the right by Avoid, the result would have been the blue car driving straight ahead and off the road. In Figure 3.13 we can see that the blue car has a green arrow that is larger than the red arrow, which indicates a higher weighting of the KIR Behavior. The higher weighting of KIR causes the blue car almost to ignore the Avoid Behavior and keep driving to the left to enter the road. The existing traffic that are already on the road will be urged by the Avoid Behavior to keep at least a minimum distance from the blue car. When the blue car keeps approaching the existing traffic, the existing traffic will steer away from the blue car and let it enter the road.

Regular road traffic has to yield to emergency vehicles. Emergency vehicles

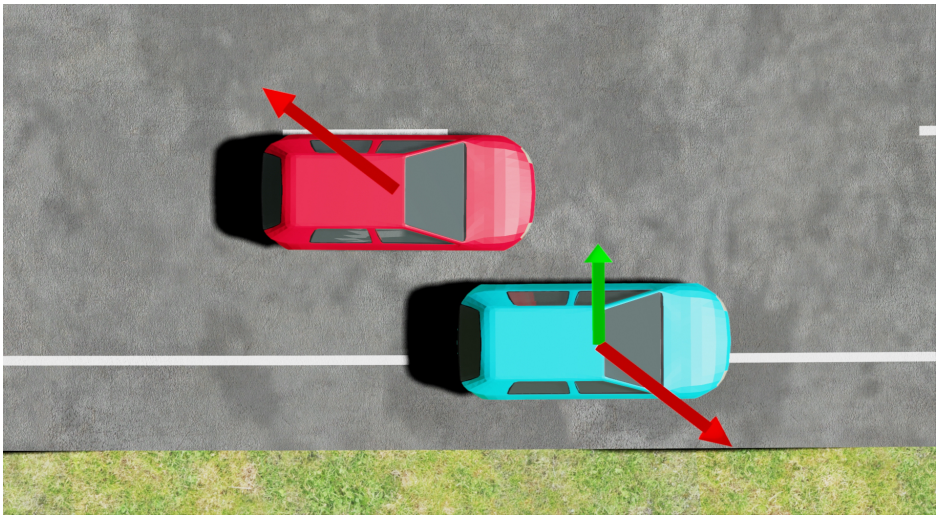


Figure 3.14: *A emergency vehicle could potentially cause regular vehicles to steer off the road, due to the Avoid Behavior (red arrows). When an emergency vehicle is approaching from behind, the regular traffic has to yield. The emergency vehicle is driving faster than regular vehicles and will hence keep pushing regular vehicles away till the emergency vehicle has passed. The KIR Behavior is needed to ensure that regular vehicles do not drive off the road when yielding to emergency vehicles. The figure illustrates a blue vehicle being urged to drive off the road, by the Avoid Behavior, to yield to an emergency vehicle. The KIR Behavior cancels out the sideways effect of the Avoid Behavior and ensures that the blue car remains on the road.*

are driving faster than regular road traffic and will try to overtake the slower vehicles. Figure 3.14 shows a scenario where a red emergency vehicle is trying to overtake a blue vehicle. The Avoid Behavior is causing the blue vehicle to move to the right, and the emergency vehicle to move to the left. The issue in this scenario is that the blue vehicle is already at the edge of the road, it can not yield anymore. When the blue vehicle is approaching the edge of the road the KIR Behavior will increase its magnitude and at a certain distance from the edge of the road, the Avoid and KIR will cancel each other out and the blue vehicle will start to follow the road straight. The result will be that the blue vehicle will move some to the right, and the emergency vehicle will move, the rest of what is needed for overtaking, to the left.

3.6.5 Avoid Oncoming

Avoid Oncoming (AO) is the behavior aimed at ensuring vehicles do not collide with oncoming vehicles. When two vehicles, vehicle v_i and vehicles v_j, are driving towards each other with no lanes to govern which vehicle has to yield, we need to define which vehicle should yield and where to move when yielding. We have chosen a relatively simple way of handling oncoming road traffic as stated in the following definition.

All vehicles shall move to the right when yielding to oncoming road traffic.

The reason for keeping a simple rule is, first of all, to avoid unnecessary complexity. If v_i is facing v_j on a collision course, we want both of them to yield as it is more efficient than having only one vehicle yield. Yielding to the right was chosen above left because vehicles in Norway are currently driving on the right-hand side, and exit ramps in Norway are adapted to vehicles driving on the right-hand side. Any vehicle, wanting to exit, that is not driving on the right would have to turn before exiting. Additionally a vehicle driving on the left will have to cross the road to be able to exit. Having to cross the road will both decrease efficiency and cause potential danger associated with crossing the road.

AO is inspired from the *obstacle avoidance* steering behavior in [Reynolds, 1999]. The significant attribute of obstacle avoidance is that the behavior does not urge an actor to steer away from the obstacle (flee from it), obstacle avoidance only affects obstacles directly in front of the actor. Hence it is possible for vehicles to drive very close (parallel) to oncoming road traffic, but not directly in front of the oncoming vehicles.

Figure 3.15 illustrates three important attributes regarding how the AO Behavior works. Where the AO Steering Behavior urges the vehicle to steer is indicated by the brown arrows. Firstly, v_2 together with v_5 illustrates the most basic scenario. Two vehicles are on a collision course and either one or both vehicles have to yield. As we have defined, both v_2 and v_5 shall move to the right. Hence, they both yield to each other.

Secondly, v_1 and v_4 illustrates an important scenario concerning how the road is distributed between driving directions. v_4 should move to the right to yield to the approaching vehicle v_1. However, v_4 is already at the edge of the road and can not yield any farther. Because vehicles are yielding to the right, the farthest a vehicles can yield (without driving off the road), is all the way over to the right side. Hence v_4 has already yielded as much as possible, and it does not make sense for v_4 to try to yield any more. Because a vehicle can only yield so much, we have configured a virtual yield-line that indicates when a vehicle does not have to yield anymore. If a vehicle is as far as possible to the right, i.e., fully beyond the yield-line, only the oncoming vehicle has to yield. This scenario is

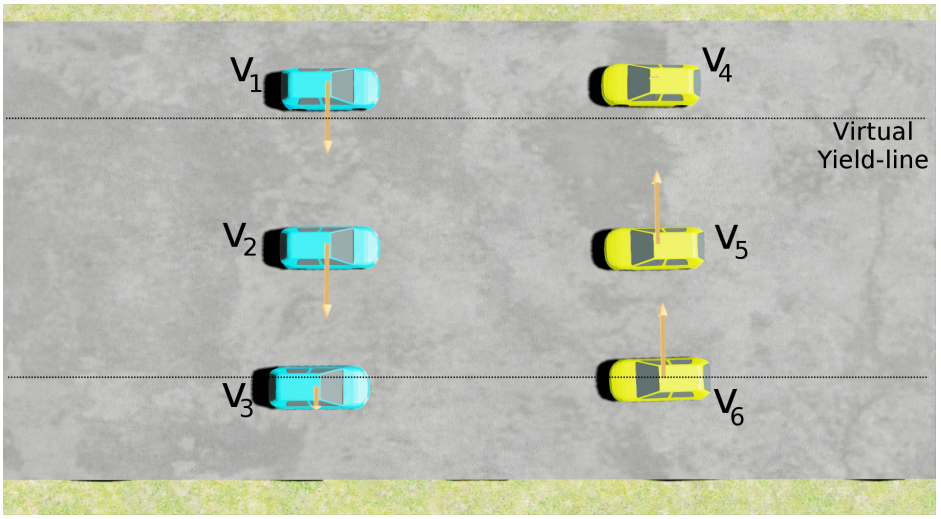


Figure 3.15: *An illustration of three important attributes of the Avoid Oncoming Behavior (AO). See the text for details. AO is the behavior aimed at ensuring oncoming vehicles do not collide. The virtual yield-line is only drawn for illustrative purposes and is not drawn in the actual simulations.*

illustrated in Figure 3.15 by v_1 yielding to v_4 (indicated by the brown arrow), while v_4 is not yielding to v_1.

The third important attribute of the AO Behavior is that it is dynamical with respect to two properties of the road traffic. Both of the properties are illustrated in Figure 3.15 by vehicle v_3 and v_6. The first property is how aligned the vehicles are laterally, i.e., if the vehicles are perfectly aligned in front of each other, the AO steering vector will have a large magnitude. As the vehicles turn away from each other and become more offset, the magnitude will decrease. The second property that determines the magnitude of the AO steering vector is how far across the yield-line a vehicle is. In Figure 3.15 vehicle v_3 illustrates how the AO steering vector magnitude decreases as a vehicle crosses the yield-line. v_3 is yielding less for v_6 than v_6 is for v_3. The reason v_3 is yielding less when crossing the yield-line is that v_3 will soon encounter the edge of the road and will be unable to yield anymore. Hence, the yielding is slowed down in advance to avoid an abrupt style of driving.

$$\mathbf{v}_{des} = \frac{\mathbf{v}_{dir}}{\|\mathbf{v}_{dir}\|} * \text{alignment} * \text{yield-line} \tag{3.6}$$

Equation 3.6 shows how the AO desired steering vector $\mathbf{v}_{des}$ is computed for a vehicle v_i. $\mathbf{v}_{dir}$ is the desired steering direction, and it will always be to the right, due to vehicles yielding to the right. *alignment* is a parameter describing how aligned v_i is laterally to an oncoming vehicle v_j. The more "in-front-of-each-other" the vehicles are, the higher *alignment* will be. $yield - line$ is similar to *alignment* in that it varies with how far across the yield-line v_i is. If v_i is all the way across the yield-line, $yield - line$ will be zero and v_i will not yield.

3.6.6 Avoid Prioritized

By Research Question 3, appropriate types of road traffic should be prioritized over others. Avoid Prioritized (AP) is the behavior that causes regular vehicles to yield when a vehicle with a higher priority is approaching. High priority vehicles will be allowed to drive faster than ordinary vehicles and will hence need the ordinary vehicles to yield. In our experiment, we have two types of vehicles with higher priority than the ordinary road traffic, buses, and emergency vehicles. Regular cars have priority 0, buses have priority 1, and emergency vehicles have priority 2. Emergency vehicles are, in other words, prioritized over buses and hence both buses and regular cars will yield to emergency vehicles.

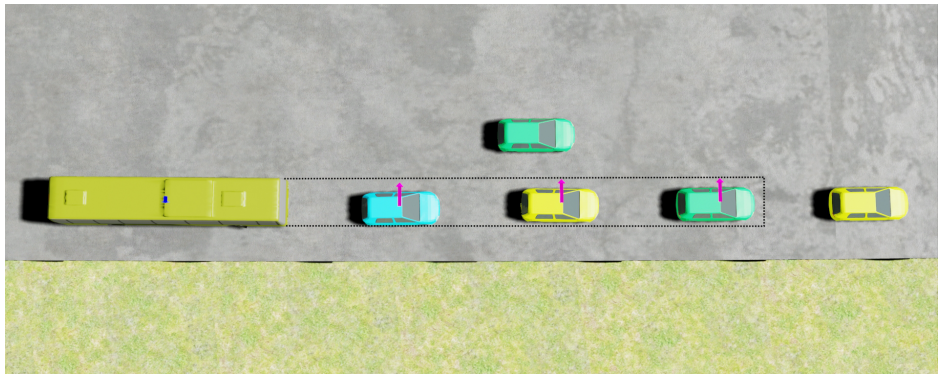


Figure 3.16: *Avoid Prioritized (AP) works by reserving an area in front of the vehicles with higher priority. The area is illustrated by a dotted line in front of the bus. Vehicles in front of the high-priority vehicle will have to move out of the area to give way to the bus. The AP Behavior is illustrated as purple arrows.*

Figure 3.16 shows a scenario where a bus is approaching a group of regular vehicles. The AP Behavior works by reserving an area in front of the high-priority vehicle. In Figure 3.16 this area is illustrated by a dotted line. All vehicles with a lower priority than the bus will have to move out of the area to give way to the approaching bus.

The AP Steering Behavior can be compared with the *leader following* behavior in [Reynolds, 1999] where boids can not enter a reserved area in front of the leader.

Figure 3.17 shows an emergency vehicle with its reserved area in front. Notice that while the bus has to keep to the right side of the road, the emergency vehicle have no limitations on where to drive. Regular vehicles in front will move to the closest side to escape the reserved area, instead of just driving to the left.

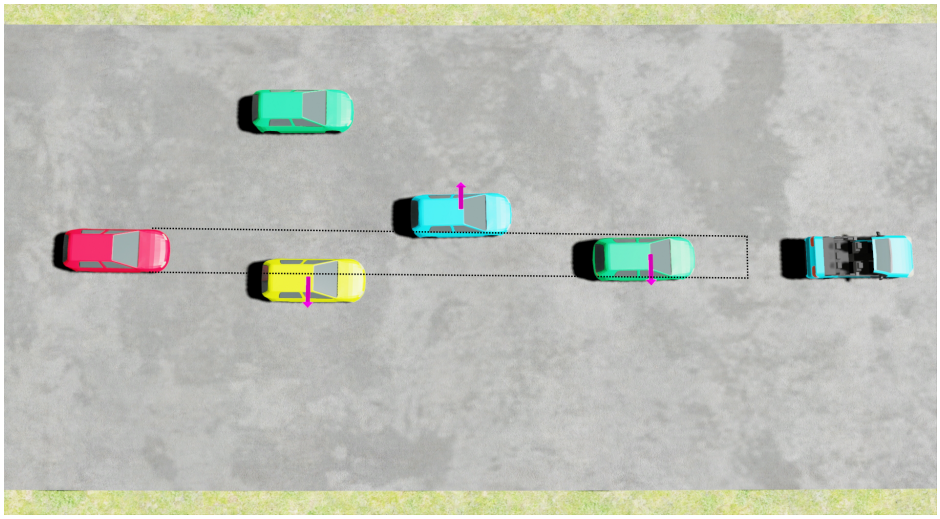


Figure 3.17: *An emergency vehicle with its reserved area in front. The emergency vehicle have no constraints on where in the road to drive and ordinary vehicles in front will move to the closest side to escape the reserved area. The AP Behavior is illustrated as purple arrows.*

The reason for reserving an area in front of the prioritized vehicles is that they are driving at a higher speed than the ordinary vehicles. Hence, the regular Avoid Behavior is not enough to avoid collisions. The regular Avoid Behavior as described in Section 3.6.3 is designed for vehicles with similar velocities and hence works for the personal vehicles that are all driving at 63km/h. Due to the emergency vehicle driving at 90km/h the margins created by the Avoid Behavior are not big enough to allow the emergency vehicle to avoid collisions. Even if the margins from the regular Avoid Behavior were big enough, the emergency vehicle would waste time waiting for the ordinary vehicles to clear the way for the emergency vehicle to pass. Hence, we decided to reserve an area in front of the prioritized vehicles both to avoid collisions and allow the prioritized vehicles to maintain a stable high speed.

3.6.7 Keep Right

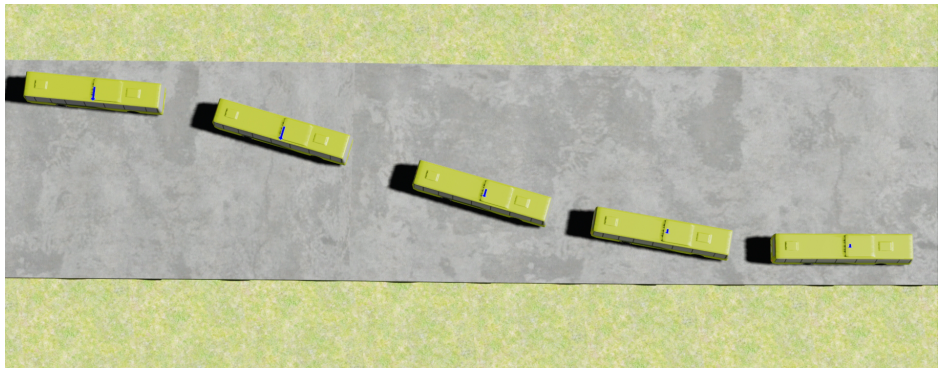


Figure 3.18: *The Keep Right Behavior (KR) is non-linear and produces a steering vector with a larger magnitude if the bus is on the left-hand side than if the bus is on the right-hand side.*

The Keep Right Behavior (KR) is designed to ensure buses stay on the right-hand side of the road. We want buses close to the edge of the road as they will probably have frequent stops to let passengers on and off, even on the highway buses may frequently exit. To ensure the buses have easy access to exit the highway or let on passengers at bus stops, we have decided that buses should always drive on the right-hand side of the road.

The KR Behavior is non-linear in the sense that if a bus is on the left-hand side of the road, the KR Behavior will produce a steering vector with a larger magnitude than if the bus were on the right-hand side, see Figure 3.18. The Avoid Behavior may cause buses to drift over to the left-hand side to avoid collisions with, e.g., vehicles on entrance ramps or emergency vehicles. We want the bus to seek back to the right-hand side as quickly as possible, but still follow a smooth driving path. A smooth ride is more stable as it avoids jagged steering, it looks better in simulations, and it feels more comfortable for the bus passengers.

$$\mathbf{v}_{des} = \frac{\mathbf{v}_{side}}{\left(\frac{l}{s}\right)^{x}} \tag{3.7}$$

Equation 3.7 demonstrates how the KR Behavior steering vector is calculated. $\mathbf{v}_{side}$ is a vector that points to the right of the bus, the vector is normalized and has length 1. We want the KR Behavior to be dynamic with respect to how far to the left-hand side of the road a bus is located. l is the distance from the bus to the left-hand side of the road, s is simply a constant scaling factor to adjust the impact of dividing by l. x is a constant exponent to ensure that KR produces

a steering vector with greater magnitude when the bus is on the left side of the road than on the right side of the road. x is what makes KR non-linear. When the bus is on the right-hand side of the road, l will be large and made ever larger by the exponent x. The result will be that $\mathbf{v}_{side}$ is divided by a large number and $\mathbf{v}_{des}$ will have a low magnitude. The other scenario is when the bus is on the left-hand side of the road, and l is small. l will be made slightly larger by the exponent x, but not as much as if the bus were on the right-hand side. The result will be that $\mathbf{v}_{side}$ is divided by a small number and hence $\mathbf{v}_{des}$ will have a larger magnitude than if the bus were on the right-hand side.

3.6.8 On Ramp

The On Ramp Behavior steers the vehicle while it is driving down the entrance ramp. The behavior is very similar to the Road Tangent Behavior, however instead of following the tangent defined by the road, it follows the tangent defined by the entrance ramp. Figure 3.19 shows where the vehicles are spawned on the ramp and where they enter the road. The dotted line represents the virtual shift from being on the entrance ramp to being on the road. Note that the dotted line is only drawn in Figure 3.19 and can not be seen in the simulations. The point on the road where a vehicle crosses the dotted line is termed the entrance point.

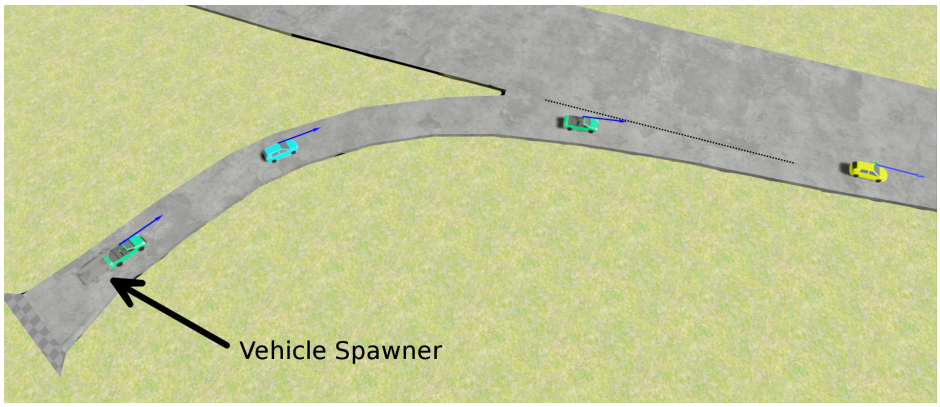


Figure 3.19: *An illustration of vehicles on the entrance ramp entering the road. The vehicle spawner is located at the start of the ramp, and the dotted line is the virtual boundary for where the entrance ramp becomes road.*

3.6.9 Waiting On Ramp

When the vehicles are spawned on the entrance ramp, they are starting in a special waiting behavior set. This set includes only the Waiting On Ramp Behavior. The concept is similar to the unaligned collision avoidance behavior described in [Reynolds, 1999]. After a vehicle is spawned on the ramp, the vehicle will check if a road entrance at time t will conflict with any higher priority vehicles driving past the entrance point. If no such conflicts exist, the vehicles will enter the road. The time t is the current time of the simulation, plus the time it will take the vehicles to drive down the entrance ramp. We have measured the time it takes a vehicle to drive down the entrance ramp to be 6 seconds. In other words, when spawned, a vehicle will listen to all vehicles within CAM range and detect any higher priority vehicles that would conflict with a road entrance 6 seconds into the future. If no conflict exist, the vehicle will start to enter the road. If a conflict exists, the vehicle will wait till the conflicting vehicle have passed. The implementation specific `GoalSpeed` field of the CAM messages are used to determine future position of possibly conflicting vehicles. The `GoalSpeed` is more accurate, changes faster, and reflects the future speed of a vehicle, and is hence better suited for accurate road traffic predictions.

There also needs to be an additional margin between the entrance point and any higher priority vehicles. The margin is necessary due to the higher priority vehicles having a higher speed than the regular vehicle. If a spawned vehicle drove down the ramp and entered the road at a speed of 63km/h, right in front of a bus that was driving 72km/h, the bus would not have time to slow down. The result would be a collision occurring. Another reason for having margins is that the road traffic is generally unpredictable, a scenario may arise that would cause the bus to slow down or speed up and arrive at the entrance point at time different than predicted.

3.6.10 Cohesion

Cohesion is the most important behavior for flocking to occur and the behavior is our implementation of the Cohesion Behavior described in [Reynolds, 1999] Animals flock in order to protect themselves from predators, benefit from a larger collective search for food, and social and mating activities [Shaw, 1970]. A vehicle does not have any of these urges and hence cohesion is only useful if the road traffic efficiency is improved. A flock of birds typically requires all birds to fly at similar velocities. Otherwise collisions would occur. If all our vehicles were driving at the same speed, cohesion could help keep the vehicles close to each other and occupy less road space. A possible disadvantage of cohesion is that keeping vehicles close could increase the risk of collisions. All vehicles driving at the same speed would interfere with prioritizing buses and emergency vehicles as

they need to be driving faster than regular vehicles.

An advantage of keeping vehicles close is that platooning could occur, i.e., vehicles driving in a column that reduces drag. The drag reduction is highest when the vehicles are ordered in a line. To increase the chance of vehicles aligning, we have designed the Cohesion Behavior to only consider lateral (sideways) distance. There are many factors affecting how the road traffic evolves (e.g., oncoming vehicles, emergency vehicles, merging from entrance ramps). Simply having a Cohesion Behavior does not guarantee that platooning will occur, but the chances are increased.

The Cohesion Behavior only affects vehicles traveling the same direction, naturally we do not want vehicles to align with oncoming vehicles as that could induce head-on collisions. Cohesion will additionally reduce the chance of colliding with oncoming vehicles since vehicles will group together and increase the distance to the oncoming vehicles.

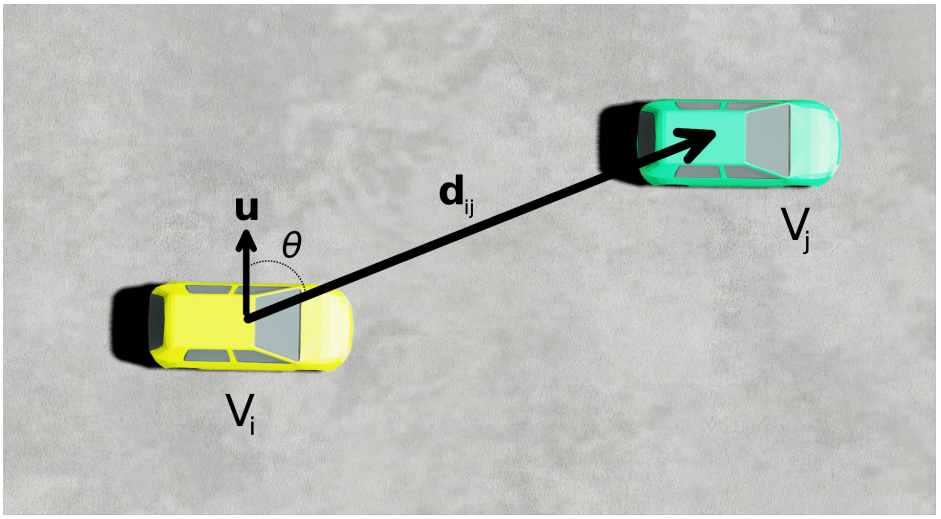


Figure 3.20: *This figure illustrates the vectors and angles needed for computing the steering vector for the Cohesion Behavior.* $\mathbf{u}$ *is a unit vector that points to the side of* v_i*, and* $\mathbf{d}_{ij}$ *is a vector from the center of* v_i *to the center of* v_j*.*

Specifically the Cohesion Behavior works by projecting a vector to each neighbor onto a vector pointing to the side of the vehicle. E.g., a vehicle v_i will compute a vector pointing from v_i to a neighbor v_j as shown in equation 3.4. The vector $\mathbf{d}_{ij}$ is then projected onto a unit vector $\mathbf{u}$ pointing to the side of v_i. Figure 3.20 illustrates the scenario with labeled vehicles and vectors, and Equation 3.8 shows how the projection is calculated. The resulting $\mathbf{v}_{proj}$ is the projection of $\mathbf{d}_{ij}$ onto

the unit vector $\mathbf{u}$.

$$\mathbf{v}_{proj} = \|\mathbf{d}_{ij}\| \cos\theta \tag{3.8}$$

A $\mathbf{v}_{proj}$ vector is computed for each vehicle within a defined range, the upper bound on this range is the CAM range. All the projected vectors are then summarized to form the final steering vector $\mathbf{v}_{des}$ from the Cohesion Behavior. $\mathbf{v}_{des}$ may point to either side of v_i depending on how many neighbors are on each side, and how far away (laterally) each neighbor is.

Figure 3.21: *The Cohesion Behavior can be thought of a creating a virtual line between the center of a group of vehicles that the group will try to align themselves on laterally.*

Figure 3.21 illustrates the effect of the Cohesion Behavior. Due to the Cohesion Behavior only considering lateral alignment, the resulting effect can be thought of as a virtual line, drawn where the vehicles desire to be located. Our approach to the Cohesion Behavior is similar to what was done in [Kala and Warwick, 2012] by breaking the steering down into lateral and longitudinal (spatial) steering.

The road traffic environment is dynamic, and changes can happen at any time, particularly emergency vehicles are a catalyst for changes. If the Cohesion Behavior is too strong, a flock of vehicles will not be able to split up to let an emergency vehicle through. A flock of vehicles needs to be able to split, and do so in a short amount of time hence we have chosen to let the Cohesion Behavior

be weighted low compared to other behaviors. Consequently, the desire to avoid a prioritized emergency vehicle is stronger that the desire to flock with it and a flock is capable of splitting fast. Another reason for having a weak Cohesion Behavior is that the road traffic environment is a constrained environment. The width of the road can not be exceeded, and any yielding to oncoming traffic or overtaking by emergency vehicles will have to happen withing the limits of the road. Hence, a strong Cohesion Behavior would effectively block the road for other vehicles and passing around a flock would become a bottleneck.

To keep the system simple, we have not implemented different cohesion behaviors for different types of vehicles. All vehicles can benefit from reduced drag if only for a short while. Even for vehicles of higher priority as they will quickly have to overtake regular vehicles. We believe it could be interesting to investigate potential effects of changing the Cohesion Behavior, see future work in Section 6.3.

Cohesion is linearly dependent on the distance to the flock center (the dotted line is shown in Figure 3.21). We want the Cohesion Behavior to be smooth and hence decrease in magnitude as a vehicle is approaching the desired position.

3.7 Running Unreal Engine with Fixed Time Steps

Since Unreal Engine is a game engine, it is created to run such that the in-game time is synchronized to the time of the user playing the game, the real time. However, there is no way to guarantee the run-time of the code of the game. In a game running on the computer, the time is not continuous, it is divided into so-called time steps. The game is made up of a loop, this loop is run, the time is increased by some delta, and the loop is run again. This loop is repeated until the game is closed. All the logic that controls the game exists in this loop. Simply said, the loop starts with every actor in the game being allowed to run its tick code, the physics are run, and at last the current scene is drawn by the GPU.

Our code is added in this game loop since we are adding code to the tick functions of the actors that make up the simulations, i.e. the vehicles. Hence, how often our code is run is defined by how fast the game loop is running. In the case of Unreal Engine, this is tied to how fast the machine can run the game loop. Which again is tied to how fast the GPU is drawing to the screen, the speed of the physics, and the speed of all the other code in the loop. The speed can vary based on, e.g., how many vehicles there are on the screen and how fast the machine of the user is. Therefore *the interval the game code is run at, is not constant.*

Variable running intervals are unacceptable for our simulation. We want to run at about 30 frames per second, which means about 33.33 ms between every update. However, if some other program is using the processor for a while, there

can suddenly be half a second between two updates. If our vehicles were running 80 km/h, every vehicle would have moved 11 meters between two updates from our code since the game time and real time is tied together. That is 11 meters without control from the automatic vehicle. The physics engine is running on sub stepping, which means that it will have run all the 15 steps it should have at this time, but it would have been using the last input from the vehicle. This means that if the vehicle wanted to turn a bit to the right to avoid another vehicle, it would have been driving 11 meters with the wheels turned.

To avoid variable running intervals, we want to run the simulation with fixed time steps, meaning that the time of the simulator will always increase 33.33 ms for every step of the game loop. However, we still want the machine to run the loop as fast as possible. Hence, the game can run faster or slower than real time, depending on the load of the machine.

To achieve fixed timesteps, the documentation of unreal engine claim that we can use a parameter `BENCHMARK`. However, we have found that this does not apply in our use case. However, some digging in the source code revealed that the undocumented argument `useFixedTimeSteps` should have the same effect and work with our setup. The details with why and how is in Appendix Section E.1.

3.8 Controlling and Taking Measurements of the Simulation

For the experimental setup, we want to be able to both control the simulation, and take measurements while the simulation is running. To be able to quickly change the environment of the simulation, we will create the following model. The simulation can be thought of like a highway where vehicles are spawned from a set of spawners. Each spawner has parameters that control how often a spawning will occur, and what vehicle will be spawned. The simulation will listen to commands on a network socket, where it can be commanded to start or stop the spawners, and where all settings can be specified.

On the other side of this socket, there will be a controller program. Since network sockets can be used from any language, we can write this controller program in any language we want. We will choose Python since we have experience with using python, and the Python code is easy to change. A typical session will proceed like the following. (1) The controller is started by the user (or some automatic script). (2) The controller spawns the simulator program by asking the Operating System to start the executable that is the simulator. (3) The simulator starts a listening network socket, and the controller connects to this socket. (4) The controller asks the simulator to provide the behaviors compiled into the simulator, and which version of the code that is used to compile the

simulator. (5) The controller asks the simulator to pause all spawners, remove the vehicles that might be on the highway and set all spawners to inactive. (6) The controller sends the wanted configuration for each of the active spawners, including which vehicles should be spawned for each spawner and how often each spawner should spawn a vehicle. (7) The controller asks the simulator to start recording the session to a file it specifies. (8) The controller asks the simulator to unpause the spawners and reset all stats.

The simulation will now be started. Each time one second of game time has passed, the simulator will send the statistics for the simulation to the controller. This includes the number of seconds of game time passed, the number of incidents and the total number of vehicles spawned. When the controller received the information that a pre-defined time has passed, called the warm-up time, it will determine that the simulator is warmed up, i.e. has reached its steady state. The controller will then start to save all statistics received from the simulator. The simulation is completed when the controller receives information that another pre-defined number of seconds, called the run-time, has passed. On receiving this information, the controller asks the simulator to stop the recording of the session, pause the spawners, and in some configuration close the simulator program. The controller will gather all information it has received from the simulator and store it, either locally or to a central server.

The workflow includes some technical interesting solutions. Firstly the communication between the controller and the simulator. As mentioned, this will be done using a network socket. The reason for this is that a network socket is simple to implement in most languages. Making the controller a separate application, beside the simulator, gives us the freedom to change more easily how the simulator is used. It also makes us able to have different types of controller applications, for different use cases.

We will be creating a simple remote control protocol, on top of the standard socket for communications between the simulator and the controller. The messages that go back and forth will be encoded using MessagePack [Furuhashi, 2013]. MessagePack is a simple encoding without a schema, which makes it easy and fast to use. It has implementations in most languages, including both C, C++ and Python. We will be using the simpler C implementation, instead of the C++ implementation. This is due to how Unreal Engine compiles the source code, which makes it easier to use C bindings. Unreal Engine also has a separate implementation from most of the C++ standard library. The C++ binding uses the standard library of C++. Using the C bindings, we will make a new implementation that uses the containers accompanying Unreal Engine.

To make sure we are using the same parameters for the behaviors in all experiments, our code will also gather and store the behaviors used. Since this is defined using C++ classes and the GUI of Unreal Engine, we needed to use

the reflection provided by Unreal Engine. Reflection is a programming mechanism that provides meta information about classes. Using reflection, we will be able to extract the information from each enabled behavior, without hard coding the variables used for each behavior class. This information is encoded using MessagePack and transferred to the controller.

3.9 Capturing and Playing Replays

As mentioned above, the simulation will be capturing replays for each experiment. The replays can be used after the experiment to inspect events and look at the location of each vehicle in every timestep of the simulation. Simply said, the replay is a file where the location and other data for each vehicle is saved for every timestep. The information that will be saved for each tick for each vehicle is.

- Unique ID.
- Location
- Rotation
- Vectors for all the behaviors
- What behavior set is used
- Velocity
- Goal Speed
- Current Gear
- Engine Rotation Speed
- Current Throttle

Using all the information is enough to replay the simulation exactly how it happened. We will be creating a replay player, using the same simulation, but another vehicle class, called a Mocking Vehicle. This Mocking Vehicle will be created by subclassing a plain Actor. Meaning that the Mocking Vehicle is not actually a real vehicle, but merely looks like it. The Mocking Vehicle will be like a token, moved around by the replay player using the replay recorded during the actual simulation.

We will also create the replay player such that the simulation can be run both backward and forwards. Hence, when a vehicle is finished and removed, we need

to store the same information as when a vehicle is created. The reason is that the event of removing a vehicle, is the same event as spawning a vehicle when the replay is run backwards.

The greatest challenge when implementing replays is the amount of information that is needed to be read and written in a short amount of time. An estimate is about 190 bytes per vehicle per tick. If there are 50 vehicles on the road, this gives for 30 ticks per second: $190\,B/car * 50\,cars * 30\,s = 285\,kB/s$. For 60 minutes, this is $285\,kB/s * 60\,s/m * 60\,m/h \approx 1\,GB/h$. We will be using MessagePack to encode the data that is to be saved to disk since we already will be using MessagePack for socket communications.

The large size is the reason that the replays are stored directly to file from the simulator, instead of being passed on the socket. The information is saved to memory for each tick and then saved to file after some pre-determined amount of memory is filled. Since the file system has its own buffers, this should make the process relatively fast.

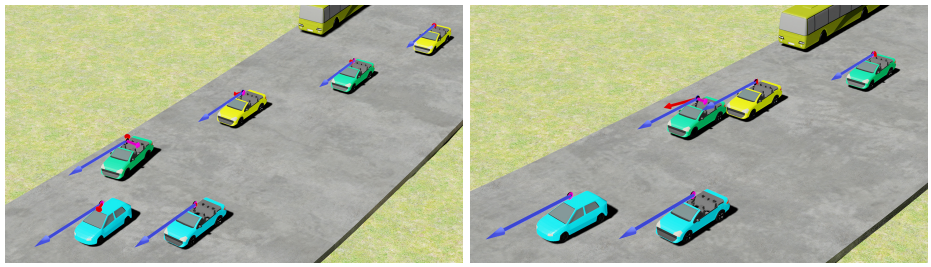
The process of replaying the recording is a bit more difficult, the challenge being that we need to be able to play both backward and forwards in the file. The file model for C++ is great for playing a file forwards; however, the inverse is more difficult. Fortunately, there is a method called mapping a file, where the file is mapped into the logical memory of the process. Using mapping, we can move both forwards and backward in the file. The operating system will move the content automatically from the hard drive to the memory when needed, and remove the content from memory when it is no longer needed.

3.10 Model Validation

When we had created the model described above, we used some time to tweak the implementation and parameters for the result to be as good as possible. We ran the experiments using the same techniques described in the next chapter and looked at the results, and all the incidents that occurred. To be able to look at the incidents, we used the replay capturing mentioned above, which lets us see exactly when the incidents happened, and also look at what happened before and after the incident. In this section, we will take a look at two examples of incident types that were observed, describing the modifications done to reduce or eliminate the type of incident.

3.10.1 Example A: Avoiding the Bus

The first incident was frequently observed until some parameter were tweaked. One example of the incident is presented in Figure 3.22. This is what happens: (a) The green car is on the right-hand side of the road, and a bus approaches



(a) *The green car needs to yield to the bus.* **(b)** *The green car collides with the yellow car.*

Figure 3.22: *Example A: Avoiding the Bus.*

from behind. The green car wants to steer left, but there are already two cars on its left. One blue car is a slightly in front of the green, and one yellow car is slightly behind. The blue car in front is somewhat more to the left than the yellow car behind. (b) As the green car turns to the left, the blue car in front is detected as being too close because the Avoid Behavior works directly along the ForwardVector of a vehicle. Hence, the green car starts to brake even though it had a safe distance to the blue vehicle. The braking is amplified by the PID controller sometimes braking more than necessary (see more about harsh braking incidents in Section 5.1.3). The result, shown in Figure 3.22, is that the green car brakes too much and collides with the yellow car.

The main cause of the incident in Figure 3.22 is that the green car detects the blue vehicle as being too close.

Two modifications were done to avoid this type of incident. Firstly the margins for the Avoid Behavior were increased, giving the vehicles more space on the road. More space gives the cars more time for both braking and accelerating after braking. Secondly, observe how the blue car in front is only in front of the green when we define the direction to be where the car is pointed. On the highway, the road traffic moves in the direction of the highway, even though some cars might change their direction momentarily to change position on the road. Hence, it is much more interesting to use the tangent of the road direction to detect vehicles in front. The second modification was done to the Avoid Behavior, changing the behavior in such a way that it instead uses the tangent of the road to define the direction of forward. These two changes made this type of incident much rarer.

3.10.2 Example B: Bus Colliding With Car on Entrance

The second example is another frequently observed incident. As mentioned before, the cars on the entrance ramp wait until the highway is cleared before they

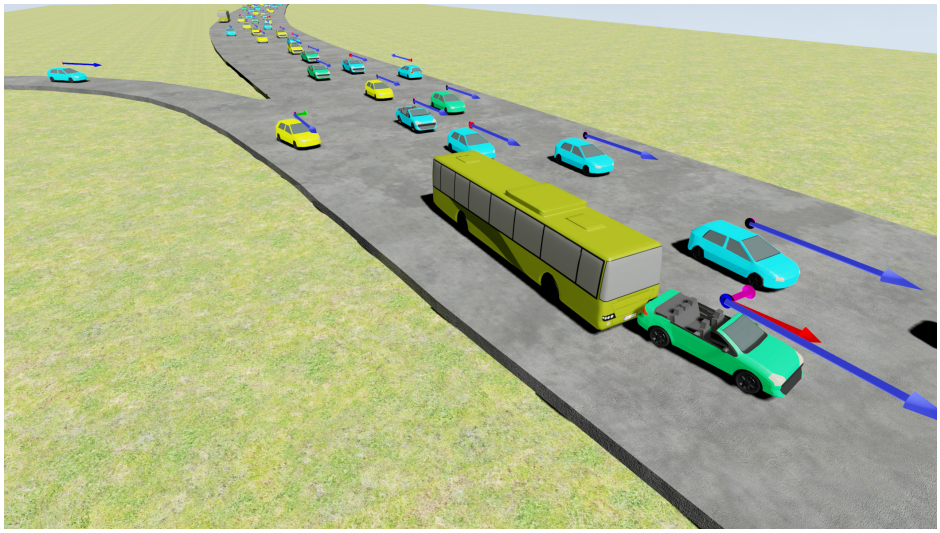


Figure 3.23: *Example B: A bus is colliding with a car that have entered the highway from the entrance.*

start to accelerate. However, to determine when the highway is cleared they have to predict the future. They do this by assuming that all priority vehicles, in this case buses, will be running with constant velocity from the point they are observed until the meeting point for the highway and entrance ramp. This assumption is not always accurate. Buses might brake or accelerate after the observed point. An example is shown in Figure 3.23. Here the bus have accelerated after being observed by the car. The car have thought that it could start to accelerate and that it would have enough time to move in front of the bus. Unfortunately, because of the change in speed, the bus have arrived at the entrance point earlier. The contrary have also been observed. The bus have braked, and a car that though it should be able to get behind a bus has ended up to the right of the bus, unable to get behind it.

This incident could have been avoided by making the car's on-entrance behavior smarter, accelerating faster or slower depending on the buses on the highway. However, such a behavior would be complicated since the car also needs to accelerate fast enough to reach highway speed before getting on the highway. Instead the margins the car uses to calculate when it should start, was increased. The increase gives the bus space for some accelerating and braking, making sure the car still avoids the bus. This type of incident has not been observed since the margins were increased.

Chapter 4

Experiments and Results

The following sections presents our experimental plan, setup, and results. In addition, we present justifications for why the experiments were conducted the way they were. We then present the results.

4.1 Experimental Plan

The experimental plan outlines the reasons for conducting experiments and what we hope the results will reveal. We also describe the different types of experiments we are planning on running and why.

4.1.1 Experimental Goals

We have designed our experiments to be able to answer our research questions. We will perform multiple runs of different scenarios with varying throughput. A key issue is to verify that multiple runs are possible without collisions. This project is a concept study, and an important aspect is to verify whether flocking is a viable concept for governing road traffic. As the road throughput increases collisions will eventually happen, we are interested in when collisions first start to happen and how fast the number of collisions increases with throughput.

4.1.2 Number of Simulations and Duration

Real-world road traffic with 3 seconds between each vehicle will have a throughput of $\frac{3600}{3} = 1200$ vehicles per hour per lane (assuming we ignore the length of the vehicles). With gaps of 2 seconds between each vehicle, the throughput becomes $\frac{3600}{2} = 1800$. Private communications with the NPRA have confirmed that a

throughput between 1200 and 1800 is realistic in real-world traffic. A road with two lanes in each direction may, at full capacity, have a throughput of $1800 * 2 = 3200$ vehicles in each direction per hour. Hence, for flocking to make sense we need to be able to manage a throughput of at least 3200 vehicles in one direction. We will mostly be simulating asymmetric road traffic with a few vehicles in the other direction to imitate morning or afternoon rush traffic, as asymmetric road traffic is where flocking is predicted to cause the greatest efficiency improvement. We will only be measuring the throughput in the direction with the highest traffic load, in the other direction there will be a vehicle on average every 25 seconds. In the direction where we measure throughput, we will start with a throughput of 2000 vehicles per hour and increase to 14800 in steps of 400, i.e., 33 different throughputs. 14800 was chosen due to test trials indicating that the number of incidents would begin to rise significantly before 14800.

We will test 33 different throughputs for each experiment, and each throughput will be tested 30 times, hence each graph in Section 4.3 will contain $33 * 30 = 990$ data points.

Each experiment will simulate 30 minutes of road traffic. We chose 30 minutes to get a decent sample of the traffic scenario while still keeping the interval short enough to allow for flexibility in the experiment scheduling.

4.1.3 The Experiments

This subsection will outline the experiments we intend to execute and the purpose of each one. We will also give an overview of critical parts of the experiments.

Experiment 1 (Baseline): Only regular vehicles travelling one direction

Experiment 1 is a baseline experiment that we will conduct to verify that our flocking implementation can manage a substantial amount of vehicles before any incidents start to occur. Figure 4.1 shows the road traffic with only regular vehicles driving only one direction.

Experiment 2: Oncoming Traffic

This experiment is concerned with Research Question 1. We want to investigate if the behaviors described in Section 3.6, particularly the Avoid and AO, are enough to ensure runs without collisions. We would also like to investigate how the number of collisions evolves as the throughput increases. We will be spawning vehicles at both ends of our road. Figure 4.2 illustrates the scenario. Five spawners will be active at one end of our road, as shown in Figure 4.2a, spawning vehicles with decreasing intervals as we turn the throughput up. At

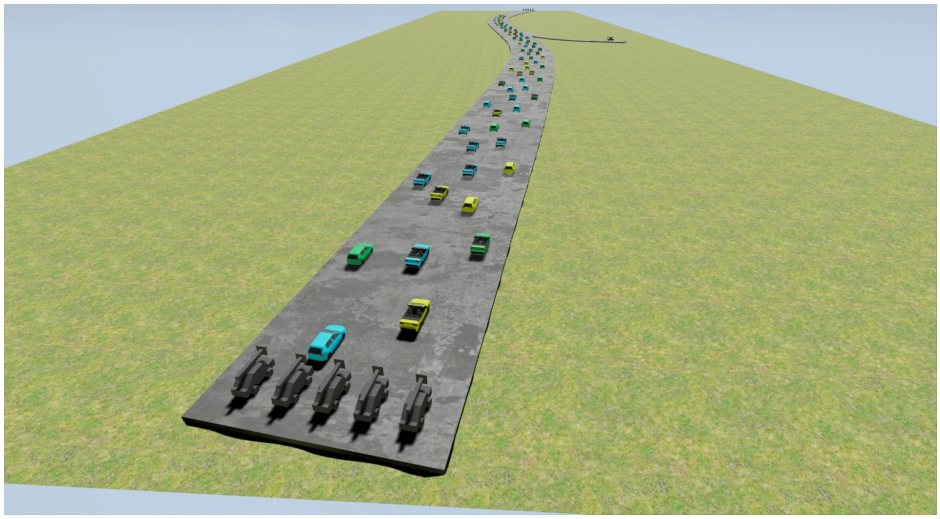


Figure 4.1: *The baseline scenario where five spawners are active. All vehicles are driving in the same direction, and traffic only consists of regular vehicles.*

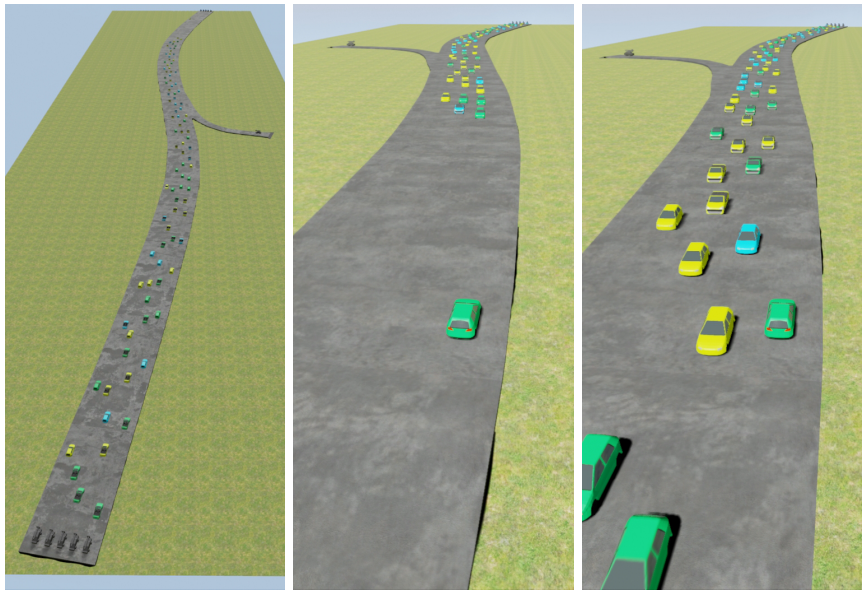
the other end of the road a single spawner will be active, spawning one vehicle with larger intervals to simulate asymmetric traffic as shown in Figure 4.2b and 4.2c. See Figure 4.3 for an overview of where all the different spawners are located. Specifically, the active spawners in this experiment will be S_{1-5} and S_7.

A defining part of Experiment 2 is that there will only be spawned regular vehicles, and no vehicles will be entering the road from the entrance ramp. Experiment 2 is the first experiment simply meant to validate the concept of flocking for steering vehicles on the road.

Experiment 3: Oncoming and Merging Traffic

Our third experiment is designed to answer Research Question 2, concerning merging traffic from an entrance ramp. As with Experiment 2; we will only be spawning regular vehicles, five spawners will be active at one end, and one spawner will be active at the other end. The difference between Experiment 2 and Experiment 3 is that Experiment 3 will have an additional active spawner at the beginning of the entrance ramp, labeled S_6 in Figure 4.3. Figure 4.4 shows an illustration of a typical scenario that arises when a vehicle is trying to merge from an entrance ramp onto the road.

Experiment 3 will evaluate whether the behaviors we have designed are enough to ensure that vehicles can enter the road without collisions. Especially the KIR



(a) **(b)** **(c)**

Figure 4.2: *Scenarios describing Experiment 2. Figure 4.2a shows the five vehicle spawners located at one end, Figure 4.2b shows a single vehicle coming from the other end at the start of the experiment, and Figure 4.2c shows how asymmetric traffic looks in the simulation.*

and Avoid Behavior will be critical for guiding the vehicles safely onto the road.

Experiment 4: Oncoming, Merging, and Buses

The third experiment focuses on answering Research Question 3. We have chosen to split Research Question 3 into three experiments; one experiment for prioritizing buses, one for emergency vehicles, and one for both. Experiment 4 will build on the same setup as Experiment 3; there will be five active spawners at one end, one active spawner in the other, and one active spawner on the entrance ramp. The new element in Experiment 4 is that two of the spawners, S_1 and S_7, will be spawning buses besides regular vehicles. The frequency of spawned buses will be 3 out of 13 from S_1, and 1 out of 11 from S_7, all other spawners will only be spawning regular vehicles. Since all spawners S_{1-7} spawns approximately the same number of vehicles, 3 out of $13 * 6$, or $\frac{3}{13*6} = 0.03846 \approx 3.8\%$ of the vehicles in the direction with highest throughput will be buses. In the direction with a low throughput $\frac{1}{11} = 0.0909 \approx 9.1\%$ will be buses.

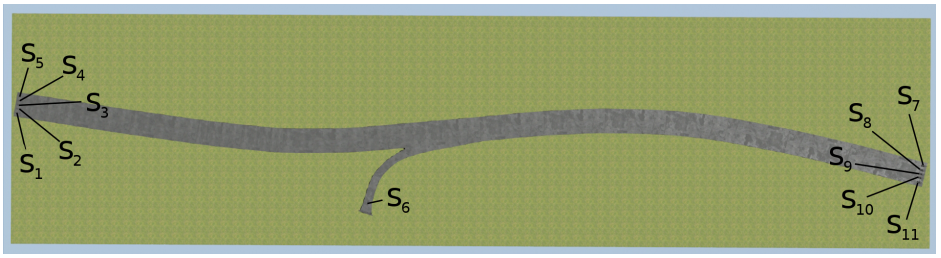


Figure 4.3: *An overview of the road segment, highlighting the location of all the spawners used in our experiments.*

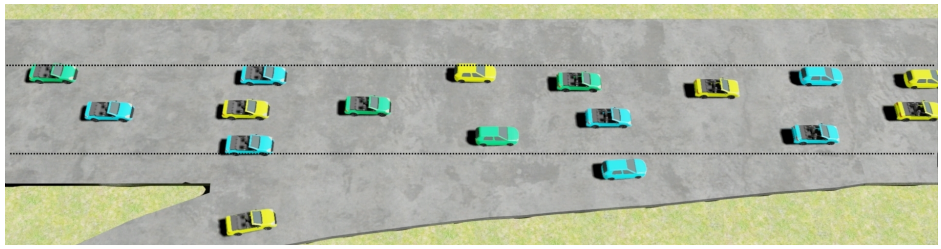
Experiment 5: Oncoming, Merging, and Emergency Vehicles

This experiment also builds on Experiment 3, but adding emergency vehicles instead of buses. Emergency vehicles travel at a default speed of 90km/h as opposed to 70km/h for the buses. They are smaller than buses, and do not abide the KR Behavior and may hence locate themselves anywhere on the road. Due to buses always driving on the right-hand side, all regular vehicles that yield to a bus will have to move left. Vehicles yielding to an emergency vehicles may move either left or right depending on which side is closest to being out of the emergency vehicle's way. The extra flexibility in movement direction when yielding to an emergency vehicle will probably cause more volatility in the road traffic, which again could increase the number of incidents.

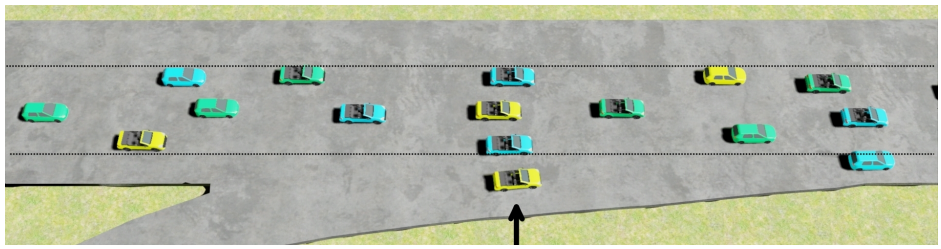
The active spawners in this experiment will be spawners S_{1-7} with spawner S_3 being the one that will spawn emergency vehicles. The frequency of emergency vehicles will be 1 out of 11, which means that S_3 will spawn ten regular vehicles for each emergency vehicle on average. All other spawners will only be spawning regular vehicles. The percentage of emergency vehicles will be: $\frac{1}{11*6} = 0.01515 \approx 1.6\%$ in the direction where we measure throughput.

Experiment 6: Oncoming, Merging, Emergency Vehicles, and Buses

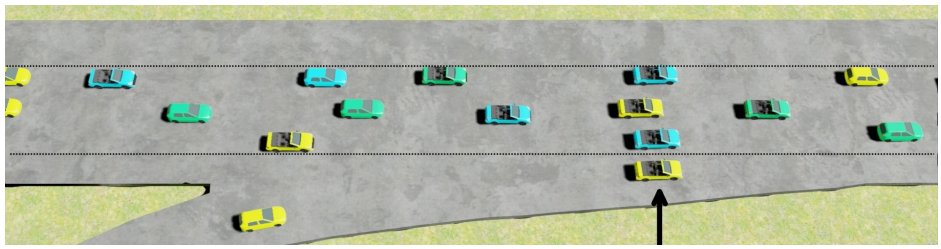
We will conduct an experiment to test how all our vehicle types behave when operating together in road traffic. In this experiment spawners S_{1-7} will be active, and the amount of buses and emergency vehicles spawned will be the same as in Experiment 4 and 5. Buses will be spawned from S_1 and S_7, Emergency vehicles will be spawned from S_3.



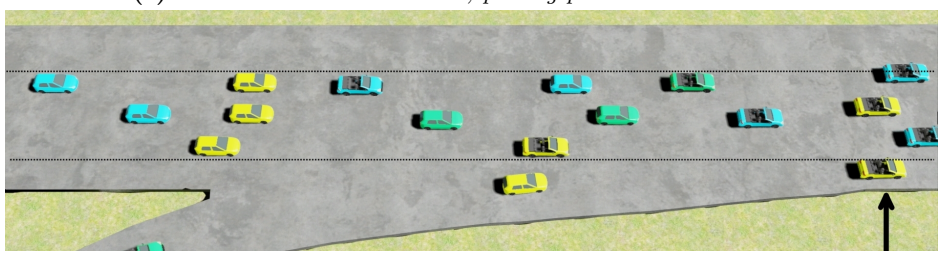
(a) *Yellow vehicle is coming from the entrance ramp.*



(b) *Yellow vehicle is starting to pressure the blue vehicle.*



(c) *Blue vehicle starts to move, putting pressure on the others.*



(d) *Yellow vehicle has successfully entered the road.*

Figure 4.4: *A time-lapse showing a yellow vehicle entering the road. The Avoid Behavior enables the yellow vehicle to enter the road by forcing the three vehicles to its left to move. Notice the dotted lines that can be used as references to lateral movement.*

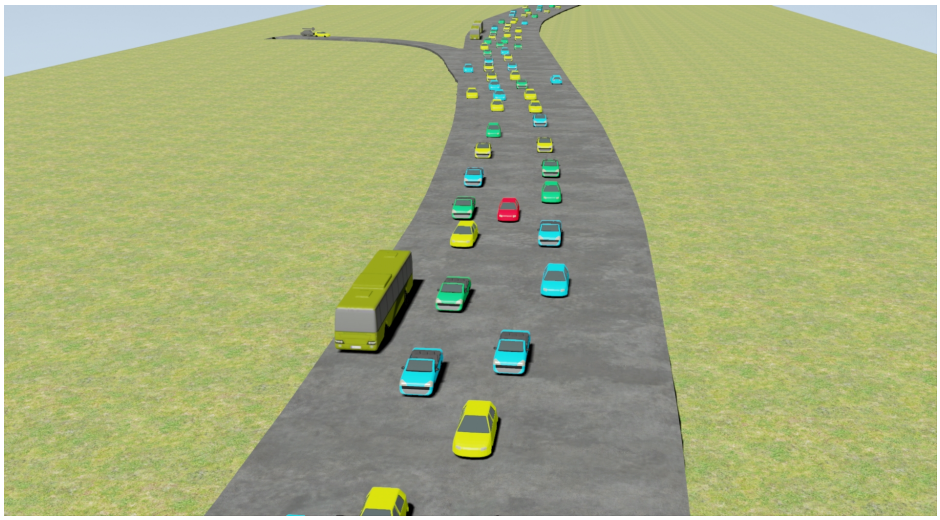


Figure 4.5: *A typical road traffic scenario when both buses, emergency vehicles, merging traffic, and oncoming traffic is active. Note the areas in front of the emergency vehicle and the buses that are cleared due to the AP Behavior.*

Experiment 7: Symmetric Road Traffic.

Asymmetric road traffic allows for the greatest performance improvement by utilizing flocking as a road traffic management system. However, we want to conduct a control experiment to show how well flocking performs for symmetric road traffic as well. In this experiment spawners S_{1-3} will be active on one end and S_{7-9} on the other. In addition, S_6 will be active on the entrance ramp. All active spawners will only spawn regular vehicles. Figure 4.6 shows an illustration of how the symmetric traffic looks when governed by flocking. Due to vehicles moving to the right when yielding to oncoming traffic, the road has a tendency of being divided into two directional flows. The crux of this experiment is the vehicles which are spawned from spawner S_3 and S_9 that are inherently on collision course with each other and will have to abide the AO Behavior to avoid collisions. Note that the measured throughput in this experiment is only in one direction. Hence, the real throughput will be approximately doubled.

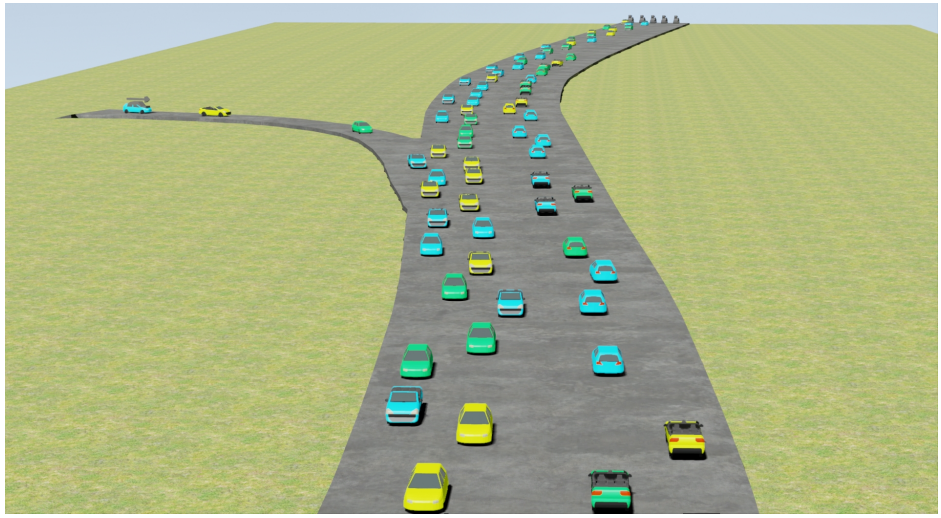


Figure 4.6: *An illustration of flocking used to manage symmetric road traffic.*

4.2 Experimental Setup

The experimental setup outlines many important parameters for our experiments that have an impact on the results. In this section, we will describe implementation details and parameters used for the experiments.

4.2.1 The Vehicle Spawners

Figure 4.7 shows how a vehicle spawner looks. Every vehicle spawner has a timer that determines when vehicles are spawned. After a vehicle is spawned, the spawner will pick a random number r from and interval between m and n, where m is the lower bound on r and n is the upper bound, and wait r seconds before the next vehicle is spawned. We have chosen to use randomness in the vehicle spawning in order to emulate real-world road traffic. With no other interferences, the throughput in cars per hour per spawner for some time period can be found by using the formula shown in equation 4.1 where I is the mean value for r. We use equation 4.1 to calculate the spawn interval range for a given throughput.

$$Throughput = \frac{3600}{I} \tag{4.1}$$

The throughput may not be exactly as predicted, due to some instances where the spawners will wait an extra amount of time before they spawn the next vehicle,

e.g., if the previous vehicle spawned has not yet left the area in front of the spawner. We have implemented a security mechanism for spawning vehicles that ensure vehicles will need to be spawned at least 10m apart to avoid incidents. The waiting period is especially frequent for the spawner on the entrance ramp because the spawned vehicle will wait for prioritized vehicles to pass before entering. Therefore, the measured throughput will be lower than the predicted throughput, especially for higher throughputs. Note that when we present results, we will always use the measured throughput.

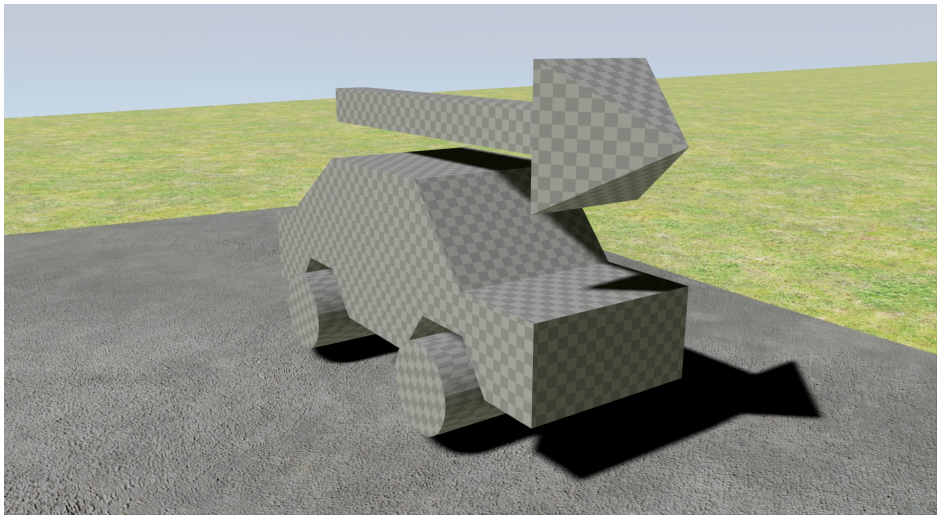


Figure 4.7: *A vehicle spawner.*

An issue with spawning is that vehicles can not be spawned with stable speed. Due to inertia, the vehicles need time to let the engine reach the appropriate rpm for driving. After spawning, a vehicle's speed will very quickly decline until the appropriate rpm has been reached. We are therefore spawning all vehicles in 100km/h, letting them brake till the desired speed is reached. Another solution could have been to spawn the vehicles with a speed of 0km/h and let them accelerate to the desired speed. However, acceleration would require a longer road segment that would lead to extra vehicles on the road and increase the experiment run times.

Multiple vehicles may be spawned next to each other with small time differences. The proximity of the vehicles will then cause the Avoid Behavior to become unstable, switching from side to side with great magnitude in short time intervals until the vehicles find their steady state. We have termed this phenomenon *unstable spawning*. Unstable spawning increases the chances of incidents, espe-

cially if an oncoming vehicle is close, which adds the AO Behavior to an already unstable system. The high speed of spawned vehicles is also contributing to instability. Incidents due to unstable spawning typically happens within the first second after spawning. The incidents due to unstable spawning is a consequence of the way we have chosen to spawn vehicles, but we have found the number of incidents to be small, and only occurring at high throughputs. Spawning incidents would also never occur in the real world since the spawning is an artifact of the simulation setup. Hence, we have not prioritized modifications that could mend the problem.

For an illustration of the magnitude of the unstable spawner problem see Section 4.3.2 and Figure 4.11, and also Figure 5.1 for an illustration of how the incident appears.

4.2.2 CAMs

We have chosen to transmit CAMs 10 times per second. When emergency vehicles are driving at 90km/h, they will manage to drive 2.5m in 0.1 seconds. All vehicles are predicting the future position of other vehicles (within CAM range) based on their last CAM, and if the acceleration is zero the prediction will be exact. Even if the vehicles have some acceleration, we have judged the predictions to be good enough. We would like to transmit CAMs as often as possible. However, if every vehicle were to transmit a CAM 1000 times per second, it would severely impact the run times of our experiments. Hence, CAM transmit frequency is a trade-off. The same would be true in a real world scenario, where a vehicle is only able to transmit its location a fixed number of times every second.

The range of a CAM is defined to be 150m. The CAM range is also a tradeoff between runtime and performance. Ideally we would like all vehicles to know about all other vehicles. However, this would require each vehicle to process more information than reasonable and severely slow down our system. We have chosen 150m as range as it gives vehicles time to react to oncoming vehicles without impacting the run times too much. An emergency vehicle driving 90km/h will drive 150m in 6 seconds.

4.2.3 The Road Segment

The spawned vehicles do not travel the entire road segment. Vehicles spawned at S_{1-6} are removed once they cross finishing line B, as illustrated in Figure 4.8. Vehicles spawned at S_{7-11} are removed at finishing line A. Once a vehicle is spawned, the steering behaviors will become active and the vehicle will start to position itself.

To mitigate the effects of unstable spawning we have decided to remove the oncoming vehicles before they reach the spawners on the other side of the road

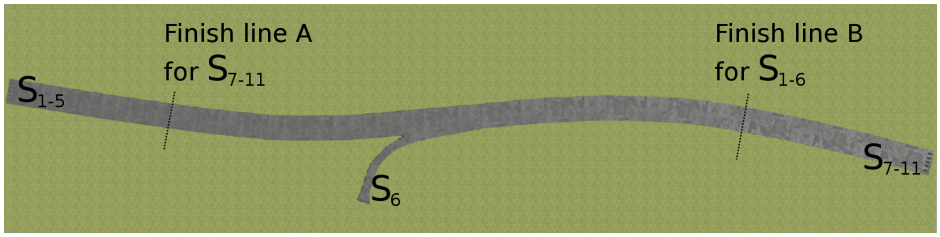


Figure 4.8: *An illustration of the road segment, showing the finishing lines for the vehicles spawned at the respective spawners. S_i indicates the spawner, and the dotted lines visualize the virtual finishing lines.*

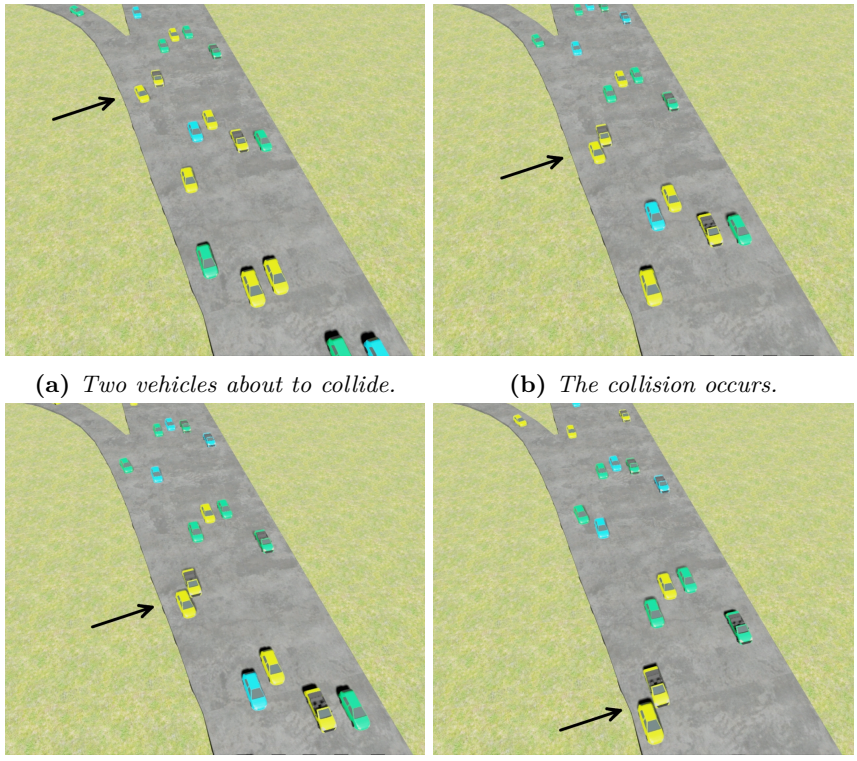
segment. The margin gives the spawned vehicles a chance to stabilize before coping with oncoming traffic. The road segment between finishing line A and B is roughly 400m.

4.2.4 Defining Incidents

In the real world there are many types of collisions and traffic accidents; some are fatal and other do not even scratch the paint coating. In our setup, collisions can be detected, but the force of an impact can not be evaluated. Hence, we have chosen to define all collisions as *incidents* regardless of severity. Figure 4.9 shows a scenario where two vehicles bump into each other, and both keep on driving as if nothing happened. While a human driver would typically not tolerate another driver bumping into his car, the computers have no predisposition neither towards nor against collisions. For a computer, an incident would simply be an event that occurred. However, due to incidents usually causing material damage that would require funds to repair, we will regard all incidents as undesirable and count them as equal. Consequently, every time two vehicles have any degree of contact we count it as an incident.

4.2.5 Incident Actions

What should happen when an incident occurs? One option is to let the simulation play on, hoping that the incident was nothing severe. Another is to stop the simulation concluding that it ended with an accident. Alternatively, we could also simply remove the vehicle(s) involved in the incident and let the simulation play on for a predefined amount of time. An incident could involve a single vehicle if the vehicle drives off the road. However, we have yet to observe such a scenario. The behaviors we have designed are indirectly adapted to stable road traffic, i.e., the behaviors can not handle emergency braking or evaluate when



(a) *Two vehicles about to collide.* **(b)** *The collision occurs.*

(c) *Both vehicles keep driving.* **(d)** *Both vehicles still unaffected.*

Figure 4.9: *A time-lapse of two vehicles bumping into each other, both vehicles continue driving as if nothing happened. The scenario is artificially created by turning off the Avoid Behavior that would otherwise have prevented this scenario from arising.*

it is safe to resume driving after an emergency stop. Thus, if a severe incident occurs it could quickly develop into a major collision as shown in Figure 4.10.

The reason for not implementing emergency braking is that an emergency is a very complex scenario. A vehicle would need a mechanism to determine firstly that a scenario is an emergency, secondly determine how much braking is needed, and thirdly accelerate when is the emergency situation over. We know automatic vehicles are capable of emergency braking as the currently build automatic vehicles are doing it. Therefore, emergency braking is not in need of concept studies to prove its viability. Another aspect of emergency braking is that it would probably cause vehicles to stop, which again would create a traffic jam. If an emergency braking occurred early in an experiment, we could risk the whole 30

min run simply being concerned with resolving the succeeding congestion. Using steering behaviors to resolve traffic jams could be interesting, but is not the scope of this study. We are interested in how steering behaviors can be used to manage regular traffic flow and hence will not spend time resolving traffic jams.

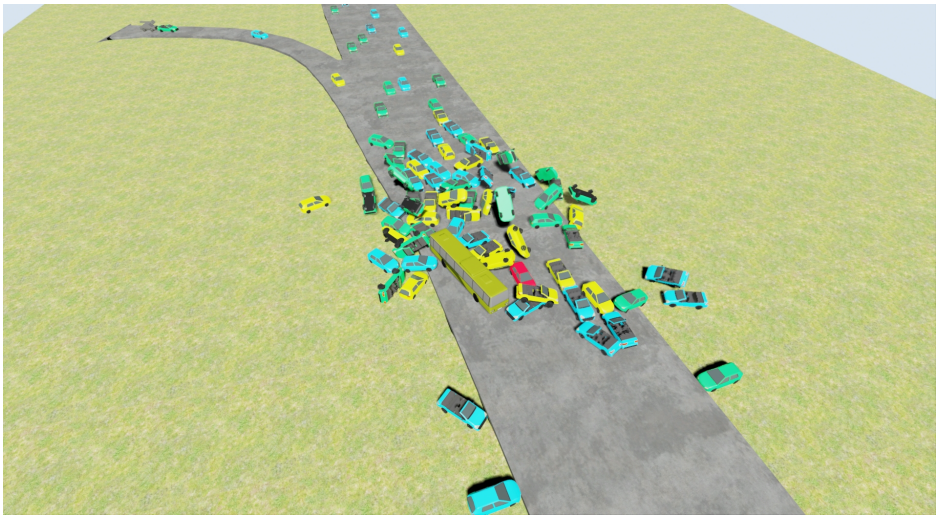


Figure 4.10: *An incident involving two vehicles may quickly develop into a major incident, somewhat resembling the car crash in the famous Blues Brother movie from 1980.*

The problem with letting collisions play out is that if a major incident occurs, a vast number of vehicles will be gathered on the road because none of them will reach the finish line, and the system will be brought to a halt due to computational load. We could stop the simulation after an incident occurs, but that could render our experiments very short if an incident occurred quickly after starting. *Thus, we have chosen to remove any vehicles involved in an incident and let the simulation continue.* There is a chance that two vehicles, vehicle v_i and v_j, having an incident could cause a neighboring vehicle, v_k, to change heading before they are removed. After v_i and v_j have had the incident and been removed, v_k will still be on its possibly flawed heading and could cause more incidents (the butterfly effect). However, we have found this effect to be negligible.

4.2.6 The Steering Behavior Parameters

All our steering behaviors have multiple parameters that can be tuned. For this project, we have used an iterative experimental approach, i.e., attempting sensible

values and verifying by considering whether the number of incidents decreased.

Automatically optimizing the steering behavior parameters requires hyper-parameter optimization, for which we could have used a grid search technique [Bergstra et al., 2011]. However, the search space is exhaustive, and each search would require at least 15 minutes of real-world time, not to mention the search implementation phase. Hence, we chose to configure the parameters manually as best we could. For an overview of the specific parameters used see Appendix F.

4.2.7 Distributed Experiments

Each experiment will be run 990 times, each run is 30 minutes of simulation time. Since we have seven different experiments, this gives $7*990*30 = 207\,900$ minutes which is equal to 3465 hours or about 144 days of simulation time. Even though we are running the simulations faster than real time, the time consumption would be substantial if we were to run all the experiments on one computer.

Fortunately, we have been given access to a computer lab with 25 powerful computers with Intel Core i7-4790 CPUs running at 3.60GHz and NVIDIA GeForce 750 Ti GPUs. We will be creating a simple task scheduling system, where 24 of these computers communicates with the last one using HTTP. The last one will be set up to be a server. The 24 workers will ask the server for a task and work on that task till they are finished. While the computers are working, they will give the server status updates. When the computers are finished, they will upload the results, including the number of incidents and the average throughput. They will also upload a file containing the replay of the whole run. The replay includes the location for each vehicle in each timestep and is used to inspect what happened during the run and why any incidents occurred.

We will also create a remote control system, which will let us easily add tasks to the system. A task is one run, normally taking 30 minutes with a particular configuration. The configuration defines the spawners to use, the random interval defining how often the individual spawners should spawn, and which vehicles should be spawned. Each combination of experiment and throughput will give one configuration.

Recall that we are using fixed timesteps. Therefore, the simulation is running as fast as it can while having 30 timesteps per second of game time. Different amounts of load will give different amounts of speedups. From experiments, it appears that a throughput of 2000 gives a speedup of 2-4 game minutes for each real-time minute. The largest throughput of about 15000 will often run slower than real time. If we assume that we can execute the simulation at an average speed of 1.5 minutes of game time for each minute of real-world time, the real-world time to run all the experiments will be $207\,900min/1.5 = 138\,600min$. Dividing on 25 machines gives $138\,600m/25 = 5\,544m = 92.4h = 3.9d$. Using the

cluster, the experiments that could have taken up to 4 months will be finished in a little more than a weekend.

4.3 Experimental Results

This section displays the results of our experiments. We have chosen to represent the results as graphs due to the results being two-dimensional and easy to understand from a graph. Incidents and causes for incidents are discussed in Chapter 5.

4.3.1 How to Read the Graphs

For each experiment, we have one graph of the results. Each of an experiment's 990 runs have been plotted, with the throughput on the x-axis and the number of incidents on the y-axis. We can see that there are clusters of points for each tested throughput. In some graphs, the clusters are very evident and in others the data point are more dispersed. We tested 33 different throughputs for each experiment and hence graphs where the clusters are visible can be seen to have 33 clusters. Each cluster contains 30 data points because we tested each throughput 30 times.

Be aware that the y-axis is of different scale in each graph due to some experiments having a small number of incidents and some having a higher.

We have used local regression [Cleveland et al., 1992] to draw the blue loess line. The blue loess line indicates where a data point is most likely to occur given its local neighborhood of data points. The blue line can hence be interpreted as the average or the most-likely value. We will use the blue line as an indication of when the traffic starts to break down, i.e. when the line begins to rise away from 0. When the blue line rises above 0.5 on the y-axis it means that a run is more likely to have 1 incident than 0. Hence, assuming that traffic breaks down as soon as the blue line rises from 0, is a very conservative estimate.

As defined in Section 4.2.4, an incident is a binary event. It either happens or it does not. Therefore, the number of incidents is discrete. To easier read the graph, we have added some random *jitter* to the y-values, i.e. the number of incidents. Without jitter, for each throughput there would be a single dot representing all runs with zero incidents and a single dot representing all runs with one incident. It would be impossible to tell the number of runs with one incident and the number of runs with zero incidents.

The throughput in the graphs is the average throughput measured in one direction, the direction from spawners S_{1-5}, which for most of the experiments is the direction with the largest traffic load. The throughput is measured after

the entrance ramp is merged. The throughput is measured by the total amount of vehicles in one direction and not on a per-lane basis as we do not have lanes.

4.3.2 Experiment 1 (Baseline)

The result from our baseline experiment can be seen in Figure 4.11. The throughput reaches 9000 vehicles per hour before any incidents occur. There were six runs that had one incident each which are the six dots plotted around 1 on the y-axis. We can also see the loess line have a slight rise under the data points with incidents, which indicates a somewhat higher chance of an incident at those throughputs.

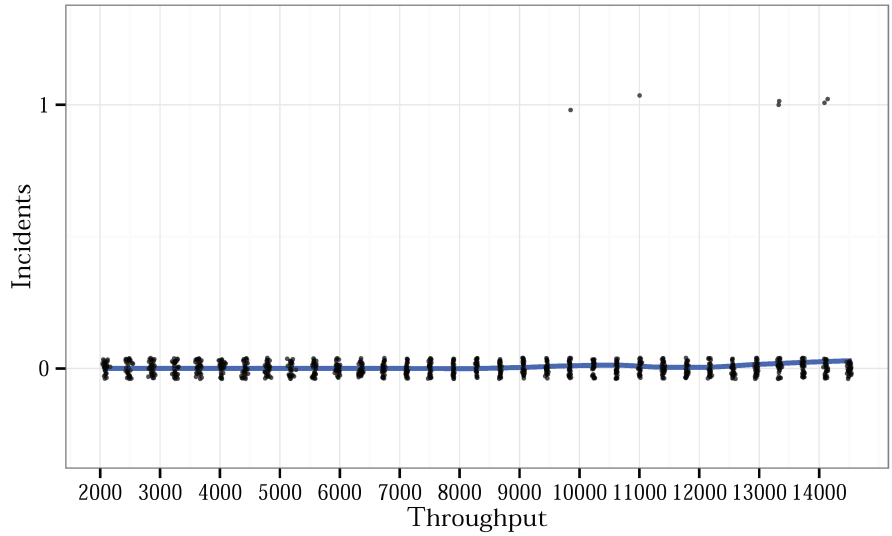


Figure 4.11: *Experiment 1 (Baseline) results. All incidents are due to unstable spawning.*

It is worth noting that all six incidents that occurred in Experiment 1 took place due to unstable spawning (see Section 5.1.1).

4.3.3 Experiment 2: Oncoming Traffic

Figure 4.12 shows the result from Experiment 2. There are no incidents before a throughput of 9000, and the loess line starts to rise at 11000 vehicles per hour.

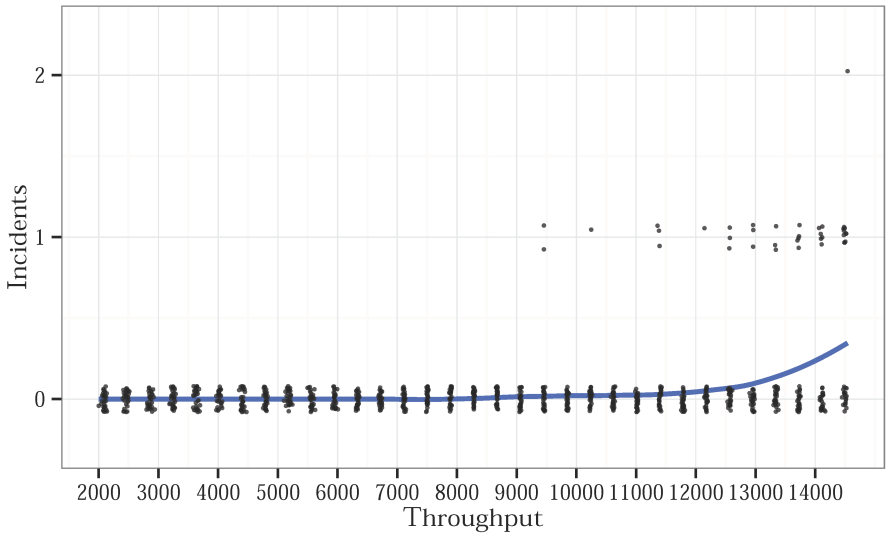


Figure 4.12: *Only oncoming road traffic.*

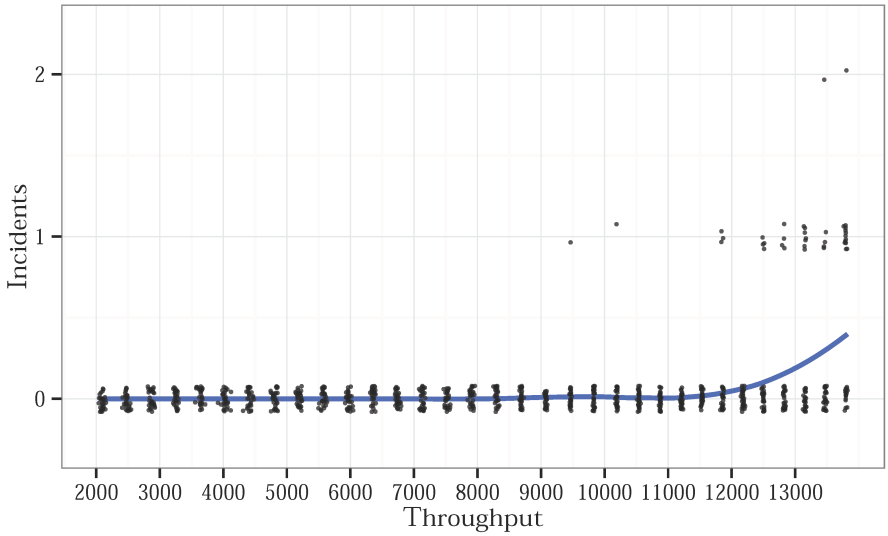


Figure 4.13: *Oncoming with merging traffic.*

4.3.4 Experiment 3: Oncoming and Merging Traffic

Figure 4.13 shows the results from Experiment 3. There are no incidents below a throughput of 9000 vehicles per hour, and the loess line starts to rise at 11000 vehicles per hour.

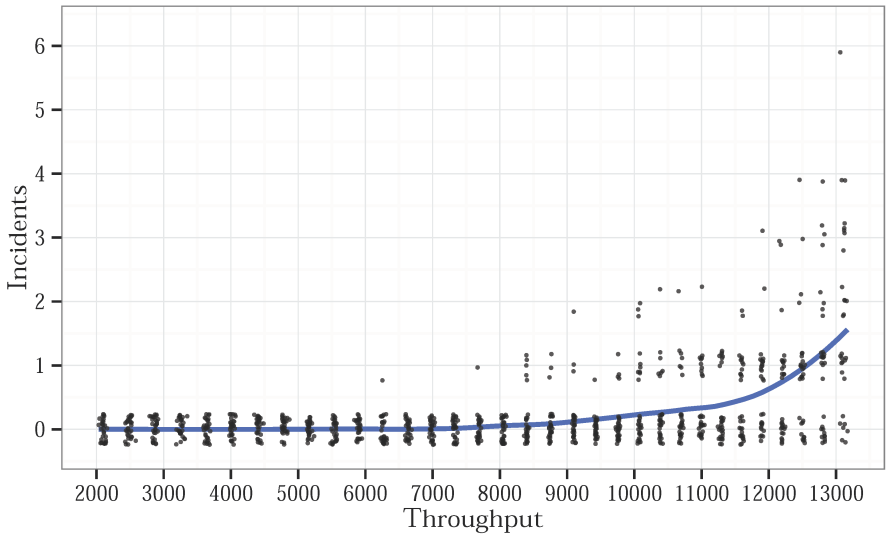


Figure 4.14: *Oncoming, merging, and buses.*

4.3.5 Experiment 4: Adding Buses

Figure 4.14 shows the results from Experiment 4. The first incident occurs after a throughput of 6000 vehicles per hour, and the loess line starts to rise at 7000 vehicles per hour.

4.3.6 Experiment 5: Adding Emergency Vehicles

Figure 4.15 shows the results from Experiment 5. The first incident occurs after 4500 vehicles per hour, and the loess line starts to rise at 7000.

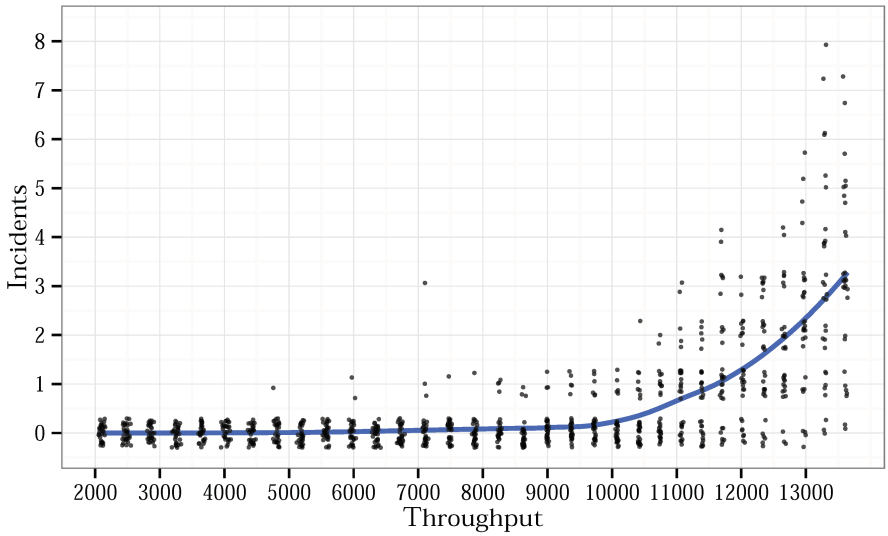


Figure 4.15: *Oncoming, merging, and emergency vehicles.*

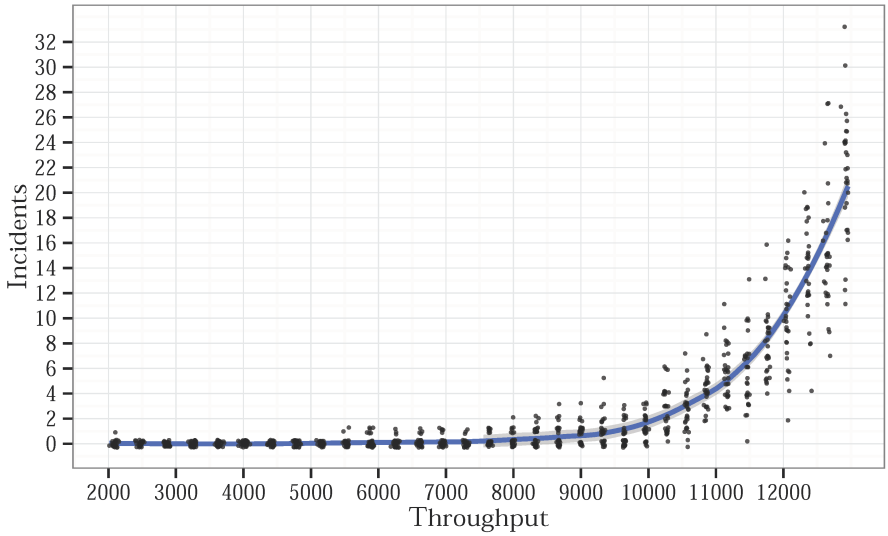


Figure 4.16: *Oncoming, merging, emergency vehicles, and buses.*

4.3.7 Experiment 6: All Vehicle Types

The result of Experiment 6 is shown in Figure 4.16. The experiment had an incident already at a throughput of 2000 vehicles per hour, this incident is discussed in Section 5.1.6. The other incidents start at 5500, and the loess line begins to rise at 7000.

4.3.8 Experiment 7: Symmetric Road Traffic

For the experiment with symmetric throughput, the first incidents occur at a throughput of slightly more than 5000 vehicles per hour, see Figure 4.17. Keep in mind that the graphs only shows the throughput in one direction. For symmetric traffic, the number of vehicles is the same in both directions and hence the real throughput before any incidents is approximately 10000 vehicles per hour.

Figure 4.17 contains two graphs. The top graph shows how the traffic suffers a total breakdown between 8000 and 9000 where there is almost no increase in throughput even though more vehicles are spawned.

The bottom graph shows a close-up of all the runs with 10 or fewer incidents to illustrate better when the incidents first start to occur and when the loess line begins to rise. For symmetric traffic, the loess line begins to increase at a throughput of 5000 vehicles per hour in one direction, which means a total throughput of 10000 vehicles per hour.

4.4 Evaluation

Our results show that our flocking implementation is capable of managing traffic flows of regular vehicles of up to at least 9000 vehicles per hour without incidents, and a throughput of at least 4500 vehicles per hour with all vehicles types. There is much randomness involved in our vehicle spawning, and hence the exact occurrence of an incident is due to both throughput and statistical chance. The blue loess line can be used as an indicator for what traffic load our flocking implementation can manage. For experiments with only regular vehicles, the line starts to rise at about 11000 vehicles per hour and for mixed traffic with all vehicle types the loess lines begins to increase at about 7000 vehicles per hour.

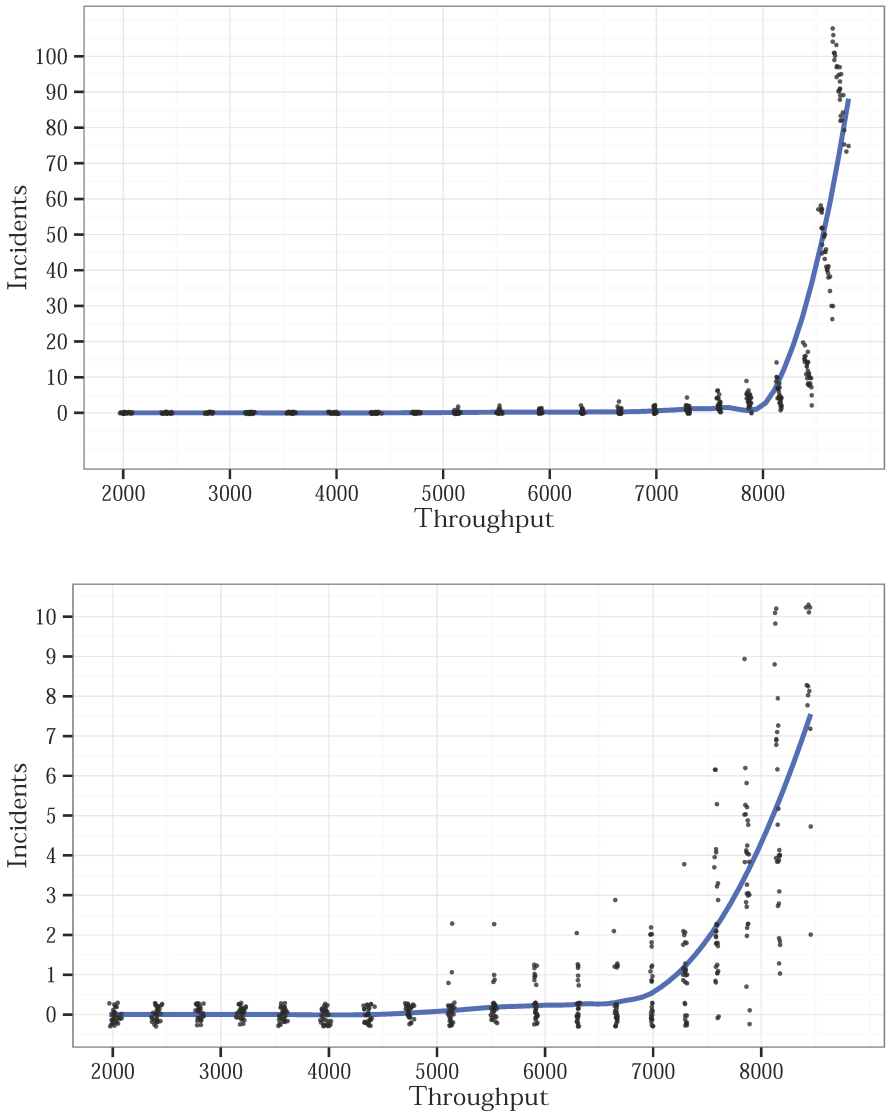


Figure 4.17: *Symmetric road traffic (The lower graph only shows the runs with 10 or fewer incidents).*

Chapter 5

Discussion

This chapter discusses the experiments performed and evaluates whether the results are reasonable. We also discuss what incidents happened and the causes why. Finally, we discuss the essence of our findings and present implications for future road traffic.

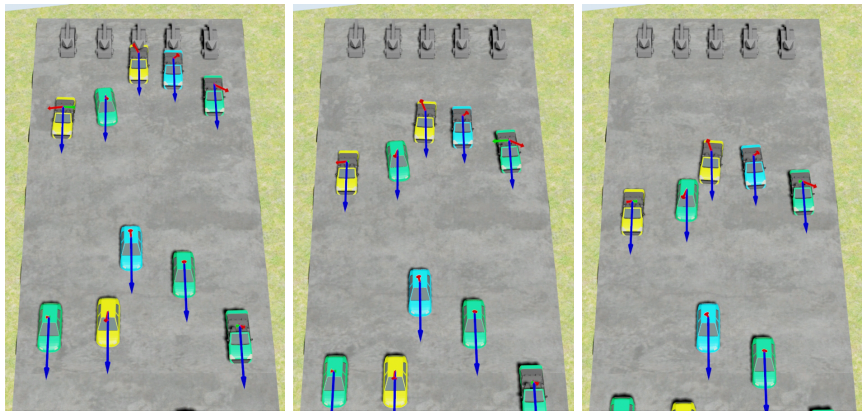
5.1 The Experiments

This section provides an analysis of each experiment and discusses incidents that occurred as well as what steps that could have been taken to prevent them. Many of the incidents are caused by the same underlying issues and we explain why the issues exist and what can be done about them.

5.1.1 Experiment 1: Baseline and Unstable Spawning

In the baseline experiment, there are no oncoming vehicles, no merging traffic, and all vehicles have equal speed. Thus, the experiments contain no dynamics, and no interaction between the vehicles should be necessary. If all vehicles keep driving at equal speed and follow the road in the lateral position where they started, there will be no collisions. Therefore, the Road Tangent Behavior alone should be enough to steer the vehicles without incidents. However, we are still using the same behaviors and parameters for Experiment 1 as we are for every experiment. Running such a basic experiments implies that all occurring incidents will reveal errors in the flocking algorithm. Fortunately, all incidents in Figure 4.11 were due to unstable spawning. Unstable spawning is a side effect of the experiment, and can be ignored. Hence, once the spawned vehicles have stabilized, our flocking

implementation can handle throughputs of more than 14.000 vehicles per hour without any incidents.



(a) **(b)** **(c)**

Figure 5.1: *An illustration of how an incident due to the unstable spawning problem appears. The green vehicle in the top left is spawned next to a yellow vehicle that causes it to steer to its left. However, another yellow vehicle has already been spawned to the left, and the green vehicle does not have time to react. See the text for more detail.*

Figure 5.1 illustrates an example of an unstable spawning incident. The green vehicle in the top left of Figure 5.1a is spawned close to a yellow vehicle on its right. The Avoid Behavior causes the green vehicle to steer left to avoid the yellow vehicle, but another yellow vehicle have already been spawned on the left. Once the green vehicle starts steering right it is too late, and the yellow vehicle hits the green vehicle from behind as shown in Figure 5.1c.

The problem of unstable spawning is a problem that occurs due to our simulation setup. Limited computational capacity prevents us from simulation large road segments, and hence we need to squeeze in both a spawning area and an experiment area on a relatively short road segment. Therefore, we need to spawn the vehicles in a state resembling steady state, so the vehicles are ready to enter the road traffic immediately. Unfortunately, we have no way to determine the exact composition of a steady state. Therefore, we have decided on a scheme where the vehicles are spawned at a random interval where the sideways distance could be much lower than what the Avoid Behavior would desire. Spawning vehicles inside the boundaries of the Avoid Behavior implies that the spawned state could be unstable, especially for high throughputs. When looking at Figure 4.11 for the baseline case, we can see that there are no collisions below a throughput of 9500 vehicles per hour. This means that unstable spawning is not a problem

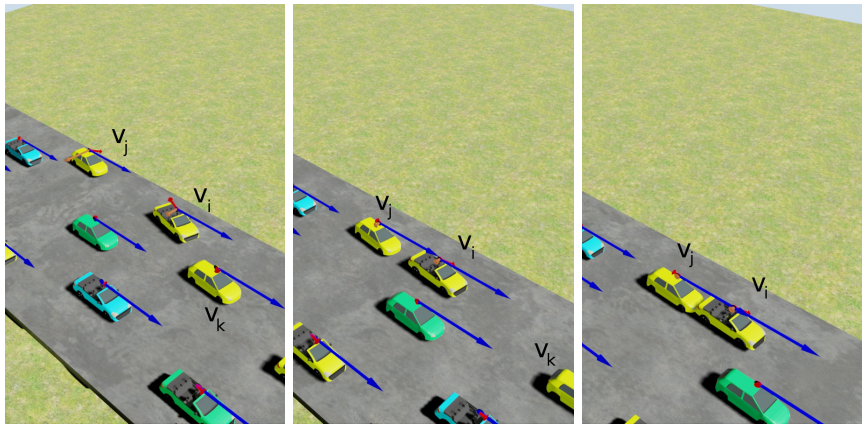
below 9500 and that the steering behaviors can achieve a steady state.

Multiple steps could be taken to spawn the vehicles in a more steady state. We could create a larger road and have all vehicles enter in a calm and stable manner. However, a large road would require more vehicles on the road to create the same throughput, and our computers can simply not effectively manage the computational load. Another solution could be to have an intelligent spawning scheme where vehicles are not spawned unless it is safe. However, such a spawning scheme would require more time spent on implementation and also somewhat reduce the pseudo-randomness in the spawned vehicles. We could also have reduced the number of spawners and had the remaining spawners increase their spawning rate. However, the spawning rate is already very close to maximum, and fewer spawners would probably have led to a reduction in throughput. Due to all the drawbacks of trying to remove the unstable spawning incidents, we have decided rather to endure the incidents and maintain high throughput. Hence, we are able to run timely experiments, and preserve the pseudo-randomness in our spawned vehicles to emulate real-world road traffic. Besides, the spawning problem only occurs in computer simulations and is not an issue in the real world.

5.1.2 Experiment 2: Oncoming Traffic

Experiment 2 with oncoming road traffic is intended to answer Research Question 1 concerned with using steering behaviors to avoid collisions. From the graph in Figure 4.12 we can see that our implementation can handle a throughput of at least 9000 vehicles per hour before any incidents start to occur.

An example of an incident that occurred in Experiment 2 is shown in Figure 5.2. A vehicle v_i and a vehicle v_j are steering left to avoid an oncoming vehicle (Figure 5.2a), as v_i is turning it is also starting to brake in order to avoid v_k. However, v_i is braking more than necessary and the sudden change in speed should have caused v_j to start emergency braking (Figure 5.2b). However, as stated in Section 4.2.5, our vehicles are not capable of emergency braking. Thus, an incident occurs (Figure 5.2c). v_i braking too much is probably due to our PID controller and gearing. When braking the vehicle loose speed fast, however gaining the same speed takes longer time. The controller asks the vehicle to break too much because it is too close to the vehicle in front. However, it is not able to gain the speed needed to not collide with the vehicle behind. v_j will detect v_i braking and also start braking a little. However, v_j is also urged to avoid the blue vehicle in the back and hence ends up having an almost unaffected speed. We do not regard the harsh barking as an extensive problem. Incidents only occur after a throughput of 9000 vehicles per hour. We believe this has to be considered acceptable performance for our implementation, given a two lanes would be able to handle $1800 * 2 = 3600$ vehicles at maximum.



(a) (b) (c)

Figure 5.2: *An incident from Experiment 2, with throughput above 9000. Occasionally our PID controller can brake more harshly than intended and cause incidents.*

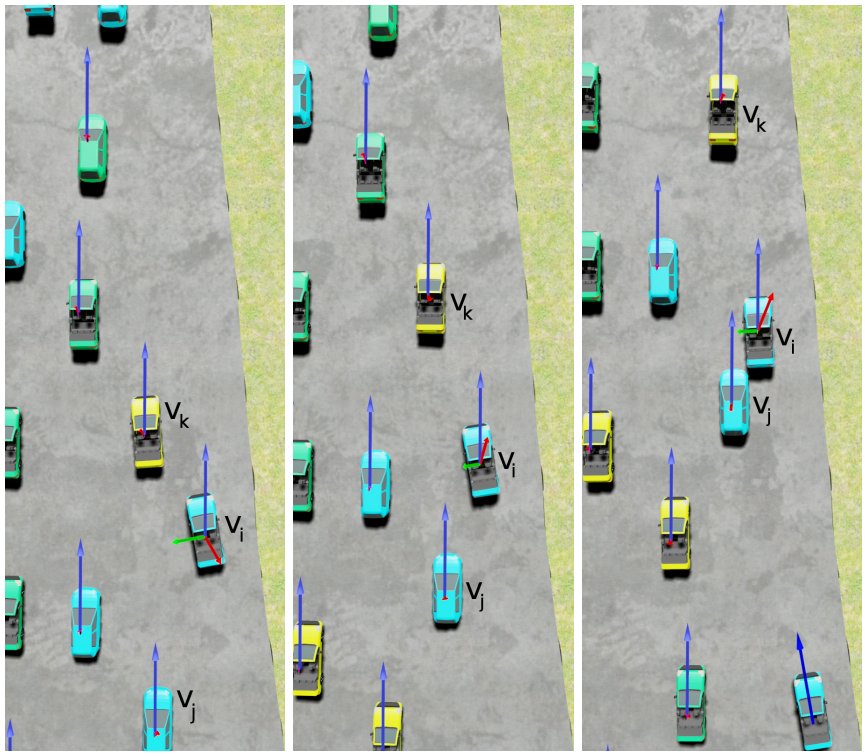
We would expect Experiment 2 to have more accidents than the baseline experiment and the result is hence reasonable.

5.1.3 Experiment 3: Oncoming and Merging Traffic

We regard Experiment 3 as an important experiment as it demonstrates how well our system works with only regular vehicles. If future vehicles should have equal priority and drive at equal speeds, Experiment 3 would be what resembles future road traffic the most.

The incidents in Experiment 3 start to happen after a throughput of 9000 vehicles per hour, similar to Experiment 2. An example of an incident that occurred in Experiment 3 is shown in Figure 5.3. A vehicle v_i is coming from the entrance ramp and trying to merge with the road traffic. In the merge process, v_i comes a little too close to v_k and starts braking to avoid collisions. As in the incident shown in Figure 5.2, v_i brakes too harshly and before it can start to accelerate again, v_i suffers a rear-end collision from v_j.

Compare the positions of v_i, v_j, and v_k between Figure 5.3a and 5.3b. You will see v_j and v_k moving forward at steady speeds, while v_i is clearly braking. The braking is also indicated by the read arrow (the Avoid Behavior) of v_i, illustrating v_i's desire to avoid v_k. In Figure 5.3b v_i have realized that it is no longer in danger of colliding with v_k, but is instead in danger of colliding with v_j and hence the red avoid arrow is pointing forward. If we again compare the



(a) (b) (c)

Figure 5.3: *An incident from Experiment 3. v_i brakes to harshly to avoid v_k and suffers a rear-end collision from v_j.*

positions of the three vehicles between Figure 5.3b and 5.3c, we can see that v_j and v_k have stable velocities and v_i is trying to accelerate, but is not doing so fast enough.

Figure 5.3 demonstrates the dynamics of the harsh braking and the PID controller's correction. In the vast majority of cases, the PID controller manages to correct the speed and avoid incidents. However, as we increase the throughput, and vehicles have less and less space to move, previously insignificant anomalies will start to have an impact.

Experiment 3 had more incidents than the baseline experiment, which it should, but a very similar amount of incidents compared to Experiment 2. One might expect the number of incidents to rise from Experiment 2 as we are adding merging traffic and hence causing more volatile traffic. However, the occurrence

of incidents is much due to misfortune as well as high throughput. Hence, the loess line gives a better indication of whether there are more incidents in Experiment 3 than Experiment 2. We can see the blue loess line having a higher y-value at throughput 13000 in Experiment 3 than Experiment 2, and hence the result is reasonable.

5.1.4 Experiment 4: Adding Buses

Experiment 4 also had its variations of the harsh-braking incident, see Appendix G for details. The result from Experiment 4 shows that more incidents occurred compared to Experiment 3. The bus is larger and occupies more road space than regular vehicles, it drives faster and causes more volatile traffic. The Avoid Prioritized Behavior forces regular vehicles to move away from the bus and squeezes the regular vehicles tighter together. Thus, we would expect more incidents when adding the bus to the experiments and the result shown in Figure 4.14 is very much what we would expect.

A side effect of buses is that a bus is three times the size of a regular vehicle, but only counts as one vehicle towards the throughput. Hence, a somewhat lower throughput before the loess line starts rising would be expected.

5.1.5 Experiment 5: Adding Emergency Vehicles

The emergency vehicles cause more incidents than the buses, as we would expect. Regular vehicles drive at 63km/h, buses at 72km/h, and emergency vehicles drive at 90km/h. The emergency vehicles are driving significantly faster than other vehicles and cause a much more volatile road traffic. The emergency vehicles are also free to roam the road wherever and are not bound by the KR Behavior, as buses are, to stay on the right-hand side of the road. When a regular vehicle enters the reserved area in front of a bus, it has to steer left to yield since the bus is already on the right and steering right would mean driving off the road. When a regular vehicle v_i enters the reserved area of an emergency vehicle, v_i will move out of the reserved area the fastest way possible. It will steer left or right depending on what is quickest. The emergency vehicles are spawned by the middle spawner (S_3) and will most likely keep driving in the midst of the road as other vehicles will move out of its path. Hence, regular vehicles may move both left and right and may interfere with both merging vehicles and oncoming vehicles. Experiment 5 has more accidents than Experiment 4, but still less than Experiment 6, which is what we would expect and the result is hence reasonable.

A variation of the harsh-braking incident that occurred in Experiment 5 is shown in Appendix G.

5.1.6 Experiment 6: Adding Emergency Vehicles and Buses

Experiment 6 is our most diversified experiment, all vehicle types are included. From Figure 4.16 we can see that it is, reasonably enough, also the experiment with the most incidents. Particularly one incident is interesting, the one incident that occurred already at a throughput of 2000 vehicles. Due to the incident occurring so early in the experiment we have termed it the *Kickoff Incident*. An overview of this incident can be seen in Figure 5.4,

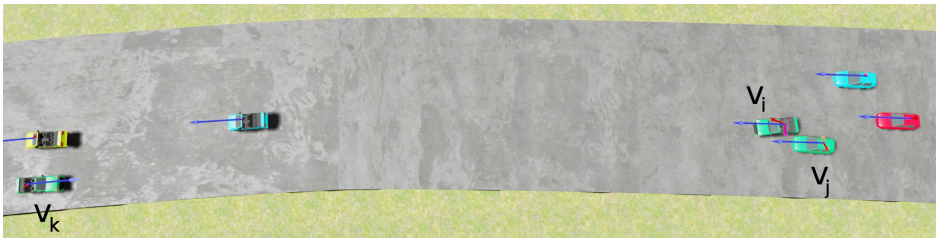


Figure 5.4: *An overview of the Kickoff Incident at throughput 2000 in Figure 4.16.*

In the Kickoff Incident two vehicles, v_i and v_j are driving side by side, v_i wants to steer left to avoid an emergency vehicle in the back. At the same time v_j wants to steer right to avoid an oncoming vehicle. The oncoming vehicle, v_k, can be seen if Figure 5.4 and a time-lapse of the events leading to the incident can be seen in Figure 5.5. Notice that the moment in time shown in Figure 5.4 is the same moment as in Figure 5.5d, and the moment in time shown in Figure 5.5a is the same is in Figure 5.6.

The Kickoff Incident is in many ways a perfect storm, v_i and v_j start out as almost perfectly aligned laterally (side by side). Hence, it will take multiple seconds before either vehicle will be able to position itself either in front or behind the other vehicle. There is an emergency vehicle coming from behind and an oncoming vehicle (v_k) from the other direction, both forcing v_i and v_j to align spatially. From Figure 5.5 we can see that v_i and v_j were about to make it. However, in the end the lateral effect of the AP Behavior overpowered the lateral effect of the Avoid Behavior and v_i hit v_j. A possible solution that could have prevented the Kickoff Incident is to divide the Avoid Behavior into a lateral part and a spatial part. The lateral part would then have to account for other behaviors threatening to overpower it. However, splitting the Avoid Behavior into multiple parts would require fundamental changes to our model and is hence regarded as future work.

In Figure 5.6 the blue arrows is the Road Tangent Behavior urging the vehicles to drive forward, the red arrows is the Avoid Behavior urging v_i to avoid v_j and

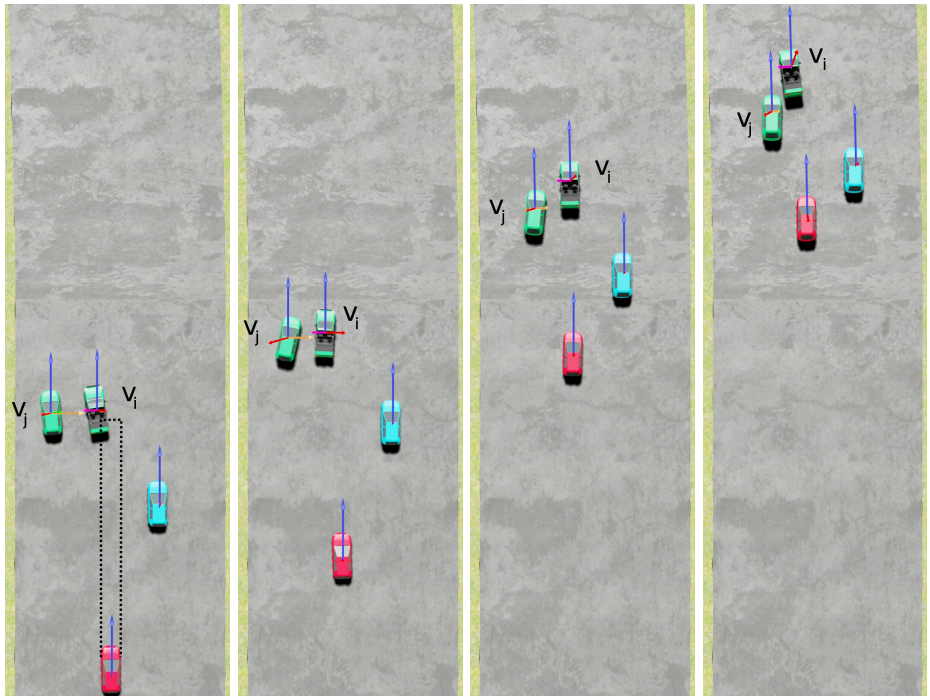


Figure 5.5: *A time-lapse of the events leading to the Kickoff Incident in Figure 4.16. In Figure 5.5a v_j wants to steer right to avoid an oncoming vehicle and v_i wants to steer left to avoid the reserved area of an emergency vehicle (Figure 5.6 shows details of this scenario). v_j starts steering right and v_i is torn between steering right to avoid v_j and steering left to avoid the reserved area and ends up steering straight ahead (Figure 5.5b). v_i is from the start located marginally farther ahead than v_j and hence will increase the speed slightly to avoiding v_j (Figure 5.5c). Once v_i is far enough in front of v_j the lateral effect of the AP Behavior will start to overpower the lateral effect of the Avoid Behavior and v_i will hit v_j (Figure 5.5d).*

likewise. The direction an arrow is pointing corresponds to the direction the behavior wants to steer, and the magnitude of the arrow corresponds to the force in the behavior. The pink arrow is the AP Behavior urging v_i to avoid the reserved area of an emergency vehicle coming from behind. The brown arrow is the AO Behavior urging v_j to avoid an oncoming vehicle. The green arrow is the KIR Behavior urging v_j to stay on the road. Recall that the KIR Behavior evaluates a point in front of v_j to determine if the vehicle will steer out of the road

in the future. This is why the KIR Behavior is actively urging v_j to steer right even though v_j is clearly not in danger of driving off the road yet. The moment in time shown in Figure 5.6 is the same as in Figure 5.5a, and from Figure 5.5a it is easier to see that both v_i and v_j are steering to the left. They are both steering to the left because v_i came inside the reserved area slightly before v_j was inside the range of the AO Behavior. Hence v_i started steering left, v_j followed to avoid v_i. The KIR Behavior noticed that v_j would be off the road if the course was kept and started urging to steer right and then the moment in Figure 5.6 occurred. To determine the how the v_i and v_j will steer from Figure 5.6, we can summarize all the arrows (which are vectors with direction and magnitude). Both vehicles will keep driving forward, as the blue arrow is large, and there is no arrow pointing backward to cancel it out. v_i will be urged left by the pink arrow and right by the red arrow and the result will be v_i driving approximately straight ahead, as can be seen in Figure 5.5b. v_j will have the red and green arrow approximately cancel each other out. The result will be the sizable brown arrow deciding on steering that will be to the right, as can also be seen in Figure 5.5b.

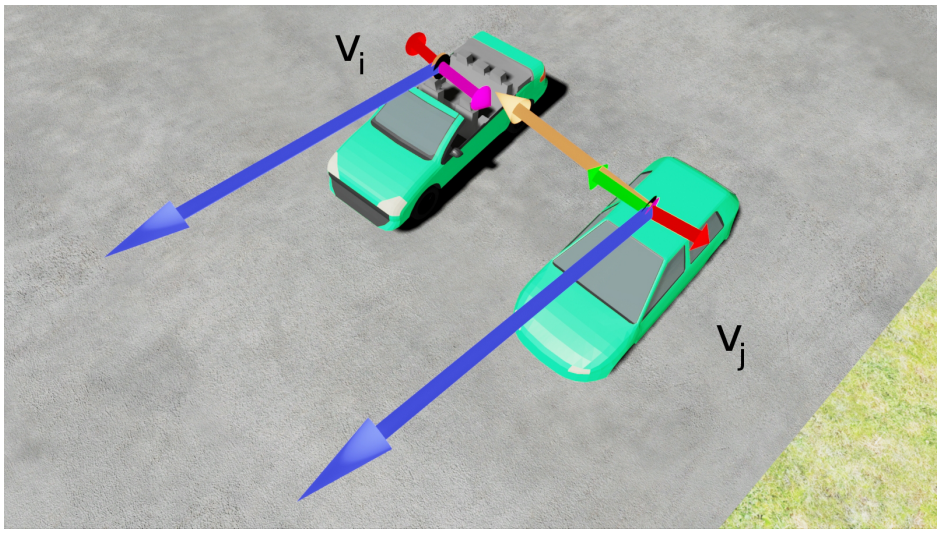


Figure 5.6: *A detailed illustration of the scenario in figure 5.5a. Multiple steering behaviors are involved in the situation, and the scenario is well suited for a discussion on complex behavior cooperation. See the text for details.*

We attribute the occurrence of the Kickoff Incident at a throughput of only 2000 vehicles per hour to pure misfortune. The incident may as well have happened at any throughput, and the chances increase with higher throughputs.

However, as stated before, the occurrence of incidents is much due to statistical chance and from the results we can see that incidents at 2000 vehicles per hour are clearly not frequent.

5.1.7 Experiment 7: Symmetric Road Traffic

The experiment with symmetric road traffic is designed as a control experiment to investigate how well our steering behaviors would perform when managing symmetric road traffic. Experiment 7 contains only regular vehicles and can hence be viewed as a variation of Experiment 3 with symmetric traffic load. Experiment 3 managed a throughput of 9000 vehicles before any incidents and a throughput of 11000 before the loess line started to rise. In comparison, Experiment 7 managed a total throughput of 10000 vehicles before any incidents, and the loess line also began to increase at 10000. We would expect Experiment 7 to perform similarly to Experiment 3, which it did, and we hence regard the result as reasonable.

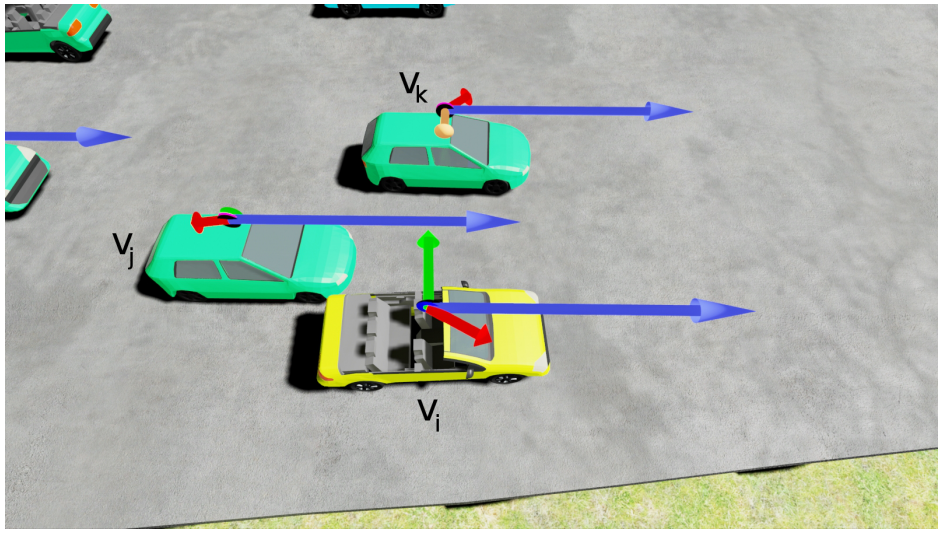


Figure 5.7: *An example of an alignment incident that occurred in Experiment 7. v_k wants to steer right to avoid an oncoming vehicle, v_j is being squeezed in the middle and wants to brake, and v_i is merging and wants to steer left to enter the road. The result for v_i is that the lateral force of the KIR Behavior (green arrow) is stronger than the lateral force of the Avoid Behavior and v_i drives into v_j.*

Figure 5.7 shows an example of an incident that occurred in Experiment 7. The incident occurred due to the same alignment problem as the Kickoff Incident.

An example of a harsh-braking incident that occurred in Experiment 7 can be seen in Appendix G.

We are pleased to see steering behaviors being able to govern symmetric road traffic as well as asymmetric. Being able to handle symmetric road traffic is a requirement for flocking to be a viable option for future road traffic management. The reasons for why our model can manage higher traffic loads than real-world symmetric traffic, illustrated by Figure 1.3 is probably multiple. One reason is that automatic vehicles can drive closer to the vehicle in front and hence we can have more vehicles per meter of the road. Another is due to the removal of the middle lane barrier, which widens the road with almost an extra lane.

The reason traffic breaks down between a measured throughput of 8000 and 9000 (a total throughput of 16000 and 18000), is that we remove vehicles involved in incidents. The removed vehicles are not counted as throughput. Hence, as more and more vehicles collide, more and more vehicles are removed, and the throughput stops increasing.

A limitation of Experiment 7 is that only regular vehicles were used. For a future project, it could be interesting to investigate if throughputs similar to what were in experiments 4, 5, and 6 could be achieved with symmetric road traffic as well.

5.2 The Big Picture

In this section, we take a step back and evaluate the general tendencies in our experiments. We discuss how the results can be interpreted and what implications the results may have for future road traffic.

5.2.1 The Essence of Our Results

First off, our results show that flocking is a viable option for traffic management, they also show that throughput during rush hours can be significantly improved. Figure 1.3 in the introduction showed that real-world road traffic reaches a throughput of 1600 vehicles per hour per lane before the speed drops and a jam develops. For a road with two lanes in each direction (the road type our road segment is modeled after), a throughput of $1600 * 2 = 3200$ could be achieved in each direction. Experiment 7 showed that we were able to manage a symmetric throughput of 5000 vehicles in each direction with only regular vehicles. Experiment 6 with all vehicle types were able to manage an asymmetric throughput of 5500 before incidents became frequent.

Through this project we have used automatic vehicles, automatic vehicles can react faster than human drivers and hence need less space between vehicles. We believe that a portion of the throughput increase is probably due to automatic

vehicles and not solely the success of flocking. Especially in Experiment 7 there were an equal number of vehicles in each direction, and the throughput reached 10000 vehicles per hour. If we assume that today's roads are capable of 1600 vehicles per hour per lane, a road with a total of four lanes would manage $1600 * 4 = 6400$ vehicles per hour, which means that flocking with symmetric traffic and automatic vehicles would create a throughput increase of $\approx 56\%$. For the asymmetric Experiment 6, the throughput increase from 3200 to 5500 vehicles is an increase of $\approx 72\%$.

We observe the emergent behavior causing two flows of traffic to appear, both flows drive on their right-hand side. Buses can maintain their position on the right-hand side, and emergency vehicles are generally able to overtake other vehicles while not causing incidents as long as the throughput stays below 5500 vehicles per hour. When traffic load is significantly higher in one direction, as in Experiment 2 - 6, we can observe that the emergent behavior functions as intended by letting the high-load flow claim the most of the road. In Experiment 7, with symmetric load, both flows claims approximately equal shares.

Spawner S_7 have been spawning oncoming vehicles through Experiment 1 - 6, but the vehicles have not been counted towards the throughput. The interval between spawned vehicles has been 25 seconds on average. 25 seconds between each vehicle would have given a throughput of 144 vehicles per hour. If the interval were reduced to 3 seconds, the throughput would have been $3600/3 = 1200$ vehicles per hour. Hence Experiment 4 - 6 may have managed a higher total throughput if S_7 had a lower spawn interval and it could be interesting to see in a future experiment.

We chose to have 25 seconds between spawned vehicles from S_7 because 25-second intervals will let vehicles spawned from S_{1-6} use the full width of the road. The emergent behavior will dynamically allocate the extra space when available and needed. We can observe that vehicles from S_{1-6} are using the full width of the road between vehicles from S_7 and we regard this feature as a significant advantage of flocking over traditional lanes.

From running the experiments we observe three fundamental types of incidents that all other incidents are variations of:

Unstable Spawning Incidents
: The unstable spawning incidents are discussed in Section 5.1.1 and we argue why they have not been removed. A key element with spawning incidents is that they are only an issue for computer simulations and do not exist in the real world.

Harsh-Braking Incidents
: The harsh-braking incident is described in Section 5.1.2 and 5.1.3 in addition to multiple scenarios in Appendix G.

Lateral Alignment Incidents

The lateral alignment incident is described in Section 5.1.6 (the Kickoff Incident) and Section 5.1.7.

Regarding the CAM standard, we implemented an extra field (the `GoalSpeed`) which is used only when vehicles are waiting on the entrance ramp to determine if it is safe to enter the road. Hence, all vehicles are able to navigate on the road without the extra field, and we regard the employment of CAMs as a success.

5.2.2 Implications for Future Road Traffic

If flocking is chosen as the future road traffic management system, roads could be built without lanes or physical lane barriers. Hence reducing the cost of future infrastructure. Road space can be allocated in the direction which gives the most efficient traffic flow. There is no additional management system required to direct when more of the road should be used in either direction, and allocation will happen automatically as an inherent part of the emergent behavior.

Flocking can allow a variety of vehicles to drive at different speeds. Hence both buses and emergency vehicles can be given priority and allowed to drive faster than regular vehicles. The CAM protocol also allows vehicles to notify other vehicles about special conditions as e.g. oversized load. Securing the ITS communications will be important, and the standard is already designed to support signing with digital certificates [ETSI, 2014].

For this project, we have assumed reliable communications (see Section 1.5). Reliable communication is a vital part of our experiments as the CAMs are the only source of information to a vehicle's position. If a number of CAMs from vehicle v_i were to be lost, a neighboring vehicle v_j will try to predict the future position of v_i based on the speed and heading in the last successfully received CAM. The predictions will be accurate till v_i makes a turn or changes speed. Hence, frequency of CAM transmission may be adjusted only to occur when a vehicle turns or changes speed. The frequency may also be adjusted up for vehicles at high speed. In the real world, CAMs may be lost. An emergency vehicle, driving 90km/h (25m/s) moves 2.5m between each CAM. If one were lost, the emergency vehicle would drive 5m before neighboring vehicles would be updated on its position. If two messages were lost the distance would be 7.5m, and so on. The loss of one CAM per second would constitute a packet loss of 10%. An emergency vehicle is 4.5m long, and a packet loss of 10% could be tolerable at stable driving on the highway. The effects of packet loss could be interesting to investigate in a future project (see Section 6.3). Till then, we believe it is important that automatic vehicles have redundant means of determining other vehicles' location as packet loss may occur in the real world.

The `GoalSpeed` field which we added to the CAMs is not necessary for driving on the road, but could be useful in enabling future vehicles to make accurate predictions for how the road traffic will develop. A field that informed other vehicles about how fast a vehicle *intends* to drive would allow for more efficient positioning as positioning-algorithms then would be able to account for not only current, but also future events.

An interesting quality of using GPS, lasers, or radar for positioning is that no light is required. Street lights will probably still be necessary due to humans or animals straying into the road, but automatic vehicles could remove the dependency on light to determine the road's location.

This project is intended as a concept study to test the viability of flocking, but some assessment of what future throughputs can be expected, may be appropriate. If future vehicles have the same dimension and speed, a throughput of at least 9000 vehicles per hour appears possible on a 20m wide road. If vehicles with varying dimensions and speeds are used, a throughput of at least 5500 vehicles per hour appears possible. However, with mixed vehicle types, expected throughput may be somewhat higher as one of our spawners waited 25 seconds between each spawning, and smaller interval may have increased the throughput without increasing the number of incidents. This scenario could be further explored in a future project (see Section 6.3). Our results show that the chance of incidents increases as the number of vehicles on the road increase.

A fundamental requirement for flocking to be possible is that the road is wide enough for more than one vehicle otherwise there will be no flocking, only platooning.

The authors have determined the speed limits in the simulations (63km/h for regular vehicles, 72km/h for buses, and 90km/h for emergency vehicles). In the real world, the limits may be determined by a central agency, such as the NPRA. The speed limit may be updated dynamically by local driving conditions through ITS communications. We can see from the experiments that vehicles of equal dimensions, driving at equal speed, gives higher throughput than mixed traffic. A possibility could be to have slow vehicles keep to one side to let faster vehicles pass, or have a minimum speed requirement that vehicles need to satisfy to be allowed on the highway.

We have chosen to weigh the behaviors for remaining on the road (KIR) and avoiding head-on collisions (AO) higher than the general Avoid Behavior. There are two major consequences: Firstly, a vehicle will rather suffer incidents with neighboring vehicles than blocking the path for an oncoming vehicle. Secondly, a vehicle will rather suffer a collision with a vehicle than moving off the road to yield. We have chosen to have vehicles collide with neighbors, driving the same direction at a similar speed, rather than an oncoming vehicle. The energy in bumping into a neighboring vehicle will probably be much less than colliding

head-on, and hence the chance of fatality or severe injury will be less. Driving off the road to avoid incidents may sometimes prove the best option. In our simulation, we chose to make a priority out of having vehicles remain on the road and rather improve the steering behaviors to avoid incidents. For a real-life flocking algorithm, it may sometimes be a better option to drive off the road and damage the vehicle, but avoid casualties.

An advantage of flocking is that it is still possible to use the intersection management described in [Dresner and Stone, 2005], the flocks will have to negotiate as a whole though, and not as individual vehicles [Astengo-Noguez and Sánchez-Ante, 2007]. Vehicular ad-hoc networks can be used for flock communications or internal flock communications [Agarwal and Little, 2009]. In our experiment we have used *leaderless flocking* as opposed to *leader-follower flocking*. In future road traffic, it is possible that it will be more efficient to negotiate a flock leader and have other vehicles follow, as in platooning.

In our experiments, there are no possibilities for pedestrians, moose, deers, or other obstructions on the road. Future road traffic will have to account either for such obstructions by adapting or preventing them from arising in the first place. A possible scenario is that pedestrians are refused on the highway and extensive usage of wildlife fences.

Using UE to create road traffic simulation have been both fun and practical. The already implemented PhysX vehicle, gravity, and lighting ensures that simulations can be created quickly with realistic results. UE is free for academic usage.[1] We believe that game engines may prove a valuable tool for future road traffic simulations and experiments.

[1]See the UE web page for specifics regarding licensing. `https://www.unrealengine.com/blog/ue4-is-free`

Chapter 6

Conclusion

6.1 The Research Goals and Questions

6.1.1 The Research Goal

Our research goal was to: *Investigate the viability of automatic vehicles in combination with steering behaviors to increase road traffic efficiency.* Through experiments we have simulated road traffic scenarios with automatic vehicles and steering behaviors and measured throughput. The results show that flocking is a viable option for future road traffic management, and that throughput can be increased. With these conclusions, we consider our research goal to be fulfilled.

6.1.2 Research Question 1: Avoiding Collisions

The first research question was: *Can steering behaviors ensure that vehicles driving in the same or opposite direction do not collide?* The question was answered by Experiment 2, which showed that steering behaviors can be used to prevent collisions between vehicles driving either direction up to a throughput of 9000 vehicles per hour.

6.1.3 Research Question 2: Merging Road Traffic

Our second research question was: *Can steering behaviors effectively manage the merging of traffic from entrance ramps into the highway road traffic?* Experiment 3 answered this question and showed that steering behaviors are capable of merging entering vehicles into the road traffic without incidents up to a throughput of 9000 vehicles per hour.

6.1.4 Research Question 3: Prioritizing

The third of our research questions was: *Can steering behaviors be used to prioritize emergency vehicles and public transport (buses) over regular traffic?* Experiment 4 - 6 demonstrated how buses and emergency vehicles can be prioritized by keeping a higher speed than the regular vehicles. Experiment 4 - 6 also demonstrated the viability of flocking with vehicles of different sizes driving at various speeds, and showed that a throughput of 5500 vehicles per hour should be manageable.

6.1.5 Research Question 4: Throughput Comparisons

Research Question 4 was: *How does the vehicle throughput of road traffic systems utilizing flocking compare to regular traffic systems using lanes?* All experiments were conducted on a 20m wide road, which would correspond to a total of four lanes in the real world. Four lanes would be capable of $1600 * 4 = 6400$ vehicles per hour during symmetric traffic, and two lanes would manage $1600 * 2 = 3200$ vehicles per hour during asymmetric traffic (rush hours).

Our experiments with asymmetric traffic have shown that a throughput of 9000 vehicles per hour can be possible with vehicles of equal size driving at equal speed and a throughput of 5500 vehicles per hour with mixed traffic. Experiment 6, which we regard as the most realistic experiment, managed a throughput increase of $\approx 72\%$ from 3200 to 5500 vehicles per hour with asymmetric mixed-traffic. Experiment 7 showed that a throughput of 5000 vehicles per hour in each direction is possible with symmetric traffic.

Hence, we regard flocking to be an efficiency improvement over the current traffic management system using lanes. However, some of the throughput increase is probably due to automatic vehicles needing less road space than human drivers and not only due to flocking allowing for more dynamic allocation of road space.

A general tendency in our experiments, which also sounds reasonable for the real world, is that higher throughputs increase the chance of incidents.

6.1.6 Research Question 5: CAMs

The fifth research question was: *Will the future communication standard of CAMs [ETSI, 2014] be able to satisfy the requirements of safely guiding automatic vehicles without reducing the overall efficiency of the road traffic?* The CAMs have been used for vehicular communications in every experiment and enabled the vehicles to determine the position of other vehicles with high accuracy. Each vehicle is transmitting a CAM 10 times per second, and we have found such a message frequency sufficient for avoiding incidents. We have not observed any incidents due to CAMs, nor have we detected any decline in vehicle throughput.

Hence, we conclude that the future communication standard of CAMs is capable of safely guiding automatic vehicles without reducing efficiency.

We have implemented an extra field in the CAMs that are used for transmitting a vehicle's *intended* future speed. The field is only used when waiting on the entrance ramp to determine when it is safe to merge with the existing traffic. However, we believe future road traffic could benefit from such a field as it may allow automatic vehicles' positioning algorithms to account for future events in addition to current ones.

6.2 Contributions

We have summarized the major contributions of this project in the list below.

Demonstrating Viability of Flocking
The major contribution of the work in this project has been to demonstrate the viability of flocking as a future road traffic management system.

Steering Behaviors Possible for Mixed Traffic
We have shown that steering behaviors can work for diversified road traffic with vehicles of different sizes driving at multiple different speeds. In nature, flocks are usually comprised of animals of the same species and hence it is not natural that applying the same flock behaviors to a set of multiple different vehicles will have the desired effects.

What Throughput can be Expected With Flocking
On a 20m wide road, asymmetric throughput can increase $\approx 72\%$ from 3200 to 5500 vehicles per hour for mixed traffic.

Identify Requirements for Real-World Implementations
Through this project, we have identified some requirements that we believe will be necessary for a real-world implementation of flocking. Firstly, automatic vehicles will be required as having every human driver adhere to the same set of steering behaviors is not realistic. The automatic vehicles must be capable of determining their location with high accuracy (by e.g. improved GPS) and also be aware of the neighboring vehicles (by e.g. CAMs).

6.3 Future Work

During this project, we have identified some topics that could form the foundation for future research projects. We have listed the topics below.

Improving the Simulation

How many incidents could be removed by splitting the Avoid Behavior (and possibly other behaviors as well) into a lateral and spatial part? How many incidents could be prevented by fine tuning the PID controller to avoid harsh braking? In Section 5.2.1 we showed that there are only three causes of incidents, and only two of them can occur in the real world. Implementing stable spawning could prove helpful to remove the spawning incidents from the statistics and ease debugging after the real problems. In general, it would be interesting to see if we could further prevent incidents by improving the simulation.

Who Determines Future Flocking Parameters?

In this project, the authors decided the parameters of all behaviors. Who should do it in the real world? How far ahead should the AO Behavior account for? A possible solution is to let the NPRA decide on parameters. Chances are that not all vehicles will have the same acceleration, braking capacities, or max speed. Hence vehicle segmentation, where similar vehicles have similar parameters, may be an option.

Implementing More Features

Exactly how would a roundabout function with flocking? Would there even be roundabouts? What about intersections? Dresner and Stone showed an efficient intersection management system adapted to lanes [Dresner and Stone, 2005]. The same system could work for flocks of vehicles as well, but the flock would have to negotiate as a whole, possibly by a *flock spirit* [Astengo-Noguez and Sánchez-Ante, 2007]. It would be interesting to see an implementation of Dresner and Stone's intersection management in combination with flocking and flock spirits. Another feature that we would like to see implemented is emergency braking. Currently, the simulated vehicles are not capable of emergency braking, and we believe more realistic simulations could be achieved with such a feature. UE have an implementation of behavior trees [Games, 2015; Dromey, 2007] which could be suitable for such a project and possibly ease the workload.

Cohesion for Similar Vehicle Types

In nature, flocks are usually comprised of individuals of the same species. Hence, the flock is homogeneous, and all individuals move with approximately the same speed and have roughly the same size. The same concepts could be applied to the Cohesion Behavior for vehicles and only let vehicles with similar properties drive together as a flock. Adapting the Cohesion Behavior to separate between vehicles and create multiple different flocks could increase efficiency as similar vehicles could flock together and require

less road space. Instead of regular vehicles trying to flock with buses and emergency vehicles, which will eventually overtake the flock.

Investigate Symmetric Throughput with Mixed Traffic
So far, the symmetric road traffic experiments have only used similar vehicles. It would be interesting to see how the throughput would change, or if it changes, for symmetric traffic by also letting buses and emergency vehicles be part of the traffic.

Spawning Intervals for Higher Throughput
The S_7 spawner, which was active through experiments 2 - 7, had an average interval of 25 seconds between each spawned vehicle. The high interval gave vehicles spawned from S_{1-6} more space to demonstrate flocking, but spawning vehicles more often may increase the total throughput. In all of our experiments we only measured throughput in one direction, it would be interesting to measure throughput in both directions to investigate the effects of flocking on the total road throughput.

Investigate Tolerable CAM Loss Rates
A fundamental assumption for our project was reliable communication (Assumption 2). CAMs were transmitted ten times per second and an emergency vehicle, traveling at 90km/h, would move 2.5m between each CAM. What loss rates of CAMs could be tolerated before incidents would start to occur? Should CAMs be transmitted more often when driving at higher speeds, turning, or accelerating?

Have Peak Car Been Reached?
Our background research for this thesis revealed that peak car may have been reached in the US, i.e. the number of miles traveled by vehicles have started to decline [Pyper and ClimateWire, 2012]. If peak car has been reached in the US, it may soon also be reached in Norway. What implications would peak car bring? Could we reduce the amount of infrastructure needed for vehicles? A research project to determine future consequences of fewer vehicles could prove interesting.

Future Role of Public Transport
Our experiments demonstrate that public transport can be prioritized over regular vehicles. According to Beirão and Cabral, public transport should be designed to accommodate the level of service required by customers, to attract more users [Beirão and Cabral, 2007]. The choice of transport is influenced by several factors, such as individual characteristics and lifestyle, the type of journey, the perceived service performance of each transport mode and situational variables. The diverse factors involved in choosing

transport, suggests the need for segmentation taking into account travel attitudes and behaviors. Policies that aim to influence car usage should be targeted at the market segments that are most motivated to change and willing to reduce the frequency of car use. However, 6.3% of car users are passionate drivers and will not convert to public transport simply by improving the public transport system [Jensen, 1999]. A research project could be committed to determining what can be done to improve public transport and increase the appeal of public transport.

Legal Issues of Automatic Vehicles

The need for human drivers disappears with automatic vehicles. Optimistic estimates claim it will be illegal for humans to drive their own cars in 15 years [DN, 2015]. Who should be allowed to drive an automatic car? Can children drive? What about drunk people? Should a drivers license be required to drive an automatic car?

If a collision is unavoidable, should the car run over a girl or a grandma? Should we start to make lists of which citizens have the highest priority? There are many legal and ethical issues that need to be resolved as automatic vehicles begin to populate the streets, and multiple research projects are probably necessary to determine what laws would benefit society the most.

Diversity in Future Automatic Vehicles

There may be multiple different producers of automatic vehicles. Hence, there may be multiple different configurations of, e.g., braking force, tire friction, or acceleration. Ideally automatic vehicles should drive close to each other to generate a high throughput, but that might not be safe for all vehicles. Should there be minimum requirements for allowing an automatic vehicle on the road? Should vehicles know their own limits and maintain their own safety margins? Should there be one or multiple standards of vehicle configurations that automatic vehicles could be constructed after? More conformity could improve efficiency as vehicles would be able to drive in closed order formation with small gaps. However, closed formation would require unison in specifying safety critical functions. How should automatic vehicles adapt to diversity?

Efficiency Gains by Transmitting Intentions?

The `GoalSpeed` field enables automatic vehicles to account for future actions of neighboring vehicles within CAM range. The field could enable more effective positioning on the road and increased throughput. What efficiency improvement could be expected? What intentions should be transmitted?

With what confidence could predictions be done? There are many questions that could prove interesting regarding transmittance of vehicle intentions.

Bibliography

Agarwal, A. and Little, T. (2009). Impact of asymmetric traffic densities on delay tolerant vehicular ad hoc networks. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1–8.

Antsaklis, P., Passino, K., and Wang, S. (1990). An introduction to autonomous control systems. In *Intelligent Control, 1990. Proceedings., 5th IEEE International Symposium on*, pages 21–26 vol.1.

ARINC Incorporated (2008). Ndgps assessment final report. `http://ntl.bts.gov/lib/31000/31300/31338/23_2008_NDGPS_Assessment_Report_Final.pdf`. Accessed: 04.05.2015.

Arkin, R. (1987). Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 264–271.

Arkin, R. C. (1989). Motor schema—based mobile robot navigation. *The International journal of robotics research*, 8(4):92–112.

Arkin, R. C. (1992). Behavior-based robot navigation for extended domains. *Adapt. Behav.*, 1(2):201–225.

ARTBA (2014). How many miles of roads are there in the u.s.? American Road & Transportation Builders Association, `http://www.artba.org/about/transportation-faqs/#9`. Accessed: 27.02.2015.

Astengo-Noguez, C. and Sánchez-Ante, G. (2007). Collective methods on flock traffic navigation based on negotiation. In Gelbukh, A. and Kuri Morales, ., editors, *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 52–60. Springer Berlin Heidelberg.

Atiyeh, C. (2014). Toyota—of all companies—defends drivers, says it won't build a fully autonomous car. Car and Driver, `http:`

//blog.caranddriver.com/toyota-defends-drivers-says-it-wont-build-a-fully-self-driving-car/. Accessed: 26.02.2015.

Avinor and Transportetatene (2013). Meld. st. 26 (2012–2013) nasjonal transportplan 2014–2023. https://www.regjeringen.no/nb/dokumenter/meld-st-26-20122013/id722102/?docId=STM201220130026000DDDEPIS&ch=1&q=. Accessed: 20.02.2015.

Barth, M. and Boriboonsomsin, K. (2008). Real-world co2 impacts of traffic congestion. http://www.uctc.net/research/papers/846.pdf. Accessed: 16.04.2015.

Bates, J., Loyall, A. B., and Reilly, W. S. (1994). *An architecture for action, emotion, and social behavior*. Springer.

BBC (2011). Driverless car: Google awarded us patent for technology. http://www.bbc.com/news/technology-16197664. Accessed: 16.02.2015. US Patent number: 8,078,349. Patent link http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=8,078,349.PN.&OS=PN/8,078,349&RS=PN/8,078,349.

BBC (2012). Google gets nevada driving licence for self-drive car. http://www.bbc.com/news/technology-17989553. Accessed: 16.02.2015.

BBC (2014). Uk to allow driverless cars on public roads in january. http://www.bbc.com/news/technology-28551069. Accessed: 23.02.2015.

Beer, R. D. (1990). Intelligence as adaptive behavior: an experiment in computational neuroethology.

Beirão, G. and Cabral, J. S. (2007). Understanding attitudes towards public transport and private car: A qualitative study. *Transport Policy*, 14(6):478 – 489.

Bennett, S. (1984). Nicholas minorsky and the automatic steering of ships. *Control Systems Magazine, IEEE*, 4(4):10–15.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Bhatia, P. (2003). Vehicle technologies to improve performance and safety. *University of California Transportation Center*.

Blumberg, B. (1994). Action-selection in hamsterdam: Lessons from ethology. In *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats*, volume 3, pages 108–117.

Blumberg, B. M. and Galyean, T. A. (1995). Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 47–54, New York, NY, USA. ACM.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23.

Bruderlin, A. and Calvert, T. W. (1989). Goal-directed, dynamic animation of human walking. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 233–242, New York, NY, USA. ACM.

California Senate (2012). Senate bill no. 1298. The California Senate, `http://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201120120SB1298`. Accessed: 16.02.2015.

Carpin, S., Lewis, M., Wang, J., Balakirsky, S., and Scrapper, C. (2007). Usarsim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405.

Chapuis, R., Laneurit, J., Aufrere, R., Chausse, F., and Chateau, T. (2002). Accurate vision based road tracker. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 666–671 vol.2.

Clark, S. and Durrant-Whyte, H. (1998). Autonomous land vehicle navigation using millimeter wave radar. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3697–3702 vol.4.

Cleveland, W. S., Grosse, E., and Shyu, W. M. (1992). Local regression models. *Statistical models in S*, pages 309–376.

Cliff, D. and Miller, G. F. (1996). *Co-evolution of Pursuit and Evasion II: Simulation Methods and Results*. From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96). Editors: Maes, Mataric, Meyer, Pollack, and Wilson.

Cohen, R. (2015). New golden gate bridge barrier draws sighs of relief. `http://www.nytimes.com/2015/01/12/us/new-golden-gate-bridge-barrier-draws-sighs-of-relief.html?_r=2`. Accessed: 9.04.2015.

Costa, M., Feijó, B., and Schwabe, D. (1990). Reactive agents in behavioral animation. *Anais do SIBGRAPI*, 95:159–165.

Cremer, J., Kearney, J., and Willemsen, P. (1996). A directable vehicle behavior model for virtual driving environments. In *In Proceedings of 1996 Conference on AI, Simulation, and Planning in High Autonomy Systems, La Jolla, CA*, pages 18–25.

CryEngine (2015). Cryengine | the complete solution for next generation game development by crytek. `http://www.cryengine.com/`. Accessed: 11.03.2015.

Cui, Y. and Ge, S. S. (2003). Autonomous vehicle positioning with gps in urban canyon environments. *Robotics and Automation, IEEE Transactions on*, 19(1):15–25.

Davies, A. (2015). Mercedes reinvents the car with a bonkers self-driving concept. Wired, `http://www.wired.com/2015/01/mercedes-benz-f-015-concept/`. Accessed: 25.02.2015.

Davila, A., Aramburu, E., and Freixas, A. (2013). Making the best out of aerodynamics: Platoons. Technical report, SAE Technical Paper.

DN (2015). Dansk bilbransjeråd: Snart er det forbudt å kjøre sin egen bil. Dagens Næringsliv, `http://www.dn.no/privat/dnBil/2015/01/14/0900/Bilsalg/dansk-bilbransjerd-snart-er-det-forbudt--kjre-sin-egen-bil`. Accessed: 25.05.2015.

Douglas, J. and Schafer, T. (1971). The chrysler "sure-brake"-the first production four-wheel anti-skid system. Technical report, SAE Technical Paper.

Dresner, K. and Stone, P. (2005). Multiagent traffic management: An improved intersection control mechanism. In *In The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477.

Dresner, K. and Stone, P. (2006). Human-usable and emergency vehicle-aware control policies for autonomous intersection management. In *Fourth International Workshop on Agents in Traffic and Transportation (ATT), Hakodate, Japan*.

Dromey, R. (2007). Principles for engineering large-scale software intensive systems. In *Slide Presentation for the 17th Australian Conference on Software Engineering*.

Duhigg, C. and Lohr, S. (2012). The patent, used as a sword. `http://www.nytimes.com/2012/10/08/technology/patent-wars-among-`

tech-giants-can-stifle-competition.html?pagewanted=all. Accessed: 18.02.2015.

ECE (2014). Consistency between the convention on road traffic (1968) and vehicle technical regulations. Economic Commission for Europe, http://www.unece.org/fileadmin/DAM/trans/doc/2014/wp1/ECE-TRANS-WP1-2014-1e.pdf. Accessed: 23.02.2015.

Engan, Ø., Johannessen, N., and Mjaaland, O. (2015). Samferdselsministeren: Bekrefter den triste situasjonen. VG, http://www.vg.no/nyheter/innenriks/norsk-politikk/samferdselsministeren-bekrefter-den-triste-situasjonen/a/23415574/. Accessed: 9.04.2015.

Epic Games (1991). Epic games. http://epicgames.com/. Accessed: 10.03.2015.

Epic Games (1998). The official unreal home page. http://web.archive.org/web/19980127121521/http://www.unreal.com/unreal_world_nc.html. Accessed: 10.03.2015. The official web page was archived in 1998, see also http://www.mobygames.com/game/game/unreal.

Epic Games (2006). Gears of war - home. http://gearsofwar.xbox.com/en-US/. Accessed: 10.03.2015.

Epic Games (2015a). Game development tools and gaming engine for game developers | unreal engine 4. https://www.unrealengine.com/what-is-unreal-engine-4. Accessed: 19.01.2015.

Epic Games (2015b). What is github? https://www.unrealengine.com/ue4-on-github. Accessed: 11.03.2015.

ERTRAC (2015). Automated driving roadmap - status: 3rd draft for public consultation. European Road Transport Research Advisory Council, http://www.ertrac.org/uploads/documentsearch/id35/ERTRAC_Automated-Driving_draft3-web.pdf. Accessed: 20.04.2015.

ETSI (1988). Etsi - european telecommunications standards institute. The European Telecommunications Standards Institute, http://www.etsi.org/. Accessed: 11.02.2015.

ETSI (2010). Etsi en 302 665 v1.1.1 (2010-09). The European Telecommunications Standards Institute, http://www.etsi.org/deliver/etsi_en/302600_302699/302665/01.01.01_60/en_302665v010101p.pdf. Accessed: 12.02.2015.

ETSI (2011). Final draft etsi es 202 663 v1.1.0 (2009-11). The European Telecommunications Standards Institute, http://www.etsi.org/deliver/etsi_es/202600_202699/202663/01.01.00_50/es_202663v010100m.pdf. Accessed: 24.02.2015.

ETSI (2014). Etsi en 302 637-2 v1.3.2 (2014-11). The European Telecommunications Standards Institute, http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf. Accessed: 11.02.2015.

EU Parliament and Counsil (2010). On the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport. The European Parliament and The Council of The European Union, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF. Accessed: 11.02.2015.

Farr, G. (1990). Traction control system. https://www.google.com/patents/US4966248. US Patent 4,966,248.

Florida Senate (2012a). Cs/hb 1207: Vehicles with autonomous technology. The Florida Senate, http://www.flsenate.gov/Session/Bill/2012/1207. Accessed: 16.02.2015.

Florida Senate (2012b). Sb 1768: Autonomous vehicle technology. The Florida Senate, http://www.flsenate.gov/Session/Bill/2012/1768. Accessed: 16.02.2015.

Free Software Foundation (2007). The gnu general public license v3.0 - gnu project - free software foundation. http://www.gnu.org/copyleft/gpl.html. Accessed: 11.03.2015.

Furuhashi, S. (2013). Messagepack: It's like json. but fast and small. http://msgpack.org/. Accessed: 18.05.2015.

Games, E. (2015). Behavior tree quick start guide. https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/QuickStart/. Accessed: 25.05.2015.

Garathun, M. G. (2014). Så mye koster det egentlig å bygge vei i norge. http://www.tu.no/samferdsel/2014/07/07/sa-mye-koster-det-egentlig-a-bygge-vei-i-norge. Accessed: 9.04.2015.

Gasser, T. (2012). Legal issues of driver assistance systems and autonomous driving. In Eskandarian, A., editor, *Handbook of Intelligent Vehicles*, pages 1519–1535. Springer London.

GitHub (2008). Github·build software better, together. `https://github.com`. Accessed: 10.03.2015.

Go, J., Vu, T. D., and Kuffner, J. J. (2006). Autonomous behaviors for interactive vehicle animations. *Graphical Models*, 68(2):90 – 112. Special Issue on {SCA} 2004.

Gomes, L. (2014). Hidden obstacles for google's self-driving cars. `http://www.technologyreview.com/news/530276/hidden-obstacles-for-googles-self-driving-cars/`. Accessed: 16.02.2015.

Graft, K. (2010). id tech 5 rage engine no longer up for external licensing. `http://www.gamasutra.com/view/news/120702/id_Tech_5_Rage_Engine_No_Longer_Up_For_External_Licensing.php`. Accessed: 11.03.2015.

Halevy, A., Norvig, P., and Pereira, F. (2009). The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12.

Harris, M. (2014). Fbi warns driverless cars could be used as 'lethal weapons'. The Guardian, `http://www.theguardian.com/technology/2014/jul/16/google-fbi-driverless-cars-leathal-weapons-autonomous`. Accessed: 27.02.2015.

Hayes-Roth, B. and Van Gent, R. (1996a). Improvisational puppets, actors, and avatars. In *Proceedings of the 1996 Computer Game Developers' Conference. Stanford Knowledge Systems Laboratory Report KSL-96-09. file://www-ksl. stanford. edu/pub/KSL_Reports/KSL-96-09. ps*. Citeseer.

Hayes-Roth, B. and Van Gent, R. (1996b). The virtual theater project. Adaptive Intelligent Systems, Stanford, `http://www-ksl.stanford.edu/projects/cait/index.html`. Accessed: 9.03.2015.

Hodgins, J. and Wooten, W. (1998). Animating human athletes. In Shirai, Y. and Hirose, S., editors, *Robotics Research*, pages 356–367. Springer London.

Howe, D. (2015). Free on-line dictionary of computing. `http://foldoc.org/Reliable+communication`. Accessed: 16.04.2015.

id Software (2011). Doom 3 gpl source release. `https://github.com/TTimo/doom3.gpl`. Accessed: 11.03.2015.

IEEE (2012). Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793.

Isaacs, R. (1999). *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation.

ISO (1994). Iso/iec 7498-1:1994. The International Organization for Standardization, `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269`. Accessed: 12.02.2015.

ISO (2013). Iso 15628:2013 intelligent transport systems – dedicated short range communication (dsrc) – dsrc application layer. The International Organization for Standardization, `http://www.iso.org/iso/catalogue_detail.htm?csnumber=59288`. Accessed: 12.02.2015.

Jaguar (1998). Jaguar teams with delphi to introduce adaptive cruise control. PRNewswire, `http://www.prnewswire.com/news-releases/jaguar-teams-with-delphi-to-introduce-adaptive-cruise-control-76713372.html`. Accessed: 13.02.2015, Article source: Coventry, England, Sept. 24 /PRNewswire/ - Jaguar Cars Limited today.

Jain, S., Sawlani, M., and Chandwani, V. (2010). Ad-hoc swarm robotics optimization in grid based navigation. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 1553–1558.

Jensen, M. (1999). Passion and heart in transport — a sociological analysis on transport behaviour. *Transport Policy*, 6(1):19 – 33.

Kahn, K. M. (1979). Creation of computer animation from story descriptions. Technical Report 540, Massachusetts Institute of Technology, 77 Massachusetts Avenue.

Kala, R. and Warwick, K. (2012). Planning autonomous vehicles in the absence of speed lanes using lateral potentials. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 597–602.

Kaneko, J. and Shimamura, A. (2000). A design of safety support maneuvers for automated vehicles cruising around intersections. In *American Control Conference, 2000. Proceedings of the 2000*, volume 5, pages 3518–3522 vol.5.

Klima- og miljødepartementet (2014). Klimaforliket. `https://www.regjeringen.no/nb/tema/klima-og-miljo/klima/innsiktsartikler-klima/klimaforliket/id2076645/`. Accessed: 20.02.2015.

Konolige, K., Agrawal, M., Bolles, R., Cowan, C., Fischler, M., and Gerkey, B. (2008). Outdoor mapping and navigation using stereo vision. In Khatib, O., Kumar, V., and Rus, D., editors, *Experimental Robotics*, volume 39 of *Springer Tracts in Advanced Robotics*, pages 179–190. Springer Berlin Heidelberg.

Larsen-Vonstett, . (2014). Hvilken av disse tror du er norges mest solgte bil? `http://www.vg.no/forbruker/bil-baat-og-motor/bil-og-trafikk/hvilken-av-disse-tror-du-er-norges-mest-solgte-bil/a/23241389/`. Accessed: 05.05.2015.

Larsson, A. (2014). Bilen som kör för egen maskin. GP - Göteborgs-Posten, `http://www.gp.se/nyheter/goteborg/1.2352618-bilen-som-kor-for-egen-maskin`. Accessed: 26.02.2015.

Levin, T. (2015). Private communications. Note: Our NPRA contact for road traffic data.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. (2011). Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168.

Lewis, J., Brown, D., Cranton, W., and Mason, R. (2011). Simulating visual impairments using the unreal engine 3 game engine. In *Serious Games and Applications for Health (SeGAH), 2011 IEEE 1st International Conference on*, pages 1–8.

Liebemann, E., Meder, K., Schuh, J., and Nenninger, G. (2004). Safety and performance enhancement: The bosch electronic stability control (esp). *SAE Paper*, 20004:21–0060.

Lowensohn, J. (2014). This is tesla's d: an all-wheel-drive model s with eyes on the road. The Verge, `http://www.theverge.com/2014/10/9/6955357/this-is-tesla-s-d-an-all-wheel-drive-car-with-eyes-on-the-road`. Accessed: 25.02.2015.

Madrigal, A. C. (2014). The trick that makes google's self-driving cars work. The Atlantic, `http://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/`. Accessed: 27.02.2015.

Maes, P., Darrell, T., Blumberg, B., and Pentland, A. (1995). The alive system: full-body interaction with autonomous agents. In *Computer Animation '95., Proceedings.*, pages 11–18, 209.

Markoff, J. (2014). A trip in a self-driving car now seems routine. The New York Times, `http://bits.blogs.nytimes.com/2014/05/13/a-trip-in-a-self-driving-car-now-seems-routine/?rref=upshot&_r=0`. Accessed: 27.02.2015.

Marks, S., Windsor, J., and Wünsche, B. (2007). Evaluation of game engines for simulated surgical training. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 273–280, New York, NY, USA. ACM.

Mataric, M. J. (1995). Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51–80.

Mayden, A. (2014). Unreal engine 4 vs. unity: Which game engine is best for you? `http://blog.digitaltutors.com/unreal-engine-4-vs-unity-game-engine-best/`. Accessed: 12.03.2015.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitsubishi (1992). History of mitsubishi motors. Mitsubishi Motors Corporation, `http://www.mitsubishi-motors.com/en/corporate/aboutus/history/1990/index.html`. Accessed: 13.02.2015.

Moberg, K. (2014). Her ser du norges vanligste bil. `http://www.dagbladet.no/2014/03/10/tema/aller/motor/bil/dinside/32226056/`. Accessed: 05.05.2015.

Mod DB (2002). About us - mod db. `http://www.moddb.com/about`. Accessed: 11.03.2015.

Mohan, Y. and Ponnambalam, S. (2009). An extensive review of research in swarm robotics. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 140–145.

Moon, A., Calisgan, E., Operto, F., Veruggio, G., and Van der Loos, H. M. (2012). Open roboethics: Establishing an online community for accelerated policy and design change. *We robot*.

Méndez, R. J. (2014). Unreal engine 4 vs. unity - a quick overview. `http://arges-systems.com/blog/2014/05/12/unreal-engine-4-vs-unity/`. Accessed: 12.03.2015.

NCA (2011). Automatic identification system (ais). The Norwegian Coastal Administration, `http://www.kystverket.no/en/EN_Maritime-Services/Reporting-and-Information-Services/Automatic-Identification-System-AIS/`. Accessed: 16.04.2015.

Nevada Senate (2011). Ab511. The Nevada Senate, `http://www.leg.state.nv.us/Session/76th2011/Reports/history.cfm?ID=1011`. Accessed: 16.02.2015.

NHTSA (2013). Traffic safety facts 2012. National Highway Traffic Saftey Administration, `http://www-nrd.nhtsa.dot.gov/Pubs/812032.pdf`. Accessed: 27.02.2015.

NHTSA (2013). U.s. department of transportation releases policy on automated vehicle development. National Highway Traffic Safety Administration, `http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development`. Accessed: 13.02.2015.

NPRA (2014a). About the norwegian public roads administration. The Norwegian Public Roads Administration, `http://www.vegvesen.no/en/The+NPRA/About+the+NPRA/About+the+NPRA`. Accessed: 11.02.2015.

NPRA (2014b). ITS towards 2020. The Norwegian Public Roads Administration, `http://www.vegvesen.no/en/Professional/Technology/Intelligent+transport+systems+ITS`. Accessed: 11.02.2015.

NVIDIA (2015). Gameworks physx overview. `https://developer.nvidia.com/gameworks-physx-overview`. Accessed: 30.04.2015.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420.

OpenGL (1992). Opengl - the industry standard for high performance graphics. `https://www.opengl.org`. Accessed: 11.03.2015.

ORi (2014). Results: If a death by an autonomous car is unavoidable, who should die? Open Roboethics Initiative, `http://www.openroboethics.org/results-if-a-death-by-an-autonomous-car-is-unavoidable-who-should-die/`. Accessed: 17.02.2015.

Ortiz, A. and Neogi, N. (2006). Color optic flow: A computer vision approach for object detection on uavs. In *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, pages 1–12.

Partridge, B. L. (1982). The structure and function of fish schools. *Scientific American*, 246(6):114–123.

Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296.

Perlin, K. and Goldberg, A. (1996). Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 205–216, New York, NY, USA. ACM.

Petit, J. and Shladover, S. (2014). Potential cyberattacks on automated vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, PP(99):1–11.

Pottinger, D. (1999a). Coordinated unit movement. *Game Developer*, 3(3):42–51.

Pottinger, D. (1999b). Implementing coordinated movement. *Game Developer Magazine*, pages 48–58.

Press, O. U. (2015). autonomous - definition of autonomous in english from the oxford dictionary. `http://www.oxforddictionaries.com/definition/english/autonomous`. Accessed: 13.02.2015.

Pyper, J. and ClimateWire (2012). Has the u.s. reached "peak car"? Scientific American, `http://www.scientificamerican.com/article/has-us-reached-peak-car-americans-driving-less/`. Accessed: 25.05.2015.

QMB (2013). Create safe, dynamic highways with moveable barrier. QMB BARRIER, `http://qmb.ca/barrier/construction/construction.html`. Accessed: 9.04.2015.

Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 349–358, New York, NY, USA. ACM.

Ramey, J. (2014). Watch an autonomous audi rs7 fly around the hockenheim circuit. `http://autoweek.com/article/car-news/watch-audis-autonomous-rs-7-fly-around-hockenheim-circuit`. Accessed: 26.03.2015.

Reeves, W. T. (1983). Particle systems—a technique for modeling a class of fuzzy objects. In *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '83, pages 359–375, New York, NY, USA. ACM.

Regjeringen (2014). Statsbudsjettet. `http://www.statsbudsjettet.no/Statsbudsjettet-2015/Dokumenter-NY/`. Accessed: 19.02.2015.

Renault, O., Thalmann, N. M., and Thalmann, D. (1990). A vision-based approach to behavioural animation. *The Journal of Visualization and Computer Animation*, 1(1):18–21.

Resnick, M. (1987). Lego, logo, and life. In *ALIFE*, pages 397–406.

Resnick, M. et al. (1998). The virtual fishtank.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.

Reynolds, C. W. (1988). Not bumping into things. *Computer Graphics*, pages G1–G13. In the notes for the SIGGRAPH 88 course Developments in Physically-Based Modeling. Published by: ACM SIGGRAPH.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782.

Ridsdale, G. J. (1987). *The director's apprentice: animating figures in a constrained environment*. PhD thesis, Simon Fraser University.

RIF (2015). Norges tilstand - state of the nation. Rådgivende Ingeniørers Forening, `http://www.rif.no/media/5486/rif_stateofthenation_2015_lavoppploeselig.pdf`. Accessed: 9.04.2015.

Rügheimer, U. (2015). 550 mile piloted drive from silicon valley to las vegas: long distance test in the audi a7 sportback piloted driving concept car. Audi, `https://www.audi-mediaservices.com/publish/ms/content/en/public/pressemitteilungen/2015/01/04/550_mile_piloted_drive.-download.gid-oeffentlichkeit.acq/qual-DownloadFileList.Single.DownloadFile.0001.File/20150104_Audi_Start%20pilotierte%20Fahrt_en.pdf`. Accessed: 24.02.2015.

Safe Car News (2014). Un amends vienna convention on road traffic to allow driverless cars. `http://safecarnews.com/un-amends-vienna-convention-on-road-traffic-to-allow-driverless-cars/`. Accessed: 21.04.2015.

Samferdselsdepartementet (1986). Veitrafikkloven. Forskrift om kjørende og gående trafikk (trafikkregler), `https://lovdata.no/dokument/SF/forskrift/1986-03-21-747?q=vikeplikt+utrykningskjøretøy`. Accessed: 23.02.2015.

SARTRE (2014). The sartre project. SAfe Road TRains for the Environment (SARTRE), `http://www.sartre-project.eu/en/Sidor/default.aspx`. Accessed: 26.02.2015.

SGS (2014). Serious games society: About. Serious Gaming Society, `http://academy.seriousgamessociety.org/about`. Accessed: 10.03.2015.

Shaw, E. (1970). Schooling in fishes: critique and review. *Development and evolution of behavior*, pages 452–480. Ed. by L.R. Aronson, E. Tobach, D.S. Lehrman and J.S. Rosenblatt. San Francisco: W.H. Freeman.

Shaw, E. (1975). Naturalist at large-fish in schools. *Natural History*, 84(8):40–46. Amer Museum Nat. History ATTN: Library Serials Unit Central PK WEST AT 79TH ST, NEW YORK, NY 10024-5192.

Shiffman, D. (2012a). Flocking / examples / processing.org. https://processing.org/examples/flocking. Accessed: 26.03.2015.

Shiffman, D. (2012b). The nature of code - chapter 6. autonomous agents. http://natureofcode.com/book/chapter-6-autonomous-agents/. Accessed: 26.03.2015.

Silva, M., Montenegro, A., Clua, E., Vasconcelos, C., and Lage, M. (2013). Real time pixel art remasterization on gpus. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*, pages 274–281.

Simonite, T. (2013). Toyota unveils an autonomous car, but says it'll keep drivers in control. Car and Driver, http://www.technologyreview.com/news/509616/toyota-unveils-an-autonomous-car-but-says-itll-keep-drivers-in-control/. Accessed: 26.02.2015.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 15–22, New York, NY, USA. ACM.

Sneddon, J.-E. (2011). Doom 3 is open sourced. http://www.omgubuntu.co.uk/2011/11/doom-3-is-open-sourced. Accessed: 11.03.2015.

SSB (2015a). Personbiler vraket mot pant, etter bilmerke. Statistics Norway, https://www.ssb.no/transport-og-reiseliv/statistikker/bilreg/aar/2015-03-25?fane=tabell&sort=nummer&tabell=220835. Accessed: 16.04.2015.

SSB (2015b). Registrerte kjøretøy. https://www.ssb.no/statistikkbanken/selectvarval/define.asp?SubjectCode=01&ProductId=01&MainTable=Bilbestand&contents=Bilbest&PLanguage=0&Qid=0&nvl=True&mt=1&pm=&SessID=4173771&FokusertBoks=1&gruppe1=Hele&gruppe2=Hele&VS1=Landet&VS2=&CMSSubjectArea=transport-og-reiseliv&KortNavnWeb=bilreg&StatVariant=&Tabstrip=SELECT&aggresetnr=1&checked=true. Accessed: 19.01.2015.

SSB (2015c). Registrerte kjøretøy, 2014. Statistics Norway, http://ssb.no/transport-og-reiseliv/statistikker/bilreg/aar/2015-03-25#content. Accessed: 20.04.2015.

Still, G. K. (1994). Simulating egress using virtual reality-a perspective view of simulation and design. *IMAS Fire Safety on Ships*.

Strassmann, S. H. (1991). *Desktop theater: automatic generation of expressive animation*. PhD thesis, Massachusetts Institute of Technology.

Sørensen, M. W. J. (2008). 3.17 reversible kjørefelt. Transportøkonomisk Institutt (TØI), `http://tsh.toi.no/doc666.htm`. Accessed: 9.04.2015.

Taylor, E. and Oreskovic, A. (2015). Apple studies self-driving car, auto industry source says. `http://www.reuters.com/article/2015/02/14/us-apple-autos-idUSKBN0LI0IJ20150214`. Accessed: 26.03.2015.

Telenor (2015). Dekningskart. `http://www.telenor.no/privat/dekning/dekning_data.jsp`. Accessed: 16.04.2015.

The Rakyat Post (2015). Toyota, nissan and honda to team up for self-driving car tech. `http://www.therakyatpost.com/business/2015/02/26/toyota-nissan-and-honda-to-team-up-for-self-driving-car-tech/`. Accessed: 26.02.2015.

Tilley, A. (2015). Bmw's self-driving car parks itself and picks you up when you're ready to go. Forbes Life, `http://www.forbes.com/sites/aarontilley/2015/01/06/bmws-self-driving-car-parks-itself-and-picks-you-up-when-youre-ready-to-go/`. Accessed: 25.02.2015.

TOMTOM (2015). Tomtom traffic index. TomTom International B.V., `http://www.tomtom.com/no_no/trafficindex/#/list`. Accessed: 14.04.2015.

Toyota (2000). Toyota motor corporation global website | 75 years of toyota | technical development | electronics parts. Toyota Motor Corporation, `http://www.toyota-global.com/company/history_of_toyota/75years/data/automotive_business/products_technology/technology_development/electronics_parts/index.html`. Accessed: 13.02.2015.

Travers, M. (1987). Animal construction kits. In *ALIFE*, pages 421–442.

Tu, X. (1999). *Artificial animals for computer animation: biomechanics, locomotion, perception, and behavior*. Number 1635. Springer Science & Business Media.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 43–50, New York, NY, USA. ACM.

UN (2015). Chapter xi transport and communications. United Nations, https://treaties.un.org/pages/ViewDetailsIII.aspx?src=TREATY&mtdsg_no=XI-B-19&chapter=11&Temp=mtdsg3&lang=en. Accessed: 20.04.2015.

UNECE (1968). Convention on road traffic. United Nations, Economic Commission for Europe, http://www.unece.org/fileadmin/DAM/trans/conventn/crt1968e.pdf. Accessed: 13.04.2015.

Unity Technologies (2015a). Unity - fast facts. http://unity3d.com/public-relations. Accessed: 11.03.2015.

Unity Technologies (2015b). Unity - game engine. http://unity3d.com/. Accessed: 11.03.2015.

Urmson, C. (2014). Just press go: designing a self-driving vehicle. Google, http://googleblog.blogspot.no/2014/05/just-press-go-designing-self-driving.html. Accessed: 26.02.2015.

Valle, M. (2009). Hva er egentlig 4g? http://www.tek.no/artikler/hva_er_egentlig_4g/67224. Accessed: 16.04.2015.

van de Panne, M., Fiume, E., and Vranesic, Z. (1990). Reusable motion synthesis using state-space controllers. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 225–234, New York, NY, USA. ACM.

VanMiddlesworth, M., Dresner, K., and Stone, P. (2008). Replacing the stop sign: Unmanaged intersection control for autonomous vehicles. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '08, pages 1413–1416, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Volvo (2014). Volvo car group's first self-driving autopilot cars test on public roads around gothenburg. Volvo Car Group, https://www.media.volvocars.com/global/en-gb/media/pressreleases/145619/volvo-car-groups-first-self-driving-autopilot-cars-test-on-public-roads-around-gothenburg. Accessed: 9.04.2015.

Volvo Group (2011). Volvo 8700 low entry - specifications. http://www.volvobuses.com/bus/sweden/sv-se/products/City%20buses/Volvo%208700%20Low%20Entry/Pages/specifications.aspx. Accessed: 05.05.2015.

Wilhelms, J. and Skinner, R. (1990). A 'notion' for interactive behavioral animation control. *Computer Graphics and Applications, IEEE*, 10(3):14–22.

William J. Hughes Technical Center (2014). Global positioning system (gps) standard positioning service (sps) performance analysis report. `http://www.nstb.tc.faa.gov/reports/PAN86_0714.pdf`. Accessed: 04.05.2015.

Zapata, R., Lepinay, P., Novales, C., and Deplanques, P. (1993). Reactive behaviors of fast mobile robots in unstructured environments: sensor-based control and neural networks. In *From Animals To Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, MA*.

Zeltzer, D. (1986). Knowledge-based animation. In *Proc. Of the ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception*, pages 318–323, New York, NY, USA. Elsevier North-Holland, Inc.

Zeltzer, D. (1991). Making them move. chapter Task-level Graphical Simulation: Abstraction, Representation, and Control, pages 3–33. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ziegler, C. (2014). Tesla's autopilot isn't special (but it's still cool). The Verge, `http://www.theverge.com/2014/10/17/6982289/tesla-autopilot-is-a-non-revolution-for-self-driving-cars`. Accessed: 25.02.2015.

Appendix A

The Attached ZIP File

The source code of the simulator we have created should be an attachment to this thesis. The code can also be viewed at `https://github.com/HighwayFlocking/HighwayFlocking`. The following gives a short overview over what is attached. For a closer look, see the comments in each file, and the README.md file. The README.md file can be viewed in html at `https://github.com/HighwayFlocking/HighwayFlocking`.

There are two parts to the source code, the highway simulator, and the simulator controller. The simulator is written in C++ using Unreal Engine, and the controller is written in Python using a few libraries. The following is a list of all the files and directories in the top folder of the attached zip-file, and their description.

- **HighwayFlocking.mp4** - Movie presenting the project.
- **HighwayFlockingSetup.exe** - Use this file to test the simulator without compiling the source code.
- **src** - Source code for the project, this contain the same content as can be viewed on GitHub.

A.1 Running the Simulator

There are two ways of running the simulator. The easiest method is to run the attached pre-compiled exe-file. Alternatively, one can compile the source code.

A.1.1 Installing the Simulator

In the attached zip file code, there is a file named *HighwayFlockingSetup.exe*. Run this file to install the simulator, the program will ask for permissions, click <Yes>. Now follows the typical installation of any program (Figure A.1), click <Next>, specify install folder and shortcuts as desired or simply leave everything with the default values. The final install screen will ask to run the program, click <Yes>. Now, the GUI (Graphical User Interface) will appear (Figure A.2). The GUI allows to run the simulations, the attributes of the simulations can be changed as desired, or everything can be left to default values, click <Start Simulation>. The network access dialog may appear, if it does, click <Allow Access>. Now the simulator will appear, giving an overview of the highway (Figure A.3). Click the new window to activate it. To move around, use the mouse and the *W*, *A*, *S* and *D* buttons on the keyboard. To quickly look around, look at the ground where you want to move to, and press *T*. Press *O* to start the spawners. Press *ESC* to exit the simulator. Click on a vehicle to follow that vehicle, press *1* to return to spectator.

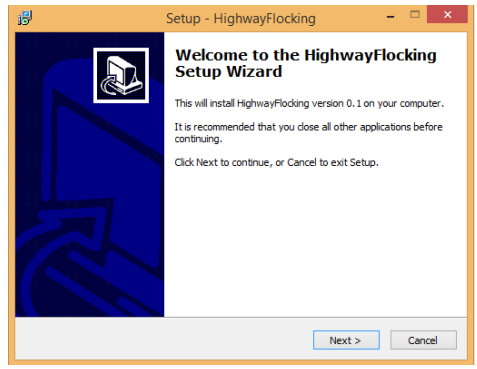


Figure A.1: *The installer welcome screen.*

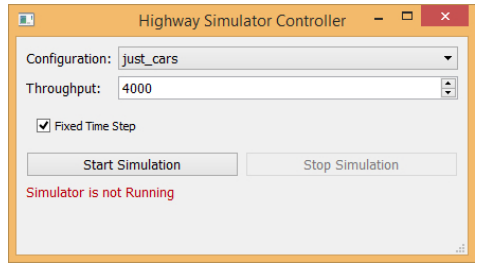


Figure A.2: *The simulator GUI.*

The following keyboard commands can be used:

- Always:
  - *N* - Toggle road markings.
  - *F4* - Toggle HUD.
  - *ESC* - Exit the simulator.
  - *Left Click* - Follow a vehicle.
- As spectator:
  - *W* - Move forward.
  - *S* - Move backward.

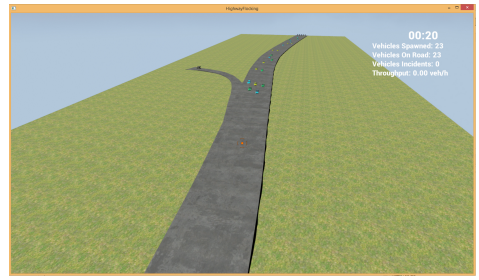


Figure A.3: *The running simulator.*

  - *A* - Move Left.
  - *D* - Move Right.
  - *O* - Toggle Spawners.
  - *T* - Teleport.
  - *Shift + Mouse Wheel* - Zoom.

- Following a flocking vehicle:
  - *1* - Return to spectator.
  - *Mouse Wheel* - Change distance to vehicle.

A.1.2 Compiling the Simulator

To compile the simulator, one first need to set up two dependencies. Firstly, *Unreal Engine* needs to be installed. For this project, we have used version 4.7 of Unreal Engine. However, it should be possible to upgrade the project to a newer version. Secondly, *Microsoft Visual Studio 2013* needs to be installed and set up for Unreal Engine.

When these components are installed, it should be possible to right click *HighwayFlocking.uproject*, and choose *Generate Visual Studio project files*. This will create the project files for Visual Studio. Start Unreal Editor by double clicking *HighwayFlocking.uproject*. You will probably get a messsage saying that *UE4Editor-HighwayFlocking.dll* is missing. Click <Yes> to compile the project. When this is done, you should be able to run the project in Unreal Editor. To view the source code in Visual Studio, press *File | Open Visual Studio*.

The files in the folder *src/controller* can be used to run distrubited experiments. The information about how to use the system is in *src/README.md* or at `https://github.com/HighwayFlocking/HighwayFlocking/blob/master/README.md`.

Appendix B

Structured Literature Review

This appendix presents The methodology used when searching for relevant literature and why we wanted to perform a literature review.

B.1 Structured Literature Review Protocol

This section presents the methodology used when performing the structured literature review. We also present the results of the review and a summary of the major findings.

B.1.1 Research Agenda

The aim of the literature review is to systematically review current solutions for applying flocking algorithms to increase road traffic efficiency. We are interested in papers based on [Reynolds, 1987] and [Reynolds, 1999], and hence we will require any paper included in the literature review to cite either of these. Not all search sites can list papers based on what other papers they cite. For these sites we have used search terms instead.

B.1.2 Background

The problem that we are trying to solve is how to increase the efficiency of the road traffic system using flocking algorithms. For flocking algorithms to work, each car must follow a set of individual rules. The rules are similar between all cars and this will cause emergent behavior to arise.

In addition to utilizing flocking for increasing road traffic efficiency, we are also interested in background material on flocking algorithms. Particularly how flocking may be applied to other research areas.

A final reason for the review, is to indirectly gain a thorough understanding of the state of research with respect to flocking algorithms and how they may be applied to increase road traffic efficiency.

B.1.3 Literature Review Questions

Specifically our literature review questions are:

1. What are the existing solutions to utilizing flocking algorithms to increase road traffic efficiency?

2. What other research areas are using flocking algorithms to benefit from emergent behavior?

3. What implications should these findings have when creating our solution?

B.1.4 Search Strategy

Below is a list of the databases we used in our search, a list of the search terms that were used in the search, and the inclusion criteria used to filter results.

Databases

- ACM Digital Library [ACM, 1947]
- IEEE Xplore [IEEE, 1946]
- ISI web of knowledge [ISI, 1960]
- ScienceDirect [ScienceDirect, 1997]
- CiteSeer^x [CiteSeer^x, 2008]
- SpringerLink [Springer, 1842]
- Wiley Inter Science [Wiley, 1807]
- Google Scholar [Google, 2004]

List of Search Terms

Flocks Herds and Schools A Distributed Behavioral Model
Steering Behaviors For Autonomous Characters
"craig reynolds"
"reynolds c"
"steering behaviour boids"
"steering behaviour"
"steering behavior traffic"
"autonomous vehicle flock"

Search Procedure

The search procedure used in this review was to list all papers at each search site that cited either [Reynolds, 1987] or [Reynolds, 1999]. The list of papers were then sorted on year (newest first), and we screened the top 100 results. The papers were then sorted on citations count (highest first), and we screened the first 100 results.

The reason for sorting on year is that we assume newer papers incorporate the knowledge and build on, older papers. The reason for sorting on citation count is that we assume papers with more citations are overview papers that give a wider perspective and presents relevant pointers to other literature.

Inclusion Criteria

To make sure we find relevant papers we used inclusion criteria to filter out non-relevant papers.

Inclusion Criteria

1. The main concern of the study is either one or multiple of the following topics:
   - Flocking
   - Steering behavior
   - Intelligent Road Traffic Systems
2. The study is a primary study presenting empirical results.

To further filter our papers we used the following quality criteria. The criteria marked with a star are seen as essential, if these are not met, the paper is rejected.

Quality Criteria

1. There is a clear statement of the aim of the research. *****
2. The study is put into context of other studies and research. *****
3. The system or algorithmic design decisions are justified.
4. The test dataset is reproducible.
5. The study algorithm is reproducible. *****
6. The experimental procedure is thoroughly explained and reproducible. *****
7. It is clearly stated in the study which other algorithms the study's algorithm(s) have been compared with.
8. The performance metrics used in the study are explained and justified.
9. The test results are thoroughly analyzed.
10. The test evidence support the findings presented.

Each paper was graded according to whether the quality criteria were satisfied or not, and whether the paper was relevant for our topic. The table below shows the grades used.

Grade	Meaning
0	The criteria was not met
1	The criteria was met, and the paper is partly relevant
2	The criteria was met, and the paper is relevant
3	The criteria was met, and the paper is highly relevant

Only the papers receiving a grade of 1 or better was included in the review.

B.2 Literature Review Process

The literature review process was conducted by manually searching for each search term in every database and screen the results against the inclusion criteria. If we had more than 100 results, we only screened the first 100, if the results returned was clearly not relevant we moved on to the next search term.

Some search sites do not enable searching citing papers based on year or citation count. For sites like these we used search terms instead and screened the 100 top results.

Some papers on Google Scholar [Google, 2004] required purchases to be accessed and were not included. Google scholar is also unable to sort citing papers on citation count and Google's own *relevance* metric were used instead.

Search Engine	Papers
ACM Digital Library	16
IEEE Xplore	11
ISI web of knowledge	0
ScienceDirect	1
CiteSeer^x	13
SpringerLink	2
Wiley Inter Science	1
Google Scholar	18

Table B.1: *How many papers were found at each search site after the quality criteria have been applied.*

Results that contained dead links were skipped.

Wiley [Wiley, 1807] does not allow searches with single characters, hence *reynolds c* had to be changed to *reynolds*.

After screening the results with regard to Inclusion Criteria 1 and 2 we were left with a total of 151 papers, and after applying the quality criteria we were left with 62. Table B.1 shows the distribution of papers per search site after the quality criteria have been applied.

B.3 Literature Overview

This section presents an overview of the literature found during our literature search. We have organized the section into subsections concerning major fields of research and a final subsection including subjects that are not directly relevant but demonstrates other usages of flocking algorithms and emergent behavior.

B.3.1 General Flocking and Steering behavior

The first heuristic rules that led to computer animations of flocking was created by Reynolds in 1987 [Reynolds, 1987]. Since then, many other researchers have build on his work. Olfati-Saber focused on distributed flocking and specifically the stability of a flock and how to guarantee a flock's convergence [Olfati-Saber, 2006]. Rodrigues et al. created an experimental novel steering behavior based on Tree Paths [Rodrigues et al., 2009]. The idea is to utilize a space colonization algorithm [Runions et al., 2005] to construct a path where there is available space. The space that can be traveled is divided into discrete areas. These areas are then represented by markers. The space colonization algorithm uses the markers to construct a path where there is a marker that is unoccupied.

An approach to steer a flock of boids can be to use artificial potential fields (AFPs) [Warren, 1989; Hernandez and Welborn, 2011]. AFPs are a widely used in motion planning for robots [Khatib, 1985; Espitia and Sofrony, 2011], much due to their easiness in implementation. The basic idea of an AFP is to treat the location of the robot as a point in a potential field that combines attraction to the goal and repulsion from obstacles. The resulting behavior is that the robot will move towards the goal while avoiding the obstacles. A problem is that the robot can get stuck in a local minima. The robot can also fail to find a way, e.g. if the repulsion from two objects is too high for the robot to approach the way between them.

When boids are controlled by the behavior described in [Reynolds, 1987] (collision avoidance, velocity matching, and flock centering), and no other behaviors, the resulting movement can appear rather random. To ensure the boids in the flock appears to have a common direction an external leading force have been proposed [Hartman and Beneš, 2006]. Here the boids in front or on the edge of the flock have a certain probability of trying to "take over leadership" of the flock. When a boid becomes a leader, the trailing boids will follow the leader and the boids appear to have a common direction.

Much of the work on flock steering focuses on the individual boids and how to construct behavior that causes a particular emergent behavior. Another angle of approach to emergent behavior is to begin with a top level view of the flock and based on the emergent behavior try to map out the individual rules of each boid. Croitoru have presented a methodology for estimating the boid steering behavior that will explain the observed emergent behavior [Croitoru, 2009].

Though the steering behavior of an agent needs to be somewhat dependent on the surrounding agents to avoid collisions, the overall goal for an agent can be individualistic. An agent can compute a *roadmap* [LaValle, 2006] of the available paths. A roadmap represents the available paths with nodes and vertices and hence enables the use of Dijkstra's algorithm [Dijkstra, 1959] to find the shortest path for each agent. This approach is demonstrated in [van den Berg et al., 2008].

Effort has been put into investigating how flocks can maintain formations without having to implement complex constraints. A possible solution is to use probabilistic methods to reorganize flock behaviors [Aoyagi and Namatame, 2006]. Behavioral forces can also be used to maintain a flexible flock formation [Ho et al., 2010]. Behavioral forces are similar to Reynolds' acceleration used in [Reynolds, 1987] where the different accelerations are accumulated together.

Interesting aspects of flocking also includes how robust a flock is when exposed to turbulent flow fields, i.e. when groups of marine micro-organisms are exposed to turbulent waters. Flierl et al. investigated how turbulent flow fields affect flocks and found that grouping behavior is enhanced up to a certain level [Flierl et al., 1999].

B.3.2 Intelligent Traffic Systems (ITS)

ITS is the main focus of the literature review. There are multiple ways of modeling road traffic [Helbing, 2001]. One possibility is to model road traffic as *Microscopic follow-the-leader models*. In these models, the acceleration is given by the neighboring vehicles. A car can not drive faster than the car in front of it (without colliding) and hence the dominant influence on the acceleration is coming from the next vehicle ahead. *Macroscopic traffic models* are more restrictive in terms of only allowing the collective vehicle dynamics to be described by spatial vehicle density per lane, and the average velocity as a function of the location on the freeway and time.

Another approach is *Cellular Automata*. These models are very efficient and have high computation speed because they use discretization of space, a final number of states, and update intervals. It is possible to simplify road traffic models even further. *Master equation* models consists of a linear equation that differs from Cellular Automata mainly by being continuous and not discrete. The reason for simplifying road traffic models to such an extent can be a desire to identify the minimum requirements for a certain phenomenon, and to facilitate analytical investigations [Helbing, 2001].

When modeling a virtual world, it can be challenging to get the correct steering behavior from the autonomous vehicles. Particularly complex environments can be challenging because the agent only have a local perspective. A solution has been proposed where information about the global environment has been embedded in the pathway [Bayazit et al., 2002]. Wang et al. used an approach where they modeled the roadway surfaces as three-dimensional ribbons [Willemsen et al., 2003] that makes the local orientation of the road explicit and eases the road distance computations [Wang et al., 2005].

There are many different types of steering behavior. Anyone can design his or her own type of steering behavior, and this leads to the need for an objective way to evaluate the different behaviors. *SteerBench* is a benchmark suite for evaluating steering behaviors [Singh et al., 2009]. The suit consists of multiple test cases (30 cases in 2009) that will evaluate the performance of the agent's steering behaviors.

The steering possibilities of future cars might be different from the current cars. Research has been conducted to investigate the behavior of cars that can steer with both back and front wheels [BenAmar, 1997] [Ploeg et al., 2002].

There is a very interesting project called *AIM* (Autonomous Intersection Management) [Dresner and Au, 2006]. This project aims to create a scalable, safe, and efficient multiagent framework for managing autonomous vehicles at intersections [Dresner and Stone, 2004, 2005, 2006, 2008; Dresner, 2009]. The idea is a reservation system where vehicles driving towards the intersection can reserve a safe path for when they arrive. Since the reservations are done before the ve-

hicles arrive, the system can plan ahead and find the most efficient order for the vehicles to pass through the intersection. The results show that it is possible for autonomous intersections to handle a much higher traffic load than the current traffic light system.

Little's law [Little, 1961] shows that there are two ways to increase the throughput of an intersection: (1) build a larger intersection, a larger intersection will have greater capacity and can handle more vehicles, or (2) reduce the time each vehicle spends in the intersection. Since the size of an intersection is typically a hard limit, especially in cities where buildings would have to be demolished to create more space, a preferable choice is to reduce the time each vehicle spends in the intersection. The AIM system has been further optimized by reducing the time each vehicle spends in the intersection [Au and Stone, 2010].

Regarding road traffic and flocking, approaches has been suggested that allow autonomous vehicles to flock together to win speed bonuses [Astengo-Noguez and Sánchez-Ante, 2007]. The flocks will have a virtual agent per flock called "the flock spirit", and it will negotiate the entrance of future flock members. An interesting feature of this work is that the algorithm is compatible with the intersection management of Dresner and Stone in [Dresner and Stone, 2004].

The road traffic system is usually regarded as an asynchronous system, but synchronous approaches to autonomous vehicles have also been suggested [Xiong et al., 2010]. In this system, there are multiple states that each vehicle goes through for each *computational cycle*. A computational cycle typically consists of observing the surroundings, performing a set of computations, and then moving to a new location. All vehicles finish the computation cycle at the same time.

Conflicts arise in road traffic all the time, and El hadouaj et al. have created a system for conflict resolution [El hadouaj et al., 2001]. However, the system is built to solve conflicts that will not arise when human drivers are replaced with autonomous drivers. The system also assumes that roads are divided into lanes which is not the case with flocking.

B.3.3 Artificial Life

Artificial life as defined by Langton et al. is concerned with attempting to recreate biological phenomena from scratch, within computers and other artificial media [Langton et al., 1992]. The approach used is to put together systems that behave like living organisms, as opposed to having living organisms and taking them apart to see how they work. Within the domain of artificial life, it is possible to create artificial chemical substances and from there conduct research on *synthetic biology*. Artificial life is the study of *life-as-it-cold-be* rather than *life-as-we-know-it* [Liekens, 2003].

Much effort has been put into modeling realistic behavior of different types

of wildlife. Particularly animals that travel in flocks and display general flocking behavior have received the majority of the interest. Schools of fish have been modeled with emphasis on their steering behavior [Inada, 2001]. The similarities and differences between schooling models have also been investigated [Parrish et al., 2002] along with the relationship between individual fish behavior and the emergent behavior of the school [Viscido et al., 2004]. The theoretical properties of the emergent behavior of schools have also been described mathematically [Toner and Tu, 1998].

Fish in an aquarium can be very visually appealing to look at and can also mount a worthy challenge to simulate. A possible approach is to model each fish as an autonomous agent where the agent's behavior depends on the observed world [Tu and Terzopoulos, 1994]. A virtual aquarium can also form an attractive exhibition for the general public [Kaufman et al., 1998]

Botanical gardens can also be visually appealing to look at, and a virtual botanical garden have been implemented [Martins et al., 2013]. The garden is part of an exhibition and is interactive to let visitors play with the organisms.

Modeling humans can be useful in many situations. A normal approach is to animate people in public areas to investigate how well suited an area is for a particular purpose. Sophisticated behavior is needed to realistically simulate humans. A comprehensive model of pedestrians that combines perceptual, behavioral, and cognitive control components have been proposed [Shao and Terzopoulos, 2005]. In particular situations, we are more interested in how humans behave in a crowd. Crowd simulations can be useful for e.g. simulating emergency evacuations of buildings or stadium entrances. The steering behaviors from flocking can be applied to each individual agent to achieve realistic crowd behavior.

When simulating a fire evacuation with thousands of people, it is often important to have algorithms that support parallelism. *ClearPath* is an algorithm that implements local collision avoidance between multiple agents for real-time simulations [Guy et al., 2009]. Simulating the emergent behavior of thousands of agents can be a very resource consuming task, especially if the agents are also supposed to have some level of individuality. An algorithm has been proposed that uses *streams* to achieve coordinated behavior, and incentives to stimulate individualism [Goethem, 2012]. Family members tend to remain nearby one another when exiting a building, e.g. a stadium. The family does not drive away till all members have arrived, this property has been modeled in [Rodriguez et al., 2012]. When simulating daily pedestrian traffic in a street, the emergency evacuation models might not be appropriate. In such scenarios, it could be better to account for the fact that people often travel in small groups of two or three persons [Wu et al., 2013].

B.3.4 Robotics

Robotics is a hot research topic, and the principles of flocking can help multiple robots stay in formation while avoiding obstacles and reaching navigational goals [Balch and Arkin, 1998]. The robots do not necessarily have to be driving around on the ground but can also be flying. Unmanned Air Vehicles (UAVs) are increasingly popular and could benefit from moving in formation, e.g. for reconnaissance purposes. Watson et al. have investigated the implementation of real-time graphical simulation of flocking of UAVs [Watson et al., 2003]. When using robots to map unknown terrain, particularly if the terrain is dangerous, it can be a great economic risk to use an expensive robot for such reconnaissance missions. Another approach is to use multiple low-cost robots where the collective whole can achieve the same task, even if a few of the robots fail [Ercan and Li, 2011].

A popular topic is to use flocking algorithms and swarm intelligence in combination to have multiple robots work together to achieve tasks [Campo et al., 2006; Oyekan and Hu, 2013].

For large swarms, it can be challenging to communicate with all agents. An approach where only a subset of the flock is informed of goals has been shown to scale well [Ferrante et al., 2010]. The idea is that the subset of informed agents will propagate the information to the rest of the swarm. However, the receiving agents do not know whether the information they get is from an informed agent or not. Swarms are not only researched by scientists; engineers are also interested in modeling, designing, and realizing swarm robotics systems [Brambilla et al., 2013].

Part of the idea with flocking is to have multiple agents travel as a group without having to specify directly any formation that should be maintained. Another approach is, of course, to specify this formation. Balch and Arkin have experimented with simulated robots that try to maintain specific formations [Balch and Arkin, 1995]. A very interesting approach from our perspective is the method introduced by Balch and Hybinette [Balch and Hybinette, 2000]. The approach is inspired by the way molecules form crystals. Each vehicle has multiple *attachment sites* which other vehicles are attracted to. A vehicle v_1 will typically have such an attachment site at least in the front and back. Then another car v_2 would position itself in the front or back to reduce drag for either of the vehicles.

B.3.5 Particle Swarm Optimization (PSO)

Particle swarms are used for optimizing nonlinear functions [Kennedy and Eberhart, 2010]. Optimization is achieved through designing the behavior of the particles such that each particle contributes to a common goal. The particles themselves might be unaware of this objective but the emergent behavior causes

the flock to behave more like a swarm, and the collective whole will cause goal achieving behavior.

The individual particles can also be aware of a global state [Eberhart and Kennedy, 1995]. In this scenario, the flock has become a swarm with a common goal between all the individual particles. It is possible to construct PSOs that can guaranteed convergence on either local or global minima [van den Bergh, 2001]. The heuristic parameters of the search can be controlled with Differential Evolution Algorithms [Storn and Price, 1997] [Parsopoulos and Vrahatis, 2002].

PSO have undergone many changes since its invention in 1995, some of the changes that have happened are described in [Poli et al., 2007].

B.3.6 Other Usages of Flocking and Steering Behaviors

Here we present some areas of research that incorporate ideas from flocking theory that are not directly relevant for intelligent road traffic systems but still use these ideas to benefit from emergent behavior.

Flocking algorithms can make up a basis for document clustering analysis [Cui et al., 2006]. An advantage of this type of clustering is that the approach does not require initial partition seeds, and also the convergence speed can be less than k-means and ant-clustering.

In computer games, there may be multiple autonomous agents. These agents could, for instance, be boats, cars, or airplanes. If all of the agents were to have a complex autonomous behavior, it would induce a high computational load on the computer. Go et al. demonstrates how simplifications can be made to the path planning process [Go et al., 2006]. The physics as well can be simplified to ease the computational load and still achieve the steering behaviors described by Reynolds (*Seek, Pursue, Avoid, Avoid Obstacle and Flee*). It is not only the behavior generation that can induce a considerable workload, after the behaviors have been generated the model needs to be animated. It would be preferable if much of the code could be reused between applications. A framework for reusing behavior and animation of boids is presented in [Whiting et al., 2010].

Musical flocks are presented in [Kamolov et al., 2013]. Here boids react to features of the music to create visualizations. The features can be, e.g., tempo, volume, or analysis of the sound spectrum. The resulting visualization represents an abstract representation of the music played. Flocking can also be used to select what music to play on a radio station [Ibáñez et al., 2003].

Autonomous vehicles are useful for exploring underwater sites as well as sites on the surface. Autonomous underwater vehicles (AUVs) can utilize flocking algorithms to ensure there are no collisions when multiple AUVs are operating together [Sahu et al., 2012; Sahu and Subudhi, 2014].

Shepherding is a scenario where an outside agent is trying to control the

members of a flock. Experiments in herding have been conducted for herding e.g. sheep [Vaughan et al., 2000] with a robot sheepdog and ducks with a simulated person [Lien et al., 2004].

B.3.7 Literature Review Conclusions

In this section, we present lessons learned from the literature review. We have divided the section into two subsections; the first detailing approaches that we will not make use of, and the second detailing the approaches we will make use of. Both subsections include argumentation for our decisions.

Proposed Techniques That We Will Not Make Use Of

In this literature review, we have looked at existing solutions of flocking algorithms to increase road traffic efficiency. There does not appear to be extensive research on road traffic applications of flocking algorithms. However, some approaches have been presented.

The negotiation approach proposed in [Astengo-Noguez and Sánchez-Ante, 2007] requires explicit flock membership and introduces an entity called the *flock spirit* that handles negotiations. Such an entity and negotiations impose extra complexity and a greater workload on each vehicle. The essence of flocking from our perspective is to keep complexity to a minimum. We want our approach to adhere to the "Keep it simple stupid" (KISS) principle [Axelrod, 1997] as far as possible. The emergent behavior from our flocking is supposed to result from each vehicle following its own rules, and not from negotiation with external forces that dictate the behavior. A vehicle should be able to come and leave the flock as it pleases. The flock spirit can be implemented by either shared state or multiple independent states, but either way the flock spirit introduces overhead to achieve fault tolerance [Vinoski, 2007; Armstrong, 2010]. We want a loosely based flock formation where no vehicle is either a member or not, but rather aligns with other vehicles to reduce drag. If all vehicles align, the emergent flocking behavior will naturally occur.

[Ho et al., 2010] has researched realistic flock movement. To maintain the flock formation while the road makes a turn can be a challenging task. The flock needs to turn in a way that appears realistic while at the same time maintain the flexibility to avoid obstacles. In the experiment performed by Ho et al., there are no constraints on the space used by the flock, apart from not moving into obstacles the flock may use the space needed. I.e. a free flocking approach is used. On a roadway, we have a limited space available for movement and hence are in need of steering behaviors for constrained flocking. We believe, however, that the specific turning movement described by Ho et al. might be useful for implementing smooth turning behavior. However, the behavior will

need modification as the approach described utilizes differences in speed. In a real-world highway with autonomous vehicles all vehicles will probably be driving at the speed limit, and hence the vehicle driving the inside of the curve will be done with the curve faster than a vehicle traveling the outside of the curve and the formation will not be maintained.

The main focus of [Ho et al., 2010] is a steering behavior that can conform to a user defined formation shape requirement. We are not interested in maintaining any formation both because it would introduce extra overhead to maintain, but also because it corresponds to all the members of the formation having some common interest in maintaining the formation. We believe the vast majority of transportation that occurs is due to individual needs, and hence there is no benefit from maintaining a certain formation for the sake of some formational advantage.

We will be using Unreal Engine (UE) to model and run our experiments and, therefore, need a method of implementing steering behaviors, roads, and landscape that is compatible with UE. Our method must also not be too complex as the graphics of UE are resource demanding, and it is not feasible to run real-time simulations with more than approximately 150 vehicles simultaneously with our current hardware. Willemsen et al. have implemented their own data structure called a ribbon network [Willemsen et al., 2003]. The ribbon structure contains information about where the road is and helps navigating the vehicle. Many of the road traffic models used in simulations assumes lanes, e.g. [Wang et al., 2005], and hence are not directly applicable to our road traffic model. In UE, we will implement our own road structure, and it is possible to get the direction of the road from UE. The main focus of the experiments will be the performance of the flocking steering behavior versus regular autonomous vehicles. Hence, we will not spend time implementing a very complex road structure.

For testing and experiments, we will use our own self-composed scenarios inspired by real road traffic. We will compare the results to real road traffic. We can not use real data about road traffic flow from autonomous vehicles because they are simply not in use yet. An alternative to implementing our own tests could have been to use the *Steerbench framework* [Singh et al., 2009]. However, the framework is made as a general framework to test general steering behavior, and few of the tests are particularly relevant for our scenarios.

Other possible approaches we could have use include the *Tree Paths* [Rodrigues et al., 2009] approach. However, this approach divides the world into many discrete areas and uses an iterative process to find the possible available paths. We do not believe an iterative approach is a viable choice for steering behavior as actions need to be decided on as quickly as possible. The tree paths approach also introduces extra computations to find all the possible roadways to take, even though they are not all necessary. The steering behavior should be a

constant-time algorithm to ensure decisions are reached in a timely fashion. It is possible to precompute an obstacle-free path [Nieuwenhuisen et al., 2007]. However, pre-computations assumes that the obstacles are stationary. Otherwise, the path might not remain obstacle free for long.

Regarding synchronous approaches to steering behavior, e.g. [Xiong et al., 2010], we believe synchronization imposes too great a constraint on the road traffic system as a whole. Not only does the vehicles have to agree on a common time, but there will probably be a phase where autonomous vehicles will have to drive alongside human-driven vehicles. The human drivers can not be expected to behave as the autonomous vehicles do, hence an asynchronous approach to steering behavior is needed.

Proposed Techniques That We Will Make Use Of

Our final steering behavior will be a combination of multiple approaches. We will mainly be inspired by the steering behaviors of Reynolds in [Reynolds, 1987, 1999], but also draw inspiration from AFPs and lateral potentials [Kala and Warwick, 2012]. We will differ some from AFPs in the sense that we will not be using a gradual increasing repulsion to avoid obstacles. Instead, we will utilize soft and hard limits on how close other vehicles it is possible to get before collision avoidance steering behavior needs to take over. We are somewhat reluctant with respect to the alignment behavior of Reynolds that states that a boid should attempt to match velocity with nearby flockmates. Velocity matching is a tool for maintaining the current state of the flock. Collision avoidance will typically ensure that no collisions happen and that that the boids keep at least a minimal distance between themselves, and then velocity matching will typically maintain that distance. The behavior of boids were inspired by the behavior of birds and birds have natural causes for wanting to stay in a flock, e.g., protection from predators, finding food, and mating opportunities [Reynolds, 1987]. Vehicles do not have the same needs but have different desires for staying in a flock formation; the main advantage being reduced drag. Vehicles may have different max speeds, and it does not make sense for a fast vehicle to maintain a slow speed simply to remain in the flock. An exception being if the human on board has made a priority out of favoring economic driving instead of a quicker arrival. There will probably be an optimization possibility in finding the lower speed limit of a vehicle before it chooses to leave the flock. A vehicle with the possibility to drive faster than what the flock is doing can choose to remain in formation and burn less fuel and save money, but arrive at a later time. The vehicle can also choose to leave the flock and receive more drag, spend more money on fuel, but arrive faster.

Another issue with the alignment behavior is best demonstrated with an example: If two vehicles v_1 and v_2, travelling alongside each other, catches up with

another vehicle v_3, either one of v_1 or v_2 might have the opportunity to overtake v_3. Let us assume v_2 ended up behind v_3, that means v_1 can overtake v_3. As v_1 and v_2 approaches v_3, v_2 will start to slow down to avoid collision with v_3 and v_1 will also start to slow down simply because it wants to match speed of v_2, even though there is really no reason for v_1 to slow down. Hence, we believe our steering behavior might benefit from the alignment behavior either not be implemented or be implemented with a low weighting.

We regard the approach presented in [Balch and Hybinette, 2000] as a very interesting approach. Using *social potentials*, it is possible to have vehicles flock in a predetermined pattern. For highways, it is typically desirable to have vehicles position themselves behind other vehicles to achieve less drag and save fuel. The approach is similar to AFPs and uses attraction towards the desired attachment-sites and repulsion from obstacles to achieve emergent flocking behavior.

Astengo-Noguez and Sánchez-Ante designed an algorithm [Astengo-Noguez and Sánchez-Ante, 2007] that is compatible with the intersection management presented in [Dresner and Stone, 2005]. We would like our flocking behavior to be compatible with this intersection management as well. The algorithm of Dresner and Stone is demonstrated at their homepage [Dresner and Au, 2006] where an intersection that uses lanes is used as an example. The usage of lanes is, however, not a requirement for the algorithm to work. The inner workings of the algorithm are to reserve a space in the intersection for a period of time. This approach actually assumes no lanes in the intersection and is hence perfectly compatible with using roads with no lanes as well.

For intersection management, we have already stated that [Dresner and Stone, 2005] is a very interesting approach. However, there is also other interesting approaches that have been presented. Road traffic intersections can be divided into two categories *managed* and *unmanaged*. The approach presented in [Dresner and Stone, 2005] is a managed approach where an external system is taking over control and directing road traffic in a safe and efficient manner. The other category called unmanaged intersections include approaches as [VanMiddlesworth et al., 2008]. In an unmanaged approach, the vehicles will handle an intersection on their own by using particular steering behavior, or communication between vehicles or both. The technique proposed in [VanMiddlesworth et al., 2008] uses message passing to coordinate road traffic flow. Since communication is based on ad-hoc networks, the message passing can not be reliant on a dialog between vehicles, but rather by repeated broadcast within a certain radius of the ad-hoc network.

It is interesting to observe that the steering behavior we envision have in some sense already been research for possible applications in India [Kala and Warwick, 2012]. Kala and Warwick have investigated a steering behavior for use on roads where there are no lanes. Some countries in the world do not have lanes and

hence need a way for automatic vehicles to steer without the support of lanes for guidance. The technique introduced uses *Lateral potential fields*. Lateral potential fields are similar to AFPs in the sense that they use repulsion of obstacles to avoid collisions. They are similar to the steering behaviors of [Reynolds, 1987] in the sense that there are multiple individual behaviors that each computes a desirable steering potential in some direction. After all the individual potentials have been computed, they are combined into the final steering and acceleration. Our steering behavior will need many of the same properties that would be obtained from the lateral potential field. The behaviors will need to account for large diversity in vehicles in terms of both speeds, sizes, and types, and be very robust in terms of collision avoidance. Our steering behaviors will probably be modeled more after Reynolds's steering behavior than Kala and Warwick, but [Shiffman, 2012] will also contribute as it shows how to implement Reynolds's steering behavior in practice.

Appendix C

The CAM message

Here we show the most central fields included in the *Cooperative Awareness Message (CAM)* which is the message that is transmitted between vehicles in the ITS systems. These are excerpts adopted from [ETSI, 2014] and [ETSI, 2013].

```
CAM ::= SEQUENCE {
    header ItsPduHeader,
    cam CoopAwareness
}

ItsPduHeader ::= SEQUENCE {
    protocolVersion INTEGER,
    messageID INTEGER,
    stationID StationID
}

CoopAwareness ::= SEQUENCE {
    generationDeltaTime GenerationDeltaTime,
    camParameters CamParameters
}

CamParameters ::= SEQUENCE {
    basicContainer BasicContainer,
    highFrequencyContainer HighFrequencyContainer,
    lowFrequencyContainer LowFrequencyContainer OPTIONAL,
    specialVehicleContainer SpecialVehicleContainer OPTIONAL,
    ...
}
```

```
BasicContainer ::= SEQUENCE {
    stationType StationType,
    referencePosition ReferencePosition,
    ...
}

LowFrequencyContainer ::= CHOICE {
 basicVehicleContainerLowFrequency BasicVehicleContainerLowFrequency,
 ...
}

BasicVehicleContainerLowFrequency ::= SEQUENCE {
 vehicleRole VehicleRole,
 exteriorLights ExteriorLights,
 pathHistory PathHistory
}

HighFrequencyContainer ::= CHOICE {
 basicVehicleContainerHighFrequency BasicVehicleContainerHighFrequency,
 rsuContainerHighFrequency RSUContainerHighFrequency,
 ...
}

BasicVehicleContainerHighFrequency ::= SEQUENCE {
    heading Heading,
    speed Speed,
    driveDirection DriveDirection,
    vehicleLength VehicleLength,
    vehicleWidth VehicleWidth,
    longitudinalAcceleration LongitudinalAcceleration,
    curvature Curvature,
    curvatureCalculationMode CurvatureCalculationMode,
    yawRate YawRate,
    accelerationControl AccelerationControl OPTIONAL,
    lanePosition LanePosition OPTIONAL,
    steeringWheelAngle SteeringWheelAngle OPTIONAL,
    lateralAcceleration LateralAcceleration OPTIONAL,
    verticalAcceleration VerticalAcceleration OPTIONAL,
    performanceClass PerformanceClass OPTIONAL,
    cenDsrcTollingZone CenDsrcTollingZone OPTIONAL
```

```
}
```

Appendix D

Avoid Forward

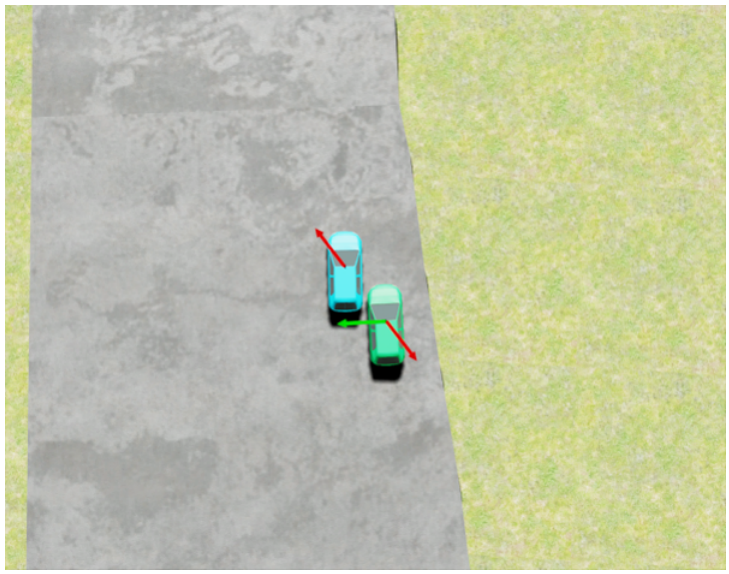


Figure D.1: *The blue vehicle in front is trying to avoid the green vehicle entering the road from the right. Because of the vehicles driving 10% slower of what they are allowed to do, the blue vehicle can increase its speed to allow the green vehicle entrance on the road without colliding.*

Figure D.1 shows a scenario where it would be beneficial for the blue vehicle in front to be able to increase its speed to allow the green vehicle to enter the road. The red arrows in the figure is the desired steering vectors from the Avoid

Behavior(see Section 3.6.3). The green arrow is the desired steering vector from the KIR Behavior (see Section 3.6.4). If the blue vehicle were already driving at its maximum allowed speed, it would be impossible for it to increase the speed any further. Due to the occasional situation where a speed increase is desirable, we have chosen to let the vehicles drive at 90% of the allowed speed. Hence, the vehicles can increase to 100% when needed to avoid collisions.

Appendix E

Diving into the Source Code of Unreal Engine

Unfortunately, the documentation of Unreal Engine have shortcomings in certain areas. However, fortunately, the source code of the editor is included which means that we can use that to find out how things actually work. We have documented our findings here. The following is interesting mostly if one wants to read our source code, or are interested in how to achieve something like what we achieved.

E.1 Running with Fixed Time Steps

As mentioned in Section 3.7, running the simulator created in unreal with fixed time steps is not as easy as the documentation would like to have it. In the Unreal Engine documentation for command line arguments[1] there is an argument, `BENCHMARK`, the documentation states.

```
BENCHMARK: Run game at fixed-step in order to process each frame
           without skipping any frames. This is useful in
           conjunction with DUMPMOVIE options.
```

This would be great, however when we try to run our binaries with this command line arguments, nothing happens. When we are looking at the source code of Unreal Engine, we find out why. This is an excerpt from the file `Runtime\Launch\Private\LanchEngineLoop.cpp`.

[1] `https://docs.unrealengine.com/latest/INT/Programming/Basics/CommandLineArguments/index.html`

```
#if UE_EDITOR

  (...)

#if !UE_BUILD_SHIPPING
    // Benchmarking.
    FApp::SetBenchmarking(FParse::Param(FCommandLine::Get(),TEXT("BENCHMARK")));
#else
    FApp::SetBenchmarking(false);
#endif // !UE_BUILD_SHIPPING

  (...)

#endif
```

This means that the `BENCHMARK` argument only applies if the macro `UE_EDITOR` is `true`. This macro is only true if the code is running in the editor, which it is not doing when we are running our binary. This means that the `BENCHMARK` argument is useless for our use case. Note that this code calls `FApp::SetBenchmarking(true)` in `Runtime\Core\Public\Misc\App.h` when the `BENCHMARK` argument is given. The result is that this function will set a private variable called `bIsBenchmarking` to `true`. The only effect of this is that `FApp::IsBenchmarking()` will return true.

Looking at where this method is used, a line at the start of the `UpdateTimeAndHandleMaxTickRate` method in `Runtime\Engine\Private\UnrealEngine.cpp` looks most interesting. This is an excerpt from that method.

```
// Figure out whether we want to use real or fixed time step.
const bool bUseFixedTimeStep = FApp::IsBenchmarking() || FApp::UseFixedTimeStep();
```

It looks like making `IsBenchmarking()` return `true` is not the only way to achieve fixed time steps, we can also make `UseFixedTimeStep()` return true. If we can make that function return true, we have achieved what we wanted. This is the complete code for the `FApp::UseFixedTimeStep()` method.

```
static bool UseFixedTimeStep()
{
    static bool bUseFixedTimeStep =
        FParse::Param(FCommandLine::Get(), TEXT("UseFixedTimeStep"));
    return bUseFixedTimeStep;
}
```

From this code, it looks like there is another command line argument: `UseFixedTimeStep`. This is not mentioned in the documentation for Unreal

Engine, however using it gives the effect we want. This means that to get fixed time step, we need to call our binary with this argument.

E.2 Extract a Roadmap From a Road

Unreal Engine has at least two different methods that can be used to define a curve. One method is using a Spline Components, and manually extruding a road using the resulting spline and blueprint. However we are using the landscape module to create landscapes, and as a part of a landscape there can be splines defined, which can form roads on the landscape. Using landscape splines is perfect for our use case, and we have used it to create our road model.

However, using landscape splines have one disadvantage, it looks like there is no way of finding the curve of the resulting spline. While Spline Component have methods to find the location and tangent of points along the way, the Landscape has no such method.

Despite that some methods are missing the case is not lost, we have found a way to extract the curve. The following will explain how we discovered how to achieve this. The full implementation is in the method `ARoadmap::CalculateRoadmap()` in the file `Roadmap.cpp` in the source code attached to this thesis.

A landscape spline is defined in the unreal editor by creating and moving control points. A landscape spline consist of multiple control points, each of the control point is connected to one or more control points, forming a graph. The connection between two control points is called a segment. After some digging, we found that the landscape splines are defined by a component called a `ULandscapeSplinesComponent`. The class for the Control Point is called `ULandscapeSplineControlPoint` while the class for the segments is called `ULandscapeSplineSegment`.

We started by comparing the `ULandscapeSplinesComponent` class with the `USplineComponent` class. Since the Spline Component had methods that did exactly what we wanted, we hoped that comparing these classes could give us some insight into how to do this for the landscape splines. Some digging in the source code showed that the class called `FInterpCurveVector` was important. `FInterpCurveVector` defines a curve that have multiple points, where each point have a location and a tangent. The class then defines a curve, where it have a t-value that goes from 0 to some number along the curve. Giving different t-values, we get various points along the curve. The t-values does not mean anything in itself, only that a higher t-value is further along the curve than a lower t-value. `FInterpCurveVector` have methods to get the position and tangent of different

t-values along the curve.[2]

Looking at the `USplineComponent`, all the methods we want are just calling methods on its `FInterpCurveVector`. This means that what we need is to get a `FInterpCurveVector`. For the Landscape, the `FInterpCurveVector` is defined in the `ULandscapeSplineSegment`. Meaning that each segment has its curve. This makes sense since a `FInterpCurveVector` can not have intersections. However, a Landscape Spline needs to be able to have intersections. Therefore, each of the segments has its curve that defines the curve of that segment. As long as the tangents on the start and end of each segment's curve fits, putting all the curves together will make a large graph, where each edge have its curve.

There is just one problem, the field holding the `FInterpCurveVector` of the segment is marked as private. This means that to get this info we need to do nasty hacks, which have a significant possibility of breaking in future versions of Unreal Engine. However, using some more digging, we found the methods that create these curves for each of the segments. By iterating over the segments, and using the information, we can create our own `FInterpCurveVector`. Since we know that our road will never have intersections[3], we can create just one `FInterpCurveVector` that the vehicles use as a road map for the highway.

The `FInterpCurveVector` that we are creating is called `SplineInfo`, and exists in `ARoadmap`. `ARoadmap::CalculateRoadmap()` is the method that creates it, and the code should be relatively easy to follow when the above is known.

[2]Actually, FInterpCurveVector is just a specialization of the FInterpCurve, where an FInterpCurve can have different outputs. FInterpCurveVector is an FInterpCurve where the output is a Vector

[3]Except for the entrance ramp, but that is hard coded using its own Spline Component.

Appendix F

The Experiment Parameters

In this appendix, we present the exact parameters used for our experiments.

F.1 Behaviors

In this section, we present the behaviors used for the experiment. Each behavior has at least a weight, but they can also have other parameters. This listing does not mention the priority since we did not use behavior priorities for our experiments. Therefore, every behavior has the highest priority.

Note that this section is using the names used by our source code. The values are also the ones used in the actual simulator. Therefore, all distance units are in cm. Behaviors not used are not mentioned.

F.1.1 Avoid

Name	Value
Weight	0.4
SideMinDistance	50.0
SideStartDistance	300.0
FrontMinDistance	200.0
FrontStartDistance	600.0
FrontStartDistancePri1	1000.0
FrontStartDistancePri2	1200.0

F.1.2 Road Tangent

Name	Value
Weight	0.9

F.1.3 Keep Inside Road

Name	Value
Weight	0.6
nrSecondsLookahead	1.0

F.1.4 Avoid Oncoming

Name	Value
Weight	0.5
Margin	100.0
NoEffectDistanceFromEdge	300.0
StartDecayDistanceFromEdge	400.0

F.1.5 Avoid Prioritized

Name	Value
Weight	0.2
AvoidMinDistance	50.0
AvoidStartDistance	225.0
FrontDistance	2000.0

F.1.6 Keep Right

Name	Value
Weight	0.2
Scaling	750.0
Exponent	2.0

F.1.7 Cohesion

Name	Value
Weight	0.01
MaxDistance	6000.0

F.1.8 Avoid (On Entrance Ramp)

Name	Value
Weight	0.4
SideMinDistance	25.0
SideStartDistance	300.0
FrontMinDistance	100.0
FrontStartDistance	400.0
FrontStartDistancePri1	800.0
FrontStartDistancePri2	1000.0

F.1.9 On Ramp

Name	Value
Weight	0.9

F.1.10 On Ramp Waiting

Name	Value
Weight	1.0
MarginBehind	2000.0
MarginInFront	4000.0
OnRampTimeToMeetingPoint	6.027

F.2 PID Gains

The following is the values for the gains and limits of the PID controllers used.

F.2.1 Throttle PID Controller

Name	Value
Proportional	0.005
Integral	0.002
Derivative	0.003
Limit	1.0

F.2.2 Steering Wheel PID Controller

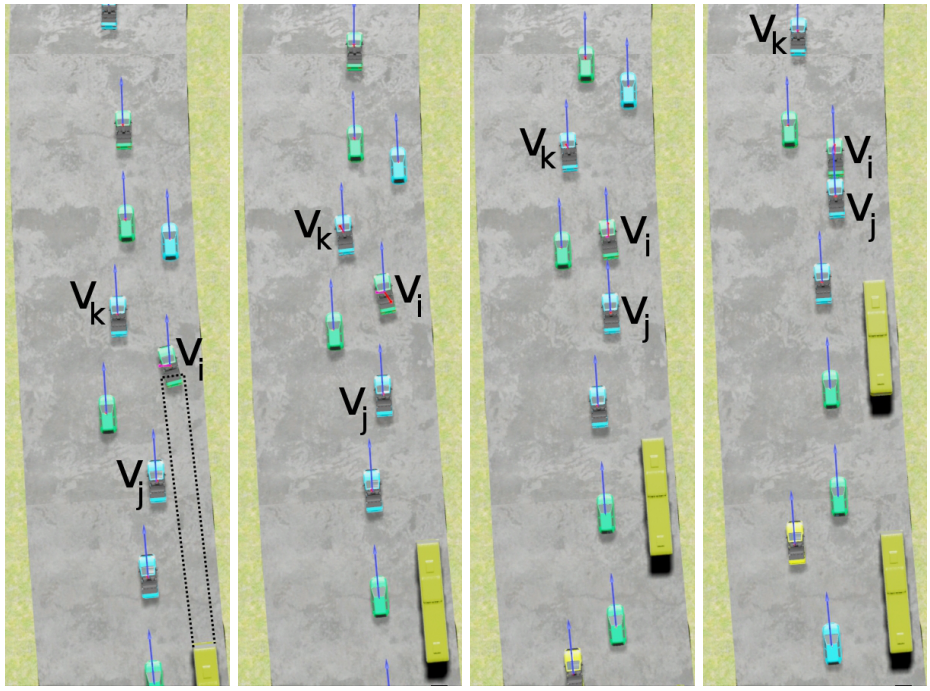
Name	Value
Proportional	1.5
Integral	0.1
Derivative	0.1
Limit	0.3

Appendix G

Harsh Braking Incidents

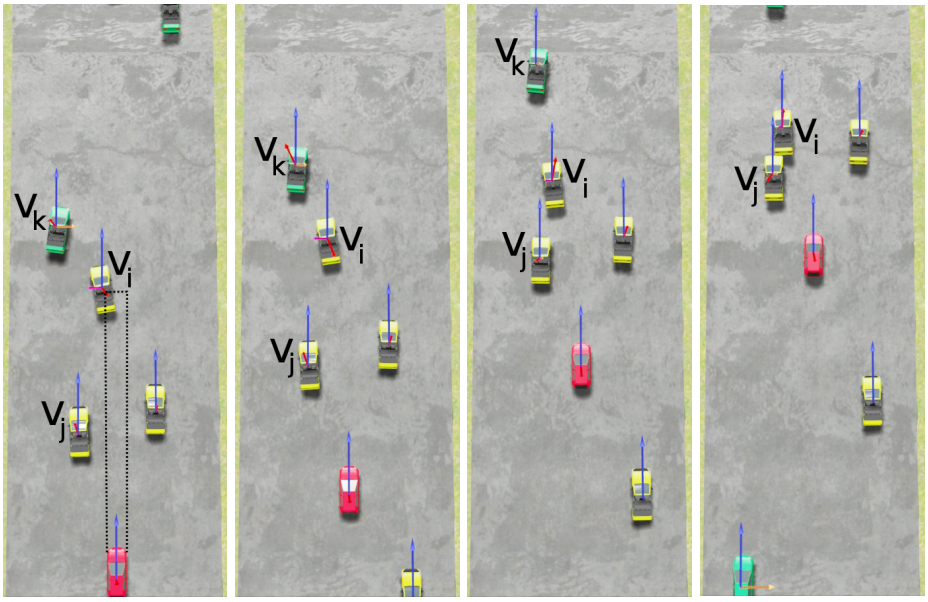
The harsh braking problem is a problem caused by occasional harsh braking by the PID controller, which causes incidents. In this Appendix, we show some of the scenarios that arose during the experiments.

Figure G.1 shows an incident from experiment 4, Figure G.2 shows an incident from experiment 5, and Figure G.3 shows an incident from Experiment 7.



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

Figure G.1: *Vehicle v_i ends up colliding with v_j due to harsh braking. v_i initially steers left to yield to a bus(Figure G.1a), then starts to brake to avoid v_k (Figure G.1b), then tries to accelerate again (Figure G.1c) but is too slow and an incident occurs (Figure G.1d).*



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

Figure G.2: *A vehicle v_i is steering left to avoid a reserved area in front of an emergency vehicle, while at the same time a vehicle v_k is steering right to avoid an oncoming vehicle (Figure G.2a). v_i starts to brake to maintain a safety margin to v_k but brakes too much (Figure G.2b). The PID controller realizes that v_i is driving too slow and starts to accelerate (Figure G.2c), but it is too late and v_i can not accelerate quickly enough and ends up in a rear-end collision with v_j (Figure G.2d)*

(a) **(b)** **(c)**

Figure G.3: *A variation of the harsh braking incident from experiment 7 with symmetric road traffic. v_k is braking to avoid a blue vehicle in front (Figure G.3a) and v_i is braking to avoid v_k (Figure G.3b), the braking is too harsh and v_i suffers a rear-end collision from v_j (Figure G.3c).*

Appendix Bibliography

ACM (1947). Acm digital library. `http://dl.acm.org/`. Accessed: 19.01.2015.

Aoyagi, M. and Namatame, A. (2006). Dynamics of emergent flocking behavior. In El Yacoubi, S., Chopard, B., and Bandini, S., editors, *Cellular Automata*, volume 4173 of *Lecture Notes in Computer Science*, pages 557–563. Springer Berlin Heidelberg.

Armstrong, J. (2010). Erlang. *Commun. ACM*, 53(9):68–75.

Astengo-Noguez, C. and Sánchez-Ante, G. (2007). Collective methods on flock traffic navigation based on negotiation. In Gelbukh, A. and Kuri Morales, ., editors, *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 52–60. Springer Berlin Heidelberg.

Au, T.-C. and Stone, P. (2010). Motion planning algorithms for autonomous intersection management. In *AAAI 2010 Workshop on Bridging The Gap Between Task And Motion Planning (BTAMP)*.

Axelrod, R. M. (1997). *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton University Press.

Balch, T. and Arkin, R. C. (1995). Motor schema-based formation control for multiagent robot teams. In *In Proceedings of the First International Conference on Multi-Agent Systems*, pages 10–16. AAAI Press.

Balch, T. and Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939.

Balch, T. and Hybinette, M. (2000). Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 73–80 vol.1.

Bayazit, O. B., ming Lien, J., and Amato, N. M. (2002). Better group behaviors in complex environments using global roadmaps. In *In Artif. Life*, pages 362–370.

BenAmar, F. (1997). Steering behaviour and control of fast wheeled robots. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 3, pages 1396–1401 vol.3.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Campo, A., Nouyan, S., Birattari, M., Groß, R., and Dorigo, M. (2006). Negotiation of goal direction for cooperative transport. In *Proceedings of the 5th International Conference on Ant Colony Optimization and Swarm Intelligence*, ANTS'06, pages 191–202, Berlin, Heidelberg. Springer-Verlag.

CiteSeer^x (2008). Citeseer^x. `http://citeseerx.ist.psu.edu/index`. Accessed: 19.01.2015.

Croitoru, A. (2009). Deriving low-level steering behaviors from trajectory data. In *Data Mining Workshops, 2009. ICDMW '09. IEEE International Conference on*, pages 583–590.

Cui, X., Gao, J., and Potok, T. E. (2006). A flocking based algorithm for document clustering analysis. *Journal of Systems Architecture*, 52(8–9):505 – 515.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Dresner, K. and Au, T.-C. (2006). Aim: Autonomous intersection management - project home page. `http://www.cs.utexas.edu/~aim/`. Accessed: 29.01.2015.

Dresner, K. and Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third INTERNATIONAL Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 530–537. IEEE Computer Society.

Dresner, K. and Stone, P. (2005). Multiagent traffic management: An improved intersection control mechanism. In *In The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477.

Dresner, K. and Stone, P. (2006). Traffic intersections of the future. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 1593. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Dresner, K. and Stone, P. (2008). A multiagent approach to autonomous intersection management. *J. Artif. Int. Res.*, 31(1):591–656.

Dresner, K. M. (2009). *Autonomous intersection management*. PhD thesis, University of Texas.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY.

El hadouaj, S., Drogoul, A., and Espié, S. (2001). How to combine reactivity and anticipation: The case of conflicts resolution in a simulated road traffic. In *Proceedings of the Second International Workshop on Multi-agent Based Simulation*, MABS 2000, pages 82–96, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Ercan, M. and Li, X. (2011). Swarm robot flocking: An empirical study. In Jeschke, S., Liu, H., and Schilberg, D., editors, *Intelligent Robotics and Applications*, volume 7102 of *Lecture Notes in Computer Science*, pages 495–504. Springer Berlin Heidelberg.

Espitia, H. and Sofrony, J. (2011). Path planning of mobile robots using potential fields and swarms of brownian particles. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 123–129.

ETSI (2013). Etsi ts 102 894-2 v1.1.1 (2013-08). The European Telecommunications Standards Institute, `http://www.etsi.org/deliver/etsi_ts/102800_102899/10289402/01.01.01_60/ts_10289402v010101p.pdf`. Accessed: 14.05.2015.

ETSI (2014). Etsi en 302 637-2 v1.3.2 (2014-11). The European Telecommunications Standards Institute, `http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf`. Accessed: 11.02.2015.

Ferrante, E., Turgut, A. E., Mathews, N., Birattari, M., and Dorigo, M. (2010). Flocking in stationary and non-stationary environments: A novel communication strategy for heading alignment. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*, PPSN'10, pages 331–340, Berlin, Heidelberg. Springer-Verlag.

Flierl, G., Nbaum, D. G., Levin, S., and Olson, D. (1999). From individuals to aggregations: the interplay between behavior and physics. *J. Theor. Biol*, pages 397–454.

Goethem, A. v. (2012). A stream algorithm for crowd simulation to improve crowd coordination at all densities. Master's thesis, Department of computer science Utrecht University, The Netherlands, Buys Ballot Laboratorium, Princetonplein 5, room 574, 3584 CC Utrecht.

Google (2004). Google. `https://scholar.google.no/`. Accessed: 21.01.2015.

Guy, S. J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 177–187, New York, NY, USA. ACM.

Hartman, C. and Beneš, B. (2006). Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4):199–206.

Helbing, D. (2001). Traffic and related self-driven many-particle systems. *Reviews of modern physics*, 73(4):1067.

Hernandez, G. and Welborn, C. (2011). Constrained 3d flocking behavior. *J. Comput. Sci. Coll.*, 27(2):29–36.

Ho, C., Nguyen, Q., Ong, Y.-S., and Chen, X. (2010). Autonomous multi-agents in flexible flock formation. In Boulic, R., Chrysanthou, Y., and Komura, T., editors, *Motion in Games*, volume 6459 of *Lecture Notes in Computer Science*, pages 375–385. Springer Berlin Heidelberg.

Ibáñez, J., Gómez-Skarmeta, A. F., and Blat, J. (2003). Dj-boids: Emergent collective behavior as multichannel radio station programming. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 248–250, New York, NY, USA. ACM.

IEEE (1946). Ieee xplore digital library. `http://ieeexplore.ieee.org/Xplore`. Accessed: 20.01.2015.

Inada, Y. (2001). Steering mechanism of fish schools. *Complexity International*, 8:1–9.

ISI (1960). Isi web of knowledge. `http://apps.webofknowledge.com/`. Accessed: 23.01.2015.

Kala, R. and Warwick, K. (2012). Planning autonomous vehicles in the absence of speed lanes using lateral potentials. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 597–602.

Kamolov, R., Machado, P., and Cruz, P. (2013). Musical flocks. In *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pages 93:1–93:1, New York, NY, USA. ACM.

Kaufman, H., Knep, B., Francois, A. O., Galyean, T. A., and Koumbis, S. (1998). Virtual fishtank. In *ACM SIGGRAPH 98 Conference Abstracts and Applications*, SIGGRAPH '98, pages 295–, New York, NY, USA. ACM.

Kennedy, J. and Eberhart, R. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505.

Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S. (1992). Preface. In *Artificial Life: Santa Fe Institute Studies in the Sciences of Complexity*, volume 5. Addison-Wesley.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, New York, NY, USA.

Liekens, A. (2003). Artificial life. In *Encyclopedia of Computer Science*, pages 93–96. John Wiley and Sons Ltd., Chichester, UK.

Lien, J.-M., Bayazit, B., Sowell, R., Rodriguez, S., and Amato, N. (2004). Shepherding behaviors. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 4159–4164 Vol.4.

Little, J. D. (1961). A proof for the queuing formula: L= λ w. *Operations research*, 9(3):383–387.

Martins, T., Machado, P., and Rebelo, A. (2013). The garden of virtual delights: Virtual fauna for a botanical garden. In *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pages 26:1–26:1, New York, NY, USA. ACM.

Nieuwenhuisen, D., Kamphuis, A., and Overmars, M. H. (2007). High quality navigation in computer games. *Sci. Comput. Program.*, 67(1):91–104.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420.

Oyekan, J. and Hu, H. (2013). Ant robotic swarm for visualizing invisible hazardous substances. *Robotics*, 2(1):1–18.

Parrish, J. K., Viscido, S. V., and Grünbaum, D. (2002). Self-organized fish schools: an examination of emergent properties. *The biological bulletin*, 202(3):296–305.

Parsopoulos, K. and Vrahatis, M. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306.

Ploeg, J., van der Knaap, A., and Verburg, D. (2002). Ats/agv-design, implementation and evaluation of a high performance agv. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 127–134 vol.1.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1):33–57.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782.

Rodrigues, R., de Lima Bicho, A., Paravisi, M., Jung, C., Magalhães, L., and Musse, S. (2009). Tree paths: A new model for steering behaviors. In Ruttkay, Z., Kipp, M., Nijholt, A., and Vilhjálmsson, H., editors, *Intelligent Virtual Agents*, volume 5773 of *Lecture Notes in Computer Science*, pages 358–371. Springer Berlin Heidelberg.

Rodriguez, S., Giese, A., Amato, N. M., Zarrinmehr, S., Al-douri, F., and Clayton, M. J. (2012). Environmental effect on egress simulation. In *in Proc. of the 5th Intern. Conf. on Motion in Games, 2012, Lecture Notes in Computer Scien ce (LNCS.*

Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A.-G., and Prusinkiewicz, P. (2005). Modeling and visualization of leaf venation patterns. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 702–711, New York, NY, USA. ACM.

Sahu, B. and Subudhi, B. (2014). The state of art of autonomous underwater vehicles in current and future decades. In *Automation, Control, Energy and Systems (ACES), 2014 First International Conference on*, pages 1–6.

Sahu, B., Subudhi, B., and Dash, B. (2012). Flocking control of multiple autonomous underwater vehicles. In *India Conference (INDICON), 2012 Annual IEEE*, pages 257–262.

ScienceDirect (1997). Sciencedirect. `http://www.sciencedirect.com/`. Accessed: 22.01.2015.

Shao, W. and Terzopoulos, D. (2005). Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28. ACM.

Shiffman, D. (2012). *The Nature of Code*. The Nature of Code, 1 edition.

Singh, S., Kapadia, M., Faloutsos, P., and Reinman, G. (2009). Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20(5-6):533–548.

Springer (1842). Springer link. `http://link.springer.com/`. Accessed: 21.01.2015.

Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Toner, J. and Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Phys. Rev. E*, 58:4828–4858.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 43–50, New York, NY, USA. ACM.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. C. (2008). Interactive navigation of multiple agents in crowded environments. In *SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES*, pages 139–147. ACM.

van den Bergh, F. (2001). *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science, University of Pretoria, Private Bag X20, Hatfield, 0028, South Africa.

VanMiddlesworth, M., Dresner, K., and Stone, P. (2008). Replacing the stop sign: Unmanaged intersection control for autonomous vehicles. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '08, pages 1413–1416, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Vaughan, R., Sumpter, N., Henderson, J., Frost, A., and Cameron, S. (2000). Experiments in automatic flock control. *Robotics and Autonomous Systems*, 31(1–2):109 – 117.

Vinoski, S. (2007). Concurrency with erlang. *Internet Computing, IEEE*, 11(5):90–93.

Viscido, S. V., Parrish, J. K., and Grünbaum, D. (2004). Individual behavior and emergent properties of fish schools: a comparison of observation and theory. *Marine Ecology Progress Series*, 273(239-249):271–272.

Wang, H., Kearney, J., Cremer, J., and Willemsen, P. (2005). Steering behaviors for autonomous vehicles in virtual environments. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pages 155–162.

Warren, C. (1989). Global path planning using artificial potential fields. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 316–321 vol.1.

Watson, N., John, N., and Crowther, W. (2003). Simulation of unmanned air vehicle flocking. In *Theory and Practice of Computer Graphics, 2003. Proceedings*, pages 130–137.

Whiting, J. S., Dinerstein, J., Egbert, P. K., and Ventura, D. (2010). Cognitive and behavioral model ensembles for autonomous virtual characters. *Computational Intelligence*, 26(2):142–159.

Wiley (1807). Wiley publishing. `http://onlinelibrary.wiley.com/`. Accessed: 23.01.2015.

Willemsen, P., Kearney, J., and Wang, H. (2003). Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments. In *Virtual Reality, 2003. Proceedings. IEEE*, pages 79–86.

Wu, Q., Ji, Q., Du, J., and Li, X. (2013). Simulating the local behavior of small pedestrian groups using synthetic-vision based steering approach. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '13, pages 41–50, New York, NY, USA. ACM.

Xiong, N., Vasilakos, A., Yang, L., Pedrycz, W., Zhang, Y., and Li, Y. (2010). A resilient and scalable flocking scheme in autonomous vehicular networks. *Mobile Networks and Applications*, 15(1):126–136.