Abstract

According to the need of Inst. of Industrial and Control Engineering this Final Grade Project will draw a roadmap to achieve the final objective of simulating full vehicular networks. Following this roadmap, the project will focus on the initial steps of this process.

First step will be to study the current state of the vehicular networks framework and types of simulator involved (both communications and traffic simulators).

After the initial study is done, the project will focus on the first technical step of the roadmap. First a deep study of the SUMO software will be done in order to fully comprehend its functioning and utilities.

After the use of SUMO has been mastered, the focus will move to the implementation of the vehicular control system. Once the control has been implemented, the first experiments will be conducted with the appropriate result analysis.
## Contents

**ABSTRACT** ................................. 1  
**CONTENTS** .................................. 3  
1. **GLOSSARY** ............................... 5  
2. **PREFACE** ................................. 6  
   2.1. Project motivation .......................... 7  
   2.2. Memory briefing ............................ 7  
3. **INTRODUCTION** ......................... 9  
   3.1. Objectives ................................. 9  
   3.2. Scope of work .............................. 9  
   3.3. Tools employed ......................... 10  
4. **PROBLEM STATEMENT** ............... 11  
   4.1. Problem roadmap ....................... 11  
   4.2. Vehicular networks ..................... 11  
      4.2.1. Introduction .......................... 11  
      4.2.2. Networking properties ............... 12  
      4.2.3. Networking challenges ............... 13  
      4.2.4. Safety applications .................. 15  
      4.2.5. Comfort applications: ............... 17  
      4.2.6. Development briefing: ............... 18  
      4.2.7. Adaptive cruise control .......... 19  
      4.2.8. Cooperative adaptive cruise control .. 20  
   4.3. Simulators ............................... 21  
      4.3.1. General description .................. 21  
      4.3.2. Network simulators ................. 21  
      4.3.3. Traffic simulators ................. 22  
5. **SUMO TOOLS AND FUNCTIONALITY** 24  
   5.1. General description ................... 24  
   5.2. Installation process ................... 24  
   5.3. Road networks implementation ....... 24  
   5.4. Vehicle definition ..................... 27  
   5.5. Demand modelling ..................... 29  
   5.6. Simulation execution ................. 30
5.7. Car following models ................................................................. 33
  5.7.1. Intelligent Driver Model .................................................... 35
5.8. Data extraction ........................................................................ 36

6. SIMULATION PROCESS AND RESULTS .............................. 38
  6.1. Model ...................................................................................... 38
    6.1.1. Track .................................................................................. 38
    6.1.2. Vehicles .............................................................................. 38
    6.1.3. Car following models .......................................................... 39
  6.2. Implementation of the model .................................................... 41
    6.2.1. Track .................................................................................. 41
    6.2.2. Vehicles .............................................................................. 42
    6.2.3. Car following model implementation ..................................... 42
  6.3. Execution of the simulations and results .................................. 46
    6.3.1. Initial parameters and tests .................................................. 46
      6.3.1.1. Numerical error test ......................................................... 46
      6.3.1.2. Positional effect test ....................................................... 48
    6.3.2. Speed oscillations ............................................................... 51
    6.3.3. Fundamental diagram .......................................................... 54

7. BUDGET ................................................................................... 59

8. ENVIRONMENTAL IMPACT .................................................. 60

CONCLUSIONS ................................................................. 61

ACKNOWLEDGEMENTS ..................................................... 62

BIBLIOGRAPHY ....................................................................... 63
  References .................................................................................... 63
  Complementary bibliography .......................................................... 64
1. Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Antilock Brake System</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation System</td>
</tr>
<tr>
<td>IVC</td>
<td>Inter-Vehicular Communication</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad-hoc Network</td>
</tr>
<tr>
<td>RSU</td>
<td>Road Side Unit</td>
</tr>
<tr>
<td>R2V</td>
<td>Road To Vehicle</td>
</tr>
<tr>
<td>SUMO</td>
<td>Simulation of Urban Mobility</td>
</tr>
<tr>
<td>TIS</td>
<td>Traffic Information System</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>V2V</td>
<td>Vehicle To Vehicle</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicular Ad-hoc Network</td>
</tr>
<tr>
<td>VEINS</td>
<td>Vehicles in Network Simulation</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
</tbody>
</table>
2. Preface

Nowadays traffic has become one of the most challenging milestones of contemporary societies, for it’s essential to development but carries important issues that can’t be overlooked. On road mortality is one of the most critical problems traffic faces, for it has to both deal with driver causes, such as distractions mainly due to phone usage, alcohol consumption and stress among others; and with causes directly related to the vehicle’s functioning and environment. Aside from this, traffic congestion and jams are another critical issue to face. In medium to large scale cities traffic jams are a day to day reality that while hindering the routine and accounting for massive time loses, serious pollution levels on huge city’s peripheries are directly derived from them. These problems coped with the essentiality of land traffic communications, are gradually degrading quality of life in modern industrialized societies.

With the major problems of traffic being stated scientific community has poured great amounts of effort to try and solve them, or at least mitigate them. Some proposed solutions have already been implemented and proven to be efficient in terms of prevention, such as sound warning lanes, dangerous situation dynamic warnings (both on board and on road), driving assistance systems, security systems (air-bag, extensible bumpers, ABS, etc.) and last generation infrastructure both aimed at increasing comfort and security, among others.

Some of the previously mentioned solutions are a part of a larger group of on board control systems, whose development has been possible due to cutting edge technological advancements. Passive systems such as ABS or positioning systems like GPS are some examples of this category, whose main goal is to keep the driver informed of the vehicle’s conditions and so preventing potentially dangerous situations as well as increasing the efficiency of conduction. But in order to go a step further, it is the advancements in the mobile communications field that have enabled a true breakthrough. These advancements have given a chance to wireless communications in the vehicular environments, where high speed of vehicles and rapidly changing topology characteristics make this combination a true challenge.

The recent integration of communications in the vehicular frame has introduced the concepts of Vehicular Networks and Intelligent Transportation Systems (ITS). These new applications aim at achieving a significant improvement on the previously mentioned problems that traffic is facing. Research community has focused on these concepts and this field is currently undergoing a constant development of new applications and promising ideas.
2.1. Project motivation

In the context of growing interest towards Vehicular Networks and ITS, the Inst. of Industrial and Control Engineering has a line of research concerning the development of a Cooperative Adaptive Cruise Control (CACC), an upgrade to previous Adaptive Cruise Control (ACC) systems that includes Inter-Vehicular Communications (IVC). Design and implementation of a CACC system requires a lot of testing, data extraction and validation of the work done, and to this end simulation is of the essence. Simulation offers a cheap, quick and infrastructure-less way of testing and gathering useful data basic for the study and development of such systems.

Of this interest, this project is born. The search of a high-performance simulation platform specific for the implementation of a CACC has come across VEINS (Vehicles in Network Simulation), a software that combines a traffic simulator, SUMO; and a networks simulator, Omnet++. VEINS manages the communication between both simulators and achieves a realistic scenario where vehicles receive information of other vehicles and road infrastructure. The first step towards using this software is fully comprehending how the traffic simulation is implemented, and so mastering the use of all the tools SUMO has to offer.

2.2. Memory briefing

The memory of this project follows the standard structure given by the faculty. The content of each chapter and section will be shortly reviewed in the following list:

BLOC 1: PREFACE AND GLOSSARY

Presentation of the problem in a general framework as well as the motivation and precedence. Glossary of terms and abbreviations used throughout the project.

BLOC 2: BODY OF THE PROJECT

Bloc 2 includes:

CHAPTER 3: INTRODUCTION

Objectives and scope of work of this project, as well as tools employed.

CHAPTER 4: PROJECT THEORETICAL BASIS

In this section the concept of VANETs and simulation will be more deeply discussed. Properties, challenges, historical development and the dynamic model of the implemented ACC are some of the topics dealt within this section.
CHAPTER 5: SUMO OVERVIEW

A deep insight on SUMO and its features. This section will detail its functioning as well as everything needed to run a simulation.

CHAPTER 6: SUMO USAGE

This chapter will focus on the implementation of a new Car Following Model as well as the analysis of the data gathered and comparison with previous works.

CHAPTERS 7 AND 8

A brief budget and environmental impact of the project will be included.

APPENDIX:

Includes a complete installation guide for SUMO and all the developed code during the project.
3. Introduction

3.1. Objectives

As it has been mentioned before, the main motivation of this project is to endow the Inst. of Industrial and Control Engineering with the tools to master the usage of VEINS platform in order to be able to run feasible simulations of vehicular networks. To this end, the first step is mastering the usage of SUMO, one of the two softwares that integrate VEINS.

To do this, this work will first begin by introducing the topic of Vehicular Ad-hoc Networks (VANETs) and simulation, and briefing its current state of the art. And only then will move into the use of SUMO itself.

In agreement to this main goal, the following milestones are defined:

- Define the roadmap to achieve the simulation of VANETs required by the Inst. of Industrial and Control Engineering.
- Give a general view on VANETs, their properties, applications and challenges they pose.
- Show the importance of simulation in the VANETs research field.
- Undergo an exhaustive study of SUMO and its components and main functions.
- Give a complete guide of installation and use of SUMO, so that it can be used in the future by the Inst. of Industrial and Control Engineering and any other interested readers.
- Model the experiments to test the first implementation of the ACC controller designed by the Inst. of Industrial and Control Engineering. Implement this model using SUMO and analyse the first results obtained.

3.2. Scope of work

The VANETs world includes knowledge of many different fields, and the study of all these fields would require a massive amount of time and research. This project will simply give a slight overview on the topic, but won’t go into detail on stuff related with communications between vehicles and the protocols employed in these communications.
Relating to VEINS this project will focus on SUMO and how to develop a CFModel for it, but will not discuss or study any subject concerning Omnet++ and the implementation of VEINS itself. Although, a future study of these platforms could surely benefit and expand from the work carried out on this project.

For the development of a CFModel and data treatment, programming languages such as C++ and Python are required. This project will assume the reader has the necessary knowledge to fully understand and if necessary reproduce any of the developed and displayed programmes, thus a guide or explanation on C++ and Python will not be included.

All the mentioned above does not mean that in specific situations some information said to be excluded could be mentioned or briefly explained in some sections to ease the understanding of the general gist, but in no circumstance it will be the focus of the section.

3.3. Tools employed

All elements concerning the employed tools and software used for this project are summed up in the following list:

- Toshiba Tecra i3, 2 GHz and 8 GB RAM
- Windows 10 x64 bits
- Ubuntu
- SUMO 0.29.0 and src 0.30.0 (for developers)
- Minitab (student distribution facilitated by the university)
- Microsoft Office Excel 2007
- Articles and publications mainly from IEEE. See references chapter for further information.
4. Problem statement

4.1. Problem roadmap

To finally achieve the objective of simulating full operating VANETs networks, a set of steps need to be followed. After research on VEINS and the VANETs framework the steps could be defined as follows:

1. Study of the VANETs framework, their properties, challenges and applications.
2. State of the art on simulations, both for communications networks and traffic.
3. Exhaustive study of SUMO, functioning and utilities. Implementation of the control system on SUMO.
5. Exhaustive study of VEINS. Final implementation of the control system by SUMO paired with the vehicular communications network by Omnet++.

In this project the initial study of VANETs and simulations has been carried out. After this, the first technical step has been solved, giving an intensive study of SUMO and an implementation of the ACC control system.

4.2. Vehicular networks

4.2.1. Introduction

As the road transportation and the communications field advance, combined implementations of both were a more than logical consequence. From this combination are Vehicular Ad Hoc Networks, from now on VANETs, born.

VANETs are a subclass of MANET (Mobile Ad Hoc Network), a well-known wireless network family that has grown in importance since the appearance of laptops back in the mid 90’s. MANETs include several different applications such as Smart Phone Ad hoc Networks (SPANs), Internet-based Mobile Ad hoc Networks (iMANETs) and the one we will focus on this section Vehicular Ad hoc Networks (VANETs). As their name implies, each node in a MANET is free to move, thus changing in links between devices is very frequent. They generally consist of a peer-to-peer self-forming network, using protocols such as TCP/UDP and IP. Communication generally occurs in radio frequencies ranging from 30 MHz to 5 GHz. The particularity of VANETs lies in the fact that they are composed by sets of vehicles and occasionally elements from the road infrastructure, and according to this the network mobility...
happens at high velocities, posing a challenge for their implementation. Radio communication between vehicles is known as Inter Vehicular Communication (IVC).

Due to the high functionality of VANETs, they are a fundamental component of the so called Intelligent Transportation Systems (ITS) a growing concept that intends to deal with traffic related problems. ITSs feed on the constantly growing communications technologies science and the opportunities they offer. ITSs are defined as systems that collect, store, process and distribute information related with transportation of goods and people. Their main goal is to use technology to increase safety on the road and safe both time and resources.

In the following section, after briefing the Vehicular Network historical development, network properties of VANETs will be explained with more depth and some attractive applications and services deriving from these properties will be discussed. This section will conclude by introducing some future work and challenges.

4.2.2. Networking properties

MANETs have no fixed infrastructure and instead rely on ordinary nodes to perform routing of messages and network management functions. However, Vehicle Ad Hoc networks behave in fundamentally different ways than the models that predominated MANET research. Driver behaviour, constraints on mobility, and high speeds create unique characteristics in IVC networks. These characteristics have important implications for design decisions in these networks. The major differences are as follows:

Rapid changes in the VANETs topology are difficult to manage. Due to high relative speed between cars, networks topology changes very fast and the links and connections between nodes with it. Researchers tried to find approximation of a link's lifetime and tried to find trajectory duration through a typical highway scenario using simulation. Although their results could be useful, they are applicable just in very specific scenarios. In the meantime, MANETs focus on networks with slower moving nodes with more durable links, considerably easing
the process of implementation.

- The IVC network is subject to frequent fragmentation, even at a high rate of IVC deployment. Although the connectivity characteristic of MANETs has been studied broadly, there is little research which tries to tackle this problem. It is mostly because VANET's connectivity depends on the scenario. Scientific effort has tried to capture some relationships between the model of vehicular mobility and connectivity of the networks, but since the results are oriented to specific purposes and cannot be broadly applied.

- The IVC network has small effective network diameter. Rapid changes in link's connectivity cause many paths to disconnect before they can be utilized. This characteristic is important for mostly comfort application as they need to establish unicast and multicast routes (e.g., to the internet gateway).

- No significant power constraints, unlike sensor and other types of mobile networks where limited battery life is a major concern.

- Potentially large-scale: In a city centre or highways at the entrance of big cities the network topology could include hundreds of thousands of nodes.

- Variable Network density: the network's density depends on vehicular density which is highly variable. In a jam scenario the network reaches elevated levels of density, while in suburban and interurban environments the space between nodes radically grows.

- The topology of the network could be affected by driver's behaviour due to his/her reaction to the messages. In other words the content of messages can change network's topology. For example, if a driver is informed of a jam in a certain position, he and potentially all drivers that received the same message will avoid that position. Thus affecting directly the network space distribution.

4.2.3. Networking challenges

Reasonably enough it can be assumed that these messages will be broadcasted by the vehicle, for the benefit of those surrounding it and itself. This comes at a cost, and poses numerous challenges for a proper broadcasting:

- Media Access Control (MAC) Layer Issues: from the definition given before, event driven messages must have the highest priority, for they are generated as a result of immediate threats. For this reason, mechanisms for message differentiation and admission control are vital. These mechanisms should prioritize messages ragging
from the event driven ones, as the most important; followed by beaconing safety messages and finally comfort messages. These mechanisms depend on the MAC layer policy, thus developing a standardized MAC layer for VANETs is the first step. To this day, IEEE 802.11 offers the best on road performance and is the chosen solution by the ASTM (American Society for Testing and Materials. Other viable options are ADHOC MAC or Directional Antenna Based MAC Protocols (D-MAC).

- **Message Dissemination**: Due to safety messages characteristics, broadcasting is the only option for message exchange in order to get complete coverage to all relevant vehicles. Message forwarding can transmit information beyond a node’s transmission range. But in reality, not all vehicles will be equipped with wireless transceivers, so emergency message forwarding becomes an even greater challenge for the industry. To solve this, some proposals are exploiting motion properties of vehicles, introduction of protocols to reduce the amount of forwarding messages, context-aware packet forwarding protocols for intra-platoon scenarios and some algorithms to help vehicles limit the effects of broadcast storm.

- **Clustering**: clustering neighbouring nodes into more manageable units is of the utmost importance if efficient and reliable safety communications are to be achieved. If no boundaries are drawn between vehicles:
  
  - Too many vehicles interfere with each other saturating the radio bandwidth transmissions.
  
  - Messages may propagate in every direction indefinitely, flooding the networking system with information.

Conventional clustering strategies may not be effective to form efficient vehicle clusters, since in a vehicle network nodes may be densely populated and lined on roadways. To solve this Local Peer Groups (LGPs), a novel grouping method for VANETs has been proposed, and it’s variant of static and dynamic LGPs.

- **Power assignment**: The exchange of messages between nodes forms a network topology, independently of the MAC type. This topology is constantly changing, both in radio channel and positional characteristics. The term Offered Traffic is defined as the density of active users per unit area, and greatly affects the network’s topology. The variance of Offered Traffic is well known to cause node or cluster isolation when the density is reduced. To cope with this low density scenario increasing the transmission power is an effective method to increase range and thus facilitate nodes to communicate in lightly populated networks. On the other hand, if the vehicle encounters itself in a high density environment, such as a huge city peripheries or a
jam, nodes compete for radio transmission resources and the average amount of radio capacity per user is excessively reduced. In this situation, increasing power would only make matters worse, and so the opposite solution, reducing transmission power, becomes the adequate since it reduces the amount of nodes competing for the radio channel.

And so transmission power becomes a key parameter to solve this problem. A fixed transmission power for the nodes will cause them to find few neighbours when traffic is low, or an excessive number of them if traffic is high. Adjusting transmission power adaptively, by reducing it in densely populated environments and increasing power when the number of neighbours is small, a node jointly copes with the isolation problem at low load and the limited system capacity at high load. Even though channel capacity and power control are broadly studied, the particularity of having safety as main goal brings VANETs new constraints not considered before. For example, previous studies proposed algorithms aim at minimizing energy consumption and maximize the overall throughput. But as previously mentioned in this work, energy consumption is of no importance in VANETs since nodes are assumed to have an unlimited supply. Aside from this, previous work also makes some unfeasible assumptions for VANETs: 1) all nodes are static and 2) all nodes use the same transmission power.

To solve this problem different approaches have been proposed. A first approach would aim at making sure that nodes close to the sender receive its messages with high probability while ensuring fairness in the overall system. Another one would take advantage of the relation between network density and traffic flow, and from this develop an algorithm that estimates density of vehicles without message exchanging, only by using flow theory. A third option would be a power control algorithm based only on local information, adjusting transmission power to set the number of neighbours at a target range, reducing power if it's surpassed or increasing power if number of neighbours falls.

4.2.4. Safety applications

Examples of vehicle-to-vehicle safety communication include collision warning, road obstacle warning, cooperative driving, intersection collision warning, lane change assistance, sudden stop warning, and all vehicle related information that could somehow improve driver’s ability to safely lead.

Safety messaging can be narrowed down to two basic types, classified depending on how they are generated. First type would be event driven, resulting from the detection of an unsafe situation, such as a car crash or nearby high speeding vehicles for example. Second group of messages is integrated by all the ones that are generated periodically, also known as beaconing. This group accomplishes several functions but when in terms of safety they
can be seen as preventive messages intended to make vehicles aware of their environment. With these, vehicles should be able to avoid and anticipate emergency or unsafe situations. Beaconing generally includes information regarding the state of the vehicle (position, speed, acceleration, etc) and sometimes can also contain data on the vehicle’s vicinities.

Among all safety messaging applications, the following can be highlighted:

1. Intersection warning: in this event driven example of messaging, the vehicles inside an intersection and/or RSUs present in the vicinities would detect the danger of a lateral collision with other incoming vehicles and inform them of the threat.

2. Lane changing assistance: aiming at reducing lateral collision threat due to blind spots when changing lanes. This application is of special interest for trucks, whose large blind areas cause potentially risky situations when lane changing, as it can be seen in figure 4.2.

3. Overtake assistance: in an event of multiple cars overtaking, there could be a situation when to vehicles start overtaking manoeuvres over the same car at the same time, thus increasing collision change largely. In this case, first car to start manoeuvring would emit a message that when read by a third party vehicle would advise it to stop the overtake as there’s a crashing risk.

4. Possible collision: all kinds of collisions could be prevented if the adequate warning is sent. Frontal collisions would require that a wrong way driving vehicle would emit a presence warning to all surrounding vehicles, increasing the reaction speed of driver. In back collision, the warning message would be sent once the vehicle detects a strong braking, a low visibility situation, wet pavement or any kind of event that would increase the probability of a successor vehicle to crash (Figure: 4.3 ). Another specific type of collision happens when both (or more than two given the case) cars detect the collision threat through a messaging exchange between them; in this situation both vehicles would act simultaneously and coordinated to advise the drivers and prevent a crash.
5. On-going collision assistance: once the collision has been determined as unavoidable vehicles will stop trying to avoid it and focus on preparing for the collision to minimize the damage intake. Vehicles and RSUs will be constantly beaconing data on car’s location, speed and weight with the purpose of enabling and optimizing the vehicle’s equipment in a collision event. Such equipment may include airbags, adjustable safety belts and extensible bumpers among others.

6. Emergency brake warning: vehicles that are forced to perform a strong sudden brake, will message other nodes using RSUs or other vehicles as channel.

7. Wrong way warning: mentioned before, if a car is driving along a wrong way road it will be messaging other vehicles and RSUs around to warn them of this situation.

8. Dangerous spot notification: notification of especially dangerous spots in the road network where extreme caution is advised. These spots may include obstacles, sliding roads, closed turns or civil works among others.

9. Loss of control warning: in an event where the driver lost control of the vehicle, this would send a warning signal to surrounding nodes so that they can take the appropriate decisions.

10. Emergency vehicle call: after a collision, vehicles itself or with the aid of RSUs would contact local authorities and emergency services automatically and decrease the time taken to help everyone involved.

4.2.5. Comfort applications:

This group includes applications aiming at improving traffic flows, coordinating vehicle routes, on road assistance, updated local map information, etc. Generally it’s narrowed down to speed control (also Cruise Control) which will be more deeply discussed in following chapters of this work and Cooperative Navigation. But due to the comfort purposes of other applications involving public service or information and entertainment, the following list has been extended including all non-safety related applications, or at least some of them:
1. Cooperative navigation: these are used to increase traffic efficiency through cooperative beaconing between vehicles and RSUs. Some examples of this could be traffic information, recommended routes and cooperative adaptive route control among others.

2. Local authorities support: in order to ease the job of local security department vehicles could include tools to identify stolen cars or detect drivers without license for example.

3. Smart Cities: this idea has recently been introduced and has multiple applications in a VANETs framework. Information on local hot spots, position of charging stations for electric vehicles, tour guides, parking management, etc. Are just a few examples of how VANETs could help to implement this new concept.

4. Internet service: by installing access points along the road or implementing mobile data transfer protocols in vehicles. These could enjoy a reliable connection to internet, even broadening the possibilities of all kinds of applications and services offered to the driver and passengers.

5. Company support: for companies with a large vehicle fleet, VANETs could offer a chance to manage and control the distribution and following of their assets.

4.2.6. Development briefing:

The idea of wireless car communication has been discussed by scientists since the 80s decade. But due to recent communications field outbreaks and modernization of technologies the effort on VANETs field has recently been greatly increased and as a result its development has accelerated.

Key factors in this development have been the adoption of IEEE 802.11 technologies (mentioned before in the MAC Layer Issues section); the automotive industry tendency towards information technologies to cope with issues involving traffic security, environment pollution and comfort, as well as the compromise of governments in assigning a specific radio electric bandwidth so that IVC can take place.

Since the late 90s, tools like GPS or Wireless Local Area Network (WLAN) transmitters are at everyone’s disposal. This has become a fundamental factor so that great numbers of projects have focused on the field of IVC. In these projects the potential of VANETs has been explored by many cooperating agents, from the automobile industry to companies operating the traffic network, with many other parties involved. All of it, generally funded by local governments, whose contribution is also reflected when facilitating licences for the bandwidth distribution, generally allocating the spectrum of 5,8/5,9 GHz to IVC.
Figure 4.4 depicts the main milestones in VANETs development around the world. Due to the great number of projects involved in this field, only the most transcendent initiatives have been included in this chart, ranging from the beginning to the consolidation of the VANET concept.

4.2.7. Adaptive cruise control

In the past recent years Advanced Driver Assistance Systems (ADAS) have drawn a lot of interest and research, being Adaptive Cruise Control (ACC) one of the most representative systems of this group.

ACC is a direct upgrade of the Cruise Control (CC), a system mainly designed to keep the vehicle at a certain speed generally chosen by the driver. This was an entirely comfort application, as it did not offer any kind of advantage in security terms. ACC on the other hand, instead of maintaining a fixed speed, is designed to maintain a safety distance with the preceding vehicle. Thus it includes sensors to determine leader’s speed, distance, etc.; these sensors can be laser, radar or camera based.

ACC system can be divided into three functional modules. First one would be the environmental perception and processing module, with the purpose of obtaining environment information in front of the vehicle and vehicle dynamic information using the available sensors. A second functional module, would determine the safety distance with the
preceeding vehicle using the previously gathered information. Finally, a third module is in charge of actuating accordingly to what has been observed and processed into the vehicle dynamics.

These systems can carry string stability issues that can cause problems when oscillations in the traffic flow occur. Leading to phantom traffic jams or head-tail collisions.

![Block diagram of an ACC controller](image)

**Fig. 4.5.** Block diagram of an ACC controller [2]

### 4.2.8. Cooperative adaptive cruise control

Cooperative Adaptive Cruise Control (CACC) systems are basically ACC systems that include V2V or R2V communication. Besides the sensor measurements, CACC vehicles include information obtained from the communication with other vehicles. With this all vehicles can perform cooperative control manoeuvres, more accurate and with less delay than in ACC systems. This upgrade on the ACC systems, offers string stability, solving the previously mentioned problems.

Apart from that, a comparison made by Qing Xu and Raja Sengupta and explained in [3] shows that when compared in equal conditions, CACC always outperforms ACC. It also explains that while an aggressive controller increases average speed and efficiency, a weaker controller can save braking effort making the system safer and more comfortable; thereby showing the adaptability of CACC. Another significant conclusion is that if market was to include more ACC and CACC systems in vehicles, this would favour the overall average speed of the traffic network and reduce braking effort.

Another benefit of CACC over ACC is the ability to reduce headway time. In densely occupied highways capacity is a limiting factor, and thus reducing the space between vehicles is crucial for increasing capacity or even decreasing drag force (which in consequence increases efficiency). In paper [4] Jeroen Ploeg implements a CACC system and determines that these type of systems can give headway time values of significantly less than 1 second while maintaining string stability. This would result in a significant increase in highway capacity and decrease of fuel consumption and emissions of heavy-duty vehicles can also be expected.
4.3. Simulators

4.3.1. General description

As it has been mentioned before, the particularities of VANETs make it unviable to apply certain traditionally standardized protocols of communication. Due to this, and the huge impact that VANETs may have in the automotive industry it is understandable that a lot of effort is currently been put to develop communication protocols and mobility models specific for VANETs.

While it's fundamental to test and evaluate these newly developed protocols in a real environment, there are obvious logic difficulties to do this. Economic problems, technologic limitations and unpredictable issues are the fundamentals causes that complicate the testing in real scenarios. This is why simulation offers a comfortable and cheap way to validate the newly developed communication protocols.

For the proper simulation of VANETs the system must include two types of simulators, mobility model simulator plus a network simulator. With both of them working co-ordinately.

Mobility simulators are a critical aspect, since they are needed to reproduce a realistic behaviour that reflects vehicle traffic. In particular, this mobility models need to be reconfigurable, since the effect of IVC must be reflected somehow on road traffic. To this end, the scientific community has started working towards developing mobility models specific for VANETs simulations. After years of development, there are now a larger number of models, fitted for different purposes and levels of detail, but all of them designed to simulate realistic VANETs mobility models.

This project particularly focuses on one of these simulators, SUMO. An open source traffic simulator for microscopic and continuous traffic simulations.

Aside from this type of simulators, if VANETs are to be fully simulated one must also take into account network simulators that will be charged with the V2V and R2V communications.

4.3.2. Network simulators

Although it is not in the scope of work of this project, it’s true that for a final goal of achieving full VANETs simulations network simulators will have to be eventually studied and used. This is why this section will briefly summarize their purpose and functionality.

As in any other field, in the context of networks for data transmission testing with prototypes and laboratories can be expensive, and the tools employed may not be reconfigurable for future use. In addition, reproducing some network factors such as interferences is
complicated. This is why network simulators offer a great opportunity to efficiently and economically run tests.

Network simulators are used to study the effect of different parameters on the network’s performance and response. They generally offer a wide range of network technologies and communication protocols. These simulators give tools to build complex networks through the use of basis blocs representing network elements, such as routers, hubs, switches, computers, etc. With the help of a simulator the user will normally be able to study multiple types of network technologies like TCP, ATM, IP, Ethernet, etc. and apply different standards to the constructed networks to analyse its performance and results.

4.3.3. Traffic simulators

In order to summarize the state of the art and explain how traffic simulators work a division of four main types of mobility models is due. These four types gather the necessary data for the model from different sources and in consequence treat it in a particular way. The following are these four main models:

1. Synthetic models

These models try to understand a particular movement, and then draw a mathematical model from it with the final objective of artificially reproduce this movement. Methods employed include from completely stochastic models to behavioural models where every movement is determined by behaviour rules. Others use hydro dynamical models or even FIFO (First In First Out) queues with cars as clients.

These mathematical models are then validated through comparison with data gathered from real traffic environments. On occasions though, certain movements and interaction between them makes the mathematical models a too complex tool to implement.

2. Data based models

Real data gathering is an invaluable source of information on macroscopic mobility. If large amounts of data and all the statistics drawn from them are included in a mobility model, the result would be a generic model able to reproduce pseudo-aleatory behaviour observed in a real urban traffic environment.

3. Trace based models

The idea behind these models is opposite to the one synthetic models use. While synthetic models develop complex mathematical models and then validate them through real mobility traces, trace based methods directly extract generic mobility patterns from the real traces.
This method faces the problem that some movement patterns are not observed through data traces, and through the use of mathematical models it is only up to a certain degree that these hidden patterns can be predicted.

4. Traffic simulators based models

Using the synthetic models and refining them with an intense process of validation based on real traces and behavioural data the development of realistic traffic simulators has been possible.

Some resulting traffic simulators are FreeSim, STRAW (Street Random Waypoint), TEXAS (Traffic Experimental and Analytical Simulation) and the one that will be deeply studied in this project SUMO (Simulations of Urban MObility).
5. Sumo tools and functionality

5.1. General description

Simulation of Urban Mobility [5], SUMO, is an open source microscopic traffic simulation package capable of handling large scale road networks. It is currently undergoing development that reflects in constant new distributions and add-ons. The project started by employees of the Institute of Transportation Systems at the German Aerospace Centre.

SUMO offers an enormous amount of possibilities for traffic simulations; it can generate all kinds of vehicles as well as pedestrians and bikes. It offers the chance to programme road elements such as bus stops, traffic lights, multiple lanes, junctions, parking, etc... This project has merely used the basic elements of this software, which has been more than enough to reproduce the previously performed simulations and extract the necessary data.

The three main elements necessary to run a simulation are a road network, vehicles that will enter the simulation and the routes this vehicles will follow. In the following chapter these elements and some other tools, parameters and procedures used to launch the simulations later studied will be detailed.

5.2. Installation process

The common user installation process is described in http://sumo.dlr.de/wiki/Installing, for the installation of developers packages with source code, please refer to this project’s Appendix, where a complete guide on SUMO installation is included.

5.3. Road networks implementation

An important part of SUMO is designing the geometry of the traffic network. SUMO is a node-edge based simulator, meaning that the networks are created by straight lanes connecting two nodes. This might sound simplistic but with the right collocation and scale it can achieve a very close approximation to a real network.

SUMO offers different ways of constructing a road network. one is creating and modifying networks with NETEDIT a program that allows the user to create the road network manually inserting the elements through a graphic interface. This tool can be useful for as an introduction or to visualize already created networks, but it’s impractical when having to create huge roads with lots of elements. Other ways involve importing networks with NETCONVERT, and again there exist different options for network importing; the most
relevant ones are listed below:

- **Defining own networks using XML**

This method will require the creation and editing of various XML files, each of them containing the necessary information for NETCONVERT to create a network usable by SUMO. The main necessary elements are:

1. **Nodes:**

They will be detailed in a .nod.xml file. A generic node is described in a single line and looks like: `<node id="<STRING>" x="<FLOAT>" y="<FLOAT>"/>`. It can include further attributes, but for the development of this project they are irrelevant. The meaning and identification of the employed attributes are listed below:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id (string)</td>
<td>Name of the node, can include numbers and alphanumeric characters</td>
</tr>
<tr>
<td>x</td>
<td>float</td>
<td>X position of the node in meters</td>
</tr>
<tr>
<td>y</td>
<td>float</td>
<td>Y position of the node in meters</td>
</tr>
<tr>
<td>z (optional)</td>
<td>float</td>
<td>Z position of the node in meters (0 as default value)</td>
</tr>
</tbody>
</table>

As example, a .nod.xml file could look like:

```
<nodes> <!-- The opening tag -->
  <node id="0" x="0.0" y="0.0" z="0.0"/> <!-- def. of node "0" -->
  <node id="1" x="-500.0" y="0.0"/> <!-- def. of node "1" -->
  <node id="2" x="500.0" y="0.0"/> <!-- def. of node "2" -->
  <node id="3" x="0.0" y="-500.0"/> <!-- def. of node "3" -->
  <node id="4" x="0.0" y="500.0"/> <!-- def. of node "4" -->
</nodes> <!-- The closing tag -->
```

2. **Edges:**

Edges are described in a .edg.xml file. A generic single edge description may look like this: `<edge id="<STRING>" from="<NODE_ID>" to="<NODE_ID>" [type="<STRING>" [numLanes="<INT>"] [speed="<FLOAT>"]/>. Attributes listed between claudators are optional and can be omitted. It can include further attributes, but for the development of this project they are irrelevant. The meaning and identification of the employed attributes are listed below:
To describe an edge, it can be done manually for each edge specifying every attribute needed. Another option is to define an edge type with all its attributes and reference that type through the edge, thus saving some time. This types are defined in type files that will be explained later.

As an example, an edge file .edg.xml would look like:

```xml
<edges>
  <edge id="1" from="0" to="1" numLanes="1" speed="11.11" />
  <edge id="2" from="1" to="2" numLanes="5" speed="22.22" />
  <edge id="3" from="2" to="3" type="a" />
  <edge id="4" from="0" to="4" type="2" />
</edges>
```

As it can be observed, a node can be the star and/or end to multiple edges. In the example above edges 1 and 2 are described in the same edge. Edges 3 and 4 on the other hand, will be described in a type file, where their respective types will have all the necessary attributes with their values.

3. Type descriptions

The description of each type is included in a .typ.xml file. This file is constructed in a similar fashion as the edges file, only that in this case the type element does not include the `from="<NODE_ID>" to="<NODE_ID>"` attributes. Aside from those, it can include every single attribute an edge has, with the same value and meaning.

A type file .typ.xml for example could look like:

```xml
<types>
  <type id="a" numLanes="3" speed="13.889"/>
  <type id="2" numLanes="2" speed="11.111"/>
</types>
```
This method for defining road networks has been employed for the development of this project. It gives full control over the construction of the network, but for large scale maps it could be impractical.

Only basic attributes needed for this project simulation are explained above. Apart from these SUMO offers multiple other parameters to edit nodes, edges and lanes. For further interest on them refer to http://sumo.dlr.de/wiki/Networks/Building_Networks_from_own_XML-descriptions.

Once the .nod.xml, .edg.xml and .typ.xml (if needed) files are created, user has to execute the program NETCONVERT to create the network readable by SUMO. This is done by accessing the Bin folder through the terminal and calling:

```
netconvert --node-files="file.nod.xml" --edge-files="file.edg.xml" --type-files="file.typ.xml" --output-file="file.net.xml"
```

This will read the specified node, edge and type files and return a network file. The --output=files” option is not required but it gives the chance to name the network file beforehand.

- **Importing non-SUMO networks**

SUMO also offers the possibility to import networks from other formats. An interesting option worth mentioning is from OpenStreetMap, which allows the conversion of a real map into the SUMO network format, this allows for a fast conversion of large scale road networks. This options though have not been used in this project and will thus not be explained with any more detail. Information about this can be found at http://sumo.dlr.de/wiki/Networks/Import.

5.4. Vehicle definition

A vehicle in SUMO consists of three parts: a type which describes all its physical properties, a route the vehicle will follow and the vehicle itself. This section will explain how to define a vehicle and its physical properties, and leave the routes for the next section.

Vehicles and vehicle types are included in a .rou.xml file. A vehicle type description should look like: ```<vType id="<STRING>" accel="<float>" decel="<float>" minGap="<float>" maxSpeed="<float>" lenght="<FLOAT>" etc... />``` , while a
vehicle description looks like: `<vehicle id="<STRING>" type="<TYPE_ID>" route="<ROUTE_ID>" depart="<float>" etc... />`.

The following table explains some of the vehicle type attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id (string)</td>
<td>-</td>
<td>Name of the vehicle type</td>
</tr>
<tr>
<td>accel</td>
<td>float</td>
<td>2.6</td>
<td>Acceleration ability of the vehicle in m/s²</td>
</tr>
<tr>
<td>decel</td>
<td>float</td>
<td>4.5</td>
<td>Deceleration ability of the vehicle in m/s²</td>
</tr>
<tr>
<td>tau</td>
<td>float</td>
<td>1.0</td>
<td>CFModel parameter. Desired headway time.</td>
</tr>
<tr>
<td>length</td>
<td>float</td>
<td>5.0</td>
<td>Length of the vehicle</td>
</tr>
<tr>
<td>minGap</td>
<td>float</td>
<td>2.5</td>
<td>Standstill distance</td>
</tr>
<tr>
<td>maxSpeed</td>
<td>float</td>
<td>70.0</td>
<td>Maximum speed of the vehicle in m/s</td>
</tr>
<tr>
<td>color</td>
<td>RGB-color</td>
<td>&quot;1,1,0&quot; (yellow)</td>
<td>Vehicle’s type colour (only for SUMO-GUI)</td>
</tr>
<tr>
<td>imgFile</td>
<td>filename</td>
<td>-</td>
<td>Image file for rendering vehicle’s type in SUMO-GUI</td>
</tr>
<tr>
<td>carFollowModel</td>
<td>CFModel name</td>
<td>&quot;Krauss&quot;</td>
<td>The model used for car following</td>
</tr>
</tbody>
</table>

Table. 5.3. Basic vehicle type attributes

Vehicle attributes are listed in the next table:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id (string)</td>
<td>Name of the vehicle</td>
</tr>
<tr>
<td>type</td>
<td>id</td>
<td>Id of the type to use for this vehicle</td>
</tr>
<tr>
<td>route</td>
<td>id</td>
<td>Id of the route the vehicle will follow</td>
</tr>
<tr>
<td>depart</td>
<td>float</td>
<td>Time step at which the vehicle will enter the road network</td>
</tr>
<tr>
<td>departSpeed</td>
<td>float</td>
<td>Speed at which the vehicle will enter the simulation (default to 0)</td>
</tr>
</tbody>
</table>

Table. 5.4. Basic vehicle attributes

Once this attributes or some of them are put together, the part concerning vehicles and vehicle types of a .rou.xml file should look as follows:

```
<routes>
  <vType id="type1" accel="2.6" decel="4.5" length="5" maxSpeed="22.22" tau="1.3"
    carFollowModel="IDM" minGap="2.0" color="0,1,1"/>
  <vType id="type2" accel="3.4" decel="5.5" length="3.9" maxSpeed="33.33"
    carFollowModel="ETSEIB" minGap="1.0" color="1,1,1" imgFile="image.png"/>
  <vehicle id="0" type="type1" route="route0" depart="0" />
  <vehicle id="1" type="type2" route="route0" depart="0" departSpeed="10.0"/>
</routes>
```
To complete the .rou.xml file there’s a last element to include that will be detailed in the following section.

### 5.5. Demand modelling

In order to define through which nodes is a vehicle going to move the route it has to follow must be specified. There are two main ways of doing this: defining a complete route for the vehicle or creating a vehicle flow. Complete routes look like `<route id="<string>" edges="<edges id>" />` and flows are written like `<flow id="<string>" begin="<float>" end="<float>" number="<int>" from="<edge id>" to="<edge id>" />`. These routes will have to be included in the .rou.xml file, the same where vehicles and vehicle types are declared.

Attributes for route definition are:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id (string)</td>
<td>Name of the route</td>
</tr>
<tr>
<td>edges</td>
<td>id list</td>
<td>Edges the vehicle shall drive along. Ids must be separated by spaces and the edges they refer to must be connected</td>
</tr>
<tr>
<td>color</td>
<td>RGB colour</td>
<td>This route’s colour</td>
</tr>
</tbody>
</table>

Flow definition uses the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id (string)</td>
<td>Name of the flow</td>
</tr>
<tr>
<td>begin</td>
<td>float</td>
<td>First vehicle departure time in s</td>
</tr>
<tr>
<td>end</td>
<td>float</td>
<td>Time to stop introducing vehicles into the simulation, in s</td>
</tr>
<tr>
<td>vehsPerHour</td>
<td>float</td>
<td>Number of vehicles introduced per hour, equally spaced</td>
</tr>
<tr>
<td>period</td>
<td>float</td>
<td>Insert equally spaced vehicles at that period</td>
</tr>
<tr>
<td>probability</td>
<td>float([0,1])</td>
<td>Probability of introducing a vehicle every second</td>
</tr>
<tr>
<td>number</td>
<td>int</td>
<td>Total number of vehicles to include in the simulation</td>
</tr>
<tr>
<td>via</td>
<td>id list</td>
<td>List of edges the vehicle will follow. Only useful if there’s more than one path from begin to end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table. 5.5. Route attributes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Table. 5.6. Flow attributes</th>
</tr>
</thead>
</table>
It has to be noted that vehPerHour, period and probability cannot coexist in the same flow as they describe three different ways of introducing vehicles to the simulation.

Routes and flows added in the .rou.xml file will look somewhat like the following example:

```xml
<routes>
  <route id="route0" color="1,1,0" edges="beg middle end rend"/>
  <flow id="f2" begin="0" end="100" number="23" from="beg" to="end" via="e1 e23 e7"/>
</routes>
```

In this project finally the option of routes has been the chosen one. This way the initial position of every car can be adjusted and they can all start at the same time, two important factors that are unachievable with flows. As counterpart this option makes the simulations more cumbersome to design.

5.6. Simulation execution

Once the .net.xml and .rou.xml files are confectioned, the last step before launching a simulation will be creating the additional (.add.xml) and configuration files (.sumo.cfg).

The additional file includes the rerouting algorithm, which is charged with keeping the vehicles driving in circles not letting them leave the track. The rerouter is placed in a circuit node and it basically changes the route of the vehicles stepping on it to a new destination. The rerouter has three separated parts, first defines the rerouting with an id and its location. Second part defines the interval time it will be operative, here if no begin attribute is defined it will start from second 0 and will end when specified. And a final part defines the new destination. The following example code is the .add.xml file with the employed rerouter in the 100 node track simulations:

```xml
<additionals>
  <rerouter id="rerouter_0" edges="2526">
    <interval end="1e9">
      <destProbReroute id="7677"/>
    </interval>
  </rerouter>
  <rerouter id="rerouter_1" edges="7576">
    <interval end="1e9">
      <destProbReroute id="2627"/>
    </interval>
  </rerouter>
</additionals>
```
Once all the previous files are constructed, the configuration file will be in charge of calling them when the simulation is launched. Configuration file defines the input and output of the simulation. Among the input files are the ones explained before (Network, Routes and additional file), and the output files will be the ones containing the data extracted from the simulation. The output files will be explained in section 5.7.

An example configuration file could be as follows:

```xml
<configuration>
    <input>
        <net-file value="networkfile.net.xml"/>
        <route-files value="routesfile.rou.xml"/>
        <additional-files value="additionafile.add.xml"/>
    </input>
    <output>
        <fcd-output value="fd0idm.xml"/>
    </output>
</configuration>
```

After the configuration file is build, the last step will be launching the simulation. The applications SUMO or SUMO-GUI are in charge of this. SUMO can only be invoked through the command line and will simply run the simulation and give the required output. SUMO-GUI is the graphical interface of SUMO and can be opened by double clicking it or also through the command line. When calling any of these applications through the command line, user has the opportunity to change simulation parameters such as the time step or the duration of the simulation. Next table includes some of the parameters changed this way and how to invoke them:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--begin</td>
<td>float (s)</td>
<td>0 s</td>
<td>When shall the simulation begin</td>
</tr>
<tr>
<td>-b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--end</td>
<td>float (s)</td>
<td>-</td>
<td>When shall the simulation end, if not specified the simulation will continue until manually stopped</td>
</tr>
<tr>
<td>-e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--step-length</td>
<td>float (s)</td>
<td>1 s</td>
<td>Duration of each time step in seconds.</td>
</tr>
<tr>
<td>-c</td>
<td>config file id</td>
<td>-</td>
<td>Id and location of the configuration file to read</td>
</tr>
</tbody>
</table>

Table. 5.7. Simulation launch parameters
A complete call to SUMO or SUMO-GUI would look like:

```
sumo-gui -e 150 -step-length 0.1 -c ~/<CONFIG LOCATION>/configurationfile.sumo.cfg
or
sumo -b 10 -e 150 -step-length 0.01 -c ~/<CONFIG LOCATION>/configurationfile.sumo.cfg
```

If called through SUMO, this is an example of what the user will see:

The command line will display every time step in a line and then show the message: “Loading configuration... done.” This means there was no error during the simulations, and the output files will be found where specified.

When instead user calls SUMO-GUI the simulation will not be automatically done, but the next window will pop up. In this window the green arrow button (Play) will launch the simulation and let the user see the movement of the vehicles.

This option gives the chance to visually follow the vehicles movement and behaviour on the track. It is useful to detect error causing spots or non-functioning vehicles. Although once the simulations have been tested and there is no further interest in visually capturing them, sumo can save a lot of time in front of sumo-gui in terms of simulation running.
5.7. Car following models

The vehicles introduced in the SUMO simulations are governed by control laws programmed in the Car Following Models. These models are complex C++ programs whose methods determine the behaviour of the vehicle. During this project only the method followSpeed, which is in charge of controlling the car’s speed when driving under regular circumstances, has been studied and developed for the implemented CFModel.

Other methods such as stopSpeed (determines the actions of the vehicle in case of a predecessor’s sudden stop) or laneChanging (controls speed when the vehicle is changing lanes) have not been implemented for they were not relevant for the experiment and the full comprehension and development of them would require a much deeper research.

Some of the included CFModels in SUMO are the Krauss following model and some variants of it, the Intelligen Driver Model (IDM) by Martin Treiber or the Wiedemann car following model among others.

To develop a new CFModel it’s advisable to begin with an already created one, and once all the identifications have been changed inside the source code, modify it to suit the user requirements. To accomplish this, the next steps are vital:

1. Copying an existing model:

First step will be accessing <SUMO_HOME>/src/microsim/cfmodels where all the already existing models are located. Then simply pick one and change the name of the .cpp and .h file. For example copying MSCFModel_KraussOrig1.h and MSCFModel_KraussOrig1.cpp and rename them MSCFModel_ETSEIB.h and MSCFModel_ETSEIB.cpp. Now open these files and rename all occurrences of MSCFModel_KraussOrig1 into MSCFModel_ETSEIB.

2. Define a new model name in <SUMO_HOME>/utils/common/SUMOXMLDefinitions.h and .cpp

When the .h file is accessed the tag where all CFModels tags are located (DO NOT ADD IT IN THE LAS POSITION OF THE LIST!!!):

```
SUMO_TAG_CF_ETSEIB,
```

In SUMOXMLDefinitions.cpp:
Furthermore you need to add an entry to SUMOXMLDefinitions::carFollowModelValues[] also in SUMOXMLDefinitions.cpp:

```cpp
{ "ETSEIB", SUMO_TAG_CF_ETSEIB },
```

3. Define model parameters:

The parameters SUMO will read from the CFModel are defined in `<SUMO_HOME>/src/util/xml/SUMOVehicleParserHelper::getAllowedCFModelAttrs()`. If there are no extra parameters for the CFModel it can simply inherit the ones from an already existing model, otherwise user will have to redefine them.

Inheritance method:

```cpp
std::set<SumoXMLAttr> krausParams;
    krausParams.insert(SUMO_ATTR_ACCEL);
    krausParams.insert(SUMO_ATTR_DECEL);
    krausParams.insert(SUMO_ATTR_APPARENTDECEL);
    krausParams.insert(SUMO_ATTR_EMERGENCYDECEL);
    krausParams.insert(SUMO_ATTR_SIGMA);
    krausParams.insert(SUMO_ATTR_TAU);
allowedCFModelAttrs[SUMO_TAG_CF_KRAUSS] = krausParams;
allowedCFModelAttrs[SUMO_TAG_CF_KRAUSS_ORIG1] = krausParams;
allowedCFModelAttrs[SUMO_TAG_CF_KRAUSS_PLUS_SLOPE] = krausParams;
allowedCFModelAttrs[SUMO_TAG_CF_ETSEIB] = krausParams;
```

New definition method:

```cpp
std::set<SumoXMLAttr> ETSEIBParams;
    ETSEIBParams.insert(SUMO_ATTR_ACCEL);
    ETSEIBParams.insert(SUMO_ATTR_DECEL);
    ETSEIBParams.insert(SUMO_ATTR_APPARENTDECEL);
    ETSEIBParams.insert(SUMO_ATTR_EMERGENCYDECEL);
    ETSEIBParams.insert(SUMO_ATTR_SIGMA);
    ETSEIBParams.insert(SUMO_ATTR_TAU);
allowedCFModelAttrs[SUMO_TAG_CF_ETSEIB] = ETSEIBParams;
```
4. Add code that selects the new class based on the new name in microsim/MSVehicleType::build()

Car-following models are instantiated in MSVehicleType::build(...) located in <SUMO_HOME>/src/microsim/MSVehicleType.cpp. You'll find a switch here, where goes the call to the model's constructor.

```cpp
case SUMO_TAG_CF_ETSEIB:
    vtype->myCarFollowModel = new MSCFModel_ETSEIB(vtype, accel, decel, emergencyDecel, apparentDecel, sigma, tau);
```

5. (Optional) Defining new CFModel parameters:

If the new CFModel included some extra parameters not already defined, user will have to add them to <SUMO_HOME>/src/utils/xml/SUMOXMLDefinitions. For example in SUMOXMLDefinitions.h:

```cpp
SUMO_ATTR_TMP1,
SUMO_ATTR_TMP2,
SUMO_ATTR_TMP3,
```

And in SUMOXMLDefinitions.cpp:

```cpp
{ "tmp1", SUMO_ATTR_TMP1 },
{ "tmp2", SUMO_ATTR_TMP2 },
{ "tmp3", SUMO_ATTR_TMP3 },
```

After this process is completed the user will have to access the terminal and recompile SUMO using the following command on the <SUMO HOME> directory:

```bash
make
sudo make install
```

If there are no errors after this, the new CFModel has successfully been implemented and it's now ready to be modified and used.

5.7.1. Intelligent Driver Model

In order to maintain the maximum similarity between the experiments undergone in this
The project and the previously performed by the Inst. of Industrial and Control Engineering, the
chose CFModel to simulate the human driving vehicles is the IDM model. This model was
developed by Martin Treiber and explained in [6].

This model’s parameters are the vehicle type with all its attributes, a headway time named Tau and other internal parameters that do not have to be specified. The followSpeed is
determined by a function that computes a safe speed that respects the safe gap between the
vehicle and its predecessor and compares it with the maximum speed achievable by
respecting the acceleration limits. It then returns the smallest of them every time step.

The code of the IDM model will be added in the appendix.

5.8. Data extraction

The objective of the simulations is no other than gathering data for a later study, and of
course SUMO offers multiple tools to extract different types of information and in different
formats from the simulations.

In order to generate the output data files, they must be defined in the .sumo.cfg file, in the
output section, as mentioned in section 6.5. There are some pre-established output file types
each one with a particular call, and all of them written in xml format. The ones employed in
this project and some other types that could be useful for further study are listed in the table
below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Output Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw vehicle positions and speed dump</td>
<td><code>&lt;netstate-dump value=&quot;filename&quot;/&gt;</code></td>
<td>Gives the information of the vehicles organized by: list of time steps, list of edges in each time step, list of lanes for every edge, information of every car in each lane. For each vehicle it gives the id, position (x,y) and speed.</td>
</tr>
<tr>
<td>Vehicle type probe</td>
<td><code>&lt;vTypeProbe id=&quot;probe1&quot; type=&quot;vehTypeid&quot; freq=&quot;float&quot; file=&quot;filename&quot;/&gt;</code></td>
<td>This option gives information of cars belonging to the chosen type. Also it gives the opportunity of writing the data not every time step but following a frequency. This output organizes the information per time steps, listing all the vehicles that match the desired type for that time step. On each vehicle it gives information of lane it occupies, its position on the lane, x and y coordinates, latitude and longitude coordinates and the cars’ speed.</td>
</tr>
<tr>
<td>FCD output</td>
<td><code>&lt;fcd-output value=&quot;filename&quot;/&gt;</code></td>
<td>This output file has the data organized in time steps and a list of vehicles for each time step. Of each vehicle it gives the id, x and y coordinates, angle in reference to an x constant line, the vehicle’s type and the speed.</td>
</tr>
</tbody>
</table>
Trajectories output

This output it's the only available one that gives acceleration data on the vehicles. Although the acceleration displayed is computed by subtracting the previous time step speed to the present speed and dividing by the time step length. Aside from that it has no further interest for this project.

For this project, the employed type of output has been the FCD output. Not only offers all the required data, but also does it in a simple way without many levels of information, making it an easy to parse file.

This is an example of the output file obtained by calling the FCD:

```xml
  <timestep time="0.00">
    <vehicle id="76" slope="0.00" lane="7778_0" pos="4.00" speed="0.00" type="idm76" angle="84.61" y="-0.95" x="173.29"/>
    <vehicle id="79" slope="0.00" lane="8081_0" pos="4.14" speed="0.00" type="etseib79" angle="73.81" y="4.58" x="203.17"/>
    <vehicle id="95" slope="0.00" lane="9697_0" pos="4.08" speed="0.00" type="etseib95" angle="16.19" y="113.43" x="313.24"/>
  </timestep>
  <timestep time="1.00">
    <vehicle id="76" slope="0.00" lane="7778_0" pos="5.12" speed="1.12" type="idm76" angle="84.61" y="-0.85" x="174.40"/>
    <vehicle id="79" slope="0.00" lane="8081_0" pos="4.14" speed="0.00" type="etseib79" angle="73.81" y="5.06" x="204.86"/>
    <vehicle id="95" slope="0.00" lane="9697_0" pos="5.35" speed="1.28" type="etseib95" angle="16.19" y="114.66" x="313.59"/>
  </timestep>
  ... further time steps...
</fcd-export>
```

Table. 5.8. Output files
6. Simulation process and results

6.1. Model

6.1.1. Track

The model track employed during the simulations tries to approximate a circumference of a perimeter of 1km. This model is built by connecting 100 nodes placed along the edge of a circle. The position in the horizontal plane of these nodes is defined by the following equation:

\[
(x, y)_n = \frac{500}{\pi} \left( \cos \left( n \frac{2\pi}{100} \right), \sin \left( n \frac{2\pi}{100} \right) \right)
\]  
(Eq. 6.1)

With n=0,1,2,…99. A radius of \( \frac{500}{\pi} \) m forces a perimeter of 1 km, which will ease the calculus of the traffic density during the study. 100 nodes will enable the introduction of 100 vehicles equidistantly distributed in the circuit which will allow to control the percentage of ACC controlled cars and their position easily.

For the road parameters, the maximum speed of the road

6.1.2. Vehicles

Vehicle’s dynamics will be modelled using the standard SUMO method. There will be two types of vehicle models, one will intend to reproduce human driving cars while the second one will model ACC controlled vehicles. Vehicle physical parameters have been taken following previously performed experiments [7].

For the human vehicles these are the used parameter and their distributions:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Mean Value</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration limit</td>
<td>1.3 m/s</td>
<td>0.1</td>
</tr>
<tr>
<td>Deceleration limit</td>
<td>3.5 m/s</td>
<td>0.4</td>
</tr>
<tr>
<td>Length</td>
<td>3.9 m</td>
<td>0.1</td>
</tr>
<tr>
<td>Tau (desired time headway)</td>
<td>1.3 s</td>
<td>0.1</td>
</tr>
<tr>
<td>Minimum Gap (standstill distance)</td>
<td>2 m</td>
<td>0.2</td>
</tr>
</tbody>
</table>
For the ACC controlled cars the distributions are as follows:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Mean Value</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration limit</td>
<td>1.3 m/s</td>
<td>0.1</td>
</tr>
<tr>
<td>Deceleration limit</td>
<td>3.5 m/s</td>
<td>0.4</td>
</tr>
<tr>
<td>Length</td>
<td>3.9 m</td>
<td>0.1</td>
</tr>
<tr>
<td>Tau (desired time headway)</td>
<td>1.3 s</td>
<td>0</td>
</tr>
<tr>
<td>Minimum Gap (standstill distance)</td>
<td>2 m</td>
<td>0</td>
</tr>
<tr>
<td>Maximum speed</td>
<td>80 km/h</td>
<td>0</td>
</tr>
<tr>
<td>CFModel</td>
<td>ETSEIB</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.2. ACC controlled vehicles' attributes distribution

For the ACC controlled cars, attributes depending on the vehicle itself have been defined following the human driver ones. But for the attributes related to the behaviour towards driving the deviation has been made 0, as all ACC controllers are supposed to be calibrated identically.

### 6.1.3. Car following models

For the human driving vehicles the IDM model predefined by SUMO will be employed.

For the ACC controlled cars the model employed was taken from internal reports of the Inst. of Industrial and Control Engineering [8].

As seen in Fig. 6.1. distance of the ith (with i=1,....,N) vehicle with its predecessor is defined as \( h_i \) and follows the next equation:

\[
h_i(t) = x_{i-1}(t) - x_i(t) + l_{i-1}
\]

(Eq. 6.2)
Where \( x_i \) and \( x_{i-1} \) are the absolute positions of \( i \)th vehicle and its predecessor and \( l_{i-1} \) is the length of the second. Since the objective of an ACC is to adapt \( h_i \) distance to a desired value, this is defined by the following equation:

\[
h_{i,r}(t) = h_o + k_v v_i(t) \tag{Eq. 6.3}
\]

\( h_o \) stands for the desired distance with the preceding vehicle; \( h_0 \) is the standstill distance which refers to the distance the driver will keep with its predecessor if the speed was 0. \( k_v \) is known as constant time headway, equivalent to the desired time that the \( i \)th vehicle should take to arrive at the position of its predecessor at a certain point in time. Finally \( v_i \) stands for the instantaneous speed of the \( i \)th vehicle.

The error then is defined as follows:

\[
e(t) = h_{i,r}(t) - h_i(t) \tag{Eq. 6.4}
\]

Using Eqs. 6.2 and 6.3 and taking the dot operator above the variable as its time derivative \((\dot{x} = \frac{dx}{dt})\) the error dynamics then results in:

\[
\dot{e}(t) = k_v \dot{v}_i(t) - v_{i-1}(t) + v_i(t) \tag{Eq. 6.5}
\]

After reordering, the control algorithm is given by:

\[
v_i(t) = -k_p e_i(t) - k_v \dot{v}_i(t) + v_{i-1}(t) \tag{Eq. 6.6}
\]

This guarantees the following closed loop dynamics:

\[
\dot{e}_i(t) = -k_p e_i(t) \tag{Eq. 6.7}
\]

Where \( k_p > 0 \) guarantees asymptotic stability with a time constant \( \tau = \frac{1}{k_p} \). Finally, combining Eqs 6.2, 6.3 and 6.6, the control law is rewritten as follows:

\[
v_i(t) = \frac{1}{k_v s + (1 + k_p k_v)} (k_p (h_i(t) - h_0) + v_{i-1}(t)) \tag{Eq. 6.8}
\]
With \( s \) as the time derivative operator \( (s = \frac{d}{dt}) \). Linking with previous sections of this project, to develop its function this control needs different types of information. \( h_0 \) and \( v_i \) are normally available within the vehicle, but the way \( v_{i-1} \) is obtained defines whether we are talking about an ACC or a CACC controller.

After obtaining the dynamic model, it must be transformed into a discrete time model, since SUMO runs his simulations on discrete time steps. For this, the time discrete version of Eq. 6.6 would be:

\[
v_i(k) = -k_\rho e_i(k) - k_\omega \dot{v}_i(k) + v_{i-1}(k) \quad \text{(Eq. 6.9)}
\]

Where \( k \) refers to the time step. Then continuous acceleration is approximated by:

\[
\dot{v}_i(t) = \frac{v_i(k) - v_i(k-1)}{T} \quad \text{(Eq. 6.10)}
\]

Where \( T \) refers to the duration of each time step. Now substituting Eq. 6.10 and the error using equation 6.4 into equation 6.9:

\[
v_i(k) = -k_\rho (h_0 + k_\omega (v_i(k) - h_0)) - k_\omega \left( \frac{v_i(k) - v_i(k-1)}{T} \right) + v_{i-1}(k) \quad \text{(Eq. 6.11)}
\]

And finally reordering:

\[
v_i(k) = \frac{1}{1 + \frac{k_\rho}{T} + k_\omega k_\rho} \left[ k_\rho (v_i(k) - h_0) + \frac{k_\omega}{T} v_i(k-1) + v_{i-1}(k) \right] \quad \text{(Eq. 6.12)}
\]

Equation 6.12 defines the time discrete control law for modelling the vehicles speed.

6.2. Implementation of the model

6.2.1. Track

The implementation of the track is done following the instructions given in section 5.2 for the road network construction. All .nod.xml and .edg.xml files of the tracks will be included in the appendix.
The final result of the implementation of the track model on SUMO is shown in the following figure:

![100 node track visualized from SUMO-GUI](image)

**Fig. 6.2.** 100 node track visualized from SUMO-GUI

### 6.2.2. Vehicles

Vehicles and routes have been implemented following the instructions given in sections 5.3 and 5.4.

For creating the vehicle type distributions the tool `createVehTypeDistributins.py` has been employed. The use of this tool as well as other utilities offered by SUMO will be added in the Appendix.

The vehicle resulting files will also be added to the Appendix.

### 6.2.3. Car following model implementation

For the IDM model, the implementation given by SUMO has been employed.

With the model described in section 6.1.3 the implementation on SUMO has posed a complex proces. After creating the new CFModel following the stepd described in section 5.6 to program the new CFModel user will have to edit two files, the .cpp ant the .h they both are written in c++. The first one includes all the methods of the program while the second one defines and calls them. In this section each part of each file will be reviewed using the developed model in this project.

1. **Creating the .cpp file:**

   The .cpp file contains the program methods, that are the main functioning part of a CFModel. The method `.:followSpeed()` will define how the vehicles speed is updated after each time step, thus being the method that must include the control law defined in Eq. 6.1.3.
This file has a first section where the included methods are invoked. The first lines call the configuration .h file while the second part defines the inheritances, in this second part the partner .h file must be invoked as well as any other employed modules in the program.

The second section is composed by the method definition, this part includes an initial class definition with its parameters and inheritances, a second call to the class itself and then it includes all the methods in the class CFModel. Finally it also includes a method that duplicates the class.

As an example the developed CFModel for this project is presented with the mentioned parts highlighted:
double headwayTime = myHeadwayTime;
const double Vant = veh->getSpeed();
double Kv = 1.3;
double Kp = 5;
const double acc = myAccel;
double h0 = myType->getMinGap();
double pas = 1;
const double v = (1 / (Kv / pas + 1 + Kv * Kp)) * (Kp * (gap2pred - h0) + predSpeed + (Kv / pas) * Vant);
const double vmin = 5;
return v;
}

MSCFModel* MSCFModel_ETSEIB::duplicate(const MSVehicleType* vtype) const {
    return new MSCFModel_ETSEIB(vtype, myAccel, myDecel, myEmergencyDecel, myApparentDecel,
    myDawdle, myHeadwayTime);
}

 **************************************************************************/

2. Creating the .h file:

This file defines and references all the methods included in the CFModel class.

It includes a first call to the file, and the it has a section of included modules very much like the .cpp file.

Next it has the class definitions, where it starts by defining the CFModel class and where it inherits from. A constructor and a destructor follow this and then it makes a call to all the methods included in the .cpp file.

As an example the developed .h file for the CFModel used in this project is presented with the main parts highlighted:

#define MSCFModel_ETSEIB_h

//===================================================================
// included modules
//--------------------------------------------------------------------
#ifdef _MSC_VER
#include <windows_config.h>
#endif

#include <windows_config.h>
#else
#include <config.h>
#endif
#include "MSCFModel.h"
#include <utils/xml/SUMOXMLDefinitions.h>
//===================================================================
// class definitions
//===================================================================
* @see MSCFModel */

class MSCFModel_ETSEIB : public MSCFModel {
 public:
 /** @brief Constructor
 * @param[in] accel The maximum acceleration
 * @param[in] decel The maximum deceleration
 * @param[in] emergencyDecel The maximum emergency deceleration
 * @param[in] apparentDecel The deceleration as expected by others
 * @param[in] dawdle The driver imperfection
 * @param[in] headwayTime The driver's reaction time
 */
 MSCFModel_ETSEIB(const MSVehicleType* vtype, double accel, double decel,
 double emergencyDecel, double apparentDecel, double dawdle,
 double headwayTime);

 /// @brief Destructor
 -MSCFModel_ETSEIB();

double followSpeed(const MSVehicle* const veh, double speed, double gap2pred, double
predSpeed, double predMaxDecel) const;

 /** @brief Returns the model's name
 * @return The model's name
 * @see MSCFModel::getModelName */
 virtual int getModelID() const {
 return SUMO_TAG_CF_ETSEIB;
 }

 void setMaxDecel(double decel) {
 myDecel = decel;
 myTauDecel = myDecel * myHeadwayTime;
}
There exist plenty of other methods implemented on SUMO but they are not useful for this project. Some of the methods in the code are not known to perform any function but were kept to avoid rising unwanted error upon deletion.

Every time a CFModel is changed SUMO must be recompiled in order to reflect the changes on the simulations. This is done by using the following command on the <SUMO HOME> directory:

```
make
sudo make install
```

### 6.3. Execution of the simulations and results

#### 6.3.1. Initial parameters and tests

The simulation parameters such as time step duration or the position of cars inside the vehicle have been defined after conducting the following studies:

##### 6.3.1.1. Numerical error test

After the first simulations where performed the analysis of the data showed some unexpected oscillations on the stationary speed. These oscillations appeared for a 100% of ACC controlled cars, which collided with the original idea of a stationary flow.

For a simulation with a density of 40 cars per kilometre, a time step of 1 second and 100% of ACC controlled vehicles the following results were obtained:
From Fig. 6.3 little can be seen, since the speed could be assumed stationary. But after scaling, some seemingly periodic oscillations can be identified. These oscillations could be due to numerical error, in which case they should not be taken into account, or could be due to flow instabilities or other factors related to the ACC controller.

In order to determine what was causing this and how these oscillations evolved 3 tests were run. Firs one expanded the simulation duration to 500 seconds in order to determine the evolution of the oscillations. Here are the results directly scaled:

The second test was to give all the vehicles in the simulation the exact same parameter values, but keeping the duration, time step and density values the same. One of the possible causes of the oscillations was that since vehicle’s acceleration, deceleration and length had been created using statistical distributions; these differences could somehow affect the stationary behaviour of the ACC cars. The results were the following:
The third and last test was to reproduce the first simulation but this time using a Time Step of 0.1 seconds. The intention of this test was to eliminate any possible numerical error produced due to too large time steps. The results are as follows:

With this graphics, different conclusions were drawn. From Fig 6.4 it appears that oscillations tend to stabilize at amplitudes around 0.05 m/s. From Fig. 6.5 can be seen that once all vehicles share the same physical attributes the oscillations are inexistent, even though this is a non-realistic scenario because vehicles will always have different lengths, accelerations and deceleration values. Finally from Fig. 6.6 it appears that the oscillations are gone, leaving only some redoubts of disturbances that not surpass the 0.01 m/s amplitude.

Based on this, it was decided that the best solution was to perform the simulations with a step of 0.1 second; since it proved that the oscillations were due to numerical error and it does not compromise the veracity of the simulation, quite the opposite. Also, it was proven that a 0.1 seconds time step did not reduce the performance of the employed computational tools.

6.3.1.2. Positional effect test

Another important parameter of the road is how the vehicles are distributed along it. If, for
example there is a 50% of ACC controlled cars, placing them altogether or mixed with human vehicles in-between will have different effects on the stationary speed.

To study the effect of the position the following tests were designed. First the number of vehicles in the simulation is fixed at 40 cars thus leaving 20 ACC cars and 20 human drivers, finally two tests would be run placing the vehicles in the two most extreme situations possible. These two extreme situations are the one that keeps ACC cars the closest together and the one that keeps the maximum possible distance between. These two tests were repeated for a 30, 60 and 90 per cent of cars with ACC control and the results are the following:

For a 30% of cars with ACC the maximum distance was obtained by placing the cars following the sequence 1 ACC, 2 human drivers, 1 ACC, 2 human drivers, 1 ACC and 3 human drivers, and repeating this sequence four times. The minimum distance was achieved when placing 12 ACC cars altogether and then 28 human drivers. The graphics obtained are directly scaled to observe the stationary area:

For a 60% of cars with ACC the maximum distance was obtained by placing the cars following the sequence 1 ACC, 1 human drivers, 1 ACC, 1 human drivers, 1 ACC and 1 human driver, 1 ACC, 1 human driver and 2 ACC’s, and repeating this sequence four times. The minimum distance was achieved when placing 24 ACC cars altogether and then 16 human drivers. The graphics obtained are directly scaled to observe the stationary area:
For a 90% of cars with ACC the maximum distance was obtained by placing the cars following the sequence 9 ACC and 1 human driver repeating this sequence four times, while the minimum distance was achieved when placing 36 ACC cars altogether and then 4 human drivers. The graphics obtained are directly scaled to observe the stationary area:

The expected effect of the ACC is the homogenization of the velocity among the traffic flow, decreasing the speed oscillations present in a human driver context. This effect is more pronounced when the ACC controlled cars are distributed among the flow, as it can be seen in Fig. 6.7 and 6.8, where the speed oscillations are far less pronounced when the distance between ACCs is maximized. This can be justified looking at the flow characteristics, when vehicles with ACC controller are left together it creates two differentiated zones in the flow, one with the characteristics of a 100% of ACC cars and one with the characteristics of 0% of ACC cars. This isolates the effect of the controller to a small area, leaving a large group of human drivers that will produce an unwanted speed oscillation. Whereas when equally distributed along the road, the effect of the ACC cars is way more pronounced as they influence smaller groups of human drivers.
Another effect is a modest increase on stationary speed, seen on Fig 6.9, where the stationary speed increases around 0.05 m/s (1.7% increase).

It’s important to remember though, that none of these scenarios is a realistic reflection of reality. In a real environment cars will neither be all of the same type together nor will they have the ACC vehicles equally distributed on the road. For this reason the expected result in a realistic scenario would be a compromise between these two situations. In other words, the more separated controller cars would be in a real environment, the deeper their effect would be on the traffic.

For the simulations carried out in this project, cars with ACC controller have been distributed equally along the road. This decision has been taken in order to keep this factor fixed, and not having to take it into account when studying the results.

6.3.2. Speed oscillations

Once all the simulation parameters have been covered and the CFModel has been implemented the tests to evaluate the performance of the ACC controller may begin. First a study of the stationary speed is due.

Previous experimentation has shown that if put on a circular track, human drivers tend to cause a spring back effect on speed. This means that while some parts of the track will have vehicles running at a high speed, other areas will be in a jam situation with a much lower speed. Also, these areas will move on the opposite way of the speed as they could be roughly approximated by a FIFO queue.

To study this phenomenon the simulations will all have a different percentage of ACC controlled cars. The percentages used will range from 0% to 100% ACC cars with a 10% step, leaving 10 simulations. Tests will run with a constant number of 50 cars in the 100 node track. For each percentage of controlled vehicles the position of the ACC ones will try to maximize the distance between them as concluded in section 6.3.1.2. Time step will be of 0.1 seconds in order to minimize the numerical error as seen in section 6.3.1.1. Vehicles will follow the distributions and parameters explained in section 6.1.2.

After performing all the simulations and treating the obtained data the results show the following:
Fig. 6.10. Mean instantaneous speed for different percentages of human driver vehicles. (original view and scaled)

Fig. 6.11. Instantaneous vehicle speed for 0 % of ACC controlled vehicles

Fig. 6.12. Instantaneous vehicle speed for 10 % of ACC controlled vehicles

Fig. 6.13. Instantaneous vehicle speed for 20 % of ACC controlled vehicles

Fig. 6.14. Instantaneous vehicle speed for 30 % of ACC controlled vehicles
Adaptive Cruise Control and simulation of traffic flow using SUMO

Vehicles speed vs Time step
40 % of ACC controlled cars

Vehicles speed vs Time step
50 % of ACC controlled cars

Vehicles speed vs Time step
60 % of ACC controlled cars

Vehicles speed vs Time step
70 % of ACC controlled cars

Vehicles speed vs Time step
80 % of ACC controlled cars

Vehicles speed vs Time step
90 % of ACC controlled cars

Fig. 6.15. Instantaneous vehicle speed for 40 % of ACC controlled vehicles

Fig. 6.16. Instantaneous vehicle speed for 50 % of ACC controlled vehicles

Fig. 6.17. Instantaneous vehicle speed for 60 % of ACC controlled vehicles

Fig. 6.18. Instantaneous vehicle speed for 70 % of ACC controlled vehicles

Fig. 6.19. Instantaneous vehicle speed for 80 % of ACC controlled vehicles

Fig. 6.20. Instantaneous vehicle speed for 90 % of ACC controlled vehicles

Fig. 6.15. Instantaneous vehicle speed for 40 % of ACC controlled vehicles

Fig. 6.16. Instantaneous vehicle speed for 50 % of ACC controlled vehicles

Fig. 6.17. Instantaneous vehicle speed for 60 % of ACC controlled vehicles

Fig. 6.18. Instantaneous vehicle speed for 70 % of ACC controlled vehicles

Fig. 6.19. Instantaneous vehicle speed for 80 % of ACC controlled vehicles

Fig. 6.20. Instantaneous vehicle speed for 90 % of ACC controlled vehicles
As it can be seen in figures from 6.11 to 6.21, with the increase in the percentage of ACC controlled vehicles, the speed oscillations are stabilized and not only that but they tend to decrease. This means that the spring back effect is eliminated, resulting in a constant flow. The benefits of this range from a power decrease, since vehicles are not subjected to constant breaking and acceleration; to a viable decrease of the distance to predecessor with everything this implies (mentioned in chapter 1).

When the mean speed of all the vehicles in the simulation is analysed (Fig. 6.10) there are less significant differences. While it is true that for percentages ranging from 70 to 100 per cent of human drivers the mean speed suffers a decrease of around 1.5 to 3.5 km/h and oscillations appear; for percentages of human drivers of 60 per cent or lower the mean speed is suffers only a small decrease in front of a completely ACC controlled scenario.

6.3.3. Fundamental diagram

The fundamental diagram is a diagram that explains the statistical relations between macroscopic traffic flow variables density, flow and average speed. It's generally used to predict the capability and behaviour of a road under given circumstances.

Density of a road refers to the number of vehicles per unit of longitude. The average speed is the mean speed of all the vehicles in the road, and the flow is the number of vehicles that would drive through a certain point in the road per unit time. According to the continuity equation the relationships of the three variables are described, and they verify the following rule:

\[ q = k \cdot u \]  \hspace{1cm} (Eq. 6.13)

Where \( k \) is the road density in Veh/km, \( u \) is the average speed in km/h and \( q \) is the flow in Veh/h. From this equation the theoretical fundamental diagram looks like:
Where all the relationships between the three variables are described. The value $k_c$ refers to the critical density. This density divides the diagram in two areas: the free flow zone, where $k<k_c$, and the synchronized flow zone where $k>k_c$. In the first area there is a stable moving flow of vehicles. In the second area the flow becomes unstable, and any sudden breaking of a car could cause a jam. A third final zone could be defined for high traffic densities, higher than the road capacity; this area describes a jam situation.

To determine the theoretical conditions for the critical point for the ACC controlled vehicles some considerations have been made. First, the space of the track is occupied by the length of the vehicles and the gap between them, as described by the next equation:

$$L_{track} = N(l + h_i)$$  \hspace{1cm} (Eq. 6.14)

Where $L_{track}$ is the length of the track, 1 km in this experiment. $l$ is the vehicle’s length, $N$ is the number of cars in the simulation and $h_i$ is the distance between vehicles. Defining density as $k = \frac{N}{L_{track}}$ and using equation 6.14:

$$k = \frac{1}{l + h_i}$$  \hspace{1cm} (Eq. 6.15)

Now considering the equation 6. the following density equation is obtained:

$$k = \frac{1}{l + h_0 + h_s v}$$  \hspace{1cm} (Eq. 6.16)

Understanding the critical point as the transition of free flow to congested flow, the speed can be set at the maximum speed of the controller. Substituting the value of the max speed and the mean values for every parameter in equation 6.16 a theoretical value of the critical
density is obtained:

\[
k = \frac{1}{l + h_0 + h_v} = \frac{1}{3.9 + 2 + 1.3 \times 22.22} = 0.02874 \frac{\text{veh}}{m} = 28.74 \frac{\text{veh}}{\text{km}} \tag{Eq. 6.17}
\]

So a \( k = 28.74 \) Veh/km is the expected critical density for a completely ACC controlled vehicle flow.

To study the effect of the introduction of ACC controlled cars over the fundamental diagram different densities of cars has been simulated on a 100 node track. Since this track has an exact perimeter of 1 km the number of introduced vehicles will directly equal the density \( k \) in Veh/km. Through the study of the output simulation data the average speed of the vehicles will be obtained and using Eq. 6.13 the value of the flow \( q \) is also obtained. This experiment will be repeated for percentages of ACC ranging from 0 % to 100 % with steps of 10 %. The densities are taken from 10 to 100 Veh/km with steps of 10 Veh/km. The results are depicted in the following diagram:

---

**Fig. 6.22.** Fundamental diagram with different percentages of ACC controlled cars

After getting the resulting graphics, it’s hard to determine the exact location of the critical density due to a small amount of data. For this reason in order to get more precise information, a second round of simulations will be executed. This time, simulations will be performed from densities ranging from 10 to 100 Veh/km with steps of 5 Veh/km. Also,
between 30 and 40 Veh/km, where the critical density is expected to be located, a stepping of 1 Veh/km will be applied. Since this second round of simulations cannot be performed without jeopardizing the established percentage of ACC cars, it is only going to be used for the scenarios with 100 % of ACC cars and 100 % of human drivers. With this, an impression of the extremes will be obtained, meaning that any other percentage of ACC cars in a road network will be a compromise between the two extreme results.

With the resulting data the next diagram is obtained:

From this graphics the following conclusions are drawn:

- Cars do not reach the maximum speed.
- Substituting the critical speed on equation 6.16 a critical density of 31 Veh/km is obtained, according to the one graphically obtained.
- Critical density is not affected by the amount of ACC controlled cars on the simulation.
- On the other hand, the critical average speed and critical flow both increase for higher percentages of ACC controlled vehicles in the network. This means that the traffic conditions in a critical density scenario improve with more controlled cars.
- For densities higher than 60 Veh/km the difference in the average speed and flow are
minimum between different amounts of ACC percentages.

After thorough testing and multiple revisions of the simulation model and implementation, it is not known why the vehicles do not reach their maximum speed under any circumstance, neither when controlled by the developed CFModel or by the pre-existing IDM model. SUMO seems to be somehow limiting their speed, but it’s not being done neither from the edge maximum speed values nor the maximum speed of the vehicle’s type. This circumstance makes the numerical results obtained unreliable.

The fact that critical density remains unchanged for any percentage of ACC controlled vehicles points out that this parameter is defined by geometrical dimensions, since the mean values for the geometrical parameters of the IDM cars where the same employed for the ACC ones, as well as the mean values of maximum speed and standstill distance. The fact that average speed increases in the free flow conditions goes according with the results of section 6.3.2, where the increasing percentage of ACC cars also increased the average stationary speed of the road. Since the average speed increases and according to equation 6.13, flow will also increase if more ACC controlled vehicles are introduced in the traffic network.
7. Budget

For the confection of the budget the engineering costs will be taken into account as well as the material resources. Software licences will not be included since only Student licences and open software has been employed.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Use (hours)</th>
<th>Price per unit (€/kWh)</th>
<th>Price per unit (€/h)</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial engineer</td>
<td>315</td>
<td>22</td>
<td></td>
<td>6930</td>
</tr>
<tr>
<td>Computer for simulation and programing</td>
<td></td>
<td></td>
<td></td>
<td>1500</td>
</tr>
<tr>
<td>Electric consumption of the computer[9]</td>
<td>250</td>
<td></td>
<td></td>
<td>7.8</td>
</tr>
<tr>
<td>Internet acces ADSL [10]</td>
<td></td>
<td></td>
<td></td>
<td>87.20</td>
</tr>
<tr>
<td>Office material and printings</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>

Total cost IVA not included (€) 8574
IVA (21%) 1800.75
Total cost (€) 10374.75

Table 7.1. Project budget

As seen in Table 7.1 the total cost of the project is 10374.75 €. This price is mostly attributed to the engineer hours. It is important to note that through the use of Student licences and open software the budged is greatly decreased for licences of full distribution of softwares usually cost a few thousands of euros.
8. Environmental impact

Since this has been a theoretical project not involving the construction of any prototype or similar, the carbon footprint is greatly reduced. A few aspects that should be taken into account to evaluate the environmental impact though are the electrical consumption of the computer, the fabrication of the computer itself and commuting the engineer has been subjected to due to the project.

After listing the sources of carbon emission in this project, it can be concluded that the total amount of emissions is minimal. It should also be mentioned that the results of this project give a perspective of consumption reduction for vehicles which would in the future bring a reduction of the carbon emissions involving traffic, though this is impossible to compute at the current stage of the project.
Conclusions

According to the results obtained in this project, different conclusions, future work and recommendations can be drawn.

First of all, to achieve the final objective of simulating full vehicular networks using VEINSs a process with different steps will have to be followed. Having seen the amount of work required to fulfil the firsts steps it is obvious that the following ones will require an even greater dedication that should probably be distributed among future projects.

SUMO is a useful tool for simulating traffic, and it has proven to offer numerous options at each stage of the simulation. It also poses difficult challenges as the documentation for an advanced use is scarce and not detailed. For this reason, some problems such as the limitation of the vehicles speed remains unsolved even after long hours of research through the SUMO online community. This problem remains to be dealt with in future work conducted on this section. As a final recommendation on the software, it is better to be used on Linux, both because the installation and compilation are simpler and because the later use of certain applications SUMO includes will also be simpler.

Future work on SUMO could focus on the simulations output treatment. A development of a specific file of data output with only the necessary fields would ease the treatment and study. Also the automatization of this process would greatly reduce the amount of time spent to extract conclusions from the information gathered from the simulations.

Concerning the experiments conducted the introduction of enough ACC controlled vehicles in a traffic flow have proven to eliminate the oscillations present when there are only human drivers. In addition to this, the mean speed of the flow is slightly increased even for small percentages of ACC cars, which would improve the traffic conditions in a congested traffic scenario. The construction of the fundamental diagram has manifested the hidden speed limit that SUMO is applying to the vehicles. Apart from that the results show that while there is no effect on the critical density, critical speed and flow are increased with increasing percentages of ACC in the simulation.
Acknowledgements

After the final wrappings of the project, many questions and doubts trouble my mind: will I make myself clear to others on the project? Was it enough? Will it have an impact on future works? ... But there is one question I clearly know the answer of: how did I manage to pull it out?

I want to thank all my family, specially my parents, for being an example to follow and an unconditional support during each stage of this project and in my entire life. Thanks to all my friends for always being there and putting up with all my complaints when things were not promising.

I also want to thank Arnau Doria and Ernest Benedito, my project directors, for giving me the chance to discover this aspect of the vehicular world, for their availability and dedication to help me in any trouble along the way and for giving purpose to long hours of work.
Bibliography

References


Complementary bibliography


