UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Design and Implementation of Traffic Flow Simulation

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:
Orozco Sánchez Andrés Sebastían

Tutor:
Pineda Arias Israel Gustavo, PhD

Urcuquí, Julio 2021

UNIVERSIDAD YACHAY TECH

Urcuquí, 10 de junio de 2021

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2021-00014-AD

A los 10 días del mes de junio de 2021, a las 16:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Miembro No Tutor	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

El(la) señor(ita) estudiante **OROZCO SANCHEZ, ANDRES SEBASTIAN**, con cédula de identidad No. **0604087080**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **DESIGN AND IMPLEMENTATION OF TRAFFIC FLOW SIMULATION**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN.**

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.	10,0
Miembro Tribunal De Defensa	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.	9,0

Lo que da un promedio de: **9.7 (Nueve punto Siete)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

OROZCO SANCHEZ, ANDRES SEBASTIAN
Estudiante

Firmado electrónicamente por:
ANDRES SEBASTIAN OROZCO SANCHEZ

Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Presidente Tribunal de Defensa

ERICK EDUARDO CUENCA PAUTA
Digitally signed by ERICK EDUARDO CUENCA PAUTA
Date: 2021.06.15 10:16:39 -05'00'

Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.
Tutor

Firmado electrónicamente por:
ISRAEL GUSTAVO PINEDA ARIAS

Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Miembro No Tutor

Firmado electrónicamente por:
FREDY ENRIQUE CUENCA LUCERO

DAYSY MARGARITA MEDINA BRITO

Firmado digitalmente por DAYSY MARGARITA MEDINA BRITO
Fecha: 2021.06.14 15:15:05 -05'00'

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Andrés Sebastián Orozco Sánchez**, con cédula de identidad **0604087080**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Marzo del 2020.

Firmado electrónicamente por:
ANDRES SEBASTIAN OROZCO SANCHEZ

Andrés Sebastián Orozco Sánchez
CI: 0604087080

Autorización de publicación

Dedication

"For all those who supported me and believe in me. Especially my mother, my brother and friends."

Acknowledgments

The year 2020 has undoubtedly marked us as a society and we have had to go through one of the worst moments of our lives. Therefore, the realization of this work would not have been possible without the support of the following people. First of all, I would like to express my sincere gratitude to my supervisor Professor Israel Pineda, Ph. D., who unconditionally decided to support me in the realization of this work, his guidance, patience, and dedication is a clear example of a person who loves what he does and is willing to help others when they need it.

An especial thanks to my friends Mabe, Erick, Rafael, Kevin, Gino, and Daniela with whom I lived unforgettable experiences and whom they became a fundamental pillar in my growth as a person. My second family, they were there for me when I need them and I also hope I could have contributed something to their lives.

Finally, thanks to my father Pedro who was the financial support for my studies. But particularly, thanks to my mother Miriam and my brother Esteban, who gave me their unconditional love, support and above all taught me not to give up despite the hard times we went through. Without their company and support, this work could never have come to an end, I love them with all my being.

Resumen

La simulación del flujo de tráfico se ha vuelto un campo de estudio importante en los últimos años. Es una alternativa para evaluar las condiciones del tráfico de varios escenarios para crear infraestructura de transporte. Existe una amplia gama de modelos que intentan describir el movimiento del vehículo en condiciones de tráfico. Sin embargo, los modelos microscópicos han prevalecido como los más utilizados en el campo de la simulación, especialmente los modelos de seguimiento de coches. Por lo general, los simuladores de tráfico se componen de tres relaciones: seguimiento del coche, cambio de carril y aceptación de espacios.

Los modelos de seguimiento de vehículos pueden reproducir varios fenómenos microscópicos del tráfico. Estos modelos son los representantes más importantes entre los simuladores microscópicos utilizados actualmente. Estos modelos se basan en la idea de que los vehículos ajustan sus parámetros (por ejemplo, velocidad, aceleración) en función del vehículo precedente. Por otro lado, cuando el vehículo no esta impedido por un vehículo al frente, este vehículo acelerara libremente hasta alcanzar su velocidad deseada.

El modelo estudiado en este trabajo es el modelo de seguimiento de coches de Gipps el cual se desarrolló en 1981. Aunque se han desarrollado varios modelos desde entonces, el modelo de Gipps sigue siendo uno de los modelos más utilizados y relevantes en la actualidad. Tanto el modelo original como sus modificaciones, siguen siendo el núcleo de varios simuladores de tráfico académicos y comerciales. Este también puede ser usado en conjunto con el modelo de cambio de carril de Gipps u otros modelos de cambio de carril para construir simuladores de tráfico más complejos. Además, el modelo de seguimiento de automóviles de Gipps puede imitar las características reales del flujo de tráfico, como la heterogeneidad del tráfico, la congestión del tráfico y la prevención de colisiones. Por ello, consideramos importante realizar una implementación y evaluación de este modelo para actualizar resultados de trabajos anteriores.

Nuestra implementación del modelo fue bidimensional y se realizó en el lenguaje de programación Python. Los experimentos presentados en este trabajo muestran que nuestra implementación del modelo se comporta como el modelo original de Gipps. Además, los experimentos muestran que podemos replicar varios fenómenos de tráfico, como la congestión del tráfico y el tráfico heterogéneo. Por lo tanto, nuestra implementación del modelo se puede utilizar en trabajos futuros como un modelo confiable para la simulación de tráfico..

Palabras Clave: Seguimiento de vehículos, Simulación, Modelo Microscópico, Velocidad, Aceleración, Flujo de Tráfico.

Abstract

Traffic flow simulation has become important in recent years. It is an alternative to evaluate traffic conditions of various transportation infrastructure scenarios. There is a wide range of models that try to describe vehicle movement in a traffic stream. However, microscopic models have prevailed as the most used in the field of simulation, specially car-following models. Commonly traffic simulators are made up of three relationships: car-following, lane changing, and gap acceptance.

Car-following models can reproduce various microscopic traffic phenomena. These models are the most important representatives among currently used microscopic simulators. These models are based on the idea that vehicles adjust their parameters (e.g., speed, headway) depending on the preceding vehicle. On the other hand, each vehicle in the simulation will freely accelerate until it reaches the desired speed when it is not blocked by a vehicle in front.

The model studied in this work is the Gipps' car-following model which was developed back in 1981. Although several models have been developed since then, the Gipps' model remains one of the most used and relevant models today. The original model and its modifications continue to be the nucleus of several academic and commercial traffic simulators. It can also cooperate with the Gipps' lane change model or other lane change models to build more complex traffic simulators. Also, Gipps' car-following model can imitate real traffic flow characteristics such as heterogeneity of traffic, traffic congestion, and collision avoidance. For this reason, we consider it important to carry out an implementation and evaluation of this model to update results from previous works.

Our implementation of the model was two-dimensional and done in the Python programming language. The experiments presented in this work show that our implementation of the model behaves like the original Gipps' model. Also, the experiments show that we can replicate various traffic phenomena, such as traffic congestion and heterogeneous traffic. Therefore, our implementation of the model can be used in future works as a reliable model for traffic simulation.

Keywords: Car-Following, Simulation, Microscopic Model, Speed, Acceleration, Traffic Flow.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Computer simulation has become indispensable in a growing number of disciplines. The list of sciences that make extensive use of computer simulation has grown to include astrophysics, materials science, engineering, fluid mechanics, decision theory, transportation, and many others [1]. Simulation is generally defined as an imitation of a system or process, while computer simulation is the replication of a system or a process on a computer. The use of computer simulation models has become particularly prevalent among transportation systems [2].

There is a wide range of models to describe traffic movement. These models are classified depending on the level at which traffic flow is being represented [3] including macroscopic, mesoscopic, and microscopic models. Microscopic models imitate the movement of every vehicle by taking into account attributes such as its speed and acceleration as a function surrounding vehicles and highway environment [2]. Macroscopic models instead track some features of the traffic flow itself, such as speed and density, which results in less detailed simulation to study than their microscopic counterpart [4, 5]. Finally, mesoscopic models are a hybrid of microscopic and macroscopic models; each vehicle is modeled with its characteristics , but also relationships such as speed-density and queuing are taking into account [6]. This type of model aims to combine the benefits of both while overcoming their respective limitations [7].

The model that we will deal with during this thesis is the Gipps' car-following model [8]. The model belongs to the class of car-following. These models describe the behavior of individual drivers when interacting with vehicles in their nearness [3]. The basic concept is that the model controls the behavior of the driver concerning the preceding vehicle in the same lane. Also, Gipps' model is considered as a safe-distance model, which are based on the assumption that the follower vehicle always keeps a minimum distance to the preceding vehicle [9], in other words, is a model that avoids collisions.

1.2 Problem statement

Transportation is generally concerned with the efficient, safe, and sustainable movement of people and goods. The life cycle of a transportation system usually consists of five stages: planning, design, construction, operation, and maintenance [2]. Transportation projects generally require a significant investment [10], so each stage of its life cycle has to be planned and justified robustly. For this reason, the traffic flow theory field has gained considerable attraction, since with it, we can evaluate the operational quality of traffic given a set of conditions, e.g., traffic demand, fixed infrastructure, and speed limits. With the increasing collection of traffic data and the increase in computational power, the simulation of traffic flow models comes into play.

Traffic flow theory and modeling started in the 1930s, pioneered by the US-American Bruce D. Greenshields [11]. Since then, many models trying to replicate the behavior of traffic has been developed. However, car-following models are the ones who still forms one of the main processes in all microscopic simulation models as well as in modern traffic flow theory [12], and traffic simulators.

Herrey and Herrey in 1945 developed the first car-following theory, who suggested that drivers keep a minimum safe distance from the preceding vehicles [3]. From here on, various authors discussed this type of model. This principle of safe distance is the basic idea of most of the car-following models until this day. In this work, a brief review of the most relevant models, both with their advantages and disadvantages, will be presented.

Most of the models until Gipps presented a lack of accuracy when modeling microscopic traffic [12]. This lack of accuracy in trying to replicate real traffic behaviors is what motivated Gipps to build a new car-following model; these are the properties that Gipps [8] propose to build a new model that is capable of replicating the traffic flow:

- The model can reproduce real traffic behavior.
- The parameters in the model should reflect characteristics of the vehicles and drivers
- The interval for the calculations must be the same as the reaction time of the driver.

The Gipps' car-following model embodies these properties but also as other authors stated the model is suitable for modifications or addition of more properties. Also, these properties allow the model to replicate various real characteristics of the traffic flow. This is one of the main reasons that keep the Gipps' car-following model one of the most commonly used and cited models of the class of "safety distance" models.

For these reasons, we decided to carry out an implementation and evaluation of this model using a modern programming language such as Python to update results from previous works.

1.3 Objectives

1.3.1 General Objective

The objective of this thesis is to review, implement and test the original Gipps' car-following model in two dimensions using a modern programming language like Python. Our implementation of the model should be able to reproduce various traffic phenomena characteristics. Therefore, our implementation of the model can be used in future works as a reliable model for traffic simulation.

1.3.2 Specific Objectives

- Understand the differential equations behind the model to be able to use a numerical integration method that helps us to calculate the formula for the position of the vehicles.
- Create a street network based on graphs.
- Implement the movement of the vehicles in two dimensions.
- Implement the model in the Python programming language.
- Create various experiments to evaluate the model capabilities for traffic simulation.

Chapter 2

Theoretical Framework

This chapter presents the necessary theory for understanding this work. First, we will review the concept of traffic flow theory, followed by the models' classification by their aggregation level. Secondly, a general review of car-following models will be presented followed by the principal characteristics of these models. The most relevant car-following models are presented until reach the Gipps' car-following model, which is the main point of this work. Also, lane-changing and cellular automata models are explained in a general way. Finally, the main characteristics of traffic flow are explained to finish with some applications of traffic simulation.

2.1 Traffic Flow Theory

The objective of traffic flow theory is to evaluate the operational quality of a traffic stream, giving a set of prevailing conditions [2]. Traffic flow theory and its modeling started in 1930 by Bruce D. Greenshields [13]. Nowadays, due to the increasing computing power, traffic demand, and easier access to data, the field has gained considerable attraction [11]. Traffic flow theory relates to the operations stage of transportation; in this stage, the main focus is the control algorithms such as traffic regulations, ramp metering, incident removal, among other aspects [2]. However, traffic flow theory tools such as simulators can be used in any other stage of transportation.

2.2 Traffic Flow Models Classification

Traffic flow models can be categorized in various ways depending on the level at which traffic flow is being represented. Here the classification of the models will be depending on the level of detail, this is called categorized by the aggregation level.

2.2.1 Macroscopic Models

At the macroscopic level, we do not look at the vehicles as separate entities [14]. Instead, macroscopic models replicate the movement of groups of vehicles (e.g., a platoon of vehicles). The aim of this type of model is to study the relationships between density, speed, and flow of the traffic stream [2]. Figure 2.1 shows a graphical representation of the macroscopic variable density.

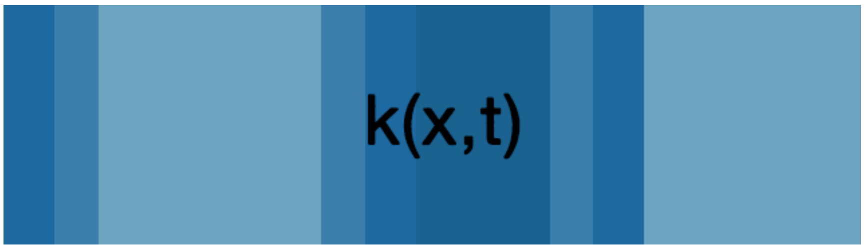


Figure 2.1: Representation of the macroscopic variable density.

Most of the macroscopic mathematical model structures for the description of traffic flow phenomena are derived from microscopic considerations within a platoon of similar vehicles [15]. Whereas, macroscopic models are more useful in different cases, Treiber [11] described some examples:

- If effects that are difficult to describe macroscopically need not to be considered.
- If one is interested in macroscopic quantities, only.
- If the computation time of the simulation is critical (e.g., requires too much computing power).
- If the input data come from heterogeneous sources and/or are inconsistent.

2.2.2 Microscopic Models

Microscopic models imitate the movement of every vehicle individually and their interactions between themselves, drivers, and the infrastructure [14] which collectively form the traffic flow. Generally, microscopic traffic flow models are examples of driven multi-particle models. The aim of these models is to try to describe the actions and reactions of the particles that make up the traffic as accurately as possible [16]. The general form of microscopic models is to compute driver and vehicle parameters at time $t + \Delta t$, where Δt is the driver reaction time. The parameters to be calculated are commonly the vehicle position $X_n(t)$, speed $V_n(t)$, and acceleration $a_n(t)$. Figure 2.2 shows how vehicles are viewed as separate entities in microscopic models.

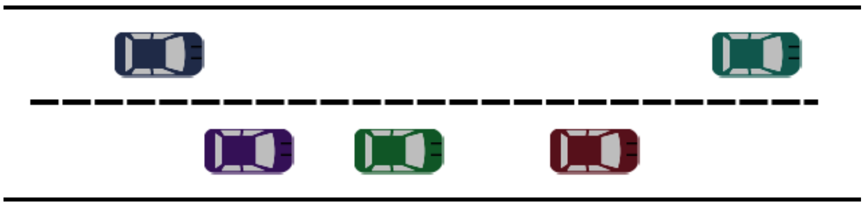


Figure 2.2: Microscopic model representation.

There are several applications for microscopic models. Here we list some applications described by Treiber [11]:

- Modeling how individual vehicles affect traffic.
- Situations where the heterogeneity of the traffic is important for the simulation.
- Incorporate human driver behavior like reaction times, inattentiveness, and anticipation.
- Visualization of interactions between various traffic participants.
- Generates surrounding traffic for scientific driving simulators, game simulators among others.

Also, depending on the application of the model, microscopic models can be divided into [14]:

- Car-following model: As mentioned in Chapter 1. This model is based on the basic idea that the vehicle adjusts its parameters depending on the preceding vehicle, called the leader.
- Lane-change model: This model assesses the idea of how a vehicle decides to change lanes depending on the preference of the driver and the situation of other vehicles and the environment around it.
- Route choice model: This model is based on the idea that drivers determine which way to go on a road network. This decision is made depending on the traffic and the surrounding infrastructure.

Since Gipps' model belongs to the class of car-following a more in-depth explanation can be found later.

2.2.3 Mesoscopic Models

The third class of traffic simulation models are mesoscopic models. These models are a hybrid of microscopic and macroscopic models; this means that individual vehicles are modeled, but at an aggregate level, usually by speed-density relationships [6]. This turns out in a high level of detail to describe the traffic entities, but their behavior and interactions are described at a lower level of detail [16]. For example, macroscopic quantities such as density, flow, and speed can be obtained from microscopic quantities by local aggregation. This is possible if we can define spatiotemporal regions that are microscopically large, such that they contain a significant number of vehicles for averaging. In the same way, these regions must be macroscopically small, smaller than the typical lengths and time scales of the traffic patterns of interest [11]. Figure 2.3 shows how through the aggregation and disaggregation of variables the mesoscopic models are created.

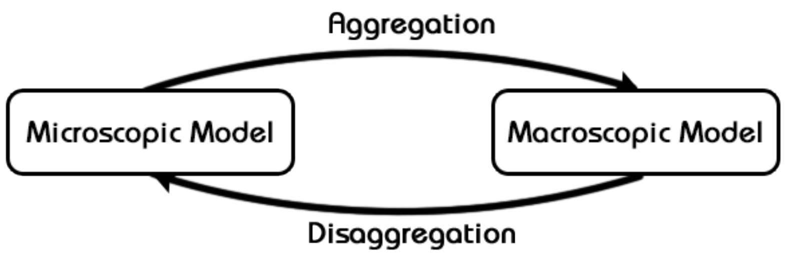


Figure 2.3: Mesoscopic model representation.

2.3 Car-following models

As described before, car-following models are a type of microscopic model. This Section will give a deeper review of the basic concept of car-following models and which are the most relevant models in the history of car-following models.

2.3.1 General Review

Car-following models are the most important representatives of microscopic traffic simulators [11]. These models attempt to understand the relationship between the individual vehicle and global behavior on a more macroscopic scale [12]. The basic idea that car-following models try to describe, is how the following vehicle reacts to the lead vehicle in the same lane. Most of the models that belong to this class, use the concept that the acceleration of the following vehicle depends on the speed of the preceding vehicle, speed difference, and the headway between the vehicle pair [17]. For example, if the lead vehicle travels free of traffic ahead, its speed corresponds to the desired speed of the driver. On the other hand, if the desired speed of the following vehicle is lower than the speed of the lead vehicle, their spacing will keep increasing. Finally, if the desired speed of the following vehicle is higher than the leader speed, at some point, the vehicle will enter the car-following state with the lead vehicle. Thus it is spacing, speed, and acceleration will be dictated by those of the lead vehicle.

The first car-following model was proposed back in the fifties by Reuschel [18] and Pipes [19]. These two models already introduced an essential element of modern microscopic modeling: The safe distance between the leader vehicle and the follower vehicle [11]. Later the General Motors nonlinear model [20] introduced the idea that the following behavior is based on a stimulus-response and a driver reaction time. However, these models refer to low-speed situations; therefore, are not suitable for high-speed networks [7]. Also, these models are considered as not complete models, since they cannot describe either free traffic or approaches to standing vehicles [11]. Afterward, Newell [21] inspired by the GHR model, studied the inclusion of nonlinear effects in the dynamics of car-following models. Newell later in 2002 [22] based on his 1961 study[21], created one of the simplest car-following models. The model proposed by Newell is nonlinear and time-discrete. Its simplicity was given by the fact that when a vehicle was under the car-following regime it only considers two parameters: The reaction time Δt and the effective length l_s. Subsequently, the PITT car-following model was one of the first models to be implemented in a micro simulator environment, specifically the model was implemented in the INTRAS simulator developed by the Federal Highway Administration [2]. The model is similar to the concept of GM models, calculating the distances between vehicles but adding different constants of calibration [23]. Eventually, Gipps [8], considering the disadvantages of these models developed in 1981 his car-following model. The model aims to solve the problems of the models mentioned before. Gipps' model is considered as a complete model or multi-regime model since it considers the desired speed of the vehicle, as well as whether the following vehicle is in car-following mode or braking mode [2].

Now that we know generally the most essential models up to the Gipps' model. We will briefly review a mathematical description of the most common variables that car-following models share. Then, a more in-depth explanation of each model mentioned will be covered.

2.3.2 Microscopic Variables

Microscopic variables are described in Figure 2.4. The follower vehicle is represented by n and the leader vehicles is represented by $n-1$. The vehicle location in a given moment is represented by the variable x_n which is the position of the vehicle n. While the position of the leader vehicle is represented by x_{n-1}. Another important parameter in car-following models is the effective size s_n. This parameter is also known as distance gap or space headway h_s by certain authors. It consists of the physical length of the vehicle L_n, plus a distance $d_n(t)$ which the following vehicle is not willing to intrude [2] (see Equation 2.1).

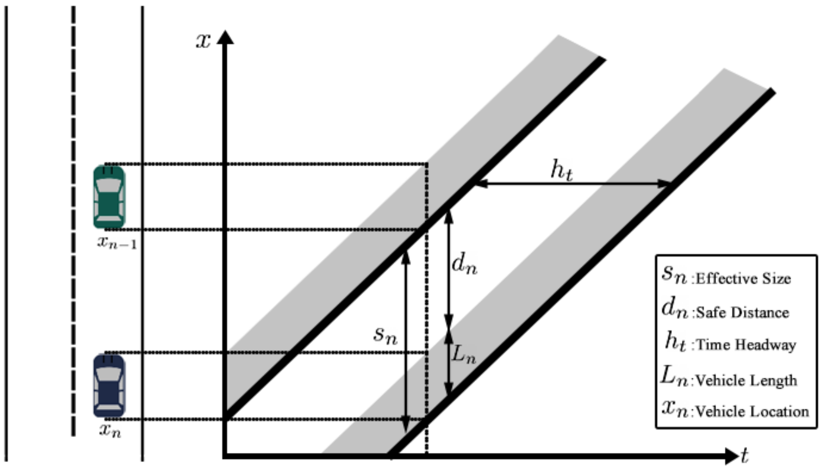


Figure 2.4: Defining the state variables of car-following models.

$$s_n(t) = L_n + d_n(t) \tag{2.1}$$

Analogously to space, vehicles also use a certain segment of time which is called time headway h_t. This parameter represents the time that would take the follower to reach the position of the lead vehicle. It consists of the interval time or gap g and the occupancy o [14]. The mathematical expression that describes this expression is the following:

$$h_t = g_n + o_n \tag{2.2}$$

Depending on the model, additional variables could be added. For example, the vehicle acceleration $a_n = du/dt$ [11] is a parameter that some models take into consideration. Also, accelerating vehicles have positive values for a_n, and braking cars have negative values for a_n [14].

2.3.3 Pipes Car-following Model

As mentioned in the general review, one of the first car-following models was introduced by Pipes [19]. This model constitutes the steady-state car-following model in some micro-simulation software such as CORSIM and VISSIM. The model, is considered as a multi regime in the sense that a different model is utilized for the congested versus uncongested regimes [24].

Pipes based his model on the California Vehicle Code which stated that ”A good rule for following another vehicle at a safe distance is to allow yourself the length of a car for every 10 miles per hour you are traveling” [2]. Consequently, his model assumed that the distance between the leader vehicle and the follower is equal to one vehicle length for every 10 miles per hour of speed. For this rule to be satisfied, the following equation must be fulfilled [19]:

$$x_n = x_{n+1} + b + L_n + smin_{n+1} \tag{2.3}$$

where
x_n and x_{n+1} are the location of vehicles n and $n + 1$, measured from the front of each vehicle,
b is the distance between vehicles when they are at a standstill,
L_n is the length of vehicle n,
$smin_{n+1}$ is the minimum distance between vehicles based on the California Vehicle Code.

The quantity $smin_{n+1}$ can also be rewritten as follows:

$$smin_{n+1} = Tv_{n+1} \tag{2.4}$$

where
v_{n+1} is the speed of vehicle n + 1,
T is the time it takes to travel this minimum distance.

Also, Pipes developed differential equations for providing the velocity and acceleration of a platoon of vehicles. These equations were written as a function of the movement of the lead vehicle [19]:

$$v_n = v_{n+1} + Ta_{n+1} \tag{2.5}$$

and

$$a_{n+1} = \frac{v_n - v_{n+1}}{T} \tag{2.6}$$

One of the problems with Pipes' model, is that it does not consider a driver reaction time, so the model assumes that the spacing is achieved instantaneously [25]. While, in reality, drivers have a time lag reacting to changes in the trajectory of the leader vehicle. Meaning that exits a significant variability in their spacing [2].

2.3.4 Gazis-Herman-Rothery (GHR) Model

The GHR model [20] was developed by Gazis, Herman and Rothery. Also commonly known as the GM model since it was developed and tested at the General Motors laboratory in Detroit [2]. Its formulation is the following [20]:

$$a_n(t) = cv_n^m(t)\frac{\Delta v(t-\Delta t)}{\Delta x^l(t-\Delta t)} \tag{2.7}$$

where
a_n is the acceleration of vehicle n implemented at time t,
v_n is the speed of the nth vehicle,
Δt is the next time interval or also know as driver reaction time,
Δx is the relative spacing between the nth vehicle and $n-1$ vehicle, at a time $t-\Delta t$,
Δv is the relative speed between the nth vehicle and $n-1$ vehicle, at a time $t-\Delta t$,
m, l, c are constants to be determined.

The model was based on an intuitive hypothesis that the driver desired acceleration was proportional to Δv or deviation from a set following distance $k-\Delta x$ which could itself be speed dependent [12]. This relationship between the follower and the leader is called the stimulus-response type [9].

One of the problems in the GHR model is the calibration of the m, l, c parameters. Some studies such as [12, 26] produced different calibrations of these parameters. Therefore, to generate different traffic conditions such as congested and non-congested conditions, different calibrations of the parameters should be used. Another problem results when the speed of the following vehicle increases since the sensitivity of the following vehicle also increases [2]. This problem leads to the model instability in high-speed networks [27]. Finally, the speed difference Δv, determines whether the following vehicle will accelerate or decelerate, regardless of the spacing of the vehicle. This behavior means that even when the vehicles are far away, the following vehicle will always decelerate in the response to the deceleration of the lead vehicle [2].

In this form, the Gazis, Herman, Rothery model is similar to the stimulus' response of the Pipes model, but with the main difference that the GHR model introduced the reaction time of the driver.

2.3.5 Newell's Car-Following Model

Newell's car-following model is arguably the simplest representative of time-discrete models [11]. The model was based on a simple rule. The rule stated that if an nth vehicle is following an $(n-1)$th vehicle on a homogeneous highway, the time-space trajectory of the nth vehicle is essentially the same as the $(n-1)$th vehicle except for a translation in space and in time [22]. Here is the equation that represents Newell's basic rule:

$$x_n(t+\tau_n) = x_{n-1}(t) - d_n \tag{2.8}$$

where
$x_n(t+\tau_n)$ is the nth vehicle location or follower vehicle at a time $(t+\tau_n)$,
$x_{n-1}(t+\tau_n)$ is the $n-1$th vehicle location or leader vehicle at a time $(t+\tau_n)$,
d_n is the distance that the driver considers to be safe from the $n-1$th vehicle (effective size),
τ_n the ability of the driver to respond to any action of the vehicle $(n-1)$ (reaction time).

This equation is trying to make that the vehicle nth trajectory will follow the trajectory of the $(n-1)$th for some appropriate values of τ_n and d_n [22]. This means that the vehicles are restricted to the car-following regime. Newell's model has two parameters: The reaction time τ_n, and the minimum gap d_n. The standard value used for the reaction time was $\tau_n = 1$, while the minimum gap was about $5m$ [11]. This allows the model to be calibrated more easily, producing competitive percentile error values for spacing prediction but producing high values of error in the prediction of speed and acceleration [28].

2.3.6 Gipps' Car-Following Model

Gipps' car-following model [8] remains as one of the most representative models nowadays and is the commonly used model pertaining to the class of collision avoidance [29]. The model has been used in a lot of packages for traffic simulation, software such as AIMSUN [10], CARSIM [30], SITRAS [31], DRACULA [32], among others, used Gipps' model or modifications of it with favorable results. The model can be categorized as a multi-regime model since it considers the desired speed of the following vehicle, as well as whether the following vehicle is in braking or car-following mode [2]. This feature was introduced in the model by calculating two speeds at a given time $(t+\tau)$. The minimum of these two values is the chosen updated speed for the vehicle. The equation is the following:

$$v(t+\tau) = min\{G_a(t), G_b(t)\}, \tag{2.9}$$

where:

$$G_a(t) = v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)\sqrt{0.025 + v_n(t)/V_n}, \tag{2.10}$$

$$G_b(t) = b_n\tau + \sqrt{b_n^2\tau^2 - b_n(2(x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}} \tag{2.11}$$

where:
$G_a(t)$ is the speed of the vehicle when it travels freely,
$G_b(t)$ is the speed of the vehicle if its movement is constrained by a vehicle in front,
$v_n(t)$ is the speed of vehicle n at time t,
$a_n(t)$ is the desired acceleration of the driver,
τ is the apparent reaction time of the driver,
V_n is the desired speed of the driver,
b_n is the most severe braking $(b_n < 0)$ of the driver,
$x_n(t)$ is the location of the front of the vehicle n,
s_{n-1} is the effective size of vehicle $n-1$ which is the vehicle length plus a safe space which the following vehicle is not willing to intrude,
$\hat{b}$ is the value of b_{n-1} estimated by the driver of vehicle n.

Equation 2.10 is also known as the accelerating part or free-flow mode. The speed resulting from this equation is the speed that the vehicle would obtain if its movement is not impeded by the movement of the vehicle in front [3]. Thus, the equation does not take into account the characteristics of the preceding vehicle. Equation 2.10 ensures that the speed of vehicle (n) will not exceed the desired speed of the driver. Also, ensures that the acceleration of vehicle (n) will reach zero when the desired speed is reached [2]. On the other hand, Equation 2.11 is known as the decelerating part or car-following mode. The speed resulting from this equation is the speed that a vehicle would obtain if its movement is constrained by a vehicle in front. This equation allows the following vehicle to be able to stop safely even for the most severe braking of the lead vehicle [2]. This behavior is achieved by taking into account the characteristics of the lead vehicle, such as its speed $v_{n-1}(t)$, its position x_{n-1}, effective size s_{n-1}, and deceleration properties $\hat{b}$. Figure 2.5 illustrates some of the parameters mentioned before.

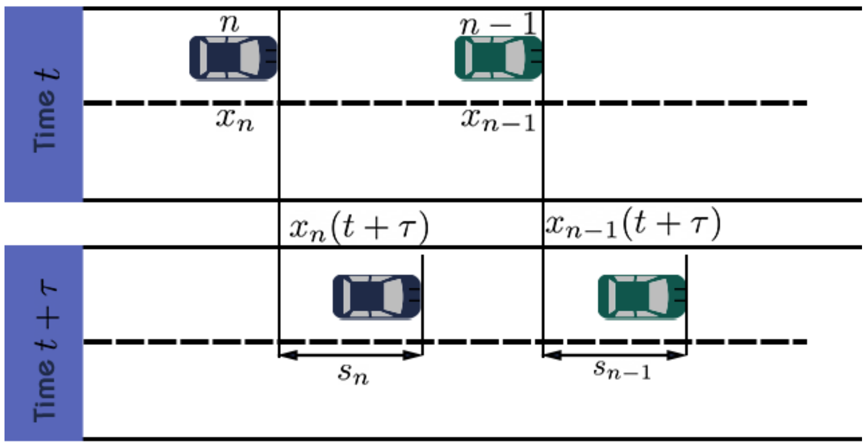


Figure 2.5: Location, effective size, and location parameters from Gipps' model.

Gipps' model, unlike the models described before, is robust in the sense that meaningful results can be produced from a comparatively wide range of parameters [11]. This is because Gipps [8], derived the model from some basic assumptions. These assumptions made by Gipps make the model be able to relate each of its parameters to a vehicle or driver characteristics that the driver of the following vehicle could estimate or perceive. Therefore, calibrating the model, compared to the other models, is relatively easy as one can measure or estimate its parameters based on vehicle and driver characteristics [2].

However, some problems might arise due to parameter b which is the most severe braking of the vehicle. Depending on the author interpretation of this parameter, the model can generate different results. For example, if the model is accident-free, every braking maneuver is performed very uncomfortably with full brakes. On the contrary, when interpreting b as the comfortable deceleration and allowing for heterogeneous traffic, the model possibly produces crashes if leading vehicles brake harder than b [11, 33].

We just reviewed some of the most relevant car-following models. Despite the high number of new car-following models, their incapability to reproduce both traffic propagation and driver-vehicle interactions without relying on the over-fitting produced by their parameters [29] is what makes the Gipps' car-following model a cornerstone in the research of traffic flow simulation. Studies like [33, 3] have shown that the model also can be improved by calibrating its parameters. Also, other studies like [34], investigate the incorporation of other human factors, such as fatigue, distraction and alcohol. More of these applications and recent studies on the model can be found in Chapter 3.

2.4 Lane-Changing Models

Car-Following models are a fundamental part when simulating traffic, but simulating any nontrivial traffic situation requires describing not only acceleration and braking but also lane changes [11]. Therefore, a realistic description of heterogeneous traffic streams is only possible within a multi-lane modeling framework. This framework allows faster vehicles to improve their driving condition by passing slower vehicles [35]. Lane-changing modeling is more complex than car-following because the decision to change lanes depends on several objectives, and at times these objectives may conflict. Thus, in coming to a decision concerning lane changing, a driver must be able to reconcile his short-term and long-term aims [36]. Drivers have various reasons for changing lanes, and their lane change maneuver is likely to be affected by its urgency. Also, drivers may change lanes with or without the cooperation from the vehicles in the target lane [2].

Figure 2.6 represents the general steps involved in the lane-change process [37]. Most lane-changing models in the literature classify lane changes as either mandatory or discretionary [38]. Mandatory changes are performed strategically. Meanwhile, the motivation of the driver for discretionary lane changes is usually the desire to improve driving conditions compared to the current situation [35].

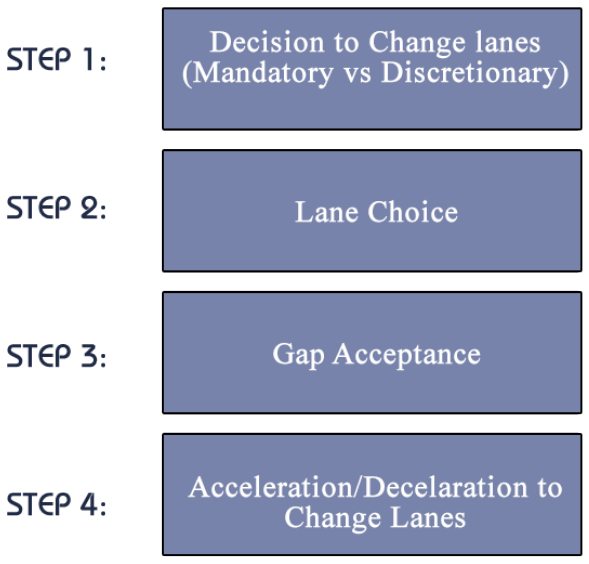

STEP 1:
Decision to Change lanes (Mandatory vs Discretionary)
STEP 2:
Lane Choice
STEP 3:
Gap Acceptance
STEP 4:
Acceleration/Decelaration to Change Lanes

Figure 2.6: General steps involving lane-changing.

2.4.1 Gipps' Lane-Changing Model

Gipps' lane-changing model [36] is one of the earliest rule-based models, and it is based on his car-following model. Gipps' model describes the lane-changing decisions and the execution of lane changes due to three factors: lane-changing possibility, the necessity for changing lanes, and lane-changing desirability [39]. The model incorporates several factors such as the safety gap, presence of heavy vehicles, permanent obstructions, the intent of turning movement, and speed advantage.

The behavior of the driver can fall in one of three zones depending on the distance to his intended turn. If the turn is remote, it does not affect his lane-changing decisions; thus, the driver tries to maintain his desired speed. If the turn enters the zone the driver regards as middle-distance he starts to ignore opportunities to improve his speed that involve changing lanes in the wrong direction. Once the driver has reached them he tends to remain in the pair of lanes most appropriate for his turn. So, by the time the driver comes close to his intended turn, he should be in the correct lane or the adjacent one. Finally, when the intended turn is close enough, the driver is interested solely in reaching the correct lane and speed is irrelevant [36]. The decision to change lanes is a rule-based event in which the output is a binary choice (change or not change) [37]. The boundaries that we just mentioned do not depend on the behavior of the driver patterns over time. Therefore, these behaviors are deterministic in nature [39].

As mentioned before, this model was designed to be used in conjunction with Gipps' car-following model [8] which applies restrictions on the braking rate by drivers to have a safe speed with respect to the preceding vehicle. Consequently, the equation to calculate the safe speed is the following [36]:

$$v_n(t+\tau) = b_n\tau + \sqrt{b_n^2\tau^2 - b_n(2(x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}} \tag{2.12}$$

where, $v_n(t+\tau)$ is the maximum safe speed for vehicle n with respect to the preceding vehicle at time $(t+\tau)$,
b_n is the most severe braking desired by the driver $(b_n < 0)$,
τ is the reaction time of the driver
$x_n(t)$ is the location of vehicle n,
s_{n-1} is the effective size of vehicle $n-1$ which is the vehicle length plus a safe space which the following vehicle is not willing to intrude,
$\hat{b}$ is the value of b_{n-1} estimated by the driver of vehicle n.

For the lane-change model. The Equation 2.12 is limited by the desired speed of the driver and a maximum braking limit of $-2ms^{-2}$. This limits the influence of stationary objects to between 100 and 200 meters depending on the desired speed of the driver [36]. This behavior allows the vehicle to fit in a smaller gap in the case where the driver is prepared to brake harder in response to the urgency of reaching a particular lane.

Gipps' lane-change model summarizes the process as a decision tree with a series of fixed conditions typically found in urban traffic [39]. This framework allows the model to be flexible, allowing the addition or replacement of new lane-changing decisions and even, as mentioned by Gipps [36], the model can be used with other car-following models. Therefore, the model has been applied in several microscopic traffic simulators [40]. However, Gipps' lane changing decision model has never been validated since it does not include any framework for its validation [39]. Also, the model does not consider variability in individual driver behavior causing some limitations in heavy traffic conditions where appropriate gaps are rarely available. These traffic conditions are created when the reaction time of the driver provides courtesy [40].

2.5 Cellular Automata Model

Cellular automata describe all aspects of dynamical systems by using integers. Space is subdivided into cells and time into time steps. At any time, each cell is in one of a small number of states [11]. The basis of traffic-related cellular automatons models was laid by Wolfram [41] who researched the most straightforward nontrivial class of CAs called ”elementary cellular automata”. The most used and well know model belonging to this class is the cellular automaton model for freeway traffic developed by Nagel and Schreckenberg [42] (NSM). This model generalizes the rule 184 [41] to a stochastic model with more than two speed levels [11]. The model is based on four steps, which are performed in parallel for all vehicles:

1. Acceleration: if the velocity u of a vehicle is lower than u_{max} and if the distance to the preceding vehicle is larger than $u+1$, the speed is advanced by one $[v \rightarrow j-1]$.
2. Deceleration due to other cars: if a vehicle at site i sees the next vehicle at site $i+u$, it reduces its speed to $j-1[u \rightarrow j-1]$.
3. Randomization: with probability p, the velocity of each vehicle is decreased by one $[u \rightarrow u-1]$.
4. Car motion: each vehicle is advanced u sites.

These steps are modeled based on integer-valued probabilistic cellular automaton rules [43]. Already this simple model shows nontrivial and realistic traffic flow behavior [42]. We can see that the particle-based formulation of cellular automata is no conceptually different from discrete-time car-following models formulated as iterated maps. Nevertheless, because of the stochastic nature of NSM and other cellular automata models, there exists no steady-state equilibrium, and consequently, no fundamental diagram in the strict sense [11].

The main difference between cellular automata models and car-following ones are the discrete scaling of time, space, and state variables. Generally, cellular automata models leads to a few advantages because of their simplicity. However, these models has some disadvantages since CAs are phenomenological rather than based on intuitive concepts. Most model parameters are not realistic, and due to their discrete nature, CAs are not suitable for modeling the movement of individual vehicles which is necessary for the research of driving styles, influence of driver-assistance or infrastructure-based control measures on traffic flow [11]. Despite this, some research based on the mixture of both models, cellular and car-following models, has been carried out. The model proposed by Bham and Benekohal [44] is an example of this mixture. This model is more detailed than CA models and has realistic acceleration and deceleration values for the vehicles.

2.6 Traffic Flow Characteristics

Before we can talk about the applications of traffic flow theory, it is necessary to understand what are the three primary characteristics of traffic. These characteristics are flow, speed, and density which are used to describe various aspects of traffic operations.

Flow

Flow is defined as the rate at which vehicles travel through a particular point or highway segment; it is the inverse of the time headway ($= 3,600/h_{avg}$) [2]. The flow rate represents the number of vehicles that pass a specific cross-section per time unit [14], typically vehicles per hour (vph). We call the maximum possible flow rate of any road it is capacity. Measuring the flow q in region $R1$ for single-traffic is done using the following equation [45]:

$$q = \frac{n}{\Delta T} \tag{2.13}$$

Where n represents the number of vehicles that have passed a location x (detector's site) during a time interval ΔT. Volume is typically expressed in units of traffic, and it represents the count of the units off traffic within a specified time interval [2]. For example, if the volume measured at a certain point is 1800 vehicles within 15 min interval, then the flow is 7200 vph. These measures can be expressed on a per lane basis, i.e., in units of vehicles per hour per lane $vphpl$. Also, in practice, the value of the time interval is often a 15-min interval.

The problem with the definition of 2.13 is that it is limited to a time interval ΔT. We can get a more general definition by multiplying the numerator and denominator with an infinitely small location interval dx around x [14]. This leads to a general definition for flow rate:

$$q(x, t, S) = \frac{\text{Total distance covered by vehicle s in S}}{\text{Area(S)}} \tag{2.14}$$

Time Headway

Time headway is often referred to as the microscopic expression of flow. It is given in units of time over a unit of traffic, typically seconds per vehicle (s/veh). Time headway is of interest not only in the modeling of the movement of vehicles, but also as expressions of the operational quality of the traffic stream since they allow us to consider microscopic traffic characteristics for the entire traffic stream [2]. Space and time headways can be visualized in a time-space diagram in Figure 2.7.

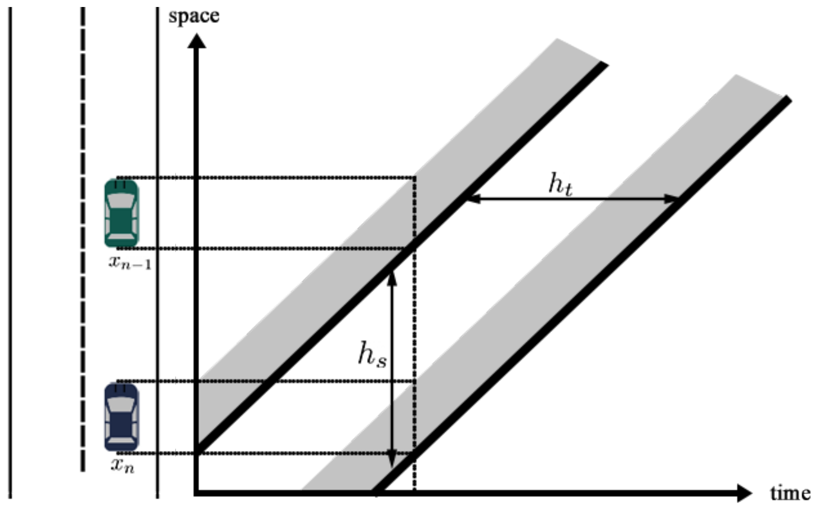


Figure 2.7: Time-space diagram showing two vehicle trajectories, as well as the space headway h_s and time headway h_t.

Here, we can see two vehicles n and $(n-1)$, their positions with respect to time are represented by x_n and x_{n-1}, tracing out two vehicle trajectories. Since both vehicles are traveling at a constant speed these trajectories are parallel linear. The headway h_t is the time it takes for the vehicle n to get to reach with its front bumper the position of the front bumper of the vehicle $(n-1)$. This can be seen as the time gap plus the occupancy time interval of the leading vehicle [11].

Speed

Speed is measured in units of distance per unit of time, typically miles per hour (mph) [2]. At the microscopic level, the speed v is used for describing the movement of an individual vehicle. Meanwhile, at the macroscopic level, the mean speed u is used to describe the average speed of the vehicles on the road [46].

Measuring average speed by definition requires an observation that stretches over time and space [46]. There are two ways to calculate the mean speed. The first way is by taking the arithmetic mean of the observation:

$$u_{avg-time} = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{2.15}$$

where
v_i are the instantaneous speeds,
n are the total number of instantaneous speeds in the sample.

Alternatively, the observer could measure the travel time of each vehicle between two particular locations, and obtain the speed of each vehicle as the inverse of their travel time [2]. This is called the harmonic average of the vehicles' instantaneous speeds:

$$u_{avg-space} = \frac{d}{\sum_{i=1}^{n} \frac{t_i}{n}} \tag{2.16}$$

where
d is the distance over which travel times were measured,
t_i are the travel times observed,
n are the total number of travel times measured.

The harmonic mean speed is sometimes called (not entirely correctly) space mean speed [11]. Moreover, one can show that the harmonic mean $u_{avg-space}$ is always lower than the arithmetic mean $u_{avg-time}$. Consequently, slower vehicles tend to stay on the subject segment longer and thus are weighted more heavily. If all vehicles travel at the same speed, the harmonic mean and the arithmetic mean are equal [2].

Additionally, we can calculate the relationship by taking the harmonic average of the instantaneous speed of the vehicle. This relationship between both, space- and time-mean speeds [45], is calculated as follows:

$$u_{avg-time} = u_{avg-space} + \frac{\sigma^2_{space}}{u_{avg-space}} \tag{2.17}$$

Where σ^2_{space} is the variance of the space-mean speed.

Similarly, the approximate relationship between space-mean speed and time-mean speed is:

$$u_{avg-space} = u_{avg-time} + \frac{\sigma^2_{time}}{u_{avg-time}} \tag{2.18}$$

Where σ^2_{time} is the variance of the time-mean speed.

Density

The macroscopic characteristic called density allow us to get an idea of how crowded a certain section of a road is [45]. Density is expressed in units of traffic per unit of distance, typically vehicles per mile (*vpm*) (or kilometer), or in vehicles per mile per lane (*vpmpl*) [2]. Density is difficult to measure, since it can only be measured in a certain spatial region [45]. However, it is a very useful measure of performance. We can estimate density from its definition, which can be expressed in terms of microscopic quantities [11]:

$$k = \frac{n}{\Delta X} \tag{2.19}$$

n is the number of vehicles present on the road section,
ΔX is the length of a road section.

Equation 2.19, is confined to a certain point in time, we can generalize the equation by multiplying the numerator and denominator by an infinitely small time interval dt, so we can assume that no vehicles arrive or depart during that time, getting the following equation:

$$k = \frac{n \cdot dt}{\Delta X \cdot dt} \tag{2.20}$$

In this case, the numerator reflects the total travel time of all vehicles traversing the region, and the denominator now becomes equal to the area of measurement interval [14].

$$k = \frac{\text{Total time spent in S}}{\text{Area(S)}} \tag{2.21}$$

This approach was initially proposed by Edie [47]. The advantage of this approach is that it allows us to obtain density using very various measurement techniques. This approach allows us to compare different traffic streams observed for different times and across different distances [2].

2.6.1 Traffic Flow Regimes

Flow, speed, and density allow us to interpret various traffic conditions, such as traffic flow regimes. We can distinguish different types of operational characteristics, called *regimes*. Depending on the author it can also be called as *phases* or *states* [45].

Completely Free-Flowing Traffic

When vehicles are not impeded by other vehicles, drivers try to reach their desired speed. Thus, they travel at a maximum speed of u_f, this is what we call *free speed.* This speed is dependent on the design speed of a road, such as road speed restrictions [14].

Free-flow traffic occurs exclusively at low densities, implying large average space headways. As a result, minor local disturbances and spatial patterns of the traffic streams have no significant effects [45]. Therefore, the traffic flow is considered stable in this regime.

Saturated Traffic

Also, called congested traffic. On saturated roads, the flow rate and speed are down to zero [14]. In this regime, we can assume that drivers are more mentally aware and alert, as they have to adapt their driving style to the smaller space and time headways under high speeds [45]. Moreover, when the number of vehicles starts to increase, the density also increases, creating a large disturbance.

Capacity Traffic

The capacity of a road is equal to the maximum flow rate q_c [14]. When the traffic density increases vehicles are driving closer to each other. Consequently, reducing their time headway, indicating the formation of packed clusters of vehicles (i.e., platoons), which are moving at a certain capacity-flow speed v_c which is usually a bit lower than the free-flow speed [45].

2.6.2 Homogeneous and Heterogeneous Traffic

The traffic flow regimes mentioned before are characteristics of homogeneous and heterogeneous traffic. Roads usually show a variety of vehicle types and drivers. If there is only one type of vehicle on the road, which means all vehicles share the same speed and trajectories are straight lines, the traffic state is called stationary or homogeneous. While dealing with homogeneous traffic, it is easy to collect and extract microscopic characteristics such as space headways (Eq. 2.1) or time headways (Eq. 2.2), and macroscopic characteristics such as flow, speed, and density [48]. However, including different drivers and vehicles is crucial when modeling the effects of active traffic management such as speed limits, message signs, or when simulating traffic-related effects. This variety of drivers and vehicles is what we call heterogeneous traffic. Heterogeneous traffic can be microscopically modeled in two ways [11]:

1. All drivers or vehicles are described by the same model using different parameter values.

2. Different drivers or vehicles can also be described with different models.

Both methods have various applications. For example, if we aim to build different vehicle classes like cars and trucks, we can choose the first method to distribute different parameters for each class. If we aim to build different classes directly we can choose the second method. This method allows us to build different driving systems such as human driving and semi-automated driving that can be modeled in the same environment with the help of adaptive cruise control systems [11].

2.6.3 Fundamental Relationship

We just reviewed the basic traffic flow characteristics: density k, mean speed u, and flow q. The relationship between these characteristics is what we call the fundamental relationship. This relationship is the following:

$$q = ku \tag{2.22}$$

Equation 2.22 shows that the flow q is proportional with both the speed u, and the density k. Intuitively, this makes sense because when the whole traffic stream moves twice as fast if the flow doubles. Similarly, if the density doubles, the flow doubles as well [46]. Thus, when two of these parameters are known, the third can be estimated.

There are many shapes proposed for the fundamental diagram. There are a multitude of models describing the relationships between each pair of these parameters have been developed over the years [2]. Even today, new shapes are proposed [46]. For the purpose of this work, we will only be reviewing the most used models.

Greenshields

The first traffic stream model was developed by Greenshields [13], who observed a platoon of vehicles and checked the density of the platoon and their speed [46]. Figure 2.8 show the relationships created by Greenshields' model.

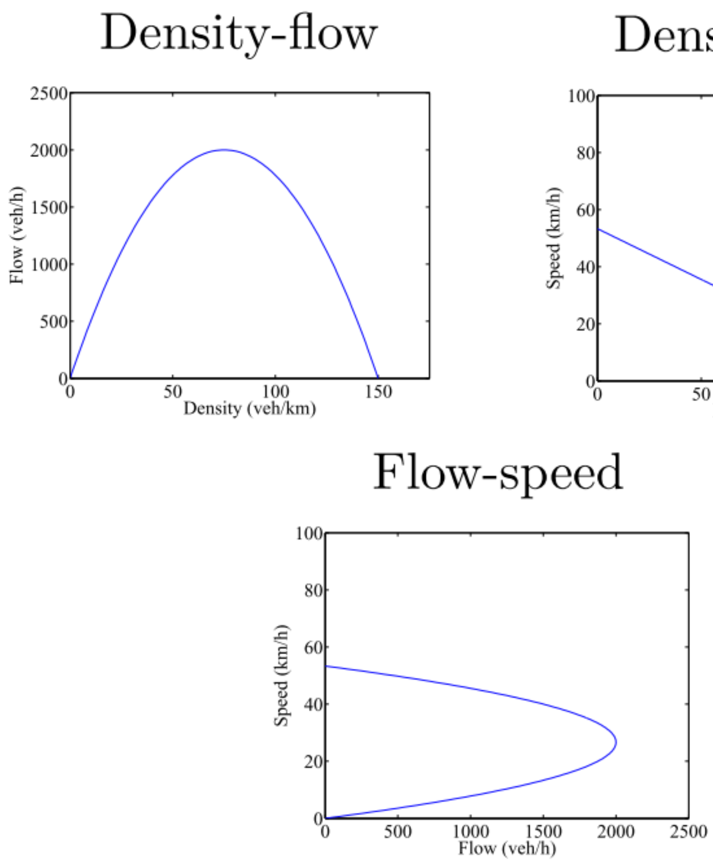


Figure 2.8: Flow, speed, and Density Relationships from Greenshields. Image from [46].

Greenshields derived the speed-flow relationship based on the assumption of a linear speed-density relationship. Also, Greenshields assumed that the speed-density relationship is linear [2]. The following equation describes this relationship:

$$u = u_f - \frac{u_f}{k_j}k \tag{2.23}$$

where
u_f is free-flow speed,
k_j is jam density.
Applying the fundamental relation gives the other relations [14]:

$$q = \frac{u_f}{k_j}k(k_j - k) \tag{2.24}$$

$$q = k_j u(1 - \frac{u}{u_f}) \tag{2.25}$$

The Greenshields model has been used extensively in transportation analysis because of its simplicity and for historical reasons. However, field observations do no support its shape [2]. Since, for a range of low densities, drivers keep the same speed, possibly limited by the current speed limit [46].

Triangular Diagram

The fundamental diagram which is often used in academia is the triangular fundamental diagram [46]. This formulation assumes that the fundamental $k - q$ diagram is triangular in shape [14]. Figure 2.9 describes the relationships in the triangular diagram.

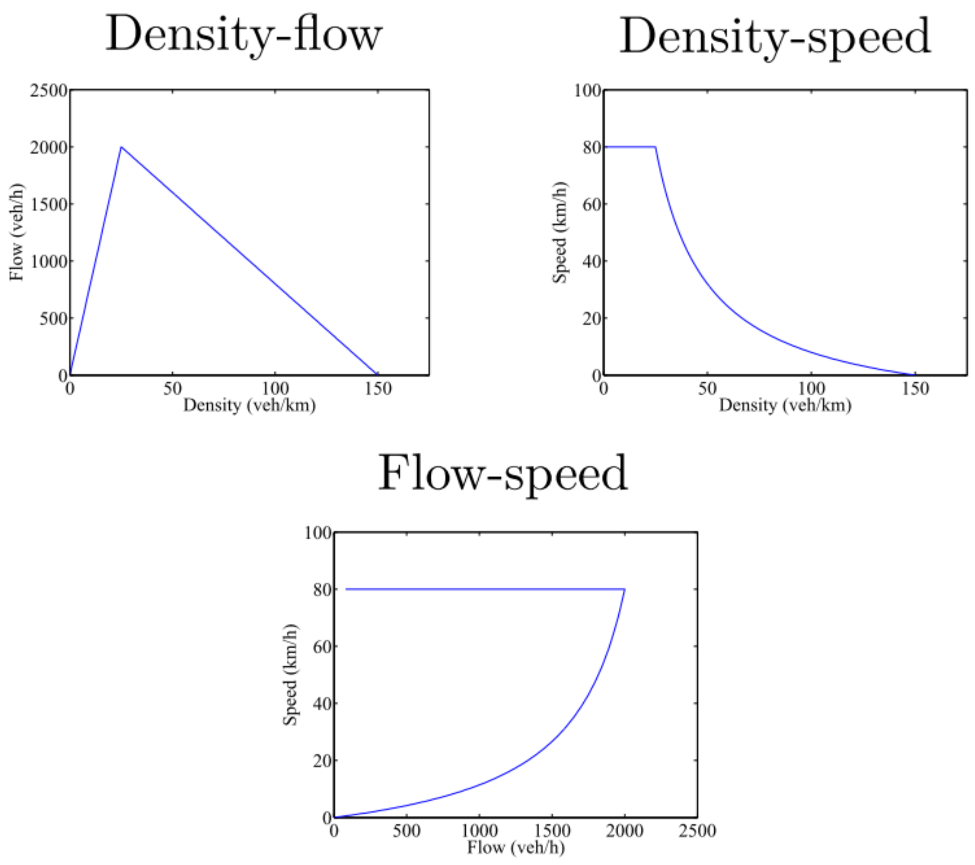


Figure 2.9: The fundamental diagrams when a triangular k-q diagram applies. Image from [46].

The equation is as follows [46]:

$$q = \begin{cases} v_0 k, & \text{if } k < k_c \\ q_c - \frac{k-k_c}{k_j-k_c} q_c, & \text{if } k >= k_c \end{cases} \tag{2.26}$$

2.7 Applications of Traffic Flow Simulation

We just reviewed the core part of traffic simulators which are the car-following models. But we have not yet reviewed which are the applications of these traffic simulators. In transportation, simulation is used to study various aspects of the system such as traffic operations, demand modeling, travel time and fuel consumption [2]. In this section, since the applications that simulators have are many, we will mention only those most commonly used.

2.7.1 Freeway Systems

When demand is below capacity, we can evaluate the freeway segments in isolation and determine its operational performance, such as its expected density and speed. When the performance of a freeway is not the desired one, an advanced traffic management method for freeway facilities can be implemented. These methods are ramp metering, variable speed limit systems, HOV/HOT lanes, shoulders, among others.

There are some methods for evaluating freeway segments, freeway waving segments, merge and diverge segments as well as freeway facilities [2]. Some of these methods are specified in the highway capacity manual [49] (HCM). On the other hand, the use of microsimulation for freeway facilities has increased in recent years. The strength of simulation is that it can analyze various types of configurations that HCM cannot address. For example, in traffic simulation, we can evaluate systems in combination, such as freeway systems with arterial systems. Also, stochastic simulators can consider the randomness inherent in traffic operations due to the differences in driver behavior as well as the wide variability in the vehicle capabilities [2]. Both methods, HCM, and microsimulation have their advantages and disadvantages. For example, in the HCM 2010, free-flow speed is estimated using the number of ramps per mile. However, in the simulator, the ramps are directly replicated in the network, and their effect is already considered in this manner [2]. Still, HCM is more accurate when it comes to considering geometric effects such as lane and shoulder widths. Thus, the user should be cautious in choosing which method to use depending on the objective of the analysis.

2.7.2 Travel Time

For most persons, the individual travel time is the most important aspect when planning a route from a giving origin to a giving destination. Aside from the human rationale, there is also an increased interest in obtaining precise information with respect to travel times in the context of advanced information systems (ATS) [45]. The methods to estimate these quantities can be applied directly to a stationary detector and probe-vehicle data, or microscopic and macroscopic models [11].

Travel time is the time that takes to travel from a given origin to a given destination. Travel time is the reciprocal of speed [2]. Mathematically, the travel time (TT) for a given distance d is:

$$TT(h) = \frac{d}{v_{avg}} \tag{2.27}$$

where
d is the distance from a given origin to a given destination,
v_{avg} is the average speed throughout the trip

The problem of measuring travel times is compounded when there are multiple origins and destinations, and when we attempt to match multiple vehicles at origins to multiple destinations [2]. Several techniques for estimating the current travel time are being used, such as probe vehicles, cumulative plots, among other methods that can be found in the literature. The travel time data collection handbook [50] explains many of these methods. However, travel time is much easier to define and measure when using a simulator or when observing travel times from within a vehicle [2].

In microscopic simulations the complete information on any vehicle is available at any time so that the individual times can be directly determined from the trajectories [11]:

$$\tau_{12}(t) = t_2^{\alpha} - t_1^{\alpha}, \hat{\tau}_{12}(t) = t_2^{\beta} - t_1^{\beta} \tag{2.28}$$

where
$\tau_{12}(t)$ is the realized travel time, is the time a vehicle needs to travel from x_1 to x_2 when it leaves the considered road section at time t.
$\hat{\tau}_{12}(t)$ is the expected travel time, is the time a vehicle needs to travel from x_1 to x_2 when it enters the considered road section at time t.
α and β denote the last vehicles that have left and entered the investigated road section.

2.7.3 Fuel Consumption and Emissions

Calculating fuel consumption and emissions is a typical offline analysis step that uses the data previously obtained by simulations or observations [11]. There are various ways to estimate the fuel consumption of a vehicle. If we consider an individual vehicle, the calculation of its energy consumption requires only the knowledge of its kinematics [51]. This can be done on real traffic-data, but also can be done by using data from microscopic simulators.

There are several models to estimate fuel consumption and emissions. Depending on the aggregation level and level of detail, these are classified into different categories:

- Macroscopic Models: Area-wide models, average-speed models, traffic-situation models, traffic-variable models.
- Microscopic Models: Speed-profile models, modal emission models.
- Physics-Based Modal Consumption Model.

For example, on a microscopic physics-based, generally, the level of detail is higher. Therefore, it takes speed and acceleration profiles as input; this data can be obtained directly from microscopic simulations. The outputs of these models are instantaneous fuel consumption and emission rates, such as CO2, on a single-vehicle basis [11]. Recent work, such as the Study [51], has shown that Gipps' car-following model can be calibrated to estimate fuel emissions in trucks. However, the author points out that better results could be achieved if the models were calibrated from the point of energy consumption.

Chapter 3

State of the Art

This chapter presents recent research on the Gipps' car-following model. The chapter begins with a compilation of simulators that use a current implementation of the original model or a modification of it. It is worth mentioning that the majority of papers where the operation of these simulators is reported were published are from a few years ago. However, due to their relevance, these simulators are still being developed and used in academia or commercially to this day. Also, recent research on the Gipps' model, its modifications, calibration, and new uses are presented.

3.1 Simulators

3.1.1 AIMSUN

The Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks (AIMSUN) was initially developed back in 1989 [52]. Nowadays, the software includes macroscopic, mesoscopic, and microscopic models, and is one of the most important providers of software and services for traffic planning, simulation, and prediction [53]. The simulator is being used by more than 2600 users in 65 countries.

Several models dealing with individual vehicles are part of AIMSUN. Where each vehicle/driver have individual behavioral attributes assigned to them when they enter the system. Figure 3.1 shows a general flowchart of the microscopic simulation model used in AIMSUN; in the flowchart we can notice that the behavior of vehicles is simulated by two models: Car-Following and Lane-Changing.

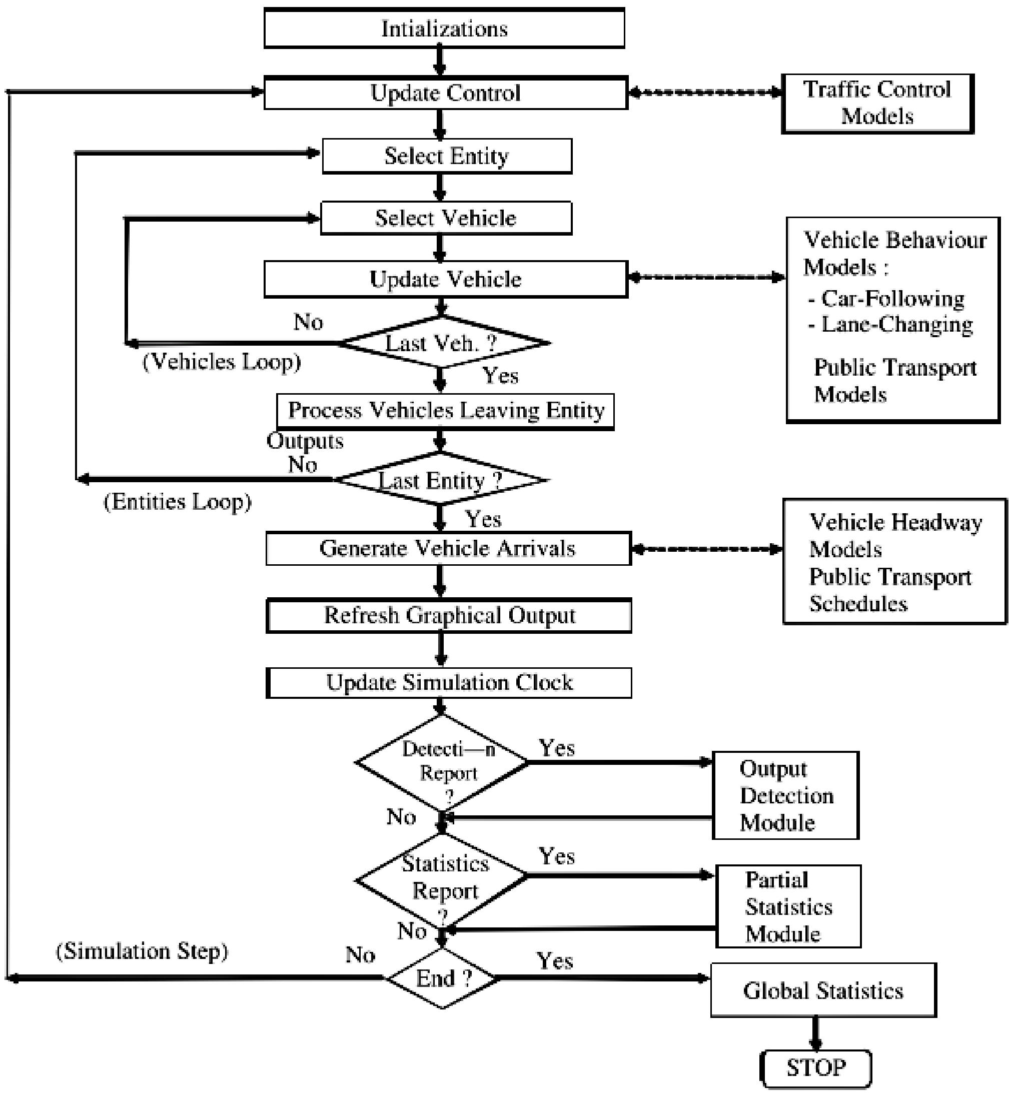


Figure 3.1: The microscopic simulation process in AIMSUN. Image from [53].

The car-following model implemented in AIMSUN is based on the Gipps' model [8] with some modifications for better results. As pointed out in the Study [10], a common drawback of how Gipps' model is implemented is that the parameters are global. Whereas it is well known that the behavior of a driver is affected by traffic conditions. Instead, in AIMSUN the model is implemented by the influence of local parameters depending on various factors such as the type of driver, the road characteristics, and the influence of vehicles on adjacent lanes [53]. Equation 2.9 is also used in the simulator, with the difference that the desired speed of the driver V_n is implemented as the desired speed of vehicle n for the current section [10]. The following parameters are used to calculate the maximum desired speed of a driver in a particular section [53]:

1. Maximum desired speed: $V_{\max}(n)$.
2. Speed acceptance of the vehicle: $\theta(n)$.

3. Speed limit of the section or turning: s: $S_{\text{limit}}(s)$.

The speed limit of the section or turning of a vehicle n is calculated as follows:

$$s_{\text{limit}}(n,s) = S_{\text{limit}}(s) \cdot \theta(n) \tag{3.1}$$

Then, the maximum desired speed of the vehicle n, is calculated as follows:

$$V_{max}(i,n) = MIN\{s_{\text{limit}}(n,s), V_n\} \tag{3.2}$$

where, V_n is the same parameter used in the original Gipps' formulation (see Eq. 2.11).

3.1.2 DRACULA

The DRACULA traffic microsimulation model was developed at the University of Leeds in 1993. The simulator is used as a tool to investigate the dynamics between demand and supply interactions in road networks [32]. DRACULA has only been used within intelligent transportation systems (ITS) so far but is still available for use on research projects. Figure 3.2 shows an image of traffic flow from the simulator.

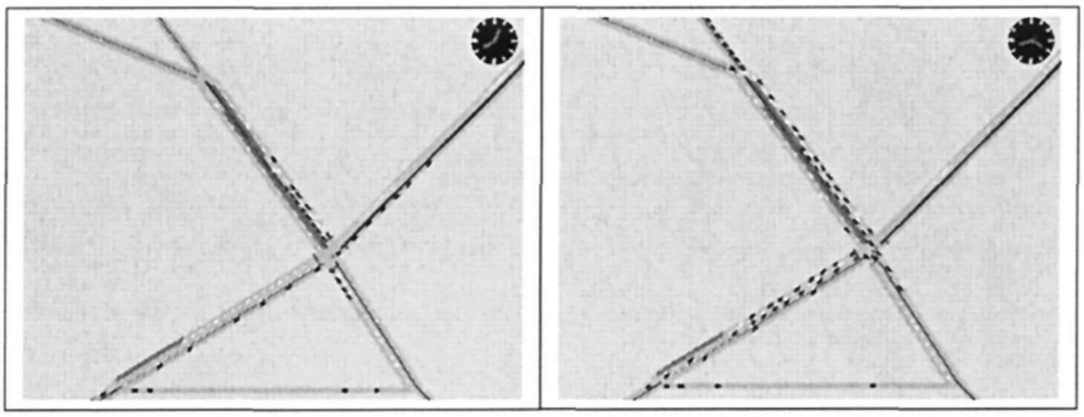


Figure 3.2: Simulated traffic with DRACULA. Image from [54].

The complete DRACULA framework combines a number of sub-models:

- Demand Model
- Traffic Simulation Model
- Learning Model

We will focus only on the traffic simulation model. This model is in charge of the movement of vehicles through the network. Drivers follow their pre-determined routes, and en-route. The vehicles can encounter traffic signals, queues and interact with other vehicles on the road. Vehicle movements in a network are determined by the desired speed of the driver, its response to traffic regulations, and interactions with neighboring vehicles [54]. In this sense, to reproduce this behavior, DRACULA uses a car-following model and a lane change model.

The car-following model that DRACULA uses is based on the rules of Gipps' model [8]. Equation 2.9 which gives the speed of each vehicle n at time $t+\tau$ is reproduced in the DRACULA model to simulate the speed on urban streets. While, in other situations to reproduce other traffic dynamics, such as high-speed, and long motorway links, a modification of the model is used. This modification was built on how drivers behave differently depending on different traffic conditions [32]. Figure 3.3 shows that the driver behavior is assumed to be on different alertness "states". These states are: "non-alert state", "alert state", and "close following" states [54]. Where Gipps' car-following model is being used to replicate the behavior of the non-alert and the alert states; but with a slight modification of the parameters reaction time, and acceleration.

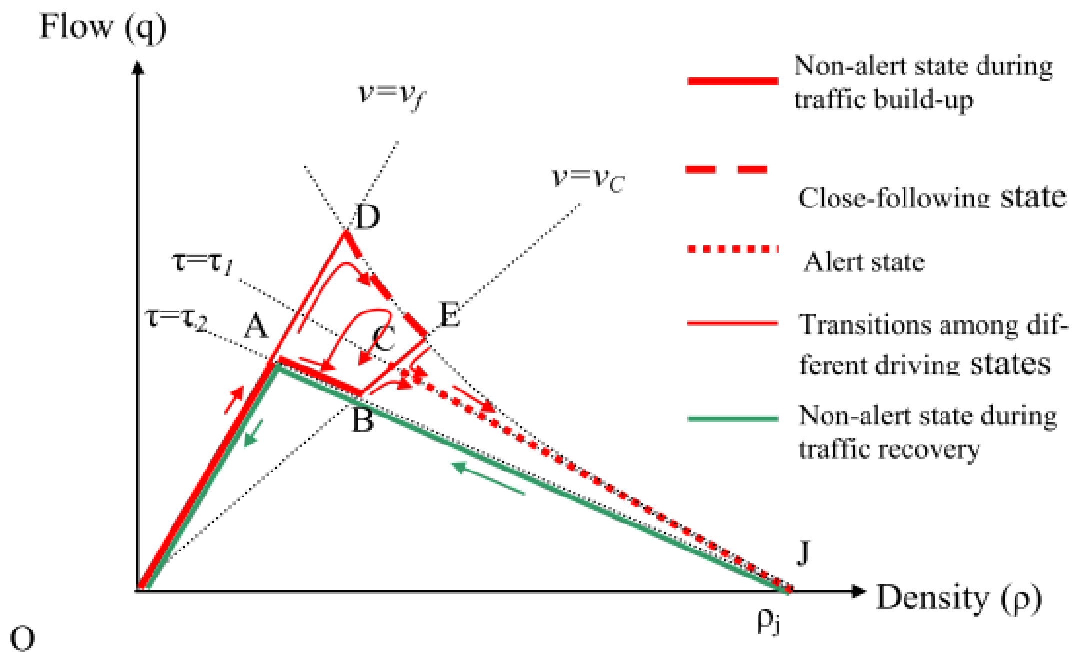

Flow (q)
v=v_f
v=v_C
τ=τ_1
τ=τ_2
A
B
C
D
E
J
O
ρ_j
Density (ρ)
Non-alert state during traffic build-up
Close-following state
Alert state
Transitions among different driving states
Non-alert state during traffic recovery

Figure 3.3: Transitions between different driving states modeled in DRACULA. Image from [32].

3.2 Calibration of Gipps' Car-Following Model.

Calibration is the iterative process of comparing the model with a real system and revising the model by making modifications to built parameters if necessary. This process is repeated until the model can accurately represent the real system [17]. This iterative process is equivalent to the solution of a constrained minimization problem where an objective function expresses the deviation of the simulated results from those observed [29]. The calibration process is usually done at an aggregated level, using macroscopic variables such as speed, flow, and density. However, recent research has focused on different procedures depending on the type of vehicle, cost of the procedure, type of data, etc.

3.2.1 Calibration of the Gipps' Car-Following Model Using Trajectory Data

Recent studies like [55] consider new methods for the calibration of Gipps' model. The central assumption of this study is that each parameter that describes a vehicle/driver has a physical meaning. Therefore, these parameters can be estimated by conducting specific experiments. The data used in this research was collected by using two instrumented vehicles. The first vehicle being the leader and the other one being the follower. Each vehicle was equipped with a data logger device (DL1 MK3) from Race Technology. These data loggers have several sensors, such as an accelerometer and a GPS, that allow obtaining the position and time data of the vehicles on the road. Also, a LIDAR rangefinder was connected to the follower vehicle to provide real-time distances to the leader vehicle [55]. The parameters calibrated in this study were the desired speed V_n and maximum acceleration a_n. Various experiments were performed, but here, we are going to only describe the results of the simplest experiment, which is the approximation of the vehicles to a stop line. This experiment was performed on arterial roads and distributor roads. The two figures at the top of Figure 3.4, plots Estimation vs. Measured values of acceleration for one of the observations made.

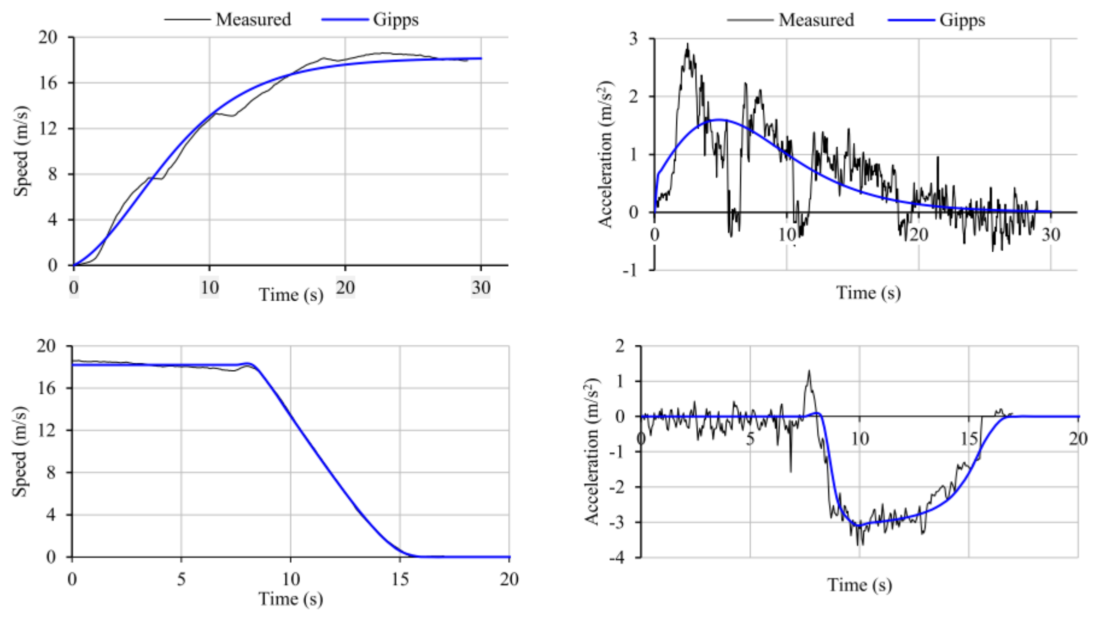


Figure 3.4: Results of optimal parameters for speed and acceleration. Image from [55].

Meanwhile, the figures at the bottom indicate the results of the vehicles approaching the stop line. The optimal values of the parameters found for this observation were an acceleration of $a_n = 1.60m/s^2$, and a desired speed of the driver of $V_n = 18.2m/s$. These values made the Gipps' model capable of reproducing typical acceleration maneuvers accurately. Whereas the deceleration maneuvers require the adjustment of the deceleration parameter b_n [55]. Also, other experiments to calibrate the reaction time τ and minimum spacing d_n were conducted. The importance of this Study [55] was to test a method to calibrate Gipps' car-following model that can be easily replicated in the future by other researchers.

3.2.2 Calibration of Gipps' Car-Following Model for Trucks and the Impacts on Fuel Consumption Estimation.

Recent studies have also focused on the importance of calibrating car-following model parameters to estimate fuel consumption correctly. This plays an important role since recently traffic simulators are being used as an alternative to obtaining data, especially on microscopic simulators where the level of detail is higher. Additionally, it is worth mentioning that truck's reactions to traffic are different from those of cars [56, 57]. For example, the spacing between vehicles is larger, acceleration capabilities are smaller and speed profiles are more complex. For these reasons, the Study [51], evaluate the calibration of Gipps' model to reproduce truck's behavior. It also evaluates the calibrated model to see if it can replicate accurate results when calculating fuel consumption.

The experiment was conducted in Lyon (France). The data were obtained using a heavy vehicle equipped with sensors measuring the truck speed and the spacing of the vehicle ahead. Then the data is sampled, obtaining more information at each time step, such as the position of the vehicle, its speed, its acceleration, the fuel-injected, and the slope [51]. Moreover, the calibration method used was the goodness-of-fit function (GoF). The calibration provided by the study has shown promising results. As shown in Figure 3.5, the errors on position and speed are very low for all cycles.

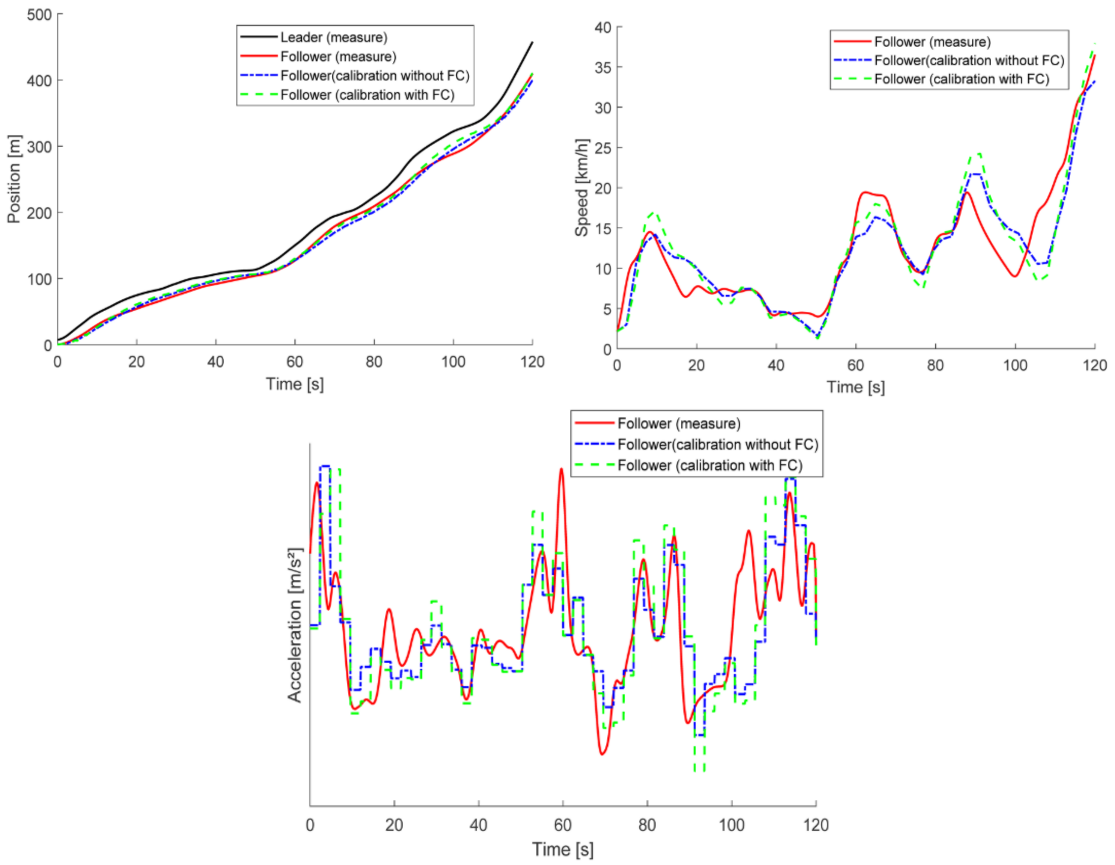
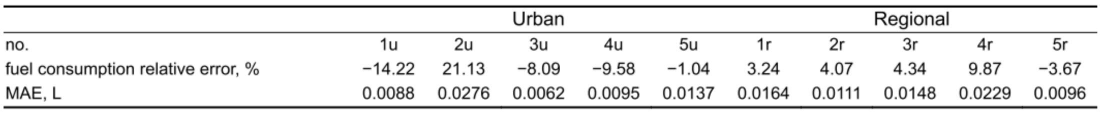


Figure 3.5: Measured and simulated trajectory of the truck after the calibration of parameters. Image from [51].

However, the errors on acceleration are higher than on the other variables. This can be explained by the fact that Gipps' model output is the speed which is derived to obtain acceleration [51]. For this reason, the author safely states that Gipps' car-following model could be relevant to simulate the behavior of a heavy vehicle.

Now, regarding the estimation of fuel consumption. The data were obtained using an internal tool. This tool takes as input the speed profiles generated with the calibrated Gipps' model, which computes the corresponding fuel consumption. The results are presented in Figure 3.6.



	Urban					Regional				
no.	1u	2u	3u	4u	5u	1r	2r	3r	4r	5r
fuel consumption relative error, %	−14.22	21.13	−8.09	−9.58	−1.04	3.24	4.07	4.34	9.87	−3.67
MAE, L	0.0088	0.0276	0.0062	0.0095	0.0137	0.0164	0.0111	0.0148	0.0229	0.0096

Figure 3.6: Fuel consumption estimation relative errors between the simulation and the measure. Image from [55].

These results show that there is a significant gap between the total fuel consumption measured and the simulation expressed in $L/100km$, and the corresponding MAE value [51]. For this reason, the authors suggest, that car-following models should not only be calibrated from a traffic point of view but also from an energy consumption point of view.

3.3 Data-Driven Car-Following Models

So far, we have reviewed in-depth the Gipps' model, and a brief introduction of other car-following models. These models pertain to the class of physics-based models. Although these models successfully interpret traffic phenomena observed in the real-world and have produced significant achievements in microscopic traffic simulation, they may not fully capture the complex cognitive process of driving [58]. With the development of data-driven technology and its widespread application in the field of transportation [59], recent years have witnessed a growing interest in modeling car-following behavior using data-driven methods thanks to big data generated from connected and automated vehicle technologies [60]. Data-driven car-following models refer to cutting-edge models that mimic human intelligence, leveraging deep neural networks, reinforcement learning, imitating learning, and other advanced machine learning methods. There is no predesignated mathematical form, and model training purely relies on observations [58]. In this section, we present some of the most recent studies regarding data-driven models.

3.3.1 Autonomous Car-Following Model with Deep Reinforcement Learning.

Nowadays, we are confronting a transition period between human-driven vehicles and self-driving vehicles. Thus, it is critical to ensure that autonomous vehicles interact with human-driven vehicles safely. The goal of autonomous vehicles is that the vehicle can adapt to the space headway, speed, and trajectory of the vehicle in front (car-following).

For this reason, it is useful to introduce a driver model that reproduces individual drivers' behaviors and trajectories. This is the main idea behind M. Zhu et al. Study [61]. This study is the first to use deep reinforcement learning to model car-following behavior.

Figure 3.7 shows a diagram of a human-like car-following framework based on deep reinforcement learning. The first stage is the data collection and storage of this data. This data is then fed into the simulator where an agent learns from trial and error interactions by the environment. These interactions include a reward function that signals how much the agent deviates from the empirical data [61]. Through these interactions, the car-following model is developed.

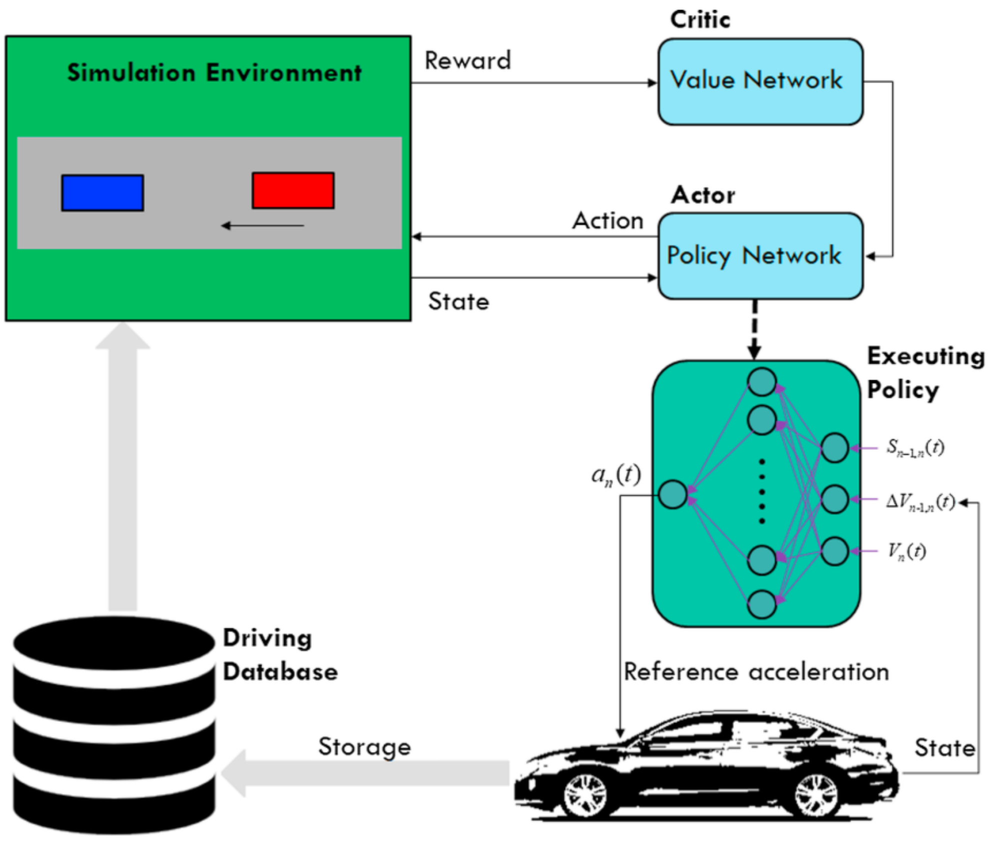
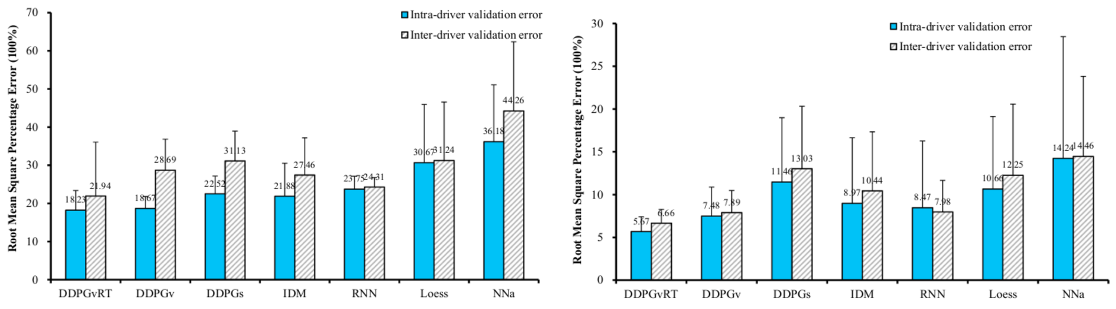


Figure 3.7: Conceptual diagram of human-like car following by deep reiforcement learning. Image from [61].

Data-driven methods are more flexible than traditional models since they contain a large number of parameters, usually over hundreds or thousands [58], and thus lead to richer models. However, this large number of parameters demands large training data and incurs high computational costs.

The DDPGvRT model proposed in the Study [61], provides promising results. The model was compared with the intelligent driver model (IDM) and other models based on neural networks and deep learning with different reward functions. These models were compared in the intra-driver validation datasets. The results of the performance of the model are presented in Figure 3.8.



(a) Spacing (b) Speed

Figure 3.8: Mean and standard deviation of intra- and inter-driver validation errors on spacing and speed for the seven models. Image from [61].

The DDPGvRT model shows the lowest mean values and standard deviation errors for both spacing and speed, outperforming the other models investigated in [61]. It is worth mentioning that the only model pertaining to the class physics-based was the IDM model, which already shows a better performance than some data-driven models.

Chapter 4

Methodology

This chapter presents in detail our implementation of Gipps' car-following model for traffic simulation. The chapter begins with a brief explanation of the phases that will be followed for the implementation of the model. Then, an explanation of the model is made. Afterward, a description of the steps performed to implement the model is made. Finally, we evaluate the simulation, comparing our results with those of the literature.

4.1 Phases of Problem Solving

Figure 4.1 shows the work-flow that has been used in order to perform this project.

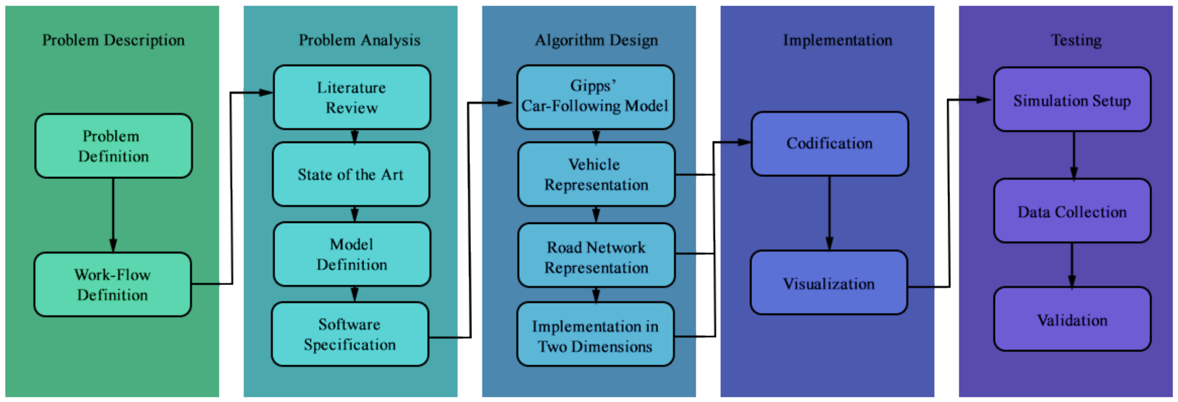


Figure 4.1: Phases of Problem Solving.

4.1.1 Description of the Problem

This is the atomic phase for the project. This phase presents the original model, its possible problems, and how the model could be implemented. First, we reviewed the original Gipps' car-following model, its parameters, and the importance of the model nowadays. Later, we discussed the advantages and disadvantages of this model, and how these disadvantages could be solved. Subsequently, we reviewed how the model is currently being implemented.

Finally, we coded the model in the Python programming language. The steps mentioned in this phase are reflected in Chapters 2 and 4.

4.1.2 Analysis of the Problem

This phase consists of the background and fundamental knowledge that allow us a deep understanding of this project. Chapter 2 presents this information in a relevant order for the development of the project. Firstly, a brief review of what is traffic flow theory and its history is presented. Then, we reviewed the traffic flow models classification by the aggregation level. Consequently, the fundamental traffic flow characteristics were analyzed to understand the most common applications of traffic flow simulation.

4.1.3 Algorithm Design

In this phase, the algorithm to implement Gipps' model was designed. The algorithm allows us to create an n number of vehicles with different parameters. The movement of each vehicle is governed by the Gipps' model. Also, we added angles (not steering) to the original model to achieve the movement of the vehicles in two dimensions. Finally, the road network was designed, making use of graphs.

4.1.4 Implementation

This phase consists of the codification of the model in a specific programming language. The chosen programming language was Python. Python is an interpreted language, it allows us to run the simulation in other operative systems. Also, Python offers libraries such as NumPy [62], which allows the implementation of the model to be much easier. The paradigm of the chosen programming language was the object-oriented paradigm. This paradigm allows us to create different objects of the class road and vehicle. Thus, the user can generate different parameters of the vehicles to replicate both, homogeneous traffic and heterogeneous traffic. We chose the Matplotlib framework for the simulation visualization since it gives us simple tools to be able to visualize both the simulation and plot the relationship between data.

4.1.5 Testing

This phase focuses on analyzing the performance and evaluation of the model. To achieve this, we compared the simulation results with previous results from the literature. The simulation was run in a controlled environment, that is, with parameters similar to those of the literature already proposed. Additionally, we recreated both homogeneous and heterogeneous traffic. Then, we evaluated the model behavior in much larger networks, with more vehicles, and with the chance for vehicles to move in two dimensions. A more in-depth explanation of these steps can be found in Section 4.3.

4.2 Proposal

This section deals with the implementation of the simulation for Gipps' car-following model. First, a brief explanation of Gipps' car-following model is made, then we review the representation of each class in the simulation and the logic behind the movement of vehicles in two dimensions.

4.2.1 Gipps' Car-Following Model

A review of the history of car-following models and a more in-depth introduction to Gipps' model can be found in Chapter 2. However, as a reminder, this is the original formulation of the model [8]:

$$v(t+\tau) = min\{G_a(t), G_d(t)\}, \tag{4.1}$$

where,

$$G_a(t) = v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)\sqrt{0.025 + v_n(t)/V_n}, \tag{4.2}$$

$$G_b(t) = b_n\tau + \sqrt{b_n^2\tau^2 - b_n(2(x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}} \tag{4.3}$$

where,
$v_n(t)$ is the speed of vehicle n at time t,
$a_n(t)$ is the desired acceleration of the driver,
τ is the apparent reaction time of the driver,
V_n is the desired speed of the driver,
b_n is the most severe braking ($b_n < 0$) of the driver,
$x_n(t)$ is the location of the front of the vehicle n,
s_{n-1} is the effective size of vehicle $n-1$ which is the vehicle length plus a safe space which the following vehicle is not willing to intrude,
$\hat{b}$ is the value of b_{n-1} estimated by the driver of vehicle n.

The model has an n number of vehicles. Where vehicle $n-1$ is the leader and n is the follower vehicle. The speed obtained by a vehicle at a time $(t+\tau)$ is represented by the Equation 4.1, which is the minimum between Equation 4.2 and Equation 4.3. As we can see, the model takes the form of a system of differential equations involving the common reaction time τ of drivers. The general idea is that Equation 4.2 is estimated by only taking into account the characteristics of the vehicle n. Meanwhile, Equation 4.3 is estimated by taking into account the characteristics of the leader vehicle, such as its position x_{n-1}, speed v_{n-1}, effective size s_{n-1} and braking properties $\hat{b}$. Thus, we can assume that Equation 4.2 is being used if the movement of the vehicle is not impeded by other vehicles in front. Therefore, the vehicle is flowing freely. Meantime, Equation 4.3 is being used if the movement of the vehicle is impeded by a vehicle in front. Thus, the vehicle speed is also dependant on the properties of the leader.

4.2.2 Vehicle Representation

Vehicles are represented using the Python constructor called class. This class contains multiple attributes. Whereas, the most important attributes are extracted directly from the model. These parameters should correspond to obvious characteristics of drivers and vehicles. Gipps [8], to mimic real traffic behavior, sample some of these parameters from a normal distribution. This distribution is also used in the simulator to generate heterogeneous traffic.

- a_n sampled from $N(1.7, 0.3^2)m/sec^2$,
- b_n equated to $-2.0a_n$,
- s_n sampled from $N(6.5, 0.3^2)m$,
- V_n sampled from $N(20.0, 3.2^2)m/sec$,
- $\tau = 2/3$ second,
- $\hat{b}$ picked from $min(-3.0, (b_n - 3.0)/2)m/sec^2$.

4.2.3 Road Network Representation

The representation of the road network is straightforward. The network is represented by a directed graph $G(N, E)$, where N is the set of nodes n_i, and E is the set of edges e_{ij}. The nodes in the graph act like intersections, while the edges act like roads. The creation of the graph is handled by NetworkX [63]. The way the graph is created is by representing the nodes as a (*key:value*) pair in a dictionary data type. The *key* is a string representing the identification of the node, while the *value* is a tuple containing its coordinates. Meanwhile, each edge is represented in a list of tuples. Each tuple contains a pair of nodes that specify a line with a direction joining these two nodes. The following code is an example of the creation of a basic road network:

Listing 4.1: Nodes and edges data representation

```
 nodes = {
        "a": (50, 0),
        "b": (50, 130),
        "c": (180, 130),
        "d": (180, 0),
        "e": (310, 0),
        "f": (310, 130),
    }

edges = [
        ("a","b"),
        ("a", "c"),
        ("b","c"),
        ("c","d"),
        ("e","c"),
        ("d","a"),
        ("e","d"),
        ("f","e"),
        ("c","f")
    ]
```

The code above produces the following visualization, presented in Figure 4.2.

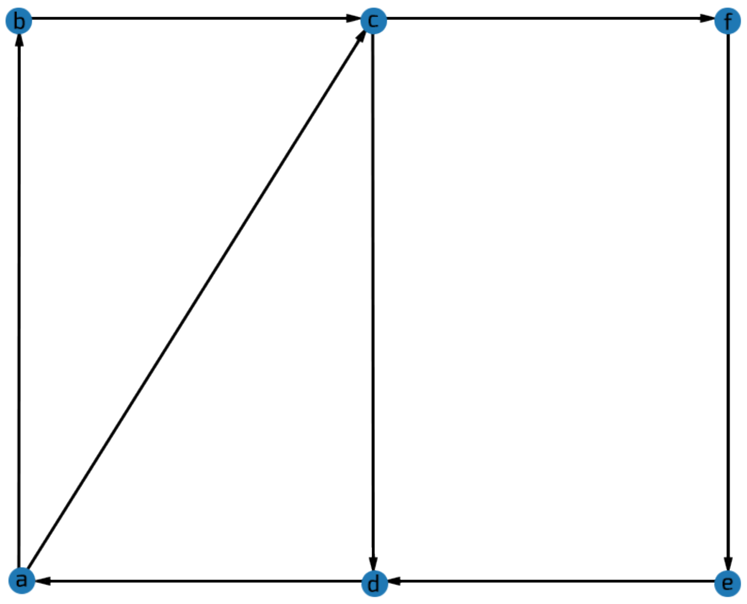


Figure 4.2: Road network representation.

The edges which are connecting two points represent the roads; meanwhile, the nodes represent intersections. Once the graph is created, the data, that was used to create the graph is also used to create a class called *road*. This class contains several attributes and methods. But the attributes *id* and *angle* are the most relevant attributes. The *id*, as its name says, is the number used to identify the road. The attribute *angle* indicates the angle of the road. This angle is used to modify the unidimensional equations of motion given by Gipps' formulation and adapt them to a bidimensional scenario. It is obtained by using a method called *get_angle()*. The following pseudocode shows the algorithm used to obtain this angle:

Listing 4.2: Function get_angle pseudo code

```
function get_angle(start_node_coordinates, end_node_coordinates):
    delta_x = end_node_coordinates[0] - start_node_coordinates[0]
    delta_y = end_node_coordinates[1] - start_node_coordinates[1]
    result  = atan2(delta_y, delta_x)
    return result
```

The function *atan2*, calculates the arc-tangent of $\delta y, \delta x$ giving y and x, which is the subtraction between the x coordinates and y coordinates of the final node and the initial node that connect the edge. Thus, the angle calculated by the algorithm is the angle between the positive x-axis and the road.

4.2.4 Movement of a Vehicle

The vehicle speed and position are the two quantities necessary to represent vehicle movement in the simulation. The speed of the vehicle can be obtained by using Equation 4.1. However, to obtain the position of the vehicle, a numerical method is needed. Several methods can be used to obtain the position depending on the accuracy to which the user would prefer. Below the method used in our implementation is described, which is also the method described by Wilson [33].

Vehicle Position

The analysis of a Differential-Difference Equation System 4.3 is technically difficult. However, Gipps [8] observed that the system behaves well when integrated with a time step τ. If we know that:

$$\frac{d}{dt}x_n(t) = v_n(t) \tag{4.4}$$

We can replace Equation 4.4 with its discretization by a suitable integration scheme; in this case, the trapezoidal rule is used:

$$\frac{x_n(t+\tau) - x_n(t)}{\tau} = \frac{1}{2}v_n(t) + \frac{1}{2}v_n(t+\tau) \tag{4.5}$$

Rearranging this equation we obtain:

$$x_n(t+\tau) = x_n(t) + \frac{\tau}{2}v_n(t) + \frac{\tau}{2}v_n(t+\tau) \tag{4.6}$$

Finally simplifying:

$$x_n(t+\tau) = x_n(t) + 0.5(v_n(t) + v_n(t+\tau))\tau \tag{4.7}$$

Now the Equation 4.7 allows us to calculate the vehicle position x_n at a time $(t+\tau)$. However, this only allows us to move the vehicle only on one axis. If we want the vehicle to move on two axes it is necessary to modify the speed formula.

Movement of a Vehicle in Two Dimensions

Suppose that the positions of the vehicles are given by $X_n = (x, y)$, and each road has angles that define its direction. The speed of the vehicles at time $t+\tau$ can be directly calculated through its components. Therefore, we can multiply the *sine* or *cosine* of the angle of the road in which the vehicle is. For example, if the vehicle is on the road with an *id* 3, as shown in Figure 4.2. The angle of this road is 45°, to calculate the speed and location of this vehicle, we must follow the following algorithm:

1. Calculate the vehicle speeds (4.2 and 4.3) for each coordinate (x, y).

(a) In the case of the coordinate x, Equation 4.3 will look as follows:

$$G_{bx}(t) = (b_n\tau + \sqrt{b_n^2\tau^2 - b_n(2(x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}}) cos(45°) \tag{4.8}$$

(b) In the case of the coordinate y, Equation 4.3 will look as follows:

$$G_{by}(t) = (b_n\tau + \sqrt{b_n^2\tau^2 - b_n(2(y_{n-1}(t) - s_{n-1} - y_n(t)) - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}}) sin(45°) \tag{4.9}$$

2. Calculate the magnitude of the vector speed through its components:

$$G_b(t) = \sqrt{G_{bx}(t)^2 + G_{by}(t)^2} \tag{4.10}$$

3. Repeat 1 and 2, for G_a which is Equation 4.2.

4. The vehicle speed of vehicle will be the minimum between G_a and G_b:

$$v(t+\tau) = min\{G_a(t), G_b(t)\} \tag{4.11}$$

5. Calculate the vehicle position using Equation 4.7.

Since we included the road angle in the speed formulas, the vehicle can move towards the road direction and angle. This implementation allows us to move the vehicles in different directions and even change lanes to follow a path. However, it is worth mentioning that this approach is different from gear steering, which is a more accurate simulation of the change of direction of a vehicle.

Vehicle Path

All vehicles at the beginning of the simulation start with a pre-determined path. A special function from NetworkX is used to create the path from a start node to a destination node. This function is called *shortest_path()*, the function compute the shortest path from a source node to a target node. The initial node is the first node where the car will start from; meanwhile, the target node is generated randomly. Finally, the path is returned as a list data-type, which is an attribute of the vehicle class.

Once the vehicle has its path, the Euclidean distance between the vehicle to the next node from the path is calculated at each time step. The purpose of calculating this distance is that once the vehicle reaches the next node on its path, the vehicle can change direction to the next node. However, due to the simulation time step, the vehicle usually ends up changing its direction a few meters after reaching the node. Figure 4.3, shows the Euclidean distance between a vehicle veh_n and a node n_i.

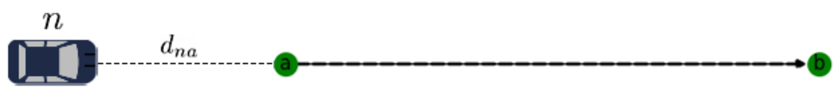


Figure 4.3: Euclidean Distance between a vehicle and a node.

However, to solve this problem, another approach is proposed. This approach consists of not only calculating the Euclidean distance between the vehicle and the next node. But also calculating the Euclidean distance between the current position of the vehicle and its previous position. If the Euclidean distance between the vehicle and the node is less than the distance of the current position and the previous position of the vehicle; the next position of the vehicle is equal to the node location. The following pseudo-code shows the algorithm that implements this logic:

Listing 4.3: Logic when a vehicle arrives at a node to change direction

```
first_distance = euclidean_distance(current_vehicle_location,
    node_location)
second_distance = euclidean_distance(current_vehicle_location,
    last_vehicle_location)
IF first_distance < second_distance:
     next_vehicle_location = node_location
```

This approach allows vehicles to stay on the road when changing direction. Although this method is better than the previous method. This method is not an accurate approach since the vehicle backs up a few meters to stay on the road. Another much more accurate method would be to implement the deceleration of the vehicle and traffic signals. Thus the vehicle can come to a complete stop when it reaches a node. However, this is a more complex approach, and it is not the purpose of this work. But, an example of how other authors solve this problem can be found in Chapter 6.

4.3 Experimental Setup

This section describes the setup for each simulation. Fourth simulations will be run with different vehicle parameters and networks. The objective of each simulation is explained in its corresponding section. Also, the visualization of each simulation is presented.

4.3.1 First Simulation

The purpose of the first simulation is to obtain data that, when visualized, these plots can be compared with those in the literature. For this reason, our aim with this experiment is to replicate homogeneous traffic conditions. Therefore, the parameters and setup for this simulation will be very simple:

- Simulation time: 60 seconds
- Number of vehicles: 10

The first vehicle will be placed directly at the first node of the network. After the first vehicle the rest of the vehicles will be placed with the following separation distance:

$$x_{n-1} = x_n - L_n - d_n \tag{4.12}$$

Vehicle Parameters

As said before, the parameters of each vehicle in the simulation will be the same. Here is a list of these parameters:

- $a_n = 1.7\ m/s^2$
- $\tau = 2/3\ s$
- $V_n = 20\ m/s$, which is the speed limit of the road.
- $L_n = 5\ m$
- $d_n = 2\ m$
- $b_n = -2.0a_n$
- $\hat{b} = min(-3.0, (b_n - 3.0)/2) m/sec^2$

Visualization

The road network consists of a single road which has a length of 1000 meters. We want to evaluate parameters such as speed, acceleration, and position with respect to time. These relationships are much easier to observe when the movement of the vehicles is on a single axis. In other words, we are creating homogeneous traffic on a single road. The visualization of this simulation is presented in Figure 4.5.

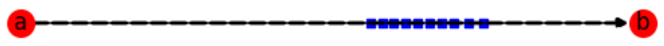


Figure 4.4: Vehicles moving in 1000 meters road, with a speed limit of 22,22 m/s.

4.3.2 Second Simulation

Whereas, the purpose of the first simulation was to create homogeneous traffic. Instead, the purpose of the second simulation is to create heterogeneous traffic. Consequently, the vehicle parameters will be randomly generated. The simulation time, the number of vehicles and the starting position of each vehicle remains the same:

- Simulation time: 60 seconds
- Number of vehicles: 10

Vehicle Parameters

As we just mentioned, each vehicle will have different parameters. The desired acceleration a_n, desired speed V_n, and effective size S_n will be sampled from a normal distribution. The distribution of these parameters was explained in Section 4.2.2. The parameters τ, b_n, and $\hat{b}$ remains the same as Simulation 1. The exact parameters of each vehicle can be seen in Table 4.1.

Table 4.1: The parameters of each vehicle for the second simulation. These parameters were obtained from a normal distribution.

Parameters			
Vehicle	Desired acceleration a_n	Desired Speed V_n	Effective Size S_n
n1	2.143 m/s^2	23.043 m/s	6.82 m
n2	1.508 m/s^2	20.249 m/s	6.75 m
n3	1.714 m/s^2	20.813 m/s	6.68 m
n4	1.590 m/s^2	17.735 m/s	6.08 m
n5	2.439 m/s^2	20.072 m/s	6.28 m
n6	1.226 m/s^2	21.128 m/s	6.54 m
n7	1.406 m/s^2	23.455 m/s	6.36 m
n8	1.485 m/s^2	22.247 m/s	6.22 m
n9	2.414 m/s^2	22.698 m/s	6.94 m
n10	1.131 m/s^2	25.525 m/s	6.54 m

Visualization

Despite the same road is simulated. We can notice a different behavior of the vehicles. Since the parameters of each vehicle are different, the behavior of each vehicle is also different. As we can see in Figure 4.5, certain vehicles reach higher speeds and are quickly reaching the end of the road. While other vehicles are prevented from advancing by the vehicle in front; creating congestion on the road. This behavior is what we call heterogeneous traffic.

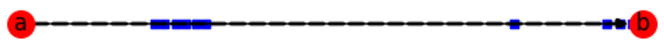


Figure 4.5: Vehicles with different parameters moving in 1000 meters road with no speed limit.

4.3.3 Third Simulation

The third simulation is more complex than the previous ones. The first objective of this simulation is to evaluate the behavior of the model concerning the movement of vehicles in two dimensions. The second objective is to simulate much more vehicles with a higher simulation time. What we try with these different values, is to obtain a lot more data, that would help us to build macroscopic variables such as the flow and mean speed.

The simulation will be run for 10 minutes where every minute 20 vehicles will enter the network. These vehicles will move from node *a* to node *c2*. Once vehicles reach point *c0*, their speed will be recorded as if it were a speed sensor. With this information, we can calculate the flow rate of vehicles at point *c0*. This procedure will be repeated 200 times to generate samples that allow us to obtain the mean speed and the vehicles/hour that have crossed this point. Meaning that 40,000 vehicles traveled the road from point *a* to point *c2*. Of course, depending on the parameters of the vehicles, some will have crossed the sensor, and others will not. Again like the previous simulation, the first vehicle will be placed directly in the first node; meanwhile, the rest of the vehicles will be placed using Equation 4.12.

Vehicle Parameters

Since this simulation aims to generate heterogeneous traffic, the parameters of the vehicles will be sampled from the normal distribution explained in Section 4.2.2. An example of what the parameters of these vehicles might be can be seen in Table 4.1.

Visualization

Figure 4.6 shows a visualization of the result of this simulation. The network consists of 25 nodes and 41 edges, which is a more realistic representation of a typical urban block. The sensor is placed at node *c0*. This sensor will record the speed at which a car has passed this point. Again since the parameters of the vehicles are different, a heterogeneous traffic behavior can be observed.

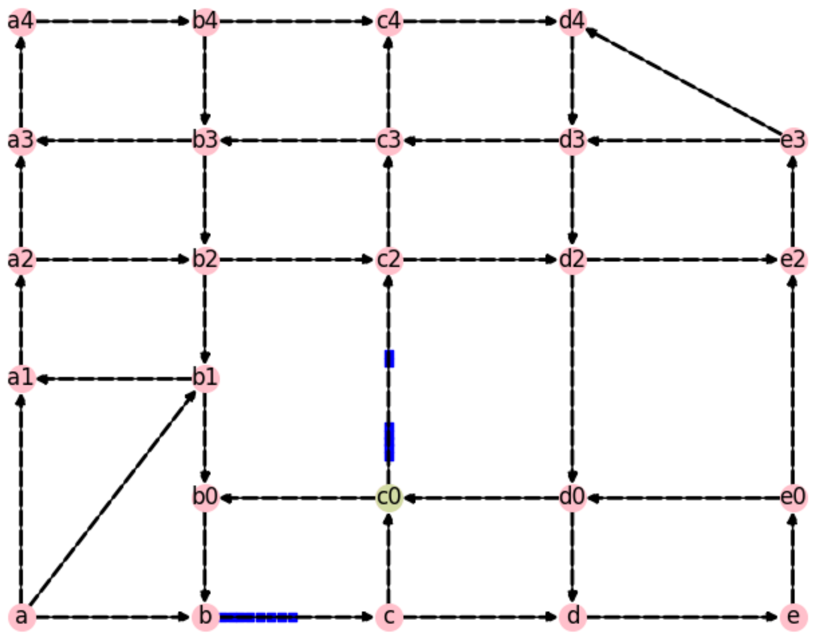


Figure 4.6: Visualization of the third simulation.

4.3.4 Fourth Simulation

The fourth simulation has similar objectives to the previous experiment, but this time we generated a random graph. The purpose of this simulation is to evaluate if the implementation of our model behaves adequately in a random graph. Even though a random graph is not a faithful representation of a road network, this allows us to generate a much more complex network; which is similar to the complexity of an entire city network. In other words, a network with many more nodes (intersections), and roads (edges).

The simulation will be run for 15 minutes where every 90 seconds 10 vehicles will join the network. Its starting point is the node s, and its target node is the node f (see Figure 4.7). Once the vehicles reach node m, the vehicle speed and space headway will be recorded. This procedure will be repeated 200 times to create different samples of speeds and space headways. If for each simulation 100 vehicles were simulated and we repeated this process 200 times this gives us a total of 20,000 vehicles generated that joined the network. Again, the vehicle parameters will be sampled from a normal distribution.

Visualization

The random graph was generated by using the Erdős-Rényi model. Exactly the $G(n, p)$ model where n is the number of nodes in the graph and p is the probability of edge creation. For this case, the number of nodes was set to 35, and the probability of edge creation to 0.1. Also, a custom seed was set to produce the graph in Figure 4.7.

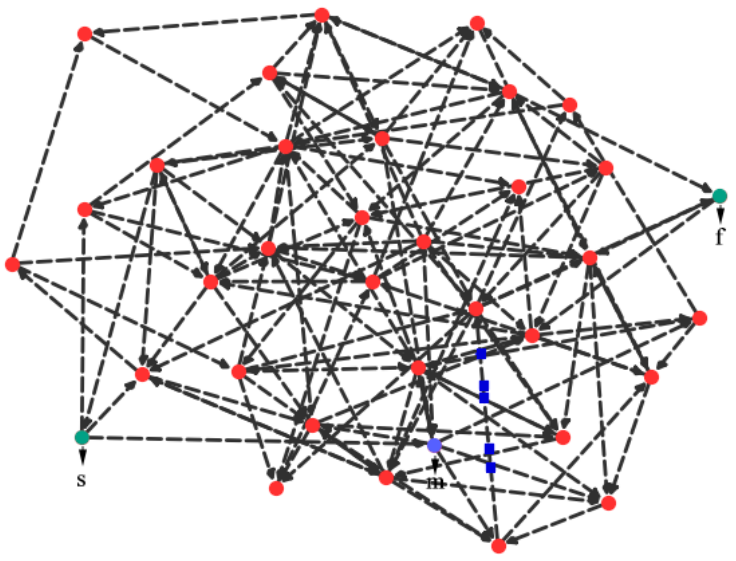


Figure 4.7: Visualization of five vehicles moving towards the node f in a random generated graph.

Chapter 5

Results and Discussion

This chapter presents the results of the experiments described in the previous section. There were four experiments/simulations, and this chapter details the results of each one of those. Each experiment was designed with the purpose of showing various aspects of the model.

We present the data generated by the different experiments proposed in Section 4.3. Our data helped us to create different microscopic and macroscopic relationships as the result of our simulation experiments. We have used visualization techniques to verify that the implementation of the model is yielding sound results. Our implementation can replicate the results proposed by Gipps and other authors, including traffic conditions such as homogeneity, and heterogeneity in the traffic composition. First, we will show the results that support that our implementation of the model can replicate the behaviors mentioned by Gipps. Subsequently, we will evaluate the capability of the model to replicate heterogeneous traffic.

5.1 Evaluation of Gipps' Model Behavior

First, we will be reviewing if our implementation is capable of replicating the behavior of Gipps' car-following model. For this purpose, we recreated homogeneous and heterogeneous traffic conditions. Experiments one and two were the ones that gave us the results for this evaluation.

From the first simulation, we recreated homogeneous traffic. This behavior was achieved in the simulation by generating the same parameters for each vehicle. The purpose of replicating this behavior was to check if the vehicles do not collide and how their speeds are affected by the leading vehicle. Figure 5.1 describes the position of each vehicle at each time step. As shown in the graphic, the trajectory of each vehicle is quite similar, but more importantly, none of these trajectories overlap each other, indicating that none of the vehicles have collided. Furthermore, we can see that a safe distance between each vehicle is maintained over time; which means that each vehicle adjusts its speed to not intrude on the effective size of its leader.

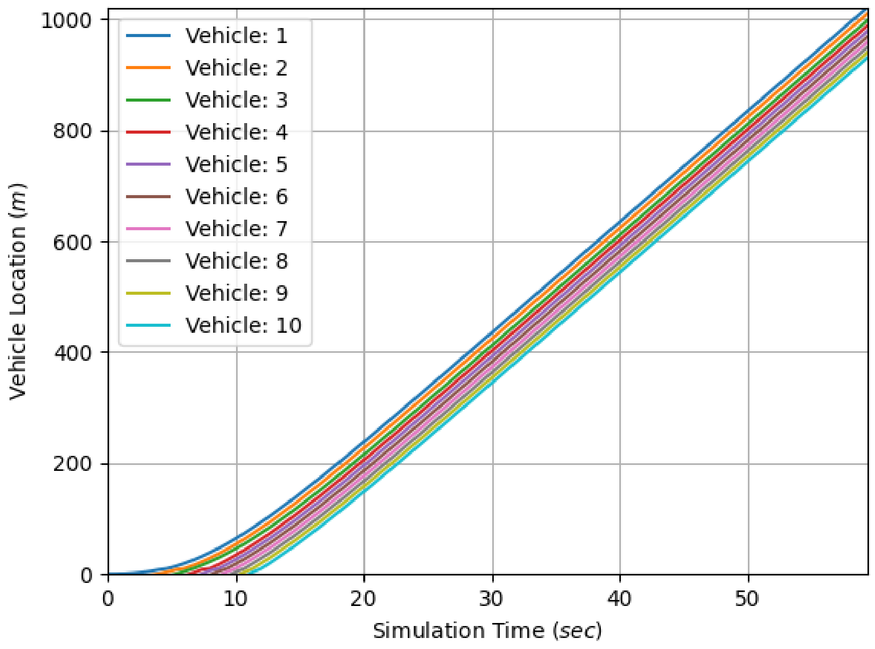


Figure 5.1: Vehicle Position vs. Simulation Time. Data obtained from the second unidimensional simulation that generates homogeneous traffic.

We can refer to Figure 5.2 to confirm the last statement. At the start of the simulation, since vehicles are trying to reach their desired speed (which is the same for each vehicle), their speed overlaps. However, as the simulation time continues, each vehicle (except the first vehicle), begins to slow down to distance itself from the leading vehicle. Meanwhile, the speed of the first vehicle starts increasing until it reaches its desired speed $20m/s$. This behavior is only visible in the first vehicle since this vehicle is not constrained by a vehicle in front. These behaviors of collision avoidance, and the willingness of the vehicle to obtain its desired speed, are one of the main characteristics of Gipps' car-following model.

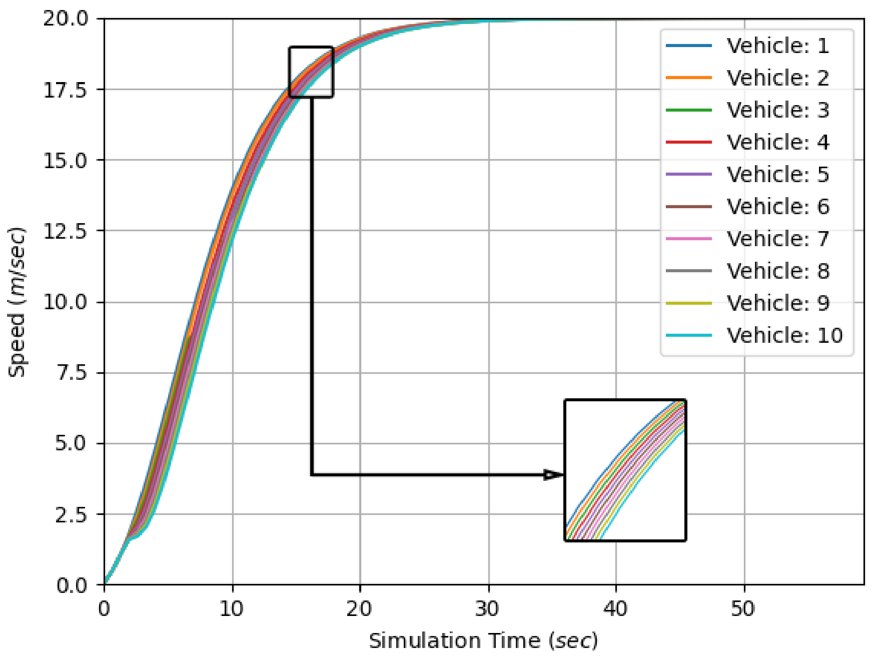


Figure 5.2: Vehicle Speed vs. Simulation Time. Data obtained from the second unidimensional simulation that generates homogeneous traffic.

We just reviewed how the model behaves if the traffic is homogeneous. But, what if vehicle parameters are "generated randomly" (see Section 4.3) to generate heterogeneous traffic. The parameters obtained for this simulation are described in Table 4.1. First, let us analyze the trajectory of the vehicles. Figure 5.3 shows us how the first vehicle is not constrained by a vehicle in front. Thus its speed is higher than the second vehicle, and the vehicle quickly completes the 1000 meters of the section. Another interesting phenomenon happens between vehicles 2 and 3. Since the second vehicle has a lower speed than its leader, this vehicle separates from its leader and begins to limit the trajectory of its follower (the third vehicle). Although the third vehicle has a higher desired acceleration, the vehicle has to limit its acceleration to decrease its speed, and not collide with the leader. Again, the behavior of safe distance is replicated, even in heterogeneous traffic. Despite some of the trajectories of these vehicles are close to each other, these trajectories never overlap.

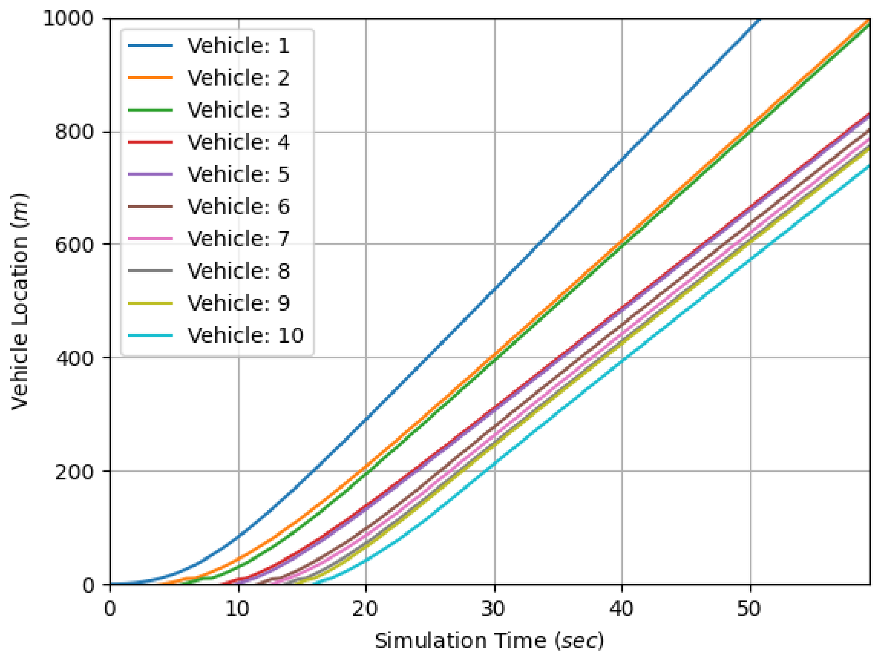


Figure 5.3: Vehicle Location vs. Simulation Time. Data obtained from the second unidimensional simulation that generates heterogeneous traffic.

The behavior mentioned above can also be observed in the analysis of their speeds. This behavior can be seen in Figure 5.4. Again, since the first vehicle is not constrained by a vehicle in front, this vehicle quickly achieves its desired speed. In fact, it only took 25 seconds for the vehicle to do it. This happens because the first vehicle is considered to be in a free-flow state, which is also another of the main characteristics of the Gipps' model. Consequently, Equation 4.2 is used by the vehicle to update its speed at each time step. Both behaviors, safe distance, and free-flow are replicated between vehicle two and three. We can observe how vehicle 3 adjusts its speed when approaching vehicle two (safe distance). However, once his leader begins to increase his speed and therefore to separate from him, the speed of vehicle three also starts to increase again, until it reaches its desired speed (free-flow). An example of how a single vehicle can create traffic congestion can be seen in vehicle fourth. The desired speed of this vehicle is the lowest among the rest of the vehicles. Thus, despite the rest of the vehicles try to reach their desired speed, at some point, they start to reduce their speed since each vehicle begins to be constrained by its leader. Therefore, the desired speed of the driver can not be reached by these vehicles.

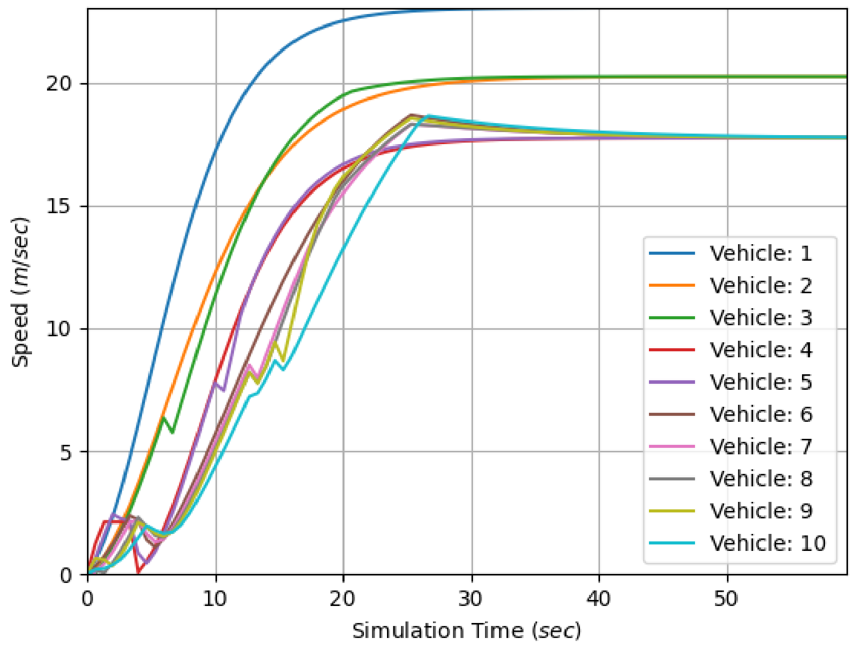


Figure 5.4: Vehicle Speed vs. Simulation Time. Data obtained from the second unidimensional simulation that generates heterogeneous traffic.

5.2 Evaluation of Gipps' model to replicate heterogeneous traffic

We just reviewed how our implementation of the model behaves just like the original Gipps' model. Additionally, we decided to evaluate the capabilities of the model to replicate heterogeneous traffic conditions. Since including different drivers and vehicles is important to model the effects of active traffic. Experiments three and four were the ones that provide us the results for this evaluation.

From the third simulation, we can obtain different microscopic values, such as the speed and location of each vehicle at a certain point in time. In this case, this point is node *c2* (see Section 4.3.3). From these values, we can build macroscopic variables such as the flow and mean speed. As shown in Figure 5.5, the average speed of the vehicles that have crossed *c2* is between $15.50m/s$ and $17.50m/s$. Also, we can see how many vehicles per hour have crossed the sensor at a certain speed. Although the mean of the normal distribution for the desired speed was $20m/s$. Most of the vehicles have crossed the sensor at a speed between $16.25m/s$ and $16.50m/s$. This behavior is similar to what Gipps presented in his original paper, where very few vehicles were able to reach their desired speed, and a cloud of points is generated below the speed limit. This behavior is due to two factors. First, the desired acceleration of the vehicles was not sufficient to reach the desired speed of the driver at the time they crossed the sensor. Second, vehicles were constrained by the speed of their leader. Therefore, these vehicles were not able to reach their desired speed.

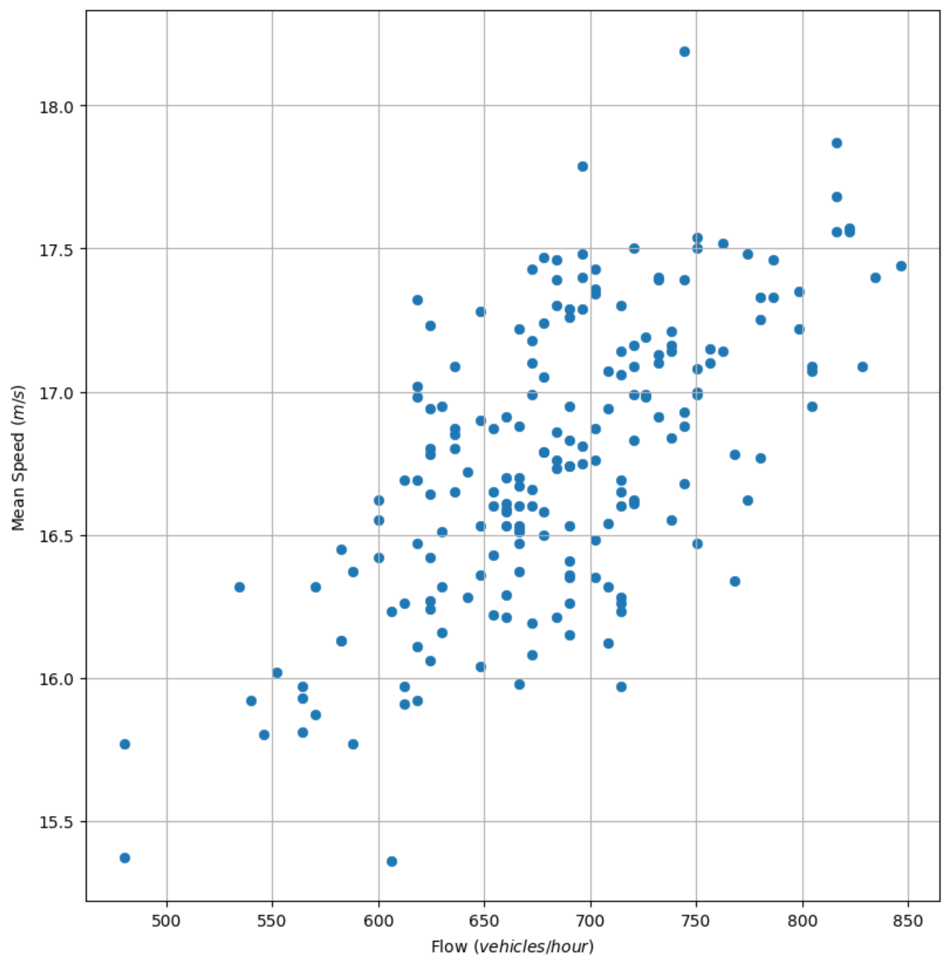


Figure 5.5: Relationship between Mean Speed and Flow. Data obtained from the third simulation. Each point represents the number of vehicles and the average speed at which the vehicles of a single sample crossed the sensor.

From simulation fourth, we obtained different mean speeds and average space headways from the vehicles that managed to cross the sensor. Figure 5.6 show us which is the average space between a vehicle and its leader at a certain speed. For example, we can notice a conglomeration of points between the average space of 30 meters and a mean speed of $10m/s$. This conglomeration of points is because the vehicles to maintain a safe distance from each other reduce their speed. Meanwhile, if the space between the vehicle and its leader increases, the speed of the follower vehicle also increases until it reaches its desired speed. Thus, we can say that the relationship between the mean speed and the average space headway is linear, which is quite similar to the behavior that happens in real traffic.

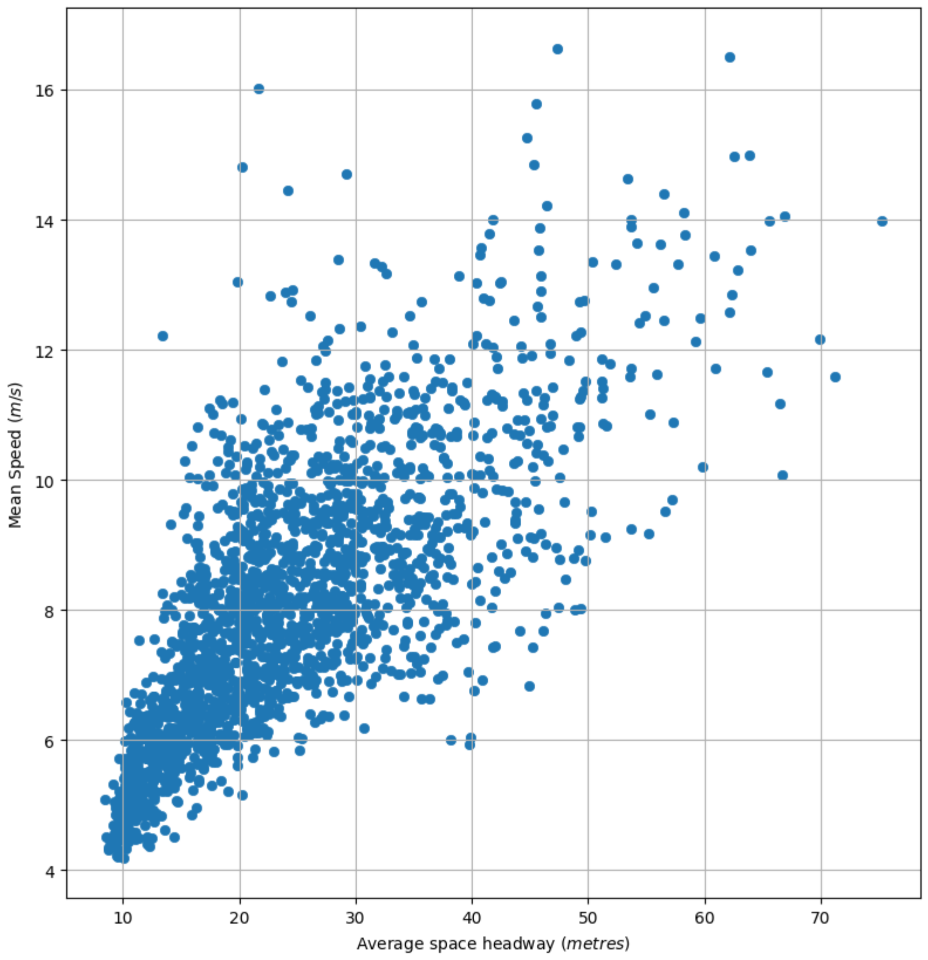


Figure 5.6: Relationship between Mean Speed and Average Space Headway. Data obtained from the fourth simulation. Each point represents the average space headway and the average speed at which the vehicles of a single sample crossed the sensor.

It is essential to understand that the Scatter Plots 5.5 and 5.6 are not fundamental diagrams. Because, as mentioned in Chapter 2, fundamental diagrams represent one-dimensional equilibrium curves. Instead, what these scatter plots are trying to represent is the traffic behavior of Gipps' model and how similar these relationships can be compared to real traffic behavior. As we saw, the traffic generated is not homogeneous, nor is it stationary. Therefore, the distribution of the cloud-like data points of the diagrams is the result of various types of behaviors. First, there is heterogeneity in the traffic composition. Secondly, the non-stationary behavior of traffic introduces a significant amount of scattering. These behaviors are quite similar to those produced in real traffic, thus Gipps' model and our implementation can mimic the behavior of certain traffic conditions.

5.2.1 Hardware and Programming Language

All these experiments were run on the following hardware:

- Ryzen 7 2700X Processor.
- Processor Speed: 3.7GHz.
- RAM: 16GB.
- OS: Ubuntu 20.04 LTS (Focal Fossa).

Our implementation of the model was done using the Python programming language. Simulators are traditionally programmed in compiled languages due to the speed of execution offered by these languages. In contrast, Python is an interpreted language. We decided to choose Python for several reasons. First of all, the produced code is platform-independent, which allows us to simulate in different operating systems. Second, Python allows us to handle data without explicit management of the typing system. Thanks to tools like NumPy, handling decimal numbers with high precision is simple. Third, Python allows us to structure our program through objects. The object-oriented programming paradigm allowed us to easily manage the creation of different vehicles, streets, and simulations. Overall, we can conclude that the Python programming language never presented any disadvantages during the simulator development. In fact all the experiments mentioned above took no more than 20 minutes to run on the hardware listed.

Chapter 6

Conclusions

6.1 Conclusion

Car-following models are widely used, both in research and in transport design studies. Although dozens of models have been developed so far, the model proposed by Gipps remains one of the most popular models. In this context, this thesis has reviewed, implemented, and tested Gipps' original car-following model in two dimensions. Our implementation of the model was validated by the experiments presented in Chapter 5. The original work of Gipps presented a one dimension version of the model. However, our implementation incorporates the movement of vehicles in two dimensions while maintaining the overall behavior of the vehicles. The experiments showed that our implementation behaves like the original Gipps' model. This behavior includes collision avoidance, a minimum safety distance between the leader and follower vehicles, and the capability of the vehicle to reach the desired speed if it is not constrained by a vehicle in front of it. Although other authors [64, 3, 33] have stated that certain choices of parameters can lead to vehicle collisions, our implementation of the model did not show this behavior. Our work does not show the collision behavior because the parameters we used are the same as those used by Gipps in its original model. We recommend limiting the allowed values of the parameters when using a different choice of parameters.

We evaluated the capability of the model to reproduce real traffic conditions. As other authors have stated, the behavior of the model is mainly controlled by three factors: the distribution of the desired speed (see Figure 5.5), the reaction time of the drivers, and the ratio of mean braking rate that the driver estimates. In this work, we reviewed how the desired speed distribution is enough to create different traffic conditions such as traffic congestion and heterogeneous traffic. This is an important aspect of traffic simulation since real traffic conditions are made of different drivers and vehicles. For these reasons, despite the maturity of the model, Gipps' car-following model remains one of the most widely used microscopic models for simulation of traffic flow.

On the other hand, we have to recall that Gipps' model also has limitations. This model is based on physics laws and includes concepts such as velocities and accelerations. However, this model lacks the ability to include elements such as the human cognitive process of the drivers and more sophisticated physics such as the mechanical dynamics of the vehicles. This is why in recent years, the popularity of data-driven models has increased since these models have considerably more parameters. However, these models require a huge amount of data and computing power, which might be restrictive in certain scenarios. Thus, Gipps' car-following and other microscopic models are still reliable and needed to understand traffic operation problems.

Finally, our implementation of the model can be used to build a more complex simulator, as mentioned in Chapter 2 and Chapter 3. The next step is to build a lane-change model from this car-following model. Although Gipps already proposed a lane-change model for its car-following model, new approaches could be built. Also, our implementation of the model can be freely used as the repository is available online. More information on the repository of the source code can be found in Appendix 1. The repository also includes additional information on the different experiments evaluated in this work.

6.2 Recommendations

In this section, we list some recommendations based on the problems and limitations found in our implementation of the model.

- The calibration of the parameters is important if the user wants to replicate more accurately traffic conditions for the area to be simulated. The most used approach is the calibration of the parameters making use of real traffic data. For example, to calibrate the driver desired speed parameters. We can obtain the speed distribution of a certain point or section by using a speed sensor, then this new speed distribution can be implemented in the simulator. Trajectory data also can be used to calibrate other parameters. We encourage the user to calibrate the model parameters, since as pointed out by other authors; improved performance of the model is only achieved via the calibration process.

- In case that the saturation flow obtained is not the desired one. Other parameters also can be included in the model to modify the saturation flow of the model. The Study [3], proposes the creation of a parameter g which is applied to the desired speed of each vehicle. Thus the saturation flow of the section is modified. Also as pointed out by Wilson [33], to reproduce other traffic mechanisms, such as flow breakdown and spontaneous traffic jam formation some restrictions should be added to the model.

- As mentioned in Chapter 4. Our implementation of the model does not incorporate stop lines or traffic signals. This behavior produces a small amount of error when vehicles turn to the next node. The addition of a ”phantom” vehicle is one of the existing alternatives. The way the ”phantom” vehicle is operated is that when the signal indication turns red; the vehicle is located in such a way that the first vehicle to approach the signal begins to follow this ”phantom” vehicle [3]. This ”phantom” vehicle should have special characteristics so that the vehicle that follows it can stop in time.

- We recommend the addition of a graphical user interface (GUI). At the moment, the only way to change the simulation parameters is by modifying the source code. A

GUI would facilitate the user to change the simulation parameters simply, without the need to know the programming logic behind the simulator.

- Implement an adequate visualization tool for the simulation. At the moment the visualization of the simulation is done by using Matplotlib. However, this library is not suitable for a large-scale simulation. Instead, libraries such as Pygame, Vpython, or PyOpenGL are libraries that can be used for the visualization of large simulations in real-time.
- Implement a better format for the output data. We recommend developing a better format output, such as XML, JSON, or CSV. These types of files are more human-readable. Also, some visualization or simulation software can receive these types of files as input. This would allow the integration of other tools into the simulator.
- We recommend creating more complex networks making use of OSMnx Python package [65]. This package allows the user to retrieve and model street networks from OpenStreetMap. We recommend this package because the result street network can be saved as a GraphML file. Thus, we can work this file with NetworkX, and represent the street network as a NetworkX graph.
- Evaluate the performance of other numerical methods to calculate the position of the vehicles. In our implementation of the model, we used the trapezoidal numerical method. However, several methods can be used to do this. Higher-order methods such as Runge-Kutta or Butchers can be used to produce a higher level of accuracy. Studies like [66], already made a comparison between some numerical methods in different car-following models. Where it is concluded that depending on the model and the numerical method, better accuracy can be obtained in the simulation results.

6.3 Future Work

In this section, we list some research that may be the subject of new studies to improve the implementation of Gipps' car-following model in a simulator.

- **Addition of a Lane-Change Model:** As mentioned in Chapter 2 and 3; complex traffic simulators are the combination of car-following models, lane-change models and gap acceptance. This work presented a car-following model capable of being used in conjunction with Gipps' lane-change model or other lane-change models. Simulating any nontrivial traffic situation requires describing not only acceleration and braking but also lane changes [11]. The lane-change framework allows vehicles to pass slower vehicles and avoid obstacles. Therefore, a more realistic description of heterogeneous traffic is achieved.
- **Addition of Human Factors:** One of the disadvantages of car-following models is the exclusion of human factors. The Study [34], presented a framework to introduce these human factors to car-following models through a Task-Capability Interface (TCI), which explains the motivations behind the decision making of the driver. The study showed how adding these human factors to a car-following model improved its performance.

- **Calibration of Parameters to Reproduce Behavior of Heavy Vehicles:** As mentioned in Chapter 3. Gipps' car-following model can reproduce other types of vehicles by calibrating its parameters. For example, heavy vehicles' reactions to traffic are different from those of cars. For this reason, it is important to calibrate the model parameters for each type of vehicle. Therefore, we can reproduce the behavior of various types of vehicles; creating new heterogeneous traffic conditions.

- **Parallelization of Gipps' Car-Following Model:** Supercomputers are capable of processing a large amount of data in less time than traditional computers. However, to achieve this, our model must be able to run in parallel. Microsimulators such as PARAMICS or the IBM Mega Traffic Simulator [4], already implemented traffic simulation in parallel. However, Gipps' car-following model has not been yet implemented in a parallel simulator. Even the literature that analyzes the parallelization of car-following models is limited. We have to remark that the parallelization of car-following models is important if we want to produce large-scale simulations (e.g., a country) in a relatively short time.

- **Gipps' Car-Following Model with Lateral Discomfort:** Car-following models, assume that vehicles travel in the middle of the lane. However, in real-world traffic conditions is not difficult to notice that not every vehicle is positioned in the center of the lane. The Study [67], proposed a car-following model with the incorporation of lateral discomfort. The model presented in this study could be added to existing car-following models. Therefore, a more complex traffic composition could be simulated.

- **Modifications of the Original Model:** As mentioned in Chapter 3. Several simulators make a modified model implementation to produce more accurate results. For example, in AIMSUN simulator [53], one of these modifications is the inclusion of local parameters. Thus, the desired speed of the driver may vary depending on the speed limit of the section. Also in the simulator, the maximum deceleration rate is considered as a local parameter. Three different versions of the parameter are implemented in its model; where the different versions refer to different driving strategies for the section. A similar approach is made in the DRACULA simulator [54]. In its modification of the model, the inclusion of different alert states is proposed. Depending on the alert state, different values for the parameters are used. These mentioned modifications can be implemented or evaluated in our implementation of the model to obtain more accurate results.

Bibliography

[1] E. Winsberg, "Computer Simulations in Science," in *The Stanford Encyclopedia of Philosophy*, winter 2019 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2019.

[2] L. Elefteriadou, *An Introduction to Traffic Flow Theory.* Springer Science+Business Media New York, 2014, vol. 84.

[3] I. Spyropoulou, "Simulation using gipps' car-following model—an in-depth analysis," *Transportmetrica*, vol. 3, pp. 231–245, 2007.

[4] T. Osogami, T. Imamichi, H. Mizuta, T. Morimura, R. Raymond, T. Suzumura, R. Takahashi, T. Idé, T. Osogami, R. Raymond, and R. Takahashi, "Ibm mega traffic simulator," 2012. [Online]. Available: http://domino.research.ibm.com/library/CyberDig.nsf/papers/7BE8B8BF8CAE073A85257B1900190ACE/$File/paper.pdf

[5] D. Krajzewicz, "Traffic simulation with sumo – simulation of urban mobility," *International Series in Operations Research and Management Science*, vol. 145, pp. 269–293, 2010.

[6] W. Burghout, H. N. Koutsopoulos, and I. Andréasson, "Hybrid mesoscopic–microscopic traffic simulation," *Transportation Research Record*, vol. 1934, no. 1, pp. 218–225, 2005. [Online]. Available: https://doi.org/10.1177/0361198105193400123

[7] V. Papathanasopoulou and C. Antoniou, "Towards data-driven car-following models," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 496–509, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.trc.2015.02.016

[8] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B*, vol. 15, pp. 105–111, 1981.

[9] M. F. Aycin and R. F. Benekohal, "Comparison of car-following models for simulation," *Transportation Research Record*, pp. 116–127, 1999.

[10] J. Barcelo and J. Casas, "Dynamic network simulation with aimsun," *Operations Research/ Computer Science Interfaces Series*, vol. 31, pp. 57–98, 2005.

[11] M. Treiber and A. Kesting, *Traffic Flow Dynamics.* Springer Berlin Heidelberg, 2013. [Online]. Available: http://link.springer.com/10.1007/978-3-642-32460-4

[12] M. Brackstone and M. McDonald, "Car-following: A historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, pp. 181–196, 1999.

[13] B. Greenshields, J. Bibbins, W. Channing, and H. Miller, "A study of traffic capacity," *Highway Research Board proceedings*, vol. 1935, pp. –, 1935. [Online]. Available: http://dx.doi.org/

[14] P. Kachroo and K. M. Özbay, "Traffic flow theory," *Advances in Industrial Control*, pp. 57–87, 2018.

[15] M. Papageorgiou, "Some remarks on macroscopic traffic flow modelling," *Transportation Research Part A: Policy and Practice*, vol. 32, pp. 323–329, 1998.

[16] W. Burghout, "Hybrid mesoscopic – microscopic traffic simulation modelling," Ph.D. dissertation, Department of Infrastructure, Royal Institute of Technology, Stockholm, Sweden, 2004.

[17] V. Kanagaraj, G. Asaithambi, C. N. Kumar, K. K. Srinivasan, and R. Sivanandan, "Evaluation of different vehicle following models under mixed traffic conditions," *Procedia - Social and Behavioral Sciences*, vol. 104, pp. 390–401, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.sbspro.2013.11.132

[18] A. Reuschel, "Fahrzeugbewegungen in der Kolonne," *Österr. Ing.-Arch.*, vol. 4, pp. 193–215, 1950.

[19] L. A. Pipes, "An operational analysis of traffic dynamics," *Journal of Applied Physics*, vol. 24, no. 3, pp. 274–281, 1953. [Online]. Available: https://doi.org/10.1063/1.1721265

[20] D. C. Gazis, R. Herman, and R. W. Rothery, "Nonlinear follow-the-leader models of traffic flow," *Operations Research*, vol. 9, no. 4, pp. 545–567, 1961. [Online]. Available: https://doi.org/10.1287/opre.9.4.545

[21] G. F. Newell, "Nonlinear effects in the dynamics of car following," *Operations Research*, vol. 9, pp. 209–229, 1961.

[22] G. Newell, "A simplified car-following theory: a lower order model," *Transportation Research Part B: Methodological*, vol. 36, no. 3, pp. 195 – 205, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261500000448

[23] I. Soria, L. Elefteriadou, and A. Kondyli, "Assessment of car-following models by driver type and under different traffic, weather conditions using data from an instrumented vehicle," *Simulation Modelling Practice and Theory*, vol. 40, pp. 208–220, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.simpat.2013.10.002

[24] H. Rakha and B. Crowther, "Comparison of greenshields, pipes, and van aerde car-following and traffic stream models," *Transportation Research Record*, vol. 1802, no. 1, pp. 248–262, 2002. [Online]. Available: https://doi.org/10.3141/1802-28

[25] D. C. Gazis, *An Introduction to Traffic Flow Theory.* Springer US, 2002, vol. 50.

[26] A. May, *Traffic Flow Fundamentals.* Prentice-Hall, 1990.

[27] R. Liu and X. Li, "Stability analysis of a multi-phase car-following model," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 11, pp. 2660 – 2671, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437113001386

[28] P. RANJITKAR, T. NAKATSUJI, and A. KAWAMUA, "Car-following models: an experiment based benchmarking," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 6, pp. 1582–1596, 2005.

[29] B. Ciuffo, V. Punzo, and M. Montanino, "Thirty years of gipps' car-following model," *Transportation Research Record*, pp. 89–99, 2012.

[30] R. Benekohal and J. Treiterer, "Carsim: Car-following model for simulation of traffic in normal and stop-and-go conditions," *Transportation research record*, vol. 1194, pp. 99–111, 1988. [Online]. Available: http://dx.doi.org/

[31] P. B. Hidas, "SITRAS: A Simulation Model For Its Applications," 1998.

[32] R. Liu, *Traffic simulation with DRACULA*, 06 2011, pp. 295–322.

[33] R. E. Wilson, "An analysis of gipps' model of highway traffic," *IMA J. Appl. Math.*, vol. 66, pp. 509–537, 2001.

[34] M. Saifuzzaman, Z. Zheng, M. Mazharul Haque, and S. Washington, "Revisiting the task–capability interface model for incorporating human factors into car-following models," *Transportation Research Part B: Methodological*, vol. 82, pp. 1 – 19, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261515002052

[35] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007. [Online]. Available: https://doi.org/10.3141/1999-10

[36] P. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B: Methodological*, vol. 20, no. 5, pp. 403 – 414, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261586900123

[37] D. J. Sun and L. Elefteriadou, "Research and implementation of lane-changing model based on driver behavior," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2161, pp. 1–10, 12 2010.

[38] A. Skabardonis, "Simulation of freeway weaving areas," *Transportation Research Record*, vol. 1802, no. 1, pp. 115–124, 2002. [Online]. Available: https://doi.org/10.3141/1802-14

[39] M. Rahman, M. Chowdhury, Y. Xie, and Y. He, "Review of microscopic lane-changing models and future research opportunities," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1942–1956, 2013.

[40] S. Moridpour, M. Sarvi, and G. Rose, “Lane changing models: a critical review,” *Transportation Letters*, vol. 2, no. 3, pp. 157–173, 2010. [Online]. Available: https://doi.org/10.3328/TL.2010.02.03.157-173

[41] S. Wolfram, “Cellular automata,” *Los Alamos Science*, vol. 2, pp. 2–21, 1983.

[42] Kai Nagel and Michael Schreckenberg, “A cellular automaton model for freeway traffic,” *J. Phys. I France*, vol. 2, no. 12, pp. 2221–2229, 1992. [Online]. Available: https://doi.org/10.1051/jp1:1992277

[43] S. Wolfram, “Theory and applications of cellular automata,” *Singapore: World Scientific*, vol. 1, pp. 485–557, 1986.

[44] G. H. Bham and R. F. Benekohal, “A high fidelity traffic simulation model based on cellular automata and car-following concepts,” *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 1, pp. 1 – 32, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X03000573

[45] S. Maerivoet and B. D. Moor, “Traffic flow theory,” *arXiv: Physics and Society*, 2008.

[46] V. L. Knoop, *Introduction to Traffic Flow Theory*, 2nd ed. Delft University of Technology, 2018. [Online]. Available: https://www.victorknoop.eu/research/book/Knoop_Intro_traffic_flow_theory_edition2.pdf

[47] L. C. Edie, *Discussion of traffic stream measurements and definitions.* Port of New York Authority, 1963.

[48] C. Mallikarjuna and K. R. Rao, “Area occupancy characteristics of heterogeneous traffic,” *Transportmetrica*, vol. 2, no. 3, pp. 223–236, 2006. [Online]. Available: https://doi.org/10.1080/18128600608685661

[49] Trasportation Research Board, National Academies of Science, “Highway capacity manual 2010,” 2010.

[50] S. M. Turner, W. L. Eisele, R. J. Benz, and D. J. Holdener, “Travel time data collection handbook,” United States. Federal Highway Administration, Tech. Rep., 1998.

[51] J. Cattin, L. Leclercq, F. Pereyron, and N.-E. El Faouzi, “Calibration of gipps’ car-following model for trucks and the impacts on fuel consumption estimation,” *IET Intelligent Transport Systems*, vol. 13, no. 2, pp. 367–375, 2019. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2018.5303

[52] J. Barcelo, J. Ferrer, and L. Montero, “Aimsun: Advanced interactive microscopic simulator for urban networks,” *Vol I: System Description, and*, vol. 2, 01 1989.

[53] J. Casas, J. Ferrer, D. García, J. Perarnau, and A. Torday, *Traffic Simulation with Aimsun*, 06 2011, pp. 173–232.

[54] R. Liu, *The DRACULA Dynamic Network Microsimulation Model.* Boston, MA: Springer US, 2005, pp. 23–56. [Online]. Available: https://doi.org/10.1007/0-387-24109-4_2

[55] L. Vasconcelos, L. Neto, S. Santos, A. B. Silva, and Álvaro Seco, "Calibration of the gipps car-following model using trajectory data," *Transportation Research Procedia*, vol. 3, pp. 952–961, 2014, 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146514002385

[56] K. Aghabayk, M. Sarvi, and W. Young, "Including heavy vehicles in a car-following model: modelling, calibrating and validating," *Journal of Advanced Transportation*, vol. 50, no. 7, pp. 1432–1446, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.1409

[57] M. Sarvi and O. Ejtemai, "Exploring heavy vehicles car-following behaviour," in *Proc. of the 34th Autralasian Transport Research Forum (ATRF), Adelaide, South Australia, Australia*, 2011, pp. 1–11.

[58] Z. Mo, X. Di, and R. Shi, "A physics-informed deep learning paradigm for car-following models," 12 2020.

[59] Z. He, L. Zheng, and W. Guan, "A simple nonparametric car-following model driven by field data," *Transportation Research Part B: Methodological*, vol. 80, pp. 185–201, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261515001575

[60] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, ser. SEFAIS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 35–38. [Online]. Available: https://doi.org/10.1145/3194085.3194087

[61] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 348–368, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X1830055X

[62] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[63] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[64] L. Lücken, "Resolving collisions for the gipps car-following model," 02 2019.

[65] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198971516303970

[66] M. Treiber and V. Kanagaraj, "Comparing numerical integration schemes for time-continuous car-following models," *Physica A: Statistical Mechanics and its Applications*, vol. 419, 03 2014.

[67] B. Gunay, "Car following theory with lateral discomfort," *Transportation Research Part B: Methodological*, vol. 41, no. 7, pp. 722–735, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261507000161

Appendices

Appendix 1. Code Repository and Instructions

Free Software for a Free Society. I firmly believe in free knowledge and in the importance of sharing our work with the rest of the world freely. For this reason, our implementation of the model is on a GitHub repository in the next URL: `https://github.com/moonR2/GippsTrafficSimulation`.

Installation Guide

This section will guide the interested person to create a new Python virtual environment and install the requirements to be able to run the simulator. The first step is to clone the GitHub repository to our desktop. We can do this in two different ways.

The first way is by downloading the source code directly from the repository page; the downloaded zip file must be unzipped. The second method is by cloning the repository using the command line, the repository can be cloned using the following command:

```
git clone https://github.com/moonR2/GippsTrafficSimulation.git
```

Now that the repository is on our desktop, we need to install the necessary dependencies. This is the list of dependencies we need to install:

- NumPy
- Matplotlib
- NetworkX

There are different ways to install these dependencies. We recommend creating a Python virtual environment, to create a new virtual environment we can use the following code:

```
python3 -m venv Gipps
```

This code will create a new virtual environment called Gipps. Now to use this environment we need to source it:

```
source /Gipps/bin/active
```

Finally, we can install the dependencies using PyPi. Remember that to run the following command we must change the directory to the repository folder:

```
pip install -r requirements.txt
```

Run the Experiments

Now that we have everything ready to run the simulation or contribute to the project. We will explain the scheme of the different files, and what are the changes that we must make to run the different experiments explained in Chapter 4.

The files called: first.py, second.py, third.py, and fourth.py; are the files corresponding to each experiment. These are the files that we will run with Python, for example, the following code will run the first experiment:

```
python3 first.py
```

However, before running each experiment a change is necessary for the simulation.py file.

The file simulation.py contains the class called Simulation. This class is in charge of creating the different types of simulations; where each simulation is a method of the class. In the case of the simulation in two dimensions, the parameter path, contains the path of the vehicles. It is important to change this parameter for each experiment; since the nodes passed to the shortest_path() function must exist in the street network. The following list describes the start and end node of each experiment:

- First experiment: The path of the vehicles goes from node **a** to node **b**.
- Second experiment: The path of the vehicles goes from node **a** to node **b**.
- Third experiment: The path of the vehicles goes from node **a** to node **c2**.
- Fourth experiment: The path of the vehicles goes from node **25** to node **14**.

Of course, we can change or randomly generate the nodes to our liking, as long as the nodes exist in the graph. But to reproduce the results described in Chapter 5 we must use the nodes listed above.

The file platoon.py contains a class called Platoon. In this class, there are several lists in charge of saving the different information generated in the simulation. Finally, the folders model and objects contain different classes. The logic of operation of these classes was described in Chapter 4.

Also, a custom simulation can be created easily. The first step is to build the street network, an example of a network was shown in Chapter 4, but it is also recommended to read the NetworkX documentation for creating more-complex graphs. Once the graph is created we need to create a street object with the edges and nodes from the graph. Finally, we need to call the desired simulation by using the methods from the class Simulation; where its parameters are:

- Number of vehicles.
- Street network speed limit.
- Street network graph.
- Street network nodes with their respective coordinates.
- Street class.
- A boolean indicating if you want the vehicle parameters to be generated randomly.
- The reaction time of the drivers.

Once the simulation is run, the vehicle data can be accessed through the platoon class. Matplotlib makes use of this data to be able to visualize the simulation. Any questions or problems can be sent to the issues section of the GitHub repository page.