Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Modelling of traffic on a German highway based on consideration of driver individualities

Daniel Philipp & Patrick Pietsch

Zürich
December 2011

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Patrick Pietsch Daniel Philipp

Abstract

Contents

1 Individual contributions

All investigations have been carried out as a team, meaning regular consultation, information sharing and discussion. However, Patrick Pietsch focused more on creating and implementing the model and its additional features, whereas Daniel Philipp worked with the simulation, generated results and compared them with the real data sets.

2 Introduction and Motivations

2.1 Motivation

Traffic volume has been increasing continuously over the last decades in industrialized countries, for example especially on German highways [3]. Considering economic and environmental consequences it is obvious that an efficient traffic capacity management is a serious issue in modern society. Simulating traffic seems to be a budding option to study the impact of individual driving behavior on the total traffic flow and thus contributes to a development of more sophisticated traffic routing. With this simulation the authors of this paper would like to understand traffic from a fundamental and microscopic point of view on the one hand and on the other hand find out how realistic the simulation is, compared to real traffic data.

Figure 1: Traffic situation on the German highway A8. This is a very common situation nowadays. The picture was found, just 'googling' 'traffic A8'. Source: [1]

2.2 Plan overview

We intend to implement the intelligent driver model (IDM) with the most important extensions that are proposed in the human driver model (HDM) (see [4]). In particular, these features are the spatial and temporal anticipation of drivers, the reaction time and the expectation errors according to the proposal in [4]. Furthermore, we plan to enrich our model with two more basic properties, finally ending up with the individual driver and car model (IDCM) and the two lane model (TLM):

Individual drivers and car engine powers Individual driving styles and engine powers will be implemented, which will aim at a somewhat more realistic approach to reality.

Two-lane Highway The model in [4] only considers a single lane. We intend to simulate a highway with two lanes where cars pass others according to traffic regulations.

Finally we intend to compare our simulation results with measurements, made in 1996 with a test vehicle on the German highway A8 between Stuttgart and Karlsruhe [2]. The empirical data will help to validate the model proposal concerning the behavior of individual drivers in different traffic situations.

2.3 Fundamental questions

There are several fundamental questions, that we would like to investigate with our simulation. In particular these are the following:

- How do different driving styles affect traffic?
- What is the impact of cars with different engine powers?
- How does the traffic flow change when considering the possibility of passing other cars according to traffic regulations?
- Are there parameters in our models that might reproduce previously recorded data from A8, a German highway?

2.4 Expected results

We expect to find an approximate agreement with the recorded data since we will make some significant extensions to the HDM by considering individual properties for each car. Clearly, our model considers a perfectly straight road which is naturally not the case in the measured real data. Thus, possible deviations in cruising speed due to braking in bends etc. are neglected. Aside from these obvious facts, we hope to improve the simulated results of IDM, resp. HDM. We hope to be able to implement all the model improvements into our simulation.

3 Description of the Model

Our traffic simulation model is built up step by step, starting from the basics features of the IDM. This model is then extended in several steps, including features of human drivers (HDM), individual driving styles and a two lane highway, allowing for outdistancing. The IDM and the the HDM features are basically taken from [4], whereas the last two features have been worked out by the authors.

3.1 The intelligent driver model (IDM)

The IDM is a microscopic traffic model, meaning that it simulates many single drivers that act deterministic according to a specified behavior. No fluctuations, probabilities or any other human properties are taken into account. Furthermore, all driver estimations of quantities and parameters, like the velocity difference to the car in front, are considered to be exact.
Every diver i evaluates at a given time his own velocity v_i, the distance $s_{i,i-1}$ to the car in front of him and the velocity difference $\Delta v_{i,i-1}$ to the car in front of him and determines his own acceleration a_i according to these input parameters. Explicitly:

$$a_{i,tot} = a_{i,tot}(v_i, s_{i,i-1}, \Delta v_{i,i-1}) \tag{1}$$

So what is now the physical model, for the acceleration, meaning the function $a_{i,tot}(v_i, s_{i,i-1}, \Delta v_{i,i-1})$?
The overall acceleration of a driver is given by:

$$a_{i,tot} = a_{i,free} + a_{i,i-1,traffic} \tag{2}$$

where $a_{i,free}$ describes the hypothetical acceleration of the vehicle i, if the next driver in front would be infinitely far away, and $a_{i,traffic}$ is a term that compensates for the traffic situation. In particular the latter term introduces a deceleration in order to avoid a crash with the driver in front.
The IDM now proposes the following relations:

$$\begin{aligned} a_{i,free} &= a\left[1-\left(\frac{v_i}{v_{end}}\right)^4\right] && (3) \\ a_{i,i-1,traffic} &= -a\left(\frac{s^*_{i,i-1}(v_i,\Delta v_{i,i-1})}{s_{i,i-1}}\right)^2 && (4) \end{aligned}$$

where a is a fixed acceleration parameter, which a driver uses to accelerate from standstill, if no driver is in front and v_{end} is the desired final velocity parameter of the driver. $s^*_{i,i-1}(v_i,\Delta v_{i,i-1})$ is some kind of 'agreeable distance' for the driver to the car in front. From equations 3 and 4 one can see that if $s^*_{i,i-1}(v_i,\Delta v_{i,i-1}) \approx s_{i,i-1}$ then $a_{i,traffic}$ compensates roughly for $a_{i,free}$ at low velocities v_i (compared to the final favored velocity v_{end}), meaning that the resulting acceleration is approximately zero in this case. Furthermore it is obvious that the deceleration decreases, as the driver becomes closer to his desired velocity.
To close the model, the IDM furthermore assumes for $s^*_{i,i-1}(v_i,\Delta v_{i,i-1})$ a term of the form:

$$s^*_{i,i-1}(v_i,\Delta v_{i,i-1}) = s_0 + v_i T_{headway} + \frac{v_i \Delta v_{i,i-1}}{2\sqrt{ab}} \qquad (5)$$

for the favored distance. The parameter s_0 describes the favored distance in standstill, whereas the term $v_i T_{headway}$ describes a dynamical part, which is mainly nothing else than the parameter time to collision $T_{headway}$ that a driver wants to have, if the car in front is decelerating very abruptly and the driver itself keeps his velocity. Finally, the third term $\frac{v_i \Delta v_{i,i-1}}{2\sqrt{ab}}$ is an additional "intelligent term" of the IDM that limits decelerations of the drivers in all cases without danger to the parameter b.
To summarize, there are five parameters considering the physical aspects of the driving behavior of the IDM: a,v_{end},s_0,$T_{headway}$ and b.

3.2 The human driver model (HDM)

There are basically four improvements of the basic IDM model that all take human driving behavior into account. The effects 'finite reaction time' and 'estimation errors' further destabilize the traffic, compared to the IDM, whereas the features 'spatial anticipation' and 'temporal anticipation' tend to improve the traffic flow, meaning a contribution to more continuous driving behavior and a lower crash risk. The causality of the different features is the following: Due to finite reaction times (1) the driver estimates the input quantities with some time delay, therewith making estimation errors (2). Since he is aware of his reaction time errors, he tries to compensate this effect by using temporal anticipation (3). Due to spatial anticipation (4) he does this for several vehicles ahead, combining the results, he obtains. We will therefore show the improvements in this order.

3.2.1 Finite reaction time

Human drivers do not react instantaneously to changes in the traffic situation. We introduce a finite reaction time $T_{reaction}$ by evaluating all input stimuli from equation 1 at time $t - T_{reaction}$.
More explicitly:

$$a_{i,tot,reaction} = a_{i,tot}(v_i, s_{i,i-1}, \Delta v_{i,i-1})|_{t-T_{reaction}} \tag{6}$$

Since this can require to evaluate the input quantities at times, which are between the points of our discrete time mesh, we use linear interpolation between the time points in the mesh that are closest to the required time:

$$a_{i,tot}(t_n - T_{reaction}) = \beta a_{i,tot}|_{t_{n-k}} + (1-\beta) a_{i,tot}|_{t_{n-(k+1)}}$$

with:

$$\beta = \frac{((t_n - T_{reaction}) - t_{n-(k+1)})}{((t_n - T_{reaction}) - t_{n-(k+1)}) + (t_{n-k} - (t_n - T_{reaction}))}$$

as linear weighting coefficient and $t_{n-(k+1)}$, t_{n-k} the points in the time mesh that are closest to $t_n - T_{reaction}$.

3.2.2 Estimation errors

It can be assumed, that human drivers can estimate their own velocity more or less exact, because they have a speedometer. However distances and velocity differences cannot be estimated arbitrary accurate. We therefore use as input quantities not the exact ones, but rather the following:

$$\begin{aligned} s_{i,i-1,est} &= s_{i,i-1} + s_{fluc}\sigma_j|_{t-T_{reaction}} & (7) \\ \Delta v_{i,i-1,est} &= \Delta v_{i,i-1} + v_{fluc}s_{i,i-1}\sigma_j|_{t-T_{reaction}} & (8) \\ v_{i,est} &= v_i|_{t-T_{reaction}} & (9) \end{aligned}$$

where σ_j denotes a standard normal distribution with zero mean and s_{fluc}, v_{fluc} are the corresponding fluctuation coefficients. As it can be seen from equations 7 and 8 the imprecision of the velocity difference is considered to be proportional to the distance to the car in front, whereas the imprecision for the distance itself has just a constant pre - factor. Therewith we have:

$$a_{i,tot,est} = a_{i,tot}(v_{i,est}, s_{i,i-1,est}, \Delta v_{i,i-1,est})|_{t-T_{reaction}} \quad (10)$$

3.2.3 Temporal anticipation

The HDM model further assumes that human drivers are aware of their finite reaction times and act accordingly, meaning that they anticipate their future speed and their distance to the next car in front, assuming constant acceleration heuristics. Consequently we have:

$$\begin{aligned} s_{i,i-1,temp} &= s_{i,i-1,est} - T_{reaction}\Delta v_{i,i-1,est}|_{t-T_{reaction}} & (11) \\ v_{i,temp} &= v_{i,est} + T_{reaction}a_{i,tot}|_{t-T_{reaction}} & (12) \\ \Delta v_{i,i-1,temp} &= \Delta v_{i,i-1,est}|_{t-T_{reaction}} & (13) \end{aligned}$$

Therewith we now have:

$$a_{i,tot,temp} = a_{i,tot}(v_{i,temp}, s_{i,i-1,temp}, \Delta v_{i,i-1,temp})|_{t-T_{reaction}} \quad (14)$$

3.2.4 Spatial anticipation

Clever human drivers do not only take the car directly in front of them into account, but also those even more ahead. For example, if a driver sees a traffic jam, he will decelerate, even if the driver directly in front does not. The effect is included by simply replacing equation 2 by:

$$a_{i,tot,HDM} = a_{i,free} + \sum_{j=i-p}^{i-1} a_{i,j,traffic,HDM} \tag{15}$$

where

$$a_{i,j,traffic,HDM} = a_{i,j,traffic}(v_{i,temp}, s_{i,j,temp}, \Delta v_{i,j,temp})|_{t-T_{reaction}} \tag{16}$$

3.3 The individual driver and car model (IDCM)

The idea was that different driving styles, as well as different engine powers of the cars should be implemented in the model. The overall physical HDM is basically left unchanged in its functional principle but the parameters $T_{headway}$, a, b and v_0 are now categorized in three driver/car groups:

Driver / car type	$\frac{T_{headway}}{T_{headway,normal}}$	$\frac{a}{a_{normal}}$	$\frac{b}{b_{normal}}$	$\frac{v_0}{v_{0,normal}}$
Normal driver	1	1	1	1
Fast driver	0.5	1.3	1.3	1.1
Slow driver	1.5	0.7	0.7	0.9

We assumed that about 70% of the drivers drive normal, 15% drive comparably fast and another 15% drive comparably slowly. These individual driving properties are especially relevant for outdistancing behavior, which we will discuss in 3.4.

3.4 The two lane model (TLM)

Individual driving styles become very interesting, only when drivers have the possibility to outdistance others. This makes the system much more dynamic and interesting. However, at the same time the crash risk is heavily increased for the same reason. The most striking point for a two lane model is to specify the lane change behavior. Several criteria for a lane change were tested, which finally resulted in the following:

Lane change from right to left:

1. **criterion** There is a another car in front of the driver, which is very close.
2. **criterion** The driver wants to drive faster than the car in front is driving.
3. **criterion** The next car in front on the left lane is at least equally far away from the driver as the car directly in front on the right lane.
4. **criterion** No driver on the left lane in the back is too close.

Lane change from left to right:

1. **criterion** No driver on the right lane in front is too close.
2. **criterion** No driver on the right lane in the back is too close.

More quantitatively, 'not too close' means a multiple of the preferred velocity v_0, since a general rule of thumb for driving is to keep at least a distance of $\frac{v_0}{2}$ where v_0 is measured in $\frac{\mathrm{km}}{\mathrm{h}}$ and the resulting distance is given in m. It should be emphasized that a 'motivation' is required to change from the right to the left lane, meaning that being allowed to change is not enough. The driver will only change, if someone in front is disturbing him, because he is driving slower. However, no 'motivation' is required to change from the left to the right lane. Therefore there is a symmetry break between the two lanes, meaning that drivers will stay on the right, if possible. The ban on outdistancing on the right is therefore automatically respected. Besides these lane change criteria, drivers do not act differently on the two lanes, meaning that rules of the individual driver and car model (IDCM) are kept.

4 Implementation

The final version of the TLM contains all features of the preceding models, such that it is sufficient to discuss its implementation. In the following the most important implementation ideas are discussed with special attention to some interesting realization details. The implementation of a second lane with the possibility for the drivers to change between the resulting lanes requires a much more complicated algorithm, compared to the one of a model for a single lane, since the fundamental iteration process needs to be changed. Cars cannot be numerated monotonically, since their order might change continuously during calculation.

4.1 Basic structure and IDCM

The following graphic illustrates the functional principle of the algorithm:

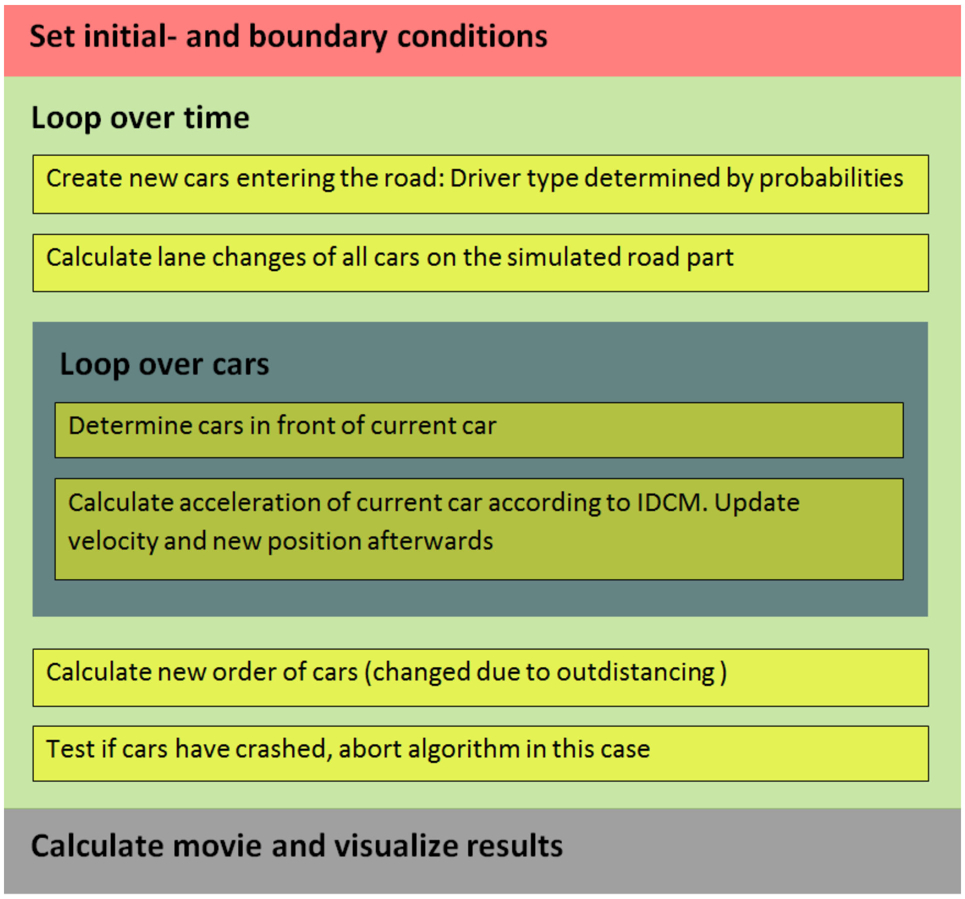
Set initial- and boundary conditions
Loop over time
Create new cars entering the road: Driver type determined by probabilities
Calculate lane changes of all cars on the simulated road part
Loop over cars
Determine cars in front of current car
Calculate acceleration of current car according to IDCM. Update velocity and new position afterwards
Calculate new order of cars (changed due to outdistancing)
Test if cars have crashed, abort algorithm in this case
Calculate movie and visualize results

Figure 2: Illustration of algorithm structure of TLM: The core parts of the algorithm are the two loops, the outer one over time and the inner one over the cars.

The implementation of the IDCM, which specifies the accelerations of all drivers, is done straight forward, meaning that the formulae are just evaluated in the order that has been described and discussed in section 3. It should be mentioned that the currently leading cars on each lane have no drivers in front and therefore have no contribution $a_{traffic}$ in their acceleration specifications. They accelerate therefore only according to a_{free}.

4.2 Storage of car order

As already mentioned in 4 a two lane system requires in each time step the storage of firstly the car order on the road and secondly the information whether a car is on the right or on the left lane. This was done with a 2 x N matrix called 'carresortmatrix', where N is the number of cars on the road. The matrix consists of zeros, and the car numbers. If car number z is located on the right lane, then its number is written in the first row. The matrix entry in the same column, but in the second row is zero. Vice versa, if the car is on the left lane, the second row contains its number and the corresponding matrix entry in the first row is zero. Furthermore the zero - car number pairs in the matrix columns are in descending order, meaning that the first column represents the car in the front, successively followed by the others. An example of such a matrix would be:

$$\text{carresortmatrix} = \begin{bmatrix} 2 & 0 & 0 & 4 \\ 0 & 3 & 1 & 0 \end{bmatrix}$$

In this case the car order (beginning with the leading car) would be 2,3,1,4. Cars number 2 and 4 are on the right lane, whereas car 3 and 1 are on the left lane.
In every time step the algorithm checks for each driver, whether he should change the lane, according to the criteria explained in 3.4. This way a car may change its position within its column of the matrix. Afterwards cars behave individually according the IDCM, where the cars in front of the driver in the same lane may now be found, using the matrix. Due to outdistancing, the different columns of the matrix may change such that the matrix has to be resorted again at the end of the loop. (See figure 2)

4.3 Simulated cars and data storage

Simulated cars Cars enter the simulated road part at a constant rate Q with a constant velocity, that can be chosen arbitrarily. If it is time for a new car to enter the road, a corresponding entry will be generated at the end of the car resort matrix and the driver- resp. car type is determined according to the given probabilities. An additional vector of size N, called "caronroad", contains the information, which cars are on the simulated road part and which not. Furthermore, the first cars on each lane are labeled, because for these cars no contribution $a_{traffic}$ exists (2 = car is the first car on its lane on the road, 1 = car on road, 0 = car not on road). As a new car enters the road it is "switched on" meaning, that the algorithm is calculating its driving behavior. As it leaves the road, it is "switched off". This way there is not more calculation effort than needed. Cars, that are not on the road part of interest, are not simulated.

Data storage The most important quantities, that are saved in each time step and for each car, are:

Quantity	Explanation
Acceleration	$a(i, j)$ is the acceleration of car j in time step i.
Velocity	$v(i, j)$ is the velocity of car j in time step i.
Position	$x(i, j)$ is the position of car j in time step i.
Car order	The carresortmatrix is saved in each time step.
Switched on	The caronroad matrix is also saved in each time step.

All entries in any of these quantities are set to zero, if a car is not yet or not any more simulated. This means that the quantities $a(i, j)$, $v(i, j)$ and $x(i, j)$ have more or less a diagonal structure.

4.4 Car crash

If in any time step the distance of a driver to the car in front becomes negative this corresponds to a car crash. The algorithm aborts the calculation in this case.

4.5 Individual modification

The different model features can be set individually in the algorithm, such that their influence on the result can be tested easily. In particular this concerns the following features:

- Finite reaction time can be set to any value larger or equal to zero.
- Estimation error coefficients can be set to any value larger or equal to zero.
- Temporal anticipation can be set to 'on' or 'off'.
- Spatial anticipation can be set to any natural number larger or equal to one. This specifies the number of cars in front that are taken into account.
- Individual drivers can be set to 'on' or 'off'.
- Two lane switch can be set to 'on' or 'off'. In the latter case, just a single lane is simulated.

5 Simulation Results and Discussion

5.1 Plausibility of model

In order to collect some results, the TLM/IDCM code with all features switched on was slightly modified for better handling of the plot generation. Furthermore, for the simulation the desired velocity v_0 was connected to the initial velocity v_{ini} so that the cars had to accelerate in the beginning. The assumptions and criteria under which the model was built can be seen quite nicely in the resulting plots (cf. fig. 3 - 6).

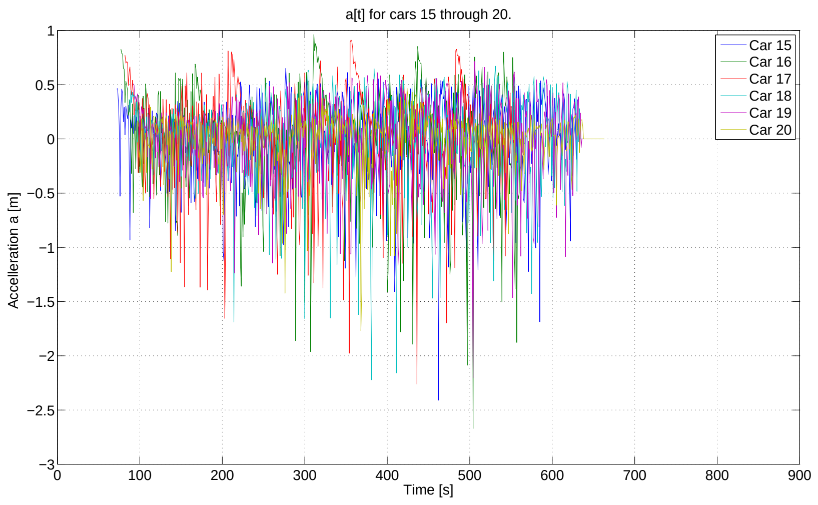

Figure 3: Acceleration $a(t)$ for 5 cars of different driver types.

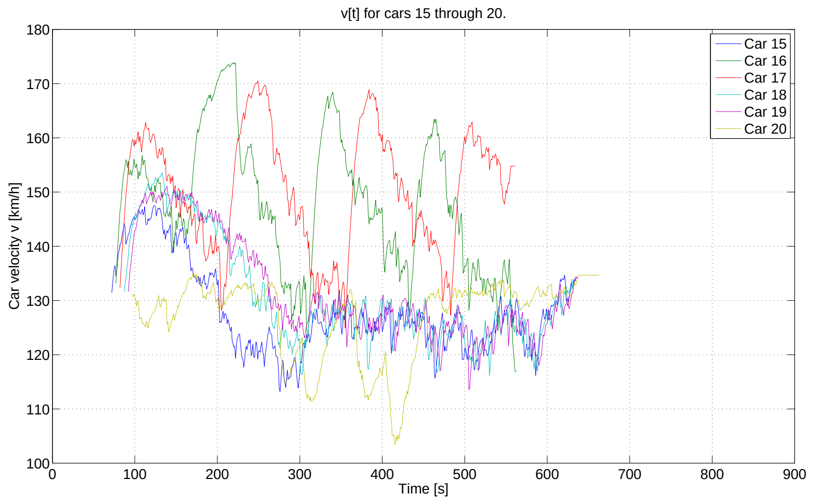

Figure 4: Velocity $v(t)$ for 5 cars of different driver types.

As we can see in figure 4, the individual driver types are of significant importance for the simulation. *Car 16* and *Car 17* are clearly more

aggressive drivers that wish to drive faster than the others. The spikes in velocity show a lane change; when the velocity drops again, the cars have fallen back into the right lane.
The lane change becomes even more evident when looking at figure 6: every discontinuity or sudden change in distance to the next car corresponds to a lane change. Again, we can see that, for example, *Car 15* and *Car 19* are of the slower type that does not overtake any other drivers since they are content with the speed on the right lane.
Figure 5 again shows the different driver types. Some cars finish the simulated distance in a significantly shorter time (*Car 16* and *Car 17*) than others. Also, in this figure we can see that *Car 20* is a car of the slower driver type. Note that in these figures the parameters have not yet been tweaked in order to fit the real data, thus the somewhat unrealistic values for x, s, v and a.

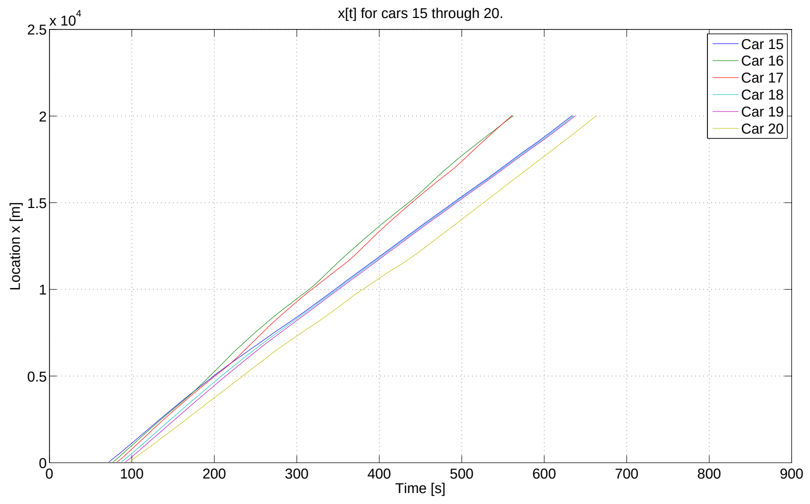

Figure 5: Position $x(t)$ for 5 cars of different driver types.

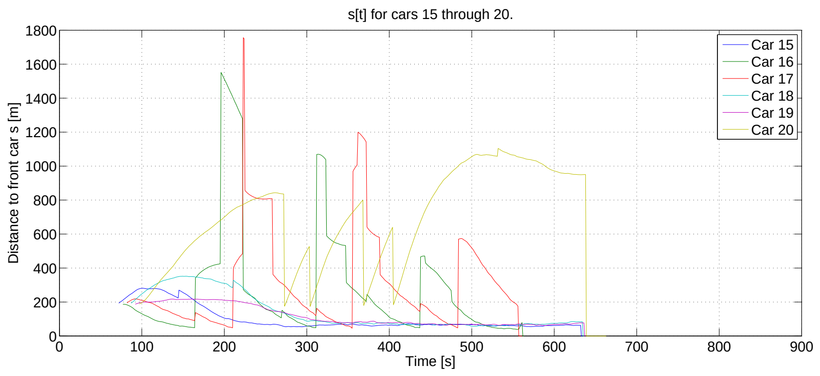

Figure 6: Distance to preceding car $s(t)$ for 5 cars of different driver types.

5.2 Measured real data

The data taken from *Deutsches Zentrum für Luft- und Raumfahrt* [2] were analyzed since the measured values were not clearly marked. The only values that were measured in every measurement were a, v, Δv and s, where Δv is the velocity difference to the front car. In contrast to our simulation, the data sets were recorded every 0.1s, which also had to be taken into account when calculating the traveled distance.
As it turned out, the distances were not the same for the different recorded data sets and the data sets had quite different lengths. Some of the sets were not useful since the driver appears to have been in a traffic jam ($v = 0$ for some times t); these data sets were discarded as our model did not aim to simulate behavior in a traffic jam. We decided to take the data set with the longest traveled distance which turned out to be measurement 1 with a traveled distance of about 5888m. This way, boundary effects in the simulation such as the initial velocity did not play a big role. The resulting plots are shown in figures 7 - 10.

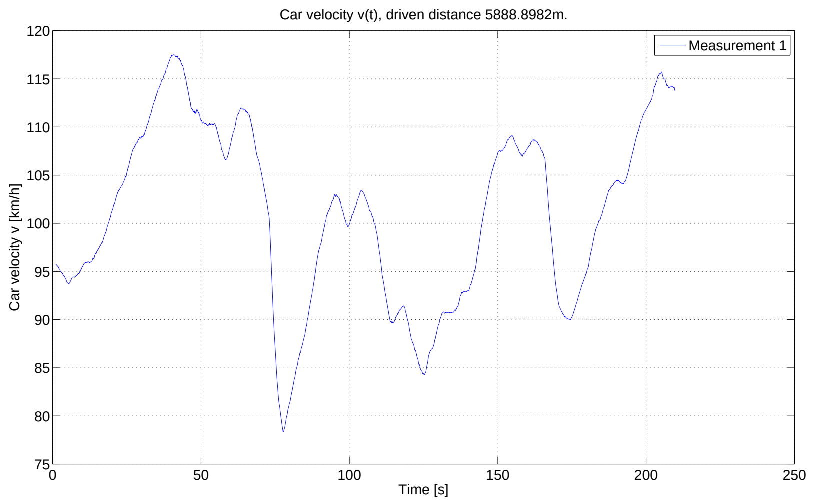

Figure 7: Velocity $v(t)$ of car of which the measurements were taken.

We can see from figure 7 that the measurement time is about 210s which, with a driven distance of 5888m, gives an average velocity of about 100km/h. This is a good value value for an average velocity since it is quite typical. Also, the average distance to the next car can be estimated from figure 8 to about 35-40m which is also fairly typical. There seem to be a couple of lane changes at the times where the distance has jumps. When comparing to the velocity (cf. fig. 7), one can identify whether these jumps in s stem from a lane change of the reference car or of the car in front.
From the above considerations we see that the situation captured in the first data set is 'typical' enough (i.e. no traffic jam, accident, etc.) to be describable by the simulation.

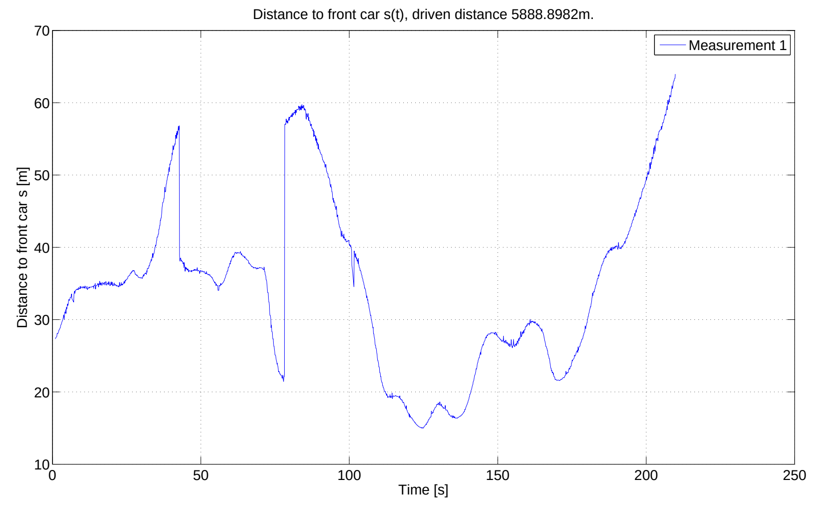

Figure 8: Distance to next car $s(t)$.

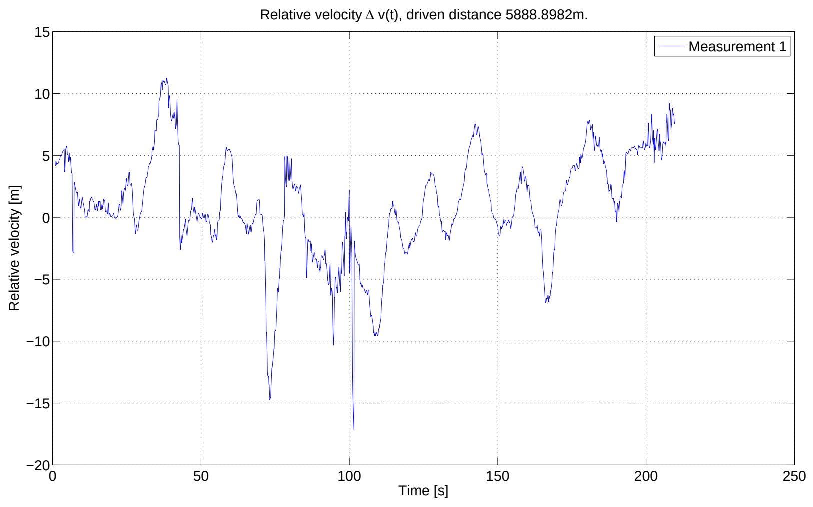

Figure 9: Velocity difference $\Delta v(t)$ of car of which the measurements were taken.

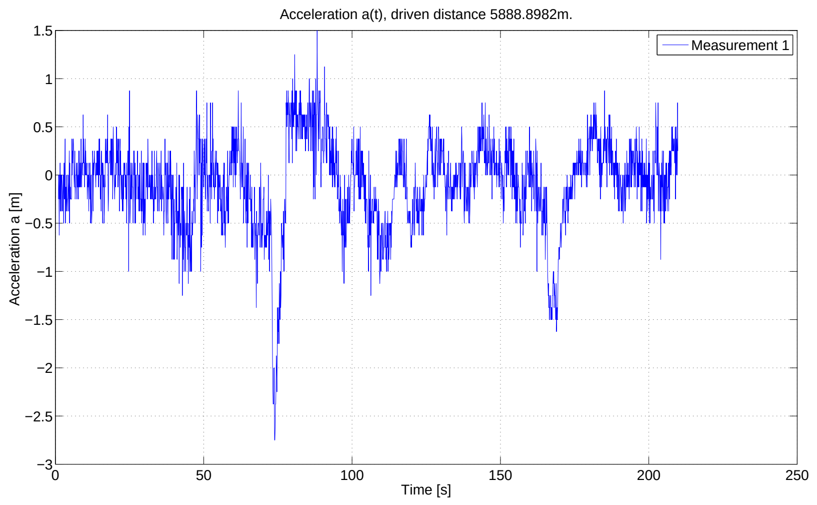

Figure 10: Acceleration $a(t)$ of car of which the measurements were taken.

5.3 Comparison of recorded data and simulation data

In order to do a comparison of the data one needed to adjust the simulation parameters to somehow fit the recorded data. Since the values for a and Δv fluctuated a lot for different simulations and in each simulation for every car, those values were not considered for the comparison. Nevertheless, the plots for these quantities are shown for each parameter set. The adjusted parameters were:

Desired velocity v_0 Describes the desired velocity of the average driver; this is the velocity a normal car would have without any other cars.

Car creation rate Q Describes the rate with which new cars enter simulated road part.

Reaction time $T_{reaction}$ Describes the time it takes for any driver to react to changes in velocity, distance, etc.

Maximum acceleration ac Corresponds to the largest acceleration possible for an average driver.

Initial velocity v_{ini} Gives the velocity of the cars when entering the simulated road part.

Headway $T_{headway}$ Describes the desired time until the car reaches the position of the next car assuming constant velocity.

Length of simulated road part L Is to be adjusted to the measured length of the road.

In the following, the influence of changing parameters is highlighted using data of one single car that nicely shows the general trend of the parameter change. Note that these cars were chosen to make the point clear and not necessarily describe the behavior of *all* cars!

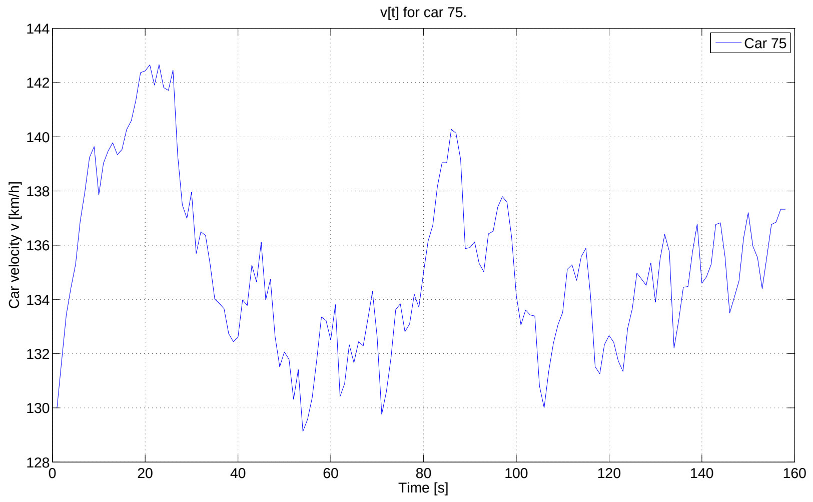

Figure 11: Velocity $v(t)$ of the simulated data for car number 75. Parameters: $v_0 = 160$, $Q = 0.2$, $T_{reaction} = 1.1$, $ac = 1$, $v_{ini} = 130$, $T_{headway} = 1.1$

Figures 11 - 13 show the starting point of the parameter fitting. The parameters that were used to generate these plots are listed in the captions.
We see that there is a qualitative agreement between the simulated and measured velocities (7 & 11). Taking a closer look it is noticeable that the velocity $v(t)$ is much smoother for the human driver. This might be due to an inherent 'speed adjustment inertia' of human drivers. This

reasoning makes sense because frequent adjustment of speed leads to a less smooth ride which is unpleasant for (human) car passengers and thus avoided by driver.

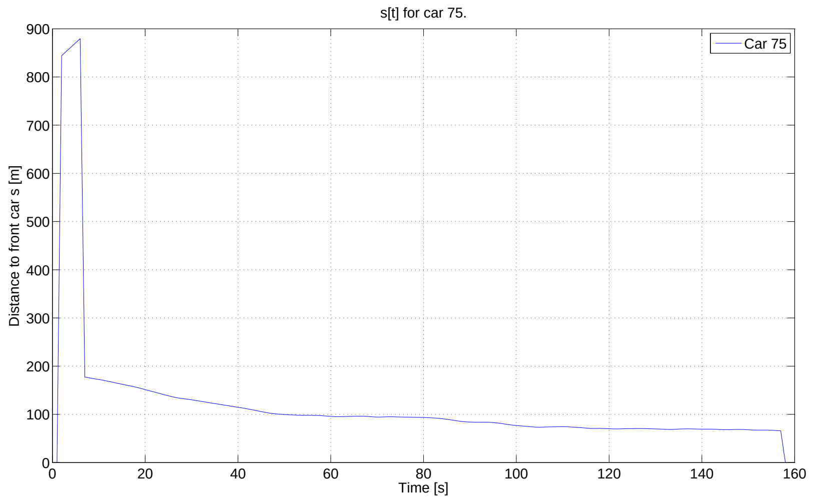

Figure 12: Distance $s(t)$ of the simulated data for car number 75 to the next car. Parameters: $v_0 = 160$, $Q = 0.2$, $T_{reaction} = 1.1$, $ac = 1$, $v_{ini} = 130$, $T_{headway} = 1.1$

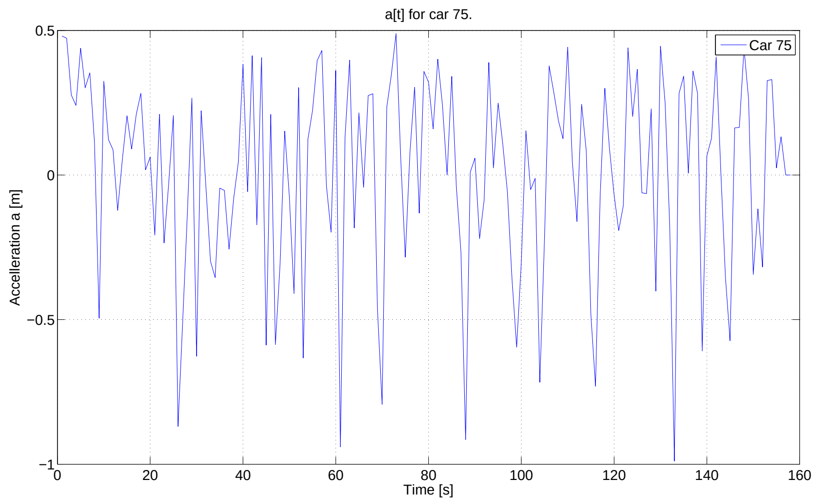

Figure 13: Acceleration $a(t)$ of the simulated data for car number 75. Parameters: $v_0 = 160$, $Q = 0.2$, $T_{reaction} = 1.1$, $ac = 1$, $v_{ini} = 130$, $T_{headway} = 1.1$

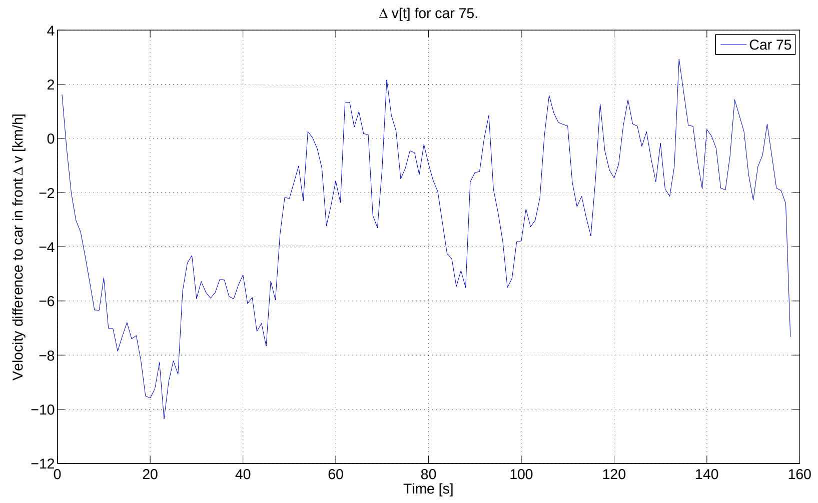

Figure 14: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 75. Parameters: $v_0 = 160$, $Q = 0.2$, $T_{reaction} = 1.1$, $ac = 1$, $v_{ini} = 130$, $T_{headway} = 1.1$

Analyzing the values for the velocity leads to an adjustment of the desired car speed, v_0. In order to have a average velocity of about 100km/h (like in the measurement), v_0 is set to 115. Also, seeing as the velocity rises in the beginning, it makes sense to link the initial velocity, v_{ini}, to v_0 such that $v_{ini} = v_0$.
Furthermore, in order to lower the distance between cars, we raise Q from 0.2 to 0.25 and, in order to prevent crashs, $T_{reaction}$ from 1.1 to 1. The resulting data is shown in figures 15 - 17.

For these modified parameters, we see an approvement in the velocity but the typical distance to the other cars is still not good enough. From the large distance to the next car, the frequent discontinuities (lane switch) (cf. fig. 16) and the rapid acceleration to a high desired velocity (cf. fig. 15) we can deduce that this must be an aggressive driver (remember that the aggressive type has higher ac and v_0 than the normal driver type).

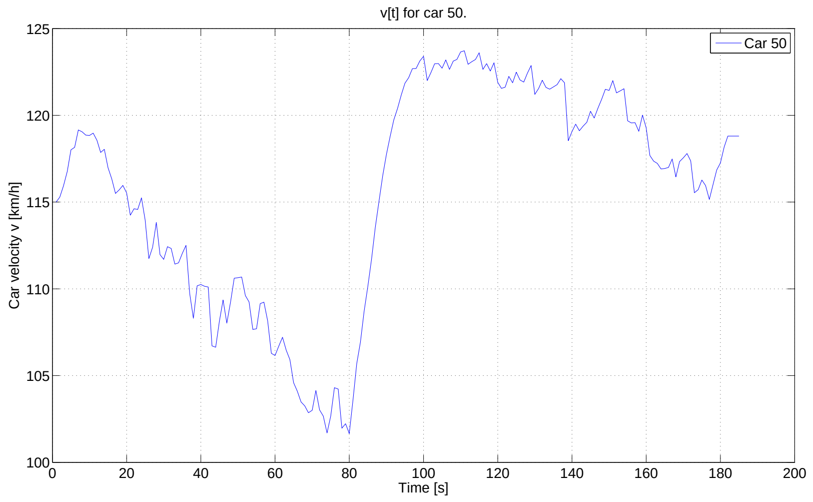

Figure 15: Velocity $v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1$, $v_{ini} = 115$, $T_{headway} = 1.1$

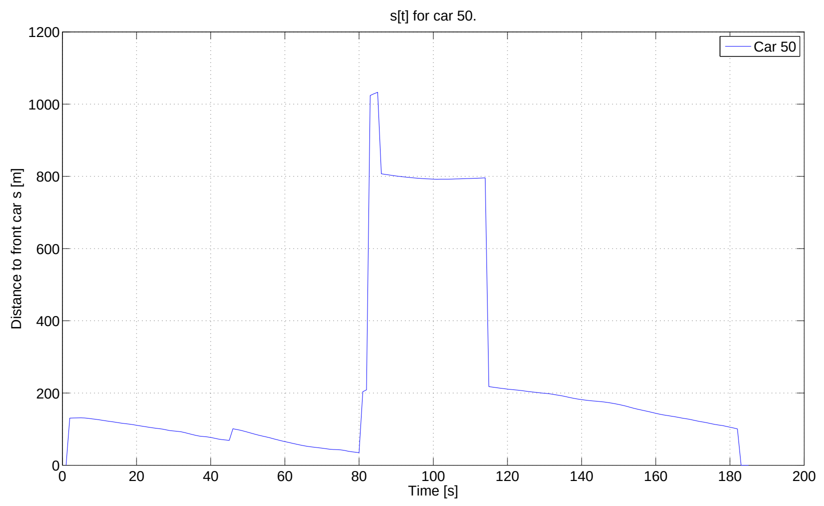

Figure 16: Distance $s(t)$ of the simulated data for car number 50 to the next car. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1$, $v_{ini} = 115$, $T_{headway} = 1.1$

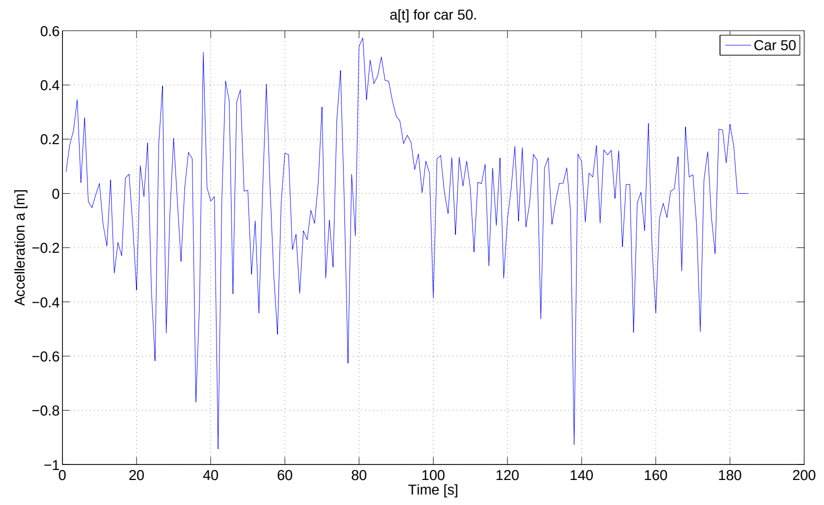

Figure 17: Acceleration $a(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1$, $v_{ini} = 105$, $T_{headway} = 1.1$

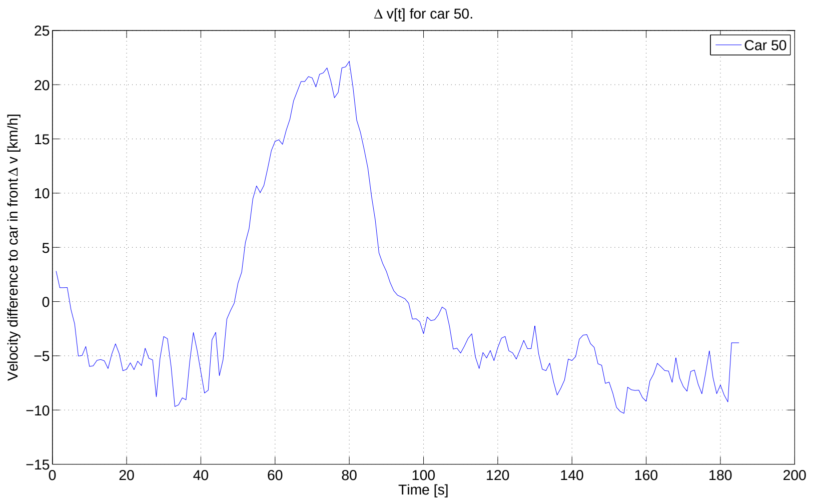

Figure 18: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1$, $v_{ini} = 115$, $T_{headway} = 1.1$

To further improve the results, we choose $v_{ini} = v_0 - 10 = 105$. Also, when comparing the accelerations with the recorded data, we notice that typical accelerations are higher in the real data (cf. fig. 10) than in the simulated data (cf. fig. 17) so that the maximum acceleration, ac, is set from 1 to 1.5.
The resulting data are shown in figs. 19 - 22.

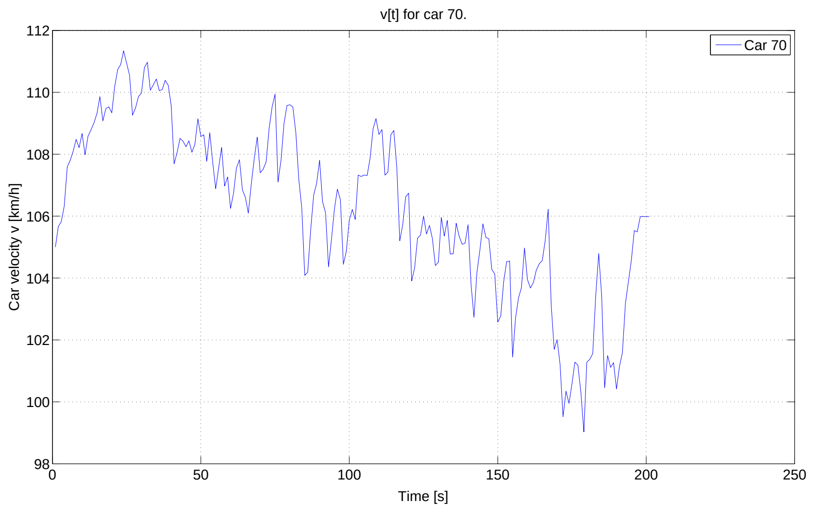

Figure 19: Velocity $v(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

We see that accelerations are now in a more reasonable region and closer to the measured values (cf. fig. 21). Looking at the results, we can see that the car's driver is of the less aggressive type since there are no lane changes and the velocity is always in the region of $v_0 = 115$. Still, the values for s (about 100) seem to be a little too high which is why the value for Q is changed to 0.3. This should make the road more crowded.
The results are shown in figs. 23 - 26.

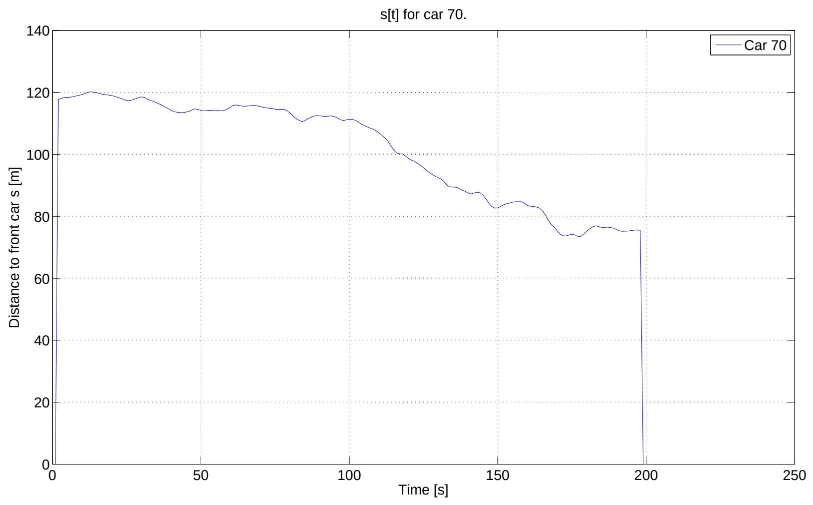

Figure 20: Distance $s(t)$ of the simulated data for car number 70 to the next car. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

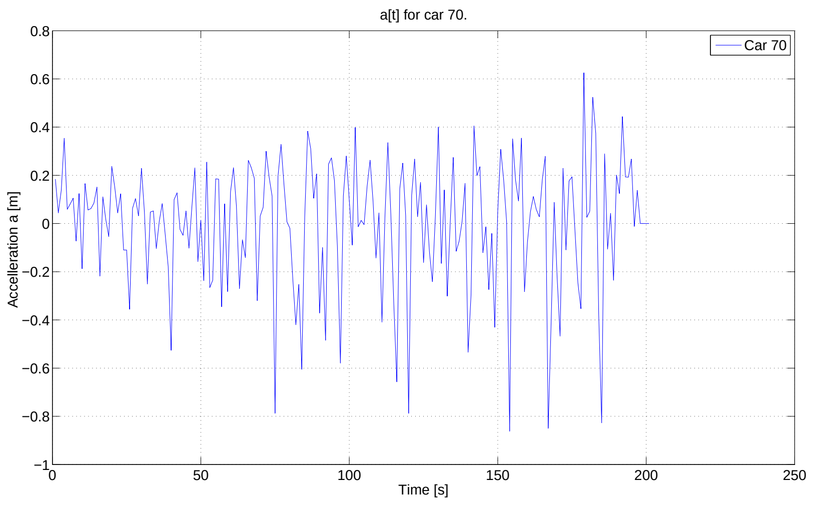

Figure 21: Acceleration $a(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

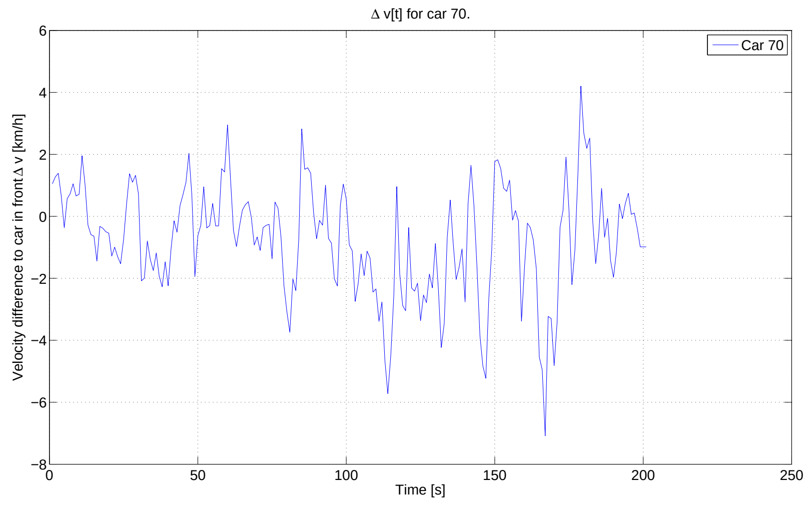

Figure 22: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.25$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

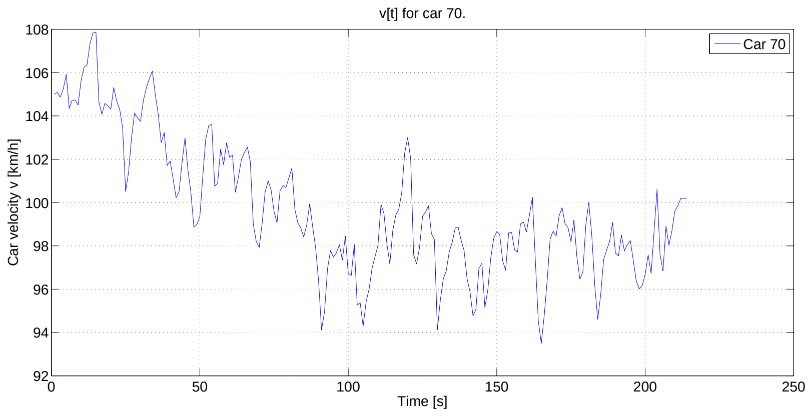

Figure 23: Velocity $v(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

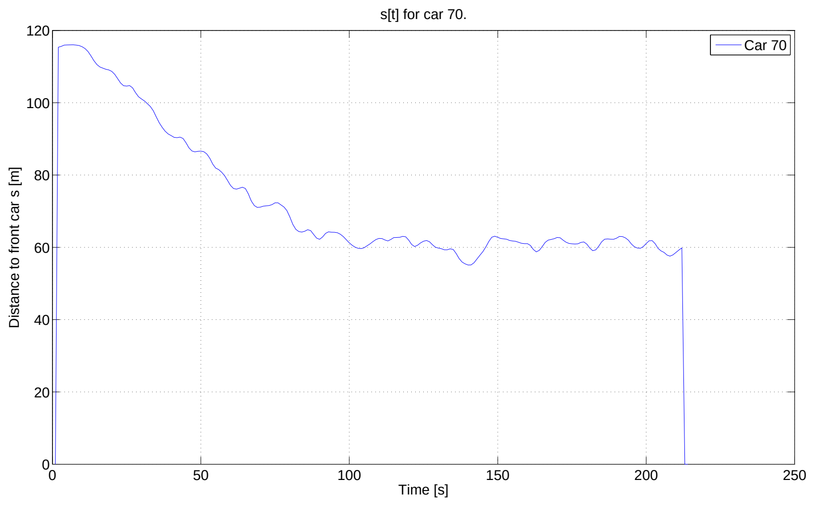

Figure 24: Distance $s(t)$ of the simulated data for car number 70 to the next car. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

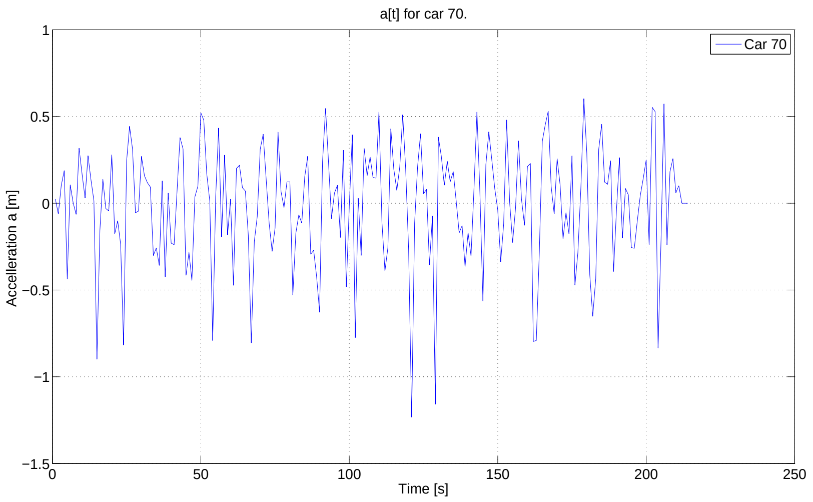

Figure 25: Acceleration $a(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

We see in the relative distance plot for the fourth parameter set (fig. 24) that s still is above the values from the measurement but the average velocity (cf. fig. 23) has gone down. We see that making the road more crowded in our model without lowering the desired headway just makes traffic more congested and the individual cars slower. This is not as one might expect in real traffic; the discrepancy has to do with the ability of humans to adapt the desired distance to the next car. For the next set, $T_{headway}$ has been modified from 1.1 to 0.8, thus allowing the cars to get closer to each other which should lower s.
The resulting plots are shown in figs. 27 - 30.

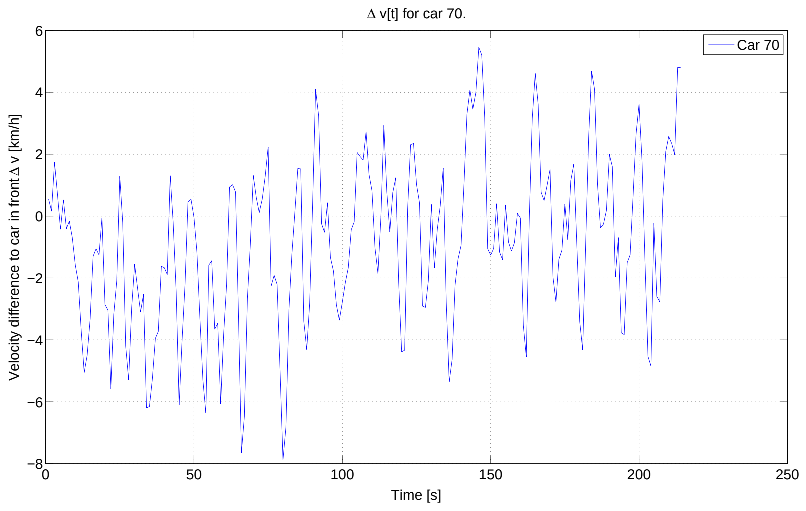

Figure 26: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 70. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 1.1$

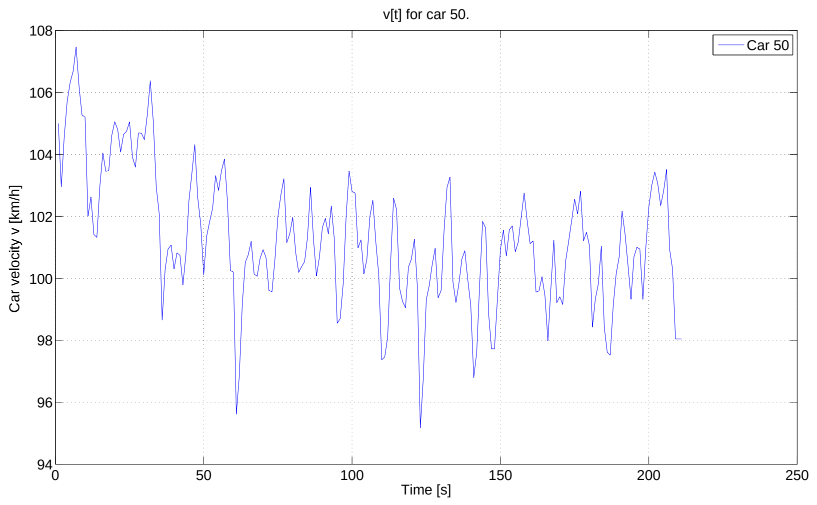

Figure 27: Velocity $v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$

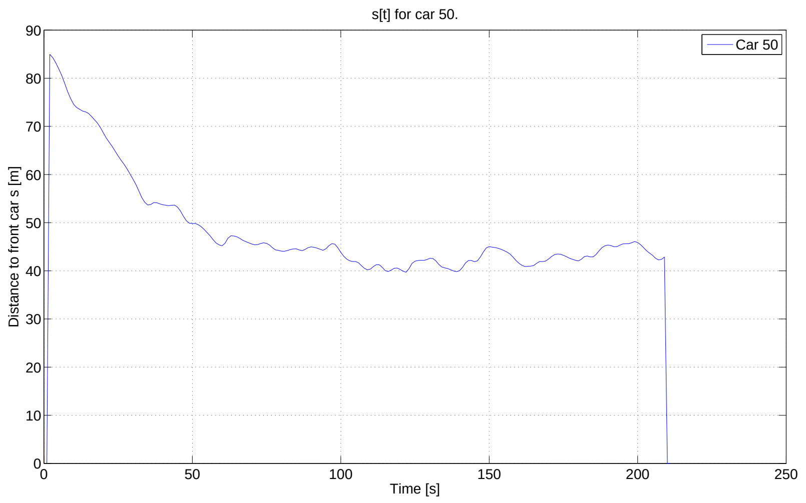

Figure 28: Distance $s(t)$ of the simulated data for car number 50 to the next car. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$

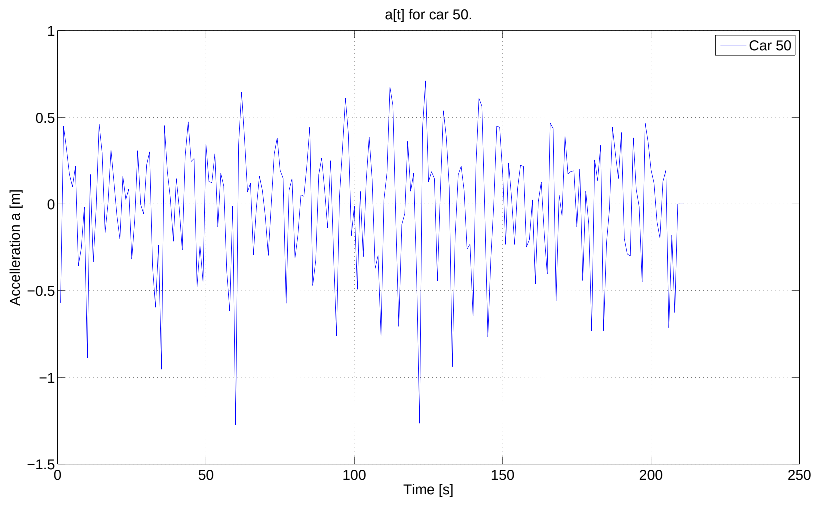

Figure 29: Acceleration $a(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$

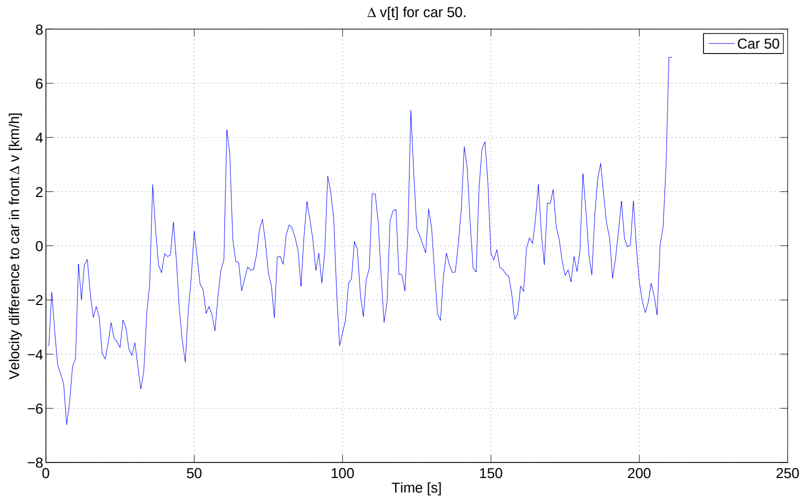

Figure 30: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$

For these parameters, the distance to the next car is, after a high initial distance, in about the right region (cf. fig. 28). Also, the velocity oscillates around 100km/h which is similar to the real data. These parameters seem to fit the real data quite well. However, when looking at the more aggressive drivers, we notice that the difference in desired velocity of about 10% seems to be a little small. To see what that would look like, we make the more aggressive drivers even more aggressive for the next set of plots, setting the desired velocity to a value that is 40% higher than the normal driver. The results are shown in figs. 31 - 34.

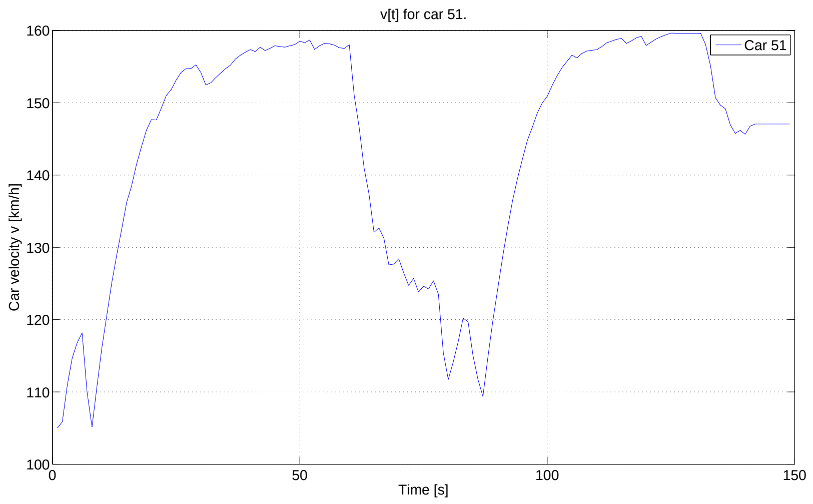

Figure 31: Velocity $v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$, faster aggressive drivers

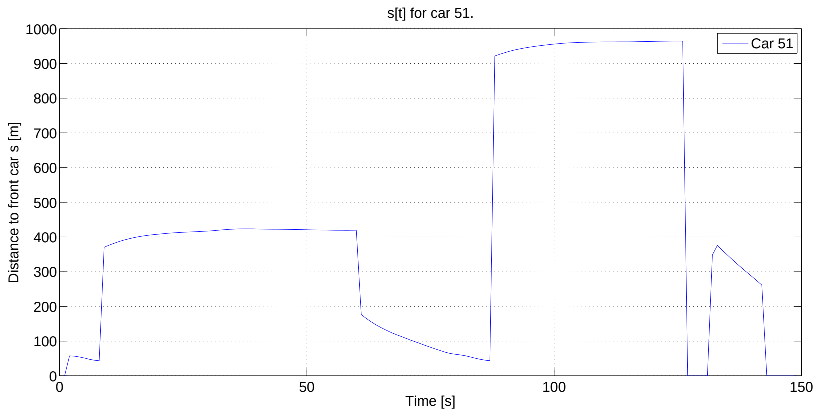

Figure 32: Distance $s(t)$ of the simulated data for car number 50 to the next car. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$, faster aggressive drivers

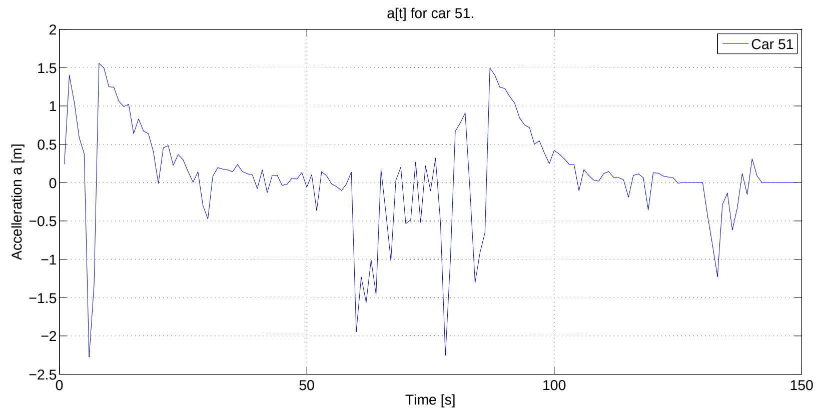

Figure 33: Acceleration $a(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$, faster aggressive drivers

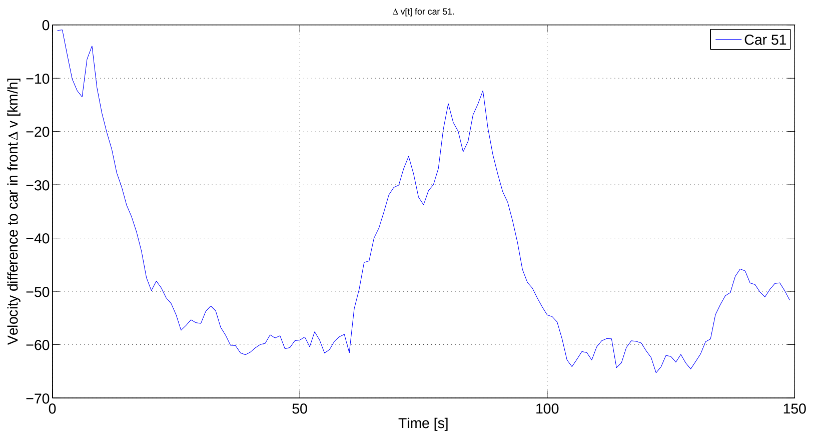

Figure 34: Velocity difference to car in front $\Delta v(t)$ of the simulated data for car number 50. Parameters: $v_0 = 115$, $Q = 0.3$, $T_{reaction} = 1$, $ac = 1.5$, $v_{ini} = 105$, $T_{headway} = 0.8$, faster aggressive drivers

If one takes a close look at s for the sixth parameter set (cf. fig. 32), the frequent jumps in s tell us that these simulated data are from an aggressive driver. Our model was designed so that a faster car changes to the right lane again whenever the there is space after overtaking the car in front. That is why there are not a lot of cars at once on the left lane. So whenever the distance to the next car is high, we know that the car is on the left lane. If we examine v (cf. fig. 31) more closely in those regions, we can see that the instead of a going straight to the desired speed, the faster car has rather high fluctuations in speed (even though the next car is very far away). This is due to the fact that the car driver looks for the next drivers and adjusts their speed accordingly. Obviously, this does not make a lot of sense realistically but it is in the scope of the assumptions and criteria that were made for the model.

6 Summary and Outlook

Traffic on a highway is a highly complex situation with many individuals that only have the goal of getting to their destiny in common. Thus, the scope of this project was to extend the HDM model to the IDCM and TLM and try to find parameters that fit a real data set well. This was accomplished to a certain extent. The word ‘well’ was used on purpose: without a more extensive analysis it is merely impossible to find the ‘best’ parameters.
A more extensive analysis would include a program that performs a large number of simulations for one parameter set, compares each simulation with the measured data and iterates over all reasonable parameter sets, leaving out those parameter combinations that cause too many crashes in the model. We aimed for this type of analysis but soon realized that it was too time-consuming and switched to our spot check type of analysis. The results are, considering the complexity of our model extension, surprisingly good. We found that an ‘intelligent’ variation of our parameters gave us good, albeit maybe not perfect, results.

Within our model, it might be interesting to find a correlation between parameters and the difference between average speed and desired speed. This way, general rules for moving faster in traffic not as an individual, but as a whole, might possibly be generated.
Furthermore, it might be interesting to find a way to smoothen out the velocity of the drivers, meaning that velocity inertia of human drivers should be taken into account. That way it is possible to generate smoother driving styles that resemble reality even more. One way could be to re-evaluate the acceleration in longer time intervals so that there are not so abrupt changes in velocity. Another could be to average over the neighboring accelerations; that way the drivers would not be prevented from braking to late and crashing into the front car. Yet another way to improve the model could be to only include the cars in front based on their distance to the reference car instead of just taking the next couple of cars. That way, cars that are, say, 1km ahead could not influence the reference car on less busy road parts.

To sum it up, there are a various more ideas to improve our TLM/IDCM to get even closer to simulating real traffic.

References

[1] http://www.thenational.ae/deployedfiles/Assets/Richmedia/Image/SaxoPress/AD20110924633701-Heavy. Visited on 06.12.2011.

[2] Deutsches Zentrum fur Luft-und Raumfahrt Clearingstelle Verkehr. Verkehrsdaten Autobahn A8 Deutschland [Stuttgart-Karlsruhe]. http://daten.clearingstelle-verkehr.de/191, 1995.

[3] Bundesministerium für Verkehr, Bau und Stadtentwicklung. Die Verkehrsentwicklung in Deutschland und deren Auswirkungen auf die Verkehrsinfrastruktur. http://www.ise.kit.edu/rd_download/SBT/Kolloquium_SBT_08-11_A._Gipper.pdf, 2008.

[4] D. Helbing M. Treiber, A. Kersting. Delays, inaccuracies and anticipation in microscopic traffic models. *Physica*, A 360:71–88, 2006.

A Presentation of Matlab code

In this section all of the matlab code, that was used for the model, can be found. This concerns the main simulation file as well as its subfunctions.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 1 Start section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%measure time
tic;

%Everything empty
clc;
close all;
clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%IDM Parameters of simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L=20000;                        %Length of simulated roadpart [L]=m
simtime=0.25*3600;              %Simulation time [s]
T_headway=1.1;                  %desired time until current position of front
                                %car would be reached assuming the actual speed
                                %[T_headway]=s. Corresponds to an average driver.
ac=1;                           %maximum acceleration of car / driver [a]=m.
                                %Corresponds to an average driver.
b=1.5;                          %Desired agreeable deceleration without danger,
                                %[b]=m/s^2. Corresponds to an average driver.
v0=160;                         %Desired maximum velocity [v0]=km/h.
                                %Corresponds to an average driver.
v_ini=130;                      %Initial velocity of cars entering the road,
                                %[v_ini]=km/h.
s0=2;                           %Desired distance to front car at rest [s0]=m
l=5;                            %Length of a vehicle [l]=m
Q=0.2;                          %Rate of cars entering the observed part of the
                                %road [Q]=1/s Note: The maximum capacity of the
                                %road under ideal conditions
                                %is Q_max=v0/(s_star+l)=0.763;
dt=1;                           %Iteration timestep [dt]=1
picturesequence=0.02;           %simulation demonstration with this picture
                                %sequency (in seconds)

%Estimation errors (Set both values to 0 if you want to neglect this effect)
deltav_coeff=0.015;             %Coefficient that specifies the imprecision of the
                                %velocity difference estimation [deltav_coeff]=1/s
s_coeff=0.015;                  %Coefficient that specifies the imprecission of
                                %the spatial difference estimation [s_coeff]=m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%HDM parameters of simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Finite reaction time (Set T_reaction to 0 if you want to neglect this effect)
T_reaction=0.8;                 %Reaction time [T_reaction]=s

%Temporal anticipation (Set this value to 0 (='off') if you want to
%neglect this effect)
temporal_anticipation_switch=1; %Can be set to 1 (='on') or 0 (='off')
                                %[temporal_anticipation_switch]=1
assert(temporal_anticipation_switch==1||temporal_anticipation_switch==0,...
'The variable "temporal_anticipation_switch" has not been assigned a valid value');
```

```
%Spatial anticipation (Set n=1 if you want to neglect this effect)
n=3;                            %Number of cars ahead used for spatial
                                %anticipation [n]=1

%Individual drivers (vehicle parameters in 3)
individual_driver_switch=1;     %Can be set to 1 (='on') or 0 (='off')
                                %[individual_driver_switch]=1
assert(individual_driver_switch==1||individual_driver_switch==0,...
'The variable "individual_driver_switch" has not been assigned a valid value');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Two lane parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
twolane=1;                      %Can be set to 1 (two lanes) or 0 (one lane)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Starting calculations and initialization of further variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cct=1/Q;                        %Car - creation time. Specifies the time
                                %interval between two cars entering the road
                                %according to Q
ionc=0;                         %index of newest (in terms of time) car that
                                %has been created on the road; start with 0
                                %cars on road
v0=v0/3.6;                      %Conversation km/h->m/s
timeidm=floor(T_reaction/dt);   %find timestep in which simulation can start
                                %(due to reaction time)
carnumbmax=ceil(1+simtime*Q);   %maximum number of cars during simulation
timenumb=ceil(simtime/dt);      %number of timesteps
cadtype=zeros(1,carnumbmax);    %car and driver type (numbers correspond
                                %to section 3)

%First row of carresortmatrix contains cars on rigth lane; second row contains
%cars on the left lane in the sequence in which they appear on the corresponding
%lane of the road at the moment t. Furthermore zeros indicate that a care on
%the other lane is between the own car and the next car on the own lane.
%Hence the matrix contains the relative order on the road.
%E.g. [3,1,0,0,2,0;0,0,6,4,0,5] would mean: car 3 is the first (right
%lane), then car 1 (right lane) then car 6 (left lane), then car 4 (left
%lane) ans so on...
carresortmatrix=zeros(2,carnumbmax);
caronroad=zeros(1,carnumbmax);           %label each car, if it is still on road:
                                         %2 = car is the first car on this line;
                                         %1 = car on road, but not the first,
                                         %0 = car not on road, resp. is "off"
x=zeros(timenumb,carnumbmax);            %space coordinate of each car
v=zeros(timenumb,carnumbmax)+v_ini/3.6;  %actual velocity
a=zeros(timenumb,carnumbmax);            %maximum acceleration
s=zeros(timenumb,carnumbmax);            %distance to car in front
beta=((1+timeidm)*dt-T_reaction)/...     %Technical coefficient for including
(((1+timeidm)*dt-T_reaction)+...         %T_reaction
(T_reaction-timeidm*dt));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% 2 Car and driver types section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%sleeping driver correspondig to 1
p_sleep=1/3;              %fraction of this driver- /cartype under all cars
T_headway_sleep=1.5*T_headway;
ac_sleep=0.7*ac;
b_sleep=0.7*b;
v0_sleep=0.9*v0;

%normal driver corresponding to 2
p_normal=1/3;             %fraction of this driver- /cartype under all cars
T_headway_normal=T_headway;
ac_normal=ac;
b_normal=b;
v0_normal=v0;

% agressive driver corresponding to 3
p_aggressive=1/3;         %fraction of this driver- /cartype under all cars
T_headway_aggressive=0.5*T_headway;
ac_aggressive=1.3*ac;
b_aggressive=1.3*b;
v0_agressive=1.1*v0;

%Define struct with car and driver parameters
switch individual_driver_switch
    case 0
        cadpara=struct('T_headway',{T_headway, T_headway, T_headway},...
            'ac',{ac, ac, ac},'b',{b, b, b},'v0',{v0, v0, v0});
        assert(p_sleep+p_normal+p_aggressive==1,...
            'Sum of probabilites for car types is no equal to 1');
    case 1
        cadpara=struct('T_headway',{T_headway_sleep, T_headway_normal,...
            T_headway_aggressive},'ac',{ac_sleep, ac_normal,...
            ac_aggressive},'b',{b_sleep, b_normal, b_aggressive},...
            'v0',{v0_sleep, v0_normal, v0_agressive});
        assert(p_sleep+p_normal+p_aggressive==1,...
            'Sum of probabilites for car types is no equal to 1');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 3 Two lane model section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for time=(timeidm+2):timenumb

    %% boundary conditions at the beginning of the road
    if(Q~=0&&cct>=1/Q)                    %is it time for a new car?
        ionc=ionc+1;                      %create new car on the right lane
                                          %of the road at the beginning of
                                          %the simulated road part
        carresortmatrix(:,ionc)=[ionc;0];
        caronroad(ionc)=1;
        %Determine type of car and driver entering roadpart according to
        %probabilities
        type=rand(1);
        if(type<=p_sleep)
            cadtype(ionc)=1;
        elseif(type>p_sleep&&type<=(p_sleep+p_normal))
            cadtype(ionc)=2;
        elseif(type>(p_sleep+p_normal)&&type<=(p_sleep+p_normal+p_aggressive))
            cadtype(ionc)=3;
        end

        cct=cct-1/Q;                      %set car creation time to zero
    end

        %Time increase
        cct=cct+dt;

      if(twolane)
      %% find new carresortmatrix that changes due to cars changing the lane
      %in this time step
      carresortmatrix=lanechangev2(carresortmatrix,caronroad,time,...
          cadpara,cadtype,x,v);
      end

      %% Set cars inactive that have left the simulated roadpart and determine
      %first cars on each lane
      caronroad=deactivate(x,carresortmatrix,caronroad,time,L);

    %% Specifing car behavior according to IDCM
    for auto=1:ionc

        if(caronroad(auto)==1)

            %Spatial anticipation and reaction time
            [r,c]=find(carresortmatrix==auto);
            gthcar=[];

            %Determine all cars up to n-th car in front of the actual car.
            %If there are less than n cars in front, use the ones that are
            %available
            for k=1:c-1 %Iterate over cars in front of actual car

                if(length(gthcar)<n&&carresortmatrix(r,c-k)~=0&&...
                        caronroad(carresortmatrix(r,c-k))~=0)
                    gthcar=[gthcar,carresortmatrix(r,c-k)];
                end

                if(carresortmatrix(r,c-k)~=0&&...
                        caronroad(carresortmatrix(r,c-k))==2)
                    break;
                end

            end

            %s_reaction contains the distances from the actual car to each car
            %in front. Same is true for the velocity differences
            s_reaction=zeros(1,length(gthcar));
            deltav_reaction=zeros(1,length(gthcar));

            for g=1:length(gthcar)
                s_reaction(g)=beta*x(time-1-timeidm,gthcar(g))+(1-beta)*...
                    x(time-2-timeidm,gthcar(g))-(beta*x(time-1-timeidm,auto)...
                    +(1-beta)*x(time-2-timeidm,auto));
                deltav_reaction(g)=beta*v(time-1-timeidm,auto)+(1-beta)*...
                    v(time-2-timeidm,auto)-(beta*v(time-1-timeidm,gthcar(g))...
                    +(1-beta)*v(time-2-timeidm,gthcar(g)));
            end

            a_reaction=beta*a(time-1-timeidm,auto)+(1-beta)*a(time-2-timeidm,auto);
            v_reaction=beta*v(time-1-timeidm,auto)+(1-beta)*v(time-2-timeidm,auto);

            %Estimation errors
            s_est=s_reaction+s_coeff....       %Distance estimation error
                *randn(1,length(gthcar));      %corresponds to standard normal
                                               %distribution
            a_est=a_reaction;                  %No information about accelleration
            deltav_est=deltav_reaction+...     %Velocity difference errors based
                deltav_coeff.*s_reaction.*...  %on standard normal distribution
                randn(1,length(gthcar));       %and proportional to the
                                               %corresponding vehicle ahead
            v_est=v_reaction;                  %No estimation errors here


            %Temporal anticipation
            switch temporal_anticipation_switch
                case 1
                    s_temp=s_est-T_reaction.*deltav_est;
                    v_temp=v_est+T_reaction*a_est;
                    deltav_temp=deltav_est;
                case 0
                    s_temp=s_est;
                    v_temp=v_est;
                    deltav_temp=deltav_est;
            end


            %Physical model for driving
            s_star=s0+v_temp*(cadpara(cadtype(auto)).T_headway)+...
                v_temp.*deltav_temp./(2*sqrt((cadpara(cadtype(auto)).ac)*...
                (cadpara(cadtype(auto)).b)));
            a_traffic=sum(-(cadpara(cadtype(auto)).ac)*(s_star./s_temp).^2);
            a_free=cadpara(cadtype(auto)).ac...
                *(1-(v_temp/(cadpara(cadtype(auto)).v0))^4);
            a(time-1,auto)=a_free+a_traffic;

            %Update quantities for this time step according to physical model
            v(time,auto)=v(time-1,auto)+dt*a(time-1,auto);
            x(time,auto)=x(time-1,auto)+v(time-1,auto)*dt+0.5*a(time-1,auto)*dt^2;

        elseif(caronroad(auto)==2)
```

```
            %Specify behavior of the first cars on each lane
            a_free=cadpara(cadtype(auto)).ac*(1-(v(time-1,auto)/...
                (cadpara(cadtype(auto)).v0))^4);
            a(time-1,auto)=a_free;
            v(time,auto)=v(time-1,auto)+dt*a(time-1,auto);
            x(time,auto)=x(time-1,auto)+v(time-1,auto)*dt+0.5*a(time-1,auto)*dt^2;

        end
    end


    %% Rebuild new carresortmatrix
    %After one timestep cars might have change order due to outdistancing
    if(twolane)
    carresortmatrix=changeorder(carresortmatrix,x,ionc,time);
    end

    %Save overall information about carorder
    totalcarressortmatrix{time}=carresortmatrix;
    totalcaronroad{time}=caronroad;


   %% Test if cars have crashed
   s_intermediate=carcrash(time,ionc,caronroad,carresortmatrix,x);
   s(time,1:ionc)=s_intermediate;

end

telapsed=toc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 4 Analyzing section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Make plotresults
results(a,v,x,s,12,22,false,simtime);

%Make video
videonew(x,totalcarressortmatrix,totalcaronroad,timeidm,timenumb,...
    picturesequence,twolane);
```

```
function carresortmatrix=lanechangev2(carresortmatrix,...
    time,cadpara,cadtype,x,v)
%% find actual carresortmatrix that changes due to changing the lane and
%outdistancing taking place in this time step

    %Physical model for changing to left lane
        %Motivation to outdistance given by criterion 1 and 2
        %1 criterion for switching to left lane: There is a driver in front
        %  very close
        %2 criterion for switching to left lane: Driver in front goes much slower

        %Criterion 3 to 4 state whether it is possible to change the lane
        %3 criterion for switching to left lane: Next car on left lane is
        %  farer or at least equal far away
        %  than next car on this (right) lane
        %4 criterion for switching to left lane: No car on left lane from
        %  behind is too close


    %Physical model for changing to right lane
       %Motivation to switch to right lane is given by traffic rule
       %to drive on right lane if possible. Criterion 1 to 2 state whether
       %it is possible to change the line
       %1 criterion for switching to right lane: On the right lane is
       %  enough space to front car in order to move in according to IDM
       %2 criterion for switching to right lane: On right lane is enough
       %  space to the rear car according to IDM


       %Create an intermediate matrix, such that lane change effects become
       %effective not before ALL cars made their decisions. The lane changes are
       %then updated in the end => Same boundary conditions for all cars.
       carresortmatrixintermediate=carresortmatrix;

       %Find out number of cars created so far => iteration boundary
       [~,cars]=find(carresortmatrix~=0);
       numberofcars=max(cars);


    for auto=1:numberofcars

        [r,c]=find(carresortmatrix==auto); %finds row and colum indices of
                                           %this car in the carresortmatrix

            switch r

            %test, if car should move to left lane in this time step, if it is
            %on right line
            case 1

                %1. criterion
                %Who is in front?
                r_front=r;

                if(c>1&&sum(carresortmatrix(r,1:c-1))~=0) %Someone in front?
            for k=1:numberofcars
                if(carresortmatrix(r,c-k)~=0)
                    c_front=c-k;
                    break;
                end
            end
                autofront=...      %found the next car in front
                    carresortmatrix(r_front,c_front);

            if((cadpara(cadtype... %This is the 1st criterion
                    (auto)).v0)>(x(time-1,autofront)-x(time-1,auto)))
                crit1=true;
            else
                crit1=false;
            end

            %2. criterion
            if(v(time-1,autofront)... %This is the 2nd criterion
                    <0.9*(cadpara(cadtype(auto)).v0))
                crit2=true;
            else
                crit2=false;
            end
        else
            crit1=false;
            crit2=false;
            autofront=[];
        end


        %3. criterion
        %Who is in front on left lane?
        r_frontleft=r+1;
        if(c>1&&sum(...            %Someone in front on the left lane?
                carresortmatrix(r_frontleft,1:c-1))~=0)
            for k=1:numberofcars
                if(carresortmatrix(r_frontleft,c-k)~=0)
                    c_frontleft=c-k;
                    break;
                end
            end

            autofrontleft=...      %Found next car in front on left lane
                carresortmatrix(r_frontleft,c_frontleft);

            if(~isempty(autofront))
                if(x(time-1,...    %This is the 3rd criterion
                        autofrontleft)>=x(time-1,autofront))
                    crit3=true;
                else
                    crit3=false;
                end
            else
                if(x(time-1,...    %Alternative criterion
                        autofrontleft)-x(time-1,auto)>4*...
                        (cadpara(cadtype(auto)).v0))
                    crit3=true;
                else
                    crit3=false;
                end
            end

        else
            crit3=true;
        end


        %4. criterion
        %Who is in the back on left lane?
        r_rearleft=r+1;
        if(sum...              %Someone in the rear on the left lane?
                (carresortmatrix(r_rearleft,c+1:end))~=0)
            for k=1:numberofcars
                if(carresortmatrix(r_rearleft,c+k)~=0)
                    c_rearleft=c+k;
                break;
                end
            end

            autorearleft=carresortmatrix(r_rearleft,c_rearleft);

            if(x(time-1,auto)-...        %This is the 5th criterion
                    x(time-1,autorearleft)>4*...
                    (cadpara(cadtype(autorearleft)).v0))
                crit4=true;
            else
                crit4=false;
            end
        else
        crit4=true;
        end

        %Car changes lane if all criteria are fulfilles
        if(crit1&&crit2&&crit3&&crit4)
           carresortmatrixintermediate(r,c)=0;
           carresortmatrixintermediate(r+1,c)=auto;
        end


    %test, if car should move to right line in this time step, if it is
    %on left line
    case 2

        %1. criterion
        r_frontright=r-1;
        if(c>1&&sum...         %Someone in front on the right lane?
                (carresortmatrix(r_frontright,1:c-1))~=0)
            for k=1:numberofcars
                if(carresortmatrix(r_frontright,c-k)~=0)
                    c_frontright=c-k;
```

```
                        break;
                    end
                end

                autofrontright=...%Found next car in front on the right lane
                    carresortmatrix(r_frontright,c_frontright);

                if((x(time-1,autofrontright)... %This is the 1st criterion
                        -x(time-1,auto))>4*cadpara(cadtype(auto)).v0)
                    crit1=true;
                else
                    crit1=false;
                end
            else
                crit1=true;
            end

            %2. criterion
            r_rearright=r-1;
            if(sum(...              %Someone in the back on the right lane?
                    carresortmatrix(r_rearright,c+1:end))~=0)
                for k=1:numberofcars
                    if(carresortmatrix(r_rearright,c+k)~=0)
                        c_rearright=c+k;
                        break;
                    end
                end

                autorearright=...   %found car in the rear on right lane
                    carresortmatrix(r_rearright,c_rearright);

                if(x(time-1,auto)-x(time-1,autorearright)>...
                        4*cadpara(cadtype(autorearright)).v0)
                    crit2=true;
                else
                    crit2=false;
                end
            else
                crit2=true;
            end

            %Car changes lane if all criteria are fulfilles
            if(crit1&&crit2)
               carresortmatrixintermediate(r,c)=0;
               carresortmatrixintermediate(r-1,c)=auto;
            end

        end
    end
    carresortmatrix=carresortmatrixintermediate;
end
```

```
%Function deactivates cars, if they have left the simulated roadpart (save
%computational ressources) This is done by setting the corresponding entry
%in caronroad to 0. Furthermore the first car on each lane that is still on
%the simulated roadpart is identified and labeld by setting the
%corresponding entry in the caronroad vector to 2 (instead of 1 as for all
%other cars)
function caronroad=deactivate(x,carresortmatrix,caronroad,time,L)

maxcolum=length(carresortmatrix(1,:));

newset=[true,true];
fc=[0,0];
           %rightlane
         for row=1:2
           for colum=maxcolum:-1:1
               if(carresortmatrix(row,colum)~=0)
                   if(x(time,carresortmatrix(row,colum))>=L)
                       caronroad(carresortmatrix(row,colum))=0;
                       newset(row)=false;
                   end
                   if(newset(row))
                   fc(row)=carresortmatrix(row,colum);
                   end
               end
           end
           if(fc(row)~=0)
           caronroad(fc(row))=2;
           end
         end
end
```

```
%% Rebuild new carresortmatrix
%After one timestep cars might have change order due to outdistancing
function carresortmatrix=changeorder(carresortmatrix,x,ionc,time)
xintermediate=[];
    for c=1:ionc
        if(carresortmatrix(1,c)~=0)
            xintermediate=[xintermediate,x(time,carresortmatrix(1,c))];
        elseif(carresortmatrix(2,c)~=0)
            xintermediate=[xintermediate,x(time,carresortmatrix(2,c))];
        end
    end

   [~,order]=sort(xintermediate,'descend');
   carresortmatrix(:,1:ionc)=carresortmatrix(:,order);
end
```

```
%Test if cars have crahsed and calculate distances
function s=carcrash(time,ionc,caronroad,carresortmatrix,x)
s=zeros(1,ionc);
for auto=1:ionc
        if(caronroad(auto)==1)
            [r,c]=find(carresortmatrix==auto);
            frontcar=[];

            for k=1:c-1 %Iterate over cars in front of actual car on the same lane
                if(isempty(frontcar)&&carresortmatrix(r,c-k)~=0&&...
                        caronroad(carresortmatrix(r,c-k))~=0)
                    frontcar=carresortmatrix(r,c-k); %Found car in front
                    break;
                end
            end

            %If order of cars on the same lane has changed in this time
            %step (car drove through another car) one of the two asserts
            %will yield an error, stating a car crash.
            assert(~isempty(frontcar),'Car crash!');
            s(auto)=x(time,frontcar)-x(time,auto); %determine real distances
            assert(s(auto)>=0,['Car crash! Car ',num2str(auto),' and car ',...
                num2str(frontcar),' have crashed.']);
        end
    end
end
```

```
function results(a,v,x,s,fstcar,lstcar,normalize,simtime)
%RESULTS generates the plotting results.
%a=acceleration values in m/s^2
%v=velocity values in m/s
%x=location values in m
%s=distance values in m
%carnumber=number of cars that should be plotted
%simtime=simulation time in s
%normalize: true <=> velocity is plotted such that every car starts at
%t=0 (normalized)

%Pre-calculations
firstcar=fstcar;
lastcar=lstcar;
plotlength=ceil(simtime);
vkmh=3.6*v;

if normalize==true
    vplt=zeros(length(vkmh(:,1)),length(vkmh(1,:)));
    splt=zeros(length(s(:,1)),length(s(1,:)));
    aplt=zeros(length(a(:,1)),length(a(1,:)));
    xplt=zeros(length(x(:,1)),length(x(1,:)));
    for jj=1:length(vkmh(1,:))                 %iteration over cars
        kk=1;
        for ii=1:length(vkmh(:,1))             %iteration over time
            if x(ii,jj)~=0 && x(ii-1,jj)~=0    %deletes all the zeros from the
                                               %columns, every car starts at
                                               %first column
                vplt(kk,jj)=vkmh(ii,jj);
                splt(kk,jj)=s(ii,jj);
                aplt(kk,jj)=a(ii,jj);
                xplt(kk,jj)=x(ii,jj);
                kk=kk+1;
            end
        end
    end
else
    vplt=vkmh;
    splt=s;
    for jj=1:length(vkmh(1,:))                %iteration over cars
        for ii=1:length(vkmh(:,1))            %iteration over time
            if  x(ii,jj)==0
                vplt(ii,jj)=NaN;              %replaces all zeros with NaN
                                              %so they don't appear in the plot
                splt(ii,jj)=NaN;
                aplt(ii,jj)=NaN;
                xplt(ii,jj)=NaN;
            else
                vplt(ii,jj)=vkmh(ii,jj);
                splt(ii,jj)=s(ii,jj);
                aplt(ii,jj)=a(ii,jj);
                xplt(ii,jj)=x(ii,jj);
            end
        end
    end
```

```
end

%create legend
for car=firstcar:lastcar
    label{car-firstcar+1}=['Car ',num2str(car)];
end

%velocity
figure;
plot(1:plotlength,vplt(1:plotlength,firstcar:lastcar));
title(['v[t] for cars ',int2str(firstcar),' through ',int2str(lastcar),'.']);
xlabel('Time [s]');
ylabel('Car velocity v [km/h]');
legend(label);
grid on;

%distance
figure;
plot(1:plotlength,splt(1:plotlength,firstcar:lastcar));
title(['s[t] for cars ',int2str(firstcar),' through ',int2str(lastcar),'.']);
xlabel('Time [s]');
ylabel('Distance to front car s [m]');
legend(label);
grid on;

%acceleration
figure;
plot(1:plotlength,aplt(1:plotlength,firstcar:lastcar));
title(['a[t] for cars ',int2str(firstcar),' through ',int2str(lastcar),'.']);
xlabel('Time [s]');
ylabel('Accelleration a [m]');
legend(label);
grid on;

%location
figure;
plot(1:plotlength,xplt(1:plotlength,firstcar:lastcar));
title(['x[t] for cars ',int2str(firstcar),' through ',int2str(lastcar),'.']);
xlabel('Time [s]');
ylabel('Location x [m]');
legend(label);
grid on;

end
```

```
% Functions plots the cars, as they move on the road
function videonew(x,totalcarressortmatrix,totalcaronroad,timeidm,...
    timenumb,picturesequence,twolane)

h=figure;
set(h,'Outerposition',[1,550,1900,250]);

for time=(timeidm+2):timenumb

  indices_of_cars_on_road=find(totalcaronroad{time}~=0);

  xvalues=zeros(1,length(indices_of_cars_on_road));
  yvalues=zeros(1,length(indices_of_cars_on_road));

for k=1:length(indices_of_cars_on_road)
    [row,colum]=find(totalcarressortmatrix{time}==indices_of_cars_on_road(k));
    if(row==1)
        yvalues(k)=-1;
    elseif(row==2)
        yvalues(k)=1;
    end

    if(~twolane)
        yvalues(k)=0;
    end

    xvalues(k)=x(time,indices_of_cars_on_road(k));
end

   plot(xvalues,yvalues,'r.');
   grid on;
   grid minor;

   if(twolane)
       title('Car behavior of HDM model on a two lane system','fontsize',14);
       ylim([-2,2]);
       ylabel('Right - left lane');
   else
       title('Car behavior of HDM model on a one lane system','fontsize',14);
       ylim([-1,1]);
       ylabel('Lane');
   end

   xlim([0,20000]);
   xlabel('Position [m]');
   pause(picturesequence);

end
```

List of abbreviations

HDM	human driver model
IDM	intelligent driver model
IDCM	individual driver and car model
TLM	two lane model

List of Figures