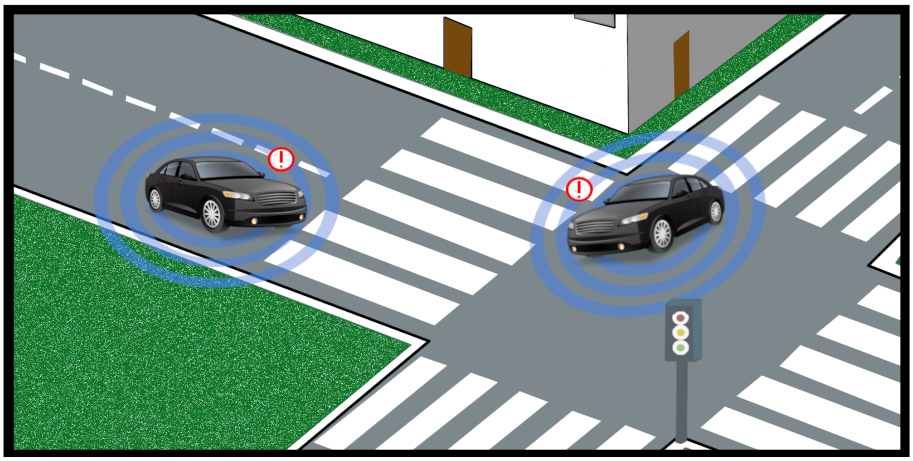
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores



Collision Avoidance Algorithm for Intelligent Vehicles using ITS-G5

José Diogo Salgado dos Reis

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor António João Nunes Serrador
Doutor Carlos Alberto Barreiro Mendes

Júri:

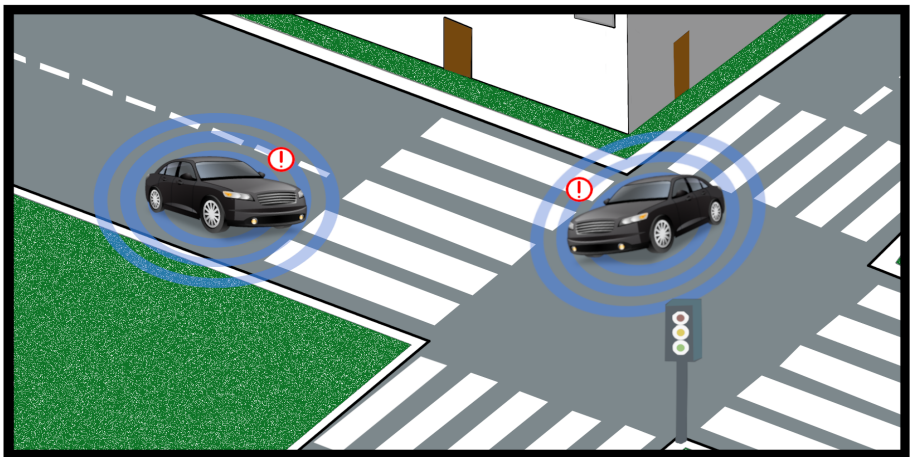
Presidente: Doutor Tiago Miguel Braga da Silva Dias

Vogais: Doutor Nuno Miguel Soares Datia
Doutor António João Nunes Serrador

Setembro, 2022

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores



Collision Avoidance Algorithm for Intelligent Vehicles using ITS-G5

José Diogo Salgado dos Reis

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor António João Nunes Serrador
Doutor Carlos Alberto Barreiro Mendes

Júri:

Presidente: Doutor Tiago Miguel Braga da Silva Dias

Vogais: Doutor Nuno Miguel Soares Datia
Doutor António João Nunes Serrador

Setembro, 2022

Acknowledgments

Initially, I would like to thank Professors António Serrador and Carlos Mendes, for all the support and availability provided throughout the work developed. Next, I would like to thank everyone involved in the C-Streets project, for allowing the development of this work.

Finally, I would like to thank all my family and friends for the support and patience necessary to make it possible to finish all my goals successfully. A very special thanks to my parents and my girlfriend for all the support when it was needed and for always being there when I needed it most. Without them, none of this would be possible.

A special dedication to my grandmother who, wherever she is, will surely be happy to watch me complete another level of this game, called Life.

Abstract

Given the importance of improving road safety, it is crucial to create mechanisms that reduce road accidents. Therefore, this thesis aims the development of a collision warning algorithm capable of supporting the drivers' tasks by alerting them to possible collisions. The algorithm proposed, initially uses ITS-G5 as a communication channel between vehicles. However, given the need to provide the application to most users, a smartphone application is also presented, making this solution accessible to users that don't have access to an ITS-G5 OBU.

In this work, a collision warning algorithm was defined and implemented and, to support the smartphone application, a hybrid environment between ITS-G5 and cellular network was developed. Information is exchanged among players using CAM messages.

Given the role that latency plays in the performance of this kind of application, latency measurements were performed to validate the proposed framework. In general, the best results were obtained when using 4G, obtaining an average of 42 ms in latency. The usage of the 5G network didn't meet the expectations since it promises lower latencies than 4G, but similar latencies and greater instability have been observed at this stage.

When testing the developed application, a digital twin was used to assist the algorithm tests allowing the creation of specific dangerous situations. Therefore, the digital twin was used to test a specific scenario of constant high latency messages creating a false negative situation, where a collision would be detected if latency was lower/admissible.

Keywords: V2X, ITS-G5, Collision Warning, Connected Vehicles, CAM, Hybrid Environment, Cellular Network, OBU, Collision Avoidance Algorithm

Resumo

Dada a importância de melhorar a segurança rodoviária, é extremamente importante a criação de mecanismos que auxiliem a redução do número de acidentes rodoviários. Posto isto, esta tese visa o desenvolvimento de um algoritmo para alerta de colisões veiculares, capaz de apoiar o condutor na sua tarefa. O algoritmo proposta, inicialmente utiliza ITS-G5 como canal de comunicação entre veículos. No entanto, dada a necessidade de disponibilizar a aplicação à maioria dos utilizadores, apresenta-se também uma solução para *smartphone* que permite a utilização da mesma a utilizadores que não tenham acesso a uma OBU ITS-G5.

Neste trabalho, o algoritmo de previsão de colisão foi definido e implementado e, para dar suporte à aplicação móvel, foi desenvolvido um ambiente híbrido entre ITS-G5 e redes celulares. A troca de informação é feita utilizando mensagens CAM.

Dado a importância da latência no desempenho da aplicação, foram realizadas medições de latência para validar a solução proposta. Em geral, os melhores resultados foram obtidos aquando da utilização do 4G, obtendo uma latência média de 42 ms. O uso da rede 5G não alcançou as expectativas, pois esperavam-se menores latências que o 4G, mas apenas se verificaram latências similares e com maior instabilidade.

Ao testar o algoritmo desenvolvido, foi utilizado um Digital Twin para auxiliar os testes à solução permitindo a criação de situações específicas de alto risco. Posto isto, a sua utilização foi útil para testar uma situação constante de mensagens recebidas com latência elevada originando uma situação em que a aplicação assume um cenário distinto da realidade que impede a deteção da colisão, quando a mesma existe (falso negativo).

Palavras-chave: V2X, ITS-G5, Collision Warning, Veículos Conectados, CAM, ITS, Ambiente Híbrido, Redes Celulares, OBU, Collision Avoidance Algorithm

Contents

List of Figures

List of Tables

Acronyms

BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
C-ITS	Cooperative Intelligent Transport System
CPM	Collective Perception Message
DENM	Decentralized Environmental Notification Message
DSRC	Dedicated short-range communications
DTC	Distance to collision
EU	European Union
GN	GeoNetworking
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HF	High Frequency
IoT	Internet of Things
ITS	Intelligent Transport Systems
ITS-G5	Intelligent Transport Systems operating in the 5,9 GHz frequency band
MAPEM	MAP Message
MQTT	Message Queuing Telemetry Transport

NASA National Aeronautics and Space Administration

OBU On-Board Unit

PDU protocol Data Unit
POI Point of Interest

RSU Road-Side Unit

SDK Software Development Kit
SPATEM Signal Phase and Timing Message

TTC Time to collision

V2I Vehicle-to-Infrastructure
V2V Vehicle-to-Vehicle
V2X Vehicle-to-Everything
VANET Vehicular ad hoc networks

WGS84 World Geodetic System

List of Symbols

α_w Steering wheel angle

α_{rt} Angle between the transmitter and the receptor

δ Angle between the lines that connect the transmitter and the receptor to earth's center, in degrees

σ Standard deviation

τ System delay

θ Vehicle's heading

A Safety zone's center on a Cartesian coordinate system

a_{VF} Following vehicle's acceleration

a_{VL} Leading vehicle's acceleration

B, C, D, E Safety zone's edges on a Cartesian coordinate system

d Distance

$d(braking)$ Braking distance

$d(lat)$ Distance traveled during the latency time

$d(vehicle - collision)$ Distance between the vehicle and the predicted collision location

$d(VF)$ Distance travelled by the following vehicle

$d(VF, VL)$ Distance between the following vehicle and the leading vehicle

$d(VL)$ Distance travelled by the leading vehicle

$d\theta$ Difference between vehicles' heading

d_0 Headway offset

d_s Stopping distance

d_{rt} Distance between the transmitter and the receptor

d_w Distance between following and leading vehicles to issue a warning

F Front safety distance

f Coefficient of friction between the tires and the road

G Road inclination

L Back safety distance

$l(V_F)$ Following vehicle's length

$l(V_L)$ Leading vehicle's length

L_e Distance from the zone's center to the zone's rear edge in meters;

L_f Distance from the zone's center to the zone's front edge in meters;

lat Latitude

lat_r Vehicle running the application's latitude

lat_t Neighbour's latitude

lon Longitude

lon_r Vehicle running the application's longitude

lon_t Neighbour's longitude

R Earth's radius

t A period of time

t_{pr} Driver's perception and reaction time

v Speed

v_{rel} Relative speed between the following and leading vehicles

v_{VF} Following vehicle's speed

v_{VL} Leading vehicle's speed

VF Following vehicle

VL Leading vehicle

W_l Distance from the zone's center to the zone's left edge in meters;

W_r Distance from the zone's center to the zone's right edge in meters;

1

Introduction

1.1 Objective

When it comes to human safety, road accidents are one of the most relevant factors to be considered given its impact on the population. According to official information from *Eurostat* [1], in 2019, in the European Union (EU), 22 756 inhabitants died, as victims of road accidents. Although this number is lower than that recorded in 2009 (around 33 000), it is still relatively large, showing the existence of a problem when analysing road accidents. In the portuguese case, 67 deaths per million inhabitants per year were reported, which is higher than that recorded for the EU (51 deaths per million inhabitants per year). Therefore, it is extremely important to use technology improvement to create mechanisms to reduce these numbers as much as possible.

The objective of this thesis is to propose an onboard vehicle safety algorithm to warn drivers of possible collisions with neighbouring vehicles, a collision warning application.

Applications running this kind of security algorithms are very important because they are able to improve the driver's perception of the neighbouring environment and, therefore, predict possible dangers.

The exchange of data between vehicles is done using Intelligent Transport Systems operating in the 5,9 GHz frequency band (ITS-G5) or a hybrid network between this technology and cellular networks developed in the scope of this work. To implement

this hybrid environment, two important technologies were used, ITS-G5 and Message Queuing Telemetry Transport (MQTT), which will be described next.

1.2 ITS-G5

According to Car 2 Car Communication Consortium, Cooperative Intelligent Transport System (C-ITS) "refers to transport systems, where the cooperation between two or more Intelligent Transport Systems (ITS) sub-systems (personal, vehicle, roadside and central) enables and provides an ITS service that offers better quality and an enhanced service level, compared to the same ITS service provided by only one of the ITS sub-systems" [2].

The technology ITS-G5 is the European protocol stack for supporting vehicular communications operating in the 5.9 GHz frequency band [3]. This protocol is used to allow Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Everything (V2X) communication, enabling vehicles to communicate with everything around it, like other vehicles or stationary equipment located on roads (C-ITS stations). Figure 1.1 illustrates the protocol stack for C-ITS stations.

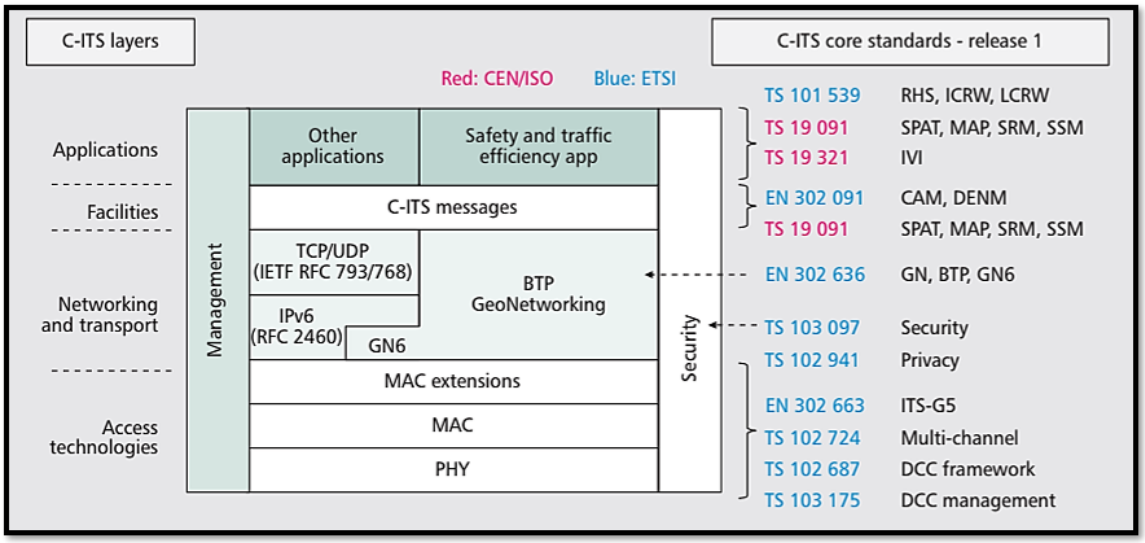


Figure 1.1: Protocol stack for C-ITS stations(extracted from [4]).

The protocol stack is divided into four main layers [4]:

- Access technologies: this layer provides network access to these devices ensuring that packets are delivered to end users. It primarily utilizes the ITS-G5 protocol or cellular networks;

- Networking and transport: this layer uses one of two options: (i) Basic Transport Protocol (BTP) and GeoNetworking (GN) or (ii) Internet protocols. The communication mode's choice lies in the application. Typically, the GeoNetworking is used for ad-hoc communication over ITS-G5 and Internet protocols (IPv6 and TCP/UDP) are used with an IP-based infrastructure over cellular networks. The GN6 sublayer has been designed to allow the transmission of IPv6 packets over GeoNetworking;
- Facilities: in this layer, a set of message types are defined to enable application functionality. These messages will be described later, in this Section;
- Applications: finally, this layer specifies applications running in C-ITS stations, mainly safety and traffic efficiency applications.

GeoNetworking is a protocol that provides packet routing in an ad hoc network, using geographical positions for packet transportation. This protocol makes use of geographical areas based on coordinates and area's shape and size creating a route path using vehicles forwarding packets through its destination [5]. This way, packets contain information about their destiny allowing intermediary nodes to forward or process the packet according to this information.

BTP provides an end-to-end transport service for the ITS ad hoc network and its main purpose is the multiplexing of different messages at the facilities layer [6].

1.2.1 C-ITS Messages

As mentioned before in this Section, in the facilities layer, a set of different message types was defined to support different applications with different purposes. Each type of C-ITS message is used with a different objective and carries different information, which leads to each message having a different structure according to its purpose.

These are some examples of the most relevant message types:

- **Decentralized Environmental Notification Message (DENM):** messages that carry information about specific events occurring on roads, such as road works, adverse weather conditions or accidents [7]. Typically originated in RSU;
- **Cooperative Awareness Message (CAM):** messages that carry vehicles' state information, for example, location, speed and direction [8]. Originated in OBU or RSU, but more commonly in OBU;

- **Point of Interest (POI):** messages that carry information about points of interest, for example, electronic vehicle charging stations [9]. Typically originated in RSU;
- **Signal Phase and Timing Message (SPATEM):** messages that carry information about the state of an intersection to vehicles approaching it and usually contain information such as traffic light state or queue state [10]. Typically originated in RSU;
- **MAP Message (MAPEM):** messages that contain geographic road information such as intersection descriptions, road segments descriptions or segments of roadway [10]. Typically originated in RSU;
- **Collective Perception Message (CPM):** messages that carry information about locally detected objects, such as obstacles, based on station sensors capable of detecting information around the station [11]. Originated in OBU;

The most used messages nowadays are CAM and DENM, however, in the scope of this work, only CAM messages are used by the developed application.

CAM messages are transmitted by connected vehicles and they are exchanged to create and maintain awareness of each neighbouring vehicle and to support cooperative performance applications that use information shared through these messages to support certain functionalities.

A CAM is composed by a header and various containers, some of them optional. Figure 1.2 shows the general structure of a CAM message.

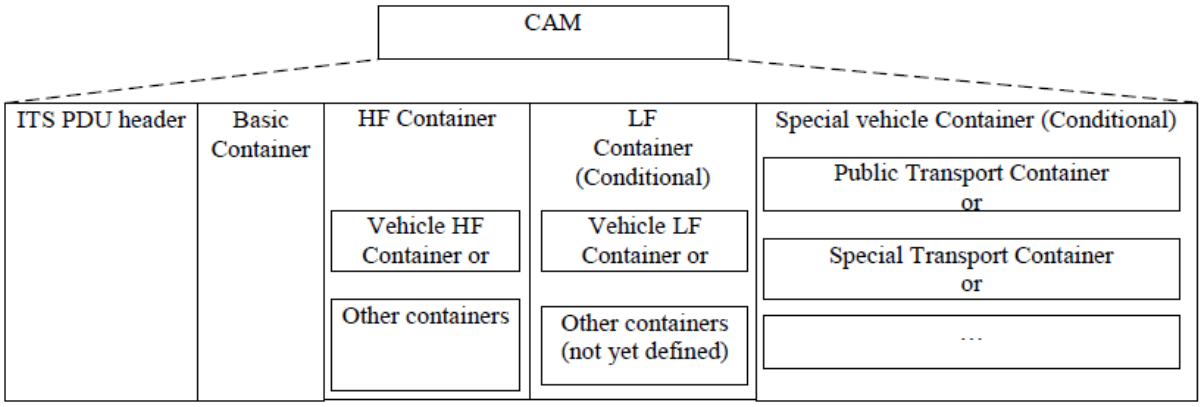


Figure 1.2: CAM message general structure (extracted from [8]).

Given the message structure extension, only the most relevant fields for the present work will be addressed.

First, the ITS protocol Data Unit (PDU) header includes:

- Protocol version;
- Message type: 2 stands for CAM;
- Originating station identifier;

The basic container is a mandatory container that includes basic information about the originating station, containing:

- Station Type, for example, pedestrian, cyclist, passenger car.
- Reference position, containing information about the station's latest geographic position.

At least one High Frequency (HF) Container shall be present in every CAM and it contains all the station's dynamic status information such as speed or heading. Since the information used in this work is contained in the HF Container, only this one will be considered. For this container, there are two options to be used: Basic Vehicle Container HF or Road-Side Unit (RSU) Container HF; being the last used by RSUs and the first by On-Board Unit (OBU) stations. The Basic Vehicle Container HF contains a lot of information about the station, but only the heading and speed fields are used in the scope of this work [8].

The generation rate of CAMs shall be between 1 and 10 Hz.

1.2.2 Equipment

In order to communicate between vehicles and infrastructure, two different kinds of equipment are necessary: OBUs and RSUs. The OBU, as the name implies, is equipment that is present inside vehicles and whose main objective is to disseminate information regarding the vehicle. This information is encapsulated inside a CAM message which is then received by an OBU in other vehicles (V2V) or by an RSU (V2I). Figure 1.3 shows an example of an OBU, from Unex [12].

Figure 1.3: Unex OBU equipment (extracted from [12]).

An RSU is an equipment located along the road with a static location whose main objective is to disseminate DENM messages to share information about events along the road. RSUs can also send CAMs using the RSU Container HF, as mentioned in Section 1.2.1. Figure 1.4 shows an example of RSU, from Siemens [13].

Figure 1.4: Siemens RSU equipment (extracted from [13]).

Although in the future, most vehicles are expected to be equipped with an ITS-G5 OBU right from the factory, currently not every vehicle circulating on public roads has access to an OBU. To overcome this issue, it is proposed the use of mobile cellular networks for these vehicles, using a smartphone to replace an OBU. This is accomplished using MQTT.

1.3 MQTT

MQTT [14] is a lightweight protocol responsible for machine-to-machine message transportation based on a publisher/subscriber model. This protocol was designed to be used on devices with resource constraints or limited bandwidth and its main objective was to lower the bandwidth used on point-to-point communication, granting message

delivery and reliability. All these characteristics make this protocol ideal for Internet of Things (IoT) systems, where devices and sensors are used to share information and low energy consumption is a very important aspect for these devices. Because of that, MQTT becomes a good solution for the IoT system.

The Publisher/Subscriber model is a pattern of message exchange in which clients do not communicate directly with each other, but with an intermediary server that is responsible for forwarding messages between clients. The clients that send messages (Publishers), send them to the intermediary server (Message Broker), specifying one topic. The Message Broker then forwards these messages to the clients that have shown interest in that specific topic (Subscribers). Figure 1.5 demonstrates this process.

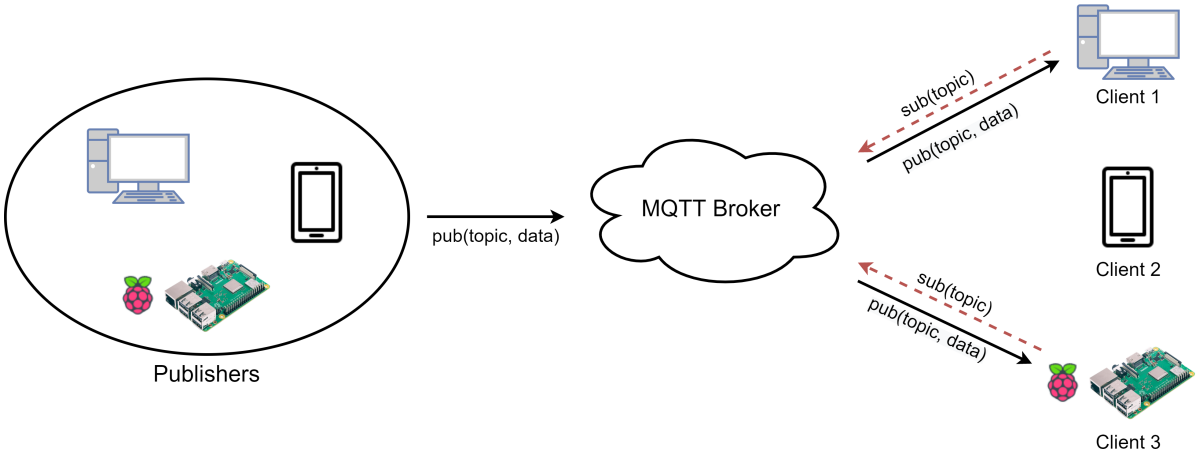


Figure 1.5: MQTT Publisher/Subscriber model.

In Figure 1.5, Clients 1 and 3 initially subscribe to a specific topic and, when Publishers send a message, the Message Broker (MQTT Broker) forwards them to the Subscribers. Since Client 2 didn't subscribe, the message is not sent to him.

1.4 Document Structure

This report is divided into five main chapters, where each describes the work's different phases. Thus, this document is divided as follows:

- The first chapter describes the thesis, including an introduction to protocols used in this work;
- The second chapter describes algorithms and systems already developed and published with the aim of collision avoidance;

- In the third chapter it is described the developed work, including the application and hybrid environment architectures.
- In the fourth chapter, the obtained results are described including the latency measurements for the hybrid environment and the latency impact on the application.
- The final chapter, the main conclusions and future work are presented.

2

State-of-the-Art

2.1 Introduction

This chapter describes algorithms already developed, commercially or under research, with the focus of providing more security for drivers, assisting their driving experience and warning them of possible collision accidents. These algorithms use information disseminated by vehicles, via inter-vehicular communication, to identify and avoid situations of possible collisions with neighbouring vehicles. In the algorithms' description, the vehicle running the algorithm will be mentioned as the main vehicle while the others around it will be the neighbouring vehicles.

2.2 Rear-End Collisions

These type of algorithms uses vehicles' information to analyse the possibility of a collision based on the calculation of a critical distance from which it is necessary to alert the driver to a dangerous situation. These algorithms consider a specific situation where two vehicles are driving in the same lane where the following vehicle is travelling faster than the leading vehicle, representing a dangerous situation.

2.2.1 A Novel Rear-End Collision Warning Algorithm in VANET

The algorithm proposed by Hexin *et al* [15] analyse the possibility of rear-end collisions in Vehicular ad hoc networks (VANET). The algorithm is processed following the steps described in Figure 2.1. First, it is clarified which vehicles are in the same lane as the following vehicle (VF), followed by the determination of the leading vehicle (VL), based on the distance between them. Finally, the collision risk is analysed.

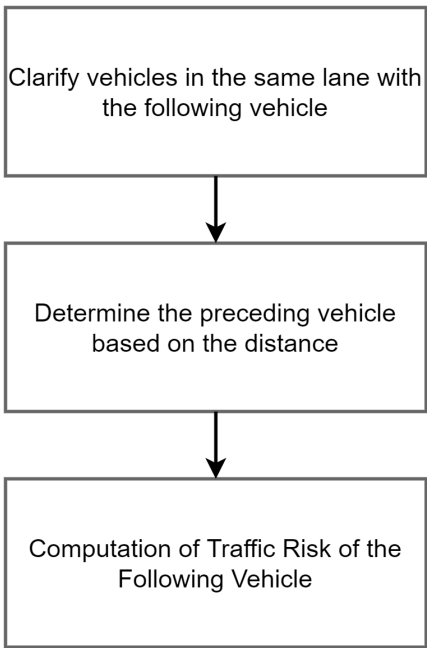


Figure 2.1: Algorithm's working process (adapted from [15]).

The first step is made based on communication with local RSUs which identifies in which lane each vehicle is driving and informs the following vehicle of this. Since this way of analysing is not important for the current work, it won't be further mentioned in this segment. After this, and having the information from the cars that circulate in the same lane as the main vehicle, the algorithm uses the vehicle's Global Navigation Satellite System (GNSS) coordinates and the information received from the other vehicles to calculate the distance between itself and each vehicle individually, concluding which vehicle is in front of it, being referred as VL. Having this information, and the distance between them, the speed and acceleration of both vehicles are then used to understand if, in a certain period t, a collision can occur. Assuming that the speed and acceleration are constant, this analysis is made as shown in Figure 2.2. Firstly, it is predicted the distance to be travelled by each vehicle in t, referred as $d(VF)$ and $d(VL)$ in Figure 2.2 b). Then, the difference between these distances is compared with the current vehicle's distance, referred as $d(VF, VL)$ in Figure 2.2 a).

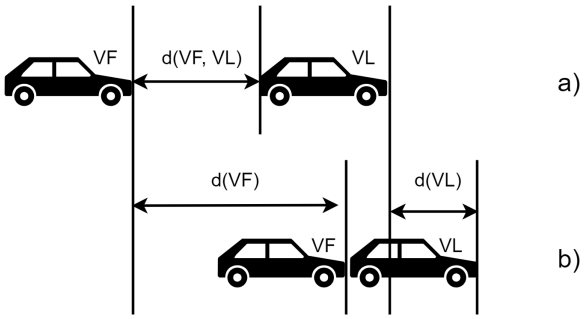


Figure 2.2: Algorithm risk calculation (adapted from [15]).

The mathematical expressions used for evaluating the collision possibility are expressed in equations (2.1) and (2.2). As described, if the current distance between vehicles is lower than the difference between the predicted distance travelled by the following and leading vehicles, then the collision is detected and a warning must be issued.

$$d(VF, VL) < d(VF) - d(VL) \tag{2.1}$$

$$d(VF) - d(VL) = (v_{VF} - v_{VL}) \times t + \frac{1}{2}(a_{VF} - a_{VL}) \times t^2 \tag{2.2}$$

The following information is used in the equations:

- t: a certain period of time, in seconds;
- $d(VF)$: predicted distance travelled by the following vehicle in t, in meters;
- $d(VL)$: predicted distance travelled by the vehicle VL, in meters;
- v_{VF}: VF vehicle's speed, in meters per second (m/s);
- v_{VL}: VL vehicle's speed, in m/s;
- a_{VF}: VF vehicle's acceleration, in m/s^2;
- a_{VL}: VL vehicle's acceleration, in m/s^2;

2.2.2 Research on a DSRC-Based Rear-End Collision Warning Model

The application proposed by Xiang *et al* [16] was created to predict rear-end collisions using Dedicated short-range communications (DSRC). In Figure 2.3 it is shown some important parameters for the used algorithm's functioning.

The following information is important to better understand the figure:

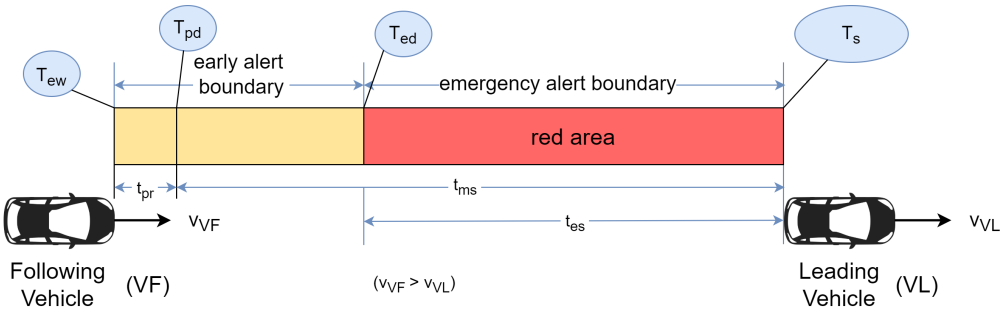


Figure 2.3: Algorithm's critical points (adapted from [16]).

- T_{ew}: time point where an early warning should be issued;
- T_{pd}: time point where the subject vehicle should start decelerating with $a_{moderate}$;
- T_{ed}: time point where an emergency warning should be issued;
- T_s: time point where the vehicle stops it's movement.
- t_{pr}: driver's perception and reaction time, in seconds;
- t_{ms}: vehicle slowing down time under moderate braking, in seconds;
- t_{es}: vehicle slowing down time under emergency braking, in seconds;
- v_{VF} and v_{VL}: VF and VL's speed, respectively, m/s.

The algorithm will function when $v_{VF} < v_{VL}$, meaning that the VF is driving faster than the VL. In this case, an early warning should be issued when the distance between VF and VL is d_w, at T_{ew}.

$$\begin{cases} d_w = (v_{VF} - v_{VL})t_{pr} + d_{ms} - d_{mf} + L \\ = (v_{VF} - v_{VL})t_{pr} + \dfrac{(v_{VF} - v_{VL})^2}{2a_{moderate}} + L \end{cases} \tag{2.3}$$

Where:

- d_{ms}: distance traveled by the VF between T_{pd} and T_s, in meters;
- d_{mf}: distance traveled by the VL between T_{pd} and T_s, in meters;
- L: $(l_{VF} + l_{VS})/2$, in meters;
- l_{VF} and l_{VS}: lengths of VF and VL, respectively, in meters;

- $a_{moderate}$: moderate deceleration, m/s^2.

When the vehicle is in the red area, hard braking is necessary with maximum deceleration. When it is possible, since the human reaction time is higher than the system's processing time, the hard braking is automatically started when an emergency warning is issued at T_{ed}.

2.2.3 Mazda Algorithm

The Mazda algorithm defined in [17] uses equation (2.4) to calculate the critical warning distance d_w between vehicles, in meters, where a warning should be issued.

$$d_w = \frac{1}{2}\left(\frac{v_{VF}^2}{a_{VF}} - \frac{(v_{VF} - v_{rel})^2}{a_{VL}}\right) + v_{VF}\tau + v_{rel}t_{pr} + d_0 \tag{2.4}$$

The equation uses the following information:

- v_{VF}: following vehicle's speed, in m/s;
- v_{rel}: relative speed between the following and leading vehicles, in m/s;
- a_{VF} and a_{VL}: following and leading vehicles' maximum decelerations respectively, in m/s^2;
- τ: system delay, in seconds;
- t_{pr}: driver delay, in seconds;
- d_0: headway offset;

2.2.4 Honda Algorithm

The Honda algorithm presented in [17] uses (2.5) to calculate the critical warning distance, d_w, in meters, where a warning should be issued. Besides this, v_{rel} is the relative velocity between the following and leading vehicles, in m/s.

$$d_w = 2.2v_{rel} + 6.2 \tag{2.5}$$

2.3 Vehicles' Overlapping

This type of algorithm uses a spacial representation of each involved vehicle and determines if a collision will occur based on the vehicles' overlapping. If this is detected, a collision alert is issued.

2.3.1 Collision Warning Algorithm for Passage of an Uncontrolled Road Intersection

The algorithm proposed by Janeks Ahrems [18], defines a safety zone for each involved vehicle to determine if there is a collision possibility based on safety zones overlapping. Figure 2.4 demonstrates the safety zones' definition.

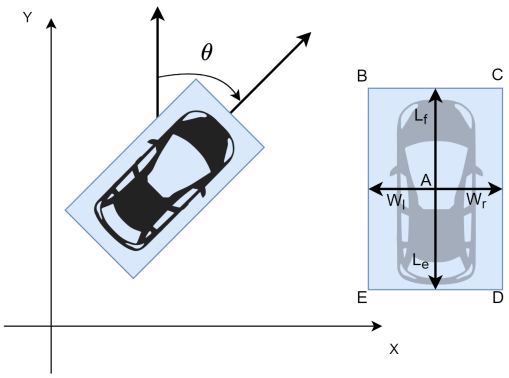


Figure 2.4: Vehicle's safety zone model (extracted from [18]).

The information shown in Figure 2.4 is described next:

- θ: Vehicle's heading;
- A: GNSS coordinates representation;
- B, C, D, E: Safety zone limits;
- L_f: Distance from the central zone to the front zone edge, in meters;
- L_e: Distance from the central zone to the rear zone edge, in meters;
- W_l: Distance from the central zone to the left zone edge, in meters;
- W_r: Distance from the central zone to the right zone edge, in meters;

The safety zone is in the shape of a rectangle to best portray a vehicle and its dimensions are not based only on the vehicle dimensions but a margin is added to guarantee a safe distance around it. The safety zone's corners coordinates are defined using the following equations:

$$\begin{cases} X_B = X_A + L_f \sin(\theta) - W_l \cos(\theta) \\ X_C = X_A + L_f \sin(\theta) + W_r \cos(\theta) \\ X_D = X_A - L_e \sin(\theta) + W_r \cos(\theta) \\ X_E = X_A - L_e \sin(\theta) - W_l \cos(\theta) \end{cases} \tag{2.6}$$

$$\begin{cases} Y_B = Y_A + L_f \cos(\theta) + W_l \sin(\theta) \\ Y_C = Y_A + L_f \cos(\theta) - W_l \sin(\theta) \\ Y_D = Y_A - L_e \cos(\theta) - W_r \sin(\theta) \\ Y_E = Y_A - L_e \cos(\theta) + W_r \sin(\theta) \end{cases} \tag{2.7}$$

This model is applied to every vehicle involved and the collision analysis is made considering the safety zones' overlapping following the condition at Equation 2.8.

$$S_X(t) \leqslant 0 \quad \textbf{OR} \quad S_Y(t) \leqslant 0 \tag{2.8}$$

In case of collision, the condition is verified. Equations (2.9), (2.10) and (2.11) demonstrate the needed calculations to fulfil the condition.

$$\begin{cases} S_X = L_X - (L1_X + L2_X) \\ S_Y = L_Y - (L1_Y + L2_Y) \end{cases} \tag{2.9}$$

$$\begin{cases} L_X = [MAX(X) - MIN(X)] \\ L_Y = [MAX(Y) - MIN(Y)] \end{cases} \tag{2.10}$$

Where:

- $X = \{X_{B1}, X_{C1}, X_{D1}, X_{E1}, X_{B2}, X_{C2}, X_{D2}, X_{E2}\}$;
- $Y = \{Y_{B1}, Y_{C1}, Y_{D1}, Y_{E1}, Y_{B2}, Y_{C2}, Y_{D2}, Y_{E2}\}$.

This way, it is used the maximum and minimum values of coordinates for X and Y, for both vehicles.

$$\begin{cases} L1_X = MAX(X1) - MIN(X1) \\ L1_Y = MAX(Y1) - MIN(Y1) \\ L2_Y = MAX(X2) - MIN(X2) \\ L2_Y = MAX(Y2) - MIN(Y2) \end{cases} \tag{2.11}$$

Where:

- $X1 = \{X_{B1}, X_{C1}, X_{D1}, X_{E1}\}$ and $X2 = \{X_{B2}, X_{C2}, X_{D2}, X_{E2}\}$;
- $Y1 = \{Y_{B1}, Y_{C1}, Y_{D1}, Y_{E1}\}$ and $Y2 = \{Y_{B2}, Y_{C2}, Y_{D2}, Y_{E2}\}$.

This way, it is used the maximum and minimum points of the edges of each vehicle to determine if safety zones are overlapping and, therefore, a risk of collision. Figure 2.5 shows how these equations can be represented in the axis X. The same projection is made for axis Y. In case of collision, which is not the case in the example, $L_X < L1_X + L2_X$ is verified.

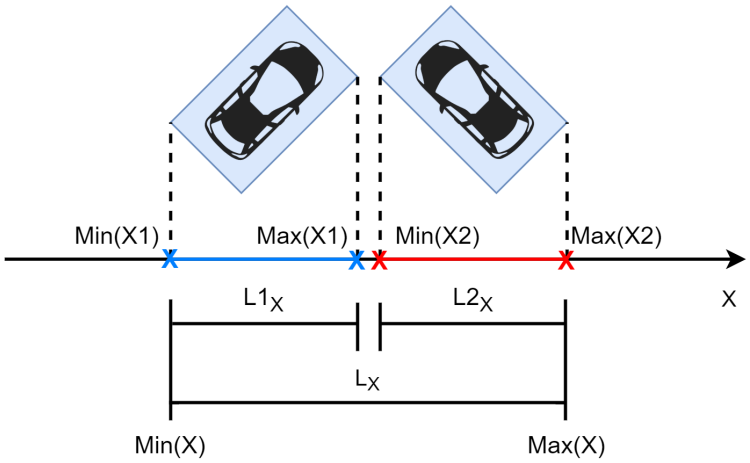


Figure 2.5: Example of projection to the axis X (adapted from [18]).

To use this algorithm, it is needed the vehicles' prediction of their future positions and for each new position predicted, the calculations described before are applied. To this effect, Janeks Ahrems [18] uses speed and acceleration information to predict the vehicle's movement, as shown in equation (2.12).

$$\begin{cases} X_1 = X_0 + vt + at^2/2 \\ Y_1 = Y_0 + vt + at^2/2 \end{cases} \tag{2.12}$$

In these equations, t is the calculation period, X_1 is the new X coordinate and X_0 is the vehicle's last X coordinate. The same terminology is used for the Y axis.

2.3.2 Cooperative Collision Warning Systems: Concept Definition and Experimental Implementation

The algorithm proposed by Raja Sengupta *et al* [19], was created with the purpose of analysing specific traffic scenarios. In Figure 2.6, it is shown the algorithm's system architecture and how it works.

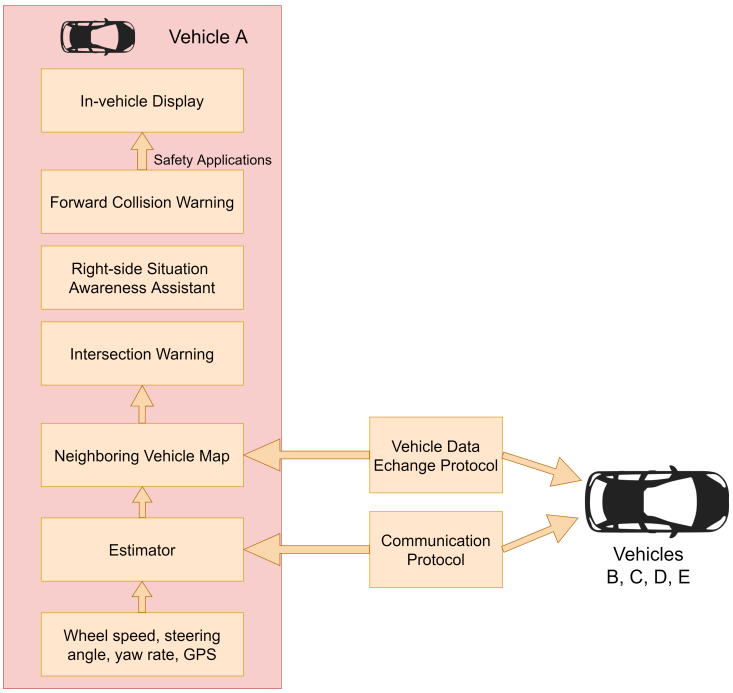


Figure 2.6: Algorithm system architecture (adapted from [19]).

This architecture includes an "Estimator" that obtains data from the Global Positioning System (GPS) and some more information from the vehicle's onboard sensors, broadcasting this information through a communication protocol. Simultaneously, it receives neighbouring information and builds an environment map, to have a scenario overview in which the vehicle is included. Having this information, it is possible to run safety applications capable of alerting the driver about possible collisions with surrounding vehicles and, if the situation is confirmed, this information is made available to the driver through an in-vehicle display.

The different scenarios analysed by the proposed algorithm are intersections and turn left situations. In the first scenario, the algorithm will only be used if the difference between both vehicle's direction is between 5 and 175 or between 185 and 355 ($5 \leqslant$

$d\theta \leqslant 175$ or $185 \leqslant d\theta \leqslant 355$, where $d\theta = |\theta_1 - \theta_2|$), meaning that both vehicles are not driving in the same lane or in opposite directions. Then, the second scenario is only verified if the difference between both vehicles' direction is between 175 and 185 ($175 \leqslant d\theta \leqslant 185$), meaning that both vehicles drive at opposite directions.

In intersection situations, as shown in Figure 2.7, the algorithm is based on the parameters, Time to collision (Time to collision (TTC)) and Distance to collision (Distance to collision (DTC)). Before calculating these parameters, a vehicle's trajectory projection is made, assuming that the speed and direction remain constant in both vehicles, to find the collision point between them characterized in Figure 2.7 as "Collision point".

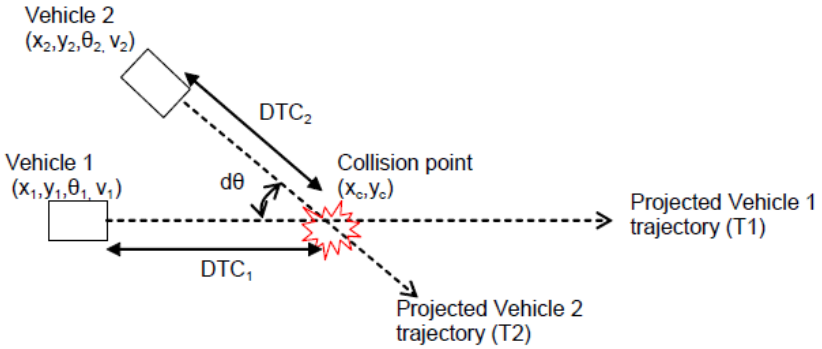


Figure 2.7: Projected collision point at intersections (extracted from [19]).

After the prediction of each vehicle's movement, the TTC parameter is calculated for each vehicle using equation 2.13.

$$TTC = DTC/v \tag{2.13}$$

Where v is the vehicle's speed. If $TTC_1 < 7s$ and $TTC_2 < 7s$, a collision is possible and a warning must be sent to the driver. If the condition is not verified, a new analysis is made and if $DTC_1 < 50m$ and $v_1 < 14km/h$ and $TTC_2 < 7s$ is verified, a warning must be issued. If none of these conditions is fulfilled, no warning is sent to the driver.

For the left turn situation, the TTC parameter's calculation is accomplished using Equation 2.14.

$$TTC = d/v_1 + v_2 \tag{2.14}$$

Where d is the distance between the vehicles and v_1 and v_2 the respective vehicles' speed, as shown in Figure 2.8.

The collision warning is triggered only if $TTC < 7s$.

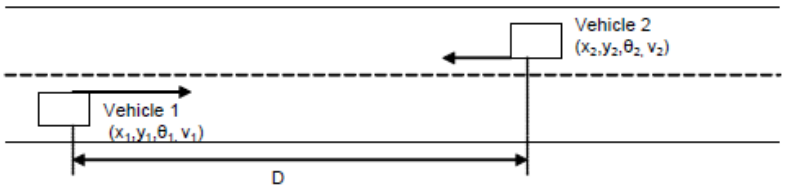


Figure 2.8: Projected collision point at the turn left situation (extracted from [19]).

2.3.3 GPS based Vehicular Collision Warning System using IEEE 802.15.4 MAC/PHY Standard

Anurag *et al* [20] proposed an algorithm also based on safety zones however using a different approach when calculating its dimensions. As shown in Figure 2.9, the safety zone covers, not only the vehicle but also a big distance in front of it (F) creating a larger safety zone than the algorithm described in Section 2.3.1.

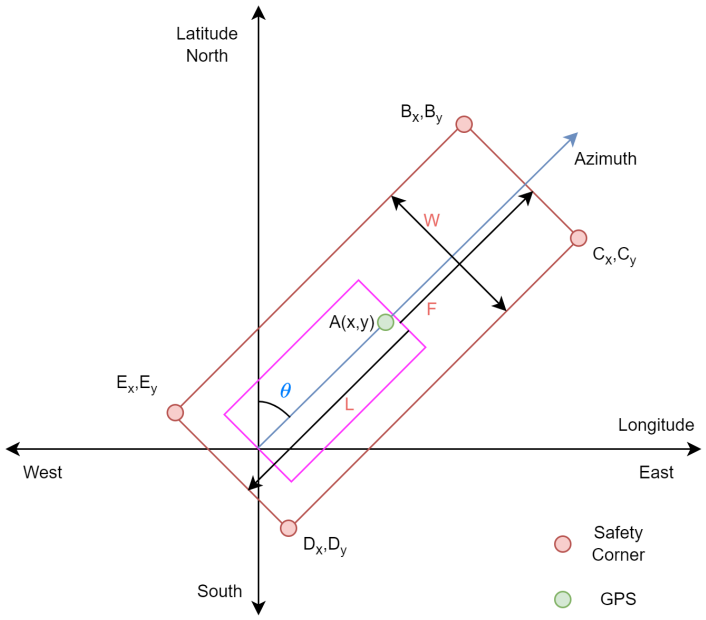


Figure 2.9: Vehicle model with safety zone (adapted from [20]).

As can be observed in Figure 2.9, points B, C, D, E represent the safety zone corners and their coordinates are calculated using equations (2.15) and (2.16). F represents the front safety distance, L represents the back safety distance, W represents the safety zone's width and θ represents the azimuth angle in degrees. It is also assumed that the GPS coordinates, represented by A, are placed at the safety zone's centre ($W/2$).

$$\begin{cases} B_x = A_x + F\sin\theta - \dfrac{W}{2}\cos\theta \\ C_x = A_x + F\sin\theta + \dfrac{W}{2}\cos\theta \\ D_x = A_x - L\sin\theta + \dfrac{W}{2}\cos\theta \\ E_x = A_x - L\sin\theta - \dfrac{W}{2}\cos\theta \end{cases} \tag{2.15}$$

$$\begin{cases} B_y = A_y + F\cos\theta + \dfrac{W}{2}\sin\theta \\ C_y = A_y + F\cos\theta - \dfrac{W}{2}\sin\theta \\ D_y = A_y - L\cos\theta - \dfrac{W}{2}\sin\theta \\ E_y = A_y - L\cos\theta + \dfrac{W}{2}\sin\theta \end{cases} \tag{2.16}$$

To calculate the front safety distance (F), it is considered the current vehicle's speed, acceleration, braking power and human reaction time, represented by the average amount of time taken to apply the brakes once an indication is provided.

D_1 and D_2, as shown in Figure 2.10, which when added together yield F. These variables can be calculated as shown in Equation 2.17.

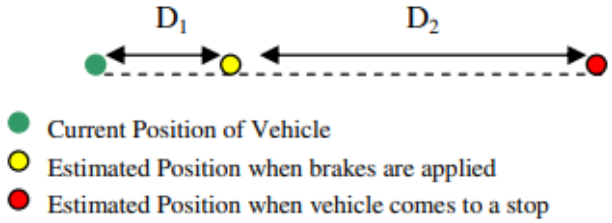


Figure 2.10: Schematic diagram for calculating the front safety distance - F (extracted from [20]).

$$\begin{cases} t_s = \dfrac{v + atpr}{B_r} \\ D_1 = vt_{pr} + \dfrac{1}{2}at_{pr}^2 \\ D_2 = V_1 t_s + \dfrac{1}{2}B_r(t_s)^2 \\ V1 = v + at_{pr} \end{cases} \tag{2.17}$$

Then F is calculated using equation (2.18).

$$F = D_1 + D_2 = vt_{pr} + \frac{1}{2}at_{pr}^2 + \frac{3}{2}\frac{(v + t_{pr})^2}{B_r} \tag{2.18}$$

For the calculation of these parameters, the following information is used:

- v: vehicle's current speed, in m/s;
- a: vehicle's current acceleration, in m/s^2;
- t_{pr}: human reaction time, in seconds;
- B_r: retardation due to the brakes, assuming that the brakes are applied at its maximum potential;

The rest of the safety zone dimensions are pre-defined based on vehicle dimensions. Having the safety zones calculated, the possibility of collision is then analysed. Unlike the algorithm described in Section 2.3.1, this one doesn't need to predict the vehicle's movement since the safety zone already takes into account the braking action and, therefore, only overlapping security zones are analysed. Figure 2.11 demonstrates this situation.

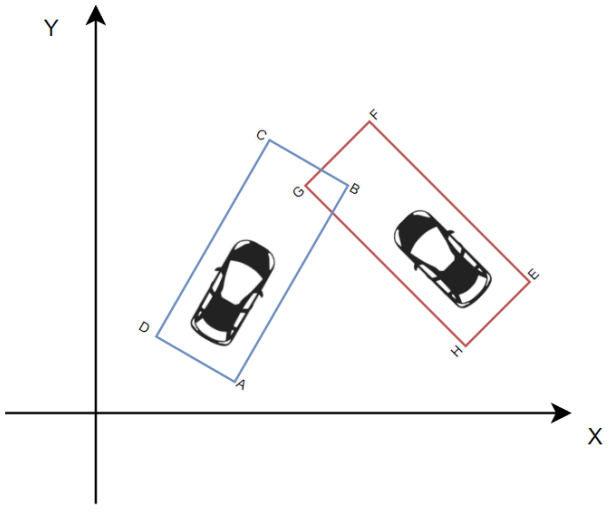


Figure 2.11: Safety zones representation on a collision scenario.

Using Figure 2.11 as an example, the point G is inside the other vehicle's perimeter if it is left of every polygon's corners while traversing the corners counterclockwise.

For example, the point *G(x,y)* is left of the segment from $A(x_1, y_1)$ to $B(x_2, y_2)$ if equation (2.19) is verified. A collision is predicted if this verification happens with all the security zone's corners, meaning that the point is in the other vehicle's safety zone's perimeter.

$$c = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) > 0 \tag{2.19}$$

2.4 Forecasting Approach in VANET based on vehicle collision alert

What Djamel *et al* [21] proposed, wasn't a collision warning algorithm, but can be used as an auxiliary to improve how these algorithms work. The authors suggested a vehicle's kinematic model using only information of position, speed, steering and heading angle.

Each vehicle has a state described in Figure 2.12 which can be represented by $(x, y, v, \theta, \alpha)$, where x and y give the rear axle's centre position, represented by $O^{'}$, v represents it's speed, θ represents the angle between the vehicle body and the horizontal axis, giving the vehicle's heading, and at last, α represents the angle between θ and the steering wheel angle.

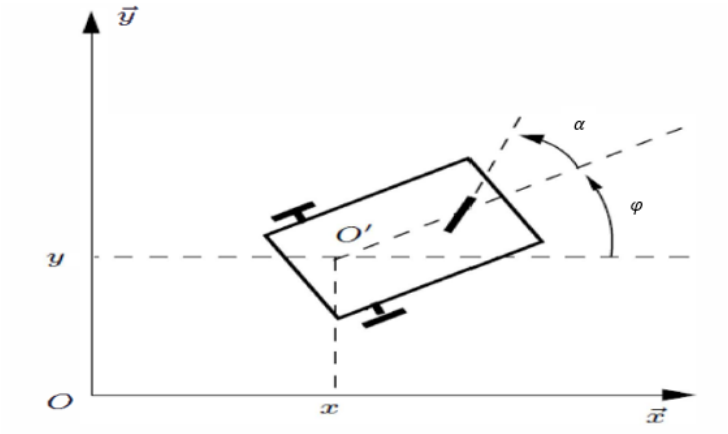


Figure 2.12: Vehicle parameters (extracted from [21]).

To create a vehicle's movement estimator, the equations in (2.20) were presented, using the vehicle's states, shown before, it is possible to create a vehicle's next position estimation and also predict the speed and heading information.

$$\begin{cases} X_{k+1} = X_k + v_k \cos(\theta_k)t \\ Y_{k+1} = Y_k + v_k \sin(\theta_k)t \\ v_{k+1} = v_k t \\ \theta_{k+1} = \theta_k + v_k \dfrac{\tan(\alpha)}{d} t \end{cases} \tag{2.20}$$

In these equations, X and Y represent the vehicle's coordinates, v represents speed and, finally, θ represents the vehicle's heading, which will be influenced by the steering wheel angle. t is referred to the sampling period and d to the distance between O' and the steering wheel (α).

2.5 Survey Conclusions

Despite the study of rear-end collision algorithms, they have limitations in terms of their versatility since they can only be used for situations when a vehicle's front has the risk of collision with the leading vehicle's rear end, not predicting situations of intersections or vehicles with different directions but with routes that could cause a collision.

In terms of the algorithms that use the vehicles' overlapping, the algorithm in Section 2.3.2 was created for specific situations using different equations for each situation, which could limit the algorithm usage in other situations. That said, the algorithms of sections 2.3.1 and 2.3.3 would be the best options to use. Given the desire to use the model developed by Djamel [21] in Section 2.4, the algorithm developed by Ahrems [18] with the kinematic model developed by Djamel [21] were used in the application proposed within the scope of this work.

3

Development and Implementation

This chapter describes the development and implementation of the proposed solution, which will be organized into four Sections. In the first place, in Section 3.1, will be explained the development of a simulator to create an application prototype to test different approaches without the danger and complexity of testing them in a real-life scenario. Then, in Section 3.2, the developed application architecture will be shown, followed by the algorithm used in it, in Section 3.3. Lastly, a hybrid environment between ITS-G5 and cellular networks has been developed allowing the interoperability between these parts to provide a solution for vehicles that do not have access to an ITS-G5 OBU, named legacy vehicles.

3.1 Simulation Application

At an early stage of the project, a simulation application was developed to create a system prototype without the need of testing it with real vehicles, lowering the risks and costs of these tests. The objective was to create a simulator as close as possible to the vehicle application and, therefore, it was necessary to find alternatives to simulate aspects like communication problems, latency or lost messages that could happen in a real system scenario. Furthermore, it is also useful to get a first touch with collision prevision algorithms and the problems associated with them.

With this objective, the application was developed working like shown in Figure 3.1. Each simulated vehicle is represented by a *Vehicle Thread* that can communicate with

different simulated vehicles and is capable of executing a collision prevision algorithm. Besides that, another thread was used to manage information with the objective of simulating communication and message delivery problems (*Information Management Thread*).

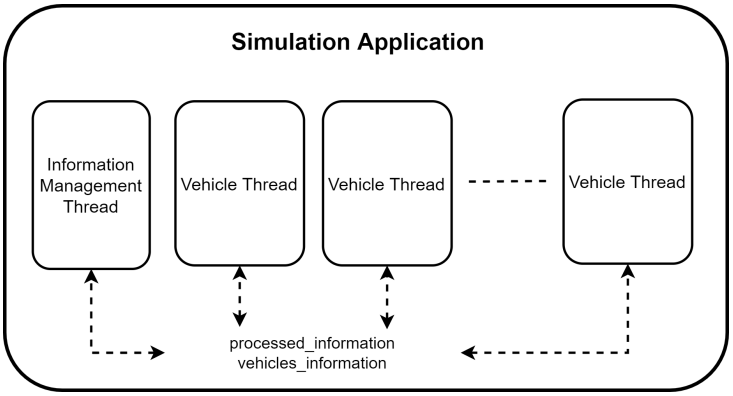


Figure 3.1: Simulation application operation.

Since ITS-G5 detailed communications were not simulated, there was the need for using another method. This was accomplished using *vehicles_information* variable, which stores messages sent by every vehicle, and *processed_information* variable that, after being managed by *Information Management Thread*, stores information to be received by each vehicle. This way, the process of communication between vehicles is simulated using a real-life simulation approach.

When looking with more detail into the *Vehicle Thread*, as shown in Figure 3.2, it must be divided into three different parts, each one of them a different thread. This separation is mandatory because it is extremely important that the send and receive parts do not depend on the algorithm's execution avoiding delays in these functions.

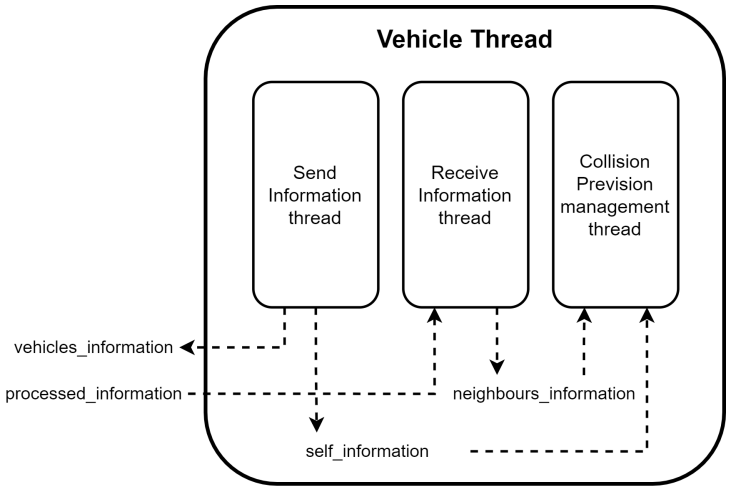


Figure 3.2: Vehicle thread architecture.

The communication between vehicles is simulated using the operation flow in Figure 3.3, which demonstrates each message's path from the moment it is created in *Send Information* until it is used in *Collision Prevision Management*. The example shown demonstrates a message created in *Vehicle 1* and received in *Vehicle 2*. The message passes through *Information Management* to simulate problems like latency.

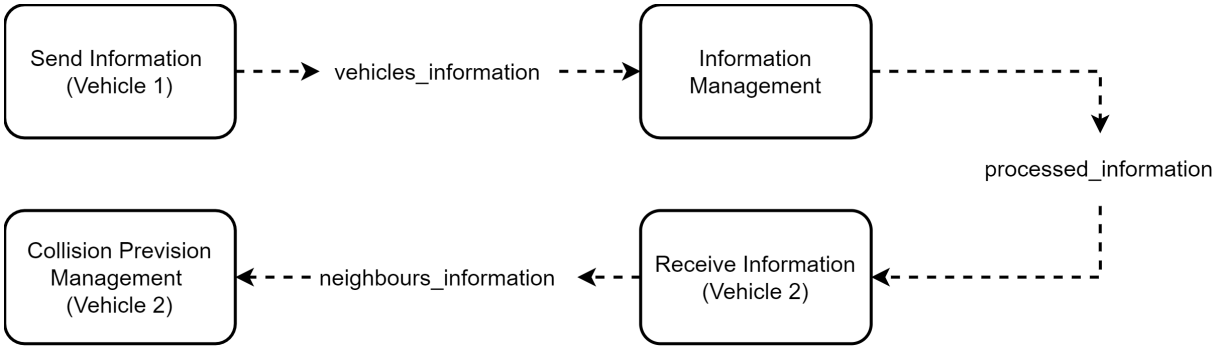


Figure 3.3: Simulator message paths.

When looking for Figure 3.4, the *Send Information Thread* starts with a movement simulation that will be responsible for creating a path simulation made by that vehicle which, in some cases, can be only one position to test a specific collision scenario. After this, the information is stored in *self_information* and *vehicles_information*. Lastly, a time sleep is applied to define the sending message frequency.

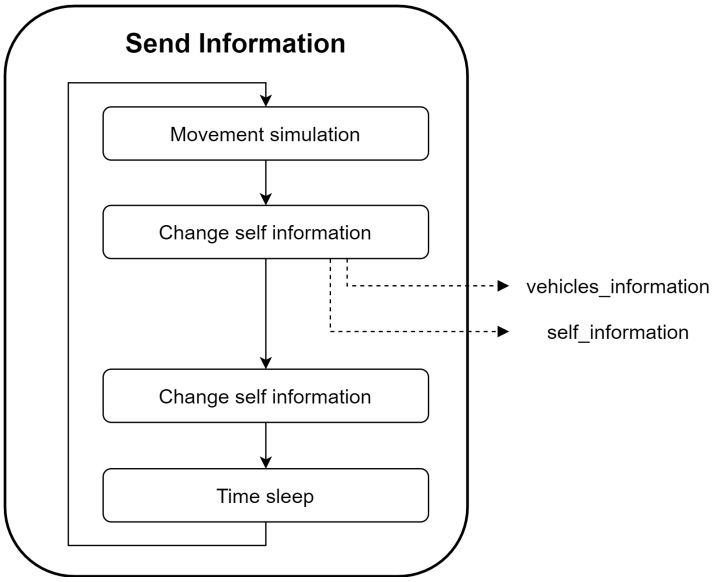


Figure 3.4: Execution flow of simulator *Send information thread*.

The *Receive Information Thread*, in Figure 3.5, starts by obtaining the receiving messages from *neighbours_information* variable to be used later by the *Collision Prevision Management Thread*. This thread could be omitted and the information could be used directly

from *processed_information*, but it would lose the simulation of receiving messages from other vehicles in a real scenario.

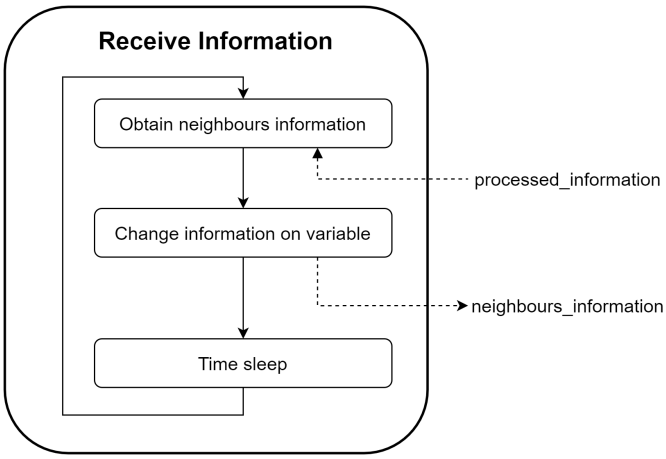


Figure 3.5: Execution flow of simulator *Receive Information Thread*.

Lastly, the *Collision Prevision Management Thread*, in Figure 3.6, will be responsible for the collision warning algorithm's management. In every loop iteration, it will get neighbouring and self vehicles' information to create another thread, called *Collision Prevision Thread* that will be responsible for analysing the possibility of collision between the main vehicle and each neighbouring vehicle. The *Collision Prevision Algorithm*'s execution flow will vary based on which algorithm is being used. After that, a time sleep is applied, based on the vehicle's speed, to create a rate of collision prevision executions low enough to predict every possible collision point. If this sleep is too big, some collision situations may not be detected creating a false negative situation for collision possibility. This issue will be approached later in this work.

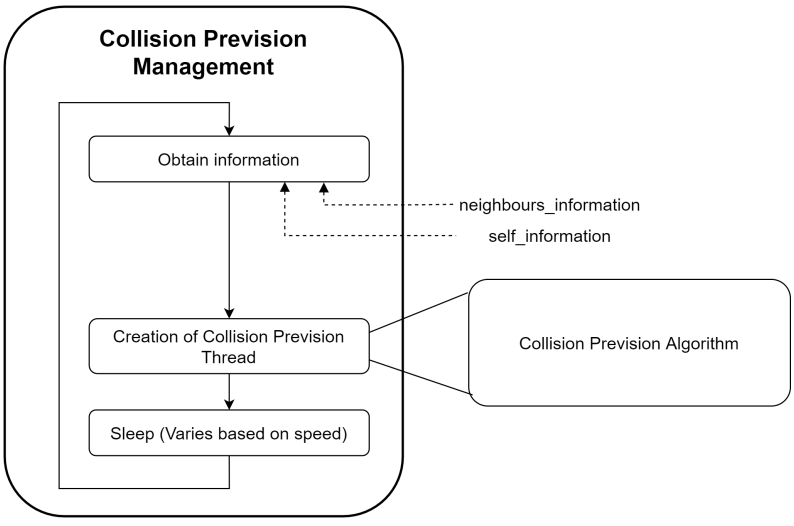


Figure 3.6: Execution flow of *Collision Prevision Management Thread*.

Having described the *Vehicle Thread*, it is now described the *Information Management Thread*, as shown in Figure 3.7. It starts by getting the information from *vehicles_information* and it is then applied the desired limitation. After that, the information is then inserted in *processed_information* to be used by the vehicles. This process of creating limitations to message delivery is a simulator's important part since it is crucial to simulate a real environment where latency and message loss are a real problem.

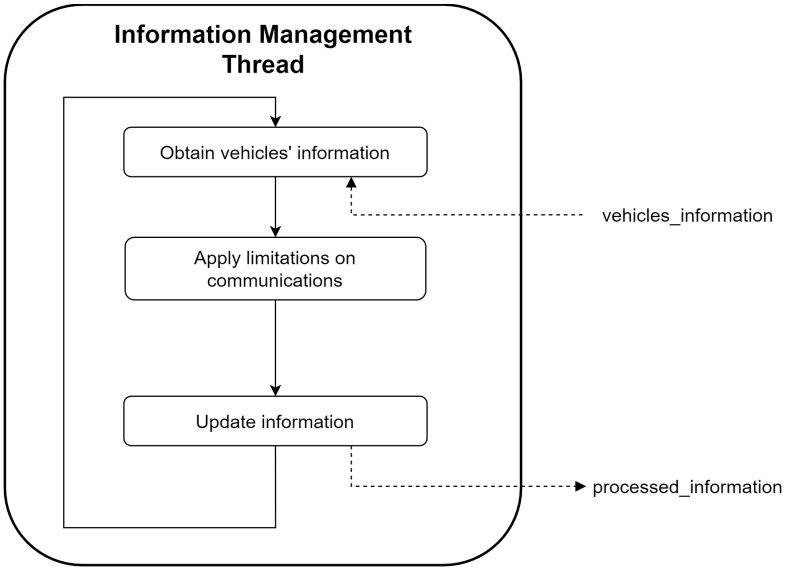


Figure 3.7: Execution flow of simulator *Information Management Thread*.

The simulator was a key factor in the work development since it allowed the creation of a functioning prototype for the core application.

3.2 Collision Warning Application Architecture

Ideally, collision warning applications should be carried out on all vehicles to improve the driving experience. As shown in Figure 3.8, these applications exchange information with the neighbouring vehicles using any DSRC channel. In this specific case, the application use ITS-G5 as communication system between vehicles using CAM messages, that contain information to be used by the application.

After defining the simulator architecture, in Section 3.1, it was expected that the application would follow a similar architecture, however, despite the simulator working as expected, when the application implementation was completed, a problem with its efficiency was detected. It was verified a high processing latency which would cause an increase in the time that the application would take to detect a collision. Because of

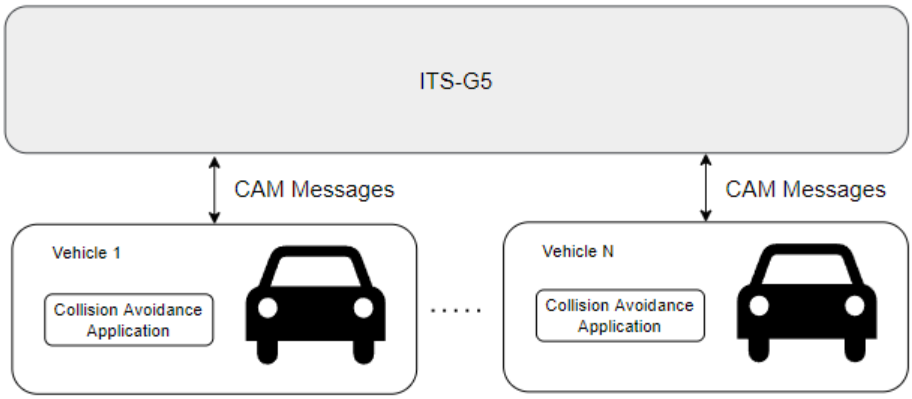


Figure 3.8: Road environment with collision warning applications.

that, a different and more time-efficient solution was designed. The application's new architecture is shown in Figure 3.9.

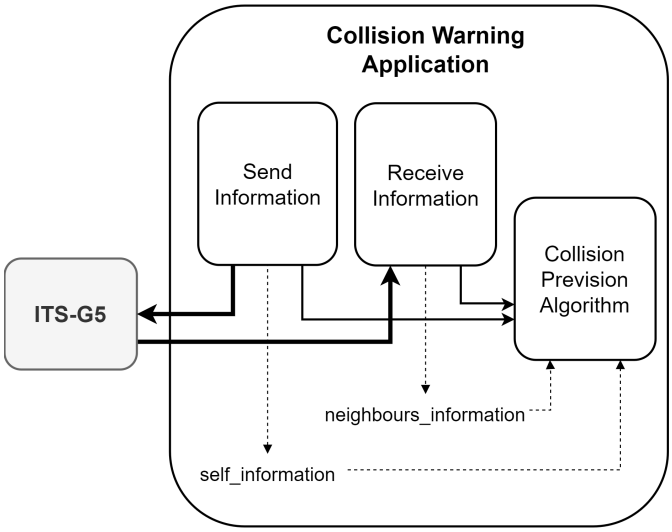


Figure 3.9: Collision warning application architecture.

The application is divided into two main parts, *Send Information* and *Receive Information*, that are responsible for the communication with surrounding vehicles, via ITS-G5. Since both parts don't depend on each other's execution and their behaviour cannot be delayed by each other, these parts must be executed in parallel. To share information between these parts and the algorithm (*Collision Prevision Algorithm* part), two variables, "self_information" and "neighbours_information", are used.

Going into the details of the different parts, Figures 3.10 and 3.12 show each part's execution flow to accomplish the proposed objective.

The *Send Information* part is divided into two different execution flows. The first part

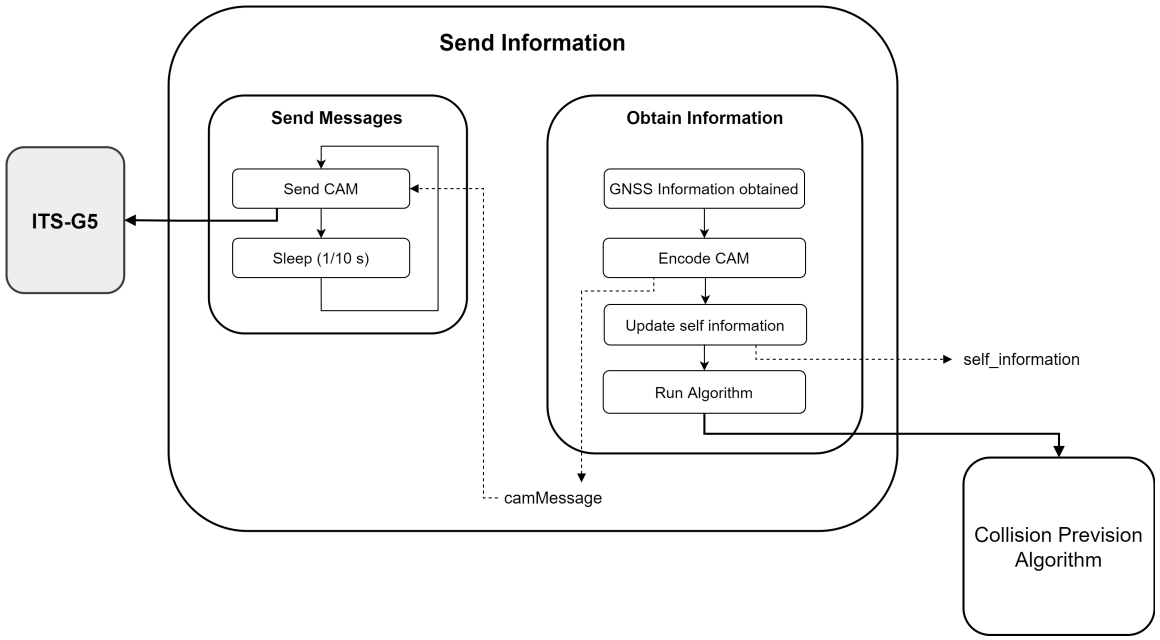


Figure 3.10: Execution flow of *Send Information* part.

(*Send Messages*) is only responsible for sending CAM messages at a defined frequency. The second part (*Obtain Information* part), is responsible for the extraction of GNSS information and, using this information, encode a CAM message that is then stored at *camMessage* variable. After this, the information is stored at *self_information* variable. Since there was a position variation, the *Collision Prevision Algorithm* is executed.

The process of sending messages has one problem related to the GNSS accuracy since the information is only updated once per second, meaning that, when sending 10 messages per second, these messages will contain the same information, congesting the network with repeated messages. To overcome this problem, little adjustments are made for every message creating a prevision of location for the vehicle according to its last speed and heading information. Figure 3.11 demonstrates this process, considering 4 messages per second. In a), all 4 messages are sent with the information obtained by the GNSS device which causes the repetition of that message. In b), calculating the predicted position according to the vehicle's speed and heading, the 3 messages contain the vehicle's predicted position. These adjustments increase the algorithm's accuracy.

To calculate this adjustment, the last information obtained from GPS is used on equations 3.1 and 3.2 [22].

$$lat_1 = \arcsin\left[\sin(lat_0) \times \cos\left(\frac{d}{R}\right) + \cos(lat_0) \times \sin\left(\frac{d}{R}\right) \times \cos(\theta)\right] \tag{3.1}$$

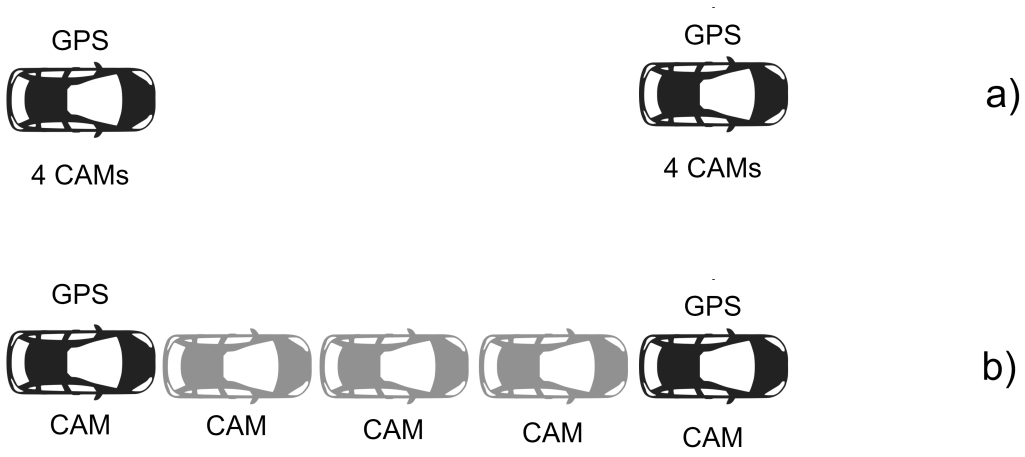


Figure 3.11: GNSS/GPS refresh rate. a) Without position adjustments. b) With position adjustments.

$$lon_1 = lon_0 + \text{atan2}\left[\sin(\theta) \times \sin\left(\frac{d}{R}\right) \times \cos(lat_0), \cos\left(\frac{d}{R}\right) - \sin(lat_0) \times \sin(lat_1)\right] \quad (3.2)$$

Where:

- lat_0: last GPS's latitude obtained;
- lon_0: last GPS's longitude obtained;
- lat_1: new latitude calculated;
- lon_1: new longitude calculated;
- d: distance between the position obtained via GPS and the new position, calculated using equation 3.3, in kilometers;
- R: Earth radius, in kilometers;
- θ: vehicle heading.

$$d = (t \times v)/1000 \quad (3.3)$$

Where:

- t: time passed since the last GPS's position was obtained, in seconds;

- v: vehicle's speed, in meters per second.

The *Receive Information* part, in Figure 3.12, waits for a message to be received and when it happens, the message is decoded and the information is stored at the *neighbours_information* variable. Since there has been a neighbour's information change, the *Collision Prevision Algorithm* is executed.

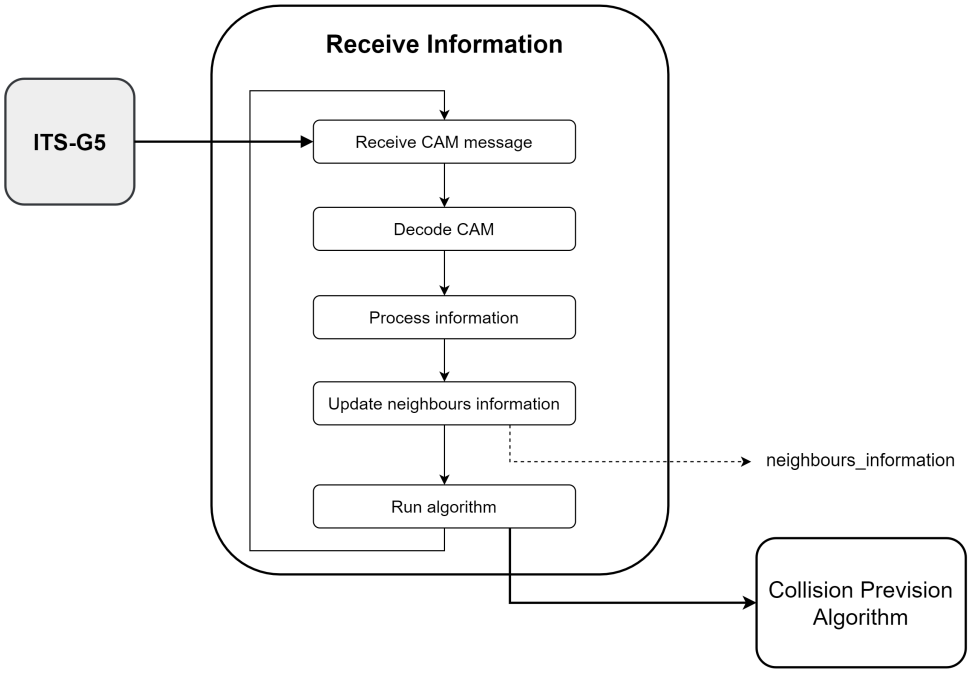


Figure 3.12: Execution flow of *Receive Information* part.

There is an important aspect to be considered before the algorithm explanation. The vehicles when sharing position information in CAM messages, the information is sent in geographical latitude and longitude in a World Geodetic System (WGS84) coordinate system. However, the algorithm was created to be used on a Cartesian coordinate system, so it is necessary to make a conversion between these systems. Figure 3.13 shows the information representation for the distance and heading's calculation between vehicles (transmitter and receptor).

The equations used to make the conversion between the coordination systems are equations 3.4, 3.5, 3.6 and 3.7.

$$\delta = \arccos\left[\sin(lat_t) \times \sin(lat_r) + \cos(lat_t) \times \cos(lat_r) \times \cos(lon_t - lon_r))\right] \tag{3.4}$$

$$d = r \times \delta \tag{3.5}$$

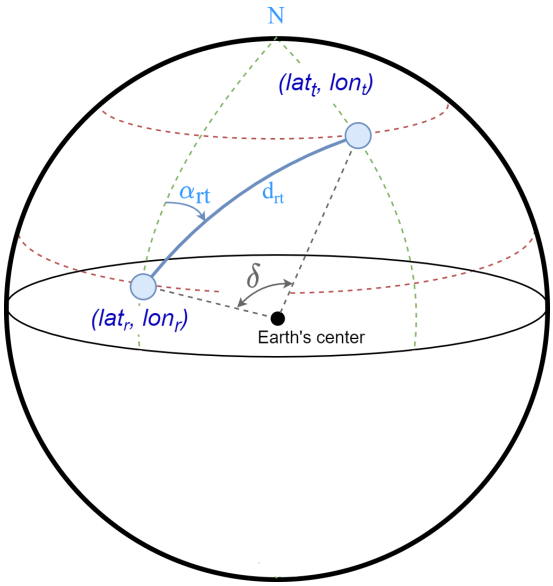


Figure 3.13: Geographical coordinates representation.

$$\alpha_{rt} = \begin{cases} 2\pi - \alpha & lon_r \geqslant lon_t \\ \alpha & lon_r < lon_t \end{cases} \tag{3.6}$$

Where:

$$\alpha = \arccos\left[\frac{\sin(lat_t) - \sin(lat_r) \times \cos(\delta)}{\sin(\delta) \times \cos(lat_r)}\right] \tag{3.7}$$

The following information is used on the equations:

- lat_t: neighbour's latitude, in radians;
- lon_t: neighbour's longitude, in radians;
- lat_r: vehicle running the application's latitude, in radians;
- lon_r: vehicle running the application's longitude, in radians;
- δ: angle between the lines that connect the transmitter and the receptor to earth's center, in degrees;
- α_{rt}: angle between the line that connects the transmitter and the receptor and the meridian in which the receptor is located, in degrees;
- d: distance between the transmitter and the receptor, in kilometers;

- *r*: Earth radius, in kilometers.

After calculating these variables, Figure 3.14 shows how these variables are used in the Cartesian coordinates system. The vehicle running the application is considered to be at position (0,0) and the distance between vehicles in the Cartesian coordinates system is represented in meters.

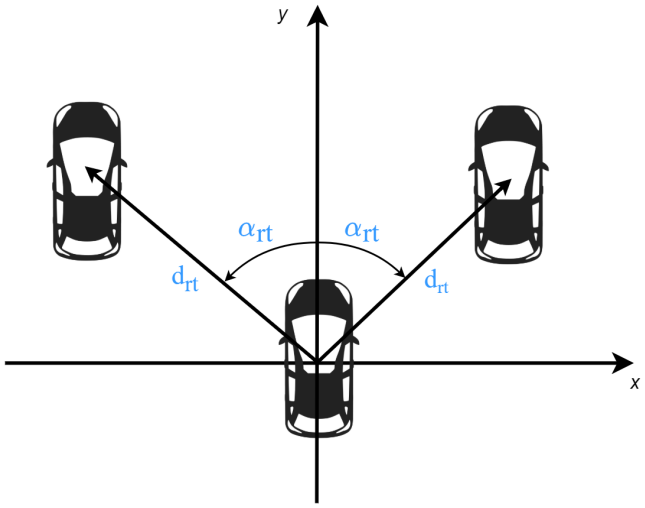


Figure 3.14: Cartesian coordinates system after conversion from Geographical coordinates.

As observed previously in this section, the algorithm is executed when there is a state alteration on any involved vehicle or when a message from a new vehicle is received, creating a *Collision Prevision Algorithm* part's instance.

3.3 Collision Prevision Algorithm

In this phase, it was defined the algorithm running on the *Collision Prevision Algorithm* part shown in Figures 3.10 and 3.12.

The algorithm is based on the work proposed by Janeks Ahrems [18] and the movement prediction calculation proposed by Djamel [21]. The algorithm consists of a cycle of 4 main steps, as shown in Figure 3.15, that are repeated while the algorithm is running:

1. Calculate safety zone for vehicles;
2. Detect collision;

3. Predict next position;

4. Calculate distance between vehicles;

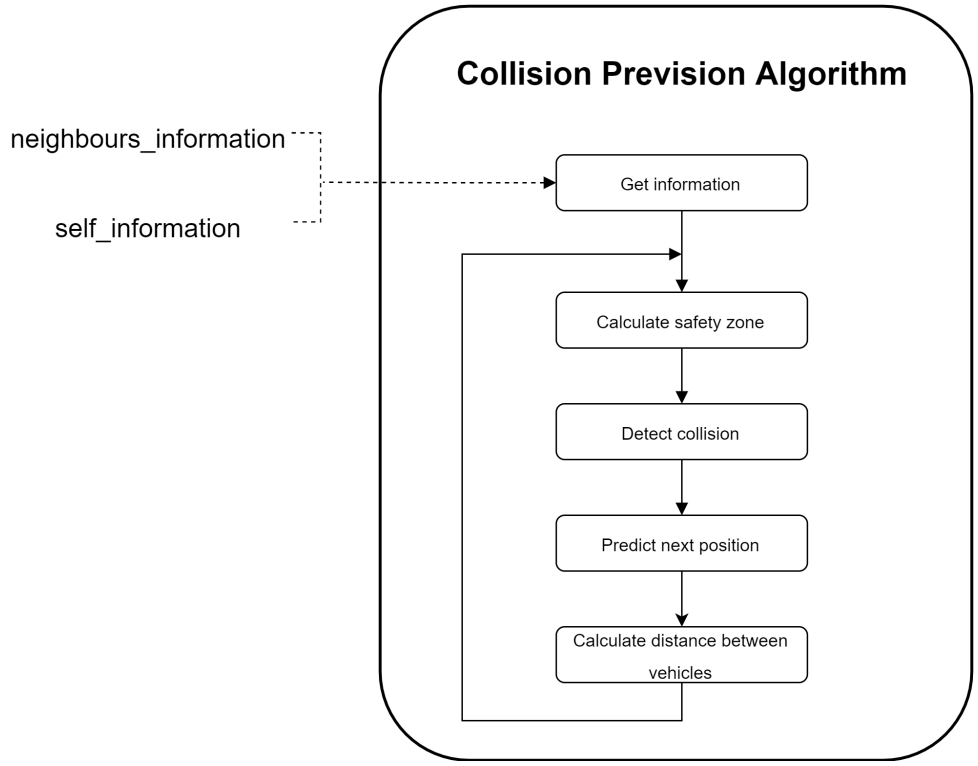


Figure 3.15: Algorithm execution flow.

To better explain the algorithm, the scenario in Figure 3.16 will be used as example.

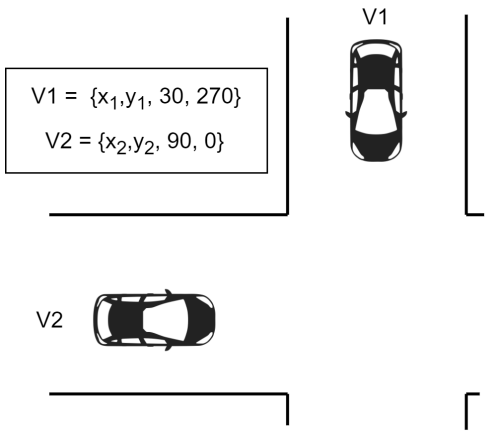


Figure 3.16: Algorithm explanation scenario.

3.3.1 Safety Zones Calculation

The algorithm starts by calculating the safety zone for every vehicle involved. Figure 3.17 shows how the safety zone is represented. The representation is similar to Figure 2.4 in Section 2.3.1, differentiating the heading representation. This alteration causes the need of changing the equations used to determine the safety zone corners.

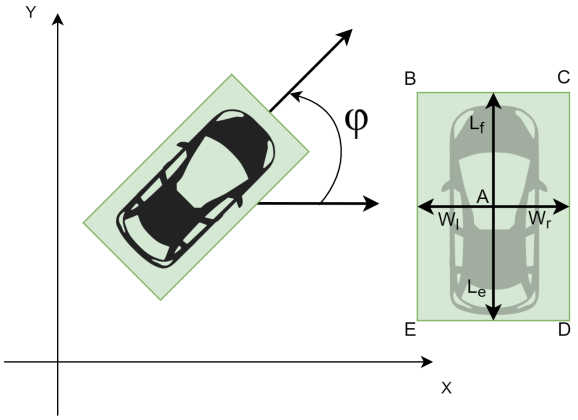


Figure 3.17: Safety zones representation.

The information used in the figure is specified in Section 2.3.1 and the equations used for the calculation of points B, C, D and E are 3.8 and 3.9 that are an adaption of equations 2.6 and 2.7.

$$\begin{cases} X_B = X_A + L_f \cos(\theta) - W_l \sin(\theta) \\ X_C = X_A + L_f \cos(\theta) + W_r \sin(\theta) \\ X_D = X_A - L_e \cos(\theta) + W_r \sin(\theta) \\ X_E = X_A - L_e \cos(\theta) - W_l \sin(\theta) \end{cases} \tag{3.8}$$

$$\begin{cases} Y_B = Y_A + L_f \sin(\theta) + W_l \cos(\theta) \\ Y_C = Y_A + L_f \sin(\theta) - W_l \cos(\theta) \\ Y_D = Y_A - L_e \sin(\theta) - W_r \cos(\theta) \\ Y_E = Y_A - L_e \sin(\theta) + W_r \cos(\theta) \end{cases} \tag{3.9}$$

The algorithm developed by Ahrems [18] uses the equations having the heading according to Figure 3.18 a). Since, in this work, it was pretended the orientation defined in Figure 3.18 b), it was necessary to make alterations on the equations used to define the safety zone corners to consider this alteration.

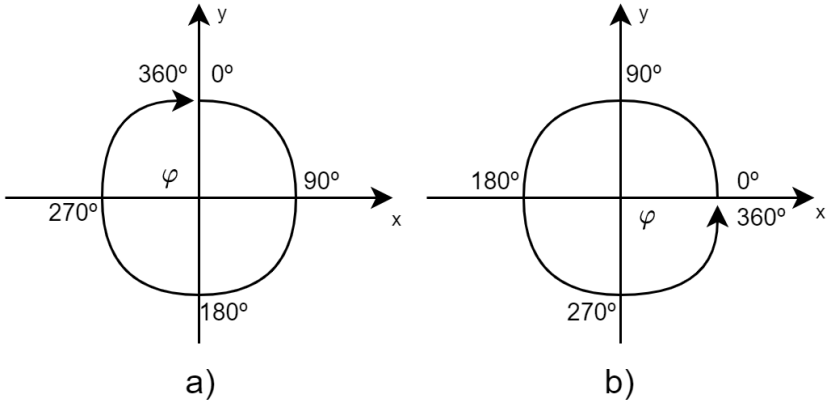


Figure 3.18: Heading representation conversion.

Using the scenario created before, Figure 3.19 demonstrates this process, where the two vehicles have their safety zones drawn.

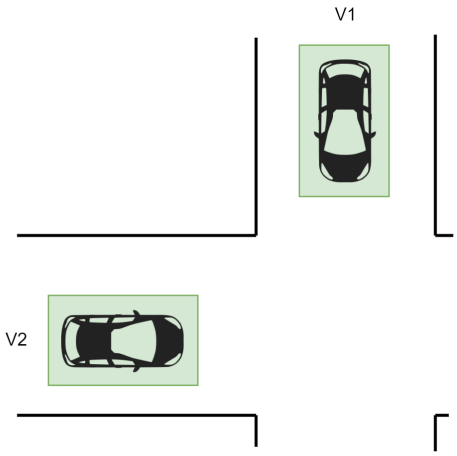


Figure 3.19: Vehicle safety zone representation on the scenario.

3.3.2 Collision Detection

Having safety zones' points defined, it is possible to use these points B, C, D and E to analyse the overlapping of safety zones, which is done using the condition on equation 2.9, in Section 2.3.1. Figure 3.20 shows an example of the overlapping of two vehicles' projection, represented by the red area.

3.3.3 Vehicles' Projection Prediction

The next step consists on the prediction of the vehicles' next position, which is done using equations 2.20, in Section 2.4. Figure 3.21 shows this process where it is possible

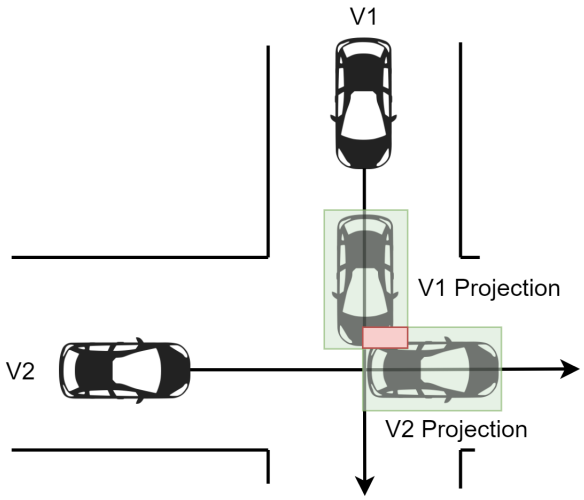


Figure 3.20: Safety zones overlapping.

to see that each vehicle projection has a different location than the vehicle itself, showing that the projection is a vehicle movement prediction considering a constant speed and direction, following the path defined by the arrow shown at the image.

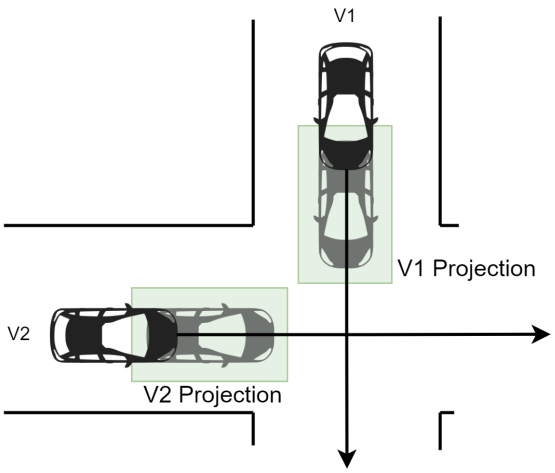


Figure 3.21: Vehicles projection representation.

Equations in 3.10 are used to predict the vehicle's movement. The equations are similar to 2.20 considering a constant speed and heading, only changing the X and Y coordinates.

$$\begin{cases} x_{k+1} = x_k + v\cos(\theta_k)t \\ y_{k+1} = x_k + v\sin(\theta_k)t \end{cases} \tag{3.10}$$

The t variable refers to the time between positions. This variable cannot be a constant

value since it would cause situations where collisions wouldn't be detected because the algorithm couldn't detect the safety zones' overlapping. As shown in Figure 3.22, when using a $t = 1s$ between iteration 1 and 2, a big space between iterations is created causing an area where the algorithm didn't evaluate the possibility of a collision that, in this case, would be detected if t were lower.

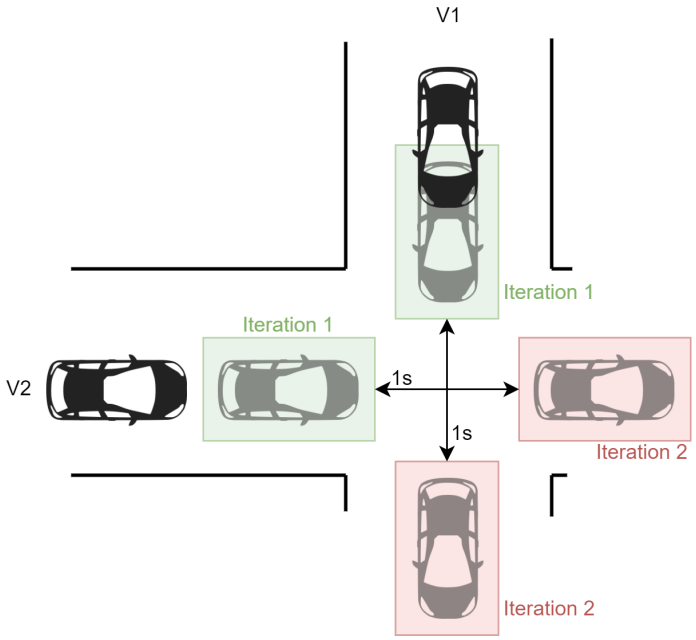


Figure 3.22: Movement prediction period between iterations.

To overcome this issue, the t variable is based on distance and not time. Based on the safety zone size (6 meters) the application calculates how much time the vehicle takes to travel 6 meters, based on its speed. This guarantees that the algorithm will evaluate the collision possibility in every area travelled by the vehicle.

The t is calculated using equation 3.11.

$$t = 6/v; \tag{3.11}$$

Where v stands for speed in meter per second.

At each algorithm's execution, the t value is determined among all vehicles involved and the vehicle with higher speed will determine the t value to be used for all vehicles.

3.3.4 Calculate Distance Between Vehicles

Finally, the next step of the algorithm computes the distance between vehicles' projections to analyse if the algorithm should continue or not. Figure 3.23 shows a situation

of when to stop the algorithm. In Figure 3.23 a) is still a situation where the algorithm should continue its execution since the vehicles' projection keeps getting closer, however, on b) V2's projection already passed the possible collision point and the distance between both vehicles' projection is increasing, meaning that there's no possibility of collision between the vehicles and, because of that, the algorithm should stop its execution.

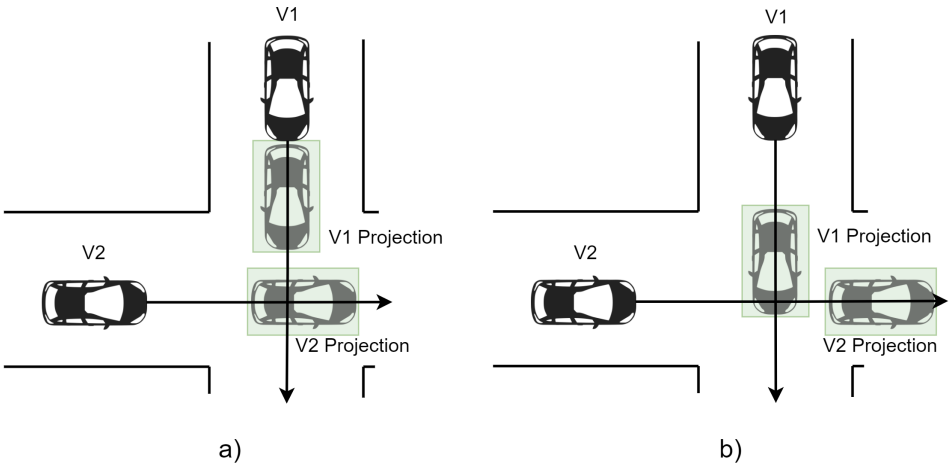


Figure 3.23: Distance between vehicles usage.

Equation 3.12 is used to calculate the distance between the vehicles' projections.

$$d = \sqrt{(neighbour_x - main_x)^2 + (neighbour_y - main_y)^2} \tag{3.12}$$

Overall, the algorithm uses the explained steps to accomplish the objective. At each algorithm's iteration, the information from each vehicle involved is used: (x position, y position, speed and direction). Then, the safety zones for each vehicle are calculated and the overlapping of safety zones is checked. If there is no collision, a vehicle's location prediction is calculated. Finally, the distance between vehicles' projections is calculated. This process is repeated for each neighbouring vehicle until: (i) there is a collision detection, (ii) the distance between the vehicles' projection starts to increase or (iii) the TTC is longer than 7 seconds. When there is no neighbouring vehicle left to analyse the collision, the algorithm ends its execution.

The application proposed communicates with neighbouring vehicles using ITS-G5, however, as mentioned in Section 1.2.2, to do this, it is necessary equipment capable of completing this task, like an OBU or a RSU. Although in the future most vehicles are expected to be equipped with OBUs right from the factory, currently not every vehicle circulating on public roads has access to it. To overcome this issue, it is proposed the

use of a smartphone, using mobile cellular networks, for these vehicles. For that, a hybrid environment between ITS-G5 and cellular networks is proposed next.

3.4 ITS-G5 and Cellular Network System

As mentioned by C-Roads [23], a hybrid environment using the ITS-G5 network and cellular networks to perform interoperability between connected and legacy vehicles is extremely important because it "is unacceptable that people would die on European roads because vehicles cannot "speak" to each other or implemented roadside units due to non-interoperable communication systems" (p.2) proving that this type of compatibility is extremely important to improve the road safety.

The solution proposed uses a smartphone application to disseminate vehicle information and to receive neighbours' information regardless of whether they are vehicles communicating on the ITS-G5 network or not. That said, an interoperability environment between the ITS-G5 network and cellular networks (3G, 4G or 5G) was built and tested.

The hybrid environment's architecture was defined to create a solution capable of communication between legacy and connected vehicles aiming at a fast solution between the involved parts. Figure 3.24 presents the environment's architecture.

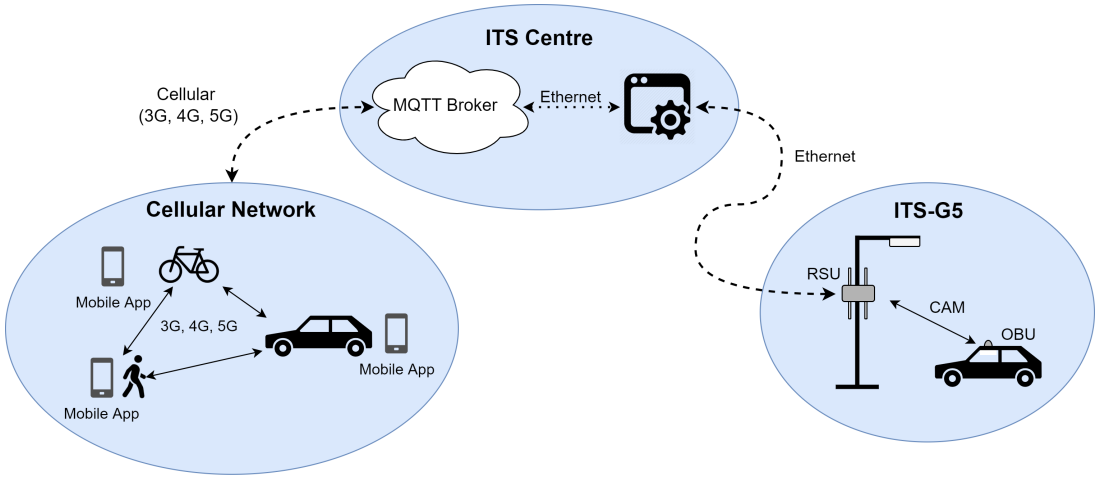


Figure 3.24: Hybrid environment architecture.

The system presented is divided into three different parts: *Cellular network*, *ITS Centre* and *ITS-G5* network. Firstly, *Cellular Network* represents the type of communication used by all the road users who were otherwise not connected to the ITS environment but, using a mobile application, become connected parts of it. This includes legacy

vehicles and soft mobility, like pedestrians and bicycles. The mobile application allows these users to communicate with the ITS-G5 network using the *ITS Centre* as the intermediary part of this communication.

The *ITS-G5* part includes all the connected vehicles that, via an OBU, communicate with the neighbouring vehicles and with the infrastructure that supports this type of communication, such as an RSU.

Finally, the *ITS Centre* is the intermediary in communication between the parts and it contains a MQTT Broker and a server application that communicates directly with the ITS-G5 network.

In terms of physical infrastructure, the smartphone connects to the MQTT Broker via cellular network (3G, 4G, 5G). The Server application is connected with the MQTT Broker and the ITS-G5 part via Ethernet connection.

Data flows bidirectionally, so both parts, *Cellular Network* and *ITS-G5*, can exchange information between them and the *ITS Centre* is responsible for the forwarding of messages between both parts. When the data is being sent from a smartphone, the information is sent to the MQTT Broker, using the available cellular network, publishing the information on a defined topic. The server application initially subscribes to the topic and when receives the messages, as shown in Figure 3.25, it forwards them to the RSU, which disseminates the message to the ITS-G5 network, to be received by connected vehicles, creating awareness of vehicles connected via smartphone.

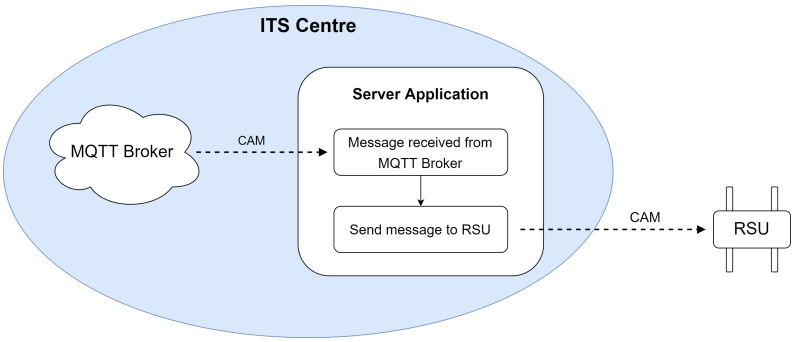


Figure 3.25: Server application's function with messages sent by smartphones.

In the opposite direction, the equipment on ITS-G5 network, sends CAM messages to an RSU that forwards this information to the server application. As shown in Figure 3.26, when the server application receives data from the RSU, it forwards the information to a specific topic in the MQTT Broker to disseminate the information to the smartphones.

Since this architecture has been defined and implemented in a small proportion, the message delivery management based on the vehicles' position wasn't considered.

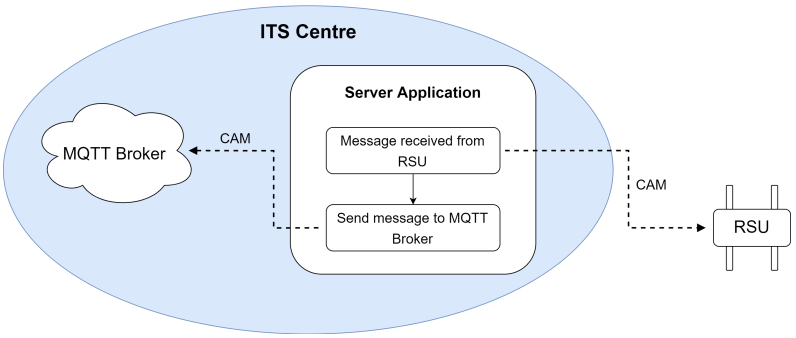


Figure 3.26: Server application's function with messages sent by OBUs.

The architecture has been developed and tested with success. Messages were exchanged between the ITS-G5 network and smartphones and all the equipment involved could exchange information between them.

When adapting the application defined in Section 3.2 to be used by smartphones, there is a need of changing the communication method. Figure 3.27 shows the application architecture with this alteration.

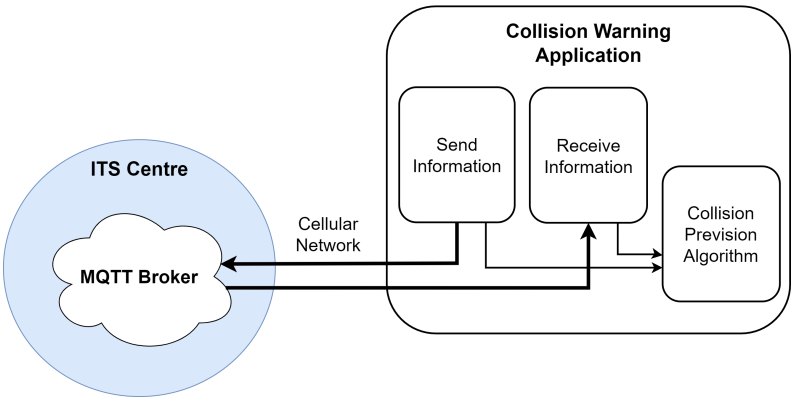


Figure 3.27: Application architecture using the hybrid environment.

Instead of using ITS-G5 to communicate with the surrounding vehicles, the communication is made using the available cellular network to publish messages to the MQTT Broker located at the *ITS Centre*.

4

Experimental Results

After defining the simulator, the application, and the hybrid environment architecture, it is now possible to run some tests that are useful to draw conclusions about the architecture designed and the work developed.

4.1 Hybrid Environment Latency Measurements

Since latency is a key performance indicator for the hybrid environment (as it measures the time it takes for a message to travel from an OBU to a smartphone), latency tests were made on this environment. To perform these tests, the OBU is responsible for sending CAM messages that are then received at the smartphone. Since there is no guarantee of clock synchronism between the smartphone and the OBU, the latency is calculated based on round-trip time and then divided by 2, having an estimated latency. Figure 4.1 shows the mounted scenario to the latency tests. The numbers used to associate with the messages' steps are mentioned between parentheses.

To measure the latency in these communications, the mobile application sends periodic CAM messages with a frequency of 1 Hz to the MQTT Broker (1). Then, the server application receives the message from the broker (2) and sends an order to the RSU to disseminate this message (3). Finally, the message is received by an OBU, which represents the message destination (4). After receiving the message, the OBU sends it all the way back to the smartphone running the mobile application represented by the red arrows (5-8). When the mobile application receives the message, it compares the

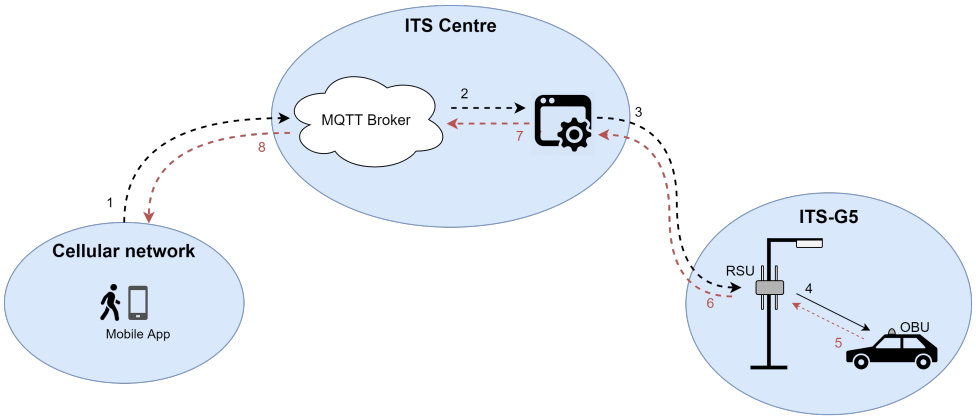


Figure 4.1: Hybrid latency measurements' scenario (extracted from [12]).

message-sending timestamp with the current message reception time and divides the result by two, thus obtaining the estimated latency. This architecture also applies in the opposite direction, where the messages are originally sent by the OBU to the mobile application and back to the OBU, allowing to understand if there are any differences when messages are originated on an OBU or smartphone. Note that the application used for latency measurements does not follow the architecture defined in section 3.2.

Using the scenario described, 2 tests of 10 minutes (around 1200 samples) were made where measurements were taken simultaneously on the OBU side and the smartphone side. There are 2 conditions that can be changed that originate different results: (i) the cellular network technology (3G, 4G and 5G) and (ii) the mobile network provider (Op. 1, Op. 2, and Op. 3). Changing these conditions, 9 combinations are available for testing.

The results are shown in Tables 4.1 and 4.2, where the column σ represents the standard deviation and all the values are represented in milliseconds.

Analysing the results obtained, there are some conclusions that can be drawn. Initially, when comparing the data obtained from the OBU and the smartphone, it is noticed an improvement on the OBU side, which should be explained by the processing in the equipment being different in these different scenarios, which may take longer on the smartphone.

In Tables 4.1 and 4.2, the results for 3G show a large discrepancy between the different providers. The one with the better results is Op.1 with latency values around 45 ms which represents a big difference when compared with the other two, which presents worse results with around 100 ms of latency, being Op. 2 the worse in terms of performance.

When comparing the different providers using 4G, the results are very similar between

Table 4.1: Latency measurements on Smartphone

Technology	Provider	Average [ms]	σ [ms]	Max [ms]	Min [ms]
3G	**Op. 1**	50	12	106	29
	Op. 2	139	110	489	43
	Op. 3	113	60	501	44
4G	**Op. 1**	41	10	91	21
	Op. 2	43	10	86	27
	Op. 3	49	12	86	24
5G	**Op. 1**	55	12	102	32
	Op. 2	-	-	-	-
	Op. 3	43	16	166	21

Table 4.2: Latency measurements on OBU

Technology	Provider	Average [ms]	σ [ms]	Max [ms]	Min [ms]
3G	**Op. 1**	39	7	90	28
	Op. 2	100	89	505	33
	Op. 3	86	64	655	36
4G	**Op. 1**	43	9	98	29
	Op. 2	40	6	87	30
	Op. 3	44	8	92	28
5G	**Op. 1**	46	14	100	28
	Op. 2	-	-	-	-
	Op. 3	42	18	206	28

them, showing that regardless of the cellular network in use, the latency results are around 42 ms.

The results obtained for 5G did not meet the expectations, since 5G promises lower latencies compared with 4G and that was not the case in the test scenarios, with similar results. It should also be noted that there was a slight increase in the standard deviation in comparison to 4G, showing instability in this new network generation. No values are presented for Op. 2 as there was no coverage in the area where the tests were carried out.

Interestingly, when using Op. 1 on the OBU side, 3G got better results than 4G unlike the other network providers, which can indicate that 3G from Op. 1 is still a reliable network for the solution.

4.2 Latency impact on Collision Warning Applications

To verify the impact of latency on the application's efficiency, the equation (4.1) was used. It is considered that latency does not have a critical impact on the application if the distance travelled during the latency time - d(lat), added to the braking distance - d(braking), is minor than the distance between the vehicle and the predicted collision location - d(vehicle-collision). This conclusion was reached since, as long as it is possible to avoid a collision by braking, there is a maximum permissible latency. Figure 4.2 demonstrates this situation.

$$d(lat) + d(braking) < d(vehicle - collision) \tag{4.1}$$

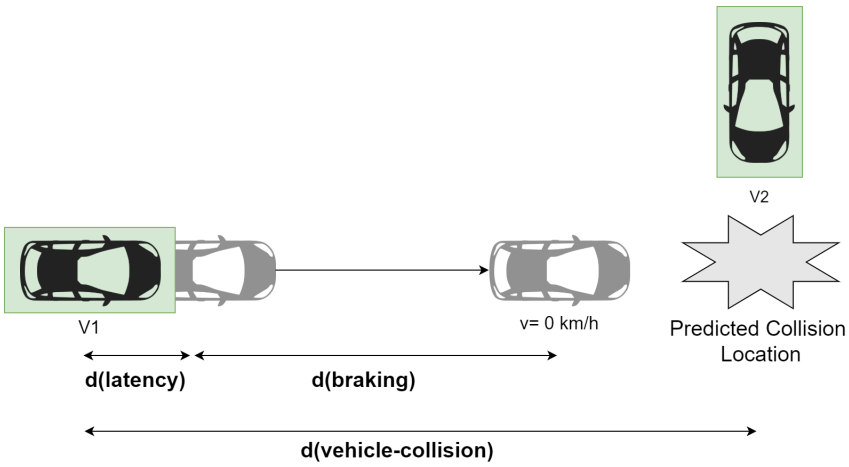


Figure 4.2: Latency impact on the collision warning application.

However, this analysis depends on specific scenarios involving the receiving vehicle's speed and the distance between the vehicle and the predicted collision location. Therefore, in Table 4.3, different scenarios were considered and the maximum latency was calculated for each scenario so that the algorithm can predict the collision while the braking is enough to avoid it.

According to AASHTO [24], the braking distance calculation depends on several factors, such as the driver's reaction time, coefficient friction and road slope. In this case, it is considered a situation of normal conditions where none of these factors has an unusual impact on the braking time and distance. Equation (4.2) was used to calculate the stopping distance considering the driver's reaction time.

$$d_s = (0.278 \times t_{pr} \times v) + v^2/(254 \times (f + G)) \tag{4.2}$$

The following information is used in the equation:

- d_s - stopping distance, in meters;
- t_{pr} - driver's reaction time, in seconds;
- v - vehicle speed, in km/h;
- G - Road inclination. It was assumed 0 inclination for the proposed scenarios;
- f - Coefficient of friction between the tires and the road. It was assumed 0,7 on a dry road.

Table 4.3: Maximum latency allowed for different scenarios

Speed [km/h]	**Braking distance [m]**	**Distance vehicle-collision [m]**	**Maximum Latency [ms]**
50	42	50	583
70	66	75	438
100	112	125	474
120	148	165	519

As shown in Table 4.3, the maximum latencies recorded in the table are higher than the latencies measured in Section 4.1. Considering that, at 120 km/h in 42 ms, a vehicle travels less than 1.5 meters, the receiving vehicle must be $d(braking) + 1.5$ meters from the predicted collision location for this latency to have a negative impact on the application's efficiency on avoiding the collision.

However, the presence of outliers in the latency can be worrying because in some cases, if the outliers occur in a critical moment, it can cause a false negative situation. In Section 4.3, the presence of outliers in the application will be tested.

4.3 Application Testing

After testing the hybrid environment and calculating the latency impact on collision warning applications, it is now possible to implement the architecture defined in Section 3.2. This architecture was used to create an application for OBUs and other for the smartphone. These applications allow the usage of a collision warning service for both connected and legacy vehicles, increasing the drivers' safety.

The implemented application for the OBU was created using the Software Development Kit (SDK) available for the equipment used (Unex OBU) [12], being developed in *C* language. For the smartphone application, *Python* was the language chosen using the *Kivy* framework [25] that allows the development of desktop and smartphone applications. The developed applications' code is available in [26].

Since the OBU doesn't allow a developed application visual representation, it only shows a command prompt alert, as shown in Figure 4.3.

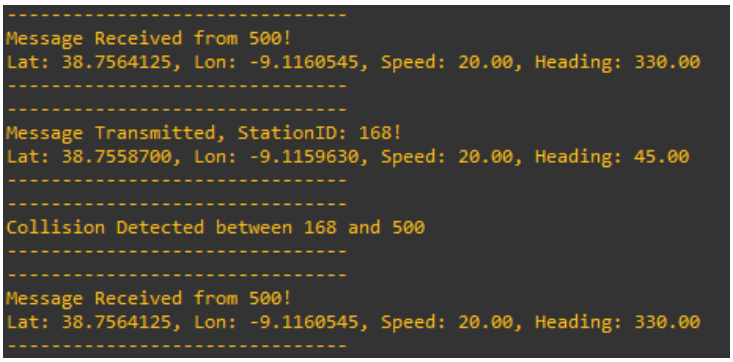


Figure 4.3: Detected collision in OBU application.

To integrate a visual or sound representation it is necessary to integrate another device that communicates with the OBU and is capable of a map and alert demonstration. That said, the application demonstration was made using the mobile application. To test the application operation, a digital twin was created to assist the tests made.

4.3.1 Digital Twin

As described by National Aeronautics and Space Administration (NASA), "A digital twin is an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin." [27]. Thus, a digital twin was created to simulate a vehicle in specific scenarios that communicate with the smartphone application.

In this work, the digital twin is useful to simulate the presence of a vehicle in a specific position with specific characteristics. Since it is only used to simulate these aspects, it was created a trajectory, shown in Figure 4.4, that the digital twin simulates and sends CAM messages at a frequency of 1 Hz.

To demonstrate the algorithm operation, Figure 4.5 shows the detection of a collision when the smartphone is on a collision course with the digital twin (represented by the black car logo), causing a yellow warning.

Figure 4.4: Vehicle Digital twin trajectory.

Figure 4.5: Collision warning application detecting a possible collision.

Although the application does not accurately represent the vehicles' heading involved in these tests, Figure 4.6 shows a better situation visualisation where it is possible to observe both vehicles' heading represented by the arrows that will collide in the collision point, if the warning is ignored.

4.3.2 Latency Outliers Impact

The usage of a digital twin can also be useful to test situations of repeated messages with high latency. The presence of outliers in the latency can be worrying in certain situations since the receiving vehicle receives outdated information about another vehicle's position. However, this situation becomes more dangerous if outliers are repeated, causing a period where a vehicle has wrong information about another vehicle.

Figure 4.6: Collision warning application detecting a collision (visual assist).

This situation was also tested.

Figures 4.7 and 4.8 demonstrate the situation which shows the impact of repeated high latency messages in the algorithm. In Figure 4.7 a), two vehicles are at a distance of 50 meters of the collision point driving at the same speed, meaning that if V1, have the correct information about V2, the collision is detected, as shown in Figure 4.7 b). This behaviour is verified when messages are received with low latency.

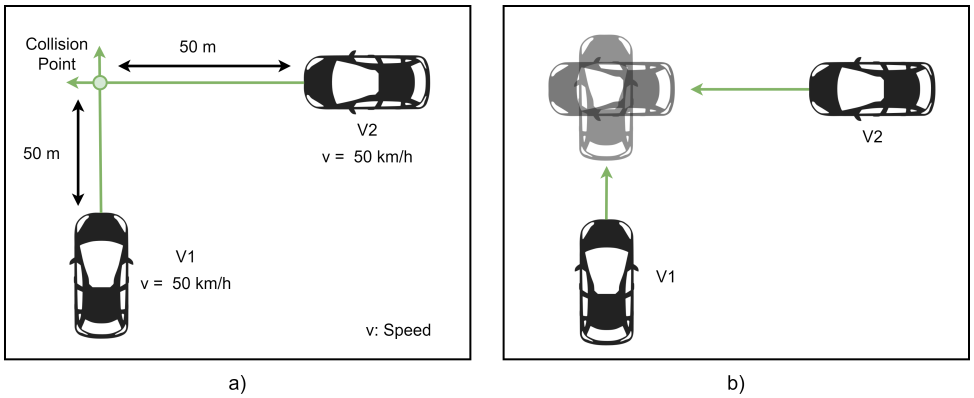


Figure 4.7: Low latency impact on the application.

However, if messages from V2 are received by V1 with a latency of 500 ms, V1 has a scenario's different perspective compared with the reality, as shown in Figure 4.8 a) where, according to V1's perspective, when it receives the message from V2, V1's distance to the collision point is 43 m while from V1's perspective, V2 is at a distance

of 50 m. The algorithm analysing this situation considers that it won't originate a collision, as shown in Figure 4.8 b), creating a situation of collision where a warning won't be issued.

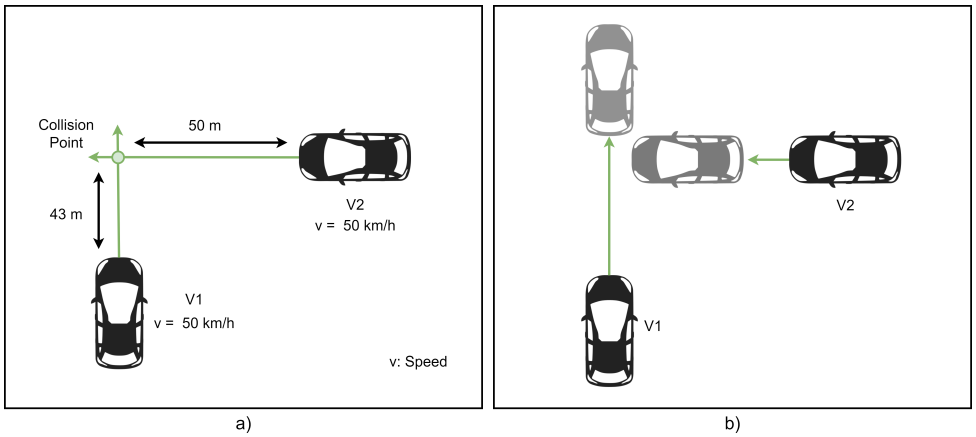


Figure 4.8: Demonstration of high latency impact on the application.

Figure 4.9 shows the simulated scenario described before using two digital twins (vehicles B and C) and the smartphone application. Vehicle C is sending messages with 500 ms of constant latency. This value of latency was chosen based on the worst maximum latency measure of the hybrid environment.

Figure 4.9: High latency impact on the application.

In this situation, all three vehicles involved are at 50 m of the collision point and travelling at a constant speed of 50 km/h, creating a situation of collision between them. However, the warning is only shown for a collision with vehicle B, as shown by the yellow warning in the left corner (if a collision with vehicle C were detected, a warning on the right side would be shown). As observed before, this happens because when vehicle A receives the message from vehicle C, vehicle A considers that vehicle C is 7 meters behind than it really is, creating an offset between the real position of vehicle C and the information received by vehicle A. When the algorithm receives outdated information, it considers that a collision won't occur, since vehicle A gets to the collision point before vehicle C, creating a false negative situation. Note that this situation is only critical if messages with high latency are received consecutively, otherwise, if only one message is received with high latency, upon receipt of another message with low latency, the algorithm will make a correct assessment of the situation and the collision will be detected.

5

Conclusions

5.1 Developed Work

The thesis objective was to develop and assess an application capable of assisting drivers with their task, increasing their experience safety. This application uses information of neighbouring vehicles to predict possible collision situations and, when predicted, warn the driver about the danger.

Firstly, a simulation application was developed with the objective of testing different approaches for the application operation. Then, the application architecture was designed based on the simulation application, however, problems with the application efficiency were detected and the architecture needed alterations to overcome this problem.

The application was designed to communicate via ITS-G5 with neighbouring vehicles but, to allow the application to be used by every road user, the application was also implemented in a smartphone to allow users that don't have access to an OBU, the possibility of improving their driving experience. To allow this type of communication, a hybrid environment between ITS-G5 and cellular networks (3G, 4G or 5G) was installed and tested.

The algorithm used by the application was inspired by the works developed by Ahrems [18] and Djamel [21], where safety zones are used to create a representation of the involved vehicles and detect collisions based on the safety zones overlapping.

5.2 Results

After developing the hybrid environment, latency measurements were taken to evaluate the environment's efficiency. After tests were done using three different cellular network providers and three different technologies, it was concluded that, nowadays, the 4G is the most reliable solution with an average latency of 42 ms. 5G despite having a similar latency compared with 4G, the standard deviation presented higher results, showing some instability in the network. Finally, overall, the 3G network showed the worst results when compared with the other two technologies.

These measurements allowed the study of the impact of latency in a collision warning application, demonstrating that the impact varies based on vehicles' speed and distance to the collision point, but, using the average results obtained, the impact of latency is low compared with the maximum permissible latency obtained for the evaluated scenarios.

Finally, the developed applications were tested using a digital twin that allowed the testing of specific scenarios, in particular the impact of regular high-latency messages in the algorithm. It was concluded that, when consecutive, messages with high latency can create false negative situations since the information that the vehicle running the application has is quite different when compared with reality. This problem creates an offset between the vehicle's real position and the position assumed by the application, causing situations when it considers that there is no risk of collision, but the collision situation is in fact present.

5.3 Future work

In the future, the application can be improved to detect some specific issues that couldn't be easily detected. Besides that, there are some improvements that can be made to improve the application efficiency, such as:

- When defining the safety zone size, instead of pre-defining the measures, make regular adjustments according to the vehicle speed in order to adapt the safety zone according to the danger associated with the increase in speed;
- The integration of maps in the algorithm to identify the scenario in which the vehicle is inserted, allows a better situation analysis. For example, in a scenario where there are two roads close together but which never cross, we have the

possibility of not considering vehicles that are not on the same road, optimizing the algorithm.

- Since the GPS has an inaccuracy in the location, the algorithm is not able to distinguish different lanes with a vehicle coming in the opposite direction. Therefore, optimization in the algorithm ignores these situations in which the vehicle direction is opposite.
- When using the OBU application, the implementation of a visual representation of the algorithm could be an important improvement to the collision warning application since it would allow a visual neighbouring representation and a visual warning to drivers.
- Finally, when using the OBU's application, the integration of communication with CAN bus allows the usage of the steering wheel angle in the movement prediction calculation, increasing the algorithm accuracy.

References

[1] Eurostat, *Road accidents: Number of fatalities continues falling*, Last accessed 21 June 2022, 2021. [Online]. Available: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20210624-1.

[2] Car 2 Car Communication Consortium, *Glossary*, Last accessed 01 September 2022. [Online]. Available: https://www.car-2-car.org/about-c-its/c-its-glossary/.

[3] ETSI EN 302-663 (V1.3.1), "Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band", ETSI, Standard, Jan. 2020.

[4] Andreas Festag, "Cooperative intelligent transport systems standards in europe", *IEEE Communications Magazine*, vol. 52, no. 12, pages 166–172, 2014. DOI: 10.1109/MCOM.2014.6979970.

[5] ETSI EN 302 636-4-1 (V1.4.1), "Intelligent transport systems (its); vehicular communications; geonetworking; part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; sub-part 1: Media-independent functionality", ETSI, Standard, Jan. 2020.

[6] ETSI EN 302 636-5-1 (V2.2.1), "Intelligent transport systems (its); vehicular communications; geonetworking; part 5: Transport protocols; sub-part 1: Basic transport protocol", ETSI, Standard, May 2019.

[7] ETSI EN 302 637-3 (V1.3.1), "Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service", ETSI, Standard, Apr. 2019.

[8] ETSI EN 302 637-2 (V1.4.1), "Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service", ETSI, Standard, Apr. 2019.

[9] ETSI TS 101 556-1 (V1.1.1), "Intelligent transport systems (its); infrastructure to vehicle communication; electric vehicle charging spot notification specification", ETSI, Standard, Jul. 2012.

[10] ETSI TS 103 301 (V1.3.1), "Intelligent transport systems (its); vehicular communications; basic set of applications; facilities layer protocols and communication requirements for infrastructure services", ETSI, Standard, Feb. 2020.

[11] ETSI TR 103 562 (V2.1.1), "Intelligent transport systems (its); vehicular communications; basic set of applications; analysis of the collective perception service (cps); release 2", ETSI, Standard, Dec. 2019.

[12] Unex, *Obu-301e information sheet*, Last accessed 01 September 2022. [Online]. Available: https://www.unex.com.tw/obu-301e-sheet/.

[13] Siemens, "Connected vehicle roadside unit (rsu)", Siemens, Standard, 2018.

[14] MQTT, *Mqtt: The standard for iot messaging*, Last accessed 04 September 2022, 2021. [Online]. Available: https://mqtt.org/.

[15] Hexin Lv, Ping Xu, Huafeng Chen, Binbin Zhou, Tiaojuan Ren, and Yourong Chen, "A novel rear-end collision warning algorithm in vanet", in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, 2016, pages 539–542. DOI: 10.1109/ICCSN.2016.7586581.

[16] Xuehai Xiang, Wenhu Qin, and Binfu Xiang, "Research on a dsrc-based rear-end collision warning model", *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pages 1054–1065, 2014. DOI: 10.1109/TITS.2013.2293771.

[17] O. Ararat, E. Kural, and B.A. Guvenc, "Development of a collision warning system for adaptive cruise control vehicles using a comparison analysis of recent algorithms", in *2006 IEEE Intelligent Vehicles Symposium*, 2006, pages 194–199. DOI: 10.1109/IVS.2006.1689627.

[18] Janeks Ahrems, "Collision warning algorithm for passage of an uncontrolled road intersection", in *2014 14th Biennial Baltic Electronic Conference (BEC)*, 2014, pages 49–52. DOI: 10.1109/BEC.2014.7320553.

[19] Raja Sengupta, Shahram Rezaei, Steven E. Shladover, Delphine Cody, Susan Dickey, and Hariharan Krishnan, "Cooperative collision warning systems: Concept definition and experimental implementation", *Journal of Intelligent Transportation Systems*, vol. 11, no. 3, pages 143–155, 2007. DOI: 10.1080/15472450701410452. eprint: https://doi.org/10.1080/15472450701410452. [Online]. Available: https://doi.org/10.1080/15472450701410452.

[20] D. Anurag, Srideep Ghosh, and Somprakash Bandyopadhyay, "Gps based vehicular collision warning system using ieee 802.15.4 mac/phy standard", in *2008 8th International Conference on ITS Telecommunications*, 2008, pages 154–159. DOI: 10.1109/ITST.2008.4740247.

[21] Bektache Djamel, Ghoualmi-Zine Nacira, and Tolba Cherif, "Forecasting approach in vanet based on vehicle collision alert", in *2012 International Conference on Multimedia Computing and Systems*, 2012, pages 573–577. DOI: 10.1109/ICMCS.2012.6320231.

[22] SunEarthTools, *Distance*, Last accessed 25 September 2022. [Online]. Available: https://www.sunearthtools.com/tools/distance.php.

[23] C-Roads, "Radio frequencies designated for enhanced road safety in europe - c-roads position on the usage of the 5.9 ghz band", *C-Roads Platform Position Paper*, 2017.

[24] American Association of State Highway and Transportation Officials, *A Policy on Geometric Design of Highways and Streets*. AASHTO, 1994.

[25] Kivy, *Welcome to kivy*, Last accessed 5 September 2022. [Online]. Available: https://kivy.org/doc/stable/.

[26] Diogo Salgado, *Collision warning application*, 2022. [Online]. Available: https://github.com/DiogoSalgado/CollisionWarningApp.

[27] Mike Shafto Chair, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang, *DRAFT Modeling, Simulation, Information Technology & Processing Roa*. National Aeronautics and Space Administration, 2010.