Implementation of Traffic Control With Heavy Duty Vehicle Anti-Platooning

JOÃO PEDRO ALVITO

KTH Electrical Engineering

Master's Degree Project

Stockholm, Sweden June 2013

TRITA-XR-EE-RT 2013:009

Dedicated to everyone that supported me in one way or another...

Acknowledgments

I started this journey 5 years ago. Looking back, I would only dream to be in this position where I find myself today. I started my studies in Portugal, in Instituto Superior Técnico, where I attended some courses with demanding professors, but that today I am very thankful of. Because they are part of the foundations of the engineer that I soon will become.

More recently I started to study at KTH, that due to its organization and innovative work platform allowed me to evolve as an engineer. I would like to thank my supervisor Karl Henrik Johansson, who with his insightful suggestions and keen eye, helped me a great deal. Jonas Mårtenson, my thesis coordinator, that with his support and guidance, helped to shape my final project.

I would like to especially acknowledge the help of my work colleague and dear friend, Pedro Lima. His help was essential to the development of my work. Together we created, in my opinion, a very good work platform that can become very useful in the future of the Smart Mobility Lab. He has been my colleague since the first years of university, and with his support and companionship, he is, without doubt, one of the biggest influences that I will have as an engineer.

Last but not least, I could not fail to express my heartfelt thanks to everybody that supported me through this journey, specially to my parents, family and closest friends.

Abstract

In the north of Sweden, there are some wealthy mines that have being actively explored. To transport the metals from the mines to its destination there are being used Heavy Duty Vehicles (HDV). This setup has two main restrictions: the HDVs must travel 4 minutes apart; and due to the roughness of the terrain there are some areas where only one vehicle can go through at a time.

The project was developed in the Smart Mobility Lab (SML), in KTH, Stockholm. It was created a test platform that simulates the behavior of HDVs in scaled trucks. The scaled trucks are controlled with PID controllers to follow paths in a road network that was also created during this project and also keep a formation with a fixed distance between vehicles. To track the position of the trucks it was used a motion capture system. Basically, it was created a system that allows to test several features of platooning and at the same time apply traffic control with virtual traffic lights. On top of the system it was applied an instantaneous fuel consumption model that makes possible a fuel consumption analysis. Several scenarios were designed in order to showcase some features of the system and at the same time analyze its benefits.

In the end of the thesis it was possible to see the benefits of using traffic control in the situation described. Also a big contribution of the thesis was the construction of the testbed that has been proven to be an effective showcase tool of platooning techniques and simple traffic control. The application developed was created in a generic way in order to serve as a base resource of knowledge for future works.

Keywords: Traffic Lights, Anti-Platooning

Resumo

No norte da Suécia existem prolíferas minas que têm vindo a ser activamente exploradas. Para transportar os metais das minas para o seu destino estão a ser usados hoje em dia veículos pesados. Este sistema tem duas restrições: os veículos necessitam de se movimentar separados por 4 minutos; e devido a limitacões no desenho das estradas em certas zonas apenas é possível a passagem de um veículo de cada vez.

O projecto foi desenvolvido no Smart Mobility Lab (KTH, Estocolmo). Foi criado uma plataforma de teste que permite simular o desempenho de veículos pesados em camiões à escala. Estes camiões são controlados com controladores PID de forma a seguir trajectórias numa rede de estradas, que foi também criada durante a execução deste projecto. O sistema tenta também manter uma formação com uma distância fixa entre os veículos. Para seguir a trajectória dos camiões à escala foi usado um sistema de motion capture. Basicamente foi criado um sistema que permite o teste de várias funcionalidades relacionadas com platooning e ao mesmo tempo aplicar controlo de tráfego com semáforos virtuais. Foi também aplicado ao sistema um modelo de consumo instantâneo de combustivel. Diversos cenários foram desenhados de forma a demonstrar algumas das funcionalidades do sistema e ao mesmo tempo analisar os seus benefícios.

No fim desta tese é possível ver os benefícios de usar controlo de tráfego na situação descrita. Uma grande contribuição deste projecto foi a construção da plataforma de testes que tem provado ser uma ferramenta de demonstração bastante eficiente das várias técnicas de platooning e de simples controlo de tráfego. A aplicação foi desenvolvida de uma forma geral de modo a servir como base de criação para futuros projectos.

Palavras-chave: Semáforos, Anti-Platooning,...

Contents

Chapter 1

Introduction

"The real voyage of discovery consists not in seeking new lands but seeing with new eyes."

Marcel Proust

We depend heavily on transportation systems in our everyday lives. Yet, continuously increasing road traffic generates serious problems in terms of congestion which imply a higher fuel consumption and of course an increase in the emission of harmful gases.

1.1 Cooperative Vehicles

Figure 1.1: Cooperative vehicles (courtesy of COACT - Grand Cooperative Driving Challenge).

Over the recent years with the technological evolution, the research in intelligent vehicles has turned to cooperative vehicles. The vehicles communicate with each other (vehicle-to-vehicle communication) and with the infrastructure (vehicle-to-infrastructure communication). The integrated system of communication significantly increases the reliability of the information relative to the vehicles, this way it is possible to use approaches that depend in this information. These approaches will lead to new services like platooning that will, in the future, bring social and economic benefits and of course transport benefits and safety. In Figure 1.1 it is possible to see an example of a setup where the vehicles are communication with each other.

1.2 Platooning

Vehicle platooning is one of the innovations in the automotive industry that aim to improve the safety, efficiency, and time of travel of vehicles while relieving traffic congestion, decreasing pollution and fuel consumption.

Figure 1.2: Formation of a platoon with bicycles (courtesy of Team Sky).

One easy way to understand the concept of platooning is resorting to the example of bicycle runners (Figure 1.2). If you have seen a bicycle race you have noticed that usually the bikers ride in a formation, where they are in placed in a line behind each other. This formation is in fact a method to reduced the effort of each of the bikers, because only the front rider has to fight against the totality of the air resistance. The riders behind the leader have what we call a smaller air drag. This concept also applies in other vehicles, but noticed that in this case the air drag will be much more important due to the bigger frontal area of the trucks.

Figure 1.3: Usual formation of a platoon (courtesy of Scania).

It has also been observed that vehicle platooning significantly reduces the drag that each vehicle experiences. This reduction of air drag translates into less fuel consumption, greater fuel efficiency and less pollution.

This system is based in automated driving which brings both positive and negative issues. First since there is not a human driver the driving is much more smoother increasing the fuel efficiency and also the reaction time of the vehicles is much faster. Nevertheless in order to form a platoon capable of fuel savings the vehicles have to maintain a small gap between each other which brings some problems because this situation is technologically feasible but break some current security rules.

1.3 Intelligent Transport Systems

Intelligent Transport Systems (ITS) consist in a variety of communications-related applications that have as end goal an improvement in travel safety, impact in the environment, traffic control and benefits of transportation to commercial users and general public.

In Figure 1.4 it is possible to see an illustration of some of the services that can be used in the transportation system.

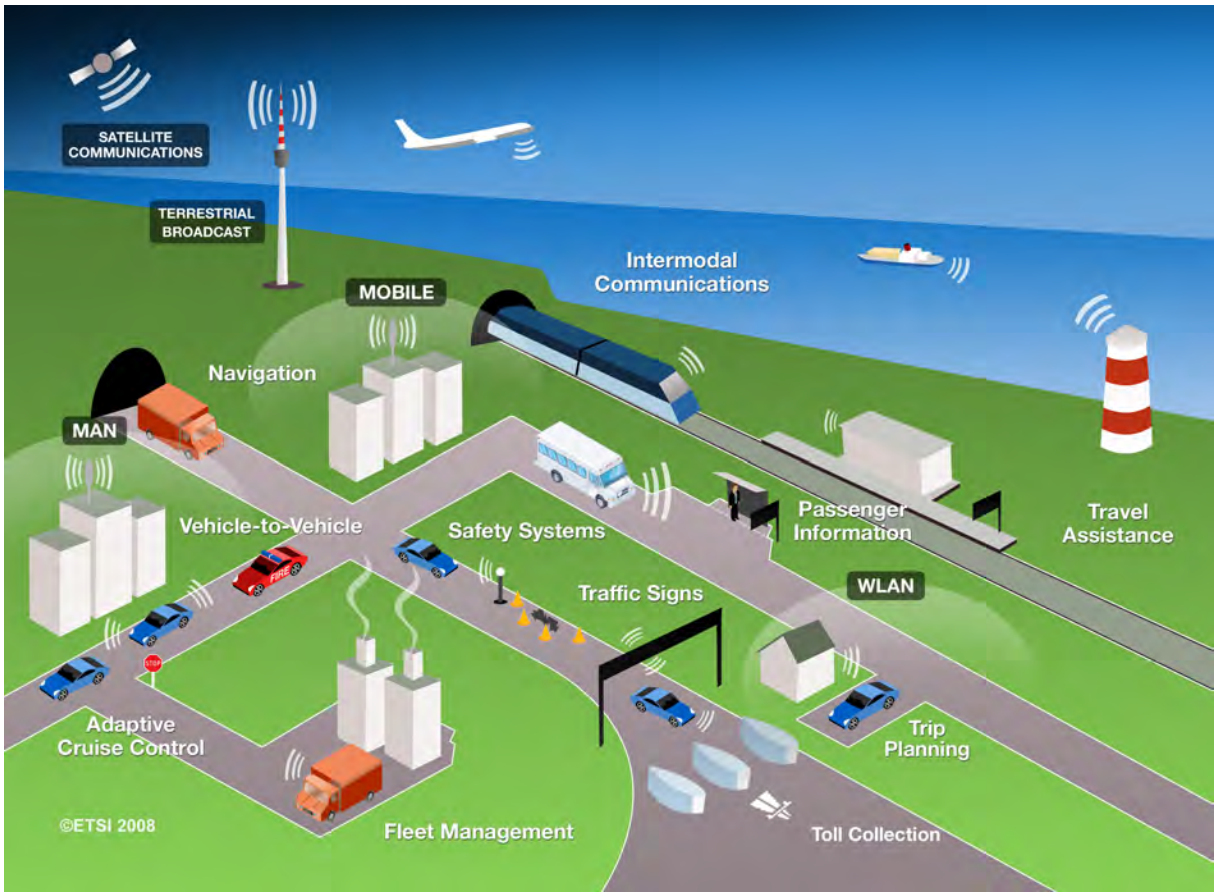


Figure 1.4: Intelligent transport systems (courtesy of ETSI).

1.4 Ore Transport

In the north of Sweden, there are some wealthy mines that have being actively explored. When the metals are extracted they need to be relocated to different places all over Sweden, and for that purpose the railways are used. Nevertheless there is still the issue of transporting the metals from the mines to the railways, currently for that are used Heavy Duty Vehicles (HDV) (like the one in Figure 1.5) that accomplish that task.

Figure 1.5: Example of the real HDV that will be used (courtesy of Scania).

Because the mines are very prolific there is a great affluence of trucks in the roads that connect the mines and railways, more precisely there is a truck ready to leave at least every 6 minutes continuously. This causes several problems because the trucks that are used are over the legal weight limit, but due to an agreement with the authorities this kind of trucks are allowed in those specific roads, which can cause big damages to them over time. After some studies we know that the time gap between each truck should be no less than 4 minutes in order to let the road re-establish itself.

Figure 1.6: Another point of the real HDV that will be used (courtesy of Scania).

Furthermore in some regions the roads are very narrow and in some cases there are also narrow bridges, so the idea is to create a control algorithm that allows the trucks to travel at least 4 minutes apart but that also avoid stops in narrow ways. This is important because stoppages are very energy inefficient, so if we can avoid them it will save, in long term, a considerable amount of energy and of course money.

The separation distance that the HDVs must maintain between each other is called the anti-platooning distance. Anti-platooning is a special form of platooning, i.e, vehicles keep a formation, where they maintain a fixed distance between each other.

So summing up, we have two different problems that must be approached. First we need to have a system that creates the anti-platooning scenario and second we must study the situations where the traffic control should be implemented.

Chapter 2

Background

2.1 Previous Work in Subjects Related to the Thesis

Next there are presented the main sources of information that were used in this project.

2.1.1 Fuel-Efficient Distributed Control

This thesis [4] presents contributions to a framework for the design and implementation of HDV platooning. The focus lies mainly on establishing and validating real constraints for fuel optimal control for platooning vehicles. Nonlinear and linear vehicle models are presented together with a system architecture, which divides the complex problem into manageable subsystems. The fuel reduction potential is investigated through simulation models and experimental results derived from standard vehicles traveling on a Swedish highway. It is shown through analytical and experimental results that it is favorable with respect to the fuel consumption to operate the vehicles at a much shorter intermediate spacing than what is currently done in commercially available systems.

2.1.2 Estimation of Fuel Consumption for Real Time Implementation

The requirements of a fuel consumption model for trucks is studied in this thesis [11]. Two estimation methods were evaluated and compared, one of the methods was estimating fuel consumption using look up tables consisting fuel consumption data. The other method was estimation doing real time calculations with a fuel consumption model. With a simplified real time model the modelling error was approximately lower than 5% for rural and highway driving, however for city driving the accuracy was significantly reduced.

2.2 Previous Work in SML

In an 8 week course at KTH, The Royal Institute of Technology, Stockholm, it was built a loading dock of the future. Wireless controlled trucks and a quadrocopter as well as a stationary tower crane were used. The three processes are controlled with PID controllers and an MPC controller and they communicate with each other via a messaging scheme designed. They receive location information from an infrared-camera motion capturing system. This project was a great inspiration and a great source of practical information for this master thesis.

2.3 Previous Work in Industry

Currently, Scania is one of the biggest companies investing worldwide to improve the platooning systems implemented in the trucks. And example can be seen in Figure 2.1.

Figure 2.1: Scania platooning (courtesy of Scania).

Chapter 3

Objectives

3.1 Objectives

The final goal of this thesis is divided in two main sub goals, where the first goal is inherently dependent on the success of the second goal.

The first main goal is to develop a testbed that enables experiments with scaled trucks that emulate real HDVs. This is an ambitious task because it entails several different systems that must be integrated in order to obtain a final system that is robust and that can be used as a test platform for other future projects.

The second main goal is to study the behavior of a HDV in situations of anti-platooning formations and when traffic control is applied to the convoys in order to avoid HDV's costly stoppages. Here it his possible to create theoretical scenarios that we want to see implemented in a real system. So, to perform some kind of analysis we need to implement in the system the desired scenarios.

3.2 Problem Definition

This project intends to recreate in a smaller scale the scenario of the ore transportation in the north of Sweden. For that the system that will be implemented shows small features that are proof for small concepts, that when combined represent the complete scenario. There are two main features that we need to simulate: platooning, with more detail to the anti-platooning scenario; traffic lights. So basically the end goal is to evaluate the fuel consumption in the different scenarios related to the system that controls the traffic lights status and also implementing the platooning feature in the simulation system in the Smart Mobility Lab. The project intends to simulate the behavior of the trucks in the real system and draw some conclusions that might be applied in the real HDVs.

3.3 Thesis Contributions

Hopefully, this thesis has created an implementation that can showcase the problem described. The most important contribution that this project leaves behind is, without a doubt, the developed testbed. It was, not only essential for the final results obtained in the thesis, but it was as well developed in such a generic way, that it can be the base system for different projects. An example of that could even be seen during the thesis creation were the system created was the base for two distinctive master thesis.

Next, it is listed with more detail the thesis' contributions.

- Active influence in the early days of the Smart Mobility Lab (SML) in the department of Automatic Control from KTH, Stockholm.
- Implementation of the Motion Capture system in controllable scaled trucks.
- Creation of a road network installed in the SML.
- Design, creation and implementation of a LabVIEW application that allows to run the scaled trucks in the road network, and at the same time it is possible show some platoon and anti-platoon scenarios and as well small platoon features (some of them not even mentioned in this report because they go out of the scope of this thesis).

3.4 Thesis Outline

Chapter 1

A brief introduction to cooperative vehicles, platooning and ore transportation.

Chapter 2

Chapter 2 briefly resumes the research done in the Master Thesis Project. It lists the previous work that is related with the thesis' subjects. Also it shows what has been previously done in the Smart Mobility Lab and what was the current status of the work platform in the first days of work. Finally it shows some real world examples of what has been done in the industry.

Chapter 3

This Chapter presents the main purpose of the thesis, explaining what are its motivations and main goals. The problem definition describes the issue that was approach and makes the relation between the real world problem and what was indeed applied in the experiments.

Chapter 4

This Chapter explains in great detail the Smart Mobility Lab setup analyzing each one of its components.

Chapter 5

In Chapter 5 is presented all the theoretical knowledge that was used in the implementation of the experiments. It explains in detail the model of the truck that was used to better understand the truck's movement. Also it shows all the reasoning in the controllers design with a brief explanation of a generic PID controller. Lastly it is presented as well the instantaneous fuel consumption model, and all the simplifications that might be done to simplify the model.

Chapter 6

The Chapter 6 is where the implementation is explained, here it is shown the performance of the system. Also there is a brief note about the calibration of the system and some decisions that were made related to the theoretical controllers previously designed.

Chapter 7

This Chapter is a description for each scenario that has been created to showcase selected features of the system. There is for each scenario a brief description of the sequence of actions, a comic strip that visually helps to understand the scenario and also what is the main goal of the scenario. The chapter also presents some results from the implementation of the scenarios previously described. The results consist in velocity, fuel consumption and platooning distances analyzes.

Chapter 8

Lastly Chapter 8 contains the conclusions that can be taken from the work done in the thesis, and also presents some of the ideas that are to be implemented in the future to improve what was created during this project.

Chapter 4

SML Setup

All the experiments realized were done in a testbed idealized by the Automatic Control department of KTH. The testbed consists of several different sub-systems that when together make the perfect configuration to control, monitor and track the trucks. This system can be used to any other object other than trucks, for example in the SML there are being done simultaneously tests with quadrcopters. The testbed that we used, as we can see in Figure 4.1, is an intercommunication between several sub-systems: Camera setup, Trucks, Markers, Communication System. Note that any question regarding the utilization can be further answered in Appendix A

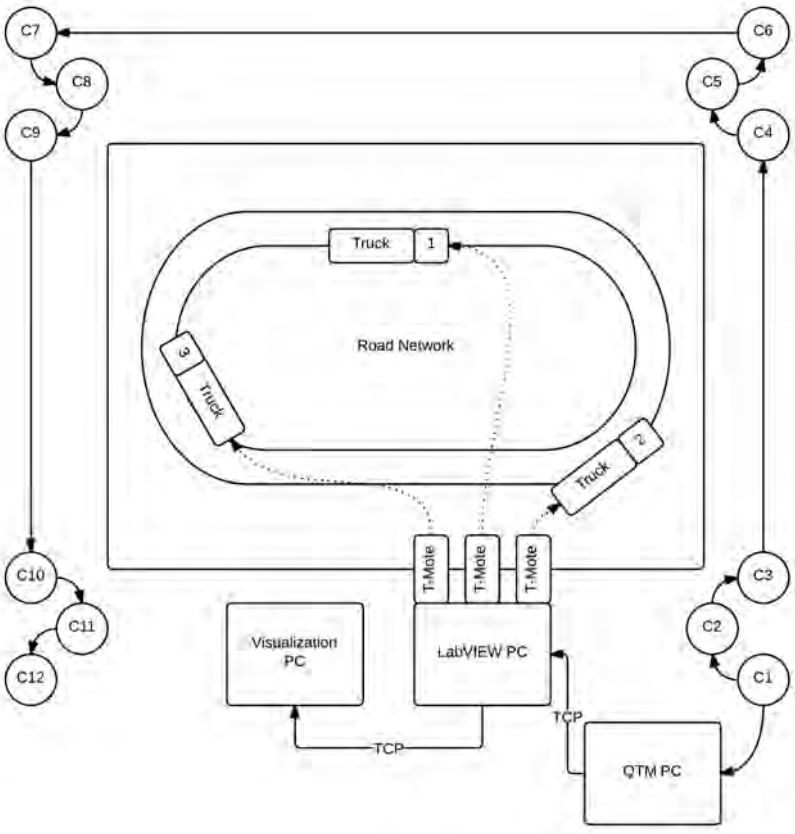


Figure 4.1: Testbed.

Figure 4.2 is an image captured from the Qualisys system that overlays the video of one of the cameras and all the system axes (the origin axis and the local axis for each truck).

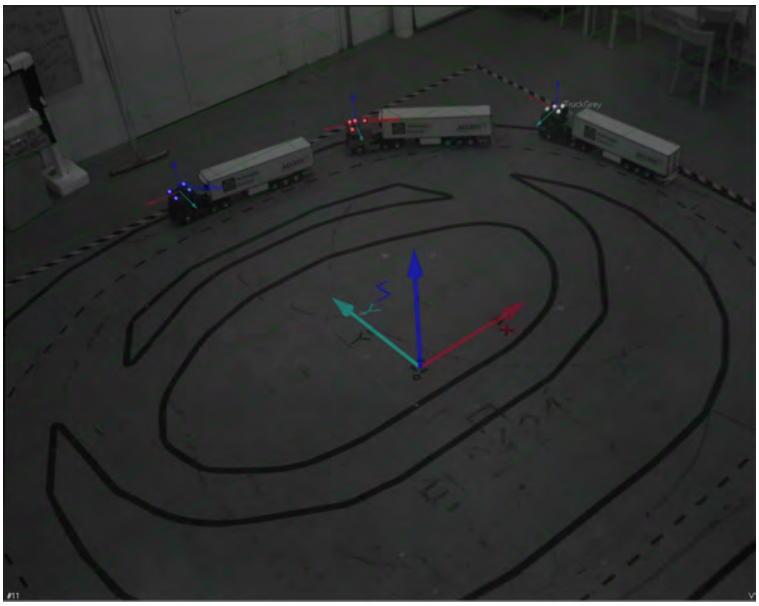


Figure 4.2: Testbed from the qualisys system point of view.

Here in Figure 4.3 you can see an overview of the system. It is possible to view the road network with its different roads and lanes. This will be further explained in section 4.5.

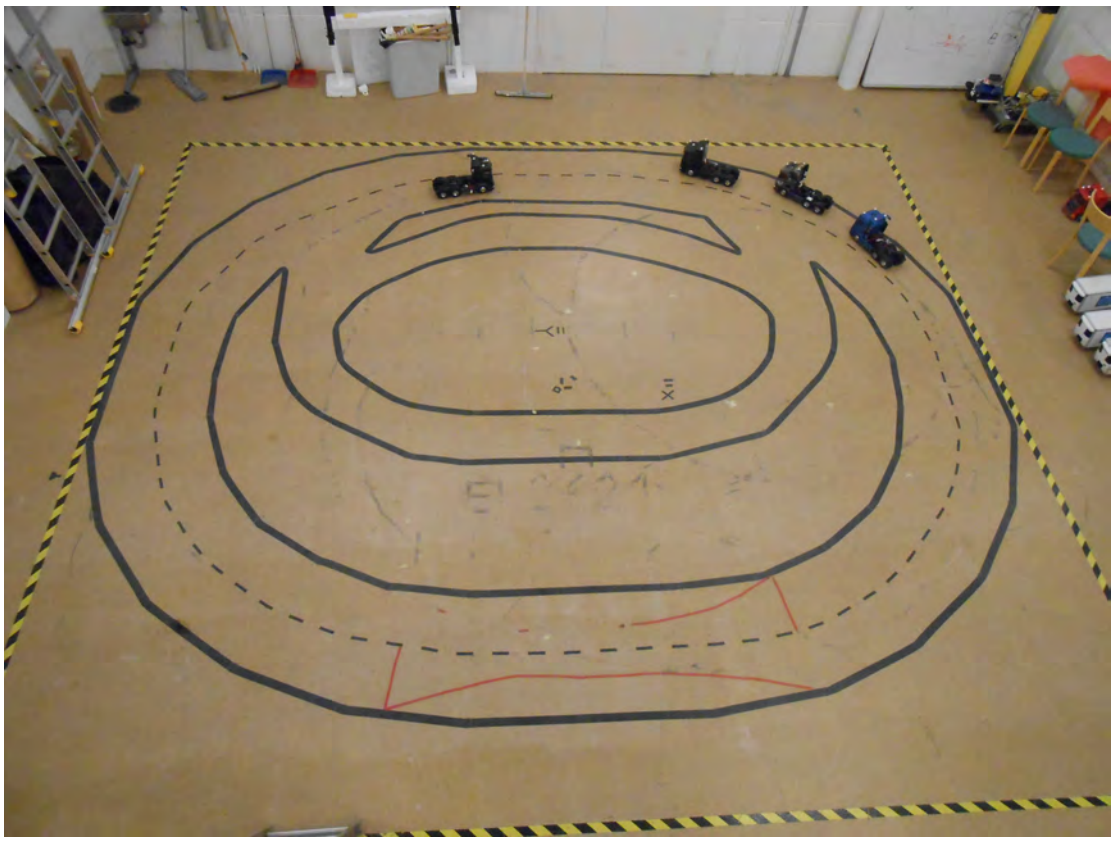


Figure 4.3: Roads network overview.

4.1 Camera Setup

The camera setup installed in the SML's testbed consists of 12 Oqus cameras. The Oqus cameras are specially developed for motion capture systems. Basically they use the infra-red markers positioned in the object that you want to track, capture the reflection and use this information to calculate the exact position of each marker. Combining the information of every camera in the system, it is possible to use triangulation in order to obtain the exact position of the marker. An example of the Oqus camera is shown in Figure 4.4.

Figure 4.4: Example of a qualiys camera (courtesy of Qualisys).

As seen in Figure 4.1 the 12 cameras are scattered along the 4 corners of the building an example of the cameras can also be seen in Figure 4.5. This disposition was stetted in such a way that all the ground area of the building is equally visible, and if possible the same point in space is seen from multiple angles. This increases the robustness of the system making the tracking of the trucks a very precise and robust measure. This disposition and effectiveness creates a system that we can compare to a GPS system, in this case we can say the we have some sort of an ”indoor GPS”. The camera system is also referred as MoCap system.

Figure 4.5: Camera setup.

Once the system has the exact position of every marker in the visible area, we can create a rigid body that can be tracked in the Qualisys system. As we can see in Figure 4.6, we can create an unique configuration of markers (further explain in section 4.3). Then the information about the rigid bodies (x,y,z, and euler angles) is sent to the main computer, where it can be use to control the trucks. Note that this subsystem is completely independent and does all the necessary processing individually and then sends the relevant information to the other parties of the system.

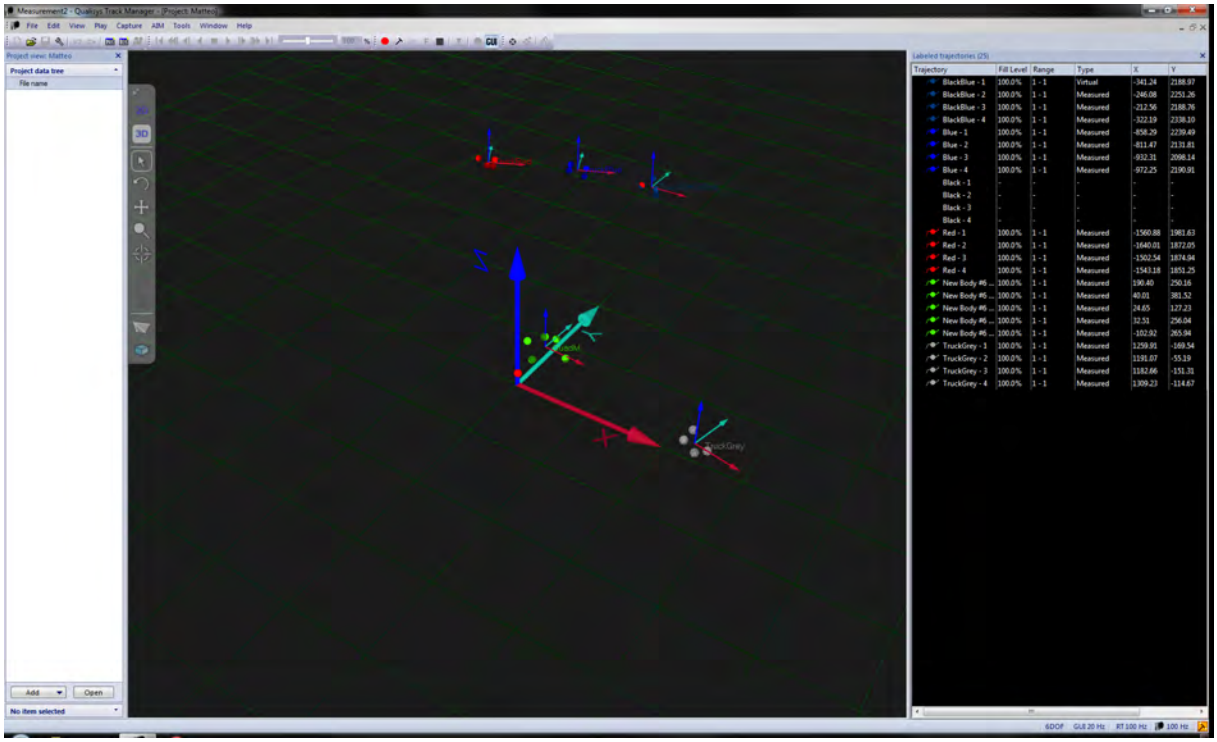


Figure 4.6: Rigid body.

Summing up, this system is a very robust and effective tracking system that was used and gave very accurate measures of the trucks' positioning. Note that a previous setup that was used, had only 4 cameras which created a very unreliable system that was usable nevertheless, but that was prone to errors. So this configuration proved to be the more effective and reliable one.

4.2 Trucks

The trucks used were the Tamiya Scania trucks that we can see in Figure 4.7.

Figure 4.7: One of the trucks used.

The trucks are scaled 1:14 and they are simple RC trucks that with a few modifications can be controlled from a computer through a LabVIEW program. Even though the trucks are very simple and were not built with the specific propose of technical experiments, they turned out to be very robust and effective. The only downside is that some of the modifications were handmade by students, so sometimes the connections between elements may fail. The trucks can be run with trailers our without them, during experiments the trailers tend to drain the battery quicker but with them the setup becomes more realistic.

These modifications use different hardware elements, therefore next is presented a list of said elements. The behavior of those elements is further explained in the communication section 4.4. The numbers in parenthesis relate to the numbers in Figure 4.8.

- TAMIYA Scania R620 Highline model truck with trailer.
- T-mote sky board (2).
- Pololu servo controller board (1).
- SAMARA ”Super-Sun-Power” 500 Ni-MH Battery (7.2V 6N 5000 mAh 36Wh) (4).

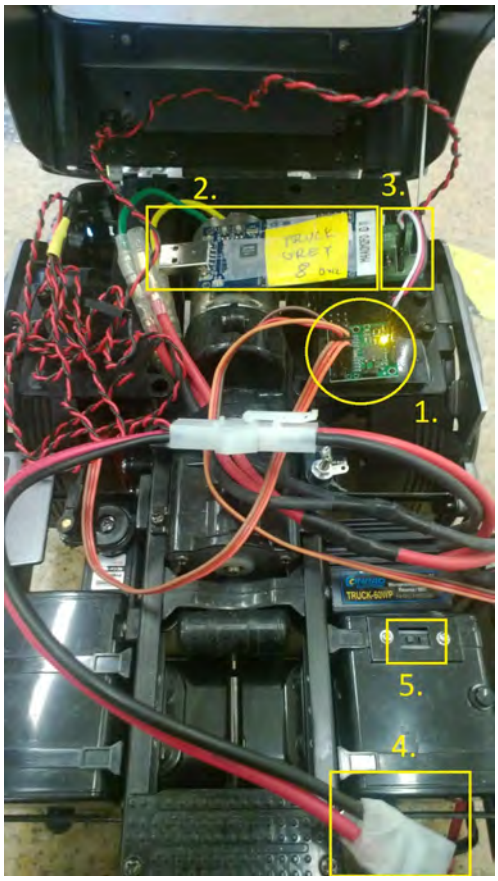


Figure 4.8: Different elements of the truck.

In Figure 4.8, apart from the elements associated with the communication system we have: number 4 that is a simple switch that enables the motors and number 5 that is the connection between the battery, that is underneath the truck, and all the other hardware elements.

4.3 Markers

The camera setup as said before identifies objects resorting to passive markers, that reflect the infrared light sent from the camera and allows the localization of a marker. An example of those markers is represented in Figure 4.9.

Figure 4.9: Example of a passive marker (courtesy of Qualisys).

In order to have a successful tracking of the trucks we need to have an unique configuration of the the rigid bodies, because if there are two similar configurations of markers, the tracking system can misidentify the trucks, which can obviously create dangerous problems. To avoid this, different and unique configurations (Figure 4.10) of markers were created.

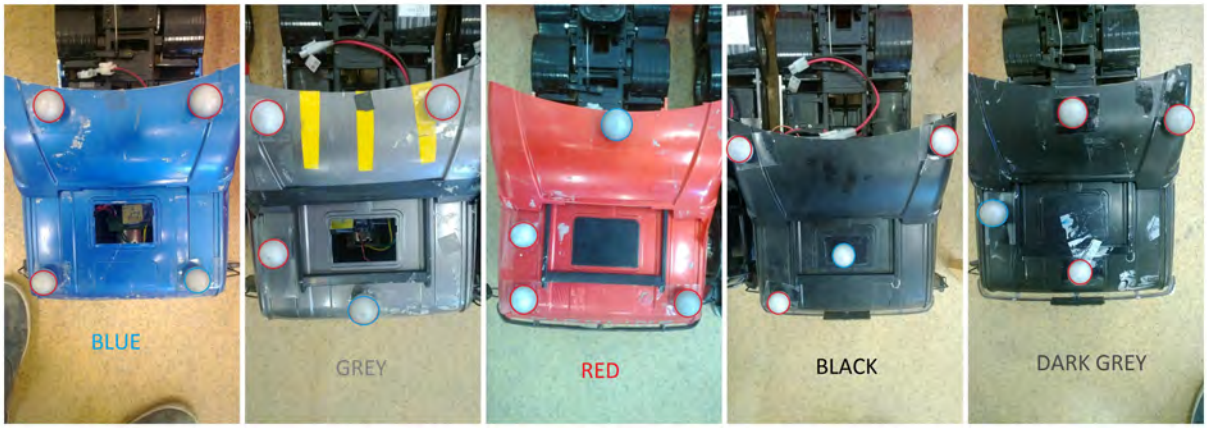


Figure 4.10: Markers positioning in the trucks.

To create an unique rigid body it is needed at least a set of 4 markers. The first 3 markers are used to create the axes of the body, and the last marker is positioned in different places for each truck in order to create unique dispositions. Those markers are placed more or less in front of the trucks, so the data from the localization of the truck is relative to the origin of the system and the front of each truck. In Figure 4.10 the markers highlighted in red are the ones that were used to create the axis, and the ones highlighted in blue are the extra markers that create the unique configurations.

4.4 Communication

The communication between subsystems has as central system a desktop computer that receives all the data from the sensors (cameras) and then calculates the control values and sends them to the actuators (trucks). The desktop computer is running a LabVIEW application that takes care off all the communications, monitors the entire system and makes the necessary decisions.

4.4.1 Communication Between PC and MoCap

As explained in section 4.1 the camera system provides the Euler angles for each truck, that information is fetched in the LabVIEW application through a special block provided by Qualisys. That block was modified because system only needs to get the pose of the truck, i.e. the Cartesian coordinates and the yaw. A new data measurement is sent continuously through a TCP connection the moment the cameras process a new reading.

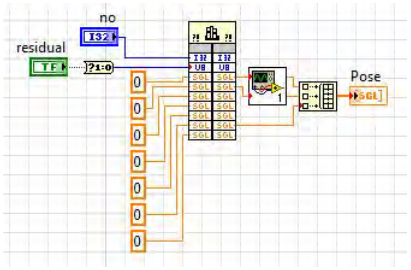


Figure 4.11: Block that receives the MoCap data.

4.4.2 Communication Between PC and Trucks

To establish the connections between the central desktop computer and the trucks, several pairs of motes are used, one in each truck and the respective pair connected to the desktop computer. A mote, also known as sensor node, is a node in a wireless sensor network. In this setup the motes used are the Tmotes Sky as seen in Figure 4.12.

Figure 4.12: Tmote Sky (courtesy of Tmote Sky).

The number of motes could have been reduced to one per truck and only one for the desktop computer, however this would decrease the transferring rate as one single mote would manage all the motes from the trucks. So this setup would be feasible and also more practical, nonetheless the chosen setup is more effective.

The motes connected to the desktop work as serial ports, this creates a problem because if you use directly the serial port only one application can interact with the mote. However there is a tool that removes this limitation, the Serial Forwarder. This way the application connects to a TCP stream whose packets source is the serial port, instead of connecting directly to the serial port.

Once the LabVIEW application has already calculated the command values, it needs to send them. It does so resorting to a known protocol between the motes, where the user only needs to specify the commands values, that are simply two values, one for the steering and the other for the velocity (the values have a range between 0 and 255). The LabVIEW block that does this process is represented in Figure 4.13.

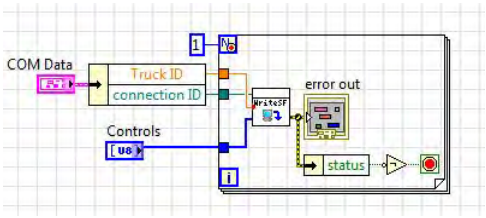


Figure 4.13: Block that sends the command values to the trucks.

Once the receiving mote has already received the command values, it needs to transform them in order to actuate in the motors' servos. The serial board adapter receives the data from the mote and passes it along to the Pololu board. Finally the Pololu board interprets the message received from the mote, and through the known protocol, it fetches the command data. With a Pulse Width Modulator board it sends the updated command values to the motors' servos.

4.4.3 Communication Between PC and Visualization Tool

A simple TCP connection is used to communicate with the VT. The TCP is ideal because it is reliable, and also because it enables the VT to run in any computer that has Matlab and an internet connection. This way the LabVIEW application sends several types of data that are helpful to create the VT, at the same time, it sends data to be stored and used in later analysis.

4.5 Road Network

The road network was idealized as seen in Figure 4.14. The main prerequisite was the dual lane feature, because it was key to have two lanes where the trucks could cross in opposite directions. Note that the inside road only exists because the road network was created in a partnership with another master student.

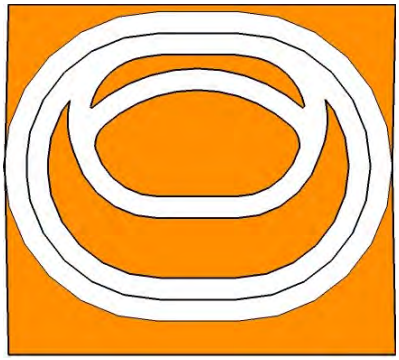


Figure 4.14: Sketch of the road network.

The road creation process is explain in detail in section A.4. Nevertheless a more technical approach is addressed in this section.

Each road is described by a set of waypoints. A Matlab application receives the information of markers placed on the ground to simulate those waypoints. The list of waypoints is sequential, so it is necessary to decide the orientation of the road. The user chooses the first two points and after that the application sequentially searches for the closest point with the equation 4.1 and this way it creates the ordered list of waypoints.

$$NextWP = min(\sqrt{(XCurrentWP - \textbf{XWPlist})^2 + (YCurrentWP - \textbf{YWPlist})^2}) \tag{4.1}$$

However if we use a larger number of markers the trajectory is much smoother, so to facilitate this task the application interpolates the waypoints, and creates a trajectory that can be exemplified in Figure 4.15.

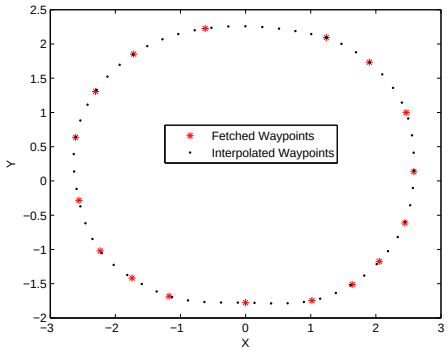


Figure 4.15: Fetched and interpolated waypoints.

4.6 System Integration

The system integration is truthfully very simple, because all subsystems are communicating with the main computer as seen in section 4.4. Nevertheless, the system can fail if some of its subsystem also fail. For example if the Qualysis system fails, it can spread some wrong position values along the chain of communication (Other issues that might arise can be seen in Appendix A).

Chapter 5

Modeling

When starting to understand the behavior of a HDV, the first step is to create a model and develop controllers for the movement that we want to implement in the experimental trucks. This intermediate phase is essential to save time and effort, because here it is possible to find some problems with the chosen approach and a first phase of tuning can also be done.

Since the project is also based in the analysis of fuel consumption, the creation of the model that represents the fuel consumed is, of course, essential.

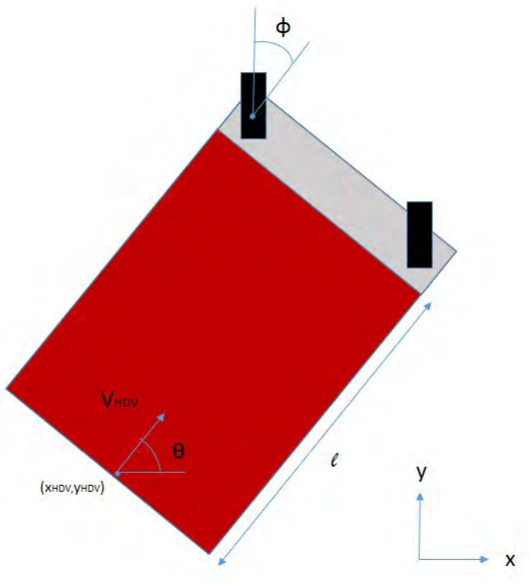


Figure 5.1: Representation of the kinematic model's notation.

5.1 Truck's Model

The trucks have a car-like representation, this means the control variables are the velocity and the rotational steering of the vehicle. This also means that the movement of the vehicle is limited, i.e., since the wheels are controlled simultaneously instead of separately, some trajectories are just not possible to be done, like for example a rotation over one point. A graphical representation of the model is pictured in Figure 5.1. With all the variables in Table 5.1, we can create a general model that can be further simplified with certain assumptions.

Notation	Description
(x_{HDV}, y_{HDV})	Spatial coordinates of the HDV
v_{HDV}	Velocity of the HDV
θ	Yaw
ϕ	Steering Angle
l	Length of HDV

Table 5.1: Description of the kinemactic model's notation

Note that the axis of the truck is represented in its back only because the rotation center is located in that area, nevertheless the axis can be placed virtually anywhere with small adjustments in the model.

The model can be represented by a state-space with the variables given in 5.1.

$$\begin{bmatrix} x_{HDV} \\ y_{HDV} \\ \dot{x}_{HDV} \\ \dot{y}_{HDV} \\ \theta \\ \phi \end{bmatrix} \tag{5.1}$$

The mathematical model 5.2 organizes in matrices all the differential equations that represent the movement of the vehicle. With a closer look we can see that a position of a vehicle is dependent, as expected, of the truck's velocity (v_{HDV}), acceleration (a_{HDV}) and orientation (θ, that has a relation with the vehicle's steering ϕ). One the limitations of the movement is created due to the steering, because this angle can only be in a small range of values.

$$\frac{d}{dt}\begin{bmatrix} x_{HDV} \\ y_{HDV} \\ \dot{x}_{HDV} \\ \dot{y}_{HDV} \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{\phi} + \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \\ 0 \\ \frac{\tan(\phi)}{l} \\ 0 \end{bmatrix} v_{HDV} \tag{5.2}$$

Where a_{HDV} and v_{HDV} are , respectively, the acceleration and velocity of the vehicle accordingly to 5.3.

$$\begin{aligned} a_{HDV} &= \sqrt{\ddot{x}_{HDV} + \ddot{y}_{HDV}} \\ v_{HDV} &= \sqrt{\dot{x}_{HDV} + \dot{y}_{HDV}} \end{aligned} \tag{5.3}$$

We can now assume that this model is highly reactive to velocities changes, this means that we can disregard the velocity transient, therefore obtaining the simpler model:

$$\frac{d}{dt}\begin{bmatrix} x_{HDV} \\ y_{HDV} \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ \frac{\tan(\phi)}{l} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{HDV} \\ \dot{\phi} \end{bmatrix} \tag{5.4}$$

It is possible to do one last simplification, if we consider that the rotational speed of the steering is very high, and so resulting in almost instantaneous steering positions. If we use this simplification the result is the model:

$$\frac{d}{dt}\begin{bmatrix} x_{HDV} \\ y_{HDV} \\ \theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \frac{\tan(\phi)}{l} \end{bmatrix} v_{HDV} \tag{5.5}$$

In this case the control variables are simply the truck's velocity (v_{HDV}) and steering angle (ϕ).

These models were used as a starting point to the implementation providing a starting point for a fesible controller. Also the models were used as a simulation tool to try out diferent configurations of controllers.

5.2 Controllers

To control the movement of a vehicle different variables need to be controlled individually, like for example, the velocity, in order to achieve the desired movement. We need to control two different aspects of the movement: first, the truck has to move to a reference spatial position (sections 5.2.3 and 5.2.4); second, when platooning, the distance between trucks must be a fixed reference value (section 5.2.5). One simple, but effective, tool that can be used is the PID controller further explained in section 5.2.1. Both the velocity and steering controllers presented in sections 5.2.3 and 5.2.4 are simply theoretical designs for the controllers. A more practical approach is presented in chapter 6.

5.2.1 PID Controller

A PID controller is a control loop feedback mechanism, that consists of tree different terms: proportional, integral and derivative. A PID controller can be used to regulate a state-space variable that you want to control. The controller calculates the difference between the measured variable and a reference

value. This difference is called the error, that has the same units as the variable that you are controlling. The final goal is to have an updated value of the variable that when sent to the actuators of your system, regulates the manipulated variable around the reference value. This controller is one of the most used worldwide in various applications, mainly because of its simplicity and effectiveness.

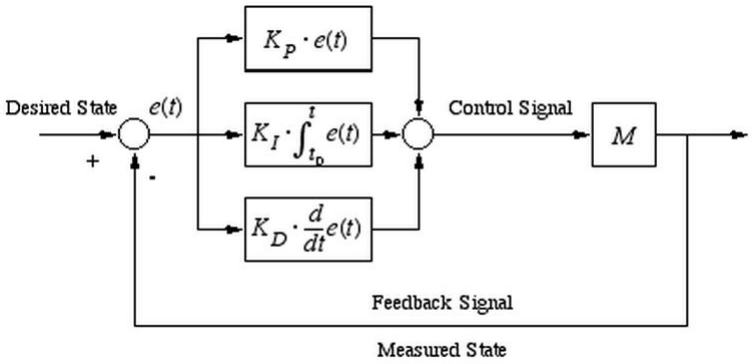


Figure 5.2: PID controller.

The manipulated variable (control signal) is simply the sum of all the terms of the PID controller.

$$ControlSignal = P_{term} + I_{term} + D_{term} \tag{5.6}$$

The proportional term just adjusts the control variable multiplying the current error by a proportional gain, K_P.

$$P_{term} = K_P e(t) \tag{5.7}$$

The integral term sums all the instances of the error over time, creating the accumulated error. This makes the response of the system more reactive and eliminates the steady-state error.

$$I_{term} = K_I \int_0^t e(\tau)\, \mathrm{d}\tau \tag{5.8}$$

The derivative term calculates the rate of change of the error and multiplies it by the differential gain, K_D. Since it measures the changes of the error, it can detect bigger ones, therefore it compensates in advance to those changes in the error. This term improves the settling time and stability of the system.

$$D_{term} = K_D \frac{d}{dt} e(t) \tag{5.9}$$

Different variations of PID can be used, like for example in some situations depending on the variable that is being controlled, it might be beneficial not to use either the integral term or the differential term.

5.2.2 Controllers Setup

When designing any controller, we start to analyze what kind of data we have beforehand. We know

that the trajectory consists of sets of waypoints and also that we don't know directly the current velocity of the truck, but only the current position of the truck.

The current waypoint is always updated as the closest waypoint to the truck. Note that the closest waypoint has to be the in front of the truck otherwise the orientation errors would be such that the truck would try to reach a point that it would have already passed by. A configuration of the errors used in the controllers is represented in Figure 5.3.

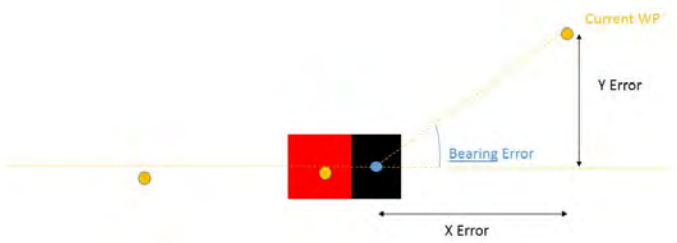


Figure 5.3: Errors representation.

5.2.3 Velocity Controller

The goal of the velocity control is to make the truck to run at a constant speed, leaving the burden of following the trajectory to the steering controller (section 5.2.4). So the error that is used in the PID controller is just the difference between the velocity that the truck should have and the velocity it has in reality.

$$e_V = V_{ref} - V \tag{5.10}$$

As said before the real velocity of the truck is not known (only the position), but it can be calculated using both components of the velocity accordingly to:

$$V = \sqrt{(V_X)^2 + (V_Y)^2} \tag{5.11}$$

Both components of the velocity can be calculated as follows, where Δt is the time gap between consecutive measures.

$$\begin{aligned} V_X &= \frac{x[k]-x[k-1]}{\Delta t} \\ V_Y &= \frac{y[k]-y[k-1]}{\Delta t} \end{aligned} \tag{5.12}$$

In 5.13 is presented the final form of the velocity controller.

$$control\_signal = K_P e_V[k] + K_I \sum_{n=0}^{k} e_V[n] + K_D \frac{e_V[k] - e_V[k-1]}{\Delta t} \tag{5.13}$$

Since the system is discrete, instead of an integral in the integrative term we have a cumulative sum, and instead of a derivative in the differential term we have the rate of change in the error for a time period of Δt.

5.2.4 Steering Controller

The steering controller is the agent that is responsible for the trajectory following since the velocity controller just keeps a constant speed. One of the information that is known is the orientation, also known as yaw, of the truck. So in order to follow the trajectory, the controller just calculates the misalignment between its orientation and the current waypoint of the trajectory. This is represented in Figure 5.3.

The misalignment (bearing to the waypoint B_{WP}) is given by the relation

$$B_{WP} = atan^{-1}(WP_y - y, WP_x - x) \tag{5.14}$$

Note that the control variable is not the orientation of the truck, but the orientation of the wheels (the steering angle, ϕ, in Figure 5.1). So the error of sterring is

$$e_S = B_{WP} - yaw \tag{5.15}$$

The final version of the PID controller is presented in

$$control\_signal = K_P * e_S + K_I \sum_{n=0}^{k} e_S[n] + K_D \frac{e_S[k] - e_S[k-1]}{\Delta t} \tag{5.16}$$

5.2.5 Platooning Distance Controller

The controller to the platooning distance is much easier to design, due to the controllers of velocity and steering the truck being already following a trajectory. So to manage the distance between trucks, we only to control the velocity of the truck. Note that this task is much easier because the distance between trucks is calculated along the trajectory.

So to control the velocity on top of the velocity controller, we must change the reference velocity value that we send to the controller. This way the velocity controller will try to adjust to a reference velocity that already tries to maintain the platoon distance.

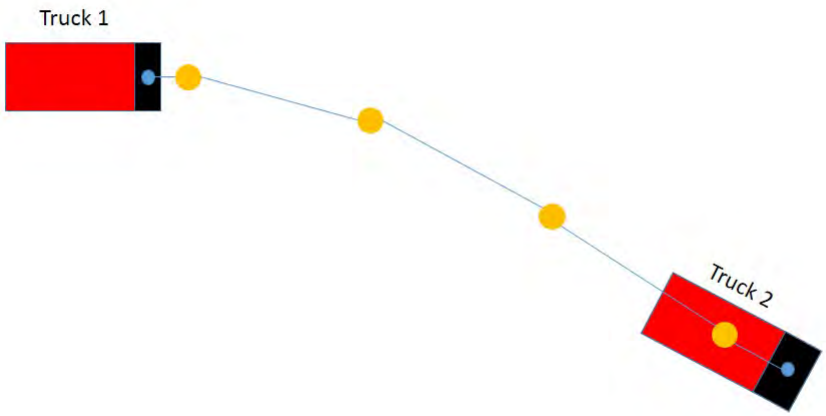


Figure 5.4: Representation of the platoon distance.

The reference value, V_{ref}, that is sent to the velocity controller can be controlled accordingly to

$$V_{ref} = V\frac{d}{d_{ref}} \tag{5.17}$$

This calculation is done to every truck that belongs to the platoon apart from the truck that leads the platoon. Where d is the distance from a given truck in the platoon to the truck that is placed right in front of it. The distance d_{ref} is the reference distance that the trucks in the platoon should maintain from each other, also called platoon distance. And V is the speed that all the trucks should maintain, that was chosen to be the velocity of the truck that leads the platoon. When the current distance is similar to the platooning distance, the reference speed is merely the velocity of the platoon's master.

If the trucks are very reactive, the velocity is sufficiently stable and all the areas where they run are mostly leveled we can assume that the platooning control can be done either by keeping the a time distance or a physical distance between trucks. Remember as well that since anti-platooning is a special case of platooning, so the controller is very similar, however the reference value is not the distance between the trucks. In this case the reference is the time between the trucks t_{ref}:

$$V_{ref} = V\frac{t}{t_{ref}} \tag{5.18}$$

The time gap separating the trucks is continously updated, and can be estimated using both the truck velocity and the distance between trucks.

5.3 Fuel Consumption

When platooning, we need to have a model that when applied in real-time gives information about the fuel consumption. The fuel model studied was referenced both in [11] and [9].

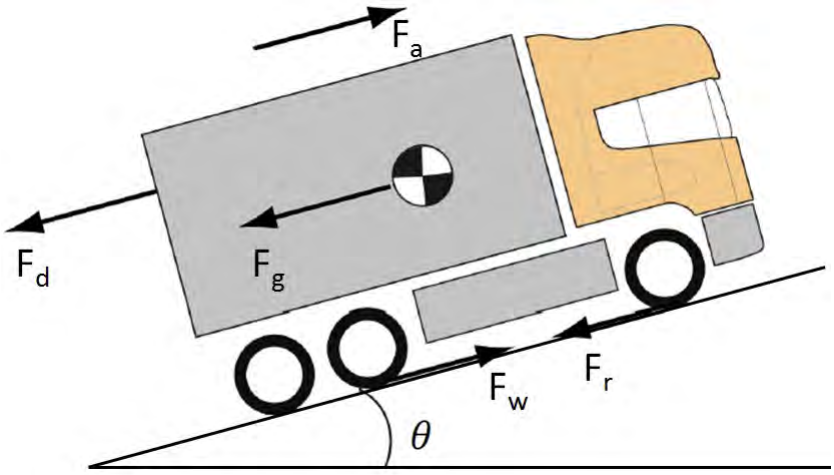


Figure 5.5: Longitudinal forces acting on a HDV (reprinted from [4]).

The Figure 5.5 is a simple representation of all forces applied to the trucks, where θ is the inclination of the road. The longitudinal equilibrium of forces can be easily calculated as 5.19, where F_w is the total resistance (N).

$$F_w(a, v, \theta) = F_a + F_d + F_r + F_g \tag{5.19}$$

In 5.20, accordingly to Newton's second law of motion, is calculated the applied force to move the truck, where m is the mass of the vehicle (kg), m_j is the equivalent mass of the inertia of the moving parts (kg) and a is the vehicle's acceleration (m/s^2).

$$F_a = (m + m_j)a \tag{5.20}$$

The air drag resistance is calculated in 5.21, where cd is the air drag coefficient, A is the frontal area of the vehicle (m^2), ρ is the air density (kg/m^2) and v is the vehicle's velocity (m/s).

$$F_d = \frac{1}{2} c_d A \rho v^2 \tag{5.21}$$

The rolling resistance is represented in 5.22, where c_r is the rolling resistance coefficient and g is the gravity acceleration (m/s^2).

$$F_r = mgc_r \cos\theta \tag{5.22}$$

Lastly it is represented the gravitational resistance.

$$F_g = mg\sin\theta \tag{5.23}$$

Combining all the resistances applied to the vehicle we get the new formulation to the total resistance (F_w).

$$F_w(a, v, \theta) = (m + m_j)a + \frac{1}{2}c_d A\rho v^2 + mgc_r\cos\theta + mg\sin\theta \tag{5.24}$$

The instantaneous tractive power, $P(a, v, \theta)$ (k**W**), can be calculated using the total resistance.

$$P(a, v, \theta) = F_w(a, v, \theta)v \tag{5.25}$$

The instantaneous fuel consumption is estimated according to

$$ft(a, v, \theta) = ft_{idle} + \frac{\delta}{\varepsilon_b H_d}\Big(\frac{P(a, v, \theta)}{\eta_t} + P_{aux}\Big) \tag{5.26}$$

$$\delta = \begin{cases} 1 & F_w > 0 \\ 0 & F_w \leq 0 \end{cases} \tag{5.27}$$

The estimation of the instantaneous fuel consumption is dependent of: ε_b, the brake thermal efficiency of the engine; η_t, the total transmission efficiency; H_d, the energy density of diesel (J/cm^3); P_{aux}, the power required for auxiliary units (k**W**); δ, a switch that disables the fuel model when F_w is negative, i.e., when the total resistance is favorable to movement.

Finally the total fuel consumption, F_{tot} (cm^3), is simply the accumulation of instantaneous fuel consumption over time.

$$F_{tot} = \int ft(a, v, \theta)\mathrm{d}t \tag{5.28}$$

This is a general model for the fuel consumption, that describes the amount of fuel that is consumed instantaneously and when accumulated gives the total amount of fuel consumed.

This model can be simplified if some assumptions are used. If we consider that the road system is completely leveled, i.e, the inclination of the road (θ) is always 0, then there is no gravitational resistance (F_g) in the truck.

This way, we get the simplified longitudinal equilibrium of forces in 5.29.

$$F_w(a, v, \theta) = F_d + F_r + F_a \tag{5.29}$$

Where the rolling resistance is also simplified.

$$F_r = mgc_r \tag{5.30}$$

Is also possible to have a different type of analysis, where the goal is not to know the exact amount

of fuel consumed, but rather to know the relative fuel consumption between two different trucks. This implies that the physical characteristics of the vehicles' motor are not relevant. Also we can further simplify the model if we consider that both the fuel consumed when stopped (ft_{idle}), and the power required for the auxiliary units (P_{aux}) are null.

This way the instantaneous fuel consumption is given in

$$ft(a, v, \theta) = \alpha F_w(a, v, \theta) \tag{5.31}$$

Where α is the parameter that gathers all the characteristics of the vehicle.

$$\alpha = \frac{\delta}{\varepsilon_b \eta_t H_d} v \tag{5.32}$$

Finally, the instantaneous fuel consumption model is given by 5.33 apart from a parameter α.

$$ft(a, v, \theta) = \frac{1}{2} c_d A \rho v^2 + mgc_r + (m + m_j)a \tag{5.33}$$

Since this model is constructed to be used to full sized trucks, the implementation in scaled trucks is not trivial. First we have the problem that the model encapsulates several parameters that are related to the characteristics of the motor (can not be translated to scaled trucks), so we can only use this model as a comparison method between scaled trucks. This is possible because the fuel consumptions are estimated to the scaled trucks apart from a constant, that way we can always compare estimations between trucks.

Chapter 6

Implementation

This chapter addresses the implementation of the concepts introduced in Chapter 2, where all the theoretical concepts were introduced. Here small examples of features that are not directly related with the scenarios are presented. This can be considered a chapter of proof of concepts in order to validate the results obtained in the scenarios (Chapter 7).

Several experiments were done, were it was tested the behavior of the trucks when traveling alone or within a platoon. In those experiments some data was stored to analyze different characteristics of the trucks' movement.

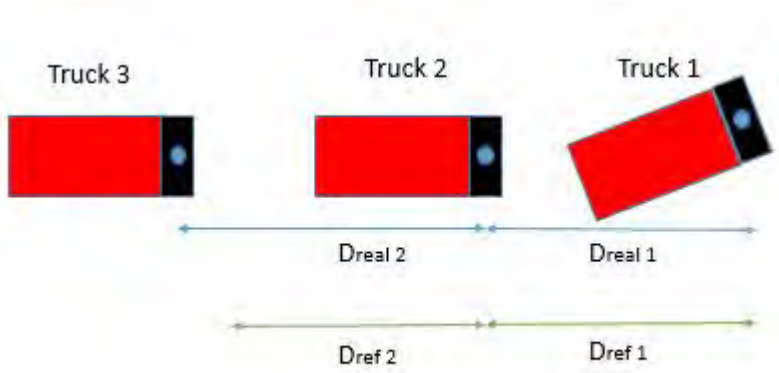


Figure 6.1: Representation analyzed parameters.

Basically, we analyze for each truck the efficiency of every controller that was implemented. That means that the trucks' velocity and steering are analyzed, both for when in an alone run and when platooning. Also when platooning, the platooning distance is monitored in order to analyze the performance of the platooning's implementation.

6.1 Calibration

One of the first tasks to do is to calibrate the trucks because they receive command values that vary between 0 and 255, so we need to find a relation between those values and the real values of steering ($degrees$) and velocity (m/s).

First, to calibrate the velocity, a small experiment was done, where constant command values were sent to the trucks, and then a Matlab application fetched the pose's information from the Qualisys system and calculated the velocity of the truck accordingly to equation 6.1. The trucks were only calibrated for the range of values that are reasonable, for example, since the backwards movement is never considered, values below 127 were not calibrated. Values higher than 170 were also not calibrate, because those speeds are too high for the experiments that were done.

$$v = \sqrt{(LastX - CurrentX)^2 + (LastY - CurrentY)^2} \tag{6.1}$$

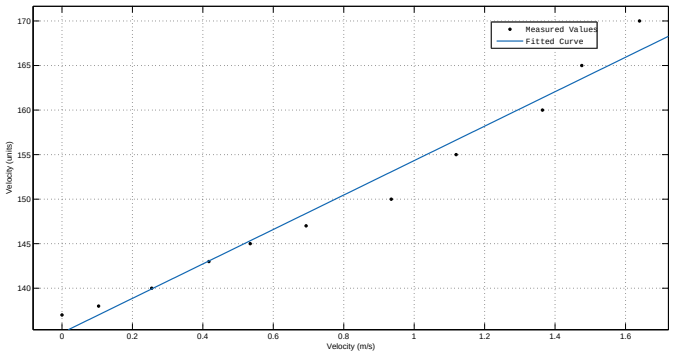


Figure 6.2: Calibration of the velocity.

A simple curve fitting (Figure 6.2) provides us the model that relates the real velocity with the command value sent to the truck (equation 6.2)

$$CommandValue = round(19 * RealSpeed + 127) \tag{6.2}$$

To calibrate the steering requires a much simpler experiment, where once again constant command values were sent to the trucks, i.e. the truck was stopped and the steering values varied. The steering (angle between wheels and the origin axis of the truck) was then measured in degrees as represented in Figure 6.3. For the steering, it were as well only calibrated the range of values considered reasonable. In this case this range is due to the technical limitations of the trucks' steering, because the command values for angles larger than 25 degrees don't have any effect on the truck's steering, i.e. the maximum value of steering does not coincide with maximum of the command value.

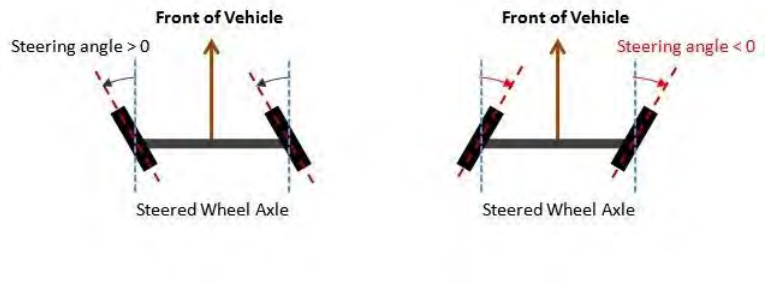


Figure 6.3: Definition of steering

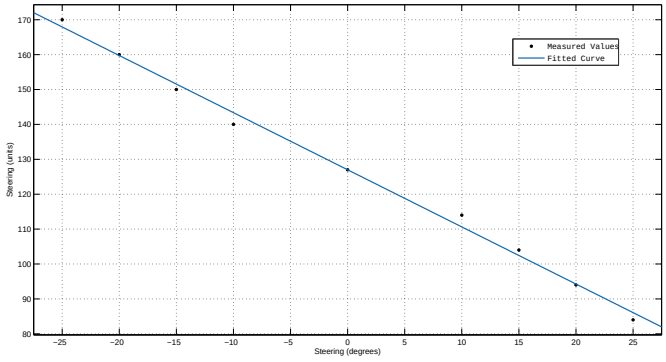


Figure 6.4: Calibration of the steering.

From the curve fitting in Figure 6.4, yields the relation between the angle of steering ($degrees$) and the command value sent to the truck (equation 6.3).

$$CommandValue = round(-1.637 * Steering + 127) \tag{6.3}$$

With the command values calibrated all the control algorithms can be done with the control variables in SI units ($radians$ for the steering and m/s for the velocity), and then converted to command values just before being sent the trucks.

6.2 Controllers' Simulation

Before implementing the controllers in the trucks, it is recommended to do some previous analysis through simulations. This way it is possible to understand if the developed controller is feasible, and simultaneously find a rough estimation of the controller gains that can be later calibrated in the real system. A simple simulation was done using Simulink, where the truck's movement was modeled as in section 5.1. The controllers for the velocity and steering were the ones explained in sections 5.2.3 and 5.2.4.

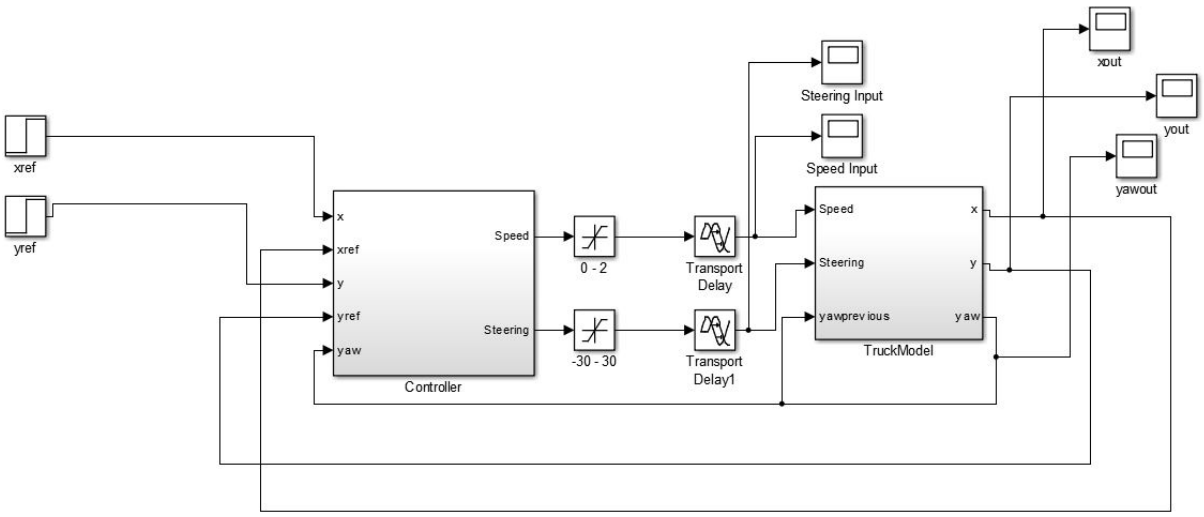


Figure 6.5: Simulink simulation.

After some fine tuning the controller gains in Table 6.1 were the ones that presented best results.

Gains	Velocity	Steering
K_P	1	1
K_I	0,2	0,05
K_D	3	0,1

Table 6.1: Simulated PID gains.

This results are not necessarily the values the were used in the trucks' controllers, instead they were a starting point that allowed to have a rough estimation of the gains that could be improved with extra tuning.

6.3 Single Truck

This section showcases the functionality of the system for one single truck. This way it is possible to analyze the effectiveness of both velocity and steering controllers. The controllers are the ones presented in chapter 2 but with some modifications, because it is common when we try to implement some theoretical concept some simplifications are possible or even required to be done.

A simple experiment was realized, where a single truck follows one of the road network's lane. During the experiment the truck had different changes in the reference speed that it should maintain.

Note that a fuel analysis for this case could be done, nevertheless it would be irrelevant because the fuel model implemented is the one in 5.33. This means that the instantaneous fuel consumption is a relative measure, because we can't quantify the parameter in 5.32. Since this is a simulation, the parameters of the truck would not be realistic.

6.3.1 Velocity

The controller implemented is represented in 6.4. This controller is basically the same as the one designed in 5.13, but now the velocity value must be converted from m/s to the units of the command value that is sent to the truck. This is done with the calibration done in 6.1 where the velocity is mapped to the units that the truck works with.

$$CommandValue = convert\Big(K_P e_V + K_I \sum_{n=0}^{k} e_V[n] + K_D \frac{e_V[k] - e_V[k-1]K_D}{\Delta t}\Big) \tag{6.4}$$

After running the trucks it was immediately noticeable that each truck needed a unique controller. Some of the trucks were more reactive than others so the controller gains were adjusted in order to have all the trucks with roughly the same behavior.

Gains	Blue	Black	Red	Grey	DarkGrey
K_P	0,7	1	0,8	1	0,8
K_I	0,2	0.2	0,3	0,15	0,2
K_D	3	1	3	3	3

Table 6.2: Implemented velocity PID controller gains.

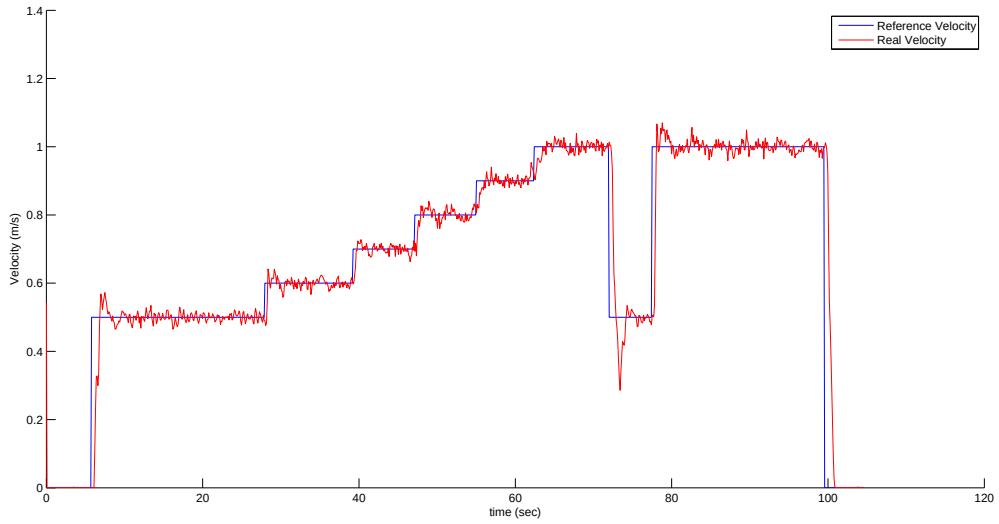


Figure 6.6: Real velocity and reference velocity for a single truck.

In Figure 6.6 it is possible to see an overview of the experiment, where it is clearly noticeable the changes of speed. We can see that the system appears to be very stable and quick to react to changes in speed. Small disturbances appear to be noise, but bear in mind that this happens because the velocity used in the controller is dependent of the readings from the Qualysis system, so if some oscillation in the readings of the truck's position occurs, it will propagate to the velocity estimation. Also the system has

a period of approximately $100ms$, due to the method of communication that uses T-motes that to avoid congestion in the communication channel sets the maximum frequency to $10Hz$. If it were possible to have a higher frequency the system would be even more stable, and would reduce the oscillation around the reference value.

Nevertheless the error between the real velocity and the reference value is quite small, for example for the entire experiment with the single truck the mean value of the error between those two values is only $0.0297m/s$.

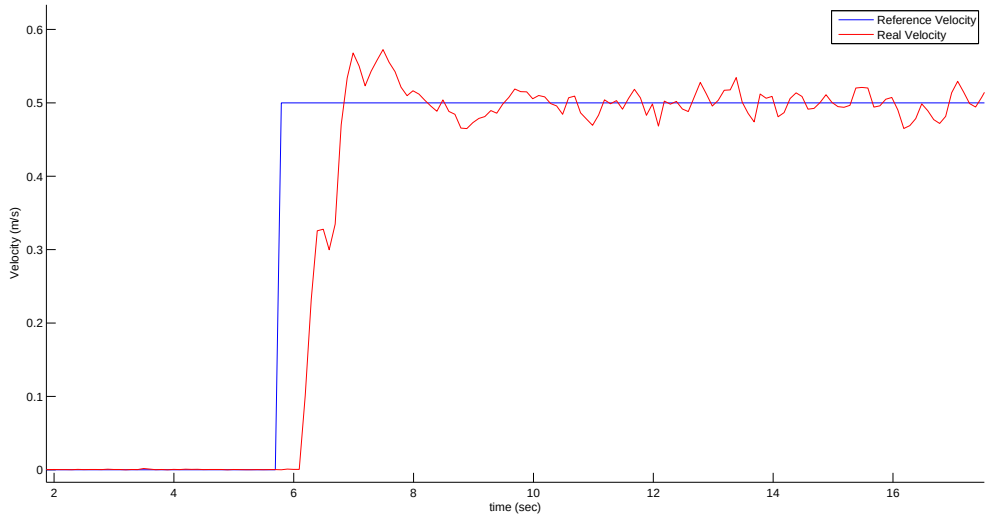


Figure 6.7: Acceleration of a single truck from a stopped position.

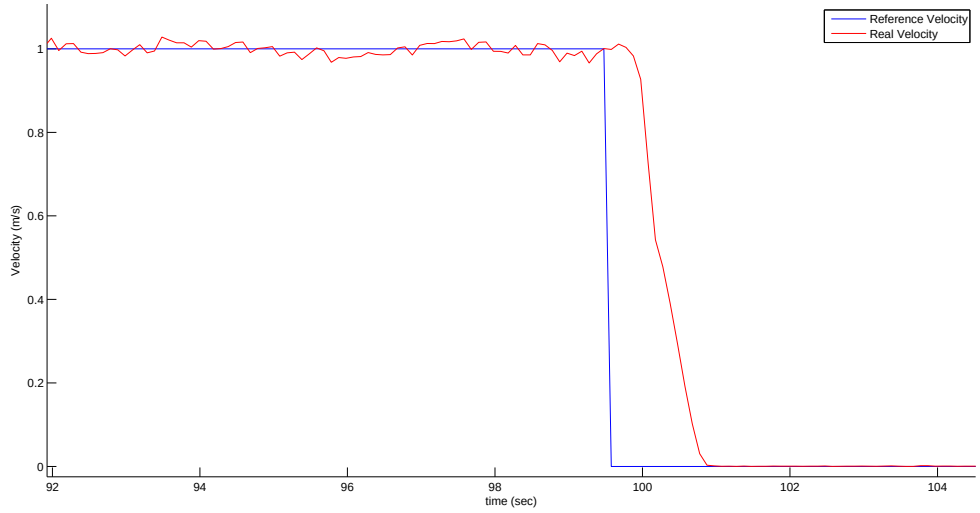


Figure 6.8: Stopping of a single truck.

In Figure 6.7 is highlighted a special section of the experiment, where the truck starts from a stopped

position and accelerates until it reaches the reference velocity. It is noticeable that the system is quite reactive, since the truck reaches the desired velocity within less than two seconds.

In Figure 6.8 we have the opposite experiment where the truck has a constant velocity of $1m/s$ and suddenly it breaks until it reaches a full stop. Again the system proves to be very reactive since it takes roughly only one second to the truck to became completely immobilized. It is visible a considerable reaction delay when the reference value changes to zero. This delay is simply one iteration of the control loop $(100ms)$, that it is the time that it takes for the new command value to be sent to the truck and a new estimation of the velocity to be calculated.

6.3.2 Steering

The controller used is the same one designed in 5.2.4, however after some experiments it was clear that the best results appeared when the controller used was simply a PD, i.e, the integral term isn't taken into account. Once again after computing the controlled value of the steering angle, it is still necessary to convert it to the units that the truck receives, once again we resort to the calibration in 6.1.

$$CommandValue = convert\Big(K_P e_S + K_D \frac{e_S[k] - e_S[k-1]}{\Delta t}\Big) \tag{6.5}$$

After some tests other modifications had to be done. The error of steering is still calculated with the same process, but now the method to chose the current waypoint has changed. As said before the closest waypoint must necessarily be in front of the truck, otherwise the truck would try to to a spacial position that had already passed by. The new notation is represented in Figure 6.9.

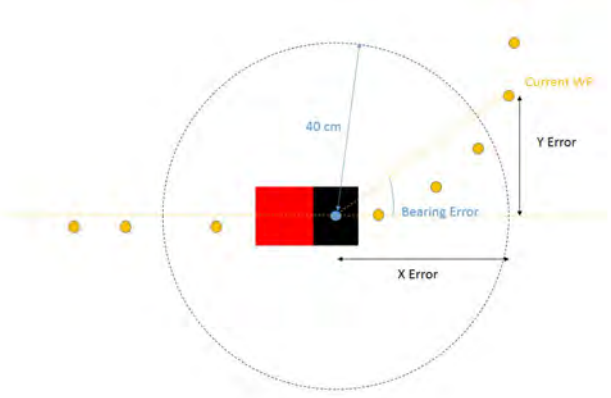


Figure 6.9: Errors representation of the implementation.

Now there is a safety margin that guarantees that the closest waypoint is at least a safety distance away, in this case $40cm$. This brings more than one benefit: first, the truck has now a safety margin that prevents the truck to choose a wrongful waypoint as the current one; now the trajectory following of the path is much smoother, because this feature enables the truck to orient itself to the waypoint that it is following much sooner.

After some fine-tuning, the trajectory controller has the gains presented in Table 6.3.

Gains	Blue	Black	Red	Grey	DarkGrey
K_P	1,5	0,9	1	0,6	1
K_D	0,05	0,05	0,05	0,05	0,01

Table 6.3: Implemented steering PD controller gains.

In Figure 6.10 one can see both the waypoints of one lane and the trajectory of one truck during a lap on that lane. It is possible to see that the system is very stable with few oscillations and the steering controller is efficiently guiding the truck along the desired trajectory.

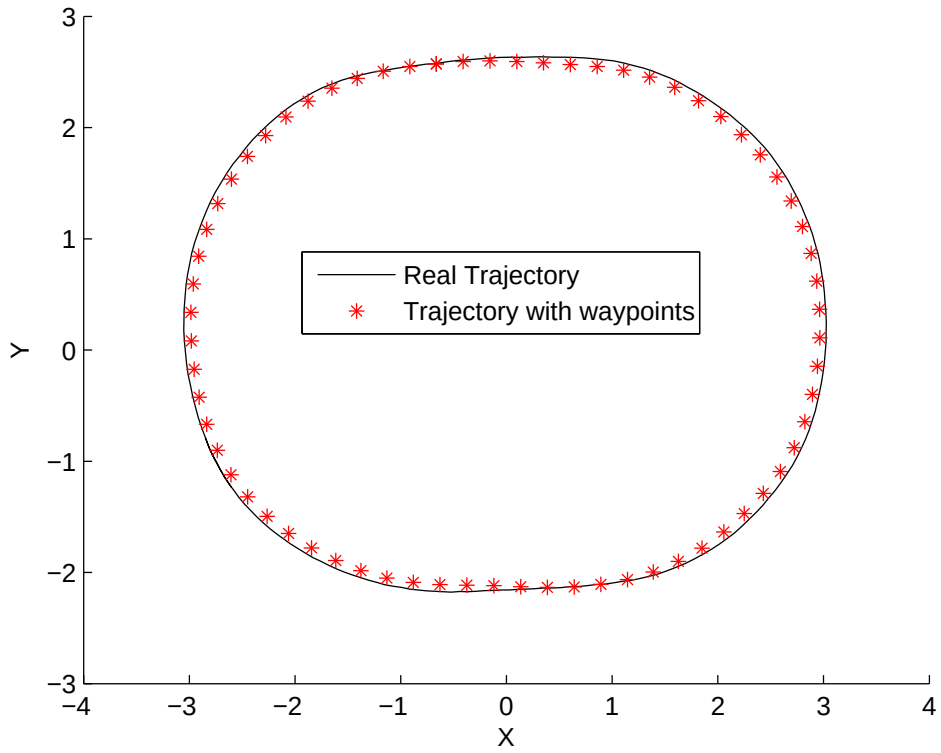


Figure 6.10: Trucks trajectory with waypoints.

In Figure 6.11 it is represented the steering of the truck during half of a loop around the lane. It is very noticeable that there is a constant static error between the the real steering and the reference value (the error is approximately $23°$). However, this error is not due to the absence of the integral term in the steering controller. As previously explained the closest waypoint is at least $40cm$ away, so the reference value for the steering is initially updated for a value that is beyond the maximum range of the trucks steering, that way it always has a constant error that the range of the steering doesn't allow to eliminate. Although, when in a strait line that bias tends to zero, but once a curve approaches again the bias starts to grow again, roughly to the same value as before.

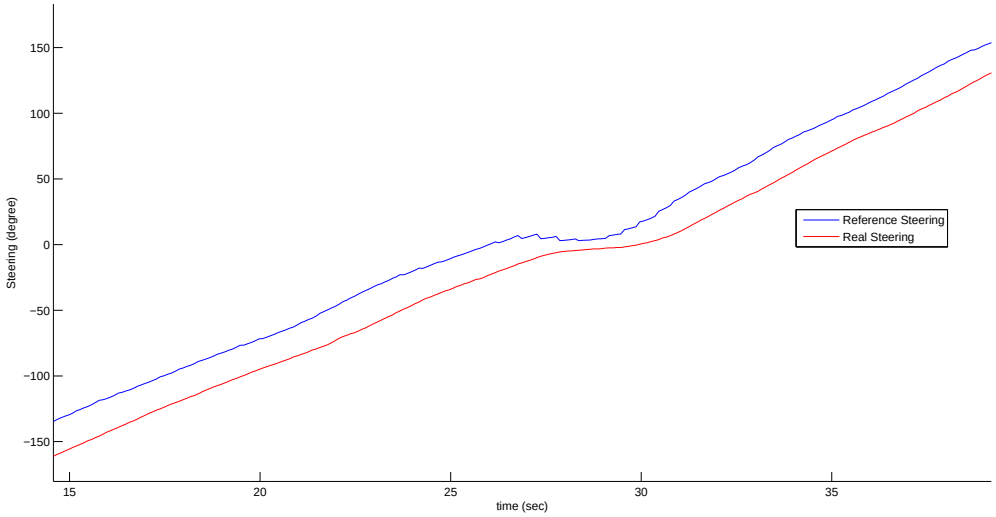


Figure 6.11: Real steering and reference steering for a single truck in half a loop.

6.4 Platooning

Now we analyze the platoon implementation, with experiments using three trucks that form a platoon, with different platoon distances. Due to the dimensions of the road it was considered that the platooning distances bigger than 3 meters are considered anti-platooning distances, i.e, the platoon still maintains the formation but the fuel savings of the platooning phenomenon are no longer relevant. With three trucks it is now possible to do fuel consumption analysis, because the instantaneous fuel consumption model used yields relative consumption, that way we can do comparisons between trucks.

6.4.1 Platooning Distance

The platooning distance is represented in Figure 6.12. Note that the platooning distance is measured as the gap between the fronts of the trucks, that way, for a platoon with three trucks we have two platooning distances, that are controlled with the same reference value. The controller technique is the one demonstrated in 5.2.5.

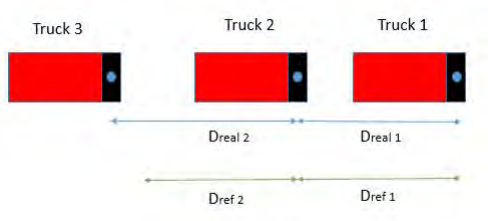


Figure 6.12: Representation of platooning distance.

In Figure 6.13 the overview of the experiment is represented, where there are different changes of the platooning distance.

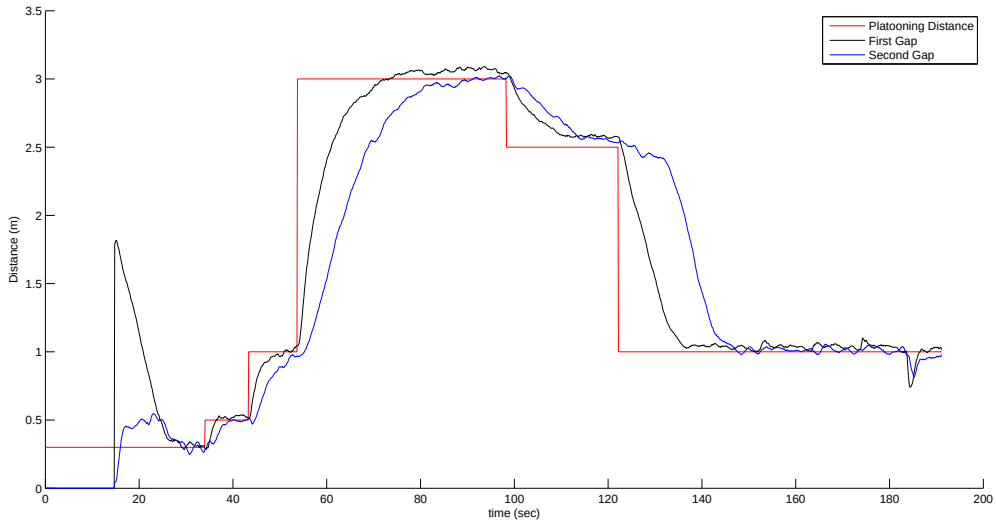


Figure 6.13: Platooning distance and gaps between trucks.

The system has proven to be very robust, for example it can create stable platoons with small platooning distances like $10cm$. The controllers were tuned to gains that allow the platooning distance to be as stable as possible (oscillations are around $2 - 6cm$), that way we made a compromise in the velocity controller, because this way it doesn't guarantees stability for very high velocity, however this velocities are not used in experiments.

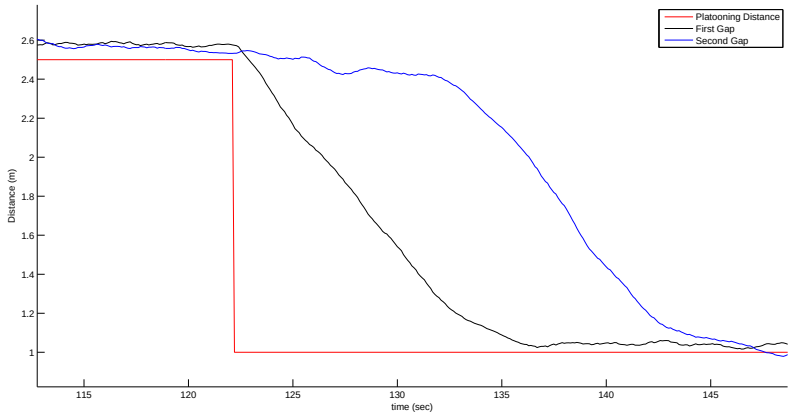


Figure 6.14: Platooning distance and gaps between trucks during an approach.

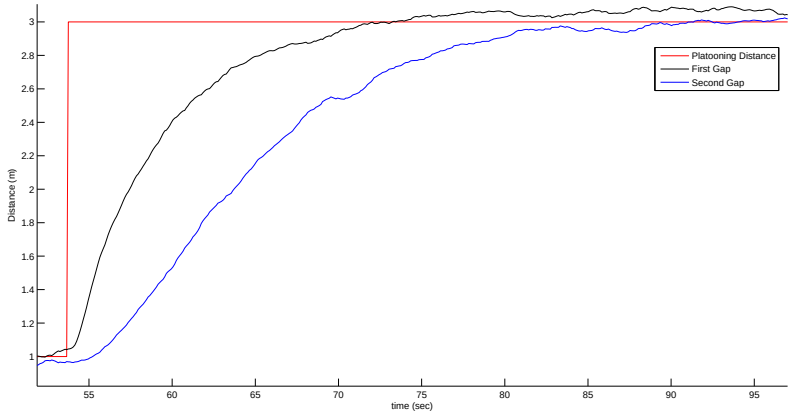


Figure 6.15: Platooning distance and gaps between trucks during a separation.

In Figure 6.14 is possible to see the representation of an approach where the gap between trucks was shortened for the second truck in the platoon in under 6 seconds, and for the third one in under 10 seconds. The approximation is not simultaneous for all the trucks in the platoon because there is an effect of propagation in this phenomenon, i.e, when the second truck tries to shorten the gap to platoon leader, it will affect the gap to the truck behind it, and only now the third truck will react to this change. The same phenomenon happens in Figure 6.15, the only difference is that now the platooning distance increases. This effect can be reduced if a controller based in vehicle-to-vehicle communication system is used.

6.4.2 Velocity

When platooning the leader of the platoon has a fixed reference velocity, the subsequent trucks have the velocity of the leader has the reference value. In Figure 6.16 is shown the platoon velocities with change of the reference velocity.

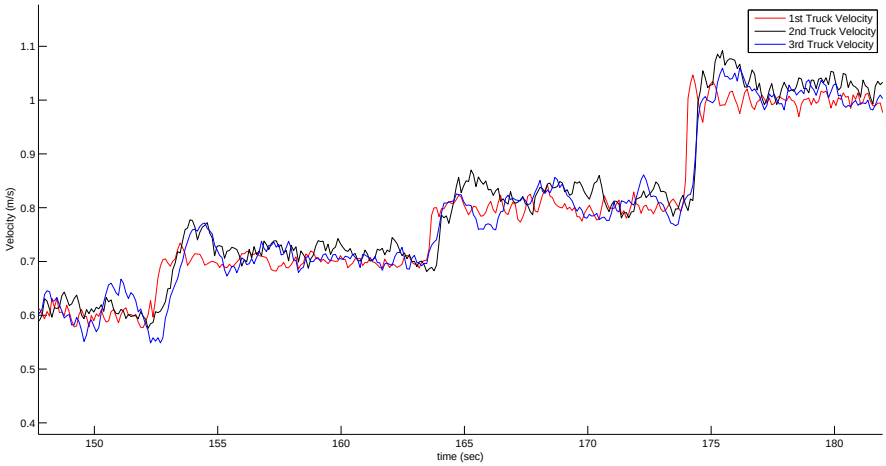


Figure 6.16: Velocities of the platoon's trucks with velocity change.

As seen to the single truck, the system is very reactive, the same happens for the trucks when in a platoon, nevertheless in this case, it has more oscillations because it is simultaneously controlling a constant velocity and the platoon distance.

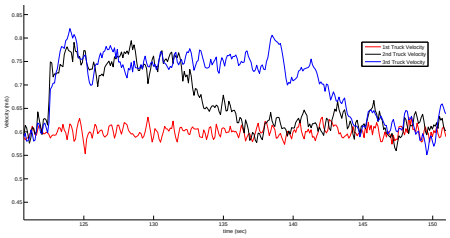


Figure 6.17: Velocities of the platoon's trucks with an approximation.

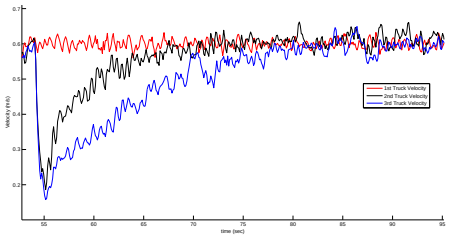


Figure 6.18: Velocities of the platoon's trucks with a separation.

Both Figures 6.17 and 6.18 present the same effect of propagation explained in section 6.4.1. Since a modification on the reference velocity creates a distortion on the gaps between trucks, the referred propagation phenomenon appears once again.

6.4.3 Fuel Consumption

It is possible to do a relative analysis of the fuel consumption between trucks resorting to the model presented in section 5.3. In Figure 6.19, it is possible to see an example the instantaneous fuel consumption in a platoon with three trucks.

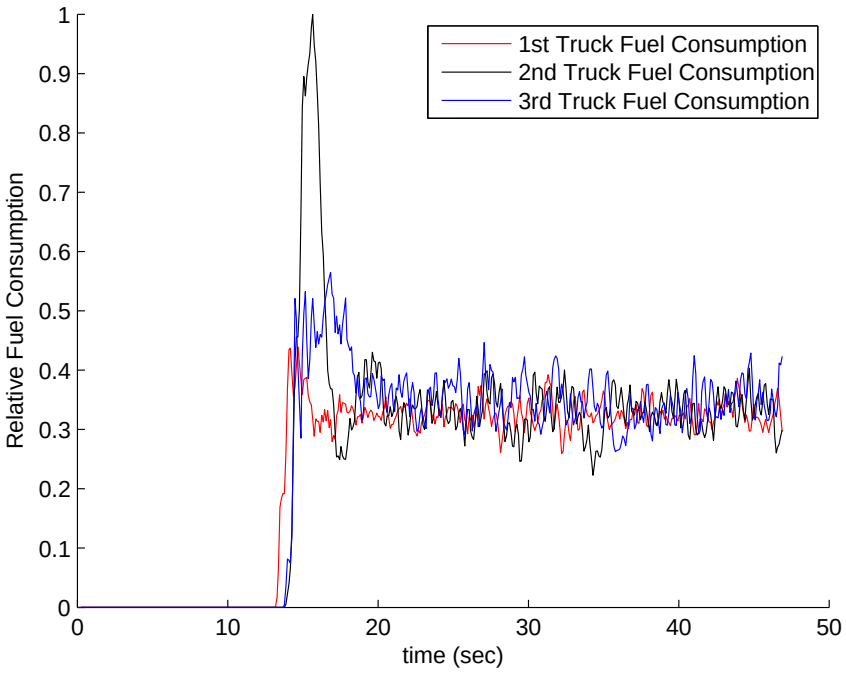


Figure 6.19: Instantaneous Fuel Consumption.

Since our problem definition states that the fuel consumption when the truck starts its movement is extremely high due to its weight, then our model has to take into account the accelerations of the truck when estimating the instantaneous fuel consumption. That can be observed in Figure 6.19. Of course the fuel model 5.33 could only be applied once all the parameters are known. Table 6.4 contains those parameters. We need as well to do a translation from the measurement units to the real world units. For that purpose it is used a gain, that currently is 30, this means that for example if the scaled truck is running at $1m/s$ then the fuel model is estimated for speeds in the order of $30m/s$, i.e. $108km/h$.

Parameters	Value
HDV mass (m)	$90000\ kg$
Front Area (A)	$10.26\ m^2$
Rolling resistance coefficient (c_r)	$7 * 10^{-3}$
Air drag coefficient (c_d)	0.6
Air Density (ρ)	$1.29\ kg/m^3$

Table 6.4: Fuel model parameters suggested in [13].

Finally it is possible to calculate the relative total fuel consumption, presented in Figure 6.20. Note that for a platoon with distances in the anti-platooning range the total fuel consumed is roughly the same for each truck within the platoon.

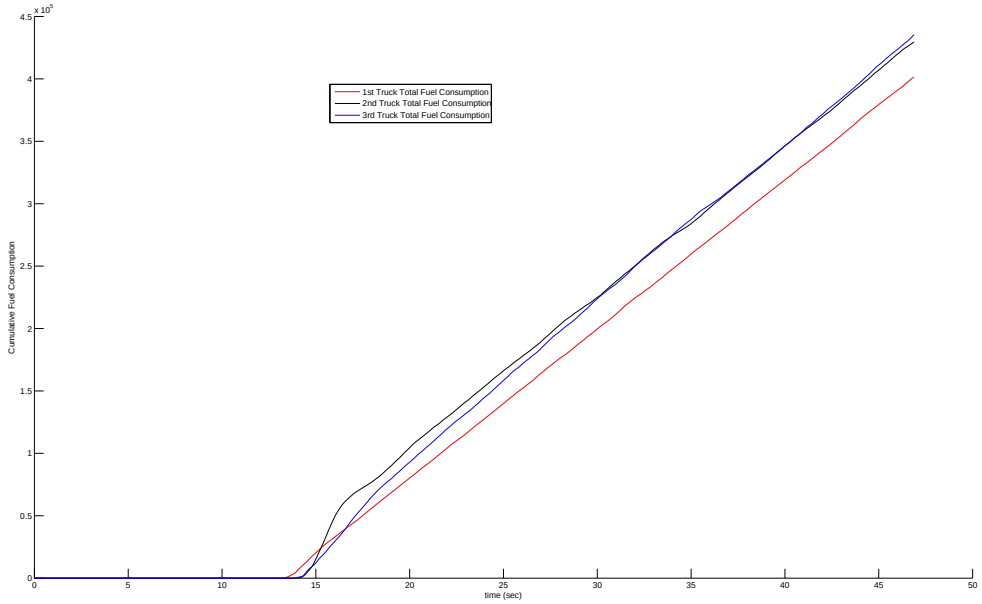


Figure 6.20: Cumulative Fuel Consumption.

Chapter 7

Scenarios

A scenario is a pre-planned sequence of actions that was put together in order to showcase a specific feature of the system. They have different levels of increasingly difficulty and show both particularities of the traffic control and also the anti-platooning. This chapter does a description of each scenario with comics strips as visual aids. The scenarios are in fact essential for the experiments, not only because they make a good showcase, but also because they are a consistent way of obtaining results that can be later analyzed.

7.1 Scenario Creation

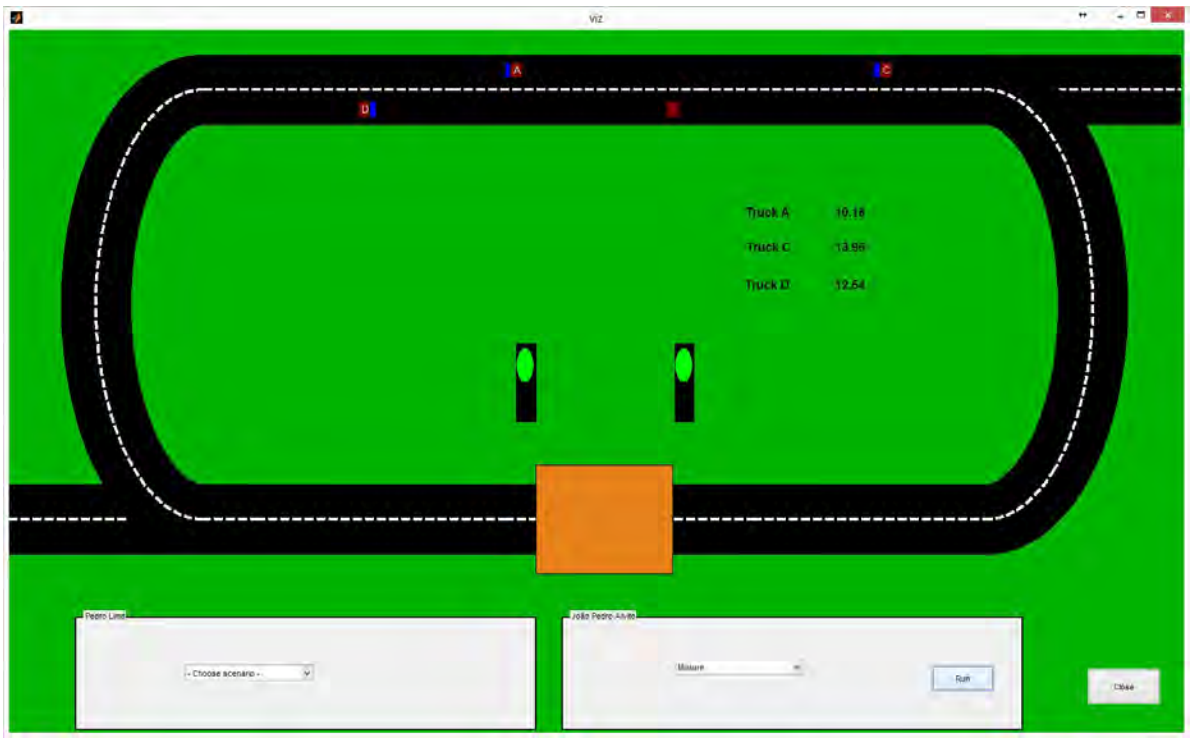


Figure 7.1: Scenario visualization tool.

When formulating the scenarios, it was created a tool that helped this task, where simple simulated trucks reproduced the actions that are to be implemented in the real trucks. As it is possible to see in Figure 7.1, the said tool has various components: colored objects representing the trucks or other type of traffic; lanes emulating the road network created in SML; traffic lights that control the traffic flow in the bridge area; in some scenarios it is also possible to see the ETA of each truck to the respective traffic light. This tool was also done in a partnership with other master student, the scenarios selection box has scenarios that are only used in [10]. Obviously there will one be explained the ones related with this project.

7.2 Scenario Implementation

A brief description will be presented for each scenario, in which are presented the main features of the scenarios and of course what are their main goals. This is a presentation of what kind of scenarios can be implemented in the testbed, various variations of the scenarios can be created depending on the number of trucks available and the final result that is intended. After the creation of the scenarios the next step is to implement them in the testbed in the SML. It is now possible to analyze the feasibility of the scenarios and if they bring any benefit to the problem that was described. The first step in every scenario is to place the trucks in the starting positions, after that, the scenarios run normally as scripted.

7.2.1 Platooning - Description

The first scenario is the simplest one, where the main goal is to show the small features with platooning. The experiment consists of two trucks running in the same lane (Figure 7.2). One of the trucks starts first and creates a gap to the second truck. After a while the convoy starts to do platooning and keeps a fixed platooning distance that after a period of time changes for a smaller value.

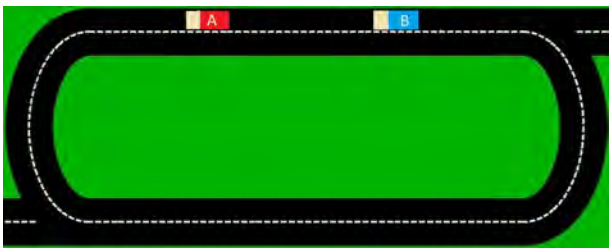


Figure 7.2: Comic strip of platooning scenario.

7.2.2 Platooning - Results

In this situation we have two trucks that when they decide to form a platoon, the truck that is in the back slows down to achieve the reference value for the the platoon distance. Once the distance between the trucks stabilizes the reference value for the platooning distance is lowered (Figure 7.3).

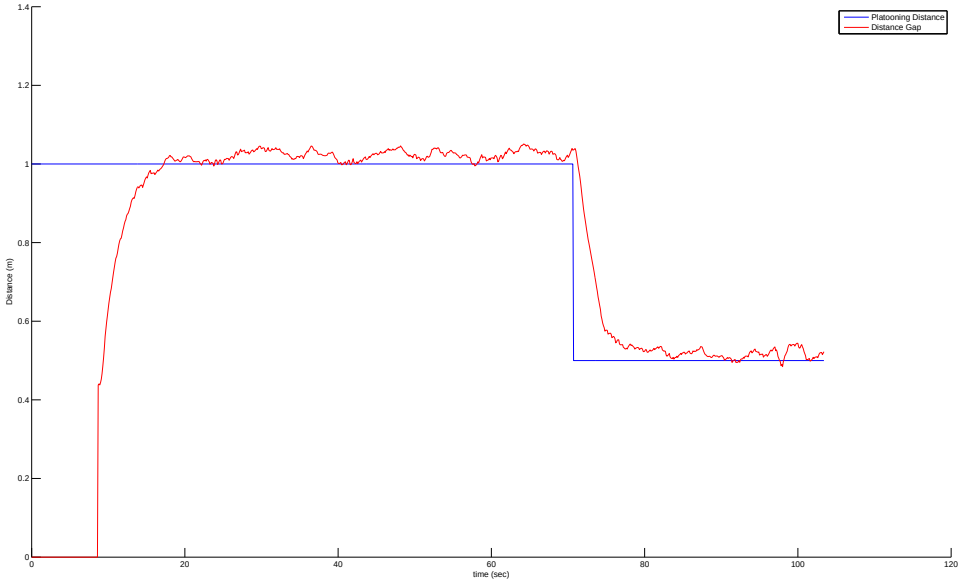


Figure 7.3: Reference platooning distance and real distance between trucks.

Now, since the reference value represents a smaller gap between trucks, the truck on the back starts to increase its velocity in order to catch-up with the leader of the platoon. Once again, after a small period of time the gap between trucks will stabilize around the reference value for platooning distance.

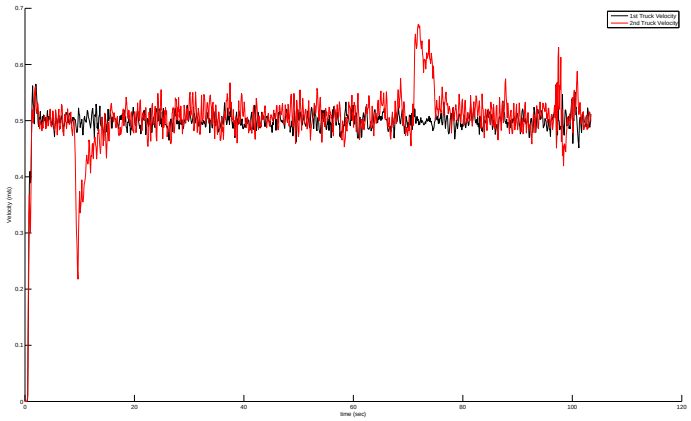


Figure 7.4: Velocities of the trucks.

This scenario is basically a follow up of the proof of concepts examples. It is the first introduction of scripted actions to the system that will increase in complexity in the next examples. Although this may be a very simple scenario, it represents the foundation of the system.

7.2.3 Traffic Lights - Description

This scenario is as well very simple. It is focused in the existence of traffic lights in the system. The goal is to study, not only the reactions of a single truck to the change of the traffic light status, but also the reactions of a platoon when they encounter a traffic light with red status. The control of the vehicles is done in the same way as the last example despite the trucks being by themselves, i.e. they are not part of a platoon. However, now the system responds to stimulus of a digital signal that represents the status of the traffic light. So to simulate a real situation (no information about the traffic light other than visual information), the truck starts to break before reaching the traffic light localization. Note that the system has a mechanism that allows platooning while waiting in the traffic light. This way is possible to do simultaneously platooning and traffic control managing.

7.2.4 Traffic Lights - Results

This scenario was created to show the effect of the traffic lights in the behavior of a truck and also in a platoon. Note that both Figure 7.5 and 7.6 have the traffic lights status represented as a digital signal. When the signal is active, it means that the traffic light is red.

Single Truck

In Figure 7.5 is represented the velocity of the truck in this experiment.

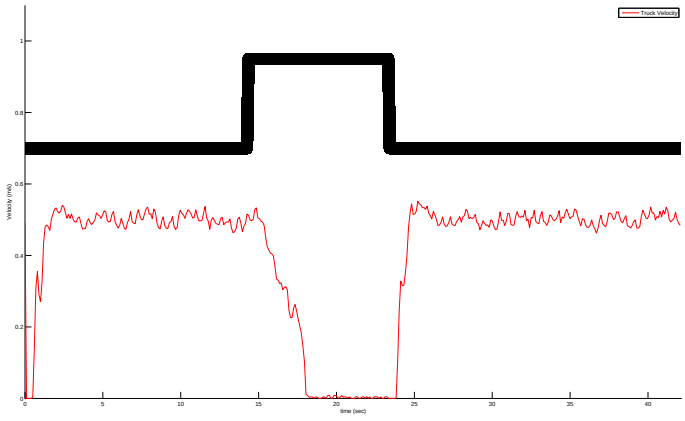


Figure 7.5: Velocities of the truck.

The system instead of having the truck to do a full break when the truck reaches the traffic light, it slows down the truck and that way the truck approaches the traffic light with more stable speeds.

Platooning

In Figure 7.6 is represented the velocities of the trucks in the platoon in this experiment.

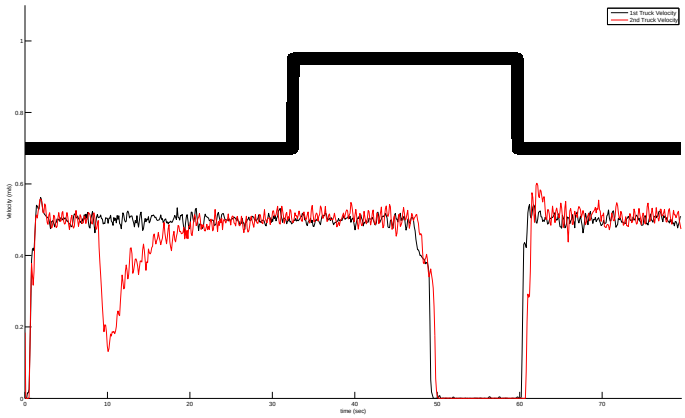


Figure 7.6: Velocities of the trucks.

The implementation was once again successful, where the platoon approaches the traffic light area. The system gives priority to the traffic light status, i.e., if the traffic light is red then the platoon will stop. However the formation will still be maintained, even when the platoon is stopped.

7.2.5 Anti-platooning - Description

The setup in this scenario is the same in Figure 7.2. Once again, two trucks are forming a platoon with a certain distance separating the trucks. However, in this case the distance is the anti-platooning distance that is a variable distance, that is calculated in order to keep a reference time gap between the trucks. So the controlling system is similar but this time the system doesn't take the distance between vehicles as reference, but as a parameter to achieve the reference time gap between trucks.

7.2.6 Anti-Platooning - Results

This experiment showcases the anti-platooning feature with two trucks forming a platoon separated by the anti-platooning distance.

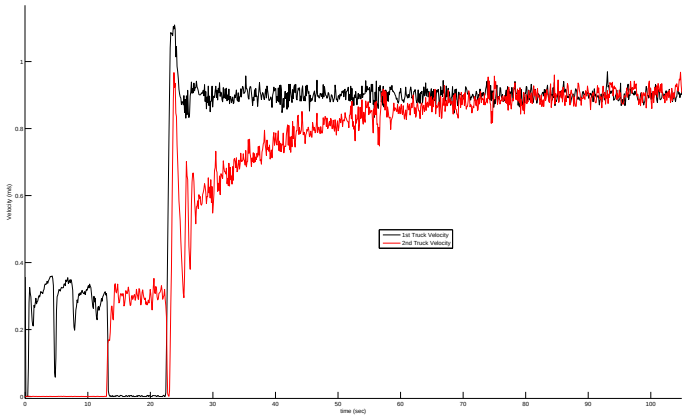


Figure 7.7: Velocities of the three trucks.

The anti-platooning distance is basically a distance that is calculated in order to keep the trucks in the platoon separated by a time gap, that is fixed during the entire experiment (7.1).

$$AntiPlatooningDistance = TruckSpeed * ReferenceTime \tag{7.1}$$

In this experiment it was used a time reference of 10 seconds. In (7.1), the truck speed corresponds to the velocity of the truck that is in the back of the platoon. A representation of the platooning distance is in Figure 7.8.

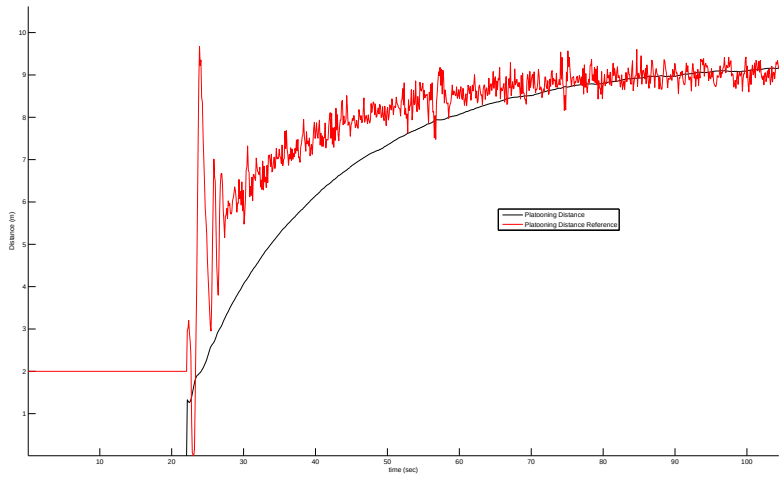


Figure 7.8: Platooning distance reference and real distance between trucks.

In Figure 7.9 is represented the time distance between the trucks in the platoon.

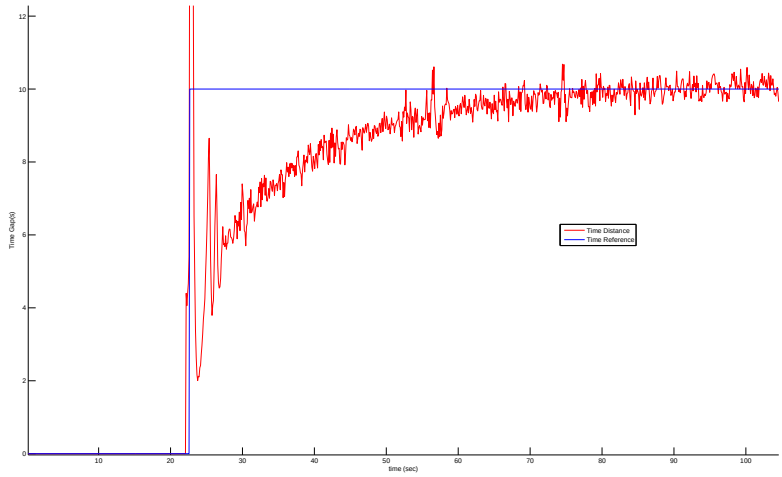


Figure 7.9: Time distance between trucks.

7.2.7 Anti-platooning in Mines - Description

This scenario is an enactment of the same exact situation that is happening in the mines of the north of Sweden. The scenario starts with the three trucks parked in the same area, this represents as if the trucks were parked in the mines waiting for each other to leave. Then the first truck has permission to leave the mine and it starts running at constant speed. After a while the second truck has also permission to leave, when it starts it will try to catch-up to the first truck and keep an anti-platooning distance.

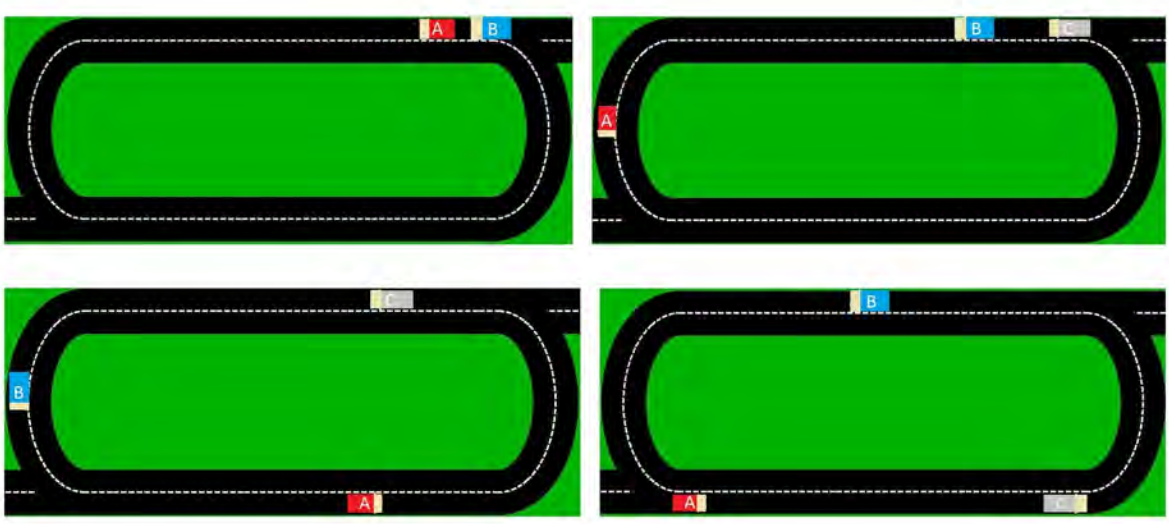


Figure 7.10: Comic strip of anti-platooning scenario.

Finally, the third truck has also permission to leave and it will try to catch-up the the second truck and keep the same anti-platooning distance. Then the three trucks will be running at constant speed and with fixed distance between every truck. A more graphic description of the scenario is represented in Figure 7.10.

This scenario was designed to create the simplest of the situations where the anti-platooning feature could be showcased. Consciously, the scenario was created with trucks separated with distances much smaller than the ones that would happen in a real situation. This choice was made because due to the limitations of the dimensions of the road network, it is not feasible to replicate the same exact dimensions of the real scenario.

7.2.8 Anti-Platooning Mines - Results

This scenario, as previously described, has as main features that should be analyzed, the anti-platooning distance and the velocities. Since they are keeping a steady formation, the goal is to maintain both the velocity and the anti-platooning distance as stable as possible.

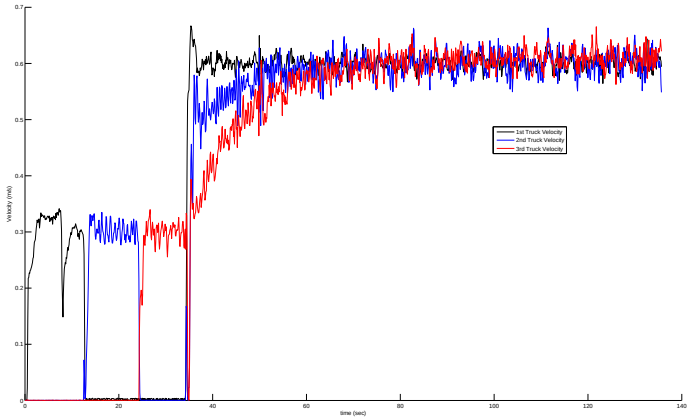


Figure 7.11: Velocities of the three trucks.

For a matter of simplicity in the implementation, the trucks were first positioned in the road in a fixed position and then they all started already separated by a fixed distance. As it is visible in Figure 7.11 after the trucks are positioned in the starting position, all the three trucks start to accelerate to the reference velocity until they stabilize around that velocity. Note that for the rest of the experiment the trucks maintain that velocity.

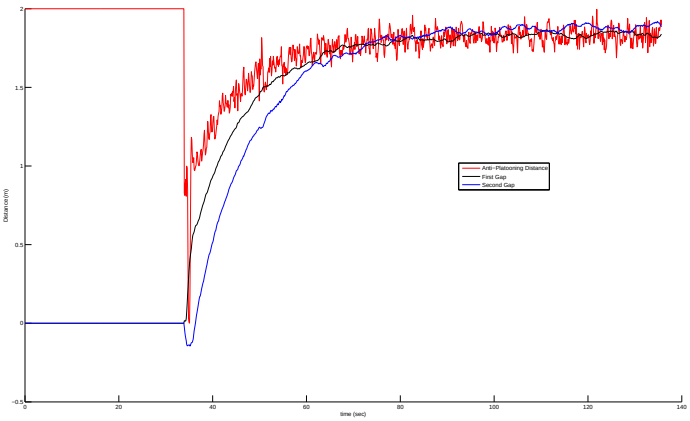


Figure 7.12: Anti-platooning distance.

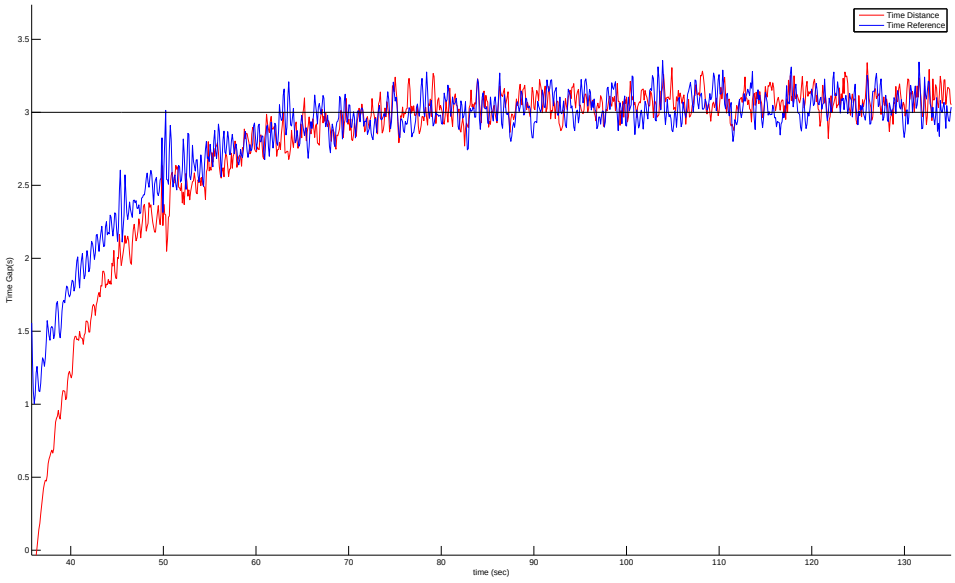


Figure 7.13: Time gap between trucks.

In Figure 7.12 the evolution of the gaps sizes between trucks during the experiment is represented. It is visible that both gaps are stable, and only have small oscillations around the reference platoon distance, that have minimum effect in the stability of the formation.

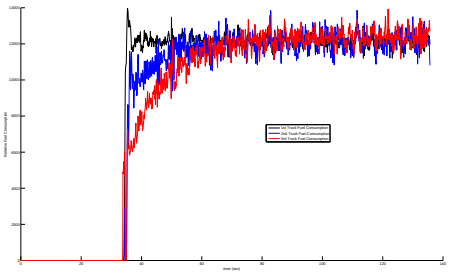


Figure 7.14: Instant fuel consumption.

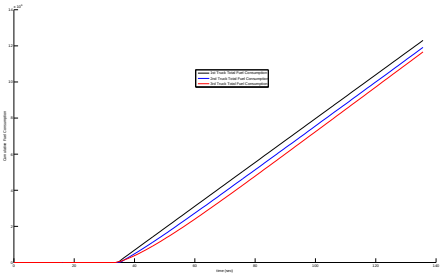


Figure 7.15: Total fuel consumption.

The Figures 7.14 and 7.15 confirm the expectations that all three trucks are consuming roughly the same amount of fuel. This happens because even though the trucks are making a platoon formation there is no fuel savings because they are in an anti-platooning situation, and so the gap between each truck to wide.

7.2.9 One truck and one car - Description

This scenario consists of one truck in one lane, and a car in a different lane. These two lanes have an area that is controlled by traffic lights that only allows one vehicle at a time in that area. This simulates a bridge or a narrow passage where only one vehicle can pass through at a time. Accordingly to the

problem definition the stoppage of a HDV must be avoided, this way if two vehicles meet in opposite sides of the traffic lights, the priority of passage belongs to the HDV and not the car.

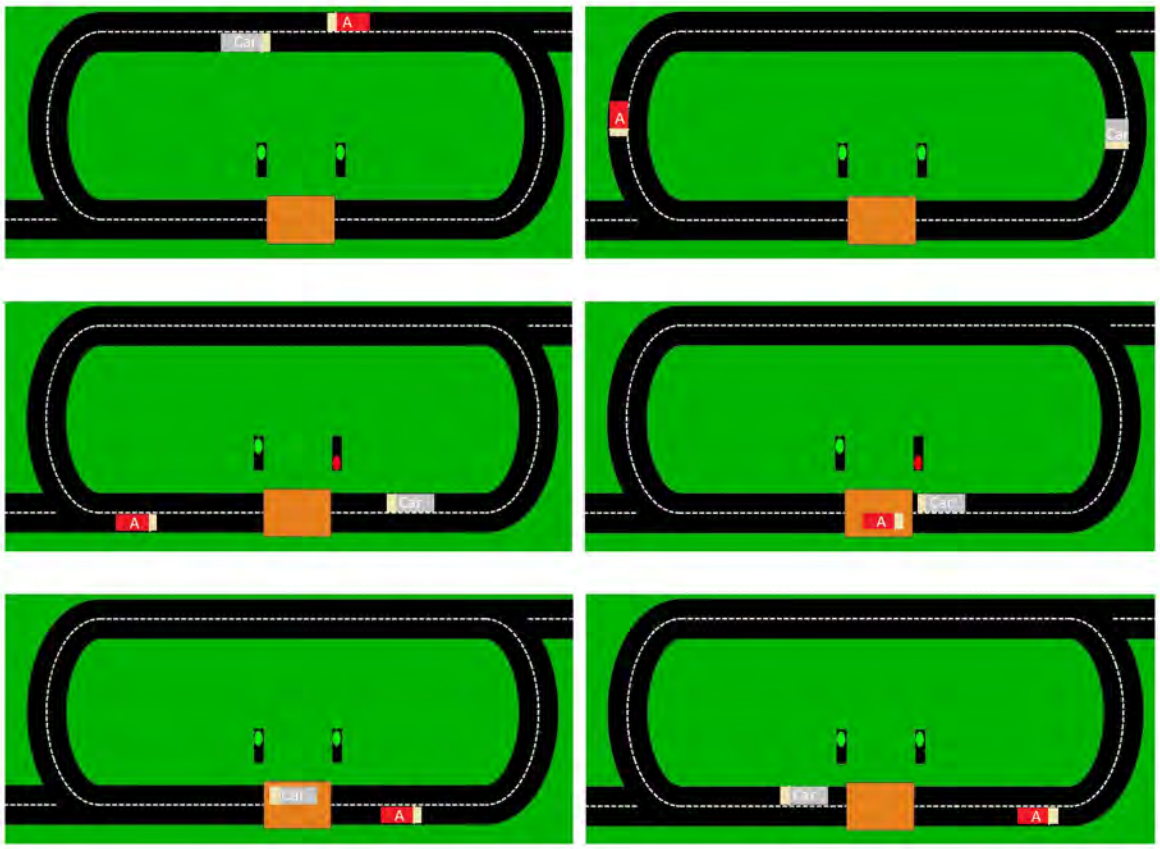


Figure 7.16: Comic strip of scenario with two trucks and traffic lights.

This setup assumes that the system has a centralized subsystem that monitors all the events and has knowledge of the position of every truck. So it is able to communicate with the trucks and also to change the status of the traffic lights.

The simulation starts with the two vehicles in the same area of the road but in different lanes, and they start running simultaneously. At some point the vehicles start to approach the area controlled by the traffic lights. When the system detects that the truck is coming close the controlled area, it changes the status of the traffic lights in order to keep the traffic light green in the lane of the truck, and changes to red the traffic light of the opposite lane. This way it allows the truck to pass through, and at the same time it prevents the car that is in the other lane to go as well to the controlled area. After the truck is cleared from the controlled area, it is now possible to the car to continue its trajectory, so the system changes the status of the car's lane traffic light back to green. This sequence of actions is shown in Figure 7.16.

7.2.10 One truck and one car - Results

To analyze this simple scenario, two separate experiments were done. Th first experiment, basically, had the car and truck running simultaneously, and when both vehicles approached the traffic lights area the car was stopped and then the truck proceeded its trajectory until it reached the starting point. Then the second experiment was very similar, but this time, it was the truck that came to a full stop. After some time the traffic light turned back to green and the truck proceed, once again, to the starting point. So summing up, there are two different experiments were a truck travels the same exact distance, but in one of those experiment it stops in a traffic light. The energy that was used to position the trucks in the initial positions was not considered.

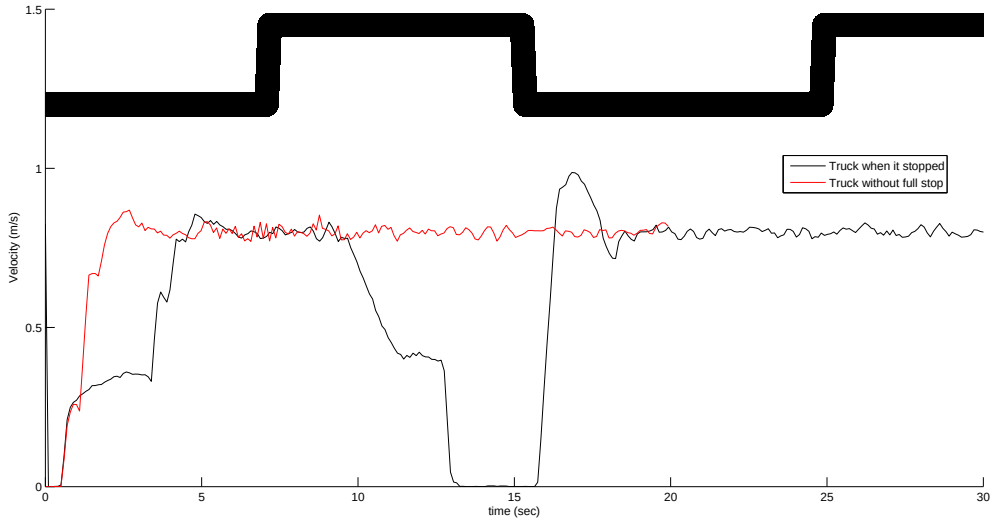


Figure 7.17: Velocities of the two experiments.

In Figure 7.17 we have a confirmation of the description, where it is possible to observe the speed of the truck in both experiments. The black digital signal represents the status of the traffic light, that when is active it means that the traffic light is red, and so the trucks must stopped if it approaches the traffic light.

Now we can compare the two experiments in terms of fuel consumption and that way understand what is the effect of a full stop on the fuel consumption. In Figure 7.18 it is represented an overview of the instantaneous fuel consumption for both experiments. It is clear that when the truck doesn't stop, the experiment goes by much faster. On the other hand when the truck stops there is a moment where the truck is decelerating, so it would be expectable for the truck that has to stop, a lower fuel consumption. However, as was said before, if a truck has to stop the amount of fuel spent to restart the movement is enormous.

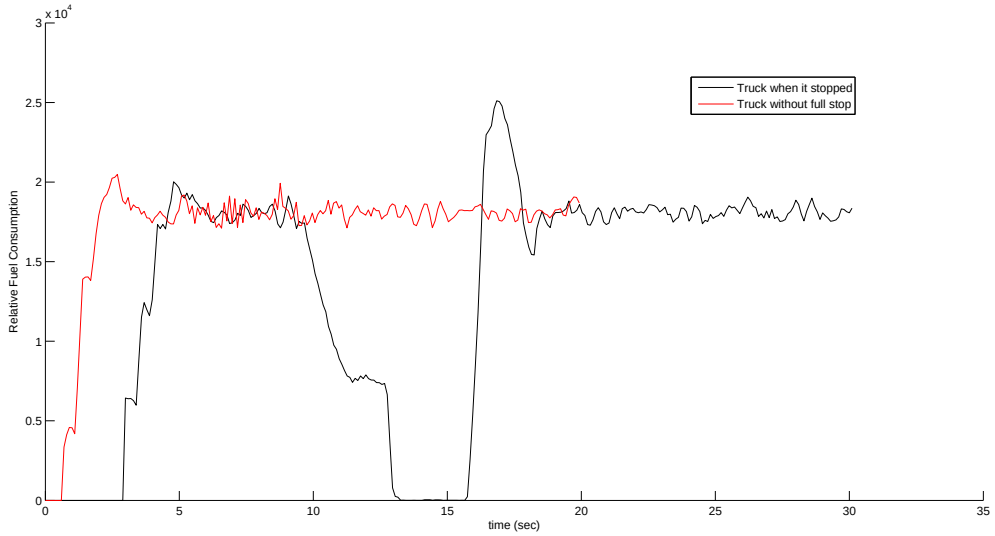


Figure 7.18: Instant fuel estimation of the two experiments.

Having said that, in Figure 7.18 it is the final piece of information to that allows to draw the conclusion that the truck spends more fuel if it has to stop in a traffic light. In this experiment in particular, the reduction of total fuel consumed is roughly 10%.

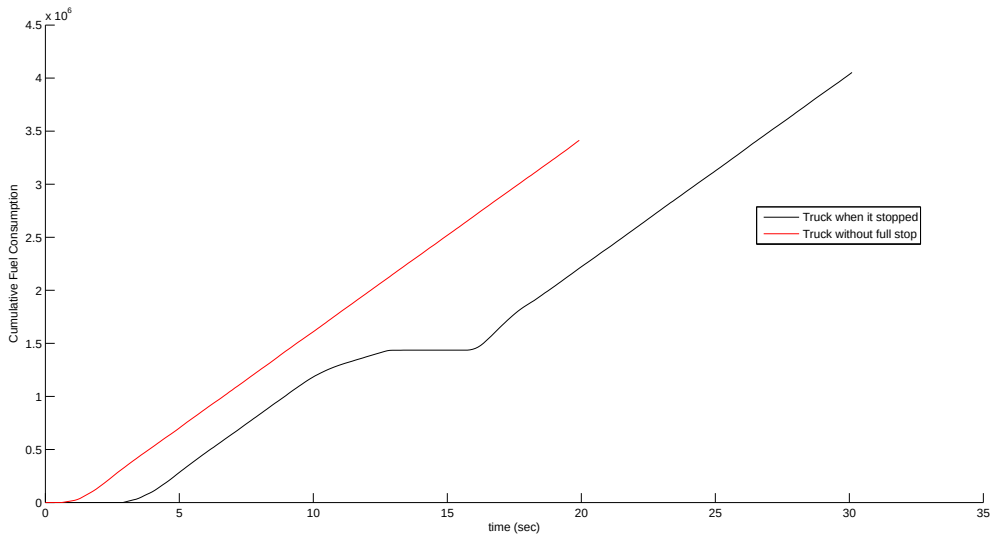


Figure 7.19: Total fuel estimation of the two experiments.

7.2.11 Two trucks and one car - Description

This scenario is similar to the scenario in section 7.16, but in this case there is an addition of a new truck that is positioned right behind the car.

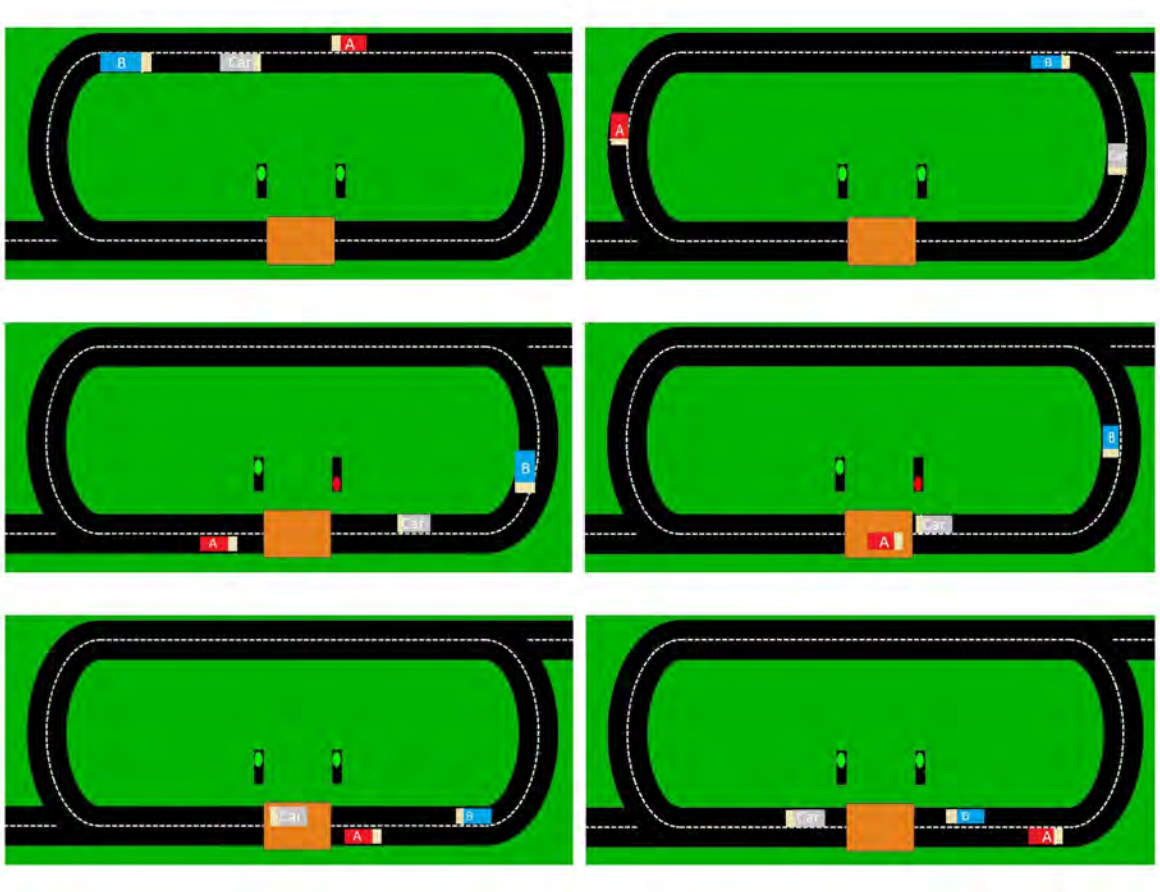


Figure 7.20: Comic strip of scenario with two trucks and one car with traffic lights.

This means that now, the decision making has to be updated, because stoppages have to be avoided simultaneously for both trucks. So the starting positions are the following: one truck is located in one lane, and the car and the other truck are positioned in the other lane. Once all vehicles are positioned, they start running simultaneously until the moment the truck that is running alone in its lane approaches the area controlled by the traffic lights. When this happens, the system immediately turns the traffic light where the car is located to red, this of course makes the car stop at the traffic light. However this would affect the truck that is running behind the car. So the system alerts the truck about the change of the traffic light and advises it to slow down. By slowing down, the truck naturally takes more time to reach the traffic light, allowing the other truck to pass through the traffic lights area, and at same the time, it does so without having to stop. After that both traffic lights turn green and the traffic starts to flow normally.

7.2.12 Two trucks and one car - Results

Since this is an improved setup from the one studied in section 7.2.10, it is only worth to analyze the improvements from that scenario. Nevertheless the analysis from that setup holds up in this situation, and its conclusions are also valid in this case. The most important feature to analyze is of course the decision making applied. For that, it is represented in Figure 7.21 the speed of the three vehicles. Also the black digital signal represented in the plot shows the status of the traffic light (from the lane where the car is, because it is the one that changes the status during the experiment). If the signal is activated it means that the traffic light from that lane is red.

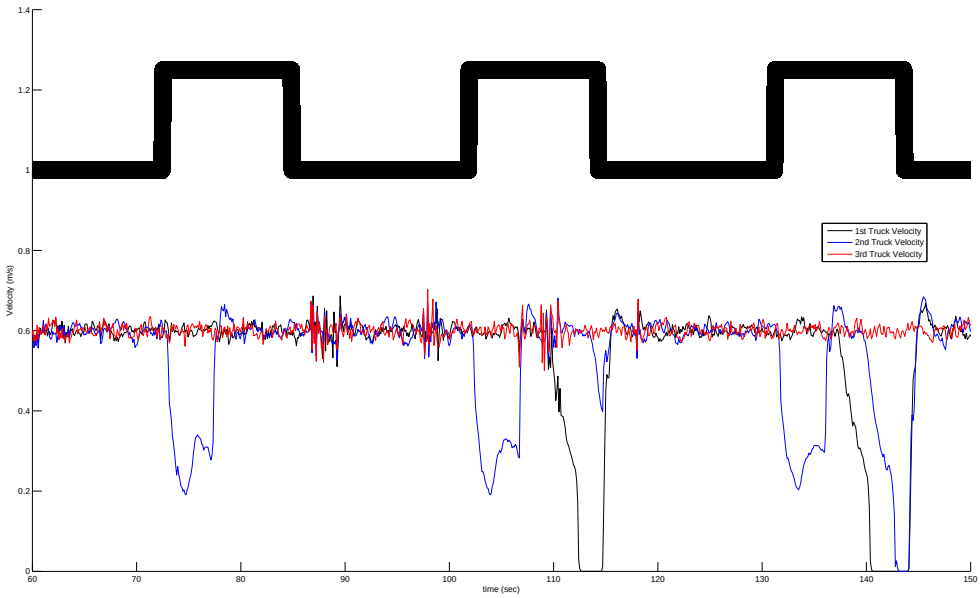


Figure 7.21: Velocities of the three trucks.

The plot in Figure 7.21 shows the trucks running for three loops in the road network, where we can see the three possible cases in this decision making system. When the traffic light goes to red for the first time, everything goes as planned. The truck that is running alone doesn't stop, neither the truck that is behind the car (that doesn't stop as well), it only slows down as explained in section 7.2.11.

The second time the traffic light turns to red, the truck that runs alone, once again, doesn't stop. But this time the car has to stop because it encountered a red traffic light. As planned, the other truck slows done when getting closer to the traffic lights, but it doesn't need to stop.

Finally in the third time the traffic light turns to red, the truck that is alone doesn't stop during loop. And also, once again the car has to fully stop, but as said that is not a problem, because the idea here is only to avoid stoppages of trucks. Lastly, the truck behind the car doesn't have enough distance between itself and the car, in order to avoid a full stop.

So this means that this decision making system works only if the gap between the truck and the car

is sufficiently large to avoid a full stop from the truck. If not the system seems to fail to avoid full stops in both trucks running in the experiment.

7.2.13 Scenario - One truck one car with GLOSA

This scenario has as similar setup as the one in section 7.2.9. Nevertheless, the decision making system for the traffic lights status is updated. Now instead of just changing the status of the traffic light for the lane where the truck is, there are two trucks (one in each lane), and the system calculates where is the truck closest to the traffic light, and changes the traffic light of the other lane to red. Also the velocities of the trucks are updated in order to allow both trucks to go through the traffic lights area without having to stop. This system is called GLOSA (Green Light Optimal Speed Advisory).

7.2.14 One truck one car with GLOSA - Results

With this new decision making mechanism we can see some improvements on the efficiency of the traffic lights system. Once again, the status of the traffic light is represented with a black digital signal in Figure 7.22. As expected the system detects which car comes closer to traffic light area and turns the traffic light status of the other vehicle's lane to red. At the same time is possible to see that the other vehicle reduces its speed to a certain speed. That speed is calculated as the velocity that allows the truck to reach the traffic light area without having to stop.

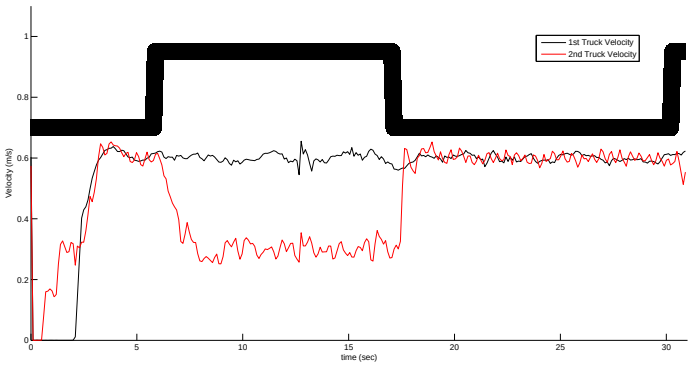


Figure 7.22: Velocities of the two trucks.

In Figure 7.23, it is possible to see the instantaneous fuel consumption of the trucks during the experiment. As expected the total fuel consumed is lower than the one presented in section 7.2.10.

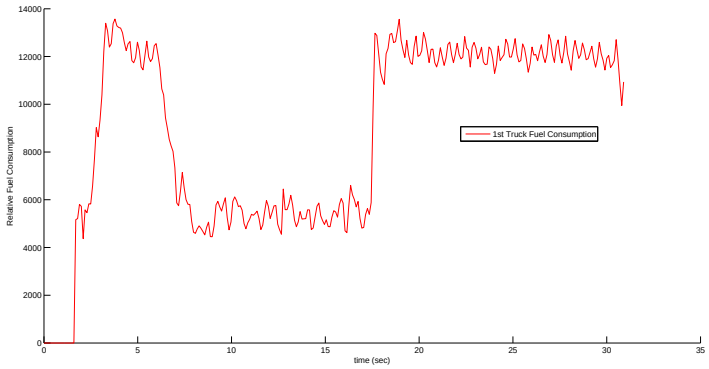


Figure 7.23: Instant fuel consumption.

This is the scripted decision making system implemented that satisfy the restrictions proposed in the problem definition. Since it controls both the traffic lights status and the speeds of all the trucks. Of course that it is safe to say that for a small number of trucks with the dimensions of our road network the system works normally. However it is safe to predict that if the road network becomes crowded there might appear some situations were a truck cannot avoid a full stop. But we cannot forget that if something like that happens in real life we also do not have any control over it.

Chapter 8

Conclusions

We have to be careful when doing some relations between the simulations that were done in this project and the real system that could be implemented in the HDVs in Figure 8.1. The resulting conclusions can not be applied directly on the real system, nevertheless they are a hint of what we would see in the real trucks if we applied the system implemented in this project.

Figure 8.1: Real HDV that will be used (courtesy of Scania).

8.1 Conclusions

In the end the project created a system the showcases several scenarios with traffic lights that are controlled in different ways in order to analyze the effectiveness of each decision making system. We can conclude that the GLOSA system is the most efficient one, however it is the one that implies more complexity if applied in the real system. With GLOSA the trucks never reach a full stoppage which obviously brings several fuel consumption benefits.

The project was a great help to build the current testbed, that now enables the creation of future projects that might be based in the project's subject. In Appendix A was also created a manual that is

essential for a better understanding of the system created with more practical details.

Resuming, this project was a successful implementation of the problem proposed, where the results obtained are a motivation to further pursue this problem and of course improve the implementation in order to to get even better applications and results.

8.2 Future Work

The project is far from to be complete, as commonly happens even if you achieve your project goals there are always features left to implement.

1. One small feature that can be added in the future is the implementation of physical traffic lights in the system.

2. The controller studied should be improved, although it shows very good results for the presented situations, when subjected to more extreme situations it starts to show some erratic behavior. One idea is to develop a trajectory controller that integrates the velocity and the steering in one single controller.

3. One good idea for the future is to implement virtual traffic in the main system running in Labview. This way we could simulate normal traffic, as it would exist in a real road. And the idea is to make interactions between the simulated traffic and the trucks running in the SML.

4. Ultimately, the main goal would be to implement the system in a full sized HDV, of course this by itself is a big project. However the implementation of a simulated system in a real life situation is the end goal of a project. It is the culmination of a journey that started from a simple idea and should finish in a complete product.

Bibliography

[1] Qualysis AB. *QTM Getting Started*, 2011.

[2] Qualysis AB. *QUALISYS MATLAB CLIENT v1.8*, 2011.

[3] Qualysis AB. *QTM User Manual*, 2011.

[4] Assad Alam. *Fuel-Efficient Distributed Control for Heavy Duty Vehicle Platooning*. PhD thesis, KTH Royal Instituteof Technology, 2011.

[5] J. P. Alvito. Implementation of traffic control with heavy duty vehicle anti-platooning. Master's thesis, KTH The Royal Institute of Technology, 2013.

[6] M. Amoozadeh. *Smart Mobility Lab Manual*, 2013.

[7] A. Hauksson, B. Mengana, C. Westermark, F. Svensson, J. Lycke, J. Sundberg, K. Imhäuser, and S. van de Hoef. Final report in automatic control project course. Technical report, KTH The Royal Institute of Technology, December 2012.

[8] A. Hernandéz and D. Huertas Péres. Communication between pc and motes is tiny os. Technical report, KTH The Royal Institute of Technology, may 2012.

[9] Liang K., J. Mårtensson, and K. H. Johansson. When is it fuel efficient for a heavy duty vehicle to catch up with a platoon? Submitted to IFAC ACC 2013, 2013.

[10] P. Lima. Heavy duty vehichles platoon catch up scenarios implementation an analysis. Master's thesis, KTH The Royal Institute of Technology, 2013.

[11] Daniel Macias. Estimation of fuel consumption for real time implementation. Master's thesis, KTH The Royal Institute of Technology, 2012.

[12] Alejandro Marzinotto. Cooperative control of ground and aerial vehicles. Master's thesis, KTH The Royal Institute of Technology, 2012.

[13] Per Sahlholm. *Distributed Road Grade Estimation for Heavy Duty Vehicles*. PhD thesis, KTH Royal Instituteof Technology, 2011.

Appendix A

User's Manual

A.1 System Overview

A block diagram of the system structure can be seen in Figure A.1.

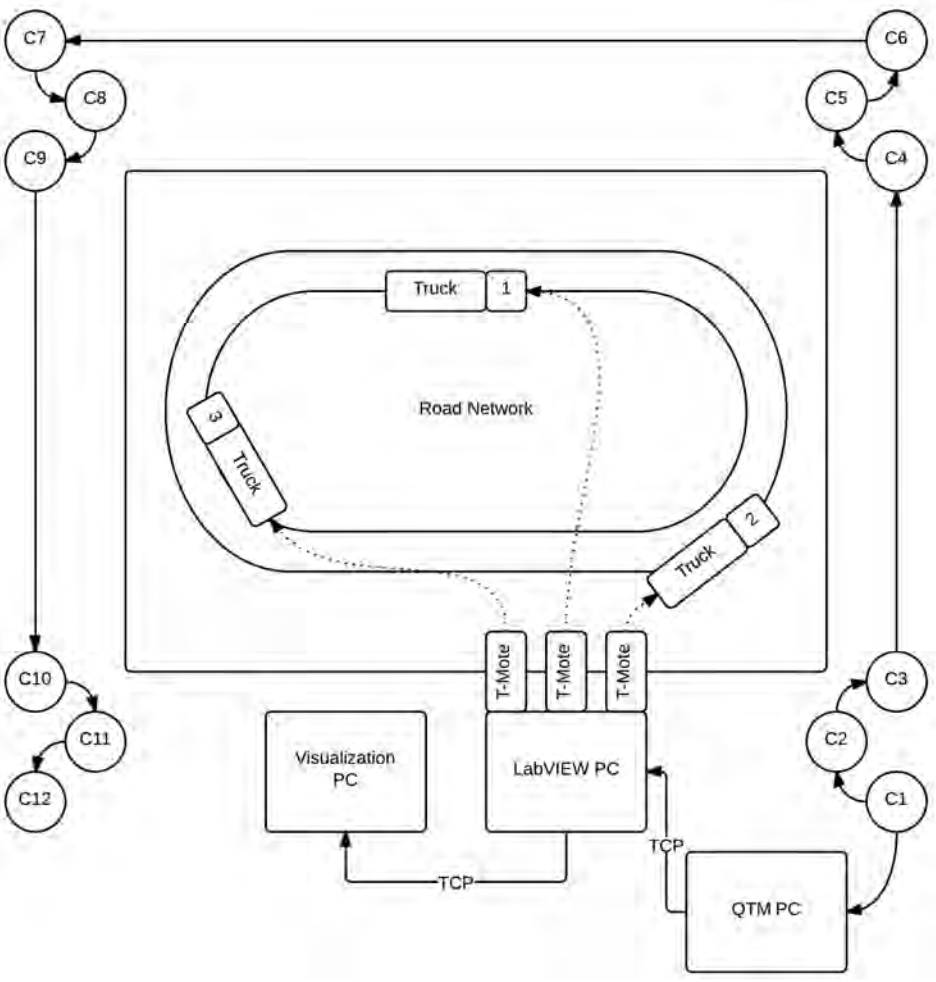


Figure A.1: Program structure block diagram.

The overall system is supposed to represent a **real-world system** in **small scale** in which the **cameras** (C1,C2,...,C12) simulate the **GPS system**, the **LabVIEW PC** simulates an **onboard computer** responsible for the **decision making** of the truck and, naturally, the **scale trucks** represent the **real trucks**. The **T-Motes** exist in order to make possible the communication between the PC and the trucks. Moreover, the QTM PC is the server that provides the **trucks' localization** retrieved by the cameras. Finally, the **Visualization PC** is used for **demonstration** purposes.

This system was developed in the first semester of 2013 in the **Smart Mobility Lab of the Automatic Control Department in KTH** in Stockholm under the supervision of **Jonas Mårtensson** and **Karl Henrik Johansson**. It is the **kernel** of two master's thesis, since it is the **test bed** for the experiments reported on them (see [10] and [5]).

A.1.1 Key Features

With the set of programs developed the user is able to:

- Use the IR markers to **create a set of waypoints** that possibly represents a trajectory, then **acquires** those points and the correspondent **interpolated trajectory** with more waypoints (see Chapter A.4);
- Define the trucks as being **6DOF bodies** using the Qualysis' QTM software. This way it is possible to **track** them in **real time** with 6DOF (see Chapter A.3);
- **Use the trucks** and do a wide range of operations from **platooning**, **platoon catch up** and **overtaking** to **traffic control with traffic lights** (see Chapter A.5);
- Run a **visualization tool** that allows the user and the target audience to easily understand what is happening (see Chapter A.6).

A.1.2 Who are we?

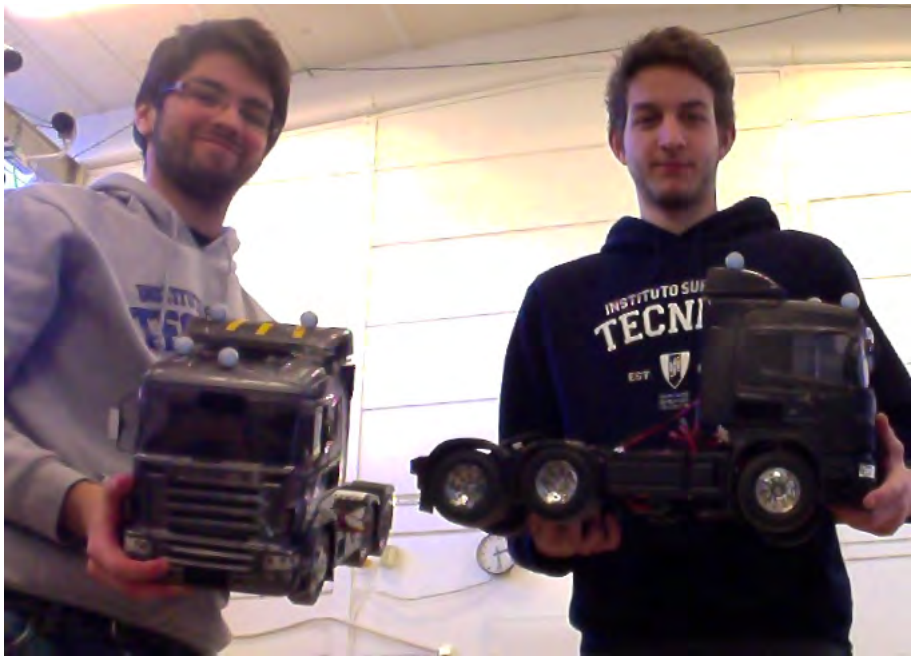


Figure A.2: The authors: Pedro Lima on the left side and João Pedro Alvito on the right side.

We are two portuguese **master students** in **Electrical and Computers Engineering** from **Instituto Superior Técnico**, Lisbon, Portugal. We both came to KTH under an agreement between IST an KTH to take the **Master Program in Systems, Control and Robotics** (TSCRM) at KTH as part of a **Dual Degree**. This agreement includes doing our master thesis in KTH.

If you want to **know more** about this project or if you have any **doubts** don't hesitate to **contact us** (`pfrdal@kth.se` and `jpfa@kth.se`).

A.2 Getting Started with the Tamiya Trucks

The **Tamiya Scania trucks** are 1/14 **scale trucks** of the real Scania V8. They are naturally one of the most important parts of the overall system.

This chapter provides a **basic tutorial of how to connect and start the trucks** and some **troubleshooting**. For a more detailed description of the material involved please refer to [6].

A.2.1 T-Motes set up

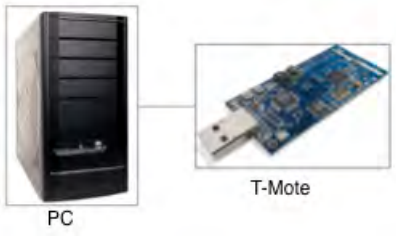


Figure A.3: Connection between a PC and a T-Mote (courtesy of [6]).

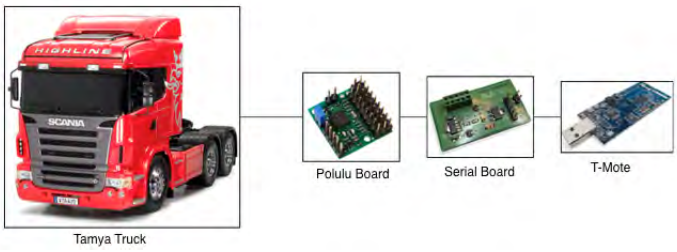


Figure A.4: Connection between a truck and a T-Mote (courtesy of [6]).

In order to communicate with the trucks from a PC you need to know how to use the T-Motes. In the project we use the **serial-forward tool** to send data from LabVIEW to the mote attached to the PC, and the **Tiny OS** to program them. In order to learn how to install the Tiny OS and serial-forward tool please refer to [8].

In **Windows**, after having the **Cygwin** and the **serial-forward tool** installed, you should be able to see the T-Motes connected to the PC (like in Figure A.5) using the `motelist` command.

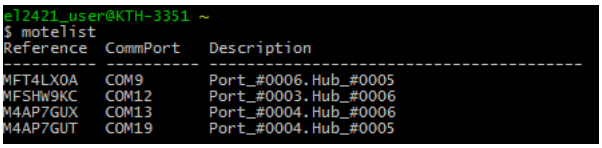


Figure A.5: `motelist` command result on a Cygwin terminal.

The T-Motes usually have its **reference number** (for example: MTF4LX0A) printed on it. If they don't, please connect **one by one** to know which COM port corresponds to each them. Then, it is possible to start a serial forward of a specific T-Mote in a specific **TCP/IP Port** like in Figure A.6. The port chosen has to be a suitable port that is not being used by any other device. The most common ports used when serial forwarding are 9002, 9003, 9004... When trying to communicate, if everything is **okay**, you should start seeing something similar with Figure A.7.

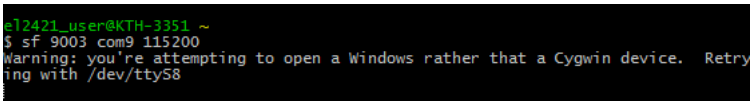


Figure A.6: `sf` command result on a Cygwin terminal.

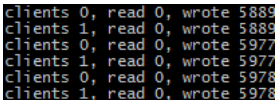


Figure A.7: `sf` normal execution on a Cygwin terminal.

It is very important that you pay a lot of attention when doing serial-forward of a T-Mote in order to be able to communicate correctly with a truck. A T-Mote connected to the PC has a unique pair connected to a truck. The T-Motes used so far for that purpose are all labeled with "<COLOR> PC <ID>" and "TRUCK <COLOR> <ID>", where <COLOR> is the **color of the truck** and the <ID> is the **communication ID**[1] of the truck. The **communication ID has to be inputed in the LabVIEW program** (see Chapter A.5).

If you receive messages like "Note: write failed" (see Figure A.9) or "unix_error" (see Figure A.8), try to kill the serial forward with CTRL+C inside of the Cygwin terminal and restart the serial forward again. If it still doesn't work, try to disconnect and connect the corresponding T-Mote and restart the serial forward. Restart the PC if nothing else works. **Always check if the IDs and COM ports you inserted in the LabVIEW program correspond to the T-Motes you are using.**

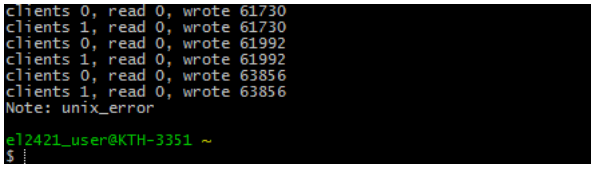


Figure A.8: `sf` "unix_error" example.

[1] It is not the same thing as the radio channel number!

```
Note: write failed
Note: write failed
```

Figure A.9: `sf` "Note: write failed" example.

If everything is okay you should see the **T-Motes' leds blinking stressfully** when you try to transmit something through them.

A.2.2 Trucks Set Up

The **truck connections** are exemplified in figure A.10 and A.11.

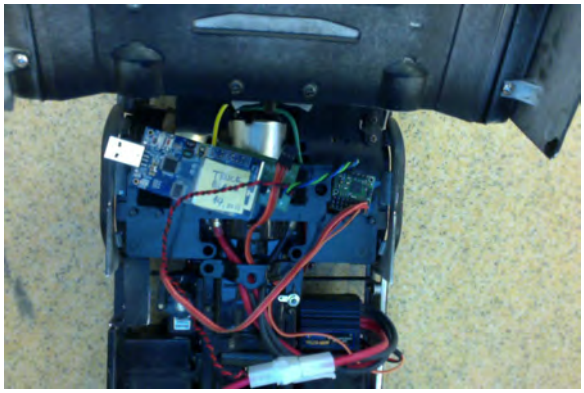


Figure A.10: Connections between each component in a truck.

Figure A.11: Detailed connections between a T-Mote and a Polulu board (courtesy of [6]).

The following connections have to be made on the truck:

- Connect the **2-pin female jumper wire** of the **power cable** to the **serial board**. **Pay attention to the polarity of the board**;
- Connect the **2-pin female-female jumper** between the **serial board** and the **Polulu board**. **Pay attention to the polarity of the board**;
- Connect the **3 wire cable** of the **speed servo** to the **Polulu board**. **Pay attention to the cable order**;
- Connect the **3 wire cable** of the **steering servo** to the **Polulu board**. **Pay attention to the cable order**;
- Connect the **T-Mote** to the **serial board** so that the **rightmost 10 pins** of the T-Mote are connected;
- Set the **motor switch** to **"On"**;
- Connect the **battery female connection plug** to the **male connection plug of the power cable**.
- Connect the **power cable female connection plug** to the **speed servo male connection plug**.

When everything is connected the **yellow LED** should turn on on the **Polulu board**. If you see a **red LED** this can mean that the **batteries are low**, that you **misconnected some of the cables** or the **T-Mote is misconnected to the serial board**.

When you try to communicate with the T-Mote that is on the truck you should see a **blinking blue LED on the T-Mote** and a **blinking green LED on the Polulu board**. If you don't see anything happening, **check if the IDs and COM ports you inserted in the LabVIEW program correspond to the T-Motes you are using.**

To charge the batteries you should connect the battery to a battery charger.[2] The charging rate must be $0.5A$ and it takes about 10 hours for the battery to be fully charged. For safety reasons, **it is important to never forget a charging battery while no one is around**.

A.3 Getting Started with QTM

Qualisys Track Manager (QTM) is a Windows-based data acquisition software with an interface that allows the user to perform 2D and 3D motion capture.

This chapter provides a **basic tutoria**l of how to use QTM in order to **define a truck as a 6DOF rigid body**. For a more detailed description of the software please refer to [1] and [3][3].

A.3.1 How to Define a truck as a 6DOF Body

The trucks to be tracked are equipped with **markers**. These are small IR-reflective spheres placed on the top of the truck. A **minimum of 3 markers** is required in order to define a **6DOF Rigid Body**. We placed 4 markers per truck so that the trucks are more robustly tracked. Each truck has its unique marker configuration in order to be unequivocally recognized.

In order to define a 6DOF body you can follow the **next steps**.

- **Step 1**: Open the **"6DOF Tracking" page** that can be found in the **"Project Options" dialog**. There you will see the already defined 6DOF Rigid Bodies, define new ones or load old ones. To acquire a new body, **right-click "Acquire Body"** on the right side of the dialog.

[2] The one used in the Smart Mobility Lab is the Vector AC/DC NX85 Variable Output Charger.
[3] Available on the Qualysis website: `http://www.qualisys.com`. It requires log in credentials.

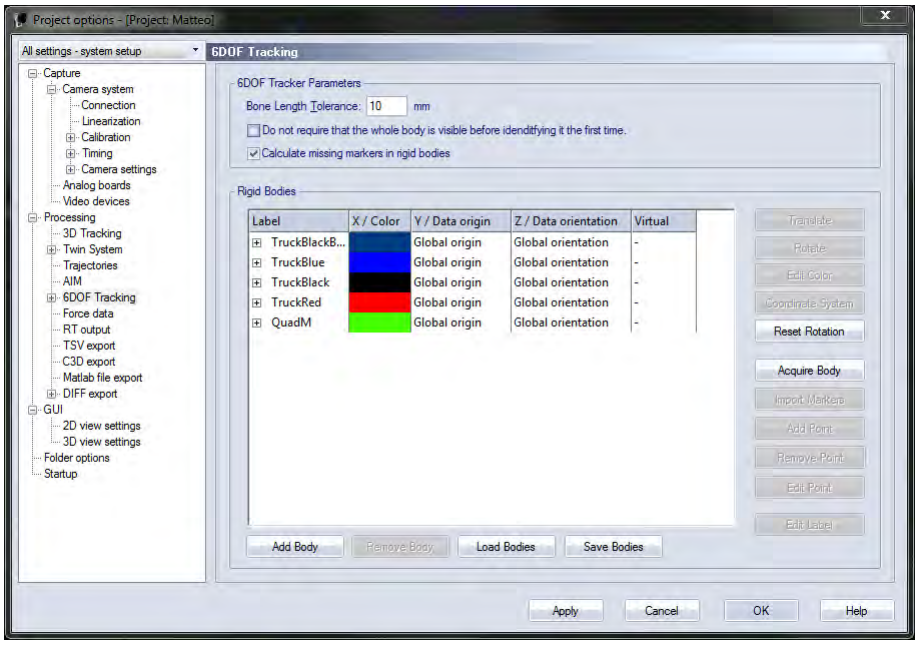


Figure A.12: ”6DOF Tracking” page in the ”Project Options” dialog.

- **Step 2**: Another dialog should appear. Then click **”Start”**.

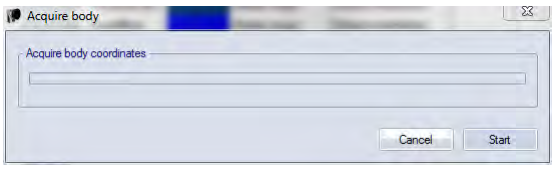


Figure A.13: ”Acquire Body” dialog.

- **Step 3**: All the markers that can be seen by the cameras will appear under the ”New Body” markers list. You should **delete from the list all the acquired markers that do not make part of the body you want to define**. The best way to do this is either by only putting the body you are defining on the visible are or knowing more or less the coordinates of the markers which make part of the new body and you deleting all the others.

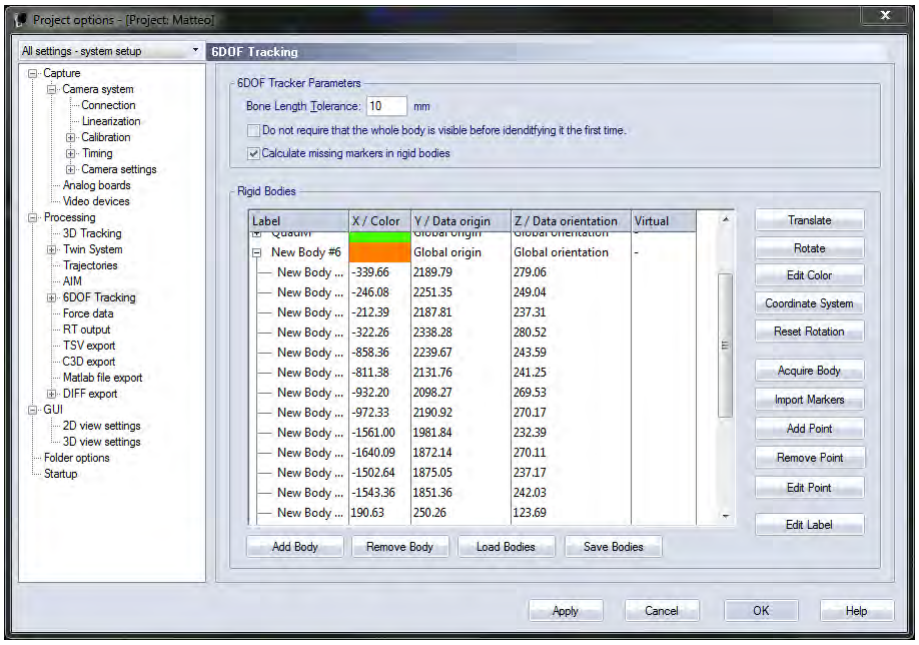


Figure A.14: The ”New Body #6” appeared on the page.

- **Step 4**: After deleting all the markers from the list, **you should have just 4 markers** in order to define a truck as a 6DOF Rigid Body. Now, you should **rename** the ”New Body” to a logical name like ”Truck Grey” and you **must renumber** all the markers acquired from 1 to N where N is the number of markers of the new Rigid Body you want to define. If you want you can also **change the color** that the Rigid Body will assume when recognized.

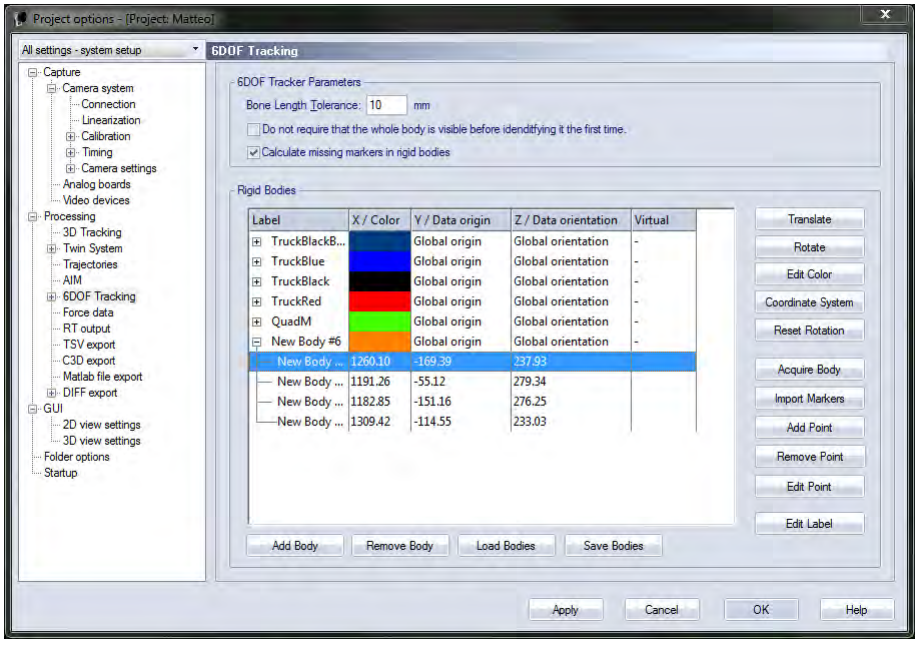


Figure A.15: The ”New Body #6” with just 4 markers associated.

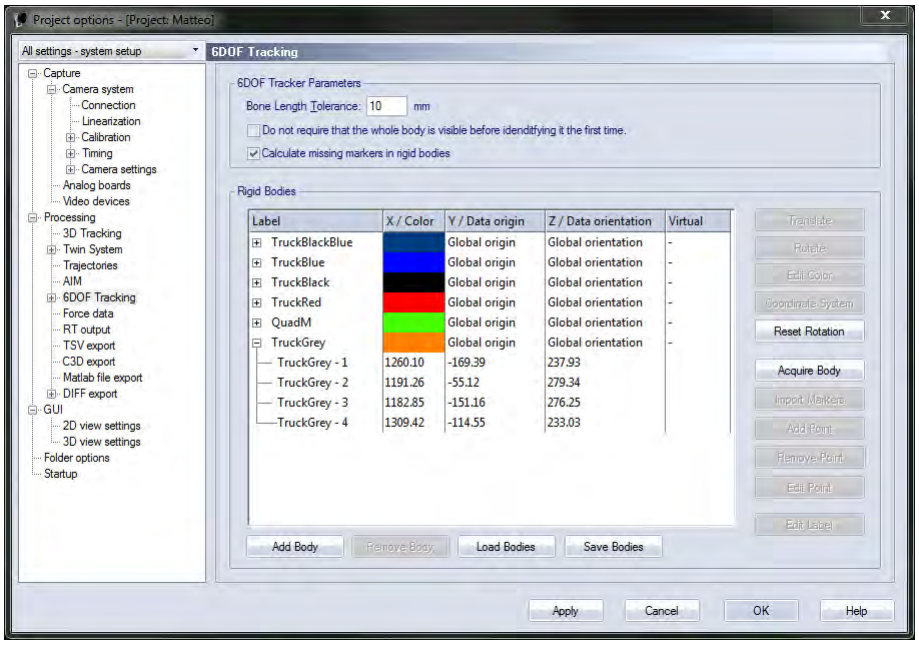


Figure A.16: The body was now renamed to ”Truck Grey”. Also the makers were renumbered.

- **Step 5**: You should now see the new body **recognized** with the color you have chosen. In Figure A.17, this corresponds to the 4 grey points next to the X axis. The next step is to **associate a local coordinate system to the Rigid Body**.

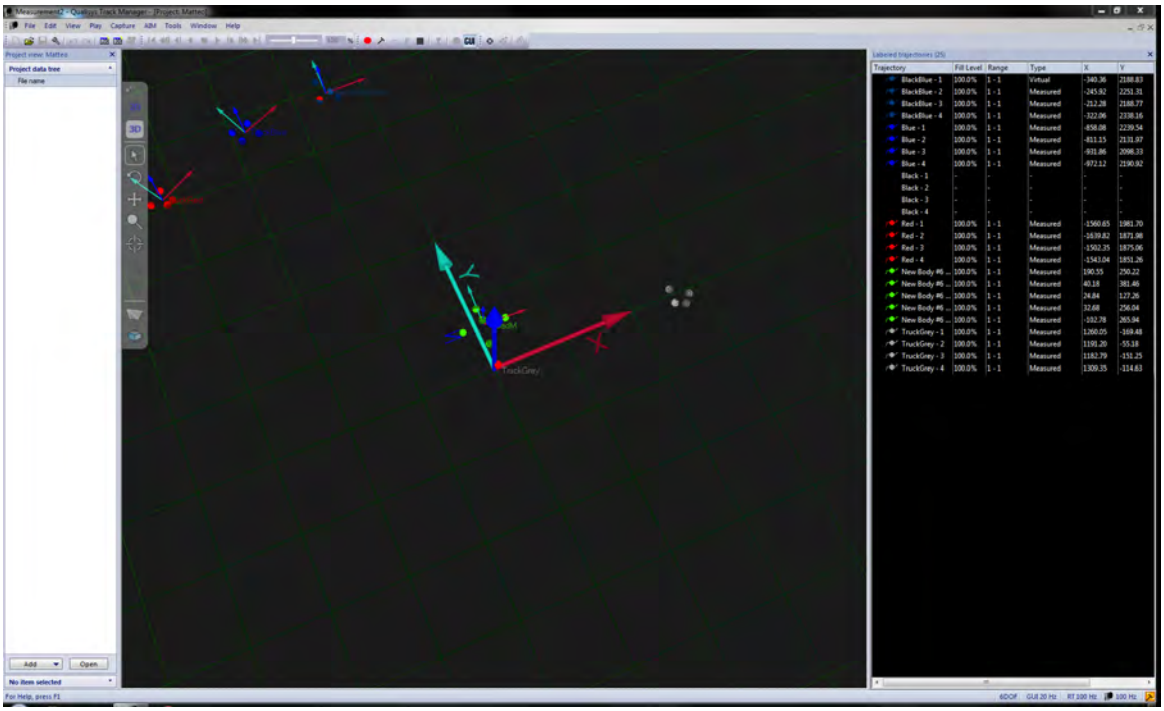


Figure A.17: 3D reconstruction to the space views by the cameras. The new body acquired appears with a new color.

- **Step 6**: To **associate the X and Y axis of local coordinate system to certain markers**, right-click **"Translate"** on the right side of the "6DOF Tracking" page in the "Project Options" dialog. A new dialog box should appear. Then select **"Align the body using its points"** and **input the numbers** of the markers you want to be recognized as the X axis and the Y axis of the body.

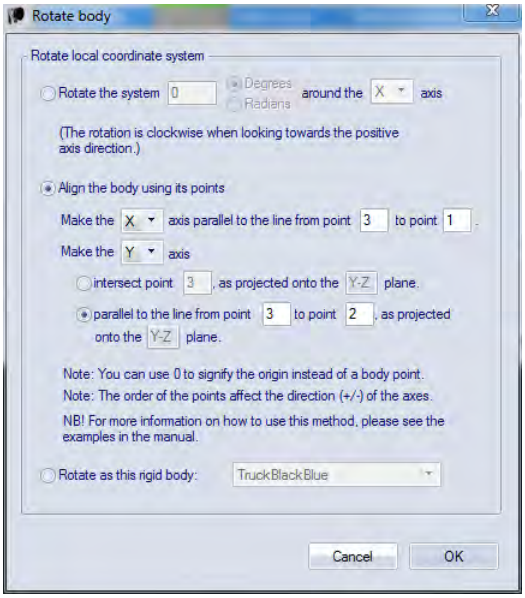


Figure A.18: "Rotate body" dialog - aligning using its own points.

- **Step 7**: Some final adjustments like **rotating the coordinate system around a certain axis** can be done. On the same **"Rotate"** dialog, **if acquiring one of our marker configurations**, you should rotate more or less -20^o degrees around the Y axis in order to align the local coordinate system with the world coordinate system.

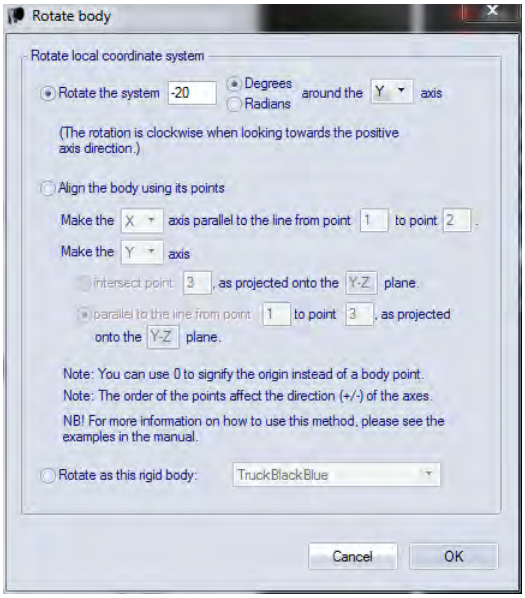


Figure A.19: "Rotate body" dialog - rotating the local coordinate system.

- **Step 8**: To **translate the coordinate system to the geometric center of the body**, right-click **"Translate"** on the right side of the dialog. A new dialog box should appear. Then select **"To the geometric center of the body (the average of the body points)"**.

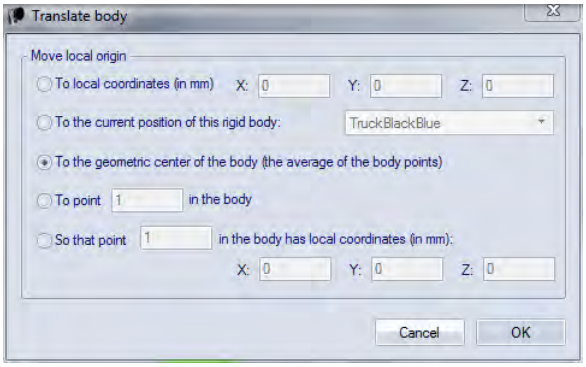


Figure A.20: "Translate body" dialog.

- **Step 9**: The new body should now appear with the **correct label and color and with the local coordinate system**.

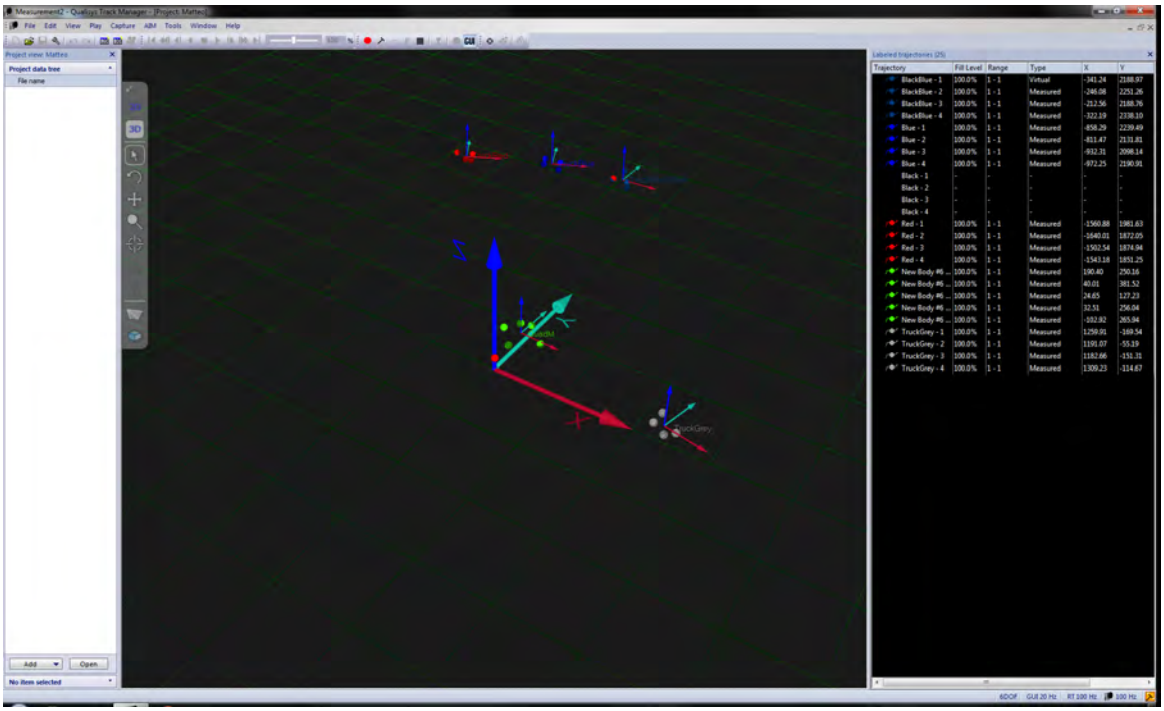


Figure A.21: The new body is now visible with correct label and color and with the local system.

If you are not getting the desired result please make sure you followed correctly all the steps and refer to [1] and [3] for more detailed information.

A.4 Trajectory Creation With Matlab

In order to move the trucks along the road network we have to create the trajectories. They are composed of a sequence of points that will be followed by the trucks. There are several possible ways to create the trajectory. Here we are going to present the method that we used and all the steps you should take if you want to replicate it in some way.

A.4.1 Prerequisites

A.4.2 Matlab Qualysis Client

To communicate with the **Qualysis** system you need the **MATLAB Client**(which you can download in the Qualysis website[4]). It will require login credentials, but if you use a computer in the **Smart Mobility Lab** you will have the client already installed. Naturally, to use the **MATLAB Client** you need to have installed in your computer a valid **MATLAB** license.

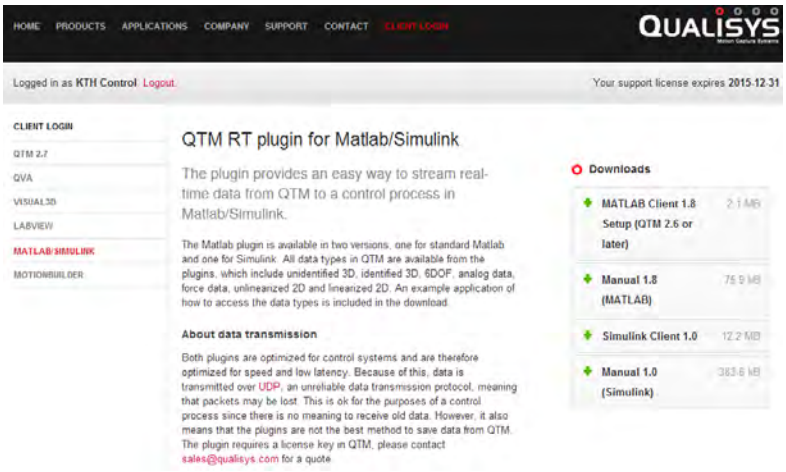


Figure A.22: Downloading the qualisys MATLAB Client.

Figure A.23: Qualisys MATLAB Client installation.

[4] www.qualysis.com

If you need to use it in a different computer just download the installation file and follow the normal procedures to have the client installed. Note that a computer with Microsoft Windows installed is required.

Configuration File

When communicating with the Qualysis system, the MATLAB Client uses a configuration file where all the settings are defined. With the purpose of creating a more dynamical experience, we created a MATLAB application that generates the configuration file according to the settings chosen by the user.

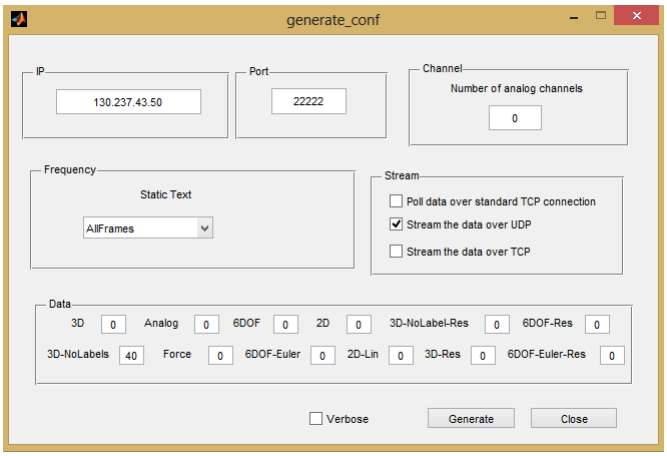


Figure A.24: Configuration file generator software.

To know more about each parameter of the configuration file we recommend you to refer to [2]. In Figure A.25, as an example, we present a sample of a generated configuration file with the settings shown in Figure A.24.

QMC_conf.txt

```
# IP address of the QTM_RT server
<IP:130.237.43.50>

# Port used
<PORT:22222>

# Frequency to fetch the data from the QTM RT server. This is only used when
# streaming data (STREAM set to 1), se below.
# <FREQ:FrequencyDivisor:n>   # The camera frequency divided by n.
# <FREQ:Frequency:n>          # Stream data in n Hz
# <FREQ:AllFrames>            # Stream data with camera frequency.
<FREQ:AllFrames>

# Stream data
# 0 : Request (poll) data over standard TCP connection.
# 1 : Stream the data over UDP.
# 2 : Stream the data over TCP.
<STREAM:0>

# Verbose
# 0 : Do not print frame number and timestamp.
# 1 : Print frame number and timestamp.
<VERBOSE:0>

# Output data size to MATLAB. Amount of objects sent to output for each
# component. Enter a value for each component. Use 0 to disable a component.
<3D:0>                    # Max number of markers to receive.
<3D-NoLabels:40>           # Max number of unlabeled markers to receive.
<AnalogSingle:0>          # Max number of analog devices to receive data from.
<ForceSingle:0>           # Max number of forces to receive.
<6DOF:0>                  # Max number of 6DOF bodies to receive.
<6DOF-Euler:0>            # Max number of 6DOF-Euler bodies to receive.
<2D:0>                    # Max number of 2D points to receive from one camera.
<2D-Lin:0>                # Max number of 2D-Lin points to receive from one camera.
<3D-Residual:0>           # Max number of markers with residual to receive.
<3D-NoLabels-Residual:0> # Max number of unlabeled markers with residual to receive.
<6DOF-Residual:0>         # Max number of 6DOF bodies to receive.
<6DOF-Euler-Residual:0>  # Max number of 6DOF-Euler bodies to receive.

# Max number of analog channels to receive data from.
<CHANNEL:0>
```

Figure A.25: Example of a configuration file.

A.4.3 Trajectory Creation

Now that you have everything needed to acquire a trajectory, start by running the file *trajectory.m* that will start the acquisition process.

In case you already have a configuration file in the same folder you are running the *trajectory.m* file, the program will use that file; otherwise, the configuration file generator will start and allow you to create a new one. This file will be used as the configuration file to the acquisition process. If the communication is started successfully, it should appear something similar to Figure A.26.

```
>> trajectory
Qualisys MATLAB client v1.8
Connecting to : 130.237.43.50 on port 22222
Active data:  3DnoLabels
```

Figure A.26: Example of a MATLAB Client communication.

After that, something similar to Figure A.27 should appear.

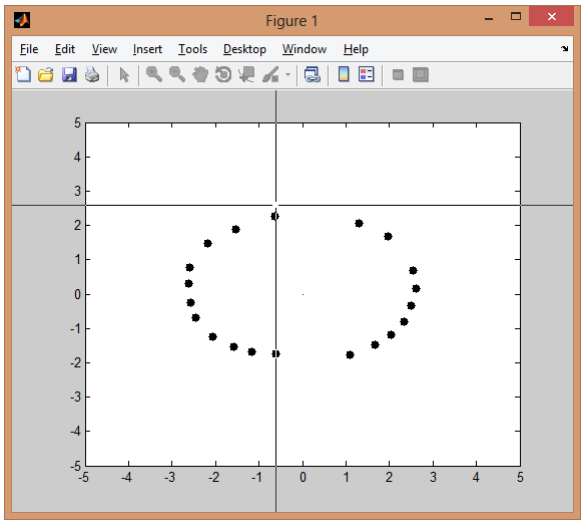


Figure A.27: Example of points acquired for one lane.

Here you are asked to choose the first two points of the trajectory. This is done with the function `ginput` of **MATLAB**, so to choose the points you just nee to click twice (once for each point) somewhere close to the desired points. We need to choose **two points** because at this point we are stipulating the direction of the lane. Since the **LabVIEW** program makes the truck follow the trajectory sequentially, it is important to choose beforehand the direction of the trajectory.

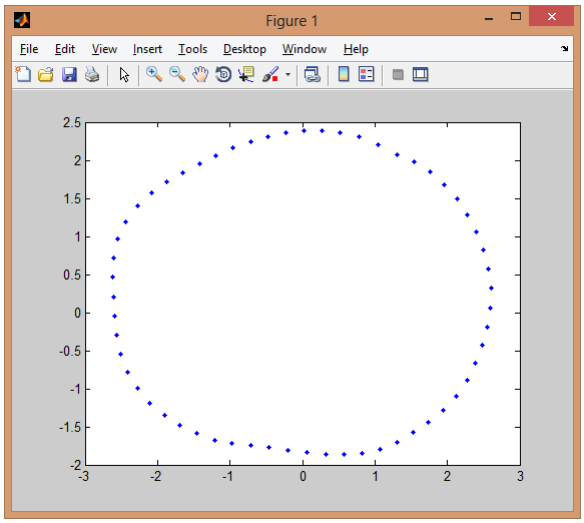


Figure A.28: Final trajectory generated.

We use an **interpolation** process to create more waypoints because this creates a **smoother trajectory** which will create a more realistic trajectory following of the trucks. After that, the result is similar to Figure A.28.

In the end, two files are created. One is *"WPList.txt"*, a file with a list of the coordinates in meters of all the trajectory waypoints. Note that it is relevant to choose if you want numbers' floating points to be '**.**' or '**,**'. Unfortunately, different versions of **LabVIEW** require different nomenclatures.

Version	Nomenclature
LabVIEW 2011	,
LabVIEW 2012	.

Table A.1: **LabVIEW** version floating point nomenclature.

It will also be created another file named *"TrajectoryLUT.txt"*. This is a lookup table with the distances between every waypoint to every other waypoints in meters. Once again, pay attention to the **nomenclature** that you choose. We opted to calculate the distances between every point of the trajectory in the trajectory creation stage, mainly because it only needs to be done once and since it is somewhat computational costly, it is beneficial to do it in **MATLAB** and not in **LabVIEW**.

Now that you have the trajectory files, the last step is to copy them to the correct folders in the computer that is running the **LabVIEW** program.

When creating a road network, you may need to create several roads or lanes. For that you need to do the process previously explained for every road you want to create. After that you get two files for every road, one containing the waypoints and the other one with the distances between the waypoints. If you need to create connected roads, i.e. roads that have transition waypoints between them, you just create the roads normally and then in the **LabVIEW** program you need to specify the number of those waypoints.

A.5 Getting Started with the LabVIEW Program

LabVIEW is a powerful **National Instruments** software that allows the user to program using a graphical interface like blocks and connecting wires. The developed LabVIEW program achieves the following objectives:

- Creates the **connection** between the **PC** and the **trucks** using the serial forwarded T-Motes;
- Creates the **TCP connection** between the PC and another PC running the **Visualization Tool**;
- Fetches the **MoCap** data;
- **Controls** the trucks behavior.

A.5.1 Prerequisites

We recommend that the program is used in a PC running **Windows 7 or higher** with **LabVIEW 2011 or higher**. Note that only the **LabVIEW 2012** is compatible with **Windows 8**. Although there are compatible versions of LabVIEW for other platforms like Linux and Mac OSX it is not guaranteed that everything works like in Windows. Problems like doing the T-Motes serial forwarding and fetching the MoCap data can arise. LabVIEW can be **downloaded** from the National Instruments website[5] and, for instance, a **student license** can be downloaded from **KTH Prodgist** website[6] and then activated. In order to do the serial forward of the T-Motes you have to install Cygwin and TinyOS (see Chapter A.2). To be able to fetch the data from the **MoCap** it is essential to download the **LabVIEW QTM Plugin** from **Qualysis** website[7]-for that it is needed a client **username** and **password**. The installation is pretty straightforward and the steps are pretty common among all the Windows installations and is explained next.

In order to find the **LabVIEW QTM Plugin** you just need to go to the **Qualysis** website and you will see its download link on the **right side** of the page.

[5] www.ni.com
[6] www.kth.se/student/support/itsc/progdist/software-for-windows/labview-2012-spring-1.337984
[7] www.qualysis.com

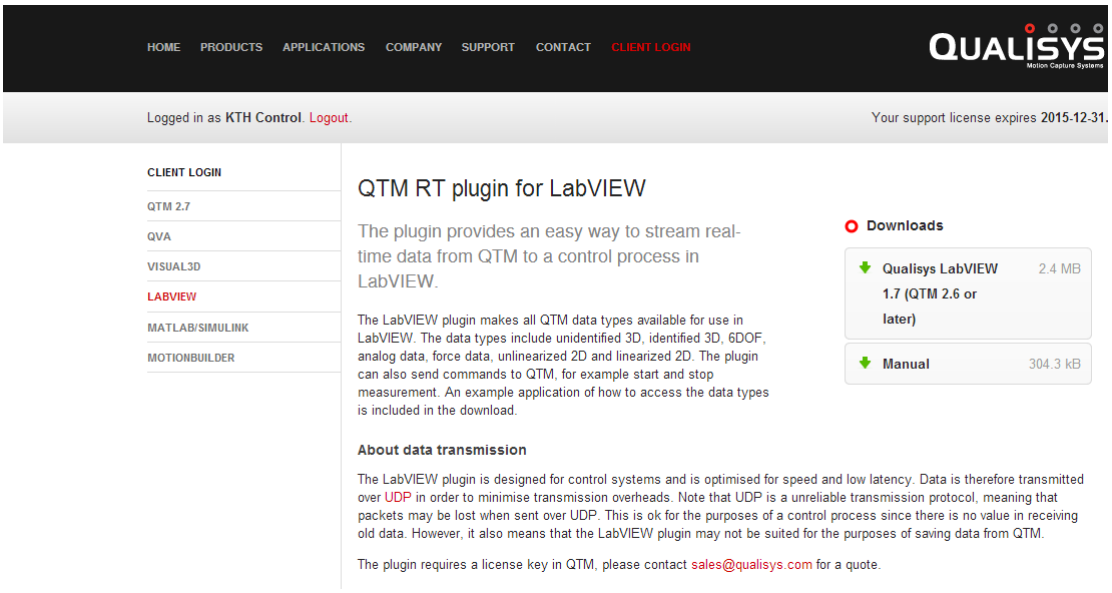


Figure A.29: LabVIEW QTM Plugin download in the Qualisys webpage.

In the folder where the LabVIEW project is in, there should be as well a folder with the name *RoadNetwork2011* and *RoadNetwork2012*, which are used inside of LabVIEW 2011 and LabVIEW 2012 respectively. Inside of those there should appear several *.txt* files that describe the **road network**. Their creation is explained in Chapter A.4.

A.5.2 Program Structure

A **block diagram** representing the LabVIEW **program structure** is shown on figure A.30.

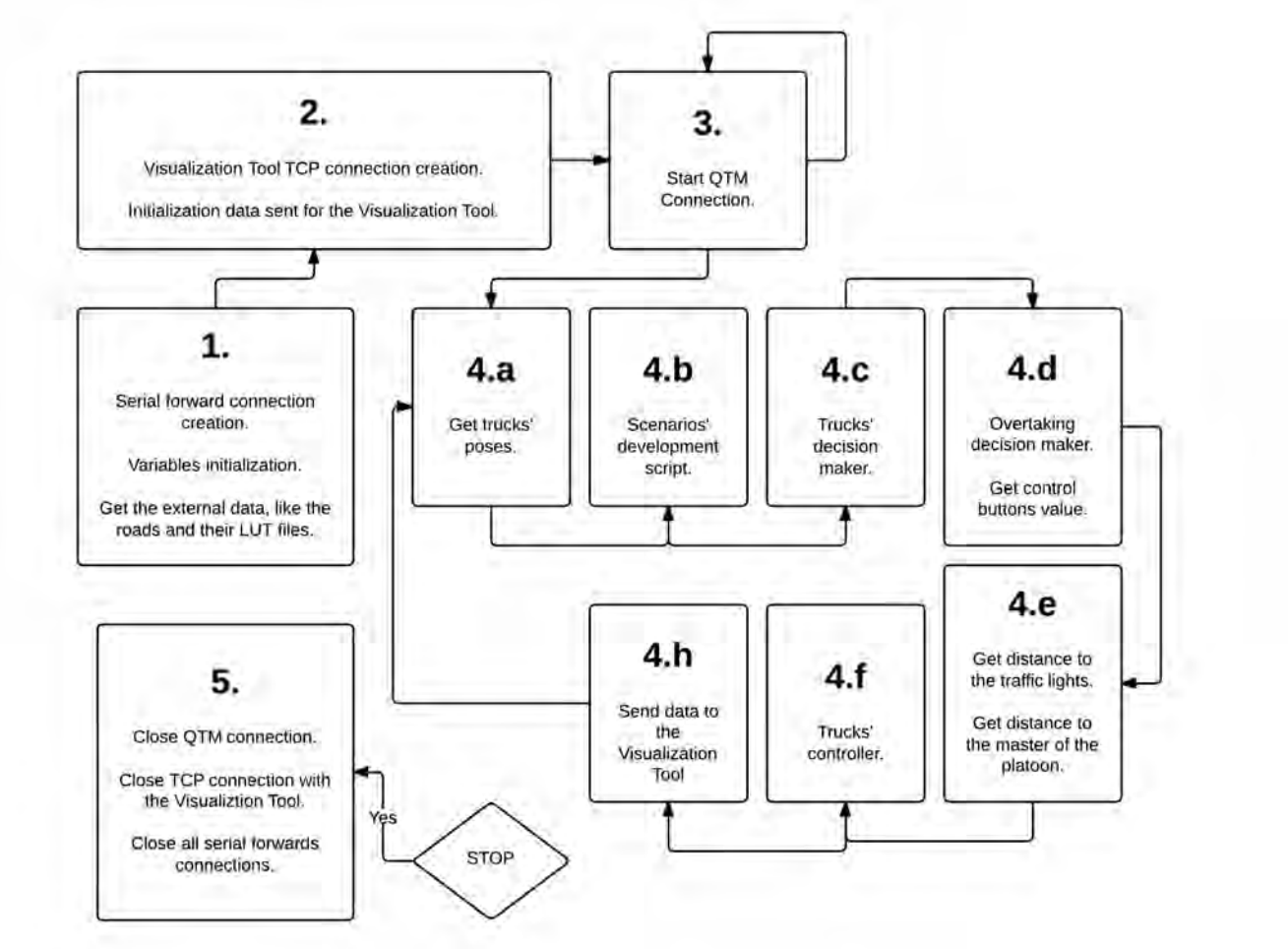


Figure A.30: LabVIEW program structure block diagram.

- **1.** This is where the program execution **starts**. At this point you should have created the **serial forward** for each mote in the right **COM ports**. You should have checked as well if the **Road Network files** exist or not. One of the main hidden aspects about this part is the setting of the **midranges values** of the trucks. The midranges values are the values that are going to be considered as 0 in the truck. This means that if we send the value 127 both for **speed** and for **steering**, the truck should be **stopped** and with its **wheels heading straightforward**. If we sent **below** that value, the truck should go **backward** (in the case of the speed value) and **turn left** (in the case of the steering value) and **vice versa for upper values**. The setting of these midranges values is the **first value sent** to the truck **after the connection** is made;

- **2.** A **TCP connection** is created and the PC where the program is running works as a **server** for the **Visualization Tool**. The **first** message sent has the following **format**:

NRoads	Road$_1$ Length	Road$_1$ WP	...	Road$_N$ Length	Road$_N$ WP	Scenario Owner

Scenario Name	NTrucks	Bridge Status

- **3.** The **QTM connection** is done in an infinite while loop in order to provide the **latest value** possible;

- **4.a** The **trucks poses** are fetched using the method presented in a **LabVIEW demo** that is installed along with the **LabVIEW QTM Plugin**. Nevertheless, we have changed the block *Q6D Euler.vi*. The following changes were made:

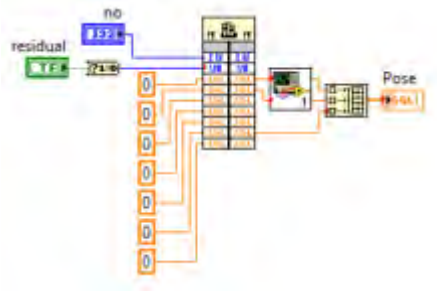


Figure A.31: Q6D Euler.vi modification step 1.

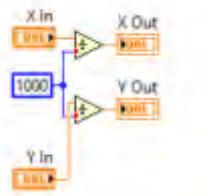


Figure A.32: Q6D Euler.vi modification step 2.

These modifications allow the Q6D Euler.vi to output an **array** with the coordinates x and y in **meters** (instead of millimeters) and the **orientation** yaw in **degrees**. The figure A.32 corresponds to the **subvi** block in figure A.31;

- **4.b** Whenever a **scenario** is selected to run, a **MathScript** will **evaluate** and **change** the control variables in order to run a **user-defined scenario**;

- **4.c** The **decision maker** is the **brain** of the trucks. It decides the **trajectory waypoint** to follow, the **reference speed** changes and the **state transitions**. In this case, each state is a **road lane**. We have an **Outer Lane** (state 1), an **Inner Lane** (state 2) and an **Inner Road** (state 3). It is only allowed to transition between the states 2 and 3;

- **4.d** The **overtaking decision maker** does nothing more often than not. When the **platoon master is changed** and that truck was **already in a platoon**, the new master will **overtake** in order to go to the front of the platoon. The overtaking decision maker decides when the truck should **start its overtaking maneuver** and when it should **finish it**.

  At this point, the **buttons' values** that the user can change in the Main.vi front panel are evaluated.

- **4.e** If there is a **platoon master**, all the trucks in the **same lane** as the master will have its **distance to the master updated**. Since the **Outer Lane** (state 1) and the **Inner Lane** (state 2) have **traffic lights,** the **distance of each truck to them** is also updated. This is only applicable to the states where traffic lights exist.

- **4.f** Using the information given by the decision maker **each truck is controlled using two PID controllers**, one for the **speed** and the other for the **steering**.

- **4.h** All the updated information is sent to the **Visualization Tool** in order to be represented in real-time. The message sent has the following **format**:

STOP	Traffic Lights State	Traffic Lights Position	MasterID	Platooning Reference Distance

Real Speed$_1$	...	Real Speed$_N$	Current State$_1$	...	Current State$_N$

Current WP$_1$	...	Current WP$_N$	Distance to Traffic Lights$_1$	...	Distance to Traffic Lights$_N$

Distance to Master$_1$	...	Distance to Master$_N$	Reference Speed$_i$	...	Reference Speed$_N$

Platoon Array	Stop Array	Truck$_1$ Pose	...	Truck$_N$ Pose	Information String

- **5.** When the button **STOP** is pressed (also called the **panic button**) everything stops. The **trucks stop** and the **connections to QTM**, to the **T-Motes** and to the **Visualization Tool** are **closed**.

A.5.3 Running the LabVIEW program

In order to run the LabVIEW program you just need to know how to use the **Main vi**. The **Main vi front panel** can be seen in Figure A.33 and is the **control panel** of the program. At first, one may think that it is too much information at the same time, but **everything is organized and has its own logic**. Each part of the front panel will be explained in detail.

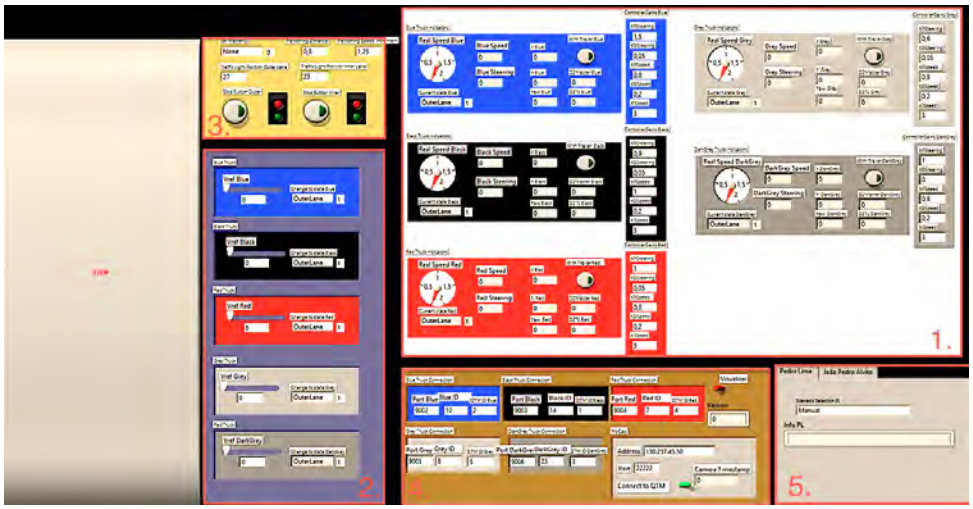


Figure A.33: Main vi front panel.

- **1.** This panel is mostly consisted of **indicators**. There we can see, for each truck, its **Real Speed**, **Current State**, **Current Pose** (x, y and yaw), **Distance to Master**, **Distance to Traffic Lights** and the values of **Speed** and **Steering** in **bits** sent to the truck. Each truck has an associated **panel color** which, naturally, corresponds to its **real color**. In this panel, one can select if the truck has a **trailer** or not, which will influence the program's perception about the **truck's length**. In order to easily tune the controllers for each truck, the **controller gains** are available as well for each truck.

- **2.** If the program is on the **Manual Mode** (see point **5.**) you will do most of the modifications on this panel. With a sliding bar you can change the **Reference Speed** of each truck and you can

Change the State where the truck is. Naturally, the truck will only change its state in the **transition points**, between the **Inner Road** and the **Inner Lane**. **If the program isn't on the Manual Mode you don't have to worry about this**.

- **3.** Once again, if the program is on the **Manual Mode** (see point **5.**) you can modify several parameters of the system. These parameters are: the **Reference Gap Distance** between trucks when they are on a platoon formation, the **Speed Increment** ($V_{ref} = SpeedIncrement \times V_{Master}$ when the truck starts catching up, the **Traffic Lights Position** in Outer and Inner Lanes and each **Traffic Light Status**. The position of the traffic lights can be hard to understand and to set. The values that are now set by default are **waypoints** of the corresponding lane. In order to change these values you have to do trial and error until you find the position wanted.

- **4.** The **missusage** of the control boxes on this panel can lead to a **strange behavior** of the program. For each truck you can set the corresponding **ID** used in the T-Motes communication, the **Port** used in the T-Motes serial forwarding (see Chapter A.2) and **QTM ID**, which is the label of that truck on the MoCap (see Chapter A.3). It's also possible to see the **Version** of the LabVIEW where the program is running on and set the **Visualization Tool** "**On**" and "**Off**". If this button is set to "**On**", you **must run the Visualization Tool** (see how to do it in Chapter A.6), otherwise **the LabVIEW won't run**. The program **will not run without having the values from the MoCap**, so if the connection is okay, you will see the **Camera Timestamp** increasing. Always check the **QTM computer IP address** if you notice that the **Camera Timestamp isn't changing**.

- **5.** In this panel you can choose two tabs. The "**Pedro Lima**" **tab** and the "**Joao Pedro Alvito**" **tab**. Those are the names of the authors of the program. The tabs exist in order to **run different scenarios** created by those using the program. In both tabs you have the option "**Manual**" where you can control the trucks using the panels explained above, otherwise you can choose a scenario from the list and **the trucks will execute a predefined set of actions**.

- The **STOP** button on the left is the biggest button in the Main vi. This works as a **panic button** that stops all trucks but also **closes all the connections correctly**. Try to use this button whenever you want to stop the program and **avoid the LabVIEW stop execution button**.

Before you press the LabVIEW run execution button, you have to **move the trucks by hand to strategic positions** . If the truck is set to start on the **Outer Lane**, you have to put the truck in the outer lane of the road network on the **opposite side of the bridge** with **anti-clockwise orientation**. If the truck is set to start on the **Inner Lane** you have to put the truck in the inner lane of the road network on the **opposite side of the bridge** with **clockwise orientation**. Finally, if the truck is set to start on the **Inner Road**, you have to put the truck in the inner road of the road network on the **opposite side of the bridge** with **clockwise orientation**. The truck is set to start at the waypoint number **1**. You can re-set this when fetching the trajectory waypoints (see Chapter A.4)

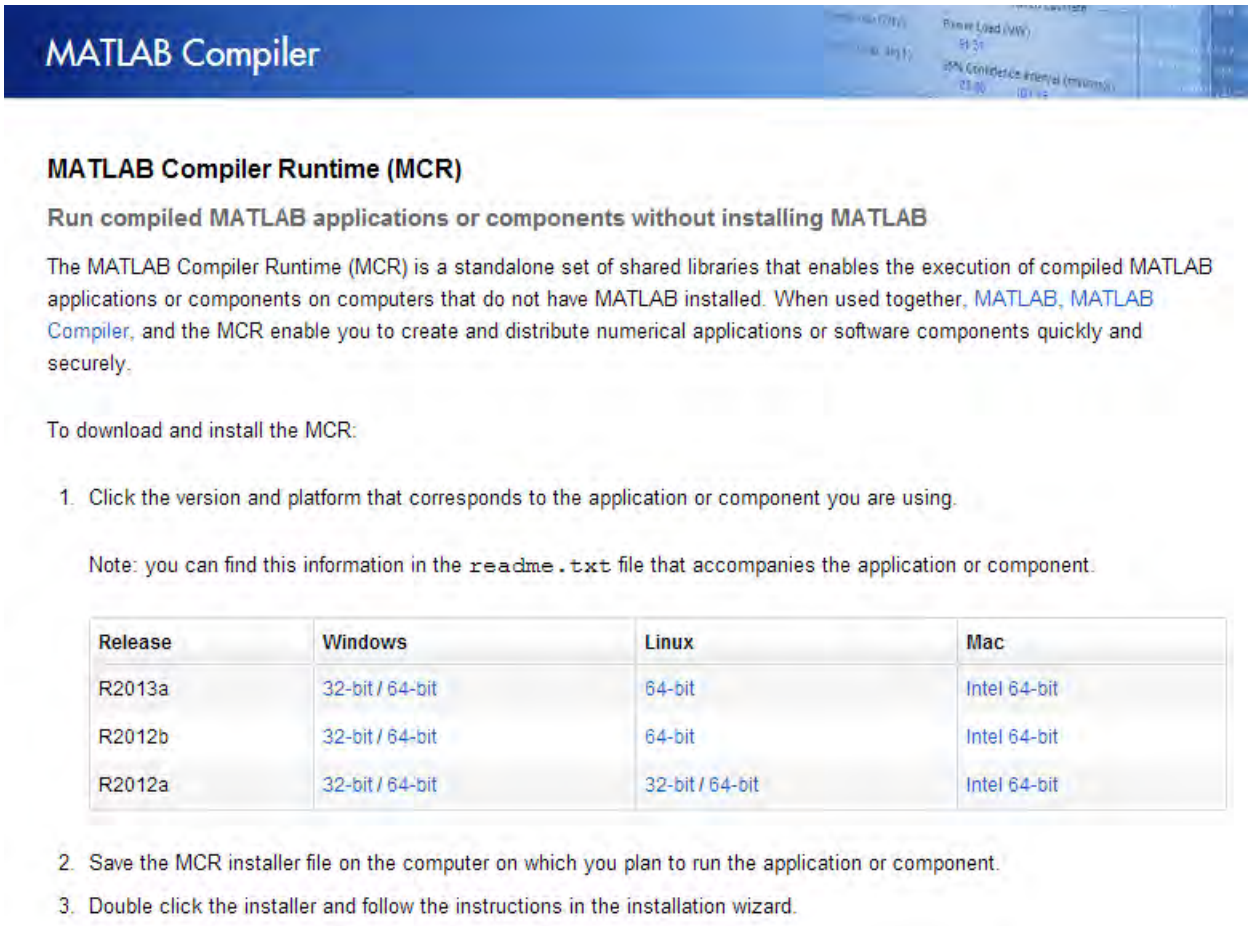
A.6 Getting Started With The Visualization Tool

To help with the visualization of the project we created a tool that shows all the important data that the user might need to know in a more user friendly way. This is done by resorting to a communication **TCP/IP** between the computer that is running the **LabVIEW** software and any computer with access to an internet connection. Note that not all the data available is displayed here, but only the sufficient to aid the user understand what is happening at the moment. Also, the data of every run is saved in a **MATLAB** data file type, which will be helpful to posterior data analysis.

A.6.1 Prerequisites

- Once the [10] is already running you only need to run the [10]. Since this tool communicates with another computer via **TCP/IP**, an internet connection is obviously needed.
- Next, you need to have a **MATLAB** version installed in the computer where the **Visualization Tool** is to be run. In case that it is not possible there is an alternative that is to only install the **MATLAB Compiler Runtime (MCR)**[8]. After downloading the version that suits your needs, you simply need to run the **executable file** that we generated in case you use **Microsoft Windows**, or, if you use **Mac OSX**, you can run the **app** that we also generated.

[8] `http://www.mathworks.se/products/compiler/mcr/`



MATLAB Compiler

MATLAB Compiler Runtime (MCR)

Run compiled MATLAB applications or components without installing MATLAB

The MATLAB Compiler Runtime (MCR) is a standalone set of shared libraries that enables the execution of compiled MATLAB applications or components on computers that do not have MATLAB installed. When used together, MATLAB, MATLAB Compiler, and the MCR enable you to create and distribute numerical applications or software components quickly and securely.

To download and install the MCR:

1. Click the version and platform that corresponds to the application or component you are using.

   Note: you can find this information in the `readme.txt` file that accompanies the application or component.

Release	Windows	Linux	Mac
R2013a	32-bit / 64-bit	64-bit	Intel 64-bit
R2012b	32-bit / 64-bit	64-bit	Intel 64-bit
R2012a	32-bit / 64-bit	32-bit / 64-bit	Intel 64-bit

2. Save the MCR installer file on the computer on which you plan to run the application or component.
3. Double click the installer and follow the instructions in the installation wizard.

Figure A.34: MATLAB Compiler Runtime (MCR) download.

- The computer that runs the **Visualization Tool** doesn't need any special graphical capabilities. Nonetheless, we recommend that you use a computer with a **considerable processing capability**. This is a requirement because the tool **receives new data** every $100ms$. If the computer isn't capable of processing the data and display it in that time window, at some point it will start processing data that is not updated. This can be seen by a **delay** between what you see in our tool and what is happening in **real-time**.

A.6.2 Running The Visualization Tool

To run the **Visualization Tool**, as mentioned in Section A.6.1, you can run the **executable file** (or the app file in case of **Mac OSX**) or, if you prefer to run it within **MATLAB**, simply run the *SML.m* file. In both cases, the first window to appear is the one in Figure A.35.

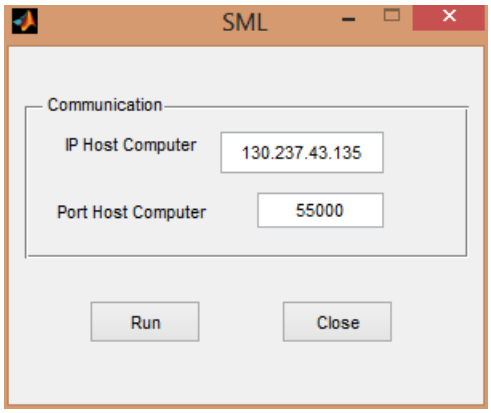


Figure A.35: Connection settings.

Here you have to define the settings for the **TCP/IP** connection between your computer and the computer running the **LabVIEW** program. You have to set both the **IP address** and the **port number** of the remote computer. If you don't know the IP address of the remote computer just type **'ipconfig'** in the command line and the IP address of that computer will be shown. The port number should preferably be a number between $49152 - 65535$, but be aware that it must be the same in both sides. So because the default value in the **LabVIEW** program is 55000, we recommend to use it as port number. If by any chance you want to change it don't forget to change in both places. The values you see in Figure A.35 are the **default values**, only because it is the current configuration in the **Smart Mobility Labs'** computers.

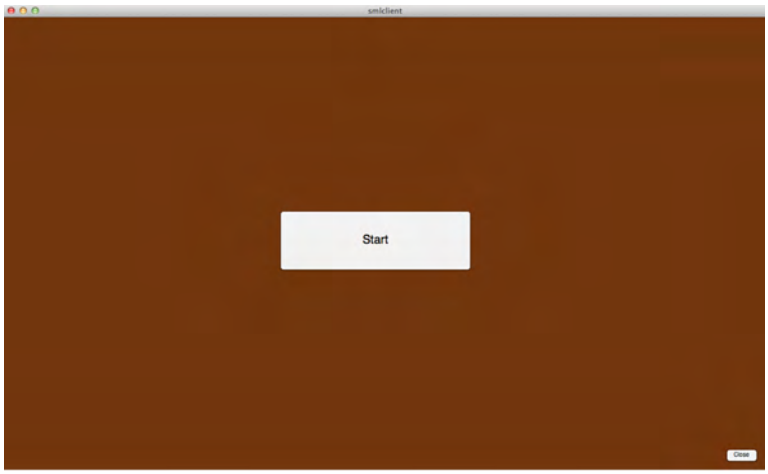


Figure A.36: Start screen of the visualization tool.

After you push the run button your connection settings are defined and a start screen should appear. Preferably, you should push the start button when you have done all the required steps in the **LabVIEW** program-this means that at this point the **LabVIEW** program should be waiting for an order to proceed. If you press it before time, it can cause some connection issues because there is a timeout in the connection. Summing-up: make sure to run the **LabVIEW** program first with the **Visualization Tool** option turned on.

A.6.3 Visualization Tool

Next, we have two examples of our **Visualization Tool**, each one representing the **layout** that we decided that better suited our scenarios. Of course you are free to improve it and adjust it to your needs. Both versions use the same base but one is more concerned in fuel related data and the other more concerned with the traffic lights status. We will briefly explain what each one of the **panels** represents.

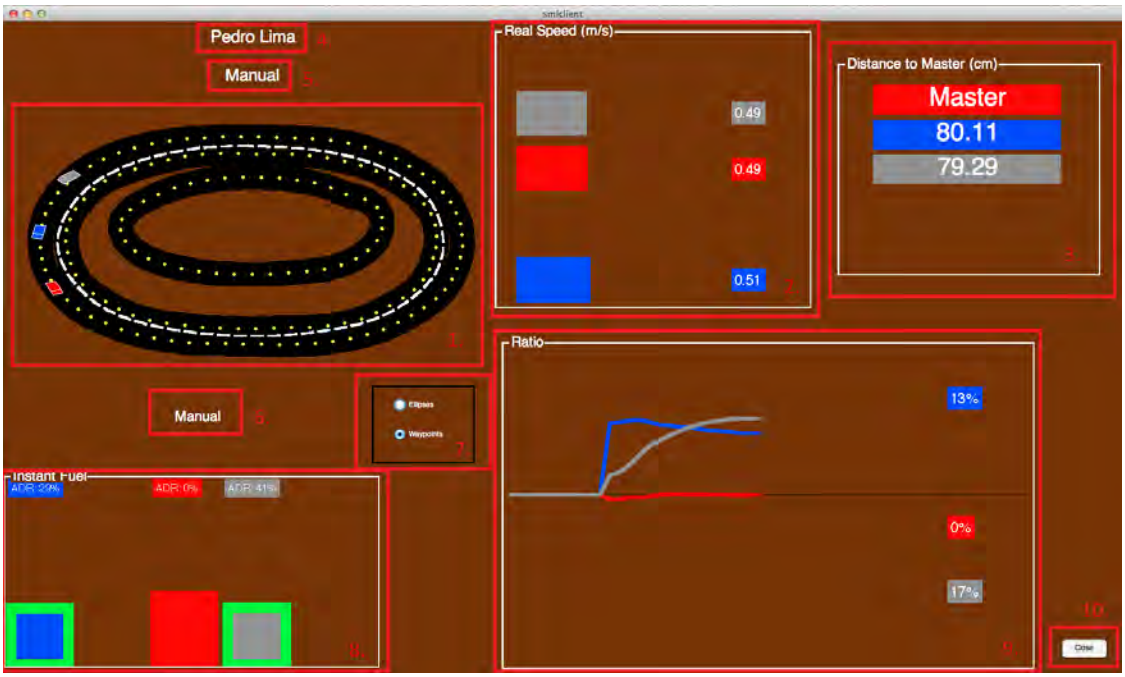


Figure A.37: Visualization Tool (Pedro Lima).

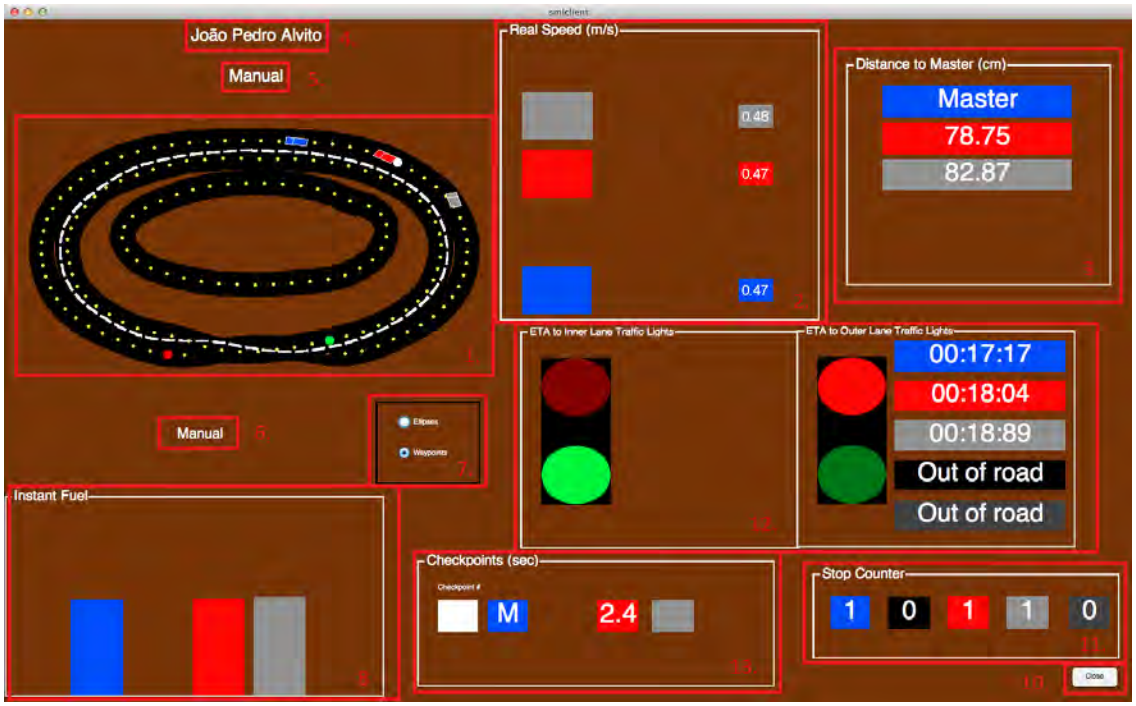


Figure A.38: Visualization Tool (João Pedro Alvito).

- **1.** This is, most likely, our most important block, mainly because it's the one that contains more information. This is basically a transcription of the reality to our display. In it, you can see the current position of each truck that is visible in the **Qualisys** system, through the small rectangular shaped trucks colored accordingly to the real trucks' colors. It's also possible to see how the **road network** looks. The **Visualization Tool** simply receives the waypoints and processes it in order to display the several roads and its lanes in black, and also the **white dashed line** that we usually see in real life. In the lanes, it is possible to display the waypoints, representing them as **yellow dots**. Additionally, in the case of Figure A.38, there are extra colored dots: the **white dot** represents a checkpoint that is used to measure the time distance between each consecutive truck (it is possible to have several checkpoints); there are **two dots** that can be either **red or green**, that at the same time give the **position** of the traffic lights of both lanes and its **status** (either if the traffic light is green or red);

- **2.** This block shows the real speeds of the trucks, *i.e.* in SI units *(m/s)*. To make it easier to distinguish between trucks, we used a color code that translates the user to the real color of the trucks that are being used. There are two fields where it is possible to get information about the speeds: a horizontal bar that is helpful to analyze the **relative speed** between trucks; a text field with the the **absolute speed** value; Note that this block only displays the speed information of the trucks that are being used, so if for example a truck is not being used or even if it is inside the visible zone but is out of the road, its speed information will be suppressed from the display;

- **3.** This block shows the most important information about the platooning that is the **Platooning Distance**. This measurement represents the size of the gap between each consecutive truck that is part of the platoon. For better understanding, we are using the measurement in *cm* and it is the real distance between the trucks and not the **real world** distances. Note that the first truck of the platoon, **the master**, naturally doesn't have any distance to display so it just identifies itself as the **master**. To improve the visualization, the list of the **Platooning Distance's** order reorders itself according for the order in the platoon; this way the information about the platoon order is embedded with the **Platooning Distance**. If a truck isn't part of the platoon the **Platooning Distance** is obviously suppressed from the display;

- **4.** This field simply shows what is the **Platooning Distance** of the current scenario;

- **5.** Here you can see what is the **name** of the current scenario. The names are pretty self-explanatory because this way it is easily understood what the scenario is meant for;

- **6.** This shows a simple explanation of what is happening at each moment. It is an one line sentence that illustrates what might not be obvious; for example, it says what a truck is trying to accomplish or if some event related to a traffic light is taking place;

- **7.** This block contains the optional choices that the user can do. The **"waypoints"** option is simply a radio button that turns "on" or "off" the waypoints display in **1.**. This option is defaulted to be "on" because we believe that the waypoints appearing are a great aid to observe the trucks' trajectory.

The other option that we have available is **"ellipses"**. We provide this option because the graphical roads in **1.** are dynamically generated, so sometimes, depending on the trajectory generated, the roads can be displayed in a rather strange way. Since the roads that we created in the **Smart Mobility Lab** are shaped like ellipses we tried to process the road data that we received from the trajectory generation and adjust it to ellipses. Sometimes, this option improves the display of the roads but of course this is not a viable option if the road is not shaped like an ellipse so feel free to try this option and see if it improves the display of the **road network** that you are currently using;

- **8.** Here you can see a relative indication of the **instantaneous fuel consumption**. It allows you to better perceive how much fuel the trucks are consuming in relation to each other. In Figure A.37 there are two extra pieces of information: the percentage of **Air Drag Reduction (ARD)** and, also surrounding the vertical bars of the instantaneous fuel, you can see either **red or green edges**. If that edge is green, it means that for that particular truck it is being beneficial in terms of fuel consumption to be part of the platoon, otherwise the edge will appear red. For more technical information refer to [5] and [10];

- **9.** Here you can see a **fuel ratio** for each truck involved in the scenario. This is the ratio between the accumulated fuel consumption, when the truck decides to do catch up, and the accumulated fuel consumption that the truck would have if it had continued alone. For more technical information refer to [10];

- **10.** This close button, when pushed, closes the program. We do not recommend to use this button as a first choice, it is preferable to use the stop button in the **LabVIEW** program which you can see in Figure A.33. This way the **LabVIEW** program sends a closing order to the Visualization Tool which allows it to close by itself and then closing all the **TCP/IP** connections from both sides;

- **11.** This blocks provides a stop counter that shows the number of times each truck has stopped. This is helpful to see if a truck stopped in a traffic light and how many times it did;

- **12.** This block has two different sub-blocks that are basically the same but the information is divided in inner and outer lane. So on the left side we have the traffic light information about the inner lane and on the right side there appears the same information for the outer lane. These sub-blocks have all the information about the **traffic lights** apart from its position: the status of the **traffic light**, either if it is red or green; the **estimated time of arrival (ETA)** of each truck to the traffic light of the correspondent lane. The time appears in a digital form (minutes:seconds:centiseconds). To improve the display we decided to order the **ETAs**, such that the closest truck to the **traffic light** will always be on the top of the list;

- **13.** Here you can check which are the time distances at a given **checkpoint** (time distance is given in seconds). Note that this feature is only valuable if there is a platoon. If a truck is the master, the letter **'M'** will appear in the same place where it would show the time. Once the master passes through the checkpoint, as showed in point **1.**, all the other fields of the trucks that are in the platoon are erased and updated with the time distance as soon as the respective truck goes trough

the same **checkpoint**. You can use several **checkpoints**, but for bigger **Platooning Distances** we recommend to use only one checkpoint. Don't forget that the only values that are shown are the ones from trucks in the platoon.

A.7 Troubleshooting

A.7.1 Trucks

If you have read all the manual and you the trucks are behaving as they supposed to, please go through the following checklist:

- Check if the batteries are **charged**;
- Check if all the **Polulu boards** have a **yellow** led **on** (see Chapter A.2);
- Check for any **"Note: write failed"** or **"unix_error"** in the **Cygwin terminal** (see Chapter A.2);
- Check all the **ID**, **Port** and **QTM ID** in the panel **4.** (see Section A.5.3);
- Check the **IP address of the QTM computer**;
- Check if the value sent for **Speed** and **Steering** in bits present in panel **1.** are different from **127** (see Section A.5.3);
- Check if the **PC's T-Motes are blinking**, meaning that they are communicating (see Chapter A.2);
- Check if the **trucks' T-Motes are blincking** as well (see Chapter A.2);
- Check if the truck motors are "**On**" (see Chapter A.2);
- Check if the trucks are **being recognized by the MoCap** (see Chapter A.3);
- Check if the trucks are **initially positioned** as they should (see Section A.5.3);
- Check if there's **any commented block**.

A.7.2 Trajectory Creation

One problem that might arise in the trajectory creation, is a message saying **"No Welcome Message Received"**. This happens when the program is trying to connect to the **Qualysis** system and it receives an error probably due to a last connection that wasn't closed. The most effective solution that we found was restating **MATLAB**.

A.7.3 Visualization Tool

When using the **Visualization Tool** sometimes you might get some error when receiving the data from the road network in MATLAB. We are not completely sure about the source of the problem; nevertheless, this issue can be solved with a simple restart of the **Visualization Tool**.

You still can't get everything to work? Please, don't hesitate to **contact us** (`pfrdal@kth.se` and `jpfa@kth.se`).