First Result of Simulating Traffic Behaviours Using Distributed Adaptive Control

Ming LU

Kay W. Axhausen

Junlong LI

IVT, ETH Zurich

April 2013

STRC | 13th Swiss Transport Research Conference

Monte Verità / Ascona, April 24 – 26, 2013

IVT, ETH Zurich

First Result of Simulating Traffic Behaviours Using Distributed Adaptive Control

Ming LU, Kay W. Axhausen
IVT
ETH Zurich
Wolfgang Pauli Str. 15,
phone: +41-44-633 38 01
fax: +41-44-633 10 57
{ming,lu}@ivt.baug.ethz.ch

Junlong LI
School of transportation
Southeast University, China

phone:
fax:

April 2013

Abstract

In this paper Distributed Adaptive Control (DAC) theory is used to simulate basic traffic behaviours such as car following, lane changing, overtaking, etc. A single link with a pre-set traffic density is simulated. In the system each car is a dependent unit to other cars, which applies an independent DAC algorithm. The car interacts with adjacent environment while driving on the road and adapts its behaviour according to front cars' behaviour to make decisions whether accelerating, decelerating or changing lanes. The DAC integrated in each car is actually an unsupervised learning algorithm modified from Hebbian learning. So these cars not only learn how to drive in the system, but also forget a small amount of their previous experiences to adapt themselves to the dynamic environment.

Compare to the other micro simulating softwares, the big advantage of this model is that each car is an intelligent unit that can learn and forget to adapt itself to the real situations instead of applying certain pre-defined rules. The simulating result is compared to a similar scenario in VISSIM as well as real data, and the result looks promising.

Keywords

traffic behaviour simulation, distributed adaptive control, lane changing, car following

1 Introduction

Traffic behaviour analysis is of great importance, in the sense of guiding traffic control, reducing traffic accidents, increasing level of service etc. The behaviours normally refer to free driving, car following, lane changing and overtaking etc. Many factors are influencing these traffic behaviours, which make the modelling very difficult. Normally all these factors that can be categorised into three parts: car properties, drivers' characteristics and road network status. For car properties, it involves car types, maximum and minimum accelerations, etc.; as for drivers, different people have different driving habits, safety distance tolerances and their attitudes towards risks are usually different. The driver enters a certain car and become one traffic unit, a certain number of these car-driver units interacting with each other and together formed the road network. The number of car-driver units, lanes of the road links, types of road nodes, traffic control units, etc. compose the road network. As there are so many dependent and independent factors combining together, the traffic behaviours are normally full of uncertainty and it is mostly difficult to investigate.

It is almost impossible to directly study the traffic behaviours due to so many factors involved during the process. What is more, it is normally difficult to get all the data needed to analyse traffic behaviours. So instead micro traffic simulation are used to analyse traffic behaviours under certain circumstances.

Traffic behaviour simulations try to simplify the traffic situations and discretize the continuous variables on the premise of closeness to reality. The advantage of simulation is that they can indicate different operations under different scenarios without changing facilities or collecting data in reality. But on the other hand, simulating can only be close to real situations, which means there might be minor errors between the simulation and actual road network operations.

There are many softwares to simulate driving behaviours, such as VISSIM, SSAM, etc. And most of them can be simulate the travel behaviours very well, there is also a common problem: the more close to the actually situation, the more parameters and therefore more complicated it needs for the software to run. For example, there are around 192 parameters(Ge and Menendez, 2013) in total needed to be calibrated in VISSIM(PTV, 2012).

Generally speaking, all these softwares are 'rule based' model to simulate the traffic behaviour, which means when certain thresholds are reached, it will trigger certain traffic behaviours changing. In this paper a simple yet efficient traffic behaviours model is proposed, based an unsupervised neural network algorithm. Instead of defining rules to indicate behaviour changing, the traffic units (cars with a certain type of drivers), interacting with other cars and the road system learn all the traffic behaviour themselves, such as driving as desired speed, car following, lane changing, etc. This way all the driving behaviours no longer needs to calibrate and when the road situation changes, each unit will adapt themselves to the actual environment.

The framework of this paper is described as follows:
In the next section, the current models simulating traffic behaviours will be discussed. Then the algorithm, which indicates all the travel behaviours, will be described. In this section, how the unsupervised neural network works as well as how to apply the theory to the traffic units are described. In the following section, a simple scenario is built using both VISSIM and the model proposed in the third section, and results are compared to see the differences. The last two sections of the paper give a short conclusion, disadvantages of the model and the further steps to improve the simulator.

2 Literature Review

There are a lot of researches focusing on driving behaviours. The driving behaviours normally mean car following, lane changing etc., and most of researches focus on car following model,q but up to a point lane changing behaviours also draws a lot of attention. The basic assumption for simulating the driving behaviours are: (1) drivers' decisions are to minimise their cost; (2) all drivers get perfect information around the environment; (3) the driving process is a continuous process for control theory; (4) there are noises in the system that makes the performance and the information of the drivers not ideally optimal. Based on this, driving behaviour models classified as: Gazis-Herman-Rothery (GHR) model(Gazis *et al.,* 1961), safety distance based model, action point model, artificial intelligent based model, etc.

The GHR model based on the theory that a vehicle can accelerate or decelerate by perceiving the stimulus between itself and the adjacent leading vehicles. In this model, the acceleration/deceleration of the car in the next step is calculated as:

$$\ddot{x}_{n+1}(t+\Delta t) = \frac{\alpha(l,m)(\dot{x}_{n+1}(t+\Delta t)^m)}{(x_n(t) - x_{n+1}(t))^l}[\dot{x}_n(t) - \dot{x}_{n+1}(t)] \tag{1}$$

where
$x_n(t), \dot{x}_n(t), \ddot{x}_n(t)$ are the position, speed and acceleration/deceleration of vehicle n at time t;
Δt is the perception reaction time as well as the time step;
$l, m, \alpha(l,m)$ are special parameters, normally needed to be specified.

The first part of the function on the right hand side is the sensitivity and the second part is the stimulus. This model, developed in the early 1950s, is a deterministic stimulus-response model. Therefore it cannot explain many of the situations that the drivers encounter in reality(Olstam and Tapani, 2004). The hypothesis used to avoid collision is reasonable, but there are many factors that influence the driving behaviours that the model neglect, such as 'retaining wall' brake(Wang *et al.,* 2006) is only a rare instance and the safety distance is only a theory value which tend to be the larger the better but in reality drivers do not keep the safety distance in most cases. Therefore, the traffic capacity calculated by the safety distance model will be normally smaller than the real maximum traffic volume.

The action point model(Cynthia and McGwin, 1999) is actually based on psycho-physics(Wiedemann, 1991) driving factors. This model is built under the hypothesis that drivers can get the relative speed of the front vehicle, perceiving the changes on the visual angle subtended by the vehicle ahead.

The safety distance model, on the other hand, based on the following two assumptions: vehicle considers emergency braking from its front car to prevent collisions; and vehicles are spaced out according to a certain spacing function behind the leader car in the simulation. For more details, please refer to Menendez (2006), Li and Wang (2003), etc.

Beside all the traditional models, as the artificial intelligent algorithm becomes more and more popular, the theory also extents to simulate the traffic behaviours. Fuzzy logic, a widely used theory in machine control field, is applied to driving behaviour simulation as well. In reality, the change of the driving behaviours is not triggered by only one factor, rather several factors combing together(Teodorović, 1999, Wu *et al.,* 1999). The rules of behaviour changing are usually not rigorous, and fuzzy logic theory is proposed to solve this kind of problems. As neural network functions similarly as fuzzy logic theory, there are also models based on it (Same and Postorino, 1994, Shen, 2006).

There are also many types of software developed for simulating micro traffic behaviours. For example,Yang and Koutsopoulos (1996) developed a simulator called MITSIM for modelling traffic networks with advanced traffic controls, route guidance and surveillance systems.Kosonen (2003) also developed a simulator called HUTSIM exclusively for urban traffic simulation and control model. Another simulator named Paramics(Cameron and Duncan, 1996) intends to solve the congestion problems and fully uses computer hardware configuration, as parallel computing technology is introduced to the model. There are also many similar simulation softwares, for example TRANSIMS (Smith *et al.,* 1995),VanetMobiSim(Haerri *et al.,* 2006). And last but not least, VISSIM from PTV is by far the most well-developed commercialised micro traffic simulation software. All these models make their efforts to recur the traffic situations base on certain driving behaviour models and most of them function quite well.

3 Model framework

In this section, basic theory will be discussed first. Distributed adaptive control (DAC), an artificial intelligent algorithm, which applies unsupervised learning process, or more precisely, Hebbian learning, is introduced. Then DAC is integrated into the traffic unit (car-driver). These unit use 'sensors' to detect the proximity of adjacent cars and whether they crash into another or not; considering all these inputs, the agent uses Hebbian learning to gradually learn its driving behaviours.

3.1 Distributed adaptive control

The basic theory behind the distributed adaptive control is Hebbian learning, so before the discussion of DAC, it is necessary to briefly introduce Hebbian learning algorithm. Hebbian learning is one of the unsupervised learning algorithms, which means its weights update themselves without the error propagation corrections, and it originally comes from the dynamics of biological systems. Unlike the other unsupervised learning algorithms, such as Kohonen maps, which has the competitive character of feature mapping, Hebbian learning is more like a one plain competitive learning. The synapse between two neurons will be strengthened when the neurons on either side of the synapse (input and output) have highly correlated outputs. In essence, when an input neuron fires, if it frequently leads to the firing of the output neuron, the synapse is strengthened. Following the analogy to an artificial system, the tap weight is increased with high correlation between two sequential neurons. This learning process can be described as following:

$$V = \sum w_{ij}\xi_j \tag{2}$$

where V here is the output control measures;
w_{ij} is the weight, or synapse; ξ_j is the output (not the control measure). The assumption here is that the outputs take the linear form of the inputs. Long termly speaking, it can be regarded that the inputs are drawn from a probability distribution $P(\xi)$. For every step the input is applied to the network, with Hebbian learning, the agent learns, over time, how 'typical' for a certain input ξ_j is in the distribution, the higher the probability is, the larger the output V on average. With a certain learning rate , the weights are updated as:

$$w'_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} + \eta V \xi_j \tag{3}$$

This would be the end of the story if the weights stop learning at certain points otherwise the weights will grow instantly and become computationally large. As the agent is surrounded by a dynamic environment, it is required to continue learning to adapt to the real time event. Hence the weight update strategy must be ensured to be normalised so as to prevent the weight keeping increasing. This can be done by applying Oja's rule:

$$\Delta w_{ij} = \eta V(\xi_j - V w_j) \tag{4}$$

It can be proved that using this rule, the weights converge to constant length ($|w| = 1$). For more details, please refer to Verschure *et al.* (1992)*,* Verschure (1998). It can also be interoperated as follows: if the agent continues to learn, up to a point, the memory capacity will be full so it has to forget something in order to take in new learning experiences.

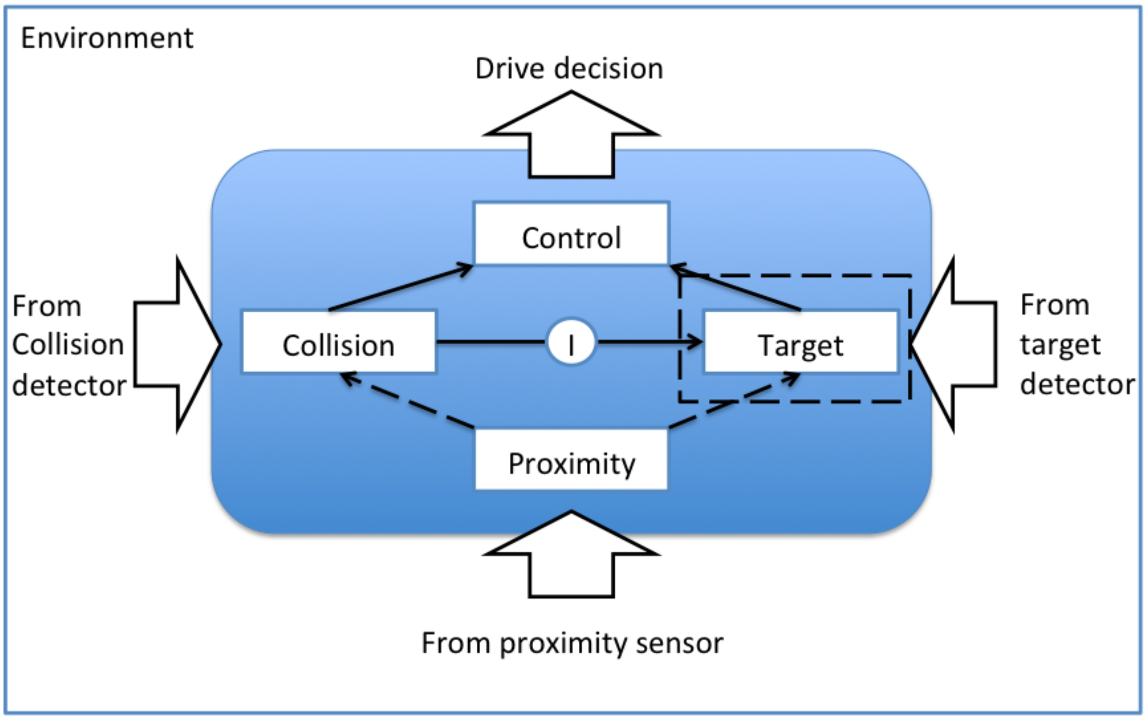
The weight updating strategy shows that Hebbian learning updates its weights only locally instead of taking the overall system input-output properties into consideration. This can be an advantage as well as a disadvantage, and for many control systems, this is the only option as the overall system performances is only possible after the operation finishes.

3.2 Model application

As shown in Fig. 1 (Verschure *et al.,* 1992), there are four blocks in each traffic unit (car plus driver, for simplification reasons, it is the same as cars mentioned later): the proximity block, the collision block, the target block and the control block. There are three inputs and one output. The proximity block feeds its measurement to the collision block and the target block; the collision block detects the results of the control block and feedback to the control block and the target block. The target block receives the inputs from proximity block and collision block, processes these information and then pass it to the control block. The control block deals with all the information from the collision layer and the target layer, then decides which control actions to take.

In the sense of a basic neural network structure, the proximity layer (block) is the input layer, while the control layer is the output layer, which indicates the driving behaviour of the traffic unit. The collision layer and the target layer is a little different than the middle layer of a backward propagation neural network: not only it propagate the inputs, they also receive inputs from another source of inputs (sensors), which in this specific model, indicates if the car crashes into the other cars, or the vertical distance between the car and adjacent front car is smaller than the predefined safety distance, or how far away it is from the its destination, etc. Also for the proximity and the collision layer they are only feed-forward block, which means they only

Figure 1: Model structure of distributed adaptive control



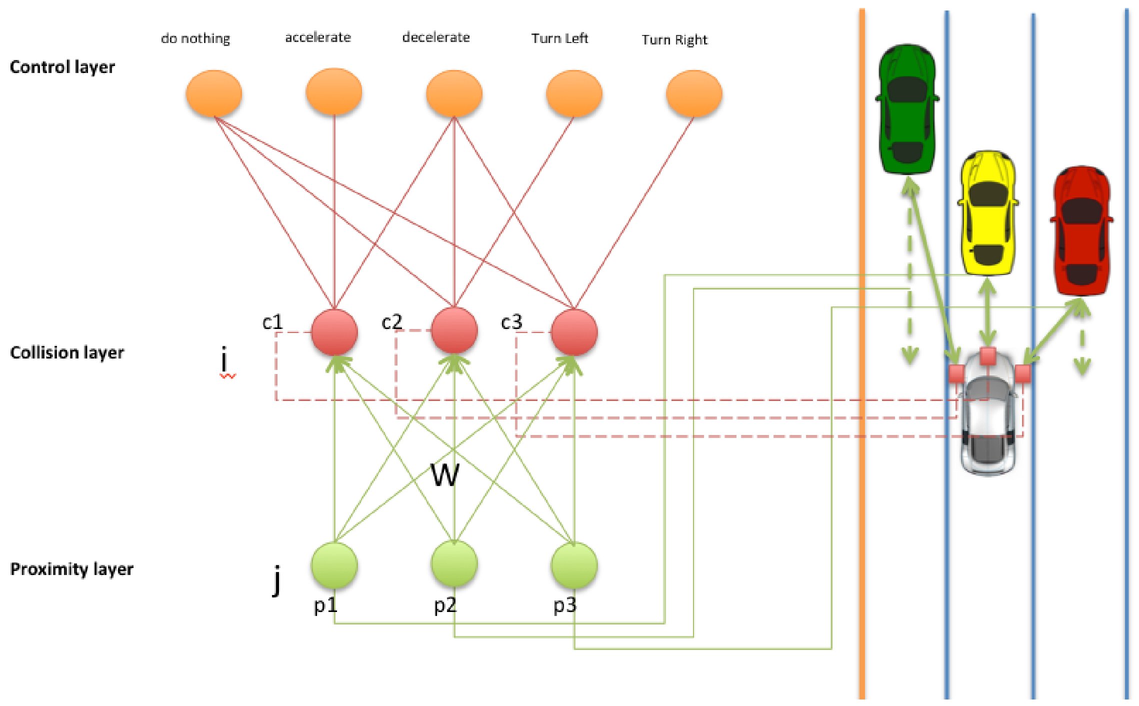
detect the adjacent distances and collision status then pass them to the next blocks, while the target layer and the control layer need to process all the inputs and then decide which control measures to take. These two layers apply Hebbian learning, over time, the weights will pick up the patterns (or correlations) of the collision layer and the control measures. And those with strong correlations will be strengthening by assigning higher weights and irrelevant events will be ignored.

3.3 Model specification

3.3.1 Layer definition

The model proposed here is only a prototype. It only has vertical movement, and for the lateral movement, which concerns lane-changing behaviour, the car will instantaneously change to the desired lane with only one time step. And for the matter of simplification, the cars can only detect the proximity and the collisions with front cars, both in the same lane and in the adjacent lanes. The network structure for a single car is shown in Fig. 2.

Figure 2: Model layer of distributed adaptive control



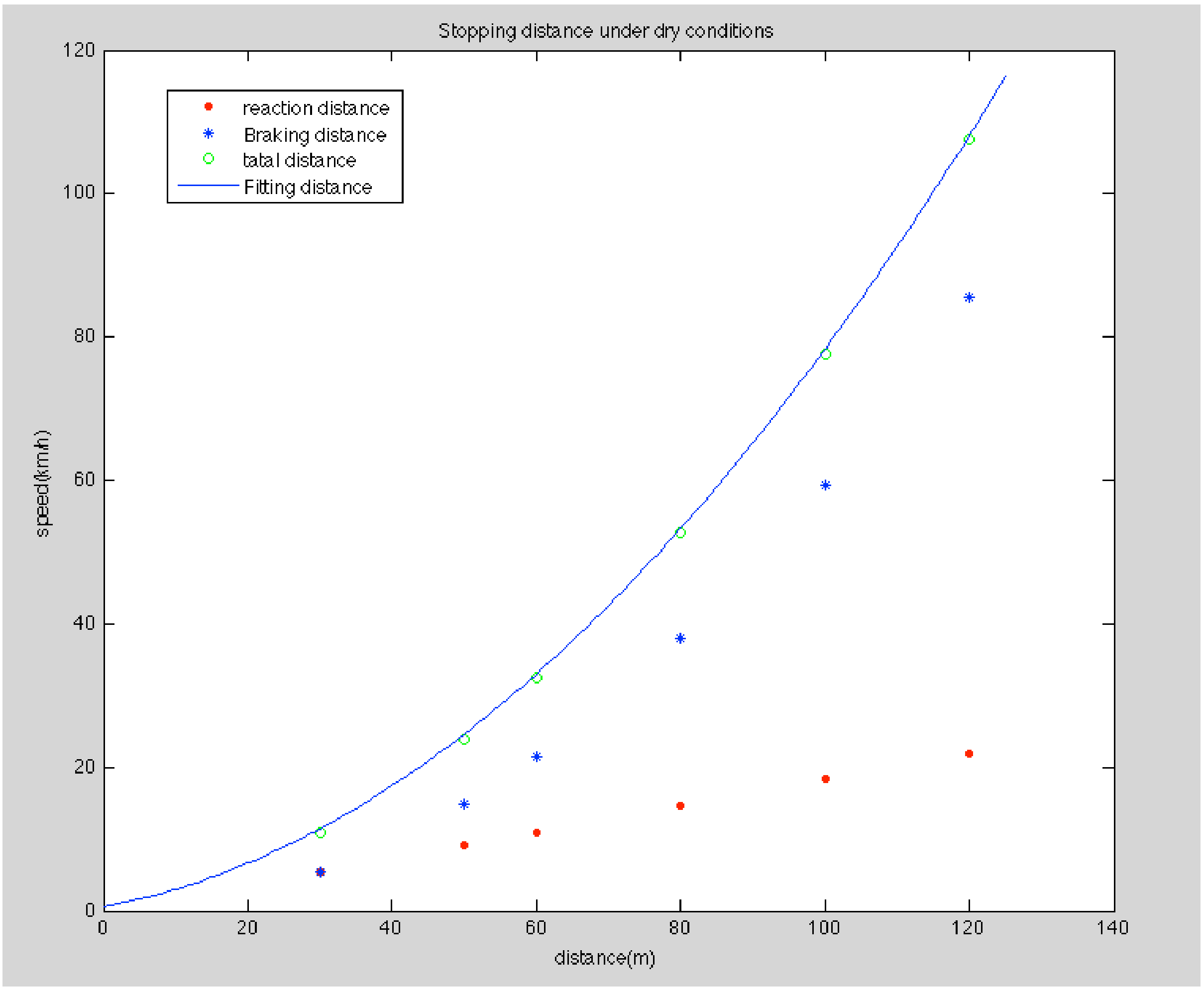
For the proximity layer, there are only three inputs that indicate the vertical distances from the front car in the same lane, the front car in the left lane if it is not currently in the most inner lane, the front car in the right lane if itself is not currently in the most outer lane. All the three distances is measured as the front of the current car to the rear of the front cars, and it is not point-to-point distances as the solid double arrows, rather the vertical distances shown as dashed double arrows in the figure. The input of the collision layer has also three inputs, respectively refers to the collision status to the front cars in all lanes. These inputs are binary values, which 1 means the current car crash into the front the car, there is no left or right lane and 0 means no crashes. For safety issues, crashing here does not mean the car literally crash into another car. Once the distance from the front car is smaller than the safety distance, the crashing mechanism will be activated and the input of the collision sensor will become 1. And the safety distance applied here is a function of its own speed, shown in Eq. (5):

$$Dis_{safety} = \alpha v^2 + \beta \cdot v + c \tag{5}$$

where c is a constant and α, β are the parameters of speed(km/h). Here we using th safety distance from British Transport Research Laboratory [1] and conducted a quadratic polynomial fit

[1]See `http://www.rulesoftheroad.ie/`, accessed on April 10th, 2013.

Figure 3: Safety distance



and the final safety distance is calculated in Eq. (6):

$$Dis_{safety} = -0.0059v^2 + 0.1872 \cdot v + 0.5000 \tag{6}$$

The control layer maps to five behaviours: do nothing, accelerate, decelerate, turn left and turn right. Do nothing means stay as the last state, this happens when the car is driving as desired speed and there is no disturbance around. Accelerate happens when current speed is lower than the desire speed and there is distance from front car is larger than the collision distances. Decelerate happens when the car have no other choice but to follow the front the car and the current speed is higher than the front car. Turn left is when the front car drives slower than the desire speed while distance from the front car in the left lane is larger than the safety distance. And turn right is similar as turn left, except the car looks for gaps on the right lane. If both adjacent lanes are available, then the car will preferentially choose left lane.

The target layer is not illustrated in Fig. 2 as it differs with the road network structures or special purposes. For example, if each car has a destination, which indicates the car's off-ramp position, so it could enter force lane changing mode in a certain distance ahead of the off-ramp. By default, the target is defined as driving the car as fast as possible without crashing into front cars.

3.3.2 Weight updating

The input for the control layer is prepared as in Eq. (7):

$$h_i = c_i + \sum_{j=1}^{n} w_{ij} \cdot p_j \tag{7}$$

where h_i is the intermediate input of the control layer;
c_i is the input of the collision layer, 1 if the distance is smaller than the safety distance and 0 otherwise;
wij is the weight from proximity node i to collision node j;
p_j is the proximity input j.
Then this value will be transferred by a threshold function (Eq. (8)):

$$a_i = \begin{cases} 0 & : h_i < \Theta \\ 1 & : h_i \geq \Theta \end{cases} \tag{8}$$

where a_i is the output of the collision layer and it is set to one if the intermediate output is higher than or equal to the threshold Θ.

The next step is mapping all the binary values to the actual controls of the car. Table 1 indicates all the mapping of the driving behaviour.

And finally, the weights update strategy is as in Eq. (9):

$$\Delta w_{ij} = \frac{1}{n}(\eta \cdot a_i \cdot p_j - \varepsilon \cdot \overline{a} \cdot w_{ij}) \tag{9}$$

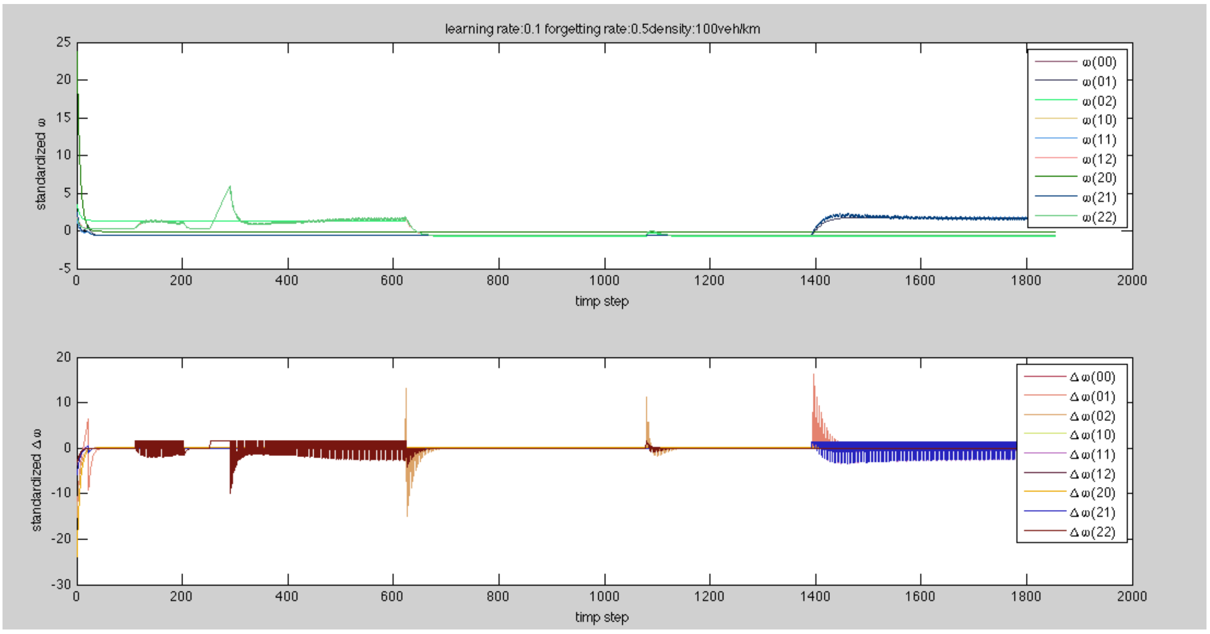
where $\Delta w_{ij})$ is the change of the weight;
n is the number of the proximity nodes;
η is the learning rate;
ϵ is the forgetting rate;

Table 1: Mappings of control and driving behaviour

Driving behaviours	Condition	Outputs $(a_1, a_2, a_3)^*$
Do nothing	$a_1 = 0, v = v_{desire}$.	000,010,011,001
Accelerate	$a_1 = 0, v < v_{desire}$.	000,010,011,001
Decelerate	$a_1 = 1, a_2 = 1, a_3 = 1$.	111
Turn left	$a_1 = 1, a_2 = 0$.	100,101
Turn right	$a_1 = 1, a_2 = 1, a_3 = 0$.	110

Source: The subscript indicates the input and output positions, 1 means the front car node, 2 means the left car node, 3 means right car node.

Figure 4: Weights updates



$\bar{a}$ is the average of the collision layer outputs.

As discussed before, based on the special structure of the Hebbian learning, the traditional online learning and offline learning strategy cannot apply here. But if the cars have the same amount of proximity nodes, collision nodes and control behaviours, they can share the weights. And for each time step, all these cars can update the weights according to their own situations. This way the weights will converge much faster than the independent ones, but if the target and the nodes are not the same, for some cars, they might behave very strange due to the common weights updates.

From Fig. 4, all weights are shared by the cars as they have the same structure. It can be seen that for the first few time steps, both the weights and the change of the weights converges quickly. Afterwards the weights are relatively stable if there is no lane changing behaviour, but whenever there is a lane changing behaviour, the weights will change to adapt the situations.

3.3.3 Learning and forgetting rates

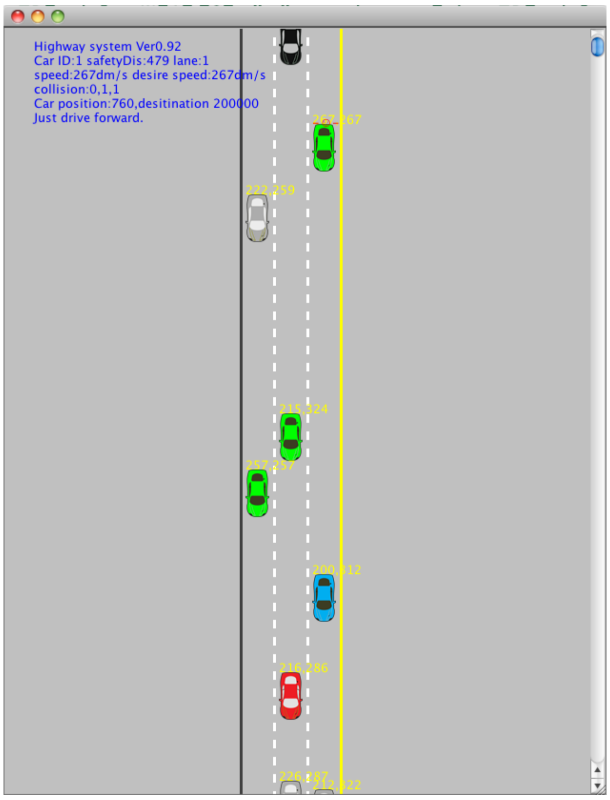
It is very complicated to decide the value of the learning and forgetting rates. The learning and forgetting rates influence the speed of the weight adjustment, therefore to a certain degree determines the convergence of the and there is not fixed rules to set these two variables. There is a whole theory and system indicates how to decide these two parameters and generally speaking they do not necessary have to be fixed. Verschure *et al.* (1992) suggest the learning and forgetting rate should be 0.5, 0.5 respectively, in this model, the forgetting rate must larger than the learning rate; otherwise the speed of the car will decrease to zero and stop. The weights and the update of the weights are shown in Fig. 4 with the learning rate 0.1, forgetting rate 0.5, and threshold is 0.5.

4 Application

4.1 Visualization

The model is written in Java, and in order to observe the parameters and the driving behaviours, a GUI is developed. The current model has only three blocks, the cars, the highway system, and the running block indicating a certain scenario or visualising the system running. As can be seen in Fig. 5, the view is fixed o the marked car (the green car with a red spot on it) and the text box in the upper left corner indicates the desired parameters of the marked car. This is a scenario for a 20 km highway with three lanes and an off-ramp located in the position of 10 km. In this specific case, car ID, real time distance, actual speed, desire speed, the collision layer inputs, car positions along the highway system, destination location, and the current driving behaviours. All the parameters can be seen by specifying them in the visualising block. The yellow numbers on each car indicate the actually speed and the desire speed. For each car, the positions, current lane positions, speed information, accelerations driving behaviours and all the learning parameters are recorded in text files.

Figure 5: DAC visualization



```
Highway system Ver0.92
Car ID:1 safetyDis:479 lane:1
speed:267dm/s desire speed:267dm/s
collision:0,1,1
Car position:760,desitination 200000
Just drive forward.
```

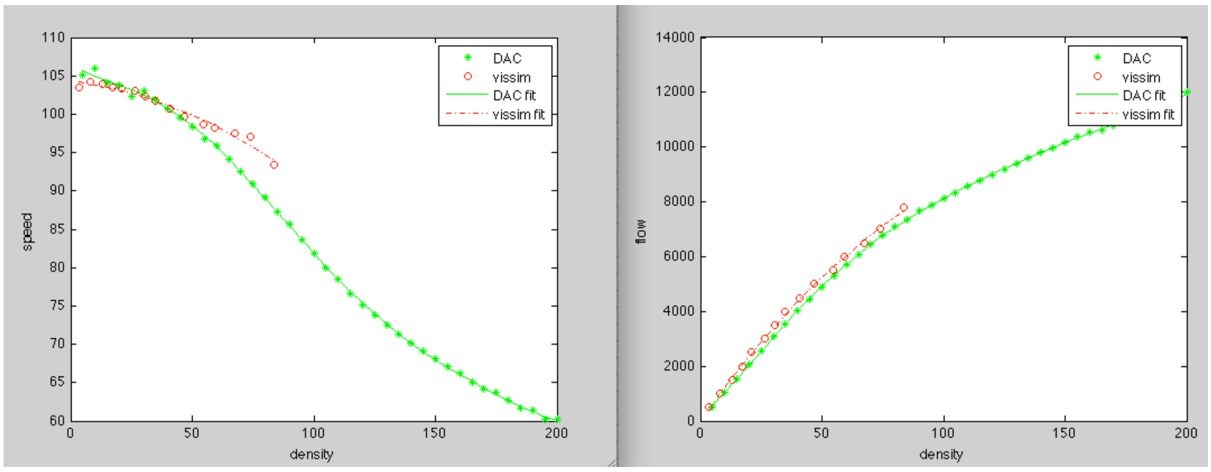
4.2 Simulation results

4.2.1 Fundamental diagram

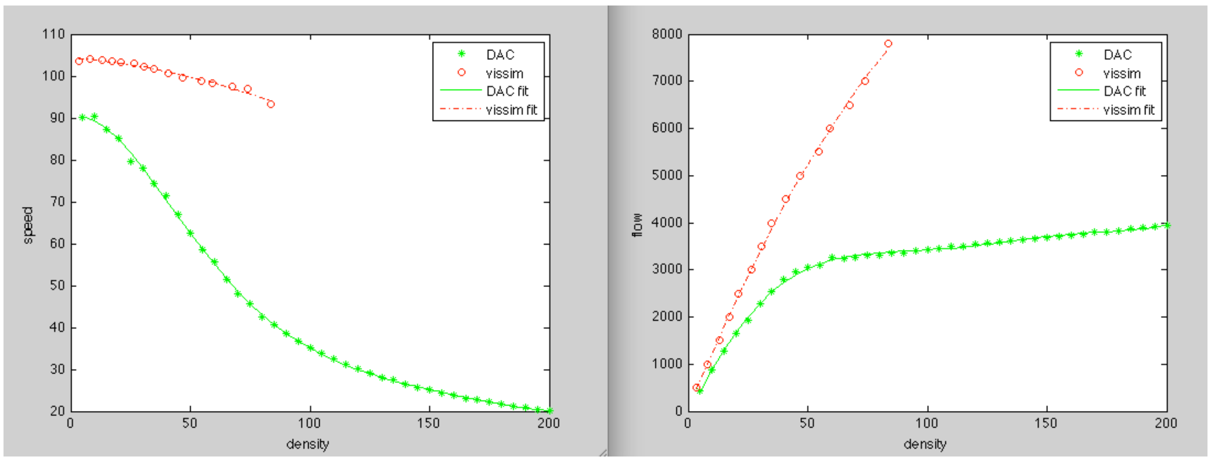
Figure 6(a) and Figure 6(b) show the fundamental diagram for both the DAC simulation and VISSIM simulation. In Figure 6(a), DAC simulation applies the safety distance as described in Eq. (6), while in Figure 6(b) , the safety distance has a linear relation with the speed. All measures are taken in average for the length of 2 km, so it is different from general fundamental diagram, which shows data collected from a certain spot on the road. For the DAC simulation, traffic density can be set when simulating the travel behaviour and the cars are originally randomly distributed on the highway system, while in VISSIM, all the traffic are generated from the origins and the input is normally the flow, and the upper limit is about 7800 vehicles per hour per three lanes (around 90 vehicles per kilometre per three lanes in density). The driving behaviour in VISSIM uses Wiederman(PTV, 2012) model. For the density-speed diagram, when the density is lower than 50 veh/km/3lanes, the results of two simulations are quite similar. As the density goes up, both simulations show speed drops, but the simulation speed from VISSIM is higher than from DAC and it fluctuates more. In the flow-density diagram, both simulations

Figure 6: Fundamental diagram with different safety distance rules

(a) Quadric safety distance



(b) Linear safety distance



do not show the tendency of a general fundamental diagram shows. The reasons for this lay in two aspects: first, the simulation is only for a single link without any disruptions like on-ramps, off ramps, or traffic controls; and secondly, these diagrams, as mentioned earlier, are results for the entire road links not a certain point on the road.

4.2.2 Time space diagram

Figure 7 shows 3D time-space diagram of both VISSIM and DAC simulations. It is a three-dimension time space diagram and the lane changing behaviour can also be observed. For the DAC simulation, the lance changing behaviours are more intense in the beginning, for the cars are generated randomly, in the first few time steps, the cars try to accommodate their positions and also try to learn all the driving behaviours. After the first few time steps, the

Figure 7: Time-space diagram for VISSIM and DAC

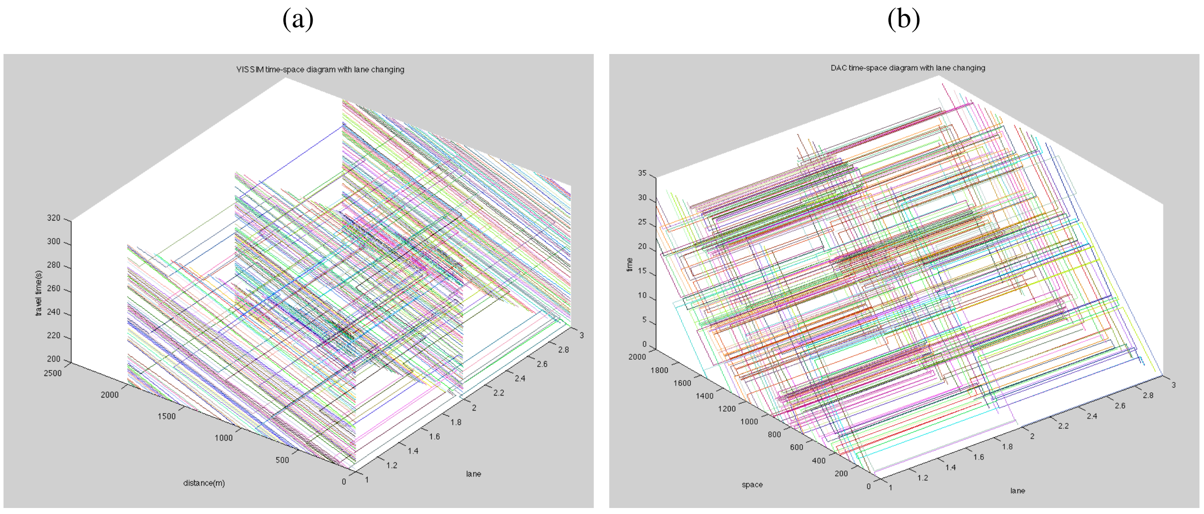
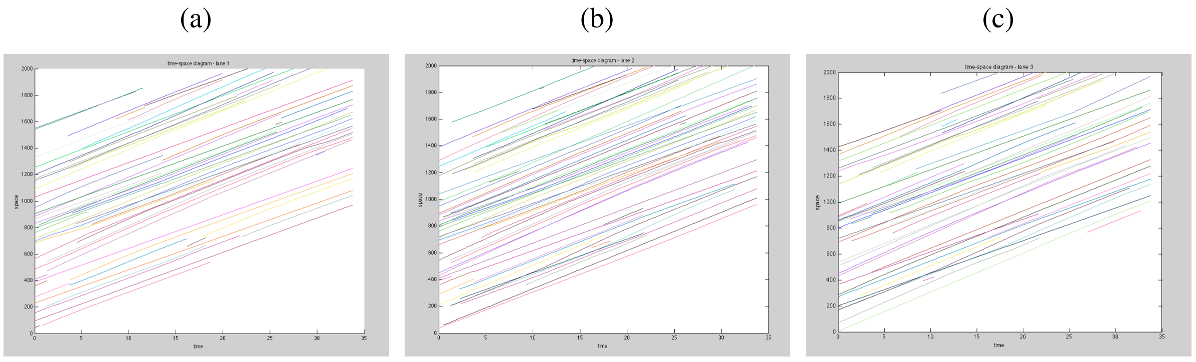


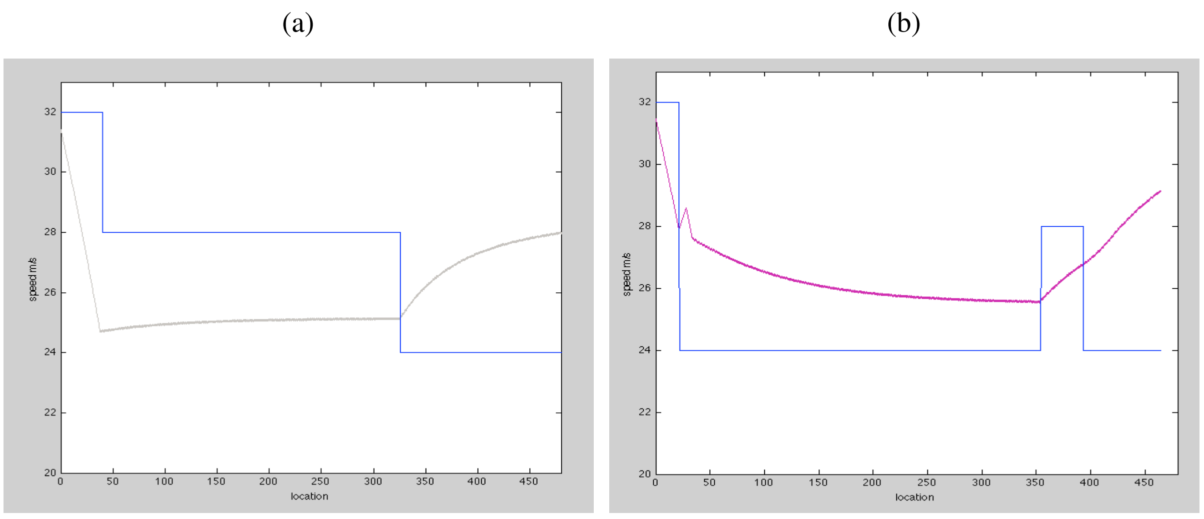
Figure 8: Time-space diagram for each lane in DAC simulation



lane changing behaviours tend to be stable. From the time-space diagram, it cannot see much difference between the two scenarios, but there is a tendency that the DAC model has more lane changing behaviours than in VISSIM (if not specified, all plots for the DAC model and VISSIM are generated under approximately same density). This is due to three reasons: the instantaneous lane changing behaviours for DAC model; the time step is twenty times more intense than VISSIM, this plus the instantaneous lane changing makes the lane changing overly flowed; lastly the lack of rear distance detection for the car also to some extent influence the lane changing times. This will be improved when introduce spatial indexing in the model, which will be discussed in the last section.

Figure 8 is time-space diagram for each lane, for the beginnings there are some crossover lines which mean some cars actually ran over other cars due to the random positions generated for initialisations. After a few time steps when the cars learns the driving behaviours, this problem

Figure 9: Speed and lane changing diagram for DAC



disappears itself. This is another way to prove the effectiveness of the learning process.

4.2.3 Speed and lane changing behaviour

Figure 9 indicate the speed and lane changing behaviour, the blue line indicate the car's current lane positions. When the car is not driving at desire speed, the car will try to change lanes. Due to the actual situation it is in, the adjacent lane might not be available, so it has to decelerate to follow the front car in the same lane. Once the adjacent lane is available, which means all the safety distances are larger than the critical values, the cars will change lanes.

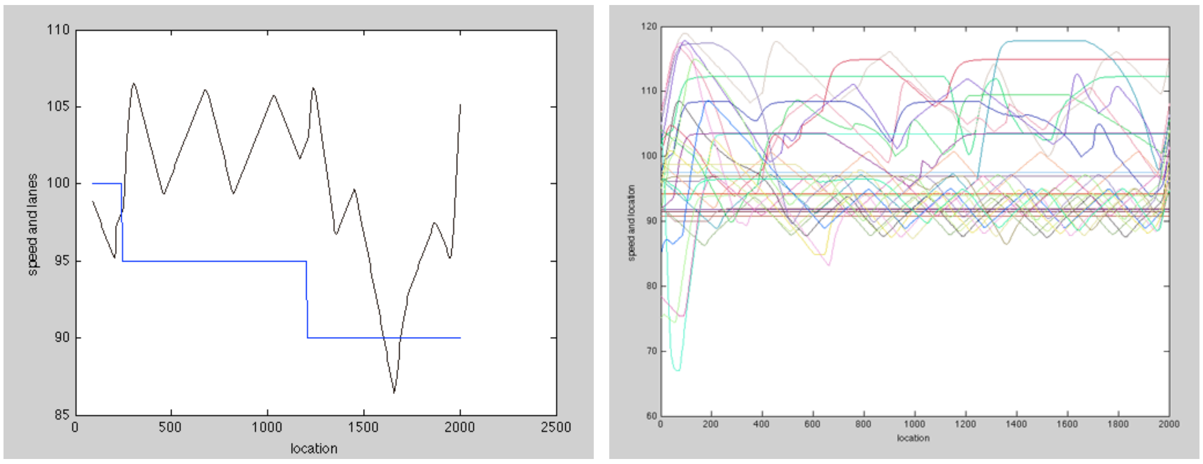
In Fig. 9(a), it can be deducted that at first, the adjacent lanes are not available. Then at around location 50, the car changes to the adjacent lane and start to accelerate for a short time, then it is more or less keeps its speed probably due to a front car. This is typical car following behaviour. And finally there is an open gap for it to change to another lane and it start to accelerate again.

In Fig. 9(b), the first part is quite similar to the car in Fig. 9(a). At first it has to decelerate because of the front car and there is no gap for it to change. Then it changes the lane and for a very short moment it start to accelerate but then probably it reaches its safety distance limit and it has to decelerate to. At around location 350, it is able to change lanes, but at location 400, it change back again. This is typical car over taking behaviour.

Figure 10 is speed and lane changing behaviour in VISSIM. As the desire speed in VISSM is not fixed, it is not easy to deduct lane-changing behaviour simply from the speed changing. And this applies for DAC model too, Fig. 9(a) and Fig. 9(b) only show typical lane changing

Figure 10: Speed and lane changing diagram for VISSIM

(a) (b)



behaviours without other disturbances. They are many factors involved for speed changing so it cannot indicate lane-changing behaviour simply from speed data.

5 Conclusions

The theory behind the simulation is really easy, yet the simulation results are promising. By allowing the agents to learn the driving behaviour themselves, the simulation avoids complicated rule-based processes but still generate similar results. The learning process is really fast and when the environment changes, the agent will learn new behaviours without any instructions or rewrite the code. Besides all the codes are written in Java, it is easy to use, easy to modify and also performs faster most of the time. By specifying certain safety distance functions, the simulation produces similar flow-density diagrams as VISSIM. And the speed plot for a single agent shows that it actually shows up a certain pattern that when performing lane changes, speeds will slightly changes.

Besides all the easy and promising features that DAC simulation has, there are also many unsolved problems. The biggest problem is that due to the lack of lateral limitation, the agents turn out to overly use lane changing behaviours. It is only a prototype and there is much room to improve and the further step for this simulator will be further discussed in the following section.

6 Further Steps

This model is only the first result of using an artificial intelligence algorithm to simulating traffic behaviours. It generates some promising results, but still it is far from perfect. There are many ways to improve the model and the following names only a few further steps of the model.

- ***Spatial indexing*** The current model is actually no lateral simulation, if the safety distances allow, the car will instantaneously change the lane. This might lead to lane changing overflow. Also detecting the cars' adjacent distances uses a traversal search, which greatly slows down the computing process. It can be approved that not the DAC algorithm slows down the program, but the distance calculating makes the program sets it back. For the next step spatial indexing will be added, the algorithm of searching will be much faster. What is more, as the quad tree indexing (a 2-d spatial indexing) is introduced, collision detection will also available, which makes the lateral lane changing behaviour possible and more realistic.
- ***Front rear detection*** For simplifying reasons, all cars in the current model only detects the distances from the front cars. If all the cars are looking after the front car distance, it will adjust its driving behaviour according to the front cars. And also due to the distance calculating problem, the searching for both front and rear cars will double the computing power. So as a trade off only front car distances are used. This actually works in the current model but not perfect, so in the next step, not only the front distances is supervised, the distances from the rear cars will also be controlled.
- ***Road network generation*** The current model needs to build the road network manually. This is not easy or even impossible when the road network is large. So the next step will add blocks for generating road networks, either from Auto CAD, shape file or xml file used by MATSim. And integrating micro simulations in MATSim simulations is one of the future steps.
- ***Traffic control*** The traffic control will also be added. What is more, for the car units will also be improved like adding special features to adapt specific uses like HOV lanes, overtaking behaviours, car2car communications, etc.
- ***HJB equation for lane changing models*** This is actually an application for the model, as the DAC algorithm has four blocks, but currently only 3 blocks are fully used. The target block can be applied to special models like using Hamilton-Jacobi-Bellman equation to optimise the strategy for off-ramp lane changing behaviours.
- ***Link speed simulation*** This is also an application for the simulation. From scatter plot of the GPS speed data, there is a typical speed drop before intersection the slightly rises. This phenomenon cannot be fully explained by the current theories. Hopefully, by improving the DAC models, it will reveal the reason of this behaviour.

7 References

Cameron, G. and G. Duncan (1996) PARAMICS—Parallel microscopic simulation of road traffic, *The Journal of Supercomputing*, **10** (1) 25–53, ISSN 0920-8542.

Cynthia, O. and G. McGwin (1999) Vision impairment and driving, *Survey of ophthamology*, **43** (6) 535–550.

Gazis, D. C., R. Herman and R. W. Rothery (1961) Nonliear follow-the-leader models of traffic flow, *Operations Research*, **9** (4) 546–567.

Ge, Q. and M. Menendez (2013) An improved approach for the sensitivity analysis of computationally expensive microscopic traffic models a case study of the zurich network in VISSIM, Washington D.C.

Haerri, J., F. Filali, C. Bonnet and M. Fiore (2006) VanetMobiSim: generating realistic mobility patterns for VANETs, paper presented at the *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, 96–97.

Kosonen, I. (2003) Microscopic freeway simulation with automatic calibration.

Li, D. and W. Wang (2003) Car following modelling and simulation in traffic flow based on driving behaviour, *Computer and Communications*, **21** (6) 32–36.

Menendez, M. (2006) An analysis of HOV lanes: Their impact on traffic, Ph.D. Thesis, University of California, Berkeley, Berkeley.

Olstam, J. J. and A. Tapani (2004) comparison of car-following models for simulation, *Technical Report*, Swedish National Road and Transport Research Institute, Swedish National Road Administration.

PTV (2012) *VISSIM 5.40 user manual*, PTV Planung Transport Verkehr AG, Karlsruhe.

Same, G. and M. N. Postorino (1994) Application of neural network for the simulation of traffic flows in a real transportation network, paper presented at the *ICANN*, 831–833, London.

Shen, G.-J. (2006) Traffic flow modeling of urban expressway using artificial neural networks, paper presented at the *Proceedings of the Third international conference on Advances in Neural Networks - Volume Part III*, ISNN'06, 15–22, Berlin, Heidelberg, ISBN 3-540-34482-9, 978-3-540-34482-7.

Smith, L., R. Beckman, D. Anson, K. Nagel and M. Williams (1995) TRANSIMS: transportation analysis and simulation system, paper presented at the *Conference: 5. National transportation planning methods applications conference, Seattle, WA (United States), 17-21 Apr 1995*.

Teodorović, D. (1999) Fuzzy logic systems for transportation engineering: the state of the art, *Transportation Research Part A: Policy and Practice*, **33** (5) 337–364.

Verschure, P. (1998) Distributed adaptive control: explorations in robotics and the biology of learning, *Informatik/Informatique*, **1**, 25–29.

Verschure, P. F., B. J. Kröse and R. Pfeifer (1992) Distributed adaptive control: The self-organization of structured behavior, *Robotics and Autonomous Systems*, **9** (3) 181–196.

Wang, X.-Y., X.-Y. Yang, G. Shan and F.-Q. Wang (2006) Review of the simulation model of driving behavior, 911–918.

Wiedemann, R. (1991) Modeling of RTI-Elements on multi-lane roads, Brussels.

Wu, J., M. Brackstone and M. McDonald (1999) Fuzzy sets and systems for a motorway microscopic simulation model, *Fuzzy Sets and Systems*, **116** (1) 65–76, November 1999.

Yang, Q. I. and H. N. Koutsopoulos (1996) A microscopic traffic simulator for evaluation of dynamic traffic management systems, *Transportation Research Part C: Emerging Technologies*, **4** (3) 113 – 129, ISSN 0968-090X.