



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

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

Project Report

**Intersection Problem**  
**Traffic flow comparison of roundabouts with crossroads**  
**controlled by trafficlights, including pedestrians**

Marcel Arikan, Nuhro Ego, Ralf Kohrt

Zurich  
Dec 2012

## **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.

Marcel Arikan

Nuhro Ego

Ralf Kohrt

## Declaration of Originality

**This sheet must be signed and enclosed with every piece of written work submitted at ETH.**

I hereby declare that the written work I have submitted entitled

Intersection Problem - Traffic flow comparison of roundabouts with crossroads  
controlled by trafficlights, including pedestrians

---

is original work which I alone have authored and which is written in my own words.\*

### Author(s)

Last name

Arikan

Ego

Kohrt

---

First name

Marcel

Nuhro

Ralf

---

### Supervising lecturer

Last name

Donnay

Balietti

---

First name

Karsten

Stefano

---

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' ([http://www.ethz.ch/students/exams/plagiarism\\_s\\_en.pdf](http://www.ethz.ch/students/exams/plagiarism_s_en.pdf)). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich, 14.12.2012

Place and date

---

Marcel Arikan

Nuhro Ego

Ralf Kohrt

Signature

---

## Contents

<b>1</b>	<b>Abstract</b>	<b>5</b>
<b>2</b>	<b>Individual contributions</b>	<b>6</b>
<b>3</b>	<b>Introduction and Motivations</b>	<b>6</b>
<b>4</b>	<b>Description of the Model and Implementation</b>	<b>6</b>
4.1	Description of the main function . . . . .	6
4.1.1	Implementation . . . . .	6
4.2	crossroad . . . . .	7
4.2.1	Implementation . . . . .	7
4.3	Roundabout . . . . .	8
4.3.1	Implementation . . . . .	8
4.4	Graphical implementation . . . . .	8
4.4.1	Preparatory work . . . . .	8
4.4.2	Implementation . . . . .	10
<b>5</b>	<b>Execution and User Instructions</b>	<b>11</b>
5.1	User Instructions . . . . .	11
<b>6</b>	<b>Simulation Results and Discussion</b>	<b>12</b>
<b>7</b>	<b>Summary and Outlook</b>	<b>16</b>
<b>A</b>	<b>Listings</b>	<b>18</b>
A.1	Matlab Codes . . . . .	18

# 1 Abstract

In our simulation, based on cellular automata, we have tried to compare roundabouts to crossroads, controlled by traffic lights, with respect to the traffic flow. We defined the traffic flow as the product of car density and average speed of the cars. Whereever reasonable, the Nagel-Schreckenberg model [1] has been implemented. The main input parameters are car density and pedestrian density. There are three different signalisation modes of the trafficlight, depending on the pedestrian density. We expected roundabouts to be more efficient at low pedestrian densities for every car-density, but if pedestrian density rises the advantage should melt. Indeed, we have found, that the flow decreases approximately linearly to zero with the pedestrian-density rising after having reached the maximum in roundabouts. In contrast, the flow in crossroad is, as expected, rather low but never vanishes.

## 2 Individual contributions

## 3 Introduction and Motivations

Several groups in this course have simulated roundabouts and crossroads before. Our work is a development of and in addition to Traffic Dynamics, written by Tony Wood and Bastian Bcheler in May 2010 [2]. In difference to their simulation we added pedestrians and implemented crossroads with lights instead of priority to the right organisation. They showed impressively, that roundabouts are much more efficient than crossroads, nearly independent of the car density. They have concluded, that their model confirms, that the increase in popularity of roundabouts over the last years is justified. In our view one important parameter was missing: the pedestrian density. As we have lived so far in cities, we have had occasions enough to observe that in the mornings and evenings some large roundabouts are just blocked, when pedestrians are allowed to cross the streets, especially when in the middle of the roundabout is a station for trams or buses. Depending on the pedestrian density we have implemented three different signalisation modes in the crossroads. For high pedestrian densities there won't be any conflicts between pedestrians and cars. So we thought that at least at this stage, crossroads may be in advantage to roundabouts.

## 4 Description of the Model and Implementation

### 4.1 Description of the main function

In our model one can compare roundabouts with crossroads, controlled by traffic lights. One can use an arbitrary combination of roundabouts and crosslights in a  $N \times M$  map.

Main input of the simulation are car and pedestrian densities, which can be entered as arrays. The simulation can be done with different probabilities for the car to go straight ahead. Cars turning left or right will have the same probability. The simulation will generate a plot over these densities as x- and y- axis and the average flow and average speed as z-axis in different colors.

$$flow = density \cdot speed$$

#### 4.1.1 Implementation

We have created a big matrix to display the simulation, containig all roads and intersections. Cars will be painted in blue and pedestrians in yellow. To the right of lanes heading towards a crossroad and to the left of lanes for cars turning left are

traffic light cells, which are red or green. Next to the lanes leaving is a traffic light too, but for pedestrians. Many matrices more are needed to store status informations that can change. So for most following matrices, there are two versions, representing current and next status. After every iteration status next will assigned to current.

## 4.2 crossroad

Depending on the pedestrian density, there are three different signalisation modes. For densities smaller than 0.3, cars that turn can always be blocked currently by a pedestrian. If the density is between 0.3 and 0.6, they can only block cars turning left. And if the density is even higher there should be no conflicts between cars and pedestrians. But if the car densities are very high, it can happen that the fixed yellow phase for changing the signalisation is too short to let all the cars leave the crossroad.

A further input parameter in the main-function is the probability of a car driving straight ahead. Cars that turn left and right have the same probability. So depending on these probabilities the relative time for light phases are different. To get the absolute time of a phase, one has to multiply it with a constant, indicating how often you change the signalisation.

It would be efficient if cars leaving one intersection would just arrive at the next one in a green-phase, so that the crossroad could take advantage of the randomisation process when entering a roundabout. A clever solution for this interesting problem is left to a next group, hopefully. We just added a phase offset between two crossroads, defined by the average time a car needs to drive from one intersection to the next and the fixed street lengths.

In contrast to the simulation of Wood and Bcheler and to the roundabout, cars entering the crossroad can have speed bigger than one cell per iteration. So cars can drive straight ahead with maximal speed of 5 cells according to the Nagel-Schreckenberg model [1]. Cars turning left or right are limited to maximal 2 cells per iteration.

### 4.2.1 Implementation

A crossroad consists of three  $6 \times 6$ -Matrices, so that for every cell information about is there a car, its speed and direction can be stored. Furthermore two  $4 \times 8$ -Matrix for 4 lanes of length 8 cells at every street heading towards the crossroad for cars

turning left are needed to decide if there's a car and store its speed. For cars driving ahead or turning right one  $4 \times 8$ -Matrix indicates the direction.

### 4.3 Roundabout

Our implementation of the roundabout consists of a circle with 12 cells and 4 roads, which lead towards it. Every street has pedestrian crossings in front of each roundabout. Like in the real world, cars inside the roundabout have priority over cars wanting to enter them and pedestrians have priority over cars at the pedestrian crossings, with the addition, that pedestrians will only walk on the road if there is no car staying or driving on the cell they wants to walk on. Inside the crossroad the speed a car can have is limited to 1 cell per iteration step.

A car which wants to leave the roundabout at the next exit will indicate, in our plot this is shown by giving these cars a darker colour. The exit a car will take is calculated from the probability ahead like in the crossroad, but with a fixed probability of 5 % for a car which will take the 4th exit (i.e. the car will turn around).

#### 4.3.1 Implementation

This is implemented with many arrays, three arrays for the circle, one which shows whether there is a car or not, and if the car wants to leave at the next exit. The second is used to store the velocity of the car and the third is used to store, how many exits the car will pass without leaving.

The entries and exits of the roundabout are randomly blocked by pedestrians. For this reason two 'buckets' are created, representing pedestrian islands between inwards and outgoing streets. If a pedestrian crosses an outgoing street, the bucket makes sure, that in the next iteration inwards street will be blocked.

### 4.4 Graphical implementation

One very important part in simulating a specific problem is visualization. First for checking if a given implementation makes sense the way one has written it, for bug fixing and for adjusting the parameters of the model to the real world problem.

#### 4.4.1 Preparatory work

Before programming something out of the head one has to create an idea of how a problem could be implemented. One has to be careful to not exaggerate the model



and fix too much on unimportant details, but to keep the main ideas clear and simple.

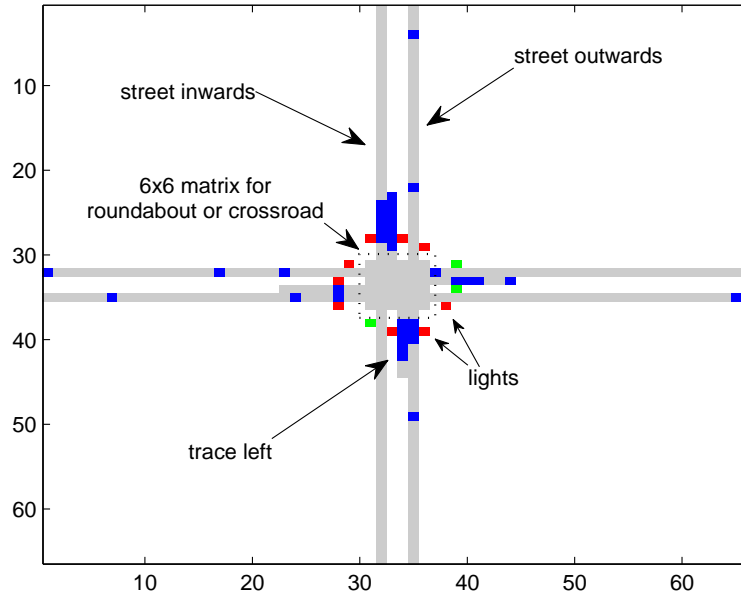


Figure 1: Overview over the implementation of the plot.

Figure 1 shows how we finally ended in, because it shows the elementary cell of each intersection. Our model consisted of

- a intersection (either roundabout or a crossroad)
- 8 streets for each direction of the crossroad (each with a out- and ingoing street)
- pedestrians (not in the figure)
- cars (in blue)
- traffic lights for the cars and pedestrians in the case of a roundabout
- a trace for the cars that turn left in the case of a crossroad

In the following sections I will explain the programming details of this figure.

#### 4.4.2 Implementation

For each type of intersections we have written a function that works out the paths of the cars and returns this information in a  $6 \times 6$  matrix. Furthermore information about the pedestrians, the traffic light phases and the cars on the left trace are returned. I implemented these by creating a large matrix

$$\text{map} = ((\text{No. of intersections in x direction}) \cdot (2 \cdot \text{streetlength} + 6) \times \\ (\text{No. of intersections in y direction}) \cdot (2 \cdot \text{streetlength} + 6))$$

in which we wrote all elements listed up above for each time step, that we looked at. The elements of the maps were encoded in a color code from 0 to 2, i.e.

- Car = 0.6
- Red light = 1.6
- Pedestrian = 0.8

that were written into this large matrix. By plotting this matrix with

`imagesc(map)`

and using a colormap we were able to create a relatively realistic model (see figure above).

We wanted to analyze how the traffic flow changes when we have many intersections in a map that are connected together to reproduce a more realistic view of the world. The configuration of each map was written into a matrix. This matrix had 0 (corresponding to roundabout) and 1 (corresponds to crossroad) as entries and corresponds to the rough structure of the map. Let's demonstrate this with Figure 2. We see in the top left and bottom right corners two crossroad and in the top right and bottom left corner two roundabouts. Naturally this would be denoted in a matrix as

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and this is how we implemented it. Then the outgoing streets of the one cell are connected to the inwards streets of the neighboring cell and programing this, this results in these dynamic maps.

Furthermore we added the support of saving a video output of the this specific configuration. For further details see the code in the listings (see Listing 13).

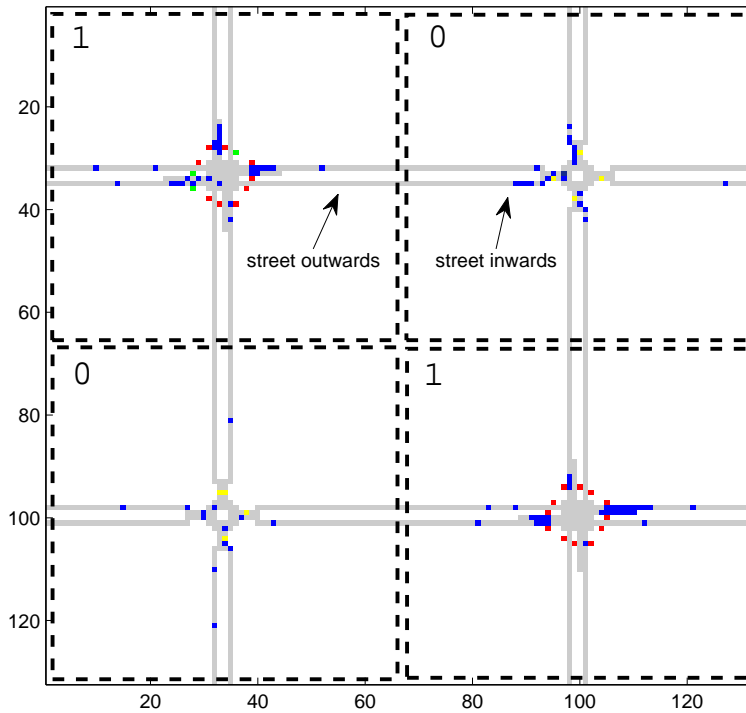


Figure 2: Details about the implementation of the plot in the case of many intersections.

## 5 Execution and User Instructions

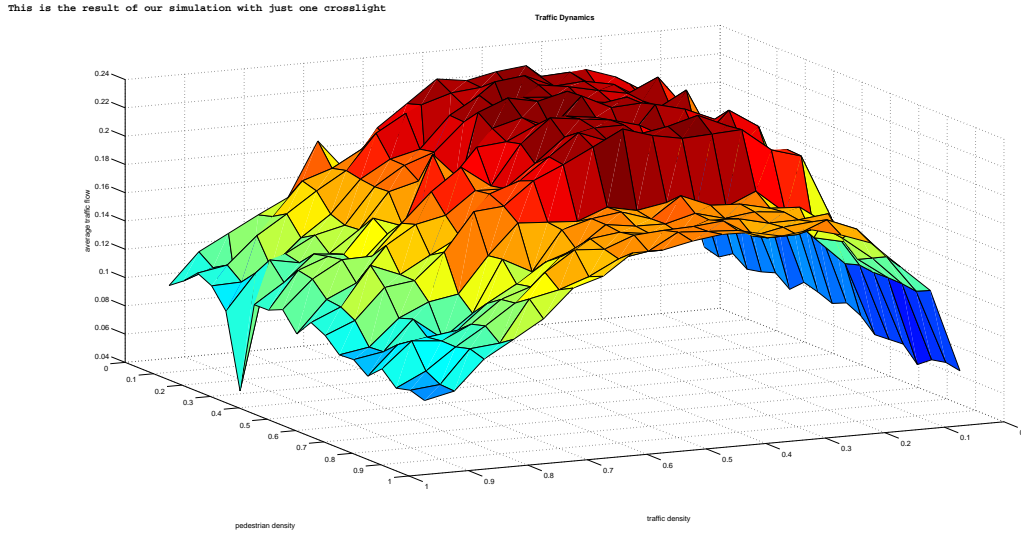
### 5.1 User Instructions

The Simulation consists of total 14 functions. Our main-function to be executed is called `traffic`. The user will be asked, what city configuration he would like to simulate. The input has to be a  $N \times M$ -Matrix with entries 0=roundabout or 1=crossroad. Then car density, probability for car driving ahead and pedestrian density are numbers between 0 and 1. A density of 0 means no cars, whereas 1 means on every single cell except the ones in the intersections stays a car. Densities can be entered as arrays, so the simulation will run for every single entry. Afterwards the user can decide, whether he wants to display the simulation, if slow motion is required and if he wants to store the data average speed and average flow.

`traffic.m` will then load `trafficloop.m`, which will then call `trafficsim.m`. Our main loop

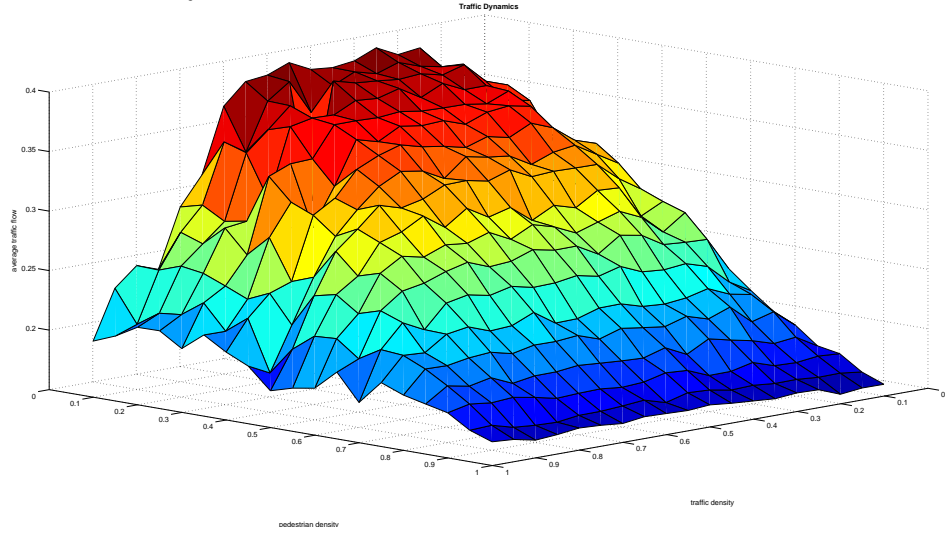
for every iteration is there and In trafficsim. is our main loop for every simulation and here the output graphics, videos etc. will be created.

## 6 Simulation Results and Discussion



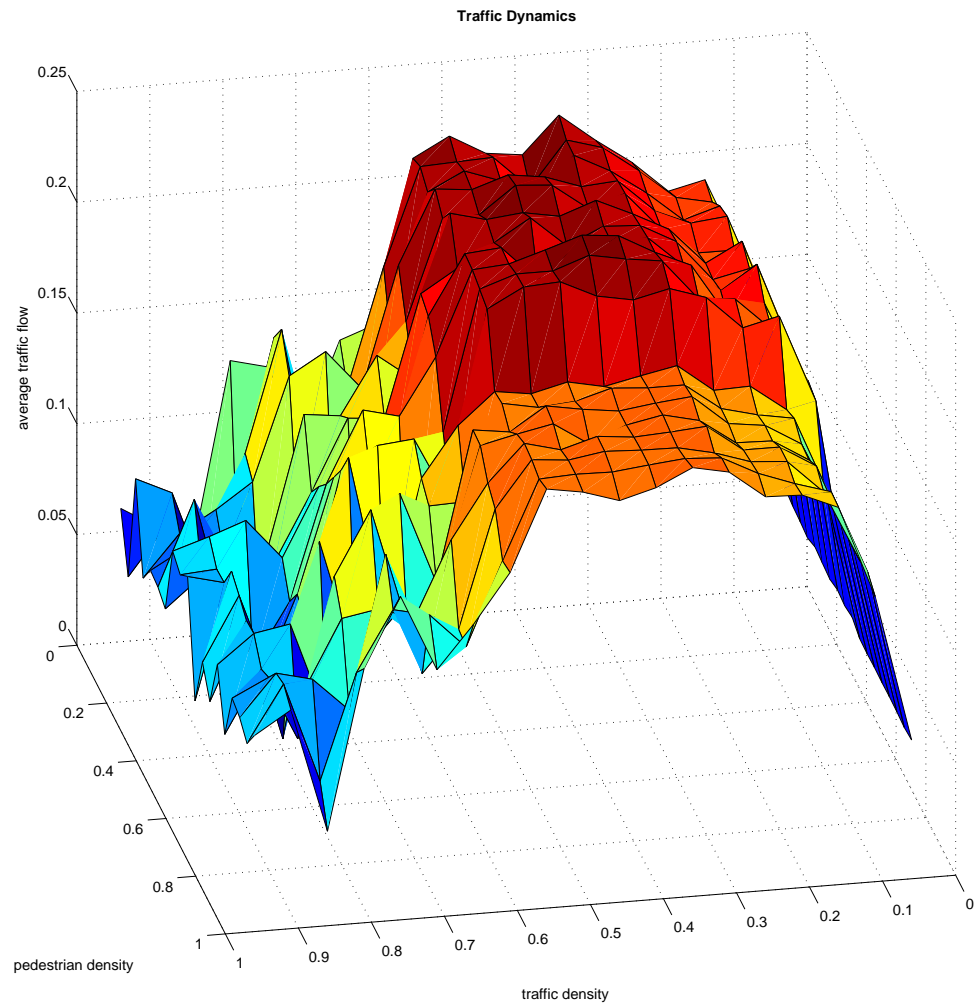
In this plot of one crosslight, one can clearly see the linear increase of traffic flow with increasing car density till 0.25 after that it is more or less constant till it drops linearly at car densities higher than 0.6. One can also see that it does depend only weakly on the pedestrian density, but for high pedestrian densities above 0.8 we see a small drop. (caused by a different traffic light mode)

This is the result of our simulation with just one roundabout



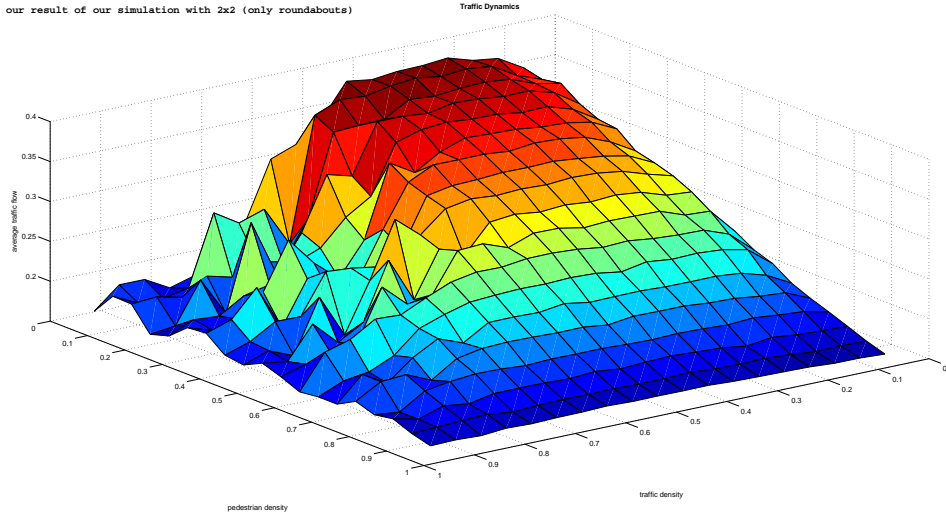
In this plot of one roundabout the linear decrease in flow with increasing pedestrian density is clearly visible. And for a const. pedestrian density one sees also the expected result, which is a linear increase, then for some time a const flow and ending with a linear decrease for increasing car densities. Compared to the flow of one trafficlight one can see that a roundabout is much more efficient (almost twice the flow) for low pedestrian densities and less efficient for low densities.

This is the result of our simulation with 2x2 (only trafficlight)



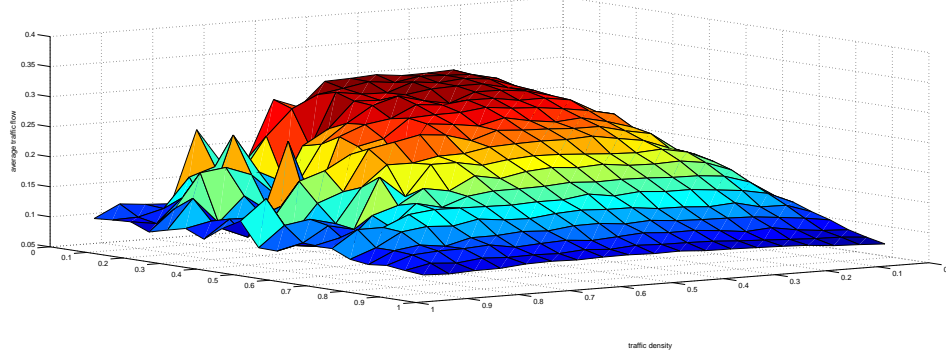
This result does not differ much from the result with just one trafficlight, but one can see that the linear in-/decrease has a higher slope.

This is our result of our simulation with 2x2 (only roundabouts)



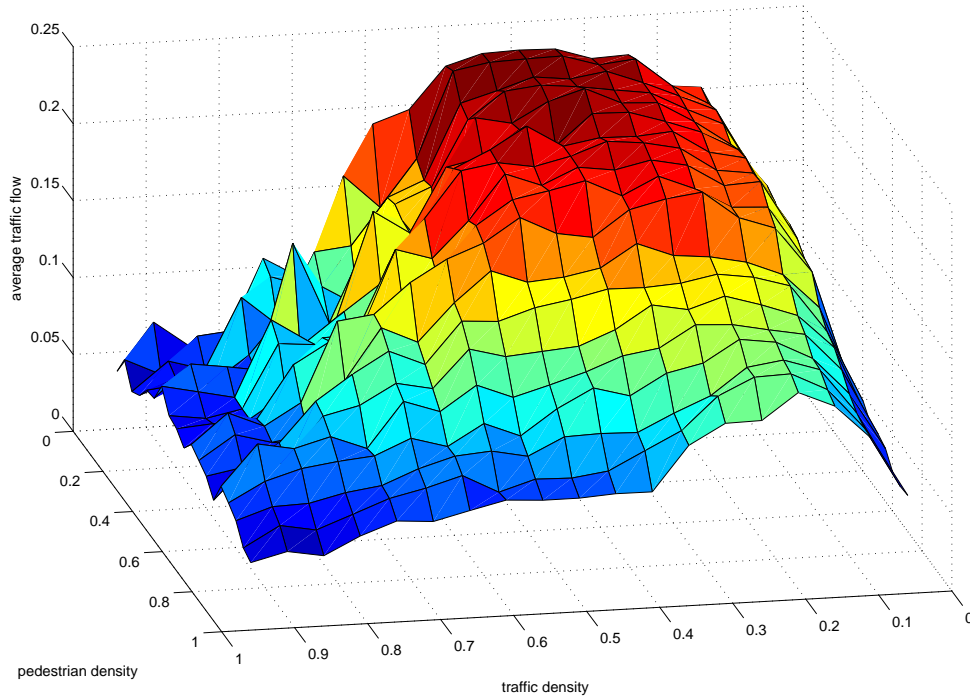
This result does not differ much from the single roundabout either, but here the slope of the linear decrease with increasing pedestrian densities is smaller.

This is our result of our simulation with 2x2 (with 2 crosslights and 2 roundabouts)



In the combination case of both methods, the characteristic looks dominated by the roundabout (i.e. it does not look much different to the graph above) but here the flow rates are lower. With higher car densities random processes are getting more important and thus the graph does not look so smooth anymore.

This is the result of our simulation with 3x3 (1 roundabout in the middle, the rest crossli  
Traffic Dynamics



Here it is visible, that just one roundabout in the middle is able to block for high pedestrian densities, were as for lower ones this is clearly dominated by the eight trafficlighs.

Here we also want to mention that we wrote our programm in a way to run our simulations on more than one computer. To simulate we used 4 student computers from ETH which were running for 2h in parallel to produce the results shown above.

## 7 Summary and Outlook

The results reflect our expectation very well. At low pedestrian densities the simulation is comparable to the one of Wood and Bcheler with no pedestrians at all. As expected, roundabouts are much more sensitive to pedestrians than crossroads. Crossroads keep their functionality to very high pedestrian densities, whereas roundabouts collapse. So it's quite reasonable to use crossroads in cities. But the maximum traffic flow in roundabouts can be twice as high than in cross rounds. For highways outside cities, roundabouts can therefore be a good choice, especially because they are simple and they normally need more space than crossroads. So, as always in life,



every system as its advantages under certain conditions.

As mentioned in the results, the different signalisation modes need to be optimized to avoid surfaces with steps, where efficiency is lost.

There are many possible modifications to develop in this simulation. As mentioned in the description, an intelligent control of the traffic light could boost the efficiency of crossroads. Vice versa a traffic light at the entry of a roundabout kicked in at high pedestrian densities could improve the efficiency and avoid a collapse. It would be interesting to analyse and simulate different hybrid models, for example with a mixed city configuration of crossroads and roundabouts, or using these controlled roundabouts.

## List of Figures

1	Overview over the implementation of the plot. . . . .	9
2	Details about the implementation of the plot in the case of many intersections. . . . .	11

## A Listings

### A.1 Matlab Codes

Listing 1: traffic.m

```
1 function traffic
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %TRAFFIC Simulation of traffic in an city map containing roundabouts and
4 %crossroads.
5 %
6 %This program requires the following subprogams:
7 %TRAFFICLOOP,TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
8 %
9 %
10 %User will be ask to determine city map,car density , pedestrian density , pahead,
    whether
11 %simulation is to be displayed or not, if the user wants to create a video
12 %of the simulation , if the user wants to show the simulaztion in slowmotion
13 %and if he wants to store the results to plot them later
14 %
15 %The city map is entered by supplying a matrix with elements '1' for
16 %crossroads and '0' for roundabouts.
17 %
18 %The density can be a scalar or a vector. If the density is a scalar
19 %TRAFFIC will run the simulation for all densities given. The elements must
20 %be in the range of [0,1].
21 %
22 %If Users chooses to display simulation (by entering 'y') a figure will
23 %open showing the animation
24 %
25 %After all simulations have finished TRAFFIC plots the average traffic flow
26 %versus the traffic density. If city map is a mix of crossroad and
27 %roundabouts the traffic distribution (cars around roundabouts or around
28 %crossroads) versus traffic density is also plotted.
29 %
30 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
31 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
32 %Fall 2012
33 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling
34 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
35 %Spring 2010
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38 close all;
39
40 %prompt city road configutation
41 c = input(['\nenter city map\n\ngive matrix elements: ', ...
42     'Priority to the right (=1) and Roundabout (=0) \n\n', ...
43     'i.e. [1 0 0;1 1 0;0 1 1]\n\n']);
44
45 %check c
46 [c_m,c_n] = size(c);
47 for a = 1:c_m
48     for b = 1:c_n
49         if ( c(c_m,c_n) ~= 1 && c(c_m,c_n) ~= 0 )
```

```

50         disp('Elements must be 0 or 1');
51         return
52     end
53 end
54 end
55
56 %prompt traffic density
57 d = input('\nenter car traffic density: ');
58 %check d
59 if ( max(d) > 1 || min(d) < 0)
60     disp('density must be in range [0,1]');
61     return
62 end
63
64 %prompt probability for car driving ahead
65 pahead = input('\nenter probability for car driving ahead: ');
66 %check pahead
67 if (max(pahead) > 1 || min(pahead) < 0)
68     disp('probability must be in range [0,1]');
69     return
70 end
71
72 %prompt pedestrian density
73 pd = input('\nenter pedestrian traffic density: ');
74 %check pd
75 if ( max(pd) > 1 || min(pd) < 0)
76     disp('density must be in range [0,1]');
77     return
78 end
79
80 %ask if simulation should be displayed
81 show = input('\ndisplay simulation graphically? yes (=y) or no (=n) ', 's');
82
83 %ask if simulation should be in slow_motion
84 slow_motion = input('\ndisplay slow-motion? yes (=y) or no (=n) ', 's');
85 if (slow_motion == 'n')
86     slow_motion = 0;
87 end
88
89 video = input('\ncreate video? yes (=y) or no (=n) ', 's');
90 if (video == 'n')
91     video = 0;
92 end
93
94
95 store_results = input('\nstore results? yes (=y) or no (=n) ', 's');
96 if (store_results == 'n')
97     store_results = 0;
98 end
99 if(store_results)
100     folder = input('\nin which folder do you want to store your results?');
101     filename = sprintf('../results/%g/config', folder);
102     save(filename, 'c', 'pahead');
103     trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results, folder);
104 else
105     trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results, 'n');
106 end
107

```

108  
109 end

## Listing 2: trafficloop.m

```
1 function trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results,
   folder)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %TRAFFIC Simulation of traffic in an city map containing roundabouts and
4 %crossroads.
5 %
6 %This program requires the following subprogams:
7 %TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
8 %
9 %
10 %This is the main loop of our simulation
11 %
12 %A project by Marcel Arian, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
13 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
14 %Fall 2012
15 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
   course "Modelling
16 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
17 %Spring 2010
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20
21 %%%
22 % define global variables
23 BUILDING = 0; %the colour for buildings
24 EMPTY_STREET = 1;
25 CAR = 0.4;
26 CAR_NEXT_EXIT = 0.6; %the colour of a car which will take the next exit
27 PEDESTRIAN = 0.8;
28
29 STREET_INTERSECTION = 7; %STREET_INTERSECTION specifies the number of elements of
   the road which will be taken care of by the crossroad/roundabout
30
31
32 if(store_results)
33     filename = sprintf(' ../results/%g/config', folder);
34     save(filename, 'c', 'pahead');
35     result = ones(1,4);
36 end
37
38 %%% runtime measurement - start
39 tic;
40
41 [c_m, c_n] = size(c);
42 %check if city map is a mix of crossroads and roundabouts or if it is made up
43 %purely of one or the other
44 mix = not( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 );
45
46 %average flow and distributions for every density supplied
47 avFlow = zeros(max(size(pd)), max(size(d)));
48 avRo = zeros(max(size(pd)), max(size(d)));
49 avCr = zeros(max(size(pd)), max(size(d)));
50
```

```

51 | if ( show == 'y' || show == 'n' ) %if show == 'y' -> simulation with graphic
    | output
52 |
53 | %create video
54 | if (video)
55 |     car_densities = mat2str(d);
56 |     pedestrian_densities = mat2str(pd);
57 |     filename = sprintf(' ../ videos/video_(%g x %g) _%s_%s.avi', c_m, c_n, ...
58 |         car_densities, pedestrian_densities);
59 |     vidObj = VideoWriter(filename);
60 |     open(vidObj);
61 | else
62 |     vidObj = 0;
63 | end
64 |
65 | for di=1:max(size(d))
66 |     for pdi=1:max(size(pd))
67 |         if (store_results)
68 |             [config_m, config_n] = size(c);
69 |             filename = sprintf(' ../ results/%g/result_(%g x %g) _%g_%g.mat',
70 |                 folder, config_m, config_n, ...
71 |                 d(di), pd(pdi));
72 |             disp(filename);
73 |             [a1,a2,a3,a4] = trafficsim(d(di),pd(pdi),c,show == 'y', ...
74 |                 BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
75 |                 STREET_INTERSECTION, ...
76 |                 pahead, slow_motion, video, vidObj);
77 |             result(1) = a1;
78 |             result(2) = a2;
79 |             result(3) = a3;
80 |             result(4) = a4;
81 |             disp(result);
82 |             save(filename, 'result');
83 |         else
84 |             [avFlow(pdi, di), avRo(pdi, di), avCr(pdi, di)] = trafficsim(d(di),pd(pdi),
85 |                 BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
86 |                 STREET_INTERSECTION, ...
87 |                 pahead, slow_motion, video, vidObj);
88 |         end
89 |     end
90 | end
91 |
92 | if (video)
93 |     close(vidObj);
94 | end
95 |
96 | if (store_results == 0)
97 |     figure(2);
98 |     %is city map is a mix of roundabout and crossroads, plot distribution
99 |     if ( mix )
100 |         %plot relativ number of cars at roundabouts and number of cars at
101 |         %crossroads versus traffic density
102 |         subplot(2,1,2);
103 |         plot(d,avRo*100,'rx',d,avCr*100,'gx');
104 |         set(gca,'FontSize',16);
105 |         title('Traffic Distribution');
106 |         xlabel('traffic density');

```

```

104         ylabel('relative numeber of cars [%]');
105         legend('around roundabouts','around crossroads');
106         ylim([0 100]);
107         subplot(2,1,1);
108     end
109
110     %plot traffic flow versus traffic density
111     plot(d,avFlow,'x');
112     set(gca,'FontSize',16);
113     title('Traffic Dynamics');
114     xlabel('traffic density');
115     ylabel('average traffic flow');
116     %ylim([0 0.5]);
117 end
118 else
119     disp('Input must be y or n!');
120 end
121
122 %%% runtime measurement - end
123 toc;
124
125 end

```

Listing 3: trafficsim.m

```

1 function [averageFlow,avCaRo,avCaCr,averageSpeed] = trafficsim(car_density ,
    pedestrian_density ,config,display , ...
2     BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,STREET_INTERSECTION, pahead ,
    slow_motion , video , vidObj)
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %TRAFFICSIM Simulation of traffic in an city map containing roundabouts and
5 %crosslights.
6 %
7 %Output:
8 %AVERAGEFLOW, Average traffic flow for given city map and density
9 %AVCARO, Average amount of cars around roundabouts
10 %AVCACR, Average amount of cars around crossroads
11 %averageSpeed, Average speed
12 %
13 %INPUT:
14 %CAR_DENSITY, CAR traffic density
15 %PEDESTRIAN_DENSITY, pedestrian traffic density
16 %CONFIG, City map
17 %DISPLAY, Turn graphics on 'true' or off 'false'
18 %+defined 'global' variables BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
    STREET_INTERSECTION
19 %PAHEAD, pobability for a car to go ahead
20 %SLOW_MOTION, show graphics in slow motion?
21 %VIDEO, generate a video?
22 %
23 %This program requires the following subprogams:
24 %ROUNDABOUT,CROSSLIGHT,CONNECTION,PDESTINATION,MEASURE_GAP,SCHRECKENBERG,PLOT_MAP
25 %
26 %A project by Marcel Arian, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
27 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
28 %Fall 2012
29 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling

```

```

30 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
31 %Spring 2010
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33
34 %dawde probability
35 dawdleProb = 0.2;
36 %street length (>5)
37 street_length = 30;
38 %number of iterations
39 nIt=201;
40
41 %dimensions of config, how many intersections in x and y direction are
42 %there?
43 [config_m, config_n] = size(config);
44
45 %initialize matrices for streets heading toward intersections
46 street_inwards = ones(4*config_m, street_length*config_n)*EMPTY_STREET;
47 inwards_speed = zeros(4*config_m, street_length*config_n);
48 %number of elements in street_inwards
49 inwards_size = sum(sum(street_inwards));
50
51 %initialize matrices for street leading away from intersections
52 street_outwards = ones(4*config_m, street_length*config_n)*EMPTY_STREET;
53 outwards_speed = zeros(4*config_m, street_length*config_n);
54
55 %initialize matrices for roundabouts
56 street_roundabout = ones(config_m, 12*config_n)*EMPTY_STREET;
57 roundabout_speed = zeros(config_m, 12*config_n);
58 roundabout_exit = zeros(config_m, 12*config_n);
59
60 %initialize matrices for crossings
61 street_crossroad = ones(6*config_m, 6*config_n)*EMPTY_STREET;
62
63 crossroad_speed = zeros(6*config_m, 6*config_n);
64 crossroad_exit = zeros(6*config_m, 6*config_n);
65 trace_left=ones(4*config_m, (STREET_INTERSECTION+1)*config_n)*EMPTY_STREET;
66 trace_left_speed=zeros(4*config_m, (STREET_INTERSECTION+1)*config_n);
67 trace_right_direction=zeros(4*config_m, (STREET_INTERSECTION+1)*config_n);
68
69 %this are the computed gaps from the crossections/roundabouts
70 inwards_gaps = zeros(config_m, config_n*4);
71
72 pedestrian_bucket = zeros(2*config_m, 4*config_n);
73
74 %initialize flow calculation variables
75 avSpeedIt = zeros(nIt+1,1);
76 %counter for cars around crossroads
77 numCaCrIt = zeros(nIt+1,1);
78 %counter for cars around crossroads
79 numCaRoIt = zeros(nIt+1,1);
80
81 %distribute cars randomly on streets for starting point
82 overall_length = sum(sum(street_inwards)) + sum(sum(street_outwards));
83 numCars = ceil(car_density * overall_length);
84 q = 1;
85
86 while ( q <= numCars )
87     w = randi(overall_length,1);

```

```

88     if ( w <= inwards_size )
89         if ( street_inwards(w) == EMPTY_STREET)
90             street_inwards(w) = CAR;
91             inwards_speed(w) = randi(5,1);
92             q = q + 1;
93         end
94     end
95     if ( w > inwards_size )
96         if ( street_outwards(w-inwards_size) == EMPTY_STREET)
97             street_outwards(w-inwards_size) = CAR;
98             outwards_speed(w-inwards_size) = randi(5,1);
99             q = q + 1 ;
100         end
101     end
102 end
103
104
105 street_roundabout_next = ones(config_m,12*config_n)*EMPTY_STREET;
106 roundabout_speed_next = zeros(config_m,12*config_n);
107 street_crossroad_next = ones(6*config_m,6*config_n)*EMPTY_STREET;
108 crossroad_speed_next = ones(6*config_m,6*config_n);
109 crossroad_exit_next = zeros(6*config_m,6*config_n);
110
111 light=zeros(config_m , 12*config_n);          %to display light signalisation
112
113 %variables for traffic light control
114 switchtime = 3;          %time to change signalement (yellow phase)
115 lighthlength = 30;      %time for staying in same signalement phase
116 aheadphase = ceil((lighthlength*pahead)/switchtime);    %time to keep green phase
117     ahead
118 turnphase = ceil((lighthlength*(1-pahead)/2)/switchtime);    %time to keep green
119     phase when turning
120 totalphase = 6 + 2*aheadphase + 4*turnphase;    %to reset the phase
121 count =0;    %counter
122 phase=0;
123 traveltime = 15+105*car_density;    %time a car needs from one intersection to the
124     next
125
126 %figure and video
127 if (display)
128     %figure for map plotting
129     fig1 = figure(1);
130     load('colormaps/colormap4', 'mycmap');
131     set(fig1, 'Colormap', mycmap);
132     titlestring = sprintf('Car density = %g, pedestrian density = %g',car_density ,
133         pedestrian_density);
134
135 %     %create video
136 %     if (video)
137 %         filename = sprintf('videos/video-(%g x %g)_%g_%g.avi', config_m , config_n ,
138 %             ...
139 %                 car_density , pedestrian_density);
140 %         vidObj = VideoWriter(filename);
141 %         open(vidObj);
142 %     end
143 end
144 %iterate over time

```



```

141 for time = 1:nIt+1
142
143 %clear values for next step
144 street_inwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
145 inwards_speed_next = zeros(4*config_m,street_length*config_n);
146 street_outwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
147 outwards_speed_next = zeros(4*config_m,street_length*config_n);
148 trace_left_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
149 trace_left_speed_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
150 trace_right_direction_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
151
152
153 %calculate traffic light phase
154 if (count == swichtime)
155     if (phase == totalphase+1)
156         phase = 0;
157     end
158     phase = phase+1;
159     count = 0;
160 else
161     count = count +1;
162 end
163
164 %iterate over all intersection
165 for a = 1:config_m
166     for b = 1:config_n
167
168         %define Index starting points for each intersection
169         tI_m = (a - 1) * 4;
170         tI_n = (b - 1) * street_length;
171
172         %positions outside intersections
173         %for every intersection iterate along streets
174         for c = tI_m + 1:tI_m +4
175             for d = tI_n + 1:tI_n+street_length
176
177                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
178                 %streets to intersections
179
180                 %deal with the STREET_INTERSECTION positions directly in front
181                 %of intersection
182                 %separately later
183                 if ( d-tI_n < street_length-STREET_INTERSECTION)
184                     %if there is a car in this position, apply
185                     %NS-Model
186                     if ( street_inwards(c,d) == CAR )
187                         %Nagel-Schreckenberg-Model
188                         gap = measure_gap(street_inwards, street_outwards,
189                                         street_length, a, b, c, d, 1, ...
190                                         inwards_gaps(a,(b - 1) *4+c-tI_m), config_m,
191                                         config_n, EMPTY_STREET,STREET_INTERSECTION);
192                         v = schreckenberg(inwards_speed(c,d), gap, dawdleProb);
193
194                         %NS 4. step: drive, move cars tspeed(c,d) cells
195                         %forward
196                         %new position
197                         street_inwards_next(c,d+v) = CAR;
198                         inwards_speed_next(c,d+v) = v;

```

```

196         end
197     end
198
199     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
200     %street from intersections
201
202     %deal with the STREET_INTERSECTION positions directly after the
203     %intersection
204     %separately later
205     if ( d-tI_n > STREET_INTERSECTION)
206         if ( street_outwards(c,d) == CAR )
207             %Nagel-Schreckenberg-Model
208             gap = measure_gap(street_inwards, street_outwards,
209                             street_length, a, b, c, d, 0, 0, ...
210                             config_m, config_n, EMPTY_STREET, STREET_INTERSECTION
211                             );
212             v = schreckenberg(outwards_speed(c,d), gap, dawdleProb);
213
214             %NS 4. step: drive, move cars fspeed(c,d) cells
215             %forward
216             %if new position is off this street, connect
217             %streets
218             if ( d + v > b * street_length )
219                 %position in new street
220                 hhh = d + v - b * street_length;
221                 %connect next street
222                 [ec,ed] = connection(a,b,c,hhh, ...
223                                     config_m, config_n, street_length);
224                 street_inwards_next(ec,ed) = CAR;
225                 inwards_speed_next(ec,ed) = v;
226             else
227                 street_outwards_next(c,d+v) = CAR;
228                 outwards_speed_next(c,d+v) = v;
229             end
230         end
231     end
232
233     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
234     %roundabouts
235
236     %check if intersection is a roundabout
237     if ( config(a,b) == 0 )
238         %define index strating point for this roundabout
239         rI_n = (b - 1) * 12;
240
241         %do roundabout calculations for this roundabout and time
242         %step
243         %call ROUNDABOUT
244         [street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-
245                             STREET_INTERSECTION:tI_n+street_length), ...
246          inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-
247                             STREET_INTERSECTION:tI_n+street_length), ...
248          street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
249                              STREET_INTERSECTION+6), ...
250          outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
251                              STREET_INTERSECTION+6), ...

```

```

247     street_roundabout_next(a, rI_n+1:rI_n+12), ...
248     roundabout_speed_next(a, rI_n+1:rI_n+12), ...
249     roundabout_exit(a, rI_n+1:rI_n+12), ...
250     pedestrian_bucket((a-1)*2+1:(a-1)*2+2, (b-1)*4+1:(b-1)*4+4)
251     , ...
252     inwards_gaps(a, (b-1)*4+1:(b-1)*4+4) ] = ...
253     roundabout(street_inwards(tI_m+1:tI_m+4, tI_n+street_length-
254     STREET_INTERSECTION:tI_n+street_length), ...
255     inwards_speed(tI_m+1:tI_m+4, tI_n+street_length-
256     STREET_INTERSECTION:tI_n+street_length), ...
257     street_outwards(tI_m+1:tI_m+4, tI_n+1:tI_n+STREET_INTERSECTION+6)
258     , ...
259     outwards_speed(tI_m+1:tI_m+4, tI_n+1:tI_n+STREET_INTERSECTION+6),
260     ...
261     street_roundabout(a, rI_n+1:rI_n+12), ...
262     roundabout_exit(a, rI_n+1:rI_n+12), ...
263     pedestrian_bucket((a-1)*2+1:(a-1)*2+2, (b-1)*4+1:(b-1)*4+4)
264     , ...
265     inwards_gaps(a, (b-1)*4+1:(b-1)*4+4), dawdleProb, ...
266     pedestrian_density, ...
267     street_inwards_next(tI_m+1:tI_m+4, tI_n+street_length-
268     STREET_INTERSECTION:tI_n+street_length), ...
269     inwards_speed_next(tI_m+1:tI_m+4, tI_n+street_length-
270     STREET_INTERSECTION:tI_n+street_length), ...
271     street_outwards_next(tI_m+1:tI_m+4, tI_n+1:tI_n+
272     STREET_INTERSECTION+6), ...
273     outwards_speed_next(tI_m+1:tI_m+4, tI_n+1:tI_n+
274     STREET_INTERSECTION+6), EMPTY_STREET, CAR, CAR_NEXT_EXIT,
275     PEDESTRIAN, STREET_INTERSECTION, pahead);
276
277 %add cars around this crossroad in this time step to
278 %counter for cars around crossroads
279 for v = tI_m+1:tI_m+4
280     for w = tI_n+1:tI_n+street_length
281         if ( street_inwards(v,w) ~= 1 )
282             numCaRoIt(time) = numCaRoIt(time) + 1;
283         end
284         if ( street_outwards(v,w) ~= 1 )
285             numCaRoIt(time) = numCaRoIt(time) + 1;
286         end
287     end
288 end
289 for y = rI_n+1:rI_n+12
290     if ( street_roundabout(a,y) ~= 1 )
291         numCaRoIt(time) = numCaRoIt(time) + 1;
292     end
293 end
294
295 end
296
297 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
298 %crossroads
299
300 %check if intersection is a crossing with priority to the right
301 if ( config(a,b) == 1 )
302     %define index starting points for this crossraod
303     pI_m = (a-1)*6;
304     pI_n = (b-1)*6;

```

```

294
295 %define trace index for this crossroad
296 traceI_m = (a - 1) * 4;
297 traceI_n = (b - 1) * 8;
298 %define light index for this crossroad
299 lightI_m = (a - 1) ;
300 lightI_n = (b - 1) * 12;
301 %calculate local off set of phase for different crossroads
302 localphase = phase+(a+b-2)*traveltime;
303 while (localphase > totalphase) %reset localphase if
    necessary
304     localphase = localphase - totalphase;
305 end
306 %do crossroad calculations for this crossroad and time step
307 %call CROSSROAD
308 [street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
309     inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
310     street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
    STREET_INTERSECTION+6), ...
311     outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
    STREET_INTERSECTION+6), ...
312     street_crossroad_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
313     crossroad_speed_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
314     crossroad_exit_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
315     pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4)
    , ...
316     inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), ...
317     trace_left_next(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n+8),
    ...
318     trace_left_speed_next(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n
    +8), ...
319     trace_right_direction_next(traceI_m+1:traceI_m+4,traceI_n+1:
    traceI_n+8), ...
320     light(lightI_m+1,lightI_n+1:lightI_n+12)] ...
321 = crosslight(street_inwards(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
322     inwards_speed(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
323     street_outwards(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6)
    , ...
324     outwards_speed(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6),
    ...
325     street_crossroad(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
326     crossroad_speed(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
327     crossroad_exit(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
328     pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4)
    , ...
329     inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), dawdleProb, ...
330     pedestrian_density, ...
331     street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
332     inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-
    STREET_INTERSECTION:tI_n+street_length), ...
333     street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
    STREET_INTERSECTION+6), ...
334     outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+

```

```

335         STREET_INTERSECTION+6),EMPTY_STREET,CAR,CAR_NEXT_EXIT,
        PEDESTRIAN,STREET_INTERSECTION, ...
        pahead, trace_left(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n+8),
        trace_left_speed(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n
        +8), trace_right_direction(traceI_m+1:traceI_m+4,traceI_n+1:
        traceI_n+8), ...
336     localphase, aheadphase, turnphase);
337
338
339     %add cars around this crossroad in this time step to
340     %counter for cars around crossroad
341     for v = tI_m+1:tI_m+4
342         for w = tI_n+1:tI_n+street_length
343             if ( street_inwards(v,w) ~= 1 )
344                 numCaCrIt(time) = numCaCrIt(time) + 1;
345             end
346             if ( street_outwards(v,w) ~= 1 )
347                 numCaCrIt(time) = numCaCrIt(time) + 1;
348             end
349         end
350     end
351     for x = pI_m+1:pI_m+6
352         for y = pI_n+1:pI_n+6
353             if ( street_crossroad(x,y) ~= 0 )
354                 numCaCrIt(time) = numCaCrIt(time) + 1;
355             end
356         end
357     end
358
359     end
360
361     end
362 end
363
364 %calculate average velocity per time step
365 avSpeedIt(time) = ( sum(sum(inwards_speed)) + sum(sum(outwards_speed)) + ...
366     sum(sum(roundabout_speed)) + sum(sum(crossroad_speed)) ) / numCars;
367
368 %plot the map in this timestep into the figure
369 if (display)
370     map = plot_map(street_length, config, car_density, display, ...
371         street_inwards, street_outwards, street_roundabout, street_crossroad,
372         ...
373         BUILDING,EMPTY_STREET, light, trace_left, STREET_INTERSECTION);
374     %illustrate traffic situation (now, not of next time step)
375     imagesc(map);
376     title(titlestring, 'FontWeight','bold');
377     drawnow;
378     if (video)
379         % get the current frame
380         currFrame = getframe(fig1);
381         % add the current frame
382         writeVideo(vidObj,currFrame);
383     end
384 end
385
386 if (slow_motion)
387     pause(1);

```

```

387     end
388
389     %move on time step on
390     street_inwards = street_inwards_next;
391     inwards_speed = inwards_speed_next;
392     street_outwards = street_outwards_next;
393     outwards_speed = outwards_speed_next;
394     street_roundabout = street_roundabout_next;
395     roundabout_speed = roundabout_speed_next;
396     street_crossroad = street_crossroad_next;
397     crossroad_speed = crossroad_speed_next;
398     crossroad_exit = crossroad_exit_next;
399     trace_left = trace_left_next;
400     trace_left_speed = trace_left_speed_next;
401     trace_right_direction = trace_right_direction_next;
402
403 end
404
405 % if (video)
406 %     close(vidObj);
407 % end
408
409 %overall average velocity
410 averageSpeed = sum(avSpeedIt) / max(size(avSpeedIt));
411 %overall average flow
412 averageFlow = car_density * averageSpeed;
413
414 %average relative amount of cars around roundabouts
415 avCaRo = sum(numCaRoIt) / ( max(size(numCaRoIt)) * numCars );
416 %average relative amount of cars around crossroads
417 avCaCr = sum(numCaCrIt) / ( max(size(numCaCrIt)) * numCars );
418
419 end

```

Listing 4: measure-gap.m

```

1 function [ gap ] = measure_gap(street_inwards, street_outwards, street_length, a, b,
    c, d, inwards, inwards_gap, config_m, config_n, EMPTY_STREET, STREET_INTERSECTION
    )
2 %MEASURE_GAP this measures the gap to the next car
3 % how big is gap (to car ahead or intersection)?
4
5
6 e = 0;
7 iterate = 1;
8 while (iterate) %iterate while iterate is 1
9     if(inwards)
10         e = e + 1;
11         iterate = e <= 5 && d + e <= b * street_length - STREET_INTERSECTION +
            inwards_gap && ...
12         street_inwards(c,d+e) == EMPTY_STREET; %STREET_INTERSECTION
            specifies the number of elements of the road inwards which will be taken
            care of by the crossroad/roundabout
13     else
14         e = e + 1;
15         %if gap is bigger than distance to edge, connect
16         %steets
17         if ( d + e > b * street_length)

```

```

18         %testing position in new street
19         hh = d + e - b * street_length;
20         %connect to next street
21         [ec,ed]=connection(a,b,c,hh, ...
22             config_m,config_n,street_length);
23         while ( street_inwards(ec,ed) == EMPTY.STREET && e <= 5 )
24             e = e + 1;
25             %testing position in new street
26             hh = d + e - b * street_length;
27             %connect to next street
28             [ec,ed]=connection(a,b,c,hh, ...
29                 config_m,config_n,street_length);
30         end
31         iterate = 0;
32     else
33         iterate = e <= 5 && street_outwards(c,d+e) == EMPTY.STREET;    %% <= 4 b
34                                     .c. it'll be 5 after this loop
35     end
36 end
37 gap = e - 1;
38
39 end

```

Listing 5: connection.m

```

1 function [cNew,dNew] = connection(aOld,bOld,cOld,posNew,m,n,length)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %CONNECTION Deside to which street a certain street connects to
4 %
5 %INPUT:
6 %AOLD column index of intersection
7 %BOLD, row index of intersection
8 %COLD, column index in t of old position
9 %posNEW, position in new street
10 %M, number of columns in city map
11 %N, number of rows in city map
12 %LENGTH, Length of a street
13 %
14 %OUTPUT:
15 %CNEW, Column index in t of new position
16 %DNEW, Row index in t of new position
17 %
18 %A project by Marcel Arian, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
19 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
20 %Fall 2012
21 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
22 %course "Modelling
23 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25
26 %street heading up from intersection
27 if ( mod(cOld,4) == 1 )
28     %if there is a intersections above, connect to it
29     if ( aOld > 1)
30         cNew = (aOld - 2) * 4 + 3;
31         dNew = (bOld - 1) * length + posNew;

```

```

32     %otherwise connect to other side of map
33     else
34         cNew = (m - 1) * 4 + 3;
35         dNew = (bOld - 1) * length + posNew;
36     end
37 end
38
39 %street heading left from intersection
40 if ( mod(cOld,4) == 2 )
41     %if there is a intersection to the left, connect to it
42     if ( bOld > 1 )
43         cNew = aOld * 4;
44         dNew = (bOld - 2) * length + posNew;
45     %otherwise connect to other side of map
46     else
47         cNew = aOld * 4;
48         dNew = (n - 1) * length + posNew;
49     end
50 end
51
52 %street heading down from intersection
53 if ( mod(cOld,4) == 3 )
54     %if there is a intersection below, connect to it
55     if ( aOld < m )
56         cNew = aOld * 4 + 1;
57         dNew = (bOld - 1) * length + posNew;
58     %otherwise connect to other side of map
59     else
60         cNew = 1;
61         dNew = (bOld - 1) * length + posNew;
62     end
63 end
64
65 %street heading right from intersection
66 if ( mod(cOld,4) == 0 )
67     %if there is a intersection to the right, connect to it
68     if ( bOld < n )
69         cNew = (aOld - 1) * 4 + 2;
70         dNew = bOld * length + posNew;
71     %otherwise connect to other side of map
72     else
73         cNew = (aOld - 1) * 4 + 2;
74         dNew = posNew;
75     end
76 end

```

Listing 6: pdestination.m

```

1 function [pfirst] = pdestination
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %PDESTINATION Deside where a car is going
4 %
5 %OUTPUT:
6 %PFIRST = 0.1 car turns right
7 %         = 0.4 car goes straight ahead
8 %         = 0.7 car turns left
9 %
10 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling

```



```

11 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
12 %Fall 2012
13 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
   course "Modelling
14 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
15 %Spring 2010
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 %decide which direction car is going
19 u = randi(12,1);
20 %probabilty 6/12 car goes straight ahead
21 if ( u <= 6 )
22     pfirst = 0.4;
23 end
24 %probabilty 3/12 car turns right
25 if ( u >= 7 && u <= 9 )
26     %indicate right
27     pfirst = 0.7;
28 end
29 %probabilty 3/12 car turns left
30 if ( u >= 10 && u <= 12 )
31     pfirst = 0.1;
32 end
33
34 end

```

Listing 7: schreckenberg.m

```

1 function [ speed ] = schreckenberg(speed, gap, dawdleProb)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %SCHRECKENBERG Nagel-Schreckenberg-Model
4 %
5 %OUTPUT: new speed of the selected car
6 %
7 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
8 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
9 %Fall 2012
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 %NS 1. step: increase velocity if < 5
13 if ( speed < 5)
14     speed = speed + 1;
15 end
16
17 %NS 2. step: adapt speed to gap
18 %reduce speed if gap is too small
19 if ( speed > gap )
20     speed = gap;
21 end
22
23 %NS 3. step: dawdle
24 if ( rand < dawdleProb && speed ~= 0 )
25     speed = speed - 1;
26 end
27
28 end

```

Listing 8: roundabout.m

```

1 function [street_inwards_next, ...
2     inwards_speed_next, ...
3     street_outwards_next, ...
4     outwards_speed_next, ...
5     street_roundabout_local_next, ...
6     roundabout_speedlocal_next, ...
7     roundabout_exit_local_next, ...
8     pedestrian_bucket, inwards_gaps] ...
9 = roundabout(street_inwards, ...
10    inwards_speed, ...
11    street_outwards, ...
12    outwards_speed, ...
13    street_roundabout, ...
14    roundabout_exit, pedestrian_bucket, ...
15    inwards_gaps, dawdleProb, ...
16    pedestrian_density, ...
17    street_inwards_next, ...
18    inwards_speed_next, ...
19    street_outwards_next, ...
20    outwards_speed_next, EMPTY_STREET, CAR, CAR_NEXT_EXIT, PEDESTRIAN,
    STREET_INTERSECTION, pahead)
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %ROUNDABOUT Calculation of update for a certain roundabout, density and
23 %time step
24 %
25 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
26 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
27 %Fall 2012
28 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling
29 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
30 %Spring 2010
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33 %clear local next variables
34 street_roundabout_local_next = ones(1,12)*EMPTY_STREET;
35 roundabout_speedlocal_next = zeros(1,12);
36 roundabout_exit_local_next = zeros(1,12);
37
38 temp_roundabout_pedestrian_bucket = pedestrian_bucket;
39
40 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41 %car in front of roundabout
42
43 for k = 1:4
44     if ( street_inwards(k,STREET_INTERSECTION+1) == CAR )
45         %entering roundabout with velocity 1 when possible
46         %roundabout position index
47         iR = mod(3*k+1,12);
48         % enter roundabout if car at position k*3 is about to exit and
49         % there is no car at position 3*k+1
50         if ( roundabout_exit(k*3) <= 1 && street_roundabout(iR) == EMPTY_STREET )
51             %enter roundabout
52             %decide which exit car is going to take
53             u = rand(1);
54             %if it takes 1. exit
55             if ( u <= (0.95/2*(1-pahead)))

```

```

56         roundabout_exit_local_next(iR) = 1;
57         %indicate
58         street_roundabout_local_next(iR) = CAR_NEXT.EXIT;
59         roundabout_speedlocal_next(iR) = 1;
60         %if it takes 2. exit
61         elseif ( u <= (0.95/2*(1+pahead)))
62             roundabout_exit_local_next(iR) = 2;
63             street_roundabout_local_next(iR) = CAR;
64             roundabout_speedlocal_next(iR) = 1;
65         %if it takes 3. exit
66         elseif ( u <= 0.95 )
67             roundabout_exit_local_next(iR) = 3;
68             street_roundabout_local_next(iR) = CAR;
69             roundabout_speedlocal_next(iR) = 1;
70         %if it takes 4. exit (turns around)
71         else
72             roundabout_exit_local_next(iR) = 4;
73             street_roundabout_local_next(iR) = CAR;
74             roundabout_speedlocal_next(iR) = 1;
75         end
76
77         %car waiting in front of roundabout
78         else
79             street_inwards_next(k,STREET_INTERSECTION+1) = street_inwards(k,
80                 STREET_INTERSECTION+1);
81             inwards_speed_next(k,STREET_INTERSECTION+1) = 0;
82         end
83     end
84
85     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86     %pedestrians
87
88
89     for k = 1:4
90         r = rand(1);
91         if (( street_inwards(k,STREET_INTERSECTION) == EMPTY_STREET || street_inwards(k,
92             STREET_INTERSECTION) == PEDESTRIAN) && ...
93             (r <= pedestrian_density || pedestrian_bucket(1,k) > 0))
94             street_inwards_next(k,STREET_INTERSECTION) = PEDESTRIAN;
95             inwards_speed_next(k,STREET_INTERSECTION) = 0;
96             if(r <= pedestrian_density)
97                 temp_roundabout_pedestrian_bucket(2,k) = 1;
98             end
99             if(pedestrian_bucket(1,k) > 0)
100                 temp_roundabout_pedestrian_bucket(1,k) = 0;
101             end
102         r = rand(1);
103         if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN
104             ) && ...
105             (r <= pedestrian_density || pedestrian_bucket(2,k) > 0))
106             street_outwards_next(k,2) = PEDESTRIAN;
107             outwards_speed_next(k,2) = 0;
108             if(r <= pedestrian_density)
109                 temp_roundabout_pedestrian_bucket(1,k) = 1;
110             end
111             if(pedestrian_bucket(2,k) > 0)

```

```

111         temp_roundabout_pedestrian_bucket(2,k) = 0;
112     end
113 end
114 if(0)
115     if (( street_roundabout(k*3-1) == EMPTY_STREET || street_roundabout(k*3-1)
116         == PEDESTRIAN) && roundabout_pedestrian_bucket(k) > 0)
117         street_roundabout_local_next(k*3-1) = PEDESTRIAN;
118         roundabout_speedlocal_next(k*3-1) = 0;
119         roundabout_exit_local_next(k*3-1) = 0;
120         if(roundabout_pedestrian_bucket(k) >= 1)
121             roundabout_pedestrian_bucket(k) = roundabout_pedestrian_bucket(k)-1;
122         end
123     elseif ( street_inwards(k,2) == PEDESTRIAN && roundabout_pedestrian_bucket(k)
124         == 0)
125         street_roundabout_local_next(k*3-1) = EMPTY_STREET;
126         roundabout_speedlocal_next(k*3-1) = 0;
127         roundabout_exit_local_next(k*3-1) = 0;
128     end
129 end
130 pedestrian_bucket = temp_roundabout_pedestrian_bucket;
131 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
132 %car outside roundabout
133
134
135
136 for k = 1:4
137     for j = 1:STREET_INTERSECTION
138         e = 1;
139         while (e <= 5 && ((street_outwards(k,j+e) == EMPTY_STREET &&
140             street_outwards_next(k,j+e) == EMPTY_STREET) || ...
141                 (street_outwards(k,j+e) == PEDESTRIAN && street_outwards_next(k,
142                     j+e) == EMPTY_STREET) ))
143             e = e + 1;
144         end
145         gap = e - 1;
146         v = schreckenberg(outwards_speed(k,j), gap, dawdleProb);
147         if(street_outwards(k,j) == CAR)
148             if ( (street_outwards(k,j+v) == EMPTY_STREET && street_outwards_next(k,j
149                 +v) == EMPTY_STREET) || ...
150                 (street_outwards(k,j+v) == PEDESTRIAN && street_outwards_next(k,
151                     j+v) == EMPTY_STREET) )
152                 street_outwards_next(k,j+v) = CAR;
153                 outwards_speed_next(k,j+v) = v;
154             else
155                 street_outwards_next(k,j) = CAR;
156                 outwards_speed_next(k,j) = 0;
157             end
158         end
159         e = 1;
160         while (e <= 5 && j + e <= STREET_INTERSECTION+1 && ((street_inwards(k,j+e)
161             == EMPTY_STREET && street_inwards_next(k,j+e) == EMPTY_STREET) || ...
162                 ( street_inwards(k,j+e) == PEDESTRIAN && street_inwards_next(k,j
163                     +e) == EMPTY_STREET) ))
164             e = e + 1;
165         end
166         gap = e - 1;

```

```

161     v = schreckenberg(inwards_speed(k,j), gap, dawdleProb);
162     if(j == 1)
163         inwards_gaps(1,k) = gap;
164     end
165     if(street_inwards(k,j) == CAR)
166         if ( ( street_inwards(k,j+v) == EMPTY_STREET && street_inwards_next(k,j+
167             v) == EMPTY_STREET) || ...
168             ( street_inwards(k,j+v) == PEDESTRIAN && street_inwards_next(k,j
169                 +v) == EMPTY_STREET) )
170             street_inwards_next(k,j+v) = CAR;
171             inwards_speed_next(k,j+v) = v;
172         else
173             street_inwards_next(k,j) = CAR;
174             inwards_speed_next(k,j) = 0;
175         end
176     end
177 end
178
179 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
180 %car in roundabout
181
182 for j = 1:12
183     if ( street_roundabout(j) == CAR || street_roundabout(j) == CAR_NEXT_EXIT )
184
185         %cars in roundabout not at an exit
186         if (mod(j,3) ~= 0 )
187             %if space free, move one forward
188             if ( street_roundabout(j+1) == EMPTY_STREET &&
189                 street_roundabout_local_next(j+1) == EMPTY_STREET)
190                 %take new position
191                 street_roundabout_local_next(j+1) = street_roundabout(j);
192                 roundabout_speedlocal_next(j+1) = 1;
193                 roundabout_exit_local_next(j+1) = roundabout_exit(j);
194             %if no space free, stay
195             else
196                 street_roundabout_local_next(j) = street_roundabout(j);
197                 roundabout_speedlocal_next(j) = 0;
198                 roundabout_exit_local_next(j) = roundabout_exit(j);
199             end
200
201             %car at an exit
202             else
203
204                 %if car is at its exit
205                 if ( roundabout_exit(j) == 1 )
206                     %if space free, leave roundabout
207                     if ( street_outwards(j/3,1) == EMPTY_STREET )
208                         street_outwards_next(j/3,1) = CAR;
209                         outwards_speed_next(j/3,1) = 1;
210                     %if no space free, stay
211                     else
212                         street_roundabout_local_next(j) = street_roundabout(j);
213                         roundabout_speedlocal_next(j) = 0;
214                         roundabout_exit_local_next(j) = roundabout_exit(j);
215                     end

```

```

216
217         %car at an exit but not the one its taking
218     else
219         %connect street_roundabout(12) with street_roundabout(1)
220         if ( j == 12 )
221             j1 = 1;
222         else
223             j1 = j+1;
224         end
225         %if space free , move one forward and decrease exit
226         %counter
227         if ( street_roundabout(j1) == EMPTY.STREET )
228             %decrease exit by one
229             roundabout_exit_local_next(j1) = roundabout_exit(j) - 1;
230             roundabout_speedlocal_next(j1) = 1;
231             if ( roundabout_exit_local_next(j1) == 1 )
232                 %indicate
233                 street_roundabout_local_next(j1) = CAR_NEXT_EXIT;
234             else
235                 street_roundabout_local_next(j1) = CAR;
236             end
237             %if no space free , stay
238         else
239             street_roundabout_local_next(j) = street_roundabout(j);
240             roundabout_speedlocal_next(j) = 0;
241             roundabout_exit_local_next(j) = roundabout_exit(j);
242         end
243     end
244 end
245 end
246 end
247
248 end

```

Listing 9: crosslight.m

```

1 function [street_inwards_next , ...
2     inwards_speed_next , ...
3     street_outwards_next , ...
4     outwards_speed_next , ...
5     street_crossroad_next , ...
6     crossroad_speed_next , ...
7     crossroad_exit_next , ...
8     pedestrian_bucket , inwards_gaps , ...
9     trace_left_next , trace_left_speed_next , trace_right_direction_next , trafficlight
10 ] ...
11 = crosslight(street_inwards , ...
12     inwards_speed , ...
13     street_outwards , ...
14     outwards_speed , ...
15     street_crossroad , ...
16     crossroad_speed , ...
17     crossroad_exit , pedestrian_bucket , ...
18     inwards_gaps , dawdleProb , ...
19     pedestrian_density , ...
20     street_inwards_next , ...
21     inwards_speed_next , ...
22     street_outwards_next , ...

```

```

22     outwards_speed_next,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
        STREET_INTERSECTION, ...
23     pahead, trace_left, trace_left_speed, trace_right_direction, ...
24     localphase, aheadphase, turnphase)
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 %CROSSLIGHT Calculation of update for a certain crossroad, including lane for cars
        turning left
27 %and 8 cells of street_inwards and street_outwards
28 %
29 %additional Output:
30 %inwards_gaps: gap to place car from street_inwards when entering the 8 cells in
        front of
31 %crossroad calculated by crosslight
32 %trafficlight: signalisation for every lane to plot traffic lights in
33 %display
34 %
35 %additional INPUT:
36 %pahead: probability for car driving ahead
37 %localphase: signalisation phase for this crossroad
38 %aheadphase, turnphaae: relative time for signalisation staying green for
        %car turning or driving ahead
39 %
40 %
41 %This program requires the following subprogams:
42 %settrafficlight
43 %crosslight_measure_gap
44 %crosslight_next_ij
45 %schreckenber
46 %pdestination
47 %
48 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
49 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
50 %Fall 2012
51 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
        course "Modelling
52 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
53 %Spring 2010
54 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55
56 NO_EXIT_YET = 0;
57 EXIT_LEFT = 5;
58 EXIT_RIGHT = 6;
59 EXIT_STRAIGHT_TOP = 3;
60 EXIT_STRAIGHT_LEFT = 4;
61 EXIT_STRAIGHT_BOTTOM = 1;
62 EXIT_STRAIGHT_RIGHT = 2;
63
64 %clear local next variables
65 street_crossroad_next = ones(6,6)*EMPTY_STREET;
66 crossroad_speed_next = zeros(6,6);
67 crossroad_exit_next = zeros(6,6);
68 trace_left_next = ones(4,8)*EMPTY_STREET;
69 trace_left_speed_next = zeros(4,8);
70 trace_right_direction_next = ones(4,8)*NO_EXIT_YET;
71
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73 %set traffic light
74 %trafficlight = zeros(12,1) 0=red for car and pedestrians
75 trafficlight = settrafficlight(localphase, aheadphase, turnphase, pedestrian_density

```

```

    );
76 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 %place randomly pedestrians
78 for k = 1:4
79     if (rand(1) <= pedestrian_density )
80         pedestrian_bucket(2,k) = 1; %use bucket to make sure that pedestrians
            accumulate during redphase
81     end
82     if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN
            ) && ...
83         pedestrian_bucket(2,k) > 0 && trafficlight(1+(k-1)*3,1)==1 )
84         street_outwards_next(k,2) = PEDESTRIAN;
85         outwards_speed_next(k,2) = 0;
86         pedestrian_bucket(2,k) = 0;
87     elseif ( street_outwards(k,2) == PEDESTRIAN)
88         street_outwards_next(k,2) = EMPTY_STREET;
89         outwards_speed_next(k,2) = 0;
90     end
91 end
92 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
93 %car in front of crossroad and initializing direction
94
95 for k = 1:4
96     for l=1:STREET_INTERSECTION+1
97         %initializing randomly directions
98         if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==NO_EXIT_YET)
99             u=rand(1);
100             %if it goes left
101             if ( u < ((1-pahead)/2))
102                 trace_right_direction(k,l) = EXIT_LEFT;
103                 %if it goes ahead
104             elseif ( u <= ((1+pahead)/2))
105                 trace_right_direction(k,l) = k;
106
107                 %if it goes right
108             else
109                 trace_right_direction(k,l) = EXIT_RIGHT;
110
111             end
112         end
113
114         %take cars with EXIT_LEFT waiting into trace_left if space is free
115         if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==EXIT_LEFT)
116             if (trace_left(k,l) == EMPTY_STREET)
117                 trace_left_next(k,l) = CAR;
118                 trace_left_speed_next(k,l) = inwards_speed(k,l);
119             else
120                 street_inwards_next(k,l) = CAR;
121                 inwards_speed_next(k,l) = 0;
122                 trace_right_direction_next(k,l)=EXIT_LEFT;
123             end
124         end
125     end
126
127     %inwards
128     if (street_inwards(k,l) == CAR && trace_right_direction(k,l)~=EXIT_LEFT)
129         gap = crosslight_measure_gap(-k, l, trace_right_direction(k,l) ,
            street_crossroad , ...

```



```

130         street_outwards, street_outwards_next, 1, street_inwards,
131         street_inwards_next, trafficlight(3*k,1), ...
132         EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
133         EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
134         EMPTY_STREET);
135     v = schreckenberg(inwards_speed(k,l),gap,dawdleProb);
136     if(l == 1)
137         inwards_gaps(1,k) = gap;
138     end
139     if (l+v<=STREET_INTERSECTION+1) %not yet entering crossroad
140         street_inwards_next(k,l+v) = CAR;
141         inwards_speed_next(k,l+v) = v;
142         trace_right_direction_next(k,l+v) = trace_right_direction(k,l);
143     else
144         ni = -k; %minus indices for
145         crosslight_measure_gap
146         nj = STREET_INTERSECTION+1;
147         q = 1;
148         while(q <= l+v-(STREET_INTERSECTION+1))
149             if(ni > 0 || nj == STREET_INTERSECTION+1)
150                 [ni, nj] = crosslight_next_ij(ni, nj, trace_right_direction(
151                     k,l), ...
152                     EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,
153                     EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
154                     EXIT_STRAIGHT_RIGHT);
155             else %we are already in street_outwards
156                 nj = nj+1;
157             end
158             q = q+1;
159         end
160         if (ni > 0) %place car in crossroad
161             street_crossroad_next(ni,nj) = CAR;
162             crossroad_speed_next(ni,nj) = v;
163             crossroad_exit_next(ni,nj) = trace_right_direction(k,l);
164         else
165             street_outwards_next(-ni,nj) = CAR; %ni again minus ->
166             outside_crossroad
167             outwards_speed_next(-ni,nj) = v;
168         end
169     end
170 end
171 %trace_left
172 if (trace_left(k,l) == CAR)
173     gap = crosslight_measure_gap(-k, l,EXIT_LEFT, street_crossroad, ...
174         street_outwards, street_outwards_next, 1, trace_left,
175         trace_left_next, trafficlight(2+3*(k-1),1), ...
176         EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
177         EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
178         EMPTY_STREET);
179     v = schreckenberg(trace_left_speed(k,l),gap,dawdleProb);
180     if (l+v<=STREET_INTERSECTION+1)
181         trace_left_next(k,l+v) = CAR;
182         trace_left_speed_next(k,l+v) = v;
183     else
184         ni = -k;
185         nj = STREET_INTERSECTION+1;
186         q = 1;

```

```

177         while(q <= l+v-(STREET.INTERSECTION+1))
178             if(ni > 0 || nj == STREET.INTERSECTION+1)
179                 [ni, nj] = crosslight_next_ij(ni, nj, EXIT_LEFT, ...
180                     EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,
181                     EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
182                     EXIT_STRAIGHT_RIGHT);
183             else %we are already in street_outwards
184                 nj = nj+1;
185             end
186             q = q+1;
187         end
188         if (ni > 0)
189             street_crossroad_next(ni,nj) = CAR;
190             crossroad_speed_next(ni,nj) = v;
191             crossroad_exit_next(ni,nj) = EXIT_LEFT;
192         else
193             street_outwards_next(-ni,nj) = CAR;
194             outwards_speed_next(-ni,nj) = v;
195         end
196     end
197 end
198
199 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
200 %car in crossroad
201
202 for i = 1:6
203     for j = 1:6
204         if (street_crossroad(i,j) == CAR)
205             gap = crosslight_measure_gap(i, j, crossroad_exit(i,j), street_crossroad,
206                 ...
207                 street_outwards, street_outwards_next, 0, street_inwards,
208                 street_inwards_next, trafficlight(1+3*(k-1),1), ...
209                 EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
210                 EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET.INTERSECTION,
211                 EMPTY.STREET);
212             v = schreckenberg(crossroad_speed(i,j),gap,dawdleProb);
213             ni = i;
214             nj = j;
215             q = 1;
216             while(q <= v)
217                 if(ni > 0)
218                     [ni, nj] = crosslight_next_ij(ni, nj, crossroad_exit(i,j), ...
219                         EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
220                         EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
221                 else %we are already in street_outwards
222                     nj = nj+1;
223                 end
224                 q = q+1;
225             end
226             if (ni > 0)
227                 street_crossroad_next(ni,nj) = CAR;
228                 crossroad_speed_next(ni,nj) = v;
229                 crossroad_exit_next(ni,nj) = crossroad_exit(i,j);
230             else
231                 street_outwards_next(-ni,nj) = CAR;
232                 outwards_speed_next(-ni,nj) = v;

```

```

228         end
229     end
230 end
231 end
232
233 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
234 %car outwards
235
236 for k = 1:4
237     for l = 1:STREET_INTERSECTION
238         %outwards street
239         e = 1;
240         while (e <= 5 && street_outwards(k,l+e) == EMPTY_STREET &&
241             street_outwards_next(k,l+e) == EMPTY_STREET)
242             e = e + 1;
243         end
244         gap = e - 1;
245         v = schreckenberg(outwards_speed(k,l), gap, dawdleProb);
246         if(street_outwards(k,l) == CAR)
247             street_outwards_next(k,l+v) = CAR;
248             outwards_speed_next(k,l+v) = v;
249         end
250     end
251 end
252 end

```

Listing 10: crosslight-measure-gap.m

```

1 function [ gap ] = crosslight_measure_gap(i, j, direction, street_crossroad, ...
2     street_outwards, street_outwards_next, inwards, street_inwards,
3     street_inwards_next, traffic_light, ...
4     EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
5     EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION, EMPTY_STREET)
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 %crosslight_measure_gap this function will measure the gap to the next car
8 %in a crosslight
9 %
10 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
11 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
12 %Fall 2012
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 e = 1;
15 iterate = 1;
16 ni = i;
17 nj = j;
18 while (e <= 5 && iterate)
19     if((ni < 0 && nj == STREET_INTERSECTION+1 && inwards) || ni > 0)
20         [ni, nj] = crosslight_next_ij(ni, nj, direction, ...
21             EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
22             EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
23     else
24         %ni = ni;
25         nj = nj+1;
26     end
27     if(ni > 0)
28         inwards = 0;
29     end
30     e = e + 1;
31     iterate = iterate + 1;
32 end

```

```

27     if (street_crossroad(ni, nj) == EMPTY_STREET)
28         e = e + 1;
29     else
30         iterate = 0;
31     end
32     if ((direction == EXIT_LEFT || direction == EXIT_RIGHT) && e > 2) %limit
33         speed inside the crosssection
34         e = 2;
35         iterate = 0;
36     end
37 else
38     if (inwards)
39         if (nj == STREET_INTERSECTION+1 || nj == STREET_INTERSECTION) %last or
40             second to last field in front of intersection have to wait if
41             traffic light is red
42             if (traffic_light && street_inwards(-ni, nj) == EMPTY_STREET &&
43                 street_inwards_next(-ni, nj) == EMPTY_STREET) %% traffic_light
44                 green and street empty
45                 e = e + 1;
46             else
47                 iterate = 0;
48             end
49         else
50             if (street_inwards(-ni, nj) == EMPTY_STREET && street_inwards_next(-ni
51                 , nj) == EMPTY_STREET)
52                 e = e + 1;
53             else
54                 iterate = 0;
55             end
56         end
57     end
58 end
59 end
60 gap = e - 1;
61
62 end

```

Listing 11: crosslight-next-ij.m

```

1 function [ ni, nj ] = crosslight_next_ij(i, j, direction, EXIT_LEFT, EXIT_RIGHT,
2     EXIT_STRAIGHT_TOP, EXIT_STRAIGHT_LEFT, EXIT_STRAIGHT_BOTTOM, EXIT_STRAIGHT_RIGHT)
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %crosslight_next_ij this function will return the next value for i and j
5 %which a car with a given direction and i j coordinates will have
6 %
7 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
8 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
9 %Fall 2012
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 switch (direction)

```

```

12 | case EXIT_LEFT
13 |     if(i == 1 && j == 3)
14 |         ni = 2;
15 |         nj = 3;
16 |     elseif(i == 2 && j == 3)
17 |         ni = 3;
18 |         nj = 4;
19 |     elseif(i == 3 && j == 4)
20 |         ni = 4;
21 |         nj = 5;
22 |     elseif(i == 4 && j == 5)
23 |         ni = 5;
24 |         nj = 6;
25 |     elseif(i == 5 && j == 6)
26 |         ni = -4;
27 |         nj = 1;
28 |     elseif(i == 4 && j == 1)
29 |         ni = 4;
30 |         nj = 2;
31 |     elseif(i == 4 && j == 2)
32 |         ni = 3;
33 |         nj = 3;
34 |     elseif(i == 3 && j == 3)
35 |         ni = 2;
36 |         nj = 4;
37 |     elseif(i == 2 && j == 4)
38 |         ni = 1;
39 |         nj = 5;
40 |     elseif(i == 1 && j == 5)
41 |         ni = -1;
42 |         nj = 1;
43 |     elseif(i == 6 && j == 4)
44 |         ni = 5;
45 |         nj = 4;
46 |     elseif(i == 5 && j == 4)
47 |         ni = 4;
48 |         nj = 3;
49 |     elseif(i == 4 && j == 3)
50 |         ni = 3;
51 |         nj = 2;
52 |     elseif(i == 3 && j == 2)
53 |         ni = 2;
54 |         nj = 1;
55 |     elseif(i == 2 && j == 1)
56 |         ni = -2;
57 |         nj = 1;
58 |     elseif(i == 3 && j == 6)
59 |         ni = 3;
60 |         nj = 5;
61 |     elseif(i == 3 && j == 5)
62 |         ni = 4;
63 |         nj = 4;
64 |     elseif(i == 4 && j == 4)
65 |         ni = 5;
66 |         nj = 3;
67 |     elseif(i == 5 && j == 3)
68 |         ni = 6;
69 |         nj = 2;

```

```

70     elseif(i == 6 && j == 2)
71         ni = -3;
72         nj = 1;
73     elseif(i < 0)    %here I assume the car is in the last position of the
                        innwards street
74         if(i == -1)
75             ni = 1;
76             nj = 3;
77         elseif(i == -2)
78             ni = 4;
79             nj = 1;
80         elseif(i == -3)
81             ni = 6;
82             nj = 4;
83         elseif(i == -4)
84             ni = 3;
85             nj = 6;
86         end
87     end
88 case EXIT_RIGHT
89     if(i == 1)
90         if(j == 1)
91             ni = -2;
92             nj = 1;
93         else
94             ni = -1;
95             nj = 1;
96         end
97     elseif(i == 6)
98         if(j == 1)
99             ni = -3;
100            nj = 1;
101        else
102            ni = -4;
103            nj = 1;
104        end
105    elseif(i == -1)
106        ni = 1;
107        nj = 1;
108    elseif(i == -2)
109        ni = 6;
110        nj = 1;
111    elseif(i == -3)
112        ni = 6;
113        nj = 6;
114    elseif(i == -4)
115        ni = 1;
116        nj = 6;
117    end
118 case EXIT_STRAIGHT_TOP
119     if(i > 0)
120         nj = j;
121         ni = i-1;
122         if(ni < 1)
123             ni = -EXIT_STRAIGHT_BOTTOM;
124             nj = 1;
125         end
126     elseif(i == -EXIT_STRAIGHT_TOP)    %%check if it comes from BOTTOM

```

```

127         nj = 5;
128         ni = 6;
129     else
130         ni = i;
131         nj = j+1;
132     end
133 case EXIT_STRAIGHT_BOTTOM
134     if(i > 0)
135         nj = j;
136         ni = i+1;
137         if(ni > 6)
138             ni = -EXIT_STRAIGHT_TOP;
139             nj = 1;
140         end
141     elseif(i == -EXIT_STRAIGHT_BOTTOM)
142         nj = 2;
143         ni = 1;
144     else
145         ni = i;
146         nj = j+1;
147     end
148 case EXIT_STRAIGHT_LEFT
149     if(i > 0)
150         nj = j-1;
151         ni = i;
152         if(nj < 1)
153             ni = -2;
154             nj = 1;
155         end
156     elseif(i == -4)
157         nj = 6;
158         ni = 2;
159     else
160         ni = i;
161         nj = j+1;
162     end
163 case EXIT_STRAIGHT_RIGHT
164     if(i > 0)
165         nj = j+1;
166         ni = i;
167         if(nj > 6)
168             ni = -4;
169             nj = 1;
170         end
171     elseif(i == -2)
172         nj = 1;
173         ni = 5;
174     else
175         ni = i;
176         nj = j+1;
177     end
178 otherwise
179     display(direction);
180     display(i);
181     display(j);
182     ni = 0;
183     nj = 0;
184 end

```

185  
186 end

Listing 12: plotresults.m

```
1 function plotresults(d, pd, folder)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %TRAFFIC Simulation of traffic in an city map containing roundabouts and
4 %crossroads.
5 %
6 %This function will plot the precalculated results which are stored in
7 %results/folder where folder is the variable supplied from above
8 %
9 %INPUTS:
10 %D is the car density you want to plot over (should be a vector, else the
11 %plot would only show one point
12 %PD is the pedestrian density
13 %folder is the folder your data is located, this should be an integer!!
14 %
15 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
16 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
17 %Fall 2012
18 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
19 %course "Modelling
20 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
21 %Spring 2010
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 close all;
24
25 %%% runtime measurement - start
26 tic;
27
28 filename = sprintf(' ../results/%g/config.mat', folder);
29 load(filename, 'c', 'pahead');
30
31
32 [c_m,c_n] = size(c);
33 %check if city map is a mix of crossroads and roundaoubts or if it is made up
34 %purely of one or the other
35 mix = not( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 );
36
37 %average flow and distributions for every density supplied
38 avFlow = zeros(max(size(pd)),max(size(d)));
39 avRo = zeros(max(size(pd)),max(size(d)));
40 avCr = zeros(max(size(pd)),max(size(d)));
41 avSpeed = zeros(max(size(pd)),max(size(d)));
42
43 for di=1:max(size(d))
44     for pdi=1:max(size(pd))
45         [config_m,config_n] = size(c);
46         filename = sprintf(' ../results/%g/result_%g x %g_%g_%g.mat', folder, ...
47             config_m, config_n, d(di), pd(pdi));
48         if exist(filename, 'file')
49             disp(filename);
50             load(filename, 'result');
51             disp(result);
52             avFlow(pdi,di) = result(1);
```



```

53         avRo(pdi, di) = result(2);
54         avCr(pdi, di) = result(3);
55         avSpeed(pdi, di) = result(4);
56     end
57 end
58 end
59
60 fig2 = figure(2);
61 %is city map is a mix of roundabout and crossroads, plot distribution
62 if ( mix )
63     %plot relative number of cars at roundabouts and number of cars at
64     %crossroads versus traffic density
65     subplot(2,1,2);
66     plot(d, avRo*100, 'rx', d, avCr*100, 'gx');
67     set(gca, 'FontSize', 16);
68     title('Traffic Distribution');
69     xlabel('traffic density');
70     ylabel('relative number of cars [%]');
71     legend('around roundabouts', 'around crossroads');
72     ylim([0 100]);
73     subplot(2,1,1);
74 end
75
76 %plot traffic flow versus traffic density
77 hold on;
78 % size(avFlow)
79 for i=1:length(pd)
80     pd(i);
81     avFlow_pdi = avFlow(i,:);
82     plot(d, avFlow_pdi, '-x');
83 end
84 % plot(d, avFlow(:, :), '-o')
85 set(gca, 'FontSize', 16);
86 title('Traffic Dynamics');
87 xlabel('traffic density');
88 ylabel('average traffic flow');
89 ylim([0 0.5]);
90
91 fig3 = figure(3);
92 hold on;
93 for i=1:length(d)
94     d(i);
95     avFlow_di = avFlow(:, i);
96     plot(pd, avFlow_di, '-x');
97 end
98 % plot(pd, avFlow(:, :), '-o')
99 set(gca, 'FontSize', 16);
100 title('Traffic Dynamics');
101 xlabel('pedestrian density');
102 ylabel('average traffic flow');
103
104
105 fig4 = figure(4);
106 hold on;
107 for i=1:length(pd)
108     pd(i);
109     avSpeed_pdi = avSpeed(i,:);
110     plot(d, avSpeed_pdi, '-x');

```

```

111 end
112 set(gca,'FontSize',16);
113 title('Traffic Dynamics');
114 xlabel('traffic density');
115 ylabel('average speed');
116
117
118 fig5 = figure(5);
119 hold on;
120 for i=1:length(d)
121     d(i);
122     avSpeed_di = avSpeed(:,i);
123     plot(pd,avSpeed_di, '-x');
124 end
125 set(gca,'FontSize',16);
126 title('Traffic Dynamics');
127 xlabel('pedestrian density');
128 ylabel('average speed');
129
130 fig6 = figure(6);
131 surf(d,pd,avSpeed);
132 title('Traffic Dynamics','FontWeight','bold');
133 xlabel('traffic density');
134 ylabel('pedestrian density');
135 zlabel('average speed');
136
137
138 fig7 = figure(7);
139 surf(d,pd,avFlow);
140 title('Traffic Dynamics','FontWeight','bold');
141 xlabel('traffic density');
142 ylabel('pedestrian density');
143 zlabel('average traffic flow');
144
145
146
147
148 %%% runtime measurement — end
149 toc;
150
151 end

```

Listing 13: plot-map.m

```

1 function [map] = plot_map(street_length, config, car_density, display, ...
2     street_inwards, street_outwards, street_roundabout, street_crossroad, ...
3     BUILDING,EMPTY_STREET, light, trace_left, STREET_INTERSECTION)
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %PLOT_MAP This function plots the map
6 %
7 %This program requires the following subprograms:
8 %none
9 %
10 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
11 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
12 %Fall 2012
13 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling

```

```

14 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
15 %Spring 2010
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 %dimensions of config, how many intersections in x and y direction are there?
19 [config_m, config_n] = size(config);
20
21 %initialize map
22 map = zeros(config_m*(2*street_length+6), config_n*(2*street_length+6));
23 map(1,1)=2;
24
25 %iterate over all intersection
26 for a = 1:config_m
27     for b = 1:config_n
28
29         %define Index starting points for each intersection
30         tI_m = (a - 1) * 4;
31         tI_n = (b - 1) * street_length;
32         mapI_m = (a - 1) * (2 * street_length + 6);
33         mapI_n = (b - 1) * (2 * street_length + 6);
34
35         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36         %write roundabout into map
37
38         %check if intersection is a roundabout
39         if ( config(a,b) == 0 )
40             %define index starting point for this roundabout
41             rI_n = (b - 1) * 12;
42             %write roundabout into map
43             map(mapI_m+street_length+1:mapI_m+street_length+6,...
44                 mapI_n+street_length+1:mapI_n+street_length+6) = ...
45                 [ BUILDING EMPTY_STREET street_roundabout(a, rI_n+4)
46                   street_roundabout(a, rI_n+3) EMPTY_STREET BUILDING;
47                   EMPTY_STREET street_roundabout(a, rI_n+5) EMPTY_STREET EMPTY_STREET
48                   street_roundabout(a, rI_n+2) EMPTY_STREET;
49                   street_roundabout(a, rI_n+6) EMPTY_STREET BUILDING BUILDING
50                   EMPTY_STREET street_roundabout(a, rI_n+1);
51                   street_roundabout(a, rI_n+7) EMPTY_STREET BUILDING BUILDING
52                   EMPTY_STREET street_roundabout(a, rI_n+12);
53                   EMPTY_STREET street_roundabout(a, rI_n+8) EMPTY_STREET EMPTY_STREET
54                   street_roundabout(a, rI_n+11) EMPTY_STREET;
55                   BUILDING EMPTY_STREET street_roundabout(a, rI_n+9) street_roundabout(
56                     a, rI_n+10) EMPTY_STREET BUILDING];
57
58             %write streets into map
59             %normal street
60             for i = 1:street_length-3
61                 map(mapI_m+i, mapI_n+street_length+2) = street_inwards(tI_m+1, tI_n+i)
62                     ; % top, inwards
63                 map(mapI_m+street_length+5, mapI_n+i) = street_inwards(tI_m+2, tI_n+i)
64                     ; % left, inwards
65                 map(mapI_m+2*street_length+7-i, mapI_n+street_length+5) =
66                     street_inwards(tI_m+3, tI_n+i); % bottom, inwards
67                 map(mapI_m+street_length+2, mapI_n+2*street_length+7-i) =
68                     street_inwards(tI_m+4, tI_n+i); % right, inwards
69             end
70         for i = 1+3:street_length

```

```

62     map(mapI_m+street_length+1-i, mapI_n+street_length+5) =
        street_outwards(tI_m+1, tI_n+i); % top, outwards
63     map(mapI_m+street_length+2, mapI_n+street_length+1-i) =
        street_outwards(tI_m+2, tI_n+i); % left, outwards
64     map(mapI_m+street_length+6+i, mapI_n+street_length+2) =
        street_outwards(tI_m+3, tI_n+i); % bottom, outwards
65     map(mapI_m+street_length+5, mapI_n+street_length+6+i) =
        street_outwards(tI_m+4, tI_n+i); % right, outwards
66 end
67 %'last mile'
68 for i = street_length-3+1:street_length
69     map(mapI_m+i, mapI_n+street_length+3) = street_inwards(tI_m+1, tI_n+i)
        ; % top, inwards
70     map(mapI_m+street_length+4, mapI_n+i) = street_inwards(tI_m+2, tI_n+i)
        ; % left, inwards
71     map(mapI_m+2*street_length+7-i, mapI_n+street_length+4) =
        street_inwards(tI_m+3, tI_n+i); % bottom, inwards
72     map(mapI_m+street_length+3, mapI_n+2*street_length+7-i) =
        street_inwards(tI_m+4, tI_n+i); % right, inwards
73 end
74 for i = 1:3
75     map(mapI_m+street_length+1-i, mapI_n+street_length+4) =
        street_outwards(tI_m+1, tI_n+i); % top, outwards
76     map(mapI_m+street_length+3, mapI_n+street_length+1-i) =
        street_outwards(tI_m+2, tI_n+i); % left, outwards
77     map(mapI_m+street_length+6+i, mapI_n+street_length+3) =
        street_outwards(tI_m+3, tI_n+i); % bottom, outwards
78     map(mapI_m+street_length+4, mapI_n+street_length+6+i) =
        street_outwards(tI_m+4, tI_n+i); % right, outwards
79 end
80 %filling fields for optics
81 map(mapI_m+street_length+1-4, mapI_n+street_length+3) = EMPTY_STREET;
        % top, left
82 map(mapI_m+street_length+1-4, mapI_n+street_length+4) = EMPTY_STREET;
        % top, right
83 map(mapI_m+street_length+3, mapI_n+street_length+1-4) = EMPTY_STREET; %
        left, top
84 map(mapI_m+street_length+4, mapI_n+street_length+1-4) = EMPTY_STREET; %
        left, bottom
85 map(mapI_m+street_length+6+4, mapI_n+street_length+3) = EMPTY_STREET; %
        bottom, left
86 map(mapI_m+street_length+6+4, mapI_n+street_length+4) = EMPTY_STREET; %
        bottom, right
87 map(mapI_m+street_length+3, mapI_n+street_length+6+4) = EMPTY_STREET; %
        right, top
88 map(mapI_m+street_length+4, mapI_n+street_length+6+4) = EMPTY_STREET; %
        right, bottom
89 end
90
91 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92 %write crossing into map
93
94 %check if intersection is a crossing with priority to the right
95 if ( config(a,b) == 1 )
96     %define index starting points for this crossroad
97     pI_m = (a - 1) * 6;
98     pI_n = (b - 1) * 6;
99     pIl_n = (b - 1) * 12; % index for light

```

```

100     pIt_m = (a - 1) * 4;    % m-index for trace left
101     pIt_n = (b - 1) * 8;    % n-index for trace left
102     %write crossroad into map
103     map(mapI_m+street_length+1:mapI_m+street_length+6,...
104         mapI_n+street_length+1:mapI_n+street_length+6) = ...
105         street_crossroad(pI_m+1:pI_m+6,pI_n+1:pI_n+6);
106
107     %traffic lights
108     GREEN_LIGHT = 1.3;
109     RED_LIGHT = 1.6;
110     light(light==1) = GREEN_LIGHT;
111     light(light==0) = RED_LIGHT;
112
113     map(mapI_m+street_length-2, mapI_n+street_length+1) = light(a, pIl_n
114         +0*3+3); % top, inwards
115     map(mapI_m+street_length-2, mapI_n+street_length+4) = light(a, pIl_n
116         +0*3+2); % top, trace_left
117     map(mapI_m+street_length-1, mapI_n+street_length+6) = light(a, pIl_n
118         +0*3+1); % top, pedestrians
119
120     map(mapI_m+street_length+1, mapI_n+street_length-1) = light(a, pIl_n
121         +1*3+1); % left, pedestrians
122     map(mapI_m+street_length+3, mapI_n+street_length-2) = light(a, pIl_n
123         +1*3+2); % left, trace_left
124     map(mapI_m+street_length+6, mapI_n+street_length-2) = light(a, pIl_n
125         +1*3+3); % left, inwards
126
127     map(mapI_m+street_length+6+2, mapI_n+street_length+1) = light(a, pIl_n
128         +2*3+1); % bottom, pedestrians
129     map(mapI_m+street_length+6+3, mapI_n+street_length+3) = light(a, pIl_n
130         +2*3+2); % bottom, trace_left
131     map(mapI_m+street_length+6+3, mapI_n+street_length+6) = light(a, pIl_n
132         +2*3+3); % bottom, inwards
133
134     map(mapI_m+street_length+1, mapI_n+street_length+6+3) = light(a, pIl_n
135         +3*3+3); % right, inwards
136     map(mapI_m+street_length+4, mapI_n+street_length+6+3) = light(a, pIl_n
137         +3*3+2); % right, trace_left
138     map(mapI_m+street_length+6, mapI_n+street_length+6+2) = light(a, pIl_n
139         +3*3+1); % right, pedestrians
140
141     %trace left
142     trace_left_length = STREET_INTERSECTION+1;
143     for i = 1:trace_left_length
144         map(mapI_m+street_length+7+trace_left_length-i, mapI_n+street_length
145             +4) = trace_left(pIt_m+3,pIt_n+i); % bottom, trace_left
146         map(mapI_m+street_length+3, mapI_n+street_length+7+trace_left_length-i)
147             = trace_left(pIt_m+4,pIt_n+i); % right, trace_left
148         map(mapI_m+street_length-trace_left_length+i, mapI_n+street_length+3)
149             = trace_left(pIt_m+1,pIt_n+i); % top, trace_left
150         map(mapI_m+street_length+4, mapI_n+street_length-trace_left_length+i)
151             = trace_left(pIt_m+2,pIt_n+i); % left, trace_left
152     end
153
154     %write streets into map
155     for i = 1:street_length
156         map(mapI_m+i, mapI_n+street_length+2) = street_inwards(tI_m+1,tI_n+i)
157         ; % top, inwards

```

```

141         map(mapI_m+street_length+5,mapI_n+i) = street_inwards(tI_m+2,tI_n+i)
142         ; % left , inwards
143         map(mapI_m+2*street_length+7-i ,mapI_n+street_length+5) =
144         street_inwards(tI_m+3,tI_n+i); % bottom , inwards
145         map(mapI_m+street_length+2,mapI_n+2*street_length+7-i) =
146         street_inwards(tI_m+4,tI_n+i); % right , inwards
147         map(mapI_m+street_length+1-i ,mapI_n+street_length+5) =
148         street_outwards(tI_m+1,tI_n+i); % top , outwards
149         map(mapI_m+street_length+2,mapI_n+street_length+1-i) =
150         street_outwards(tI_m+2,tI_n+i); % left , outwards
151         map(mapI_m+street_length+6+i ,mapI_n+street_length+2) =
152         street_outwards(tI_m+3,tI_n+i); % bottom , outwards
153         map(mapI_m+street_length+5,mapI_n+street_length+6+i) =
154         street_outwards(tI_m+4,tI_n+i); % right , outwards
155     end
156 end
157
158 % %illustrate trafic situation (now, not of next time step)
159 % fig1 = figure(1);
160 % imagesc(map);
161 % load('colormap2', 'mycmap')
162 % set(fig1, 'Colormap', mycmap)
163 % titlestring = sprintf('Density = %g',car_density);
164 % title(titlestring);
165 % drawnow;
166 end

```

## References

- [1] Andreas Schadschneider Michael Schreckenberg Elmar Brockfeld, Robert Barlovic. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E, Volume 64s*, 2001. 5, 7
- [2] Bastian Bcheler Tony Wood. Traffic flow comparison of roundabouts and cross-roads. 2010. 6