ETH

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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	First name
Arikan	Marcel
Ego	Nuhro
Kohrt	Ralf

Supervising lecturer

Last name	First name
Donnay	Karsten
Balietti	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). 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

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

Contents

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 6×6-Matrices, so that for every cell information about is there a car, its speed and direction can be stored. Furthermore two 4×8 -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 4×8-Matrix indicates the direction.

4.3 Roundabout

Our implementation of the roundabout consits 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.

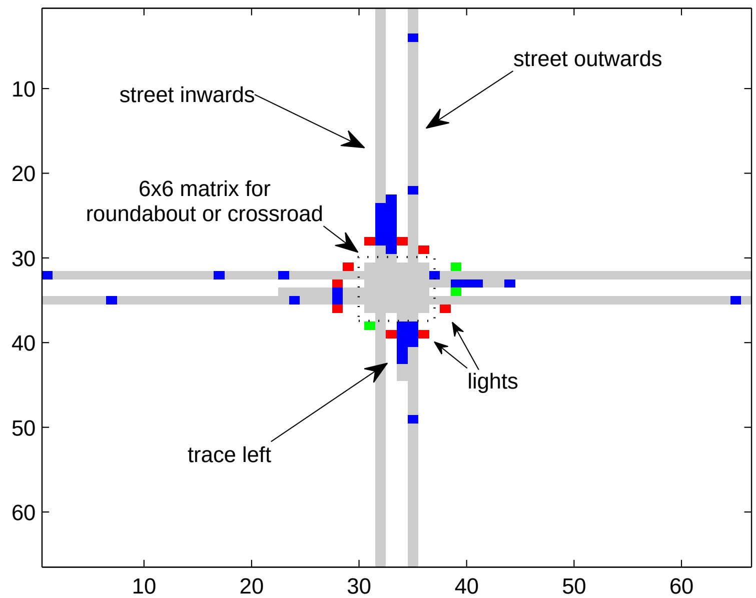


Figure 1: Overview over the implementation of the plot.

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 6×6 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

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

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).

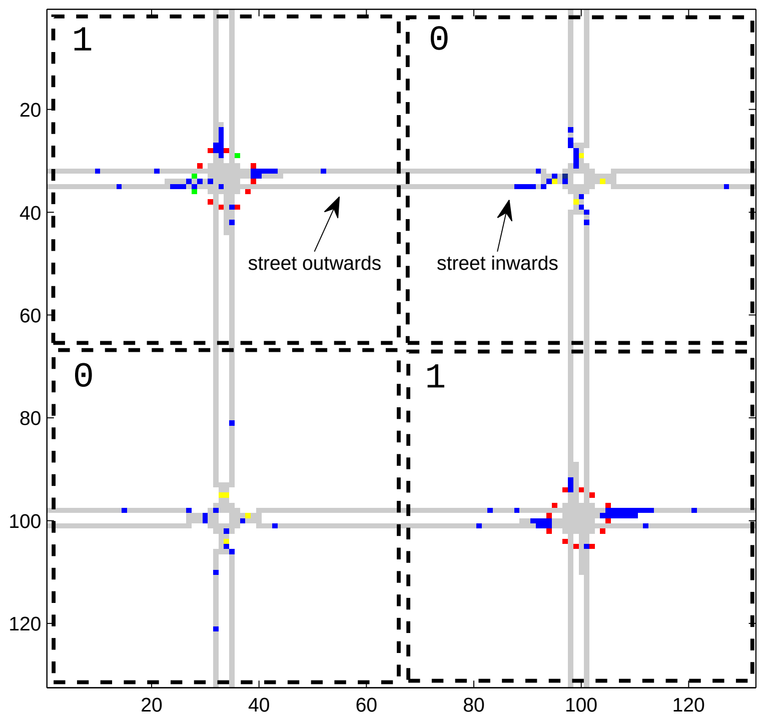


Figure 2: Details about the implementation of the plot in the case of many intersections.

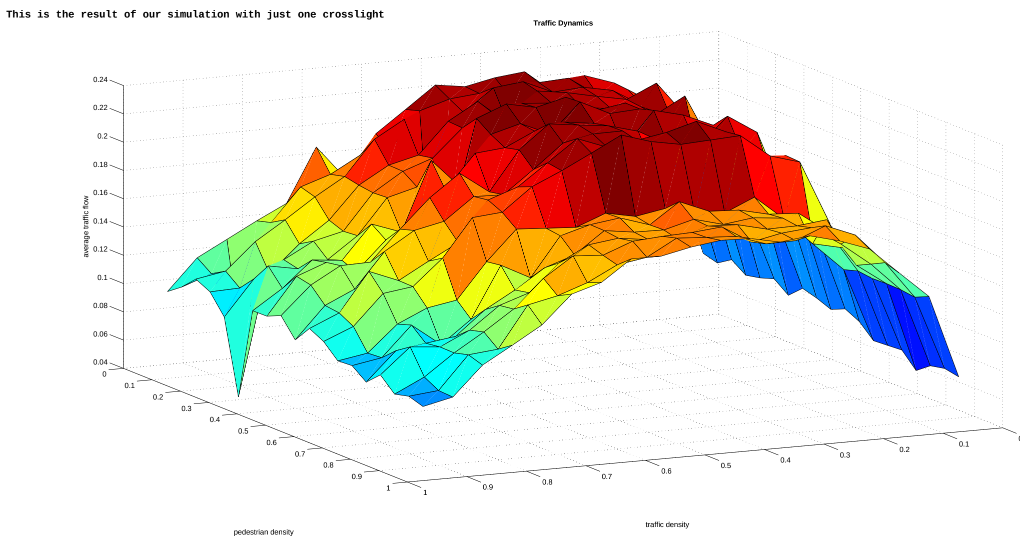
5 Execution and User Instructions

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

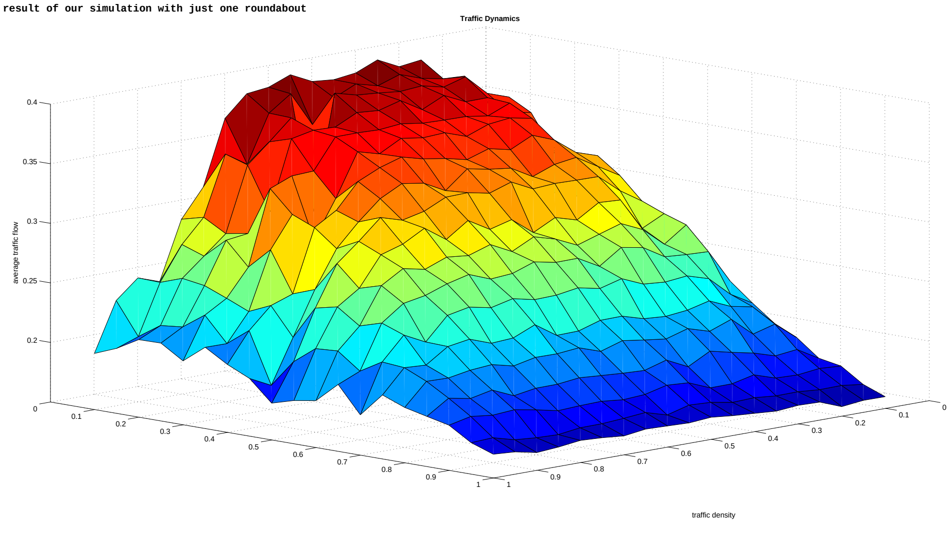
for every iteration is there and In trafficsim. is our main loop for every simulation and here the output graphics, videos etc. will be created.

6 Simulation Results and Discussion



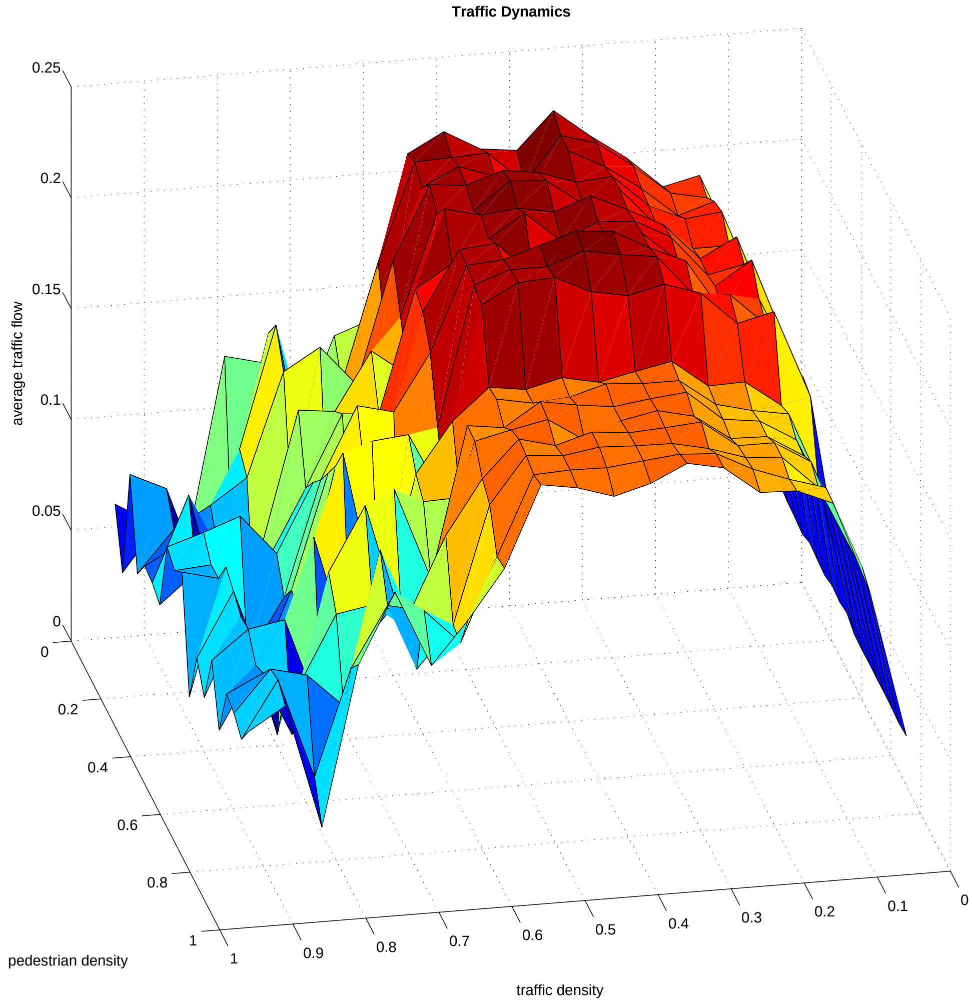
In this plot of one crosslight, one can clearly see the linear increase of traffic flow with increasing car density till 0.25 after that it is more or less constant till it drops linearly at car densities higher than 0.6. One can also see that it does depend only weakly on the pedestrian density, but 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

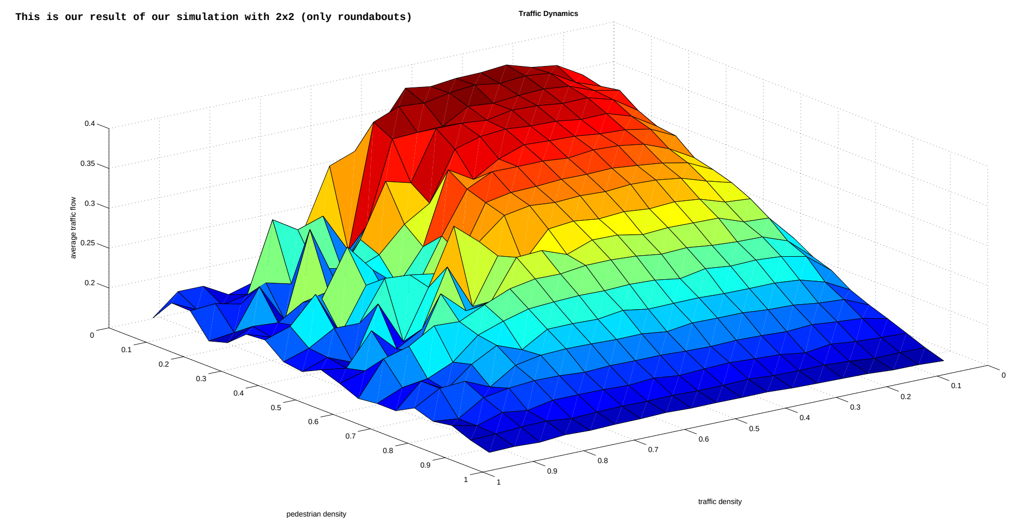


In this plot of one roundabout the linear decrease in flow with increasing pedestrian density is clearly visible. And for a const. pedestrian density one sees also the expected result, which is a linear increase, then for some time a const flow and ending with a linear decrease for increasing car densities. 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.

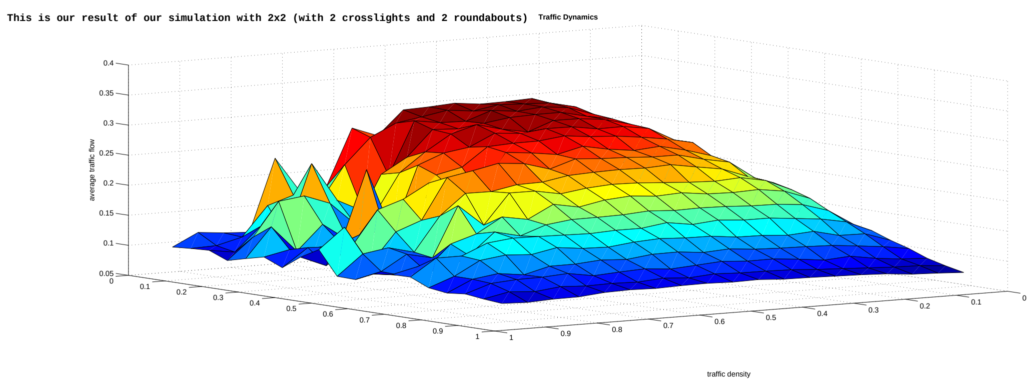
This is the result of our simulation with 2x2 (only trafficlight)



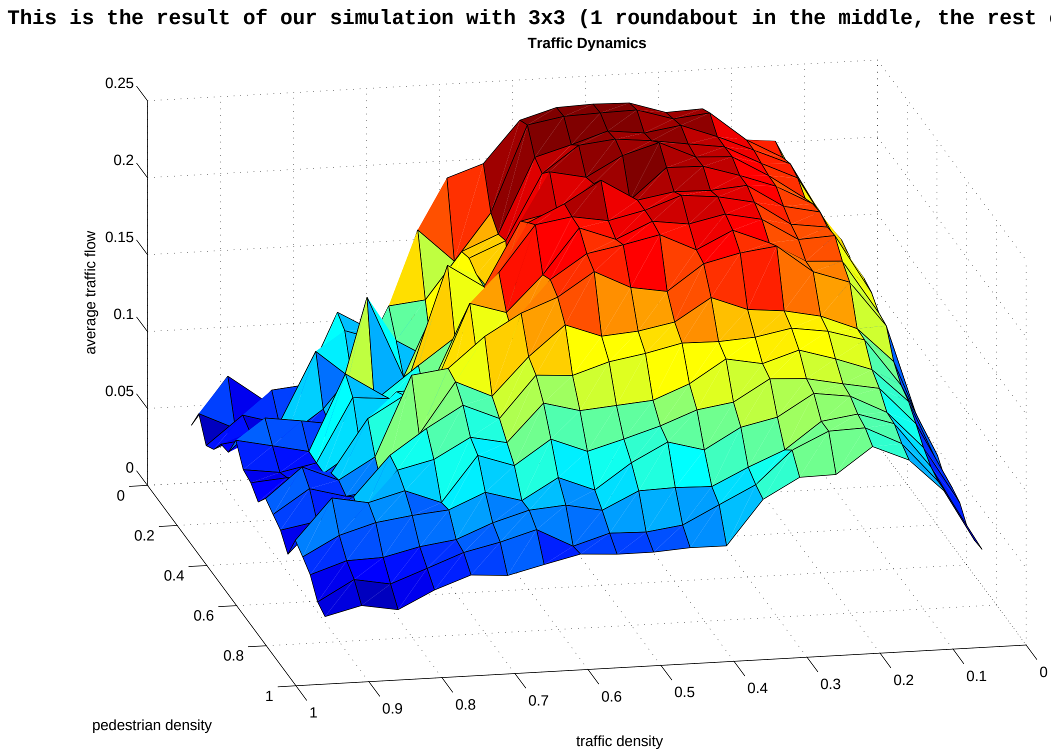
This result does not differ much from the result with just one trafficlight, but one can see that the linear in-/decrease has a higher slope.



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



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 dnsities random processes are getting more important and thus the graph does not look so smooth anymore.



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 trafficlights.

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.

There are many possible modifications to develop in this simulation. As mentioned in the description, an intelligent control of the traffic light could may 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

A Listings

A.1 Matlab Codes

Listing 1: traffic.m

```
function traffic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFIC Simulation of traffic in an city map containing roundabouts and
%crossroads.
%
%This program requires the following subprogams:
%TRAFFICLOOP,TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
%
%
%User will be ask to determine city map,car density, pedestrian density, pahead,
    whether
%simulation is to be displayed or not, if the user wants to create a video
%of the simulation, if the user wants to show the simulaztion in slowmotion
%and if he wants to store the results to plot them later
%
%The city map is entered by supplying a matrix with elements '1' for
%crossroads and '0' for roundabouts.
%
%The density can be a scalar or a vector. If the density is a scalar
%TRAFFIC will run the simulation for all densities given. The elements must
%be in the range of [0,1].
%
%If Users chooses to display simulation (by entering 'y') a figure will
%open showing the animation
%
%After all simulations have finished TRAFFIC plots the average traffic flow
%versus the traffic density. If city map is a mix of crossroad and
%roundabouts the traffic distribution (cars around roundabouts or around
%crossroads) versus traffic density is also plotted.
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;

%promt city road configutation
c = input(['\nenter city map\ngive matrix elements: ', ...
    'Priority to the right (=1) and Roundabout (=0) \n\n', ...
    'i.e. [1 0 0;1 1 0;0 1 1]\n\n']);

%check c
[c_m,c_n] = size(c);
for a = 1:c_m
    for b = 1:c_n
        if ( c(c_m,c_n) ~= 1 && c(c_m,c_n) ~= 0 )
```

```
            disp('Elements must be 0 or 1');
            return
        end
    end
end

%promt traffic density
d = input('\nenter car traffic density: ');
%check d
if ( max(d) > 1 || min(d) < 0)
    disp('density must be in range [0,1]');
    return
end

%prompt probability for car driving ahead
pahead = input('\nenter probability for car driving ahead: ');
%check pahead
if (max(pahead) > 1 || min(pahead) < 0)
    disp('probability must be in range [0,1]');
    return
end

%promt pedestrian density
pd = input('\nenter pedestrian traffic density: ');
%check pd
if ( max(pd) > 1 || min(pd) < 0)
    disp('density must be in range [0,1]');
    return
end

%ask if simulation should be displayed
show = input('\ndisplay simulation graphically? yes (=y) or no (=n) ','s');

%ask if simulation should be in slow_motion
slow_motion = input('\ndisplay slow_motion? yes (=y) or no (=n) ','s');
if (slow_motion == 'n')
    slow_motion = 0;
end

video = input('\ncreate video? yes (=y) or no (=n) ', 's');
if (video == 'n')
    video = 0;
end


store_results = input('\nstore results? yes (=y) or no (=n) ', 's');
if (store_results == 'n')
    store_results = 0;
end
if(store_results)
    folder = input('\nin which folder do you want to store your results?');
    filename = sprintf('../results/%g/config', folder);
    save(filename,'c', 'pahead');
    trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results, folder);
else
    trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results, 'n');
end
```

```

end
```

Listing 2: trafficloop.m

```
function trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results, folder)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFIC Simulation of traffic in an city map containing roundabouts and
%crossroads.
%
%This program requires the following subprogams:
%TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
%
%
%This is the main loop of our simulation
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%
% define global variables
BUILDING = 0;    %the colour for buildings
EMPTY_STREET = 1;
CAR = 0.4;
CAR_NEXT_EXIT = 0.6;    %the colour of a car which will take the next exit
PEDESTRIAN = 0.8;

STREET_INTERSECTION = 7;    %STREET_INTERSECTION specifies the number of elements of the road which will be taken care of by the crossroad/roundabout


if(store_results)
    filename = sprintf('../results/%g/config', folder);
    save(filename,'c', 'pahead');
    result = ones(1,4);
end

%%% runtime measurement - start
tic;

[c_m,c_n] = size(c);
%check if city map is a mix of crossroads and roundaoubts or if it is made up
%purely of one or the other
mix = not( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 );

%average flow and distributions for every density suppied
avFlow = zeros(max(size(pd)),max(size(d)));
avRo = zeros(max(size(pd)),max(size(d)));
avCr = zeros(max(size(pd)),max(size(d)));

```

```
if ( show == 'y' || show == 'n' ) %if show == 'y' -> simulation with graphic output

    for di=1:max(size(d))
        for pdi=1:max(size(pd))
            if(store_results)
                [config_m,config_n] = size(c);
                filename = sprintf('../results/%g/result_(%g x %g)_%g_%g.mat', folder, config_m, config_n, ...
                    d(di), pd(pdi));
                disp(filename);
                [a1,a2,a3,a4] = trafficsim(d(di),pd(pdi),c,show == 'y', ...
                    BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,STREET_INTERSECTION, ...
                    pahead, slow_motion, video);
                result(1) = a1;
                result(2) = a2;
                result(3) = a3;
                result(4) = a4;
                disp(result);
                save(filename,'result');
            else
                [avFlow(pdi,di),avRo(pdi,di),avCr(pdi,di)] = trafficsim(d(di),pd(pdi),c,show == 'y', ...
                    BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,STREET_INTERSECTION, ...
                    pahead, slow_motion, video);
            end
        end
    end

    if(store_results == 0)
        figure(2);
        %is city map is a mix of roundabout and crossroads, plot distribution
        if ( mix )
            %plot relativ number of cars at roundabouts and number of cars at
            %crossroads versus traffic density
            subplot(2,1,2);
            plot(d,avRo*100,'rx',d,avCr*100,'gx');
            set(gca,'FontSize',16);
            title('Traffic Distribution');
            xlabel('traffic density');
            ylabel('relative numeber of cars [%]');
            legend('around roundabouts','around crossroads');
            ylim([0 100]);
            subplot(2,1,1);
        end

        %plot traffic flow versus traffic density
        plot(d,avFlow,'x');
        set(gca,'FontSize',16);
        title('Traffic Dynamics');
        xlabel('traffic density');
        ylabel('average traffic flow');
        %ylim([0 0.5]);
    end
else
    disp('Input must be y or n!');
```

```
end

%%% runtime measurement - end
toc;

end
```

Listing 3: trafficsim.m

```
function [averageFlow,avCaRo,avCaCr,averageSpeed] = trafficsim(car_density, pedestrian_density,config,display, ...
    BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,STREET_INTERSECTION, pahead, slow_motion, video)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFICSIM Simulation of traffic in an city map containing roundabouts and
%crosslights.
%
%Output:
%AVERAGEFLOW, Average traffic flow for given city map and density
%AVCARO, Average amount of cars around roundabouts
%AVCACR, Average amount of cars around crossroads
%averageSpeed, Average speed
%
%INPUT:
%CAR_DENSITY, CAR traffic density
%PEDESTRIAN_DENSITY, pedestrian traffic density
%CONFIG, City map
%DISPlAY, Turn graphics on 'true' or off 'false'
%+defined 'global' variables BUILDING,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN, STREET_INTERSECTION
%PAHEAD, pobability for a car to go ahead
%SLOW_MOTION, show graphics in slow motion?
%VIDEO, generate a video?
%
%This program requires the following subprogams:
%ROUNDABOUT,CROSSLIGHT,CONNECTION,PDESTINATION,MEASURE_GAP,SCHRECKENBERG,PLOT_MAP
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%dawde probability
dawdleProb = 0.2;
%street length (>5)
street_length = 30;
%number of iterations
nIt=1001;

%dimensions of config, how many intersections in x and y direction are
%there?
[config_m,config_n] = size(config);

%initialize matrices for streets heading toward intersections
```

```
street_inwards = ones(4*config_m, street_length*config_n)*EMPTY_STREET;
inwards_speed = zeros(4*config_m, street_length*config_n);
%number of elements in street_inwards
inwards_size = sum(sum(street_inwards));

%initialize matrices for street leading away from intersections
street_outwards = ones(4*config_m, street_length*config_n)*EMPTY_STREET;
outwards_speed = zeros(4*config_m, street_length*config_n);

%initialize matrices for roundabouts
street_roundabout = ones(config_m,12*config_n)*EMPTY_STREET;
roundabout_speed = zeros(config_m,12*config_n);
roundabout_exit = zeros(config_m,12*config_n);

%initialize matrices for crossings
street_crossroad = ones(6*config_m,6*config_n)*EMPTY_STREET;

crossroad_speed = zeros(6 *config_m,6*config_n);
crossroad_exit = zeros(6*config_m,6*config_n);
trace_left=ones(4*config_m,(STREET_INTERSECTION+1)*config_n)*EMPTY_STREET;
trace_left_speed=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
trace_right_direction=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);

%this are the computed gaps from the crossections/roundabouts
inwards_gaps = zeros(config_m, config_n*4);

pedestrian_bucket = zeros(2*config_m,4*config_n);

%initialize flow calculation variables
avSpeedIt = zeros(nIt+1,1);
%counter for cars around crossroads
numCaCrIt = zeros(nIt+1,1);
%counter for cars around crossroads
numCaRoIt = zeros(nIt+1,1);

%distribute cars randomly on streets for starting point
overall_length = sum(sum(street_inwards)) + sum(sum(street_outwards));
numCars = ceil(car_density * overall_length);
q = 1;

while ( q <= numCars )
    w = randi(overall_length,1);
    if ( w <= inwards_size )
        if ( street_inwards(w) == EMPTY_STREET)
            street_inwards(w) = CAR;
            inwards_speed(w) = randi(5,1);
            q = q + 1;
        end
    end
    if ( w > inwards_size )
        if ( street_outwards(w-inwards_size) == EMPTY_STREET)
            street_outwards(w-inwards_size) = CAR;
            outwards_speed(w-inwards_size) = randi(5,1);
            q = q +1 ;
        end
    end
end

```

```
street_roundabout_next = ones(config_m,12*config_n)*EMPTY_STREET;
roundabout_speed_next = zeros(config_m,12*config_n);
street_crossroad_next = ones(6*config_m,6*config_n)*EMPTY_STREET;
crossroad_speed_next = ones(6*config_m,6*config_n);
crossroad_exit_next = zeros(6*config_m,6*config_n);

light=zeros(config_m, 12*config_n);      %to display light signalisation

%variables for traffic light control
switchtime = 3;      %time to change signalement (yellow phase)
ligthlength = 30;    %time for staying in same signalement phase
aheadphase = ceil((ligthlength*pahead)/switchtime);     %time to keep green phase ahead
turnphase = ceil((ligthlength*(1-pahead)/2)/switchtime);    %time to keep green phase when turning
totalphase = 6 + 2*aheadphase + 4*turnphase;    %to reset the phase
count =0;   %counter
phase=0;
traveltime = 15+105*car_density;   %time a car needs from one intersection to the next

%figure and video
if (display)
    %figure for map plotting
    fig1 = figure(1);
    load('colormaps/colormap4', 'mycmap');
    set(fig1, 'Colormap', mycmap);
    titlestring = sprintf('Density = %g',car_density);

    %create video
    if (video)
        filename = sprintf('videos/video_(%g x %g)_%g_%g.avi', config_m, config_n, ...
            car_density, pedestrian_density);
        vidObj = VideoWriter(filename);
        open(vidObj);
    end
end

%iterate over time
for time = 1:nIt+1

    %clear values for next step
    street_inwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
    inwards_speed_next = zeros(4*config_m,street_length*config_n);
    street_outwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
    outwards_speed_next = zeros(4*config_m,street_length*config_n);
    trace_left_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
    trace_left_speed_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
    trace_right_direction_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);


    %calculate taffic light phase
    if (count == switchtime)
        if (phase == totalphase+1)
            phase = 0;
        end
```

```
        phase = phase+1;
        count = 0;
    else
        count = count +1;
    end

    %iterate over all intersection
    for a = 1:config_m
        for b = 1:config_n

            %define Index starting points for each intersection
            tI_m = (a - 1) * 4;
            tI_n = (b - 1) * street_length;

            %positions outside intersections
            %for every intersection iterate along streets
            for c = tI_m + 1:tI_m +4
                for d = tI_n + 1:tI_n+street_length

                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %streets to intersections

                    %deal with the STREET_INTERSECTION positions directly in front of intersection
                    %separately later
                    if ( d-tI_n < street_length-STREET_INTERSECTION)
                        %if there is a car in this position, apply
                        %NS-Model
                        if ( street_inwards(c,d) == CAR )
                            %Nagel-Schreckenberg-Model
                            gap = measure_gap(street_inwards, street_outwards, street_length, a, b, c, d, 1, ...
                                inwards_gaps(a,(b - 1) *4+c-tI_m), config_m, config_n, EMPTY_STREET,STREET_INTERSECTION);
                            v = schreckenberg(inwards_speed(c,d), gap, dawdleProb);

                            %NS 4. step: drive, move cars tspeed(c,d) cells
                            %forward
                            %new position
                            street_inwards_next(c,d+v) = CAR;
                            inwards_speed_next(c,d+v) = v;
                        end
                    end

                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %street from intersections

                    %deal with the STREET_INTERSECTION positions directly after the intersection
                    %separately later
                    if ( d-tI_n > STREET_INTERSECTION)
                        if ( street_outwards(c,d) == CAR )
                            %Nagel-Schreckenberg-Model
                            gap = measure_gap(street_inwards, street_outwards, street_length, a, b, c, d, 0, 0, ...
                                config_m, config_n, EMPTY_STREET,STREET_INTERSECTION);
                            v = schreckenberg(outwards_speed(c,d), gap, dawdleProb);
```

```

                                %NS 4. step: drive, move cars fspeed(c,d) cells
                                %forward
                                %if new position is off this street, connect
                                %streets
                                if ( d + v > b * street_length )
                                    %position in new street
                                    hhh =  d + v - b * street_length;
                                    %connect next street
                                    [ec,ed] = connection(a,b,c,hhh, ...
                                        config_m,config_n,street_length);
                                    street_inwards_next(ec,ed) = CAR;
                                    inwards_speed_next(ec,ed) = v;
                                else
                                    street_outwards_next(c,d+v) = CAR;
                                    outwards_speed_next(c,d+v) = v;
                                end
                            end
                        end
                    end
                end

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %roundabouts

                %check if intersection is a roundabout
                if  ( config(a,b) == 0 )
                    %define index strating point for this roundabout
                    rI_n = (b - 1) * 12;

                    %do roundabout calculations for this roundabout and time
                    %step
                    %call ROUNDABOUT
                    [street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                        inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                        street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                        outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                        street_roundabout_next(a,rI_n+1:rI_n+12), ...
                        roundabout_speed_next(a,rI_n+1:rI_n+12), ...
                        roundabout_exit(a,rI_n+1:rI_n+12), ...
                        pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4), ...
                        inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4) ] = ...
                        roundabout(street_inwards(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                        inwards_speed(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                        street_outwards(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                        outwards_speed(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                        street_roundabout(a,rI_n+1:rI_n+12), ...
                        roundabout_exit(a,rI_n+1:rI_n+12), ...
                        pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4)
```

```
                    , ...
                    inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), dawdleProb, ...
                    pedestrian_density, ...
                    street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6),EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,STREET_INTERSECTION,pahead);

                %add cars around this crossroad in this time step to
                %counter for cars around crossroads
                for v = tI_m+1:tI_m+4
                    for w = tI_n+1:tI_n+street_length
                        if ( street_inwards(v,w) ~= 1 )
                            numCaRoIt(time) = numCaRoIt(time) + 1;
                        end
                        if ( street_outwards(v,w) ~= 1 )
                            numCaRoIt(time) = numCaRoIt(time) + 1;
                        end
                    end
                end
                for y = rI_n+1:rI_n+12
                    if ( street_roundabout(a,y) ~= 1 )
                        numCaRoIt(time) = numCaRoIt(time) + 1;
                    end
                end

            end

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %crossroads

            %check if intersection is a crossing with priority to the right
            if ( config(a,b) == 1 )
                %define index starting points for this crossraod
                pI_m = (a - 1) * 6;
                pI_n = (b - 1) * 6;

                %define trace index for this crossraod
                traceI_m = (a - 1) * 4;
                traceI_n = (b - 1) * 8;
                %define light index for this crossroad
                lightI_m = (a - 1) ;
                lightI_n = (b - 1) * 12;
                %calculate local off set of phase for different crossroads
                localphase = phase+(a+b-2)*traveltime;
                while (localphase > totalphase)        %reset localphase if necessary
                    localphase = localphase - totalphase;
                end
                %do crossroad calculations for this crossroad and time step
                %call CROSSROAD
                [street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
```

```
                    inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    street_crossroad_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                    crossroad_speed_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                    crossroad_exit_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                    pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4), ...
                    inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), ...
                    trace_left_next(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n+8), ...
                    trace_left_speed_next(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n+8), ...
                    trace_right_direction_next(traceI_m+1:traceI_m+4,traceI_n+1:traceI_n+8), ...
                    light(lightI_m+1,lightI_n+1:lightI_n+12)] ...
                    = crosslight(street_inwards(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    inwards_speed(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    street_outwards(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    outwards_speed(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    street_crossroad(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                    crossroad_speed(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                    crossroad_exit(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ....
                    pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4), ...
                    inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), dawdleProb, ...
                    pedestrian_density, ...
                    street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-STREET_INTERSECTION:tI_n+street_length), ...
                    street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6), ...
                    outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+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), ...
                    localphase, aheadphase, turnphase);


                %add cars around this crossroad in this time step to
                %counter for cars around crossroad
                for v = tI_m+1:tI_m+4
                    for w = tI_n+1:tI_n+street_length
                        if ( street_inwards(v,w) ~= 1 )
                            numCaCrIt(time) = numCaCrIt(time) + 1;
                        end
                        if ( street_outwards(v,w) ~= 1 )
```

```
                            numCaCrIt(time) = numCaCrIt(time) + 1;
                        end
                    end
                end
                for x = pI_m+1:pI_m+6
                    for y = pI_n+1:pI_n+6
                        if ( street_crossroad(x,y) ~= 0 )
                            numCaCrIt(time) = numCaCrIt(time) + 1;
                        end
                    end
                end

            end

        end
    end

    %calculate average velosity per time step
    avSpeedIt(time) = ( sum(sum(inwards_speed)) + sum(sum(outwards_speed)) + ...
        sum(sum(roundabout_speed)) + sum(sum(crossroad_speed)) ) / numCars;

    %plot the map in this timestep into the figure
    if (display)
        map = plot_map(street_length, config, car_density, display, ...
            street_inwards, street_outwards, street_roundabout, street_crossroad,
                ...
            BUILDING,EMPTY_STREET, light, trace_left, STREET_INTERSECTION);
        %illustrate trafic situation (now, not of next time step)
        imagesc(map);
        title(titlestring, 'FontWeight','bold');
        drawnow;
        if (video)
            % get the current frame
            currFrame = getframe(fig1);
            % add the current frame
            writeVideo(vidObj,currFrame);
        end
    end

    if (slow_motion)
        pause(1);
    end

    %move on time step on
    street_inwards = street_inwards_next;
    inwards_speed = inwards_speed_next;
    street_outwards = street_outwards_next;
    outwards_speed = outwards_speed_next;
    street_roundabout = street_roundabout_next;
    roundabout_speed = roundabout_speed_next;
    street_crossroad = street_crossroad_next;
    crossroad_speed = crossroad_speed_next;
    crossroad_exit = crossroad_exit_next;
    trace_left = trace_left_next;
    trace_left_speed = trace_left_speed_next;
    trace_right_direction = trace_right_direction_next;

end
```

```

if (video)
    close(vidObj);
end

%overall average velocity
averageSpeed = sum(avSpeedIt) / max(size(avSpeedIt));
%overall average flow
averageFlow = car_density * averageSpeed;

%average relative amount of cars around roundabouts
avCaRo = sum(numCaRoIt) / ( max(size(numCaRoIt)) * numCars );
%average relative amount of cars around crossroads
avCaCr = sum(numCaCrIt) / ( max(size(numCaCrIt)) * numCars );

end
```

Listing 4: measure-gap.m

```
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
    )
%MEASURE_GAP this measures the gap to the next car
%   how big is gap (to car ahead or intersection)?


e = 0;
iterate = 1;
while (iterate )    %iterate while iterate is 1
    if(inwards)
        e = e + 1;
        iterate = e <= 5 && d + e <= b * street_length - STREET_INTERSECTION +
            inwards_gap && ...
        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
    else
        e = e + 1;
        %if gap is bigger than distance to edge,connect
        %steets
        if ( d + e > b * street_length)
            %testing position in new street
            hh = d + e - b * street_length;
            %connect to next street
            [ec,ed]=connection(a,b,c,hh, ...
                config_m,config_n,street_length);
            while ( street_inwards(ec,ed) == EMPTY_STREET && e <= 5 )
                e = e + 1;
                %testing position in new street
                hh = d + e - b * street_length;
                %connect to next street
                [ec,ed]=connection(a,b,c,hh, ...
                    config_m,config_n,street_length);
            end
            iterate = 0;
        else
            iterate = e <= 5 && street_outwards(c,d+e) == EMPTY_STREET;     %% <= 4 b
                .c. it'll be 5 after this loop
```

```
        end
    end
end
gap = e - 1;

end
```

Listing 5: connection.m

```
function [cNew,dNew] = connection(aOld,bOld,cOld,posNew,m,n,length)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CONNECTION Deside to which street a certain street connects to
%
%INPUT:
%AOLD column index of intersection
%BOLD, row index of intersection
%COLD, column index in t of old position
%posNEW, position in new street
%M, number of columns in city map
%N, number of rows in city map
%LENGTH, Length of a street
%
%OUTPUT:
%CNEW, Column index in t of new position
%DNEW, Row index in t of new position
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%street heading up from intersection
if ( mod(cOld,4) == 1 )
    %if there is a intersections above, connect to it
    if ( aOld > 1)
        cNew = (aOld - 2) * 4 + 3;
        dNew = (bOld - 1) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = (m - 1) * 4 + 3;
        dNew = (bOld - 1) * length + posNew;
    end
end

%street heading left from intersection
if ( mod(cOld,4) == 2 )
    %if there is a intersection to the left, connect to it
    if ( bOld > 1 )
        cNew = aOld * 4;
        dNew = (bOld - 2) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = aOld * 4;
        dNew = (n - 1) * length + posNew;
```

```
    end
end

%street heading down from intersection
if ( mod(cOld,4) == 3 )
    %if there is a intersection below, connect to it
    if ( aOld < m )
        cNew = aOld * 4 + 1;
        dNew = (bOld - 1) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = 1;
        dNew = (bOld - 1) * length + posNew;
    end
end

%street heading right from intersection
if ( mod(cOld,4) == 0 )
    %if there is a intersection to the right, connect to it
    if ( bOld < n )
        cNew = (aOld - 1) * 4 + 2;
        dNew = bOld * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = (aOld - 1) * 4 + 2;
        dNew = posNew;
    end
end
```

Listing 6: pdestination.m

```
function [pfirst] = pdestination
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PDESTINATION Deside where a car is going
%
%OUTPUT:
%PFIRST = 0.1 car turns right
%       = 0.4 car goes straight ahead
%       = 0.7 car turns left
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%decide which direction car is going
u = randi(12,1);
%probabilty 6/12 car goes straight ahead
if ( u <= 6 )
  pfirst = 0.4;
end
%probabilty 3/12 car turns right
if ( u >= 7 && u <= 9 )
  %indicate right
```

```
  pfirst = 0.7;
end
%probabilty 3/12 car turns left
if ( u >= 10 && u <= 12 )
  pfirst = 0.1;
end

end
```

Listing 7: schreckenberg.m

```
function [ speed ] = schreckenberg(speed, gap, dawdleProb)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SCHRECKENBERG Nagel-Schreckenberg-Model
%
%OUTPUT: new speed of the selected car
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%NS 1. step: increase velocity if < 5
if ( speed < 5)
    speed = speed + 1;
end

%NS 2. step: adapt speed to gap
%reduce speed if gap is too small
if ( speed > gap )
    speed = gap;
end

%NS 3. step: dawdle
if ( rand < dawdleProb && speed ~= 0 )
    speed = speed - 1;
end

end
```

Listing 8: roundabout.m

```
function [street_inwards_next, ...
    inwards_speed_next, ...
    street_outwards_next, ...
    outwards_speed_next, ...
    street_roundabout_local_next, ...
    roundabout_speedlocal_next, ...
    roundabout_exit_local_next, ...
    pedestrian_bucket, inwards_gaps] ...
    = roundabout(street_inwards, ...
    inwards_speed, ...
    street_outwards, ...
    outwards_speed, ...
    street_roundabout, ...
    roundabout_exit ,pedestrian_bucket, ...
    inwards_gaps, dawdleProb, ...
```

```
    pedestrian_density , ...
    street_inwards_next , ...
    inwards_speed_next , ...
    street_outwards_next ,...
    outwards_speed_next ,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
        STREET_INTERSECTION, pahead)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ROUNDABOUT Calculation of update for a certain roundabout, density and
%time step
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear local next variables
street_roundabout_local_next = ones(1,12)*EMPTY_STREET;
roundabout_speedlocal_next = zeros(1,12);
roundabout_exit_local_next = zeros(1,12);

temp_roundabout_pedestrian_bucket = pedestrian_bucket;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in front of roundabout

for k = 1:4
    if ( street_inwards(k,STREET_INTERSECTION+1) == CAR )
        %entering roundabout with velocity 1 when possible
        %roundabout position index
        iR = mod(3*k+1,12);
        % enter roundabout if car at position k*3 is about to exit and
        % there is no car at position 3*k+1
        if ( roundabout_exit(k*3) <= 1 && street_roundabout(iR) == EMPTY_STREET )
            %enter roundabout
            %decide which exit car is going to take
            u = rand(1);
            %if it takes 1. exit
            if ( u <= (0.95/2*(1-pahead)))
                roundabout_exit_local_next(iR) = 1;
                %indicate
                street_roundabout_local_next(iR) = CAR_NEXT_EXIT;
                roundabout_speedlocal_next(iR) = 1;
            %if it takes 2. exit
            elseif ( u <= (0.95/2*(1+pahead)))
                roundabout_exit_local_next(iR) = 2;
                street_roundabout_local_next(iR) = CAR;
                roundabout_speedlocal_next(iR) = 1;
            %if it takes 3. exit
            elseif ( u <= 0.95 )
                roundabout_exit_local_next(iR) = 3;
                street_roundabout_local_next(iR) = CAR;
                roundabout_speedlocal_next(iR) = 1;
            %if it takes 4. exit (turns around)
            else
```

```
                    roundabout_exit_local_next(iR) = 4;
                    street_roundabout_local_next(iR) = CAR;
                    roundabout_speedlocal_next(iR) = 1;
                end

          %car waiting in front of roundabout
          else
              street_inwards_next(k,STREET_INTERSECTION+1) = street_inwards(k,
                  STREET_INTERSECTION+1);
              inwards_speed_next(k,STREET_INTERSECTION+1) = 0;
          end
     end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%pedestrians


for k = 1:4
     r = rand(1);
     if (( street_inwards(k,STREET_INTERSECTION) == EMPTY_STREET || street_inwards(k,
          STREET_INTERSECTION) == PEDESTRIAN) && ...
              (r <= pedestrian_density || pedestrian_bucket(1,k) > 0))
          street_inwards_next(k,STREET_INTERSECTION) = PEDESTRIAN;
          inwards_speed_next(k,STREET_INTERSECTION) = 0;
          if(r <= pedestrian_density)
              temp_roundabout_pedestrian_bucket(2,k) = 1;
          end
          if(pedestrian_bucket(1,k) > 0)
              temp_roundabout_pedestrian_bucket(1,k) = 0;
          end
     end
     r = rand(1);
     if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN
          ) && ...
              (r <= pedestrian_density || pedestrian_bucket(2,k) > 0))
          street_outwards_next(k,2) = PEDESTRIAN;
          outwards_speed_next(k,2) = 0;
          if(r <= pedestrian_density)
              temp_roundabout_pedestrian_bucket(1,k) = 1;
          end
          if(pedestrian_bucket(2,k) > 0)
              temp_roundabout_pedestrian_bucket(2,k) = 0;
          end
     end
     if(0)
          if (( street_roundabout(k*3-1) == EMPTY_STREET || street_roundabout(k*3-1)
              == PEDESTRIAN) && roundabout_pedestrian_bucket(k) > 0)
              street_roundabout_local_next(k*3-1) = PEDESTRIAN;
              roundabout_speedlocal_next(k*3-1) = 0;
              roundabout_exit_local_next(k*3-1) = 0;
              if(roundabout_pedestrian_bucket(k) >= 1)
                   roundabout_pedestrian_bucket(k) = roundabout_pedestrian_bucket(k)-1;
              end
          elseif ( street_inwards(k,2) == PEDESTRIAN && roundabout_pedestrian_bucket(k
              ) == 0)
              street_roundabout_local_next(k*3-1) = EMPTY_STREET;
              roundabout_speedlocal_next(k*3-1) = 0;
```

```
            roundabout_exit_local_next(k*3-1) = 0;
        end
    end
end

pedestrian_bucket = temp_roundabout_pedestrian_bucket;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car outside roundabout



for k = 1:4
    for j = 1:STREET_INTERSECTION
        e = 1;
        while (e <= 5 && ((street_outwards(k,j+e) == EMPTY_STREET && street_outwards_next(k,j+e) == EMPTY_STREET)  || ...
                (street_outwards(k,j+e) == PEDESTRIAN && street_outwards_next(k,j+e) == EMPTY_STREET) ))
            e = e + 1;
        end
        gap = e - 1;
        v = schreckenberg(outwards_speed(k,j), gap, dawdleProb);
        if(street_outwards(k,j) == CAR)
            if ( (street_outwards(k,j+v) == EMPTY_STREET && street_outwards_next(k,j+v) == EMPTY_STREET) || ...
                    (street_outwards(k,j+v) == PEDESTRIAN && street_outwards_next(k,j+v) == EMPTY_STREET) )
                street_outwards_next(k,j+v) = CAR;
                outwards_speed_next(k,j+v) = v;
            else
                street_outwards_next(k,j) = CAR;
                outwards_speed_next(k,j) = 0;
            end
        end
        e = 1;
        while (e <= 5 && j + e <= STREET_INTERSECTION+1 && ((street_inwards(k,j+e) == EMPTY_STREET && street_inwards_next(k,j+e) == EMPTY_STREET) || ...
                ( street_inwards(k,j+e) == PEDESTRIAN && street_inwards_next(k,j+e) == EMPTY_STREET) ))
            e = e + 1;
        end
        gap = e - 1;
        v = schreckenberg(inwards_speed(k,j), gap, dawdleProb);
        if(j == 1)
            inwards_gaps(1,k) = gap;
        end
        if(street_inwards(k,j) == CAR)
            if ( ( street_inwards(k,j+v) == EMPTY_STREET && street_inwards_next(k,j+v) == EMPTY_STREET) || ...
                    ( street_inwards(k,j+v) == PEDESTRIAN && street_inwards_next(k,j+v) == EMPTY_STREET) )
                street_inwards_next(k,j+v) = CAR;
                inwards_speed_next(k,j+v) = v;
            else
                street_inwards_next(k,j) = CAR;
                inwards_speed_next(k,j) = 0;
            end
        end
```

```
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in roundabout

for j = 1:12
    if ( street_roundabout(j) == CAR || street_roundabout(j) == CAR_NEXT_EXIT )

        %cars in roundabout not at an exit
        if (mod(j,3) ~= 0 )
            %if space free, move one forward
            if ( street_roundabout(j+1) == EMPTY_STREET && street_roundabout_local_next(j+1) == EMPTY_STREET)
                %take new position
                street_roundabout_local_next(j+1) = street_roundabout(j);
                roundabout_speedlocal_next(j+1) = 1;
                roundabout_exit_local_next(j+1) = roundabout_exit(j);
            %if no space free, stay
            else
                street_roundabout_local_next(j) = street_roundabout(j);
                roundabout_speedlocal_next(j) = 0;
                roundabout_exit_local_next(j) = roundabout_exit(j);
            end

        %car at an exit
        else

            %if car is at its exit
            if ( roundabout_exit(j) == 1 )
                %if space free, leave roundabout
                if ( street_outwards(j/3,1) == EMPTY_STREET )
                    street_outwards_next(j/3,1) = CAR;
                    outwards_speed_next(j/3,1) = 1;
                %if no space free, stay
                else
                    street_roundabout_local_next(j) = street_roundabout(j);
                    roundabout_speedlocal_next(j) = 0;
                    roundabout_exit_local_next(j) = roundabout_exit(j);
                end

            %car at an exit but not the one its taking
            else
                %connect street_roundabout(12) with street_roundabout(1)
                if (j == 12 )
                    j1 = 1;
                else
                    j1 = j+1;
                end
                %if space free, move one forward and decrease exit
                %counter
                if ( street_roundabout(j1) == EMPTY_STREET )
                    %decrease exit by one
                    roundabout_exit_local_next(j1) = roundabout_exit(j) - 1;
                    roundabout_speedlocal_next(j1) = 1;
                    if ( roundabout_exit_local_next(j1) == 1 )
```

```
                        %indicate
                        street_roundabout_local_next(j1) = CAR_NEXT_EXIT;
                    else
                        street_roundabout_local_next(j1) = CAR;
                    end
                %if no space free, stay
                else
                    street_roundabout_local_next(j) = street_roundabout(j);
                    roundabout_speedlocal_next(j) = 0;
                    roundabout_exit_local_next(j) = roundabout_exit(j);
                end
            end
        end
    end
end

end
```

Listing 9: crosslight.m

```
function [street_inwards_next, ...
    inwards_speed_next, ...
    street_outwards_next, ...
    outwards_speed_next, ...
    street_crossroad_next, ...
    crossroad_speed_next, ...
    crossroad_exit_next, ...
    pedestrian_bucket, inwards_gaps, ...
    trace_left_next, trace_left_speed_next, trace_right_direction_next, trafficlight
        ] ...
    = crosslight(street_inwards, ...
    inwards_speed, ...
    street_outwards, ...
    outwards_speed, ...
    street_crossroad, ...
    crossroad_speed, ...
    crossroad_exit, pedestrian_bucket, ...
    inwards_gaps, dawdleProb, ...
    pedestrian_density, ...
    street_inwards_next, ...
    inwards_speed_next, ...
    street_outwards_next, ...
    outwards_speed_next,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
        STREET_INTERSECTION, ...
    pahead, trace_left, trace_left_speed, trace_right_direction, ...
    localphase, aheadphase, turnphase)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CROSSLIGHT Calculation of update for a certain crossroad, including lane for cars
    turning left
%and 8 cells of street_inwards and street_outwards
%
%additional Output:
%inwards_gaps: gap to place car from street_inwards when entering the 8 cells in
    front of
%crossroad calculated by crosslight
%trafficlight: signalisation for every lane to plot traffic lights in
%display
%
```

```
%additional INPUT:
%pahead: probabiltiy for car driving ahead
%localphase: signalisation phase for this crossroad
%aheadphase, turnphaae: relative time for signalisation staying green for
%car turning or driving ahead
%
%This program requires the following subprogams:
%settrafficlight
%crosslight_measure_gap
%crosslight_next_ij
%schreckenberg
%pdestination
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NO_EXIT_YET = 0;
EXIT_LEFT = 5;
EXIT_RIGHT = 6;
EXIT_STRAIGHT_TOP = 3;
EXIT_STRAIGHT_LEFT = 4;
EXIT_STRAIGHT_BOTTOM = 1;
EXIT_STRAIGHT_RIGHT = 2;

%clear local next variables
street_crossroad_next = ones(6,6)*EMPTY_STREET;
crossroad_speed_next = zeros(6,6);
crossroad_exit_next = zeros(6,6);
trace_left_next = ones(4,8)*EMPTY_STREET;
trace_left_speed_next = zeros(4,8);
trace_right_direction_next = ones(4,8)*NO_EXIT_YET;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%set traffic light
%trafficlight = zeros(12,1) 0=red for car and pedestrians
trafficlight = settrafficlight(localphase, aheadphase, turnphase, pedestrian_density);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%place randomly pedestrians
for k = 1:4
    if (rand(1) <= pedestrian_density )
        pedestrian_bucket(2,k) = 1;     %use bucket to make sure that pedestrians accumulate during redphase
    end
    if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN ) && ...
            pedestrian_bucket(2,k) > 0 && trafficlight(1+(k-1)*3,1)==1 )
        street_outwards_next(k,2) = PEDESTRIAN;
        outwards_speed_next(k,2) = 0;
        pedestrian_bucket(2,k) = 0;
    elseif ( street_outwards(k,2) == PEDESTRIAN)
        street_outwards_next(k,2) = EMPTY_STREET;
```

```
        outwards_speed_next(k,2) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in front of crossroad and initializing direction

for k = 1:4
    for l=1:STREET_INTERSECTION+1
        %initializing randomly directions
        if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==NO_EXIT_YET)
            u=rand(1);
            %if it goes left
            if ( u < ((1-pahead)/2))
                trace_right_direction(k,l) = EXIT_LEFT;
                %if it goes ahead
            elseif ( u <= ((1+pahead)/2))
                trace_right_direction(k,l) = k;

                %if it goes right
            else
                trace_right_direction(k,l) = EXIT_RIGHT;

            end
        end

        %take cars with EXIT_LEFT waiting into trace_left if space is free
        if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==EXIT_LEFT)
            if(trace_left(k,1) == EMPTY_STREET)
                trace_left_next(k,1) = CAR;
                trace_left_speed_next(k,1) = inwards_speed(k,l);
            else
                street_inwards_next(k,l) = CAR;
                inwards_speed_next(k,l) = 0;
                trace_right_direction_next(k,l)=EXIT_LEFT;
            end
        end

        %inwards
        if (street_inwards(k,l) == CAR && trace_right_direction(k,l)~=EXIT_LEFT)
            gap = crosslight_measure_gap(-k, l, trace_right_direction(k,l) ,
                street_crossroad , ...
                street_outwards, street_outwards_next, 1, street_inwards,
                    street_inwards_next, trafficlight(3*k,1), ...
                EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                    EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
                    EMPTY_STREET);
            v = schreckenberg(inwards_speed(k,l),gap,dawdleProb);
            if(l == 1)
                inwards_gaps(1,k) = gap;
            end
            if (l+v<=STREET_INTERSECTION+1)      %not yet entering crossroad
                street_inwards_next(k,l+v) = CAR;
                inwards_speed_next(k,l+v) = v;
                trace_right_direction_next(k,l+v) = trace_right_direction(k,l);
            else
                ni = -k;                         %minus indices for
                    crosslight_measure_gap
```

```
                nj = STREET_INTERSECTION+1;
                q = 1;
                while(q <= l+v-(STREET_INTERSECTION+1))
                    if(ni > 0 || nj == STREET_INTERSECTION+1)
                        [ni, nj] = crosslight_next_ij(ni, nj, trace_right_direction(k,l) , ...
                            EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
                    else    %we are already in street_outwards
                        nj = nj+1;
                    end
                    q = q+1;
                end
                if (ni > 0)                 %place car in crossroad
                    street_crossroad_next(ni,nj) = CAR;
                    crossroad_speed_next(ni,nj) = v;
                    crossroad_exit_next(ni,nj) = trace_right_direction(k,l);
                else
                    street_outwards_next(-ni,nj) = CAR;     %ni again minus -> outside crossroad
                    outwards_speed_next(-ni,nj) = v;
                end
            end
        end

        %trace_left
        if (trace_left(k,l) == CAR)
            gap = crosslight_measure_gap(-k, l,EXIT_LEFT , street_crossroad, ...
                street_outwards, street_outwards_next, 1, trace_left, trace_left_next, trafficlight(2+3*(k-1),1), ...
                EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,EMPTY_STREET);
            v = schreckenberg(trace_left_speed(k,l),gap,dawdleProb);
            if (l+v<=STREET_INTERSECTION+1)
                trace_left_next(k,l+v) = CAR;
                trace_left_speed_next(k,l+v) = v;
            else
                ni = -k;
                nj = STREET_INTERSECTION+1;
                q = 1;
                while(q <= l+v-(STREET_INTERSECTION+1))
                    if(ni > 0 || nj == STREET_INTERSECTION+1)
                        [ni, nj] = crosslight_next_ij(ni, nj, EXIT_LEFT, ...
                            EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
                    else    %we are already in street_outwards
                        nj = nj+1;
                    end
                    q = q+1;
                end
                if (ni > 0)
                    street_crossroad_next(ni,nj) = CAR;
                    crossroad_speed_next(ni,nj) = v;
                    crossroad_exit_next(ni,nj) = EXIT_LEFT;
                else
```

```
                    street_outwards_next(-ni,nj) = CAR;
                    outwards_speed_next(-ni,nj) = v;
                end
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in crossroad

for i = 1:6
    for j = 1:6
        if (street_crossroad(i,j) == CAR)
            gap = crosslight_measure_gap(i, j,crossroad_exit(i,j), street_crossroad, ...
                street_outwards, street_outwards_next, 0, street_inwards, street_inwards_next, trafficlight(1+3*(k-1),1), ...
                EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT, EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION, EMPTY_STREET);
            v = schreckenberg(crossroad_speed(i,j),gap,dawdleProb);
            ni = i;
            nj = j;
            q = 1;
            while(q <= v)
                if(ni > 0)
                    [ni, nj] = crosslight_next_ij(ni, nj, crossroad_exit(i,j), ...
                        EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT, EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
                else    %we are already in street_outwards
                    nj = nj+1;
                end
                q = q+1;
            end
            if (ni > 0)
                street_crossroad_next(ni,nj) = CAR;
                crossroad_speed_next(ni,nj) = v;
                crossroad_exit_next(ni,nj) = crossroad_exit(i,j);
            else
                street_outwards_next(-ni,nj) = CAR;
                outwards_speed_next(-ni,nj) = v;
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car outwards

for k = 1:4
    for l = 1:STREET_INTERSECTION
        %outwards street
        e = 1;
        while (e <= 5 && street_outwards(k,l+e) == EMPTY_STREET && street_outwards_next(k,l+e) == EMPTY_STREET)
            e = e + 1;
        end
```

```
        gap = e - 1;
        v = schreckenberg(outwards_speed(k,l), gap, dawdleProb);
        if(street_outwards(k,l) == CAR)
            street_outwards_next(k,l+v) = CAR;
            outwards_speed_next(k,l+v) = v;
        end
    end
end

end
```

Listing 10: crosslight-measure-gap.m

```
function [ gap ] = crosslight_measure_gap(i, j, direction, street_crossroad, ...
    street_outwards, street_outwards_next, inwards, street_inwards,
        street_inwards_next, traffic_light, ...
    EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
        EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION, EMPTY_STREET)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%crosslight_measure_gap this function will measure the gap to the next car
%in a crosslight
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = 1;
iterate = 1;
ni = i;
nj = j;
while (e <= 5 && iterate)
    if((ni < 0 && nj == STREET_INTERSECTION+1 && inwards) || ni > 0)
        [ni, nj] = crosslight_next_ij(ni, nj, direction, ...
            EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
    else
        %ni = ni;
        nj = nj+1;
    end
    if(ni > 0)
        inwards = 0;
        if(street_crossroad(ni,nj) == EMPTY_STREET)
            e = e + 1;
        else
            iterate = 0;
        end
        if((direction == EXIT_LEFT || direction == EXIT_RIGHT) && e > 2)  %limit
            speed inside the crossection
            e = 2;
            iterate = 0;
        end
    else
        if(inwards)
            if(nj == STREET_INTERSECTION+1 || nj == STREET_INTERSECTION) %last or
                second to last field in front of intersection have to wait if
                traffic light is red
```

```
                    if(traffic_light && street_inwards(-ni,nj) == EMPTY_STREET && street_inwards_next(-ni,nj) == EMPTY_STREET)  %% traffic_light green and street empty
                        e = e + 1;
                    else
                        iterate = 0;
                    end
                else
                    if(street_inwards(-ni,nj) == EMPTY_STREET && street_inwards_next(-ni,nj) == EMPTY_STREET)
                        e = e + 1;
                    else
                        iterate = 0;
                    end
                end
            else
                if(street_outwards(-ni,nj) == EMPTY_STREET && street_outwards_next(-ni,nj) == EMPTY_STREET)
                    e = e + 1;
                else
                    iterate = 0;
                end
            end
        end
end
gap = e - 1;

end
```

Listing 11: crosslight-next-ij.m

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

switch(direction)
    case EXIT_LEFT
        if(i == 1 && j == 3)
            ni = 2;
            nj = 3;
        elseif(i == 2 && j == 3)
            ni = 3;
            nj = 4;
        elseif(i == 3 && j == 4)
            ni = 4;
            nj = 5;
        elseif(i == 4 && j == 5)
            ni = 5;
            nj = 6;
        elseif(i == 5 && j == 6)
            ni = -4;
```

```
            nj = 1;
        elseif(i == 4 && j == 1)
            ni = 4;
            nj = 2;
        elseif(i == 4 && j == 2)
            ni = 3;
            nj = 3;
        elseif(i == 3 && j == 3)
            ni = 2;
            nj = 4;
        elseif(i == 2 && j == 4)
            ni = 1;
            nj = 5;
        elseif(i == 1 && j == 5)
            ni = -1;
            nj = 1;
        elseif(i == 6 && j == 4)
            ni = 5;
            nj = 4;
        elseif(i == 5 && j == 4)
            ni = 4;
            nj = 3;
        elseif(i == 4 && j == 3)
            ni = 3;
            nj = 2;
        elseif(i == 3 && j == 2)
            ni = 2;
            nj = 1;
        elseif(i == 2 && j == 1)
            ni = -2;
            nj = 1;
        elseif(i == 3 && j == 6)
            ni = 3;
            nj = 5;
        elseif(i == 3 && j == 5)
            ni = 4;
            nj = 4;
        elseif(i == 4 && j == 4)
            ni = 5;
            nj = 3;
        elseif(i == 5 && j == 3)
            ni = 6;
            nj = 2;
        elseif(i == 6 && j == 2)
            ni = -3;
            nj = 1;
        elseif(i < 0)   %here I assume the car is in the last position of the
            inmwards street
            if(i == -1)
                ni = 1;
                nj = 3;
            elseif(i == -2)
                ni = 4;
                nj = 1;
            elseif(i == -3)
                ni = 6;
                nj = 4;
            elseif(i == -4)
```

```
                    ni = 3;
                    nj = 6;
                end
            end
        case EXIT_RIGHT
            if(i == 1)
                if(j == 1)
                    ni = -2;
                    nj = 1;
                else
                    ni = -1;
                    nj = 1;
                end
            elseif(i == 6)
                if(j == 1)
                    ni = -3;
                    nj = 1;
                else
                    ni = -4;
                    nj = 1;
                end
            elseif(i == -1)
                 ni = 1;
                 nj = 1;
            elseif(i == -2)
                 ni = 6;
                 nj = 1;
            elseif(i == -3)
                 ni = 6;
                 nj = 6;
            elseif(i == -4)
                 ni = 1;
                 nj = 6;
            end
        case EXIT_STRAIGHT_TOP
            if(i > 0)
                nj = j;
                ni = i-1;
                if(ni < 1)
                    ni = -EXIT_STRAIGHT_BOTTOM;
                    nj = 1;
                end
            elseif(i == -EXIT_STRAIGHT_TOP)  %%check if it comes from BOTTOM
                nj = 5;
                ni = 6;
            else
                ni = i;
                nj = j+1;
            end
        case EXIT_STRAIGHT_BOTTOM
            if(i > 0)
                nj = j;
                ni = i+1;
                if(ni > 6)
                    ni = -EXIT_STRAIGHT_TOP;
                    nj = 1;
                end
            elseif(i == -EXIT_STRAIGHT_BOTTOM)
```

```
            nj = 2;
            ni = 1;
        else
            ni = i;
            nj = j+1;
        end
    case EXIT_STRAIGHT_LEFT
        if(i > 0)
            nj = j-1;
            ni = i;
            if(nj < 1)
                ni = -2;
                nj = 1;
            end
        elseif(i == -4)
            nj = 6;
            ni = 2;
        else
            ni = i;
            nj = j+1;
        end
    case EXIT_STRAIGHT_RIGHT
        if(i > 0)
            nj = j+1;
            ni = i;
            if(nj > 6)
                ni = -4;
                nj = 1;
            end
        elseif(i == -2)
            nj = 1;
            ni = 5;
        else
            ni = i;
            nj = j+1;
        end
    otherwise
        display(direction);
        display(i);
        display(j);
        ni = 0;
        nj = 0;
end

end
```

Listing 12: plotresults.m

```
function plotresults(d, pd, folder)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFIC Simulation of traffic in an city map containing roundabouts and
%crossroads.
%
%This function will plot the precalculated results which are stored in
%results/folder where folder is the variable supllied from above
%
%INPUTS:
%D is the car density you want to plot over (should be a vector, else the
```

```
%plot would only show one point
%PD is the pedestrian density
%folder is the folder your data is located, this should be an integer!!
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;

%%% runtime measurement - start
tic;

filename = sprintf('../results/%g/config.mat', folder);
load(filename,'c', 'pahead');


[c_m,c_n] = size(c);
%check if city map is a mix of crossroads and roundaoubts or if it is made up
%purely of one or the other
mix = not( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 );

%average flow and distributions for every density suppied
avFlow = zeros(max(size(pd)),max(size(d)));
avRo = zeros(max(size(pd)),max(size(d)));
avCr = zeros(max(size(pd)),max(size(d)));
avSpeed = zeros(max(size(pd)),max(size(d)));

for di=1:max(size(d))
    for pdi=1:max(size(pd))
        [config_m,config_n] = size(c);
        filename = sprintf('../results/%g/result_(%g x %g)_%g_%g.mat', folder, ...
            config_m, config_n, d(di), pd(pdi));
        if exist(filename, 'file')
            disp(filename);
            load(filename,'result');
            disp(result);
            avFlow(pdi,di) = result(1);
            avRo(pdi,di) = result(2);
            avCr(pdi,di) = result(3);
            avSpeed(pdi,di) = result(4);
        end
    end
end

fig2 = figure(2);
%is city map is a mix of roundabout and crossroads, plot distribution
if ( mix )
    %plot relative number of cars at roundabouts and number of cars at
    %crossroads versus traffic density
    subplot(2,1,2);
    plot(d,avRo*100,'rx',d,avCr*100,'gx');
    set(gca,'FontSize',16);
```

```
    title('Traffic Distribution');
    xlabel('traffic density');
    ylabel('relative number of cars [%]');
    legend('around roundabouts','around crossroads');
    ylim([0 100]);
    subplot(2,1,1);
end

%plot traffic flow versus traffic density
hold on;
% size(avFlow)
for i=1:length(pd)
    pd(i);
    avFlow_pdi = avFlow(i,:);
    plot(d,avFlow_pdi, '-x');
end
% plot(d,avFlow(:,:), '-o')
set(gca,'FontSize',16);
title('Traffic Dynamics');
xlabel('traffic density');
ylabel('average traffic flow');
%ylim([0 0.5]);

fig3 = figure(3);
hold on;
for i=1:length(d)
    d(i);
    avFlow_di = avFlow(:,i);
    plot(pd,avFlow_di, '-x');
end
% plot(pd,avFlow(:,:), '-o')
set(gca,'FontSize',16);
title('Traffic Dynamics');
xlabel('pedestrian density');
ylabel('average traffic flow');


fig4 = figure(4);
hold on;
for i=1:length(pd)
    pd(i);
    avSpeed_pdi = avSpeed(i,:);
    plot(d,avSpeed_pdi, '-x');
end
set(gca,'FontSize',16);
title('Traffic Dynamics');
xlabel('traffic density');
ylabel('average speed');


fig5 = figure(5);
hold on;
for i=1:length(d)
    d(i);
    avSpeed_di = avSpeed(:,i);
    plot(pd,avSpeed_di, '-x');
end
set(gca,'FontSize',16);
```

```
title('Traffic Dynamics');
xlabel('pedestrian density');
ylabel('average speed');

fig6 = figure(6);
surf(d,pd,avSpeed);
title('Traffic Dynamics','FontWeight','bold');
xlabel('traffic density');
ylabel('pedestrian density');
zlabel('average speed');


fig7 = figure(7);
surf(d,pd,avFlow);
title('Traffic Dynamics','FontWeight','bold');
xlabel('traffic density');
ylabel('pedestrian density');
zlabel('average traffic flow');




%% runtime measurement - end
toc;

end
```

Listing 13: plot-map.m

```
function [map] = plot_map(street_length, config, car_density, display, ...
    street_inwards, street_outwards, street_roundabout, street_crossroad, ...
    BUILDING,EMPTY_STREET, light, trace_left, STREET_INTERSECTION)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PLOT_MAP This function plots the map
%
%This program requires the following subprogams:
%none
%
%A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Fall 2012
%Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
    course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%dimensions of config, how many intersections in x and y direction are there?
[config_m,config_n] = size(config);

%initialize map
map = zeros(config_m*(2*street_length+6),config_n*(2*street_length+6));
map(1,1)=2;

%iterate over all intersection
for a = 1:config_m
    for b = 1:config_n

```

```
        %define Index starting points for each intersection
        tI_m = (a - 1) * 4;
        tI_n = (b - 1) * street_length;
        mapI_m = (a - 1) * (2 * street_length + 6);
        mapI_n = (b - 1) * (2 * street_length + 6);


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %write roundabout into map

        %check if intersection is a roundabout
        if  ( config(a,b) == 0 )
            %define index starting point for this roundabout
            rI_n = (b - 1) * 12;
            %write roundabout into map
            map(mapI_m+street_length+1:mapI_m+street_length+6,...
                mapI_n+street_length+1:mapI_n+street_length+6) = ...
                [ BUILDING EMPTY_STREET street_roundabout(a,rI_n+4) street_roundabout(a,rI_n+3) EMPTY_STREET BUILDING;
                EMPTY_STREET street_roundabout(a,rI_n+5) EMPTY_STREET EMPTY_STREET street_roundabout(a,rI_n+2) EMPTY_STREET;
                street_roundabout(a,rI_n+6) EMPTY_STREET BUILDING BUILDING EMPTY_STREET street_roundabout(a,rI_n+1);
                street_roundabout(a,rI_n+7) EMPTY_STREET BUILDING BUILDING EMPTY_STREET street_roundabout(a,rI_n+12);
                EMPTY_STREET street_roundabout(a,rI_n+8) EMPTY_STREET EMPTY_STREET street_roundabout(a,rI_n+11) EMPTY_STREET;
                BUILDING EMPTY_STREET street_roundabout(a,rI_n+9) street_roundabout(a,rI_n+10) EMPTY_STREET BUILDING];

            %write streets into map
            %normal street
            for i = 1:street_length-3
                map(mapI_m+i,mapI_n+street_length+2) = street_inwards(tI_m+1,tI_n+i); % top, inwards
                map(mapI_m+street_length+5,mapI_n+i) = street_inwards(tI_m+2,tI_n+i); % left, inwards
                map(mapI_m+2*street_length+7-i,mapI_n+street_length+5) = street_inwards(tI_m+3,tI_n+i); % bottom, inwards
                map(mapI_m+street_length+2,mapI_n+2*street_length+7-i) = street_inwards(tI_m+4,tI_n+i); % right, inwards
            end
            for i = 1+3:street_length
                map(mapI_m+street_length+1-i,mapI_n+street_length+5) = street_outwards(tI_m+1,tI_n+i);  % top, outwards
                map(mapI_m+street_length+2,mapI_n+street_length+1-i) = street_outwards(tI_m+2,tI_n+i);  % left, outwards
                map(mapI_m+street_length+6+i,mapI_n+street_length+2) = street_outwards(tI_m+3,tI_n+i);  % bottom, outwards
                map(mapI_m+street_length+5,mapI_n+street_length+6+i) = street_outwards(tI_m+4,tI_n+i);  % right, outwards
            end
            %'last mile'
            for i = street_length-3+1:street_length
                map(mapI_m+i,mapI_n+street_length+3) = street_inwards(tI_m+1,tI_n+i); % top, inwards
                map(mapI_m+street_length+4,mapI_n+i) = street_inwards(tI_m+2,tI_n+i); % left, inwards
```

```
            map(mapI_m+2*street_length+7-i,mapI_n+street_length+4) = street_inwards(tI_m+3,tI_n+i); % bottom, inwards
            map(mapI_m+street_length+3,mapI_n+2*street_length+7-i) = street_inwards(tI_m+4,tI_n+i); % right, inwards
        end
        for i = 1:3
            map(mapI_m+street_length+1-i,mapI_n+street_length+4) = street_outwards(tI_m+1,tI_n+i);  % top, outwards
            map(mapI_m+street_length+3,mapI_n+street_length+1-i) = street_outwards(tI_m+2,tI_n+i);  % left, outwards
            map(mapI_m+street_length+6+i,mapI_n+street_length+3) = street_outwards(tI_m+3,tI_n+i);  % bottom, outwards
            map(mapI_m+street_length+4,mapI_n+street_length+6+i) = street_outwards(tI_m+4,tI_n+i);  % right, outwards
        end
        %filling fields for optics
        map(mapI_m+street_length+1-4,mapI_n+street_length+3) = EMPTY_STREET;  % top, left
        map(mapI_m+street_length+1-4,mapI_n+street_length+4) = EMPTY_STREET;  % top, right
        map(mapI_m+street_length+3,mapI_n+street_length+1-4) = EMPTY_STREET;  % left, top
        map(mapI_m+street_length+4,mapI_n+street_length+1-4) = EMPTY_STREET;  % left, bottom
        map(mapI_m+street_length+6+4,mapI_n+street_length+3) = EMPTY_STREET;  % bottom, left
        map(mapI_m+street_length+6+4,mapI_n+street_length+4) = EMPTY_STREET;  % bottom, right
        map(mapI_m+street_length+3,mapI_n+street_length+6+4) = EMPTY_STREET;  % right, top
        map(mapI_m+street_length+4,mapI_n+street_length+6+4) = EMPTY_STREET;  % right, bottom
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %write crossing into map

    %check if intersection is a crossing with priority to the right
    if ( config(a,b) == 1 )
        %define index starting points for this crossroad
        pI_m = (a - 1) * 6;
        pI_n = (b - 1) * 6;
        pIl_n = (b - 1) * 12;   % index for light
        pIt_m = (a - 1) * 4;    % m-index for trace left
        pIt_n = (b - 1) * 8;    % n-index for trace left
        %write crossroad into map
        map(mapI_m+street_length+1:mapI_m+street_length+6,...
            mapI_n+street_length+1:mapI_n+street_length+6) = ...
            street_crossroad(pI_m+1:pI_m+6,pI_n+1:pI_n+6);

        %traffic lights
        GREEN_LIGHT = 1.3;
        RED_LIGHT = 1.6;
        light(light==1) = GREEN_LIGHT;
        light(light==0) = RED_LIGHT;

        map(mapI_m+street_length-2, mapI_n+street_length+1) = light(a, pIl_n+0*3+3); % top, inwards
```

```
        map(mapI_m+street_length-2, mapI_n+street_length+4) = light(a, pIl_n+0*3+2); % top, trace_left
        map(mapI_m+street_length-1, mapI_n+street_length+6) = light(a, pIl_n+0*3+1); % top, pedestrians

        map(mapI_m+street_length+1, mapI_n+street_length-1) = light(a, pIl_n+1*3+1); % left, pedestrians
        map(mapI_m+street_length+3, mapI_n+street_length-2) = light(a, pIl_n+1*3+2); % left, trace_left
        map(mapI_m+street_length+6, mapI_n+street_length-2) = light(a, pIl_n+1*3+3); % left, inwards

        map(mapI_m+street_length+6+2, mapI_n+street_length+1) = light(a, pIl_n+2*3+1); % bottom, pedestrians
        map(mapI_m+street_length+6+3, mapI_n+street_length+3) = light(a, pIl_n+2*3+2); % bottom, trace_left
        map(mapI_m+street_length+6+3, mapI_n+street_length+6) = light(a, pIl_n+2*3+3); % bottom, inwards

        map(mapI_m+street_length+1, mapI_n+street_length+6+3) = light(a, pIl_n+3*3+3); % right, inwards
        map(mapI_m+street_length+4, mapI_n+street_length+6+3) = light(a, pIl_n+3*3+2); % right, trace_left
        map(mapI_m+street_length+6, mapI_n+street_length+6+2) = light(a, pIl_n+3*3+1); % right, pedestrians

        %trace left
        trace_left_length = STREET_INTERSECTION+1;
        for i = 1:trace_left_length
            map(mapI_m+street_length+7+trace_left_length-i,mapI_n+street_length+4) = trace_left(pIt_m+3,pIt_n+i); % bottom, trace_left
            map(mapI_m+street_length+3,mapI_n+street_length+7+trace_left_length-i) = trace_left(pIt_m+4,pIt_n+i); % right, trace_left
            map(mapI_m+street_length-trace_left_length+i,mapI_n+street_length+3) = trace_left(pIt_m+1,pIt_n+i); % top, trace_left
            map(mapI_m+street_length+4,mapI_n+street_length-trace_left_length+i) = trace_left(pIt_m+2,pIt_n+i); % left, trace_left
        end

        %write streets into map
        for i = 1:street_length
            map(mapI_m+i,mapI_n+street_length+2) = street_inwards(tI_m+1,tI_n+i); % top, inwards
            map(mapI_m+street_length+5,mapI_n+i) = street_inwards(tI_m+2,tI_n+i); % left, inwards
            map(mapI_m+2*street_length+7-i,mapI_n+street_length+5) = street_inwards(tI_m+3,tI_n+i); % bottom, inwards
            map(mapI_m+street_length+2,mapI_n+2*street_length+7-i) = street_inwards(tI_m+4,tI_n+i); % right, inwards
            map(mapI_m+street_length+1-i,mapI_n+street_length+5) = street_outwards(tI_m+1,tI_n+i);  % top, outwards
            map(mapI_m+street_length+2,mapI_n+street_length+1-i) = street_outwards(tI_m+2,tI_n+i);  % left, outwards
            map(mapI_m+street_length+6+i,mapI_n+street_length+2) = street_outwards(tI_m+3,tI_n+i);  % bottom, outwards
            map(mapI_m+street_length+5,mapI_n+street_length+6+i) = street_outwards(tI_m+4,tI_n+i);  % right, outwards
        end
```

```
            end

        end
end

% %illustrate trafic situation (now, not of next time step)
% fig1 = figure(1);
% imagesc(map);
% load('colormap2', 'mycmap')
% set(fig1, 'Colormap', mycmap)
% titlestring = sprintf('Density = %g',car_density);
% title(titlestring);
% drawnow;

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 crossroads. 2010. 6