A simulation model for I-GAME

Technische Universiteit
Eindhoven
University of Technology

A simulation model for I-GAME

M.W.H. Bergmans
DC 2014.055

Supervisor:
Prof. Dr. H. Nijmeijer
Dr. J. Ploeg TNO
Dr. E. Semsar TNO
By:
M.W.H. Bergmans **0744005**

Open space project 9 ECTS

October 30, 2014

List of symbols

Symbol	Quantity	Unit	Abbreviation unit
h	User-time headway	Second	s
L	Vehicle length	Meter	m
n_v	Number of vehicles	[-]	[−]
r	Standstill distance	Meter	m
t	Time	Second	s
v	Velocity	Meters per second	m/s
x	Longitudinal position	Meter	m
y	Lateral position	Meter	m
ψ	Vehicle heading	Radians	rad

Nomenclature

Symbol	Description
$GCDC$	Grand cooperative driving challenge
CC	Cruise control
ACC	Adaptive cruise control
$CACC$	Cooperative adaptive cruise control
$STOM$	Safe to maneuver

Contents

1. Introduction

Road traffic faces more and more challenges in terms of congestion, emission and road safety. These challenges are mobility challenges. In order to change the standards regarding the congestion, emission and road safety, European and national legislations are introduced. The call objective that the I-GAME project aims to answer is "to make use of co-operative mobility technologies to develop supervised automated driving". The goal of I-GAME is to fulfill these mobility challenges and live up with the legislations by means of automated driving. The focus of I-GAME is on using wireless communication for realizing a safe and reliable automated driving. The driver is still present in order to supervise the automated driving. The I-GAME is a succession of the Grand Cooperative Driving Challenge (GCDC) back in 2011 [2]. One of the features of the GCDC was to let the vehicles exchange longitudinal information with each other in order to improve the vehicle flow on a road. At the I-GAME, the vehicles will also exchange lateral information. Therefore three tasks are introduced that the vehicles have to fulfill. The first task is merging of the vehicles, where the vehicles drive initially on two lanes and have to merge to one lane. The second task is a t-junction, at this junction the vehicles on a road have to get priority to turn into a main road. At the last task, an emergency vehicle enters the road and all the other vehicles have to move aside for the emergency vehicle. The main focus of this report will be on facilitating the simulation of the first task [1].

TNO Automotive has already developed a simulation model for a platoon of vehicles with splitting and merging capabilities. This simulation model is an extension of the simulation model used for the GCDC, which only included longitudinal platooning. The simulation model consists of multiple identical cars which are connected to each other by means of communication signals. Lateral platooning and obstacle avoidance is added to the simulation model in order to simulate merge and split maneuvers of the vehicles. In order to perform the maneuvers, the velocity and position of the vehicles are tuned with respect to the platoon. The simulation model is however not very user-friendly in simulating lateral maneuvers.

This report focuses therefore on improvements performed to make the simulation model more user-friendly. In order to make the simulation more user-friendly, some sort of menu have to be developed in order to initiate the maneuvers very easily. To initiate the maneuvers, some initial conditions need to be changed as well, like the initial position and velocity. Therefore it should also be possible to change those initial conditions as well using the menu. The report is organized as follows. In Chapter 2, the simulation model functionality and the simulation model are explained. In Chapter 3, features are added to the simulation model in order to make it more user-friendly. In Chapter 4, the results are explained and using these results, a conclusion is drawn in Chapter 5.

2. Simulation model

In this chapter, the original simulation model is explained. At first, the functionality of the simulation model is explained. After that the main parts of the simulation model are explained and at last the parameter file which is used at the initialization of the simulation model is explained.

2.1 Simulation model functionality

The maneuvers defined at the I-GAME project are, as stated in the introduction, merging two lanes of vehicles, prioritizing a traffic junction and clearing the road for an emergency vehicle. The focus of this report is on merging two lanes of vehicles. First, this is done by using a simulation model and later it will be tested with real vehicles. The basic task is explained in Figure 2.1. The simulation model has two functionalities. The first functionality is to simulate a merge maneuver. On the left lane there is one vehicle, car 2, and on the right lane there are two vehicles, car 1 and 3. The vehicle on the left lane wants to merge in between of the vehicles on the right lane. First the velocity of the vehicle on the left lane is tuned with respect to the platoon velocity. When this velocity is equal to the platoon velocity, the car on the left lane sends out a request to the other cars showing that it wants to merge. When the cars on the right lane receives this message, they create a gap between them. The merging vehicle (car 2 in Figure 2.1) will decelerate in order to make a gap with car 1. Also car 3 will decelerate in order to make a gap with car 2. The distance between the cars is measured and when the gaps between the vehicles are large enough, the vehicle on the left lane receives a Safe To Maneuver (STOM) signal and is then able to initiate the actual merging maneuver.

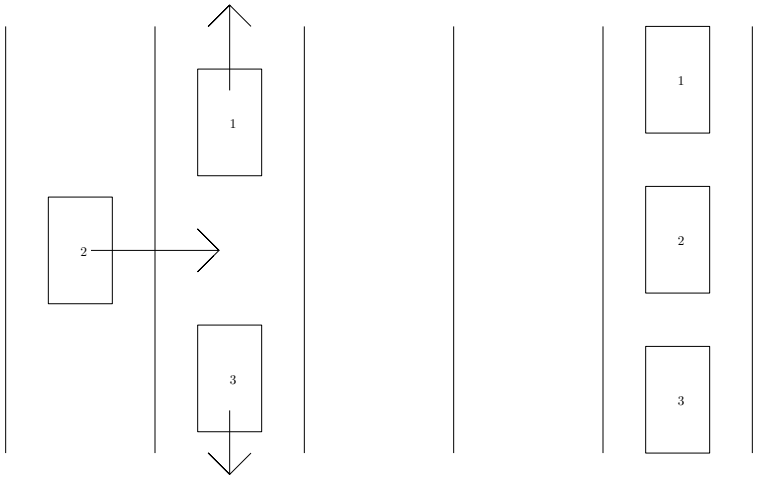

Figure 2.1: Merging of two lanes.

The second functionality of the simulation model is to simulate a split maneuver to the left lane. At the start of this maneuver, the vehicle that is about to execute the maneuver is driving on the right lane. When the vehicle is about to execute the split maneuver, it measures the distance with any cars on the left lane. When this distance is not large enough, the car decelerates in order to make a gap with the car on the left lane. When this gap is large enough, the car receives a message that it is safe to do the maneuver and it will be able to initiate the actual split maneuver.

2.2 Simulation model overview and explanation

The original simulation model made by TNO Automotive consists of 10 cars. The cars are all the same and linked to a library. The first car is the lead vehicle which means that all the other cars base their maneuver on the movement of the lead vehicle. The cars are connected to each other by means of wireless communication, as shown in Figure 2.2. Each car has its own ID and its parameters are taken from a m-file, which is discussed later in this chapter. The system under the mask of each car is shown in Figure 2.3. From this figure it can be seen that each car contains four major components. One module represents the communication signals, there is a subsystem that contains a vehicle model, a user interface and a masked subsystem that contains the control system. On the bottom left of Figure 2.2 it can also be seen that there is a start/pause button.

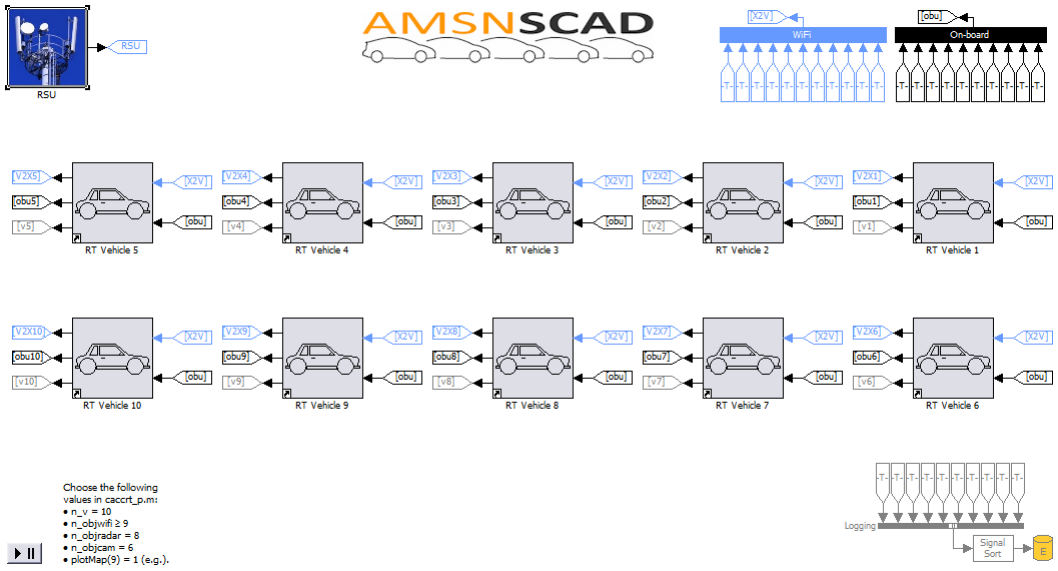

Figure 2.2: Simulation model with 10 cars.

Communication signals and sensors model There are four means of obtaining information in the cars: radar, camera, GPS and sensors. The radar, camera and sensors are on-board signals while the GPS is a wireless signal. The radar receives data of 8 objects including the x-range and the y-range of each object. The camera does basically the same as the radar only it receives data from 6 objects. Also the camera receives data about the lane information, for example the lane type and the lane offset. The camera detects curvature, like a turn, of the lane and plots a polynomial through that curvature in order for the car to follow the lane. The GPS signal contains data from the ego vehicle, like the vehicle position and velocity. The wireless signals of all vehicles are collected and bundled to one signal that is sent back to all vehicles. This signal contains both infrastructure to vehicle information as well as vehicle to vehicle information. All the other cars know therefore the position and velocity of the other vehicles and can use this signal to determine

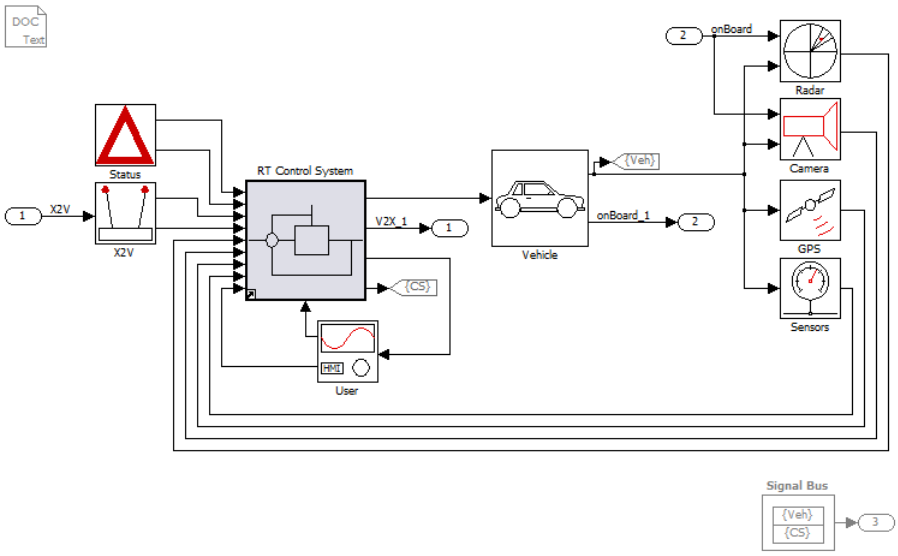

Figure 2.3: Vehicle system under mask.

their maneuver. At last there are the sensors. Each car contains a number of sensors which measure the cars longitudinal velocity and acceleration, lateral acceleration, yaw rate, steering angle, drive force, wheel speed and an estimation of the vehicle mass.

During real time driving, lane information is taken from the road by means of the camera. During the simulation, there is however not a physical road which means that fake lane information should be used. This should be done in the camera block because the camera is the sensor which 'sees' the road information. Faking the road information has been done by changing the lane confidence grade and the lane validity parameters. Both parameters can be varied between 0 and 3. For the lane validity, 0 means that the lane is undefined, 1 means that the lane is new, 2 means that the lane is valid and 3 means that the lane is invalid. The lane confidence grade should become equal to 3 which is the highest confidence and the lane validity should become equal to 2 which means that the lane is valid.

Vehicle model The vehicle model contains a simple bicycle model for modeling the tyre-road contact. The steering wheel input is translated to the angle of the vehicle and the vehicle yaw rate. A coordinate transformation is used to determine the x- and y-coordinates of the vehicle. The velocity in x- and y-direction can be determined using the vehicle velocity and the vehicle heading ψ in $[rad]$, see (2.1) and (2.2). Integrating the velocity in x- and y-direction gives the x- and y-coordinates of the vehicle.

$$v_x = v \cos \psi \tag{2.1}$$

$$v_y = v \sin \psi \tag{2.2}$$

Control system The control block contains all controllers that are needed for cooperative driving. In total, there are 5 different modules, as can be seen from Figure 2.4. Figure 2.4 shows the system under the mask of the controller block. The host tracking determines the state of the ego vehicle. It figures out the vehicle's position and other information that the vehicle needs to know from itself. The target tracking tracks the relevant target vehicles. It searches for objects around the vehicle and sends out relevant information. The maximum number of target vehicles is predefined but the actual number can be dependent on the present vehicles in the neighborhood of the ego vehicle.

The lateral controller controls the lateral position of the vehicle by controlling the steering angle or the yaw rate. The longitudinal controller consists of four controllers. The user can choose between a CC, ACC and CACC controller and the longitudinal controller contains also an obstacle avoidance controller. The CC controller is a simple Cruise controller which keeps the velocity constant. The ACC controller stands for Adaptive Cruise Control where the velocity of a car is based on the velocity of the car in front of it. The CACC controller is a Cooperative Adaptive Cruise Control where cars communicate with each other and pass information to each other. In normal operation mode, the CACC controller is chosen. When there is no car in front of the ego car, the CC controller is chosen. The obstacle avoidance is responsible for making a gap when a maneuver is initiated.

The supervisory control supervises the lateral and longitudinal controllers, it selects the right settings for each controller. Besides that it also contains a split/merge activation and STOM generation. The split/merge activation is active in the maneuvering car after the host selects the split or merge maneuver. It decides if the split or merge maneuver can actually be performed. The STOM generation is active in the vehicles that are not going to perform the maneuver. STOM stands for Safe To Maneuver and decides whether it is safe to perform the maneuver based on the distance between the vehicles.

User interface For this project, the user interface is the most important part of the model. In the user interface, all the signals from the controller are displayed and all the vehicles actions, controller types and some initial conditions are defined in this block. The output of the user interface is connected with the input of the controller. The actions are initiated by changing the value of the input. This is done in the parameter file, where a value of 0 means that the action is off, and a value of 1 means that the action is on. So changing the value from 0 to 1 initiates the action. Also the velocity profile, cruiser speed and vehicle type are defined in this block. Two maneuvers can be performed using the user interface, which are the merging and splitting action.

For the merging maneuver, three actions have to be initiated by means of a pulse from 0 to 1. First, the car tunes its velocity with respect to the platoon velocity. The signal that initiates the second action is a merge request. The merging car decelerates in order to enlarge the gap with the car in front of it. When the rear car receives this pulse it starts to decelerate in order to enlarge the gap with the merging car. When the gap between the front and rear car is large enough, the supervisory control sends a Safe To Maneuver (STOM) message to the user interface. When this STOM message is equal to 1; it is safe for the merging vehicle to merge. From this moment, the actual merge maneuver can be initiated and the car will merge between the front and rear car. For

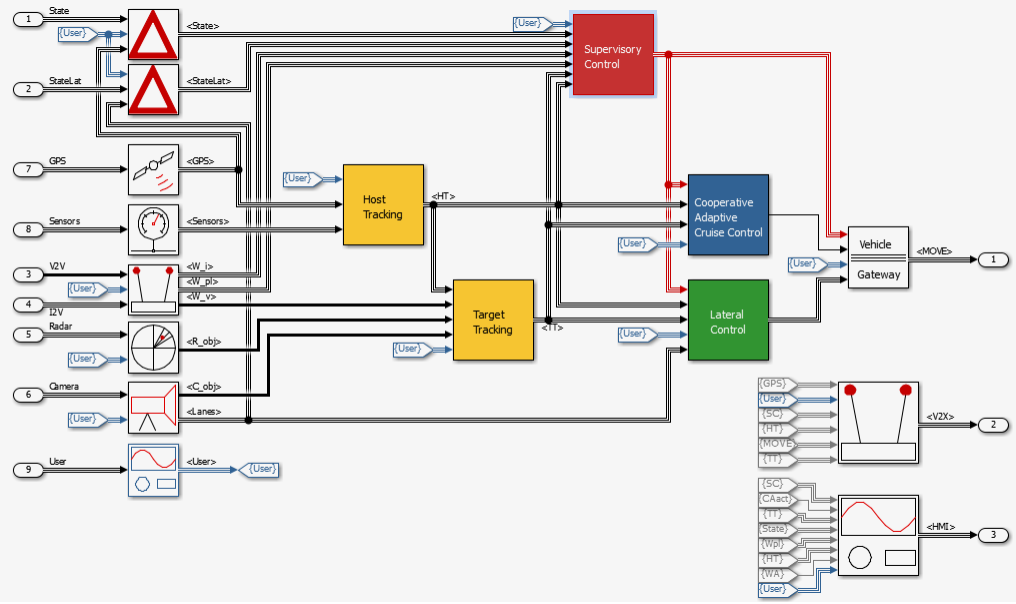

Figure 2.4: System under the controller mask.

the splitting maneuver only two actions have to be initiated. First, if there is a car on the left lane, the ego car decelerates in order to make a gap with the car on the left lane. When this gap is large enough, the actual split maneuver can be initiated. In this case the STOM message will also turn from 0 into 1. From this moment, the car can perform the split maneuver and drives from the right lane to the left lane.

Also the longitudinal controller, lateral controller and target tracking can be activated or the control mode can be adjusted in the user interface. For the longitudinal controller, the user can choose between three types of controllers. There can be chosen between a CC, ACC and CACC controller. CC stands for Cruise Control, ACC stands for Adaptive Cruise Control and CACC stands for Cooperative Adaptive Cruise Control. The lateral controller has three operating modes: vehicle following, lane keeping and spare. The first vehicle is always operating in the lane keeping mode. The other cars are either operating in the vehicle following mode of the lane keeping mode depending on the maneuver that will be initiated. The controller can be switched on and off. Also the lateral controller type can be selected. There are three lateral controller types: steering angle control, yaw rate control and spare. In the spare case, no lateral controller is selected.

For the target tracking, there are four different types of tracking: virtual reference vehicle, direct measurement, multiple target tracking and spare. The first vehicle is always the virtual reference vehicle and all the other vehicles are in the multiple target tracking mode. Multiple target tracking can see up to 6 objects using the camera and radar. Direct measurement is for simulation and test purposes only so it won't be used a lot. Also the host tracking type can be changed, which has four different types of tracking: direct measurement, simple host tracking, advanced host tracking and spare. At direct measurement, the position and yaw rate are directly taken from the GPS while at simple host tracking, the yaw angle and position are estimated by means of observers. At advanced host tracking, all states are chosen using a single extended Kalmann filter. Only simple

host tracking is used because advanced host tracking is not fully tested yet and direct measurement is for simulation and test purposes only.

Start/pause button A new menu is developed by TNO in order to run the model. The menu, which can be found in Figure 2.5, is basically a figure with buttons and checkboxes. The buttons and checkboxes are linked to callback functions in order to execute a function when the button is pressed. With the upper checkboxes, there can be decided what will be executed when the start button is pressed. When 'Calculate Parameters' is checked, the parameter file will be executed which means that all parameters are calculated and stored at the workspace. When 'Run simulation' is checked, the simulation will be executed when the start button is pressed. There are also checkboxes for only initializing the model and saving the simulation data.

The simulation is run from a m-file. The name of this m-file can be entered in the 'Simulation file' box. By pressing edit, the m-file be opened in order to make adjustments. The 'Stop' button stops the simulation when it is running. From the start/pause menu, one can also make plots of the position, inter-vehicle spacing and velocity of the vehicles by pressing plot. Also for the plotting, a m-file have to be made in order to execute the plotting. The name of this m-file can be entered in the 'Plot file' box just like with the 'Simulation file'. Possible warnings and messages are displayed in the frame below the plot button. At last there are 'Help', 'Undo', 'Apply' and 'Close' buttons for displaying a help window, reloading the menu settings as they were at initialization, saving menu settings and closing the menu.

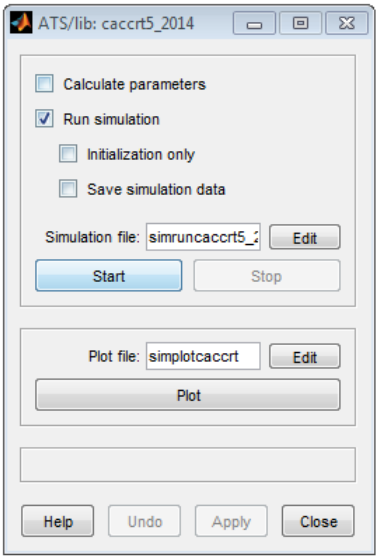

Figure 2.5: Start/pause menu.

2.3 Parameter file

The parameter file contains all the parameters and initial conditions of the model. The parameters can be tuned by changing the value in the parameter file. The parameters and initial conditions are taken from the parameter file during the initialization of the model. This means that during the simulation, no parameters can be changed in the parameter file. The parameter file contains some general simulation set-up parameters like the number of vehicles and simulation time. Next the parameters of the communication systems are introduced and after that the switches for the user interface, parameters for the controller subsystem and initial position with lane information are introduced. The parameters of the controller subsystem are the controller gains, filters and other parameters that are needed for the controllers, supervisory control and tracking systems.

For making a simulation model, the switches for the user interface and the initial positions are the most important. The switches and initial positions are defined as vectors with the same length as the number of vehicles n_v. Each vehicle has its own ID number varying between 1 and n_v and can therefore take its initial conditions from the vector. In this way the initial conditions for all cars don't have to be the same. The lanes are defined as straight lanes directed according to an angle. The road has three lanes with a width of 3.5 $[m]$, the middle lane is defined at $y = 0$ $[m]$, the left lane is defined at $y = -3.5$ $[m]$ and the right lane is defined at $y = 3.5$ $[m]$. For all the cars to drive in a straight line, the y-coordinate is set to 0. The x-coordinate is dependent on the initial velocity.

2.4 Summary

The parameters cannot be changed during the simulation using the parameter file. This means that, in order to initiate a split or merge maneuver, the values of those switches have to be changed in the user interface of the car itself. The same counts for changing the controller modes or cruiser velocity during the simulation. This is however not very user-friendly because the model contains multiple cars and is very complex. Going to the user interface of each car to change the variables during the simulation is not useful so therefore another more user friendly approach is wanted.

3. New approach of simulating

As mentioned in the previous chapter, the parameters and initial conditions are taken from the parameter file during the initialization of the model, which means that the parameters can not be adjusted during the simulation by using the parameter file. The only way to change the parameters is by changing them in the model during the simulation. Therefore a more user-friendly approach is wanted. At first, the objectives which the simulation model should meet are explained. After that two approaches are discussed which are used to meet these objectives. The best approach is used in the next chapter to show the results.

3.1 Objectives

A simulation model with some sort of menu needs to be made in order to make the current model more user-friendly. Changing parameters and initiating maneuvers through online simulations results in a more user-friendly simulation model. Therefore, some objectives are determined that the simulation model has to meet. The most important objective is that the simulation model can perform one of the three main tasks, which is a merging maneuver. Therefore the three actions that should be initiated for merging have to be controllable in the simulation model, for example by means of buttons in the menu. Besides that it is also desired to be able to do a simple splitting maneuver which means that the two actions that should be initiated for splitting have to be controllable as well.

When multiple cars are going to perform a merge maneuver it needs to be possible to select which car should do the maneuver first or to let the cars do the maneuver simultaneously. So another objective is to select which car is going to do the maneuver during the simulation. In order to see if the car is able to do the merge maneuver, the STOM message also has to be displayed. This means that when multiple cars are doing the maneuver, also multiple STOM messages should be displayed.

Besides the controlling of the model during the simulation, it would also be nice to be able to easily change some initial conditions. The initial conditions that are changed the most are the initial position, initial velocity and the cruiser speed. The initial x-coordinate in the original model is dependent on the initial velocity. However, it is desirable to be be able to change the initial x-coordinate of the merging vehicle as well. Therefore it is desired to keep the x-coordinate dependent on the initial velocity but to have still some room to replace the merging vehicles.

For doing the split maneuver, the lateral control mode differs per vehicle. All cars, except for the car that is going to perform the split maneuver, should have ‘lane keeping’ as control mode. The car that is going to perform the split maneuver should have ‘vehicle following’ as control mode. Therefore it is desired to be able to change the lateral control mode of each vehicle. In some cases it is also desired to switch off the lateral control mode which means that this switch should also be implemented in the menu of the simulation model. At last the number of cars should be easily changeable, so adding cars should not have to result in much extra work in changing the simulation model.

Two approaches have been used to see which way of controlling the model is the most user-friendly. The number of cars in the original model has been brought back first to five cars instead of ten. This has been done in order to test the simulation model and visualize the results clearly. The two approaches that have been used are the menu approach and the control panel approach.

3.2 Menu approach

At first, the menu approach will be discussed. The menu approach consists of a button that opens a parameter menu. This approach is basically the same as the approach TNO used to make the start/pause menu. The button and the parameter menu can be seen in Figures 3.1 and 3.2 respectively. The button is basically a masked subsystem that is linked to a m-file. The subsystem itself is empty and is not linked, by means of 'from/goto' blocks, to any other parts of the model. This has therefore the advantage that it can be copied to every other simulation model without adjusting the model.

Figure 3.1: Parameter menu button.

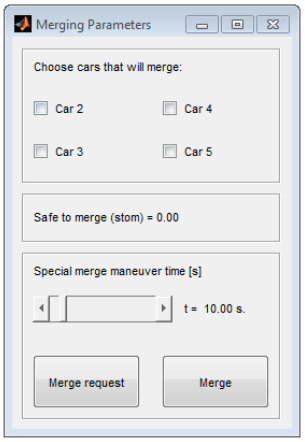

Figure 3.2: Parameter menu.

Like with the start/pause menu, buttons and checkboxes are added to execute or select types of maneuvers. These buttons and checkboxes are linked to callback functions in the m-file in order to execute that function when the button is pressed. Using the command 'setparam' the value of a 'constant block' in the simulation model can be changed during the simulation, so for this reason no physical connection needs to be made to the desired 'constant block'. The menu is divided into three parts; in the first part the cars can be selected which are going to do the maneuver, in the second part the STOM message is displayed and in the last part the buttons are implemented for doing the merge maneuver.

In the first frame, the cars that are going to do the maneuver can be selected. The first car cannot be selected because this vehicle is the reference vehicle. The checkbox of a car needs to be checked before the start of the simulation. When the simulation starts, the car will drive on the left lane which means that the initial y-coordinate is changed from 0 to 3.5 m. The third frame can than be used for the execution of the merge maneuver. At first, before the start of the simulation, the time at which the car adjusts its velocity to the platoon velocity can be defined with the slider. During the simulation, the 'Merge request' and 'Merge' buttons can be used to send out the merge request and for initiating the actual merge maneuver. When the merge request is send to the other cars, the ego car and rear cars start to decelerate in order to make a gap with the vehicles on the right lane.

In the second frame the STOM message is displayed. When the STOM message is equal to 1, the gap between the vehicles is large enough and it is safe for the vehicle to do the maneuver. Therefore, this signal should change from 0 to 1 during the simulation, which means that the signals should be updated in the menu during the simulation. From Figure 3.2 can be seen that only the merging maneuver is implemented in the menu. All the other objectives, like the splitting and initial conditions have not been implemented. This is because during the developing of this menu, it became clear that this is not the right approach. When all the other objectives would be added to the menu, the menu would become very large and not very clear. Also when extra cars would be added to the model, the m-file which executes the menu should be adjusted. Multiple buttons and checkboxes should be added when extra cars are added. This is a lot of work, especially for someone who is not familiar with the m-file of the menu, and therefore not user-friendly.

3.3 Control panel approach

Because the menu approach is not very user-friendly, another approach has been conceived. In this approach, a control panel is used to control the model. Like the menu approach, the control panel approach makes use of a subsystem. This subsystem is however not linked to a m-file, but a control panel with buttons opens up when the subsystem is double clicked. The subsystem is in the top layer of the simulation model. Signals coming from the control panel are transported to the cars by means of 'from/goto' blocks.

Figure 3.3: Control panel.

The control panel itself is given in Figure 3.3. The control panel consists of multiple buttons and masked subsystems. When the masked subsystems are double clicked, a window opens in which parameters can be selected. On the bottom left there is a masked subsystem in which the initial velocity can be selected. For each car there is a masked subsystem in which the initial conditions can be selected and there are a number of buttons to initiate maneuvers. Each vehicle has also two LED's to display messages.

Position Determination button The first button that will be used is the 'Position Determination' button. This button is the lower left green button in Figure 3.3. When this button is double clicked, the menu given in Figure 3.4 is opened. In this menu, the initial velocity can be entered. The x-coordinate is determined using the initial velocity so therefore all cars have the same initial velocity.

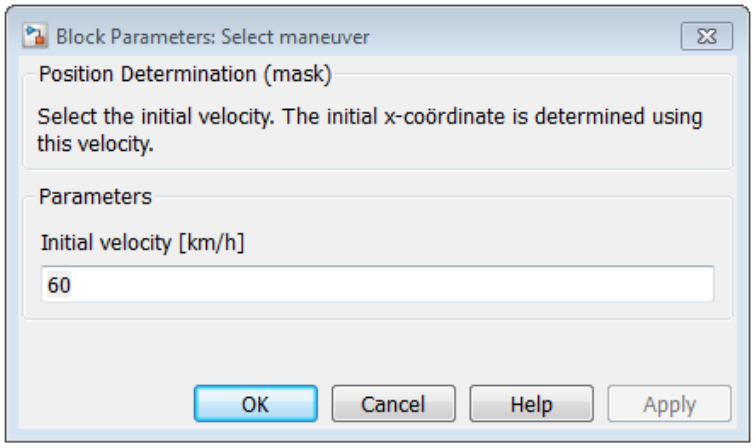

Figure 3.4: Position Determination menu.

The variable selected in the menu is introduced in a 'constant block' under the mask. The variable v_0 is the initial velocity in $[km/h]$. The initial x-coordinates of the cars are calculated in this subsystem using the initial velocity.

$$L_s = L + r + h_u v_0$$

$$L_p = \sum_{i=1}^{n_v} L_s$$

$$q_0(k) = L_p - \sum_{i=1}^{k} L_s, \quad k = 1, 2, .., n_v$$

$$x_0(k) = q_0 cos(\psi_0); \tag{3.1}$$

The calculation of each x-coordinate is given in (3.1). In this equation, L is the vehicle length in $[m]$, r the standstill distance in $[m]$, h_u the user-time headway in $[s]$ and ψ_0 the initial vehicle heading in $[rad]$. The initial velocity v_0 is in $[m/s]$ in this equation instead of $[km/h]$. At first the slot length of each vehicle L_s is calculated, this is the distance the car needs for safe driving at a given velocity. The total platoon length of all cars L_p is then calculated. The initial x-coordinate of each car is then calculated by subtracting the platoon length with the cumulative sum of the slot lengths and multiplying that with the cosine of the initial vehicle's heading. Because the road is supposed to be a straight lane, the initial heading is supposed to be 0. When a car is placed on the left lane, so the y-coordinate of the vehicle is larger than 0, the difference between the x-coordinates of two successive cars is half the difference of two cars that are driving on the same lane. This is shown in Figure 3.5. From this figure it can be seen that the vehicle on the left lane is placed between two vehicles on the right lane.

Initial Conditions button The red subsystems on top of the push buttons are the subsystems to set the initial conditions for each car. When these buttons are double clicked, a menu opens up where the initial conditions can be entered. The 'Initial Conditions' menu can be found in Figure 3.6. At first the car ID needs to be entered. This car ID links this menu and the set of buttons to the desired car. Next, the cruiser velocity can be chosen, this is the velocity the car is aiming for during the simulation. When the cruiser velocity is larger than the initial velocity, the car will accelerate. Each car has the option to make a maneuver or to do no maneuver. This can be chosen in the pop-up bar at 'Select maneuver'. When no maneuver is chosen, the push buttons of the vehicle are not active and nothing will happen when they are pressed. When the maneuver option is chosen, the car is able to make a split or merge maneuver so all push buttons are active.

Next, there are two options for the lateral controller. The lateral controller can be switched on or off and the user lateral control mode can be selected with a pop-up bar. The lateral control mode can be varied between 'lane keeping', 'vehicle following' and 'spare'. At 'lane keeping' the vehicle follows the lane it is driving, at 'vehicle following' the ego vehicle follows the vehicle in front of it and at 'spare' the lateral controller is not active. The first vehicle has always 'lane keeping' as lateral control mode because the first vehicle is the reference vehicle. For the splitting maneuver, all the vehicles should have the lateral control mode 'lane keeping' except for the vehicle that is going to perform the split maneuver. Otherwise the vehicles behind the splitting vehicle will also perform the split maneuver.

At last, the initial conditions can be selected. When the car is driving on the left lane, the x-coordinate of the car with respect to the cars on the right lane can be changed. This is done by means of a value varying between 0 and 1. When this value is 0, the merging car is placed right besides its frontal car and when this value is 1, the car is placed right besides its rear car. Also the initial lateral position of the vehicle can be changed. When the vehicle is driving on the left lane, the lateral position should be 3.5 $[m]$.

The values that are selected in this menu and the values of the push buttons are bundled into a 'bus' and connected to the cars by means of a 'goto tag' and a 'from tag'. The name of the 'goto tag' is linked to the car ID that is selected in the menu. In that case when more cars are added to

the simulation model, the 'Initial Condition' block can be copied. Giving this 'Initial Condition' block another car ID links the block automatically to the right car, which makes the control panel more user friendly.

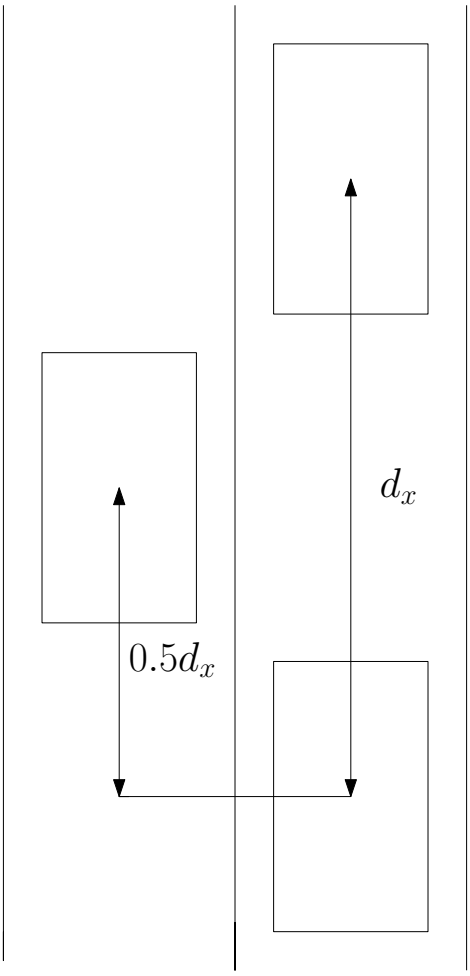

Figure 3.5: Distances of driving on two lanes.

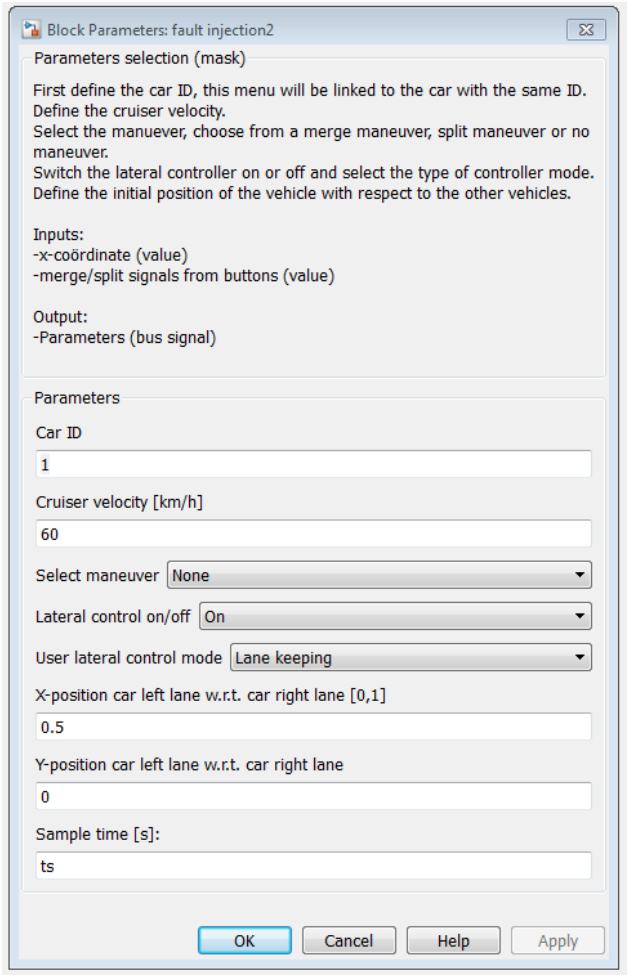

Figure 3.6: Initial Conditions menu.

Push buttons For each car, there are five push buttons for initiating the maneuvers. The push buttons can be found in Figure 3.7. When the push buttons are pressed, a step or a pulse from 0 to 1 is sent to the corresponding signal in the 'Initial Conditions' block. For the merge maneuver, the swMerge, swMergeReq and swDoMerge buttons are used to initiate the actions. First there is the swMerge button which initiate the first action of the merge maneuver. When this button is pressed, the vehicle tunes its velocity to the platoon velocity. This signal is a step from 0 to 1. In order to send out the merge request, the swMergeReq button needs to be pressed. This signal is pulse of 25

seconds from 0 to 1. When the gap between the vehicles is large enough, the swDoMerge button can be used to initiate the actual merge maneuver. This signal is a pulse of one sample time.

For the split maneuver, the swSplit and swDoSplit buttons are used to initiate the actions. When the swSplit button is pressed, the vehicle decelerates in order to make a gap with a possible car on the left lane. This signal is a step from 0 to 1. When this gap is large enough, the swDoSplit button can be used to initiate the actual split maneuver. This signal is a pulse of one sample time. From Figure 3.3 it can be seen that there are no buttons and LED's for the first car. This is because the first car is a (virtual) reference vehicle and therefore not able to do any maneuvers.

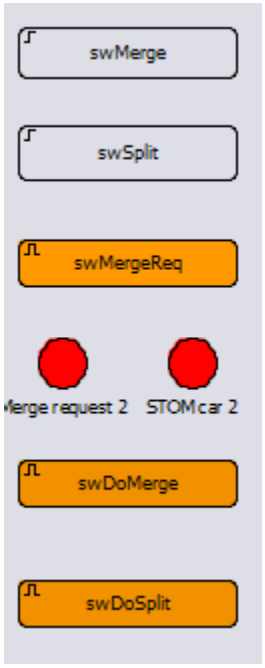

Figure 3.7: Push buttons and LED's.

STOM messages The last items of the control panel are the LEDs that display the STOM messages and the merge request messages. These LEDs can be found between the swMergeReq button and the swDoMerge button in Figure 3.7. The STOM and merge request messages are coming from a 'goto tag' in each car and are linked to a 'from tag' in the LED. When the STOM message or the merge request message turns from 0 into 1, its LED turns from green to red. This means that when the STOM LED of a certain vehicle is red, it is safe to maneuver for that vehicle and the swDoMerge or swDoSplit button can be pressed. When the merge request LED of a vehicle is red, it means that the vehicle has sent out a merge request to other vehicles.

3.4 Summary

Looking at both approaches, it can be concluded that the control panel approach is the most user-friendly one. This approach is clear and meets all the objectives. Besides that, it is easy for the user and someone who does not understand the original model can perform simple simulations using the control panel. Adding an extra car is much easier with the control panel because only a 'Initial Conditions' block, the push buttons and LED's need to be added. The 'Initial Conditions' menu can than easily be linked to the right car by selecting the right car ID. This is much easier and more user friendly than adjusting a large m-file. A manual for using the control panel can be found in Appendix A. Also a control panel with only two buttons, one for the merge maneuver and one for the split maneuver, is made for people who are not familiar with the simulation model. By pressing one of the two buttons, the split or merge maneuver is automatically performed. An overview of this control panel can be found in Appendix B.

4. Results

The control panel is tested in order to see if it functions in line with the proposed objectives. The results of those tests are given in this chapter. At first the simulation of merging one vehicle is tested. After that the simulation of merging two vehicles in a synchronous and asynchronous manner is tested. At last the splitting maneuver is tested. The configurations of these platoons is given in Appendix C. At all simulations, car 1 is the lead vehicle.

4.1 Merging maneuver of one vehicle

In this section the merging of one single vehicle in a platoon of four vehicles is tested. The third car is used to perform the merge maneuver. The car is placed on the center of the left lane so its initial lateral position is 3.5 $[m]$. All cars have the same cruiser velocity. In order to show the results, two figures are made.

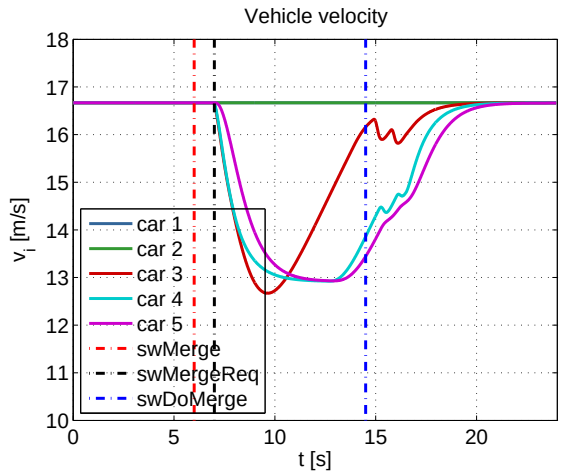

Figure 4.1: Vehicle velocity in a single merging maneuver.

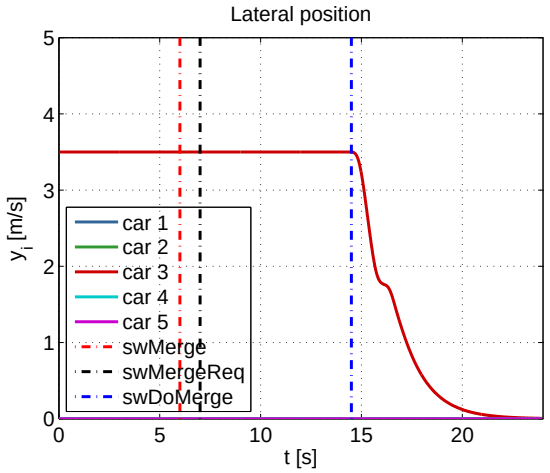

Figure 4.2: Lateral vehicle position in a single merging maneuver.

Figure 4.1 shows the velocity of all vehicles. This figure shows also the times at which the actions are initiated. At $t = 6.0$ $[s]$ the first action is initiated. The car which is selected to do the merge maneuver tunes its velocity with the platoon. The velocity of this car is equal to the platoon so for that reason there is no change in the velocity. At $t = 7.0$ $[s]$, the second action is initiated and the car sends out a merge request. From this moment, cars 3, 4 and 5 starts to decelerate in order to make a gap between car 2 and car 3. When this gap is large enough, car 3 starts to accelerate to its cruiser velocity but car 4 and car 5 continue the deceleration in order to make a gap between car 3 and car 4. When this gap is large enough, car 4 and car 5 start to accelerate back to their cruiser speed.

When both gaps are large enough, the STOM message becomes 1 and the third action can be initiated. Car 3 can start with performing the merge maneuver. This can better be shown by the lateral position of the cars. From Figure 4.2, it can be seen that car 3 has an initial lateral position of 3.5 $[m]$ and all other cars have an initial lateral position of 0 $[m]$. At $t = 14.5$ $[s]$ the car starts with the merge maneuver. It can be seen from Figure 4.2, that the lateral position of the

car goes from 3.5 $[m]$ to 0 $[m]$ within 9 seconds. It takes 9 seconds for the car to merge from the left lane to the right lane and the total maneuver, including sending the request, takes 16 seconds. The merging maneuver is however not really a smooth maneuver which is the result of the lateral controller. It can be seen that at half way of the merge maneuver, the cars steers a little bit back to the left lane. This is something that should be adjusted in the control block.

4.2 Synchronous merging of two vehicles

Next, two cars will perform a merging maneuver. For this, two approaches are possible; the cars perform the maneuver simultaneously or they perform the maneuver one after the other. First a simultaneous merging maneuver of the simulation model is tested. Car 2 and car 4 will do the merge maneuver and are therefore placed on the left lane, so both cars have an initial lateral position of 3.5 $[m]$. All cars have the same cruiser velocity. The two figures below of the vehicle velocity and the lateral position show the results.

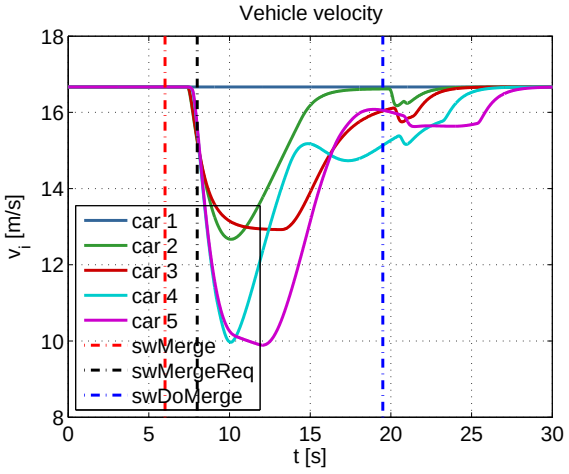

Figure 4.3: Vehicle velocity in a simultaneous merging maneuver.

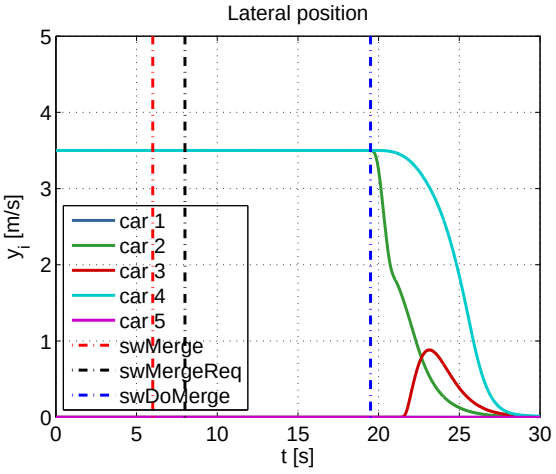

Figure 4.4: Lateral vehicle position in a simultaneous merging maneuver.

Figure 4.3 shows the velocity of all cars. The velocities of the merging cars are equal to the platoon velocity, so for that reason there is no change in the velocity when the first action is initiated. It can be seen that when the second action is initiated, all cars except the leading vehicle start to decelerate. The deceleration of car 2 and car 3 is much smaller than the deceleration of car 4 and car 5. This is because in this case four large gaps need to be made instead of two so the distance between the lead and the last vehicle should be larger than in the case where only one vehicle is merging. The deceleration of the merging cars takes less time than the deceleration of the gap making cars. This is because car 3 and car 5 also need to make a gap between car 2 and car 4. Car 2 starts to accelerate once the gap between car 1 and car 2 is large enough but car 3 still needs to make the gap between car 2 and car 3. The same counts for car 4 and car 5.

Figure 4.4 shows the lateral position of the cars. It can be seen from this figure that the initial

lateral position of the merging cars is 3.5 $[m]$, so at the center of the left lane. When the third action is initiated, so the actual merging maneuver is initiated, both cars on the left lane start to merge to the other lane. It can be seen that both cars do not follow the same profile, car 2 follows the same profile as car 3 in the previous section but car 4 has a smoother profile. Also the car that is driving in between the merging vehicles on the right lane seems to have a lateral movement which is strange because the gaps are large enough for the vehicles to merge. The time it takes from sending the merge request until the end of the merging is 22 seconds. This is a bit longer than the single merging maneuver but that is because a larger distance between the lead and the last vehicle should be created in order for two vehicles to merge.

4.3 Asynchronous merging of two vehicles

Now a non-simultaneous merging maneuver of the simulation model is tested. Car 2 and car 4 will do the merge maneuver and are therefore placed on the left lane, so both cars have an initial lateral position of 3.5 $[m]$. First car 2 will perform the merge maneuver and afterwards car 4 will perform the merge maneuver. Car 4 has a larger cruiser velocity than the other cars to make sure that car 4 won't drive besides another car, where it is not visible for the other car. Also car 4 needs to have 'lane keeping' as lateral control mode, otherwise it will follow car 2 during its merge maneuver.

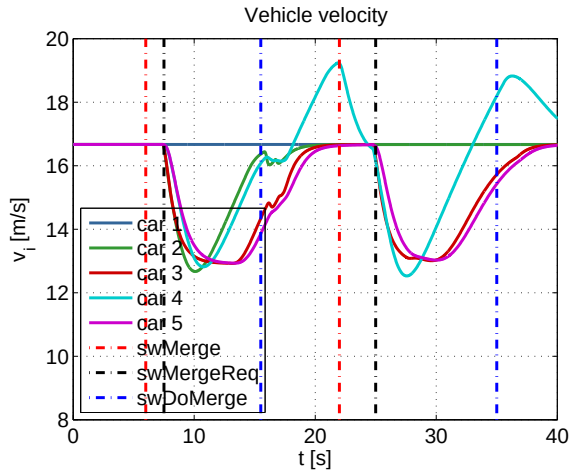

Figure 4.5: Vehicle velocity in a non-simultaneous merging maneuver.

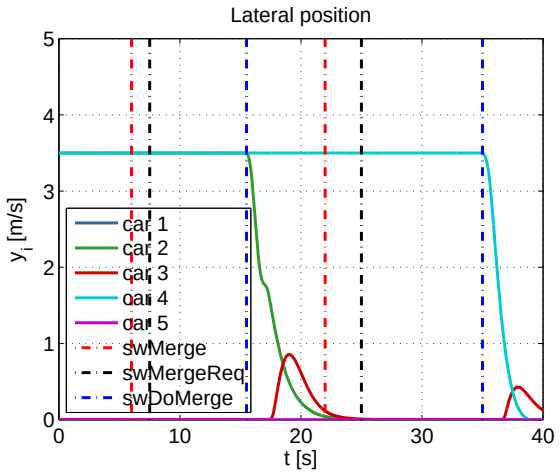

Figure 4.6: Lateral vehicle position in a non-simultaneous merging maneuver.

Figure 4.5 shows the velocities of all vehicles and Figure 4.6 shows the lateral position of all vehicles. Car 4 is driving behind car 2 and has at this moment the same velocity as car 2. When car 2 sends out the merge request, all cars will decelerate except the lead vehicle. During this deceleration, car 4 overtakes car 3 because the distance between car 2 and car 4 stays equal. When the gap between the lead vehicle and car 3 is large enough, car 2 can start with the actual merging maneuver. Figure 4.6 shows that the merging maneuver starts at $t = 15.5$ $[s]$.

When car 2 has merged to the other lane, car 4 has no other car in front of it so it starts to

accelerate to its cruiser velocity. When car 4 is driving between car 2 and car 3, the car starts to tune its velocity with the platoon. This is initiated at $t = 22.0$ $[s]$. At $t = 25.0$ $[s]$, the velocity of car 4 is equal to the platoon and car 4 sends a merge request to the other vehicles. From this moment, car 3, car 4 and car 5 starts to decelerate in order to make a gap between car 2 and car 3. When this gap is large enough, the lateral control mode of car 4 should be switched back to 'vehicle following' and car 4 is able to perform the actual merge maneuver.

4.4 Splitting maneuver

For the splitting maneuver, the second vehicle is placed on the left lane and all the other cars drive on the right lane. All cars have the same cruiser velocity. The third car is selected to do the split maneuver and should drive to the left lane. This means that the lateral position of car 3 should go from 0 $[m]$ to 3.5 $[m]$. The results are shown in the figures below.

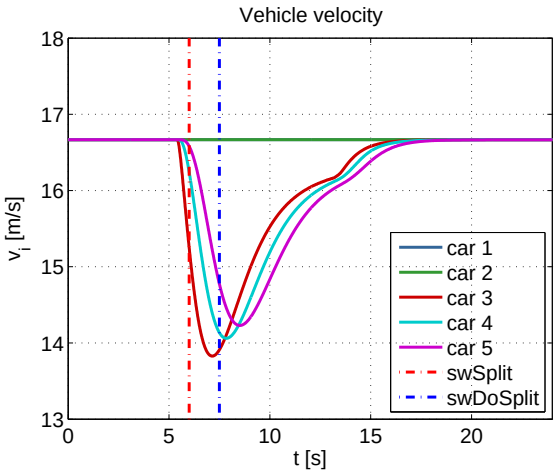

Figure 4.7: Vehicle velocity in a split maneuver.

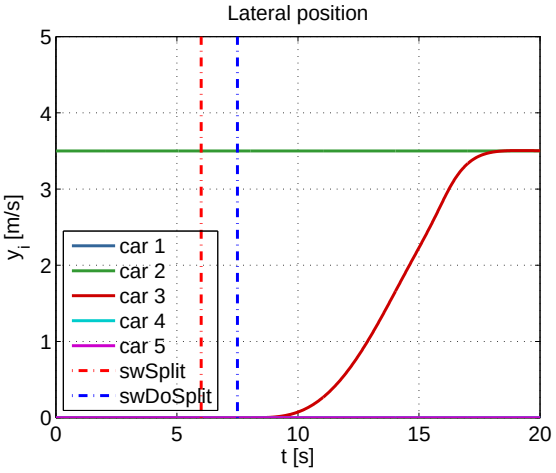

Figure 4.8: Lateral vehicle position in a split maneuver.

Figure 4.7 shows the velocity of all vehicles and Figure 4.8 shows the lateral position of all vehicles. At $t = 6.0$ $[s]$ the first action is initiated. From Figure 4.7 it can be seen that car 3 decelerates in order to make a gap with car 2. This means that also car 4 and car 5 need to decrease their velocity. When the gap between car 2 and car 3 is large enough, the safe to maneuver message becomes equal to 1 which means that it is safe to do the split maneuver. From this moment car 3, car 4 and car 5 accelerate back to their cruiser velocities. At $t = 7.5$ $[s]$ the second action can be initiated which means that car 3 can start with performing the split maneuver. From Figure 4.8, it can be seen that initially car 2 is driving on the left lane and the other cars are driving on the right lane. When the second action is initiated, car 3 starts with the split maneuver and it will be on the left lane after 10 seconds. All other cars remain at their initial lateral position. The total split maneuver takes approximately 12 seconds.

5. Conclusion and recommendations

5.1 Conclusion

A simulation model is made by TNO which is able to perform longitudinal and lateral maneuvering. The simulation model consists of multiple vehicles in order to make a platoon and perform the maneuvers. Two maneuvers can be performed using this simulation model: a merge maneuver and a split maneuver. For the merge maneuver, the vehicle on the left lane wants to merge with the vehicles on the right lane. Three actions need to be initiated in order to perform the merge maneuver. For the split maneuver, the vehicle wants to drive from the left lane to the right lane. Two actions need to be initiated in order to perform the split maneuver.

The parameters of the simulation model are taken from a parameter file during the initialization of the simulation. Therefore the parameters can not be changed during the simulation. Each car has a user interface where the signals of the controllers are displayed and the vehicles actions, controller types and some initial conditions are defined. In order to initiate the the actions of a merge and split maneuver, the values of their switches need to be changed in the user interface of the car itself. This is not very user-friendly so therefore another approach is wanted.

A number of objectives are determined that the simulation model has to meet. The main task is that the simulation model can perform multiple approaches of merging or splitting maneuvers, like multiple vehicles that perform the maneuver simultaneously or one vehicle after the other. Therefore the three actions that need to be initiated for the merging maneuver and the two actions that need to be initiated for the splitting maneuver need to be controllable by means of buttons. Besides that it is also desired to change the initial positions and velocities of the vehicles and to define the lateral control mode of each car. The last objective is that the number of cars should be changeable, so adding or removing cars should not result in much extra work in changing the simulation model.

Two approaches have been used of to make the model more user friendly: the menu approach and the control panel approach. The menu approach consists of a button that opens a parameter menu. The advantage of the menu approach is that it can be copied to every other simulation model. Buttons and checkboxes, which are linked to an m-file, are added to the menu to execute or select types of maneuvers. The menu is however not very clear and when vehicles are added to the model, buttons and checkboxes need to be added which is a lot of work because the m-file needs to be changed. This makes the menu approach not very user-friendly.

The other approach is the control panel approach. The control panel consists of multiple push buttons, LED's and blocks that open a menu. For each car there are five buttons to initiate the actions of the merge and split maneuver. Each car has also two LED's that display the STOM message and the merge request and each car has its own 'Initial Conditions' menu in which the initial position, lateral control mode, car ID and the maneuver can be selected. The first car has no push buttons because it is not able to do any maneuver. This approach is much clearer and more user-friendly than the menu approach. Adding an extra car is much easier with the control panel because only a 'Initial Conditions' block, the push buttons and LED's need to be added. The 'Initial Conditions' menu can than easily be linked to the right car by selecting the right car ID. This is much easier than adjusting a large m-file.

At last, simulations are performed in order to see if the control panel meets the objectives. Four different simulations are performed, including different approaches of merging maneuvers and a

split maneuver. From the results, it can be seen that the vehicles have performed their maneuvers well so the control panel meets the objectives. The maneuvers can be initiated easily and the parameters can be adjusted during the simulation. However, it can be seen that not all maneuvers are smooth and in some cases a vehicle had a lateral movement when it was not supposed to. This is however not caused by the control panel but by the way the the maneuver is defined in the control block.

5.2 Recommendations

As mentioned in the conclusion, some maneuvers are not smooth. The merging maneuver is not really a smooth maneuver, at half way the merge maneuver the cars steers a little bit back to the left. This is something that should be adjusted in the control system block. In the case of a merging maneuver with two vehicles, one of the cars on the right lane has a lateral movement during the merging of the two cars. This is probably caused by the control system so this is also something that needs to be looked into. The controller gains are not altered during this project and therefore it is not desired to implement them in the control panel. When it is however desired to change the controller gains of each vehicle while simulating, it is useful to implement them in the control panel. At last, the simulation model is only tested on straight lanes. It is useful to test the simulation model also on curved lanes in order to see how the cars perform their maneuvers under those conditions.

References

[1] Maas S., *Description of Work, Interoperable GCDC AutoMation Experience, I-GAME* 2013

[2] Maas S., Schrijnemakers L., Didoff J., Aparicio A., *Grand Cooperative Driving Challenge. Consulted on 2 September 2014.* http://www.gcdc.net/

A. A manual for the control panel

In this appendix, a manual is given for operating the control panel.

1. Select the initial velocity in the 'Position Determination' menu. The initial x-coordinates are determined using this initial velocity so therefore the initial velocity of all vehicles are the same.

2. For each car, open the 'Initial Conditions' block and set the following conditions:

   - Choose the car ID, which links the 'Initial Conditions' block to the right car.
   - Choose the cruiser velocity in $[km/h]$.
   - Select whether this car is going to perform a maneuver. When 'maneuver' is chosen, the push buttons are active. When 'none' is chosen, the push buttons are not active so the car won't be able to do a maneuver.
   - Switch the lateral controller on or off.
   - Select the lateral control mode. When 'lane keeping' is chosen, the vehicle follows the lane it is driving on. When 'vehicle following' is chosen, the vehicle follow its frontal vehicle. When 'spare' is chosen, the lateral controller is not active.
   - Select the x-position for the car on the left lane with respect to the frontal and rear car on the right lane. This value is varied between 0 and 1. A value close to 0 places the car closer to its frontal car. A value of 1 places the car closer to its rear car.
   - Select the y-position of the car. A y-position of 3.5 $[m]$ places the car on the left lane.
   - Choose the sample time in $[s]$

3. For a merge maneuver:

   - Use the 'swMerge' button to tune the velocity of the merging vehicle with the platoon.
   - Use the 'swMergeReq' button to send out the merge request. The 'Merge request' LED will turn red.
   - When the STOM LED turns red, it is safe for the vehicle to do the merge maneuver.
   - Use the 'swDoMerge' button to do the actual merge maneuver.

4. For a split maneuver:

   - Use the 'swSplit' button to make a gap with the car on the left lane.
   - When the STOM LED turns red, it is safe for the vehicle to do the split maneuver.
   - Use the 'swDoSplit' button to do the actual split maneuver.

B. Simplified control panel

A simple control panel is made for people who are not familiar with the simulation model. This simple control panel has only a button for the merging maneuver and a button for the splitting maneuver. Pressing the merge button will initiate the three merge actions one after the other and pressing the split button will initiate the two split action one after the other. The merge and split maneuver will automatically be performed when its button is pressed. The simple control panel is given in Figure B.1.

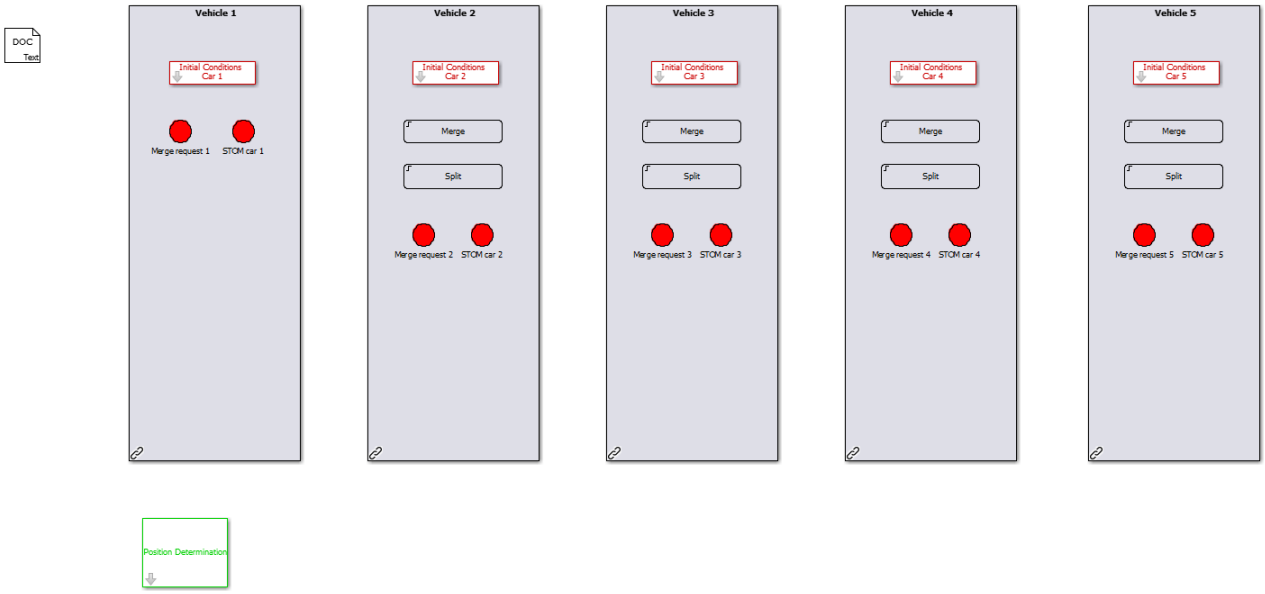

Figure B.1: Simple control panel.

C. Platoon configurations used in simulations

At Chapter 3, three configurations of platoons were used to perform the simulations. In this appendix, those configurations are displayed. For the first simulation, the third vehicle was selected to do a merge maneuver and placed on the left lane. This configuration is shown in Figure C.1. For the second and third simulation, the second and fourth vehicles were selected to do a merge maneuver and are therefore placed on the left lane. This configuration is shown in Figure C.2. For the last simulation, the third car was selected to do a split maneuver. The second car was placed on the left lane in order for the third car to make a gap with the second car. This configuration is shown in Figure C.3.

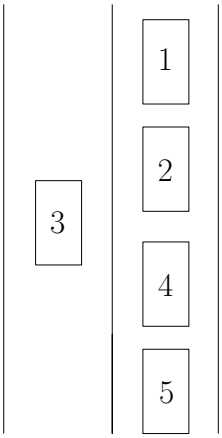

Figure C.1: Configuration of a platoon with one merging vehicle.

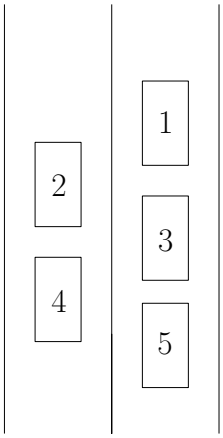

Figure C.2: Configuration of a platoon with two merging vehicles.

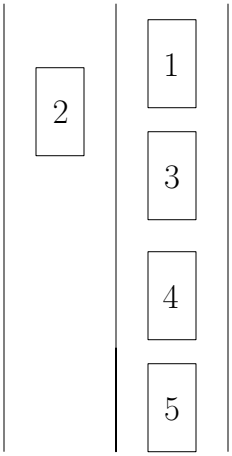

Figure C.3: Configuration of a platoon with one splitting vehicle.