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

Contents

1 Abstract

2 Individual contributions

3 Introduction and Motivations

4 Description of the Model and Implementation

4.1 Description of the main loop

In our model one can compare roundabouts with crossroads, controlled by traffic lights (which we will call crosslight), with each other. One can use an arbitrary combination of roundabouts and crosslights in a $N \times M$ map.
The simulation can be done with different probabilities for the car to go straight and left and right will have the same probability but they depend on the probability ahead. One can also choose different car and pedestrian densities. The simulation will generate a plot over these densities as x- and y- axis and the average flow and average speed as z-axis.

$$flow = density \cdot speed$$

4.1.1 Implementation

We have a big matrix which shows all roads and intersections. And many smaller ones, two for every lane, which contains all the lanes for every road, they are stored after each other. The first one contains the positions of the cars and the second one contains their speed. And we also have one for every array which is used by a crosslight or roundabout and needs to be stored for calculating the next step.

For almost every one of those arrays we have two arrays, one for the current state which is shown on the screen and one for the next step, which will be calculated cell for cell. After the calculation the next step will be stored in the first array and the calculation starts over again.

4.2 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).

4.2.1 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.

For the pedestrians we use a yellow colour on the street (a car is blue), and two 'buckets' between the lanes of each road so that they will cross both lanes of a road.

5 Simulation Results and Discussion

6 Summary and Outlook

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, traffic density and whether
%simulation is to be displayed or not.
%
%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:
%-Black cells simbolize empty space
%-White cells simbolize road
%-Red cells simbolize cars
%-Yellow cells simbolize cars indicating to the right
%-Dark red celss simbolize cars indicating to the left
%
%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\n\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
```

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

```
109 end
110
111
112 end
```

Listing 2: trafficloop.m

```
1  function trafficloop(c, d, pahead, pd, show, slow_motion, video, store_results,
       folder)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %TRAFFIC Simulation of traffic in an city map containing roundabouts and
4  %crossroads.
5  %
6  %This program requires the following subprogams:
7  %TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
8  %
9  %
10 %User will be ask to determine city map,traffic density and whether
11 %simulation is to be displayed or not.
12 %
13 %The city map is entered by supplying a matrix with elements '1' for
14 %crossroads and '0' for roundabouts.
15 %
16 %The density can be a scalar or a vector. If the density is a scalar
17 %TRAFFIC will run the simulation for all densities given. The elements must
18 %be in the range of [0,1].
19 %
20 %If Users chooses to display simulation (by entering 'y') a figure will
21 %open showing the animation:
22 %-Black cells simbolize empty space
23 %-White cells simbolize road
24 %-Red cells simbolize cars
25 %-Yellow cells simbolize cars indicating to the right
26 %-Dark red celss simbolize cars indicating to the left
27 %
28 %After all simulations have finished TRAFFIC plots the average traffic flow
29 %versus the traffic density. If city map is a mix of crossroad and
30 %roundabouts the traffic distribution (cars around roundabouts or around
31 %crossroads) versus traffic density is also plotted.
32 %
33 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
34 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
35 %Fall 2012
36 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
       course "Modelling
37 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
38 %Spring 2010
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41
42 %%%%
43 % define global variables
44 BUILDING = 0;     %the colour for buildings
45 EMPTY_STREET = 1;
46 CAR = 0.4;
47 CAR_NEXT_EXIT = 0.6;     %the colour of a car which will take the next exit
48 PEDESTRIAN = 0.8;
49
```

```
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
%crossroads.
%
%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
%
%INPUT:
%DENSITY, Traffic density
%CONFIG, City map
%DISPlAY, Turn graphics on 'true' or off 'false'
%
%This program requires the following subprogams:
%ROUNDABOUT,CROSSROAD,CONNECTION,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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%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);

%in streets cell values indicate the following:
%CAR means there is a car in this position (red in figure)
%EMPTY_STREET means there is no car in this position (white in figure)

%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 t
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);
turnphase = ceil((ligthlength*(1-pahead)/2)/switchtime);
totalphase = 6 + 2*aheadphase + 4*turnphase;
count =0;
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);
%     ax1 = gca;
    titlestring = sprintf('Density = %g',car_density);
%     title(ax1, titlestring, 'FontWeight','bold');
%     [X,Y] = meshgrid(1:config_m*(2*street_length+6),1:config_n*(2*street_length+6)
    );

    %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);
```

```
137         open(vidObj);
138     end
139 end
140
141 %iterate over time
142 for time = 1:nIt+1
143
144     %clear values for next step
145     street_inwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
146     inwards_speed_next = zeros(4*config_m,street_length*config_n);
147     street_outwards_next = ones(4*config_m,street_length*config_n)*EMPTY_STREET;
148     outwards_speed_next = zeros(4*config_m,street_length*config_n);
149     trace_left_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
150     trace_left_speed_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
151     trace_right_direction_next=zeros(4*config_m,(STREET_INTERSECTION+1)*config_n);
152
153
154     %calculate taffic light phase
155     if (count == switchtime)
156         if (phase == totalphase+1)
157             phase = 0;
158         end
159         phase = phase+1;
160         count = 0;
161     else
162         count = count +1;
163     end
164
165     %iterate over all intersection
166     for a = 1:config_m
167         for b = 1:config_n
168
169             %define Index starting points for each intersection
170             tI_m = (a - 1) * 4;
171             tI_n = (b - 1) * street_length;
172
173             %positions outside intersections
174             %for every intersection iterate along streets
175             for c = tI_m + 1:tI_m +4
176                 for d = tI_n + 1:tI_n+street_length
177
178                     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
179                     %streets to intersections
180
181                     %deal with the STREET_INTERSECTION positions directly in front
                            of intersection
182                     %separately later
183                     if ( d-tI_n < street_length-STREET_INTERSECTION)
184                         %if there is a car in this position, apply
185                         %NS-Model
186                         if ( street_inwards(c,d) == CAR )
187                             %Nagel-Schreckenberg-Model
188                             gap = measure_gap(street_inwards, street_outwards,
                                    street_length, a, b, c, d, 1, ...
189                                     inwards_gaps(a,(b - 1) *4+c-tI_m), config_m,
                                        config_n, EMPTY_STREET,STREET_INTERSECTION);
190                             v = schreckenberg(inwards_speed(c,d), gap, dawdleProb);
191
```

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

```
                    STREET_INTERSECTION:tI_n+street_length), ...
246                 street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
                        STREET_INTERSECTION+6), ...
247                 outwards_speed_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
                        STREET_INTERSECTION+6), ...
248                 street_roundabout_next(a,rI_n+1:rI_n+12), ...
249                 roundabout_speed_next(a,rI_n+1:rI_n+12), ...
250                 roundabout_exit(a,rI_n+1:rI_n+12), ...
251                 pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4)
                        , ...
252                 inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4) ] = ...
253                 roundabout(street_inwards(tI_m+1:tI_m+4,tI_n+street_length-
                        STREET_INTERSECTION:tI_n+street_length), ...
254                 inwards_speed(tI_m+1:tI_m+4,tI_n+street_length-
                        STREET_INTERSECTION:tI_n+street_length), ...
255                 street_outwards(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6)
                        , ...
256                 outwards_speed(tI_m+1:tI_m+4,tI_n+1:tI_n+STREET_INTERSECTION+6),
                         ...
257                 street_roundabout(a,rI_n+1:rI_n+12), ...
258                 roundabout_exit(a,rI_n+1:rI_n+12), ...
259                 pedestrian_bucket((a-1)*2+1:(a-1)*2+2,(b - 1) *4+1:(b - 1) *4+4)
                        , ...
260                 inwards_gaps(a,(b - 1) *4+1:(b - 1) *4+4), dawdleProb, ...
261                 pedestrian_density, ...
262                 street_inwards_next(tI_m+1:tI_m+4,tI_n+street_length-
                        STREET_INTERSECTION:tI_n+street_length), ...
263                 inwards_speed_next(tI_m+1:tI_m+4,tI_n+street_length-
                        STREET_INTERSECTION:tI_n+street_length), ...
264                 street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
                        STREET_INTERSECTION+6), ...
265                 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);
266
267             %add cars around this crossroad in this time step to
268             %counter for cars around crossroads
269             for v = tI_m+1:tI_m+4
270                 for w = tI_n+1:tI_n+street_length
271                     if ( street_inwards(v,w) ~= 1 )
272                         numCaRoIt(time) = numCaRoIt(time) + 1;
273                     end
274                     if ( street_outwards(v,w) ~= 1 )
275                         numCaRoIt(time) = numCaRoIt(time) + 1;
276                     end
277                 end
278             end
279             for y = rI_n+1:rI_n+12
280                 if ( street_roundabout(a,y) ~= 1 )
281                     numCaRoIt(time) = numCaRoIt(time) + 1;
282                 end
283             end
284
285         end
286
287         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
288         %crossroads
289
```

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

```
                    STREET_INTERSECTION:tI_n+street_length), ...
334                 street_outwards_next(tI_m+1:tI_m+4,tI_n+1:tI_n+
                        STREET_INTERSECTION+6), ...
335                 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, ...
336                 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), ...
337                 localphase, aheadphase, turnphase);
338
339
340             %add cars around this crossroad in this time step to
341             %counter for cars around crossroad
342             for v = tI_m+1:tI_m+4
343                 for w = tI_n+1:tI_n+street_length
344                     if ( street_inwards(v,w) ~= 1 )
345                         numCaCrIt(time) = numCaCrIt(time) + 1;
346                     end
347                     if ( street_outwards(v,w) ~= 1 )
348                         numCaCrIt(time) = numCaCrIt(time) + 1;
349                     end
350                 end
351             end
352             for x = pI_m+1:pI_m+6
353                 for y = pI_n+1:pI_n+6
354                     if ( street_crossroad(x,y) ~= 0 )
355                         numCaCrIt(time) = numCaCrIt(time) + 1;
356                     end
357                 end
358             end
359
360         end
361
362     end
363 end
364
365 %calculate average velosity per time step
366 avSpeedIt(time) = ( sum(sum(inwards_speed)) + sum(sum(outwards_speed)) + ...
367     sum(sum(roundabout_speed)) + sum(sum(crossroad_speed)) ) / numCars;
368
369 %plot the map in this timestep into the figure
370 if (display)
371     map = plot_map(street_length, config, car_density, display, ...
372         street_inwards, street_outwards, street_roundabout, street_crossroad,
                ...
373         BUILDING,EMPTY_STREET, light, trace_left, STREET_INTERSECTION);
374     %illustrate trafic situation (now, not of next time step)
375     imagesc(map);
376 %       hold on;
377 %       view(0,90);
378 %       surf(X,Y,map, 'EdgeColor', 'none');
379     title(titlestring, 'FontWeight','bold');
380     drawnow;
381     if (video)
382         % get the current frame
383         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 && ...
```

```
12          street_inwards(c,d+e) == EMPTY_STREET;                %STREET_INTERSECTION
                specifies the number of elements of the road inwards which will be taken
                 care of by the crossroad/roundabout
13      else
14          e = e + 1;
15          %if gap is bigger than distance to edge,connect
16          %steets
17          if ( d + e > b * street_length)
18              %testing position in new street
19              hh = d + e - b * street_length;
20              %connect to next street
21              [ec,ed]=connection(a,b,c,hh, ...
22                  config_m,config_n,street_length);
23              while ( street_inwards(ec,ed) == EMPTY_STREET && e <= 5 )
24                  e = e + 1;
25                  %testing position in new street
26                  hh = d + e - b * street_length;
27                  %connect to next street
28                  [ec,ed]=connection(a,b,c,hh, ...
29                      config_m,config_n,street_length);
30              end
31              iterate = 0;
32          else
33              iterate = e <= 5 && street_outwards(c,d+e) == EMPTY_STREET;     %% <= 4 b
                    .c. it'll be 5 after this loop
34          end
35      end
36  end
37  gap = e - 1;
38
39  end
```

Listing 5: connection.m

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

%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

```

```
23 end
```

Listing 8: roundabout.m

```
1  function [street_inwards_next, ...
2      inwards_speed_next, ...
3      street_outwards_next, ...
4      outwards_speed_next, ...
5      street_roundabout_local_next, ...
6      roundabout_speedlocal_next, ...
7      roundabout_exit_local_next, ...
8      pedestrian_bucket, inwards_gaps] ...
9      = roundabout(street_inwards, ...
10     inwards_speed, ...
11     street_outwards, ...
12     outwards_speed, ...
13     street_roundabout, ...
14     roundabout_exit ,pedestrian_bucket, ...
15     inwards_gaps, dawdleProb, ...
16     pedestrian_density, ...
17     street_inwards_next, ...
18     inwards_speed_next, ...
19     street_outwards_next,...
20     outwards_speed_next,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
           STREET_INTERSECTION,pahead)
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %ROUNDABOUT Calculation of update for a certain roundabout, density and
23 %time step
24 %
25 %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
26 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
27 %Fall 2012
28 %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
       course "Modelling
29 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
30 %Spring 2010
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33 %in roundabout cell values indicate if car is about to leave roundabout:
34 %0.4 means car is not taking next exit (red in figure)
35 %0.7 means car is taking next exit (yellow in figure)
36 %1 means no car in this position (white in figure)
37
38 %clear local next variables
39 street_roundabout_local_next = ones(1,12)*EMPTY_STREET;
40 roundabout_speedlocal_next = zeros(1,12);
41 roundabout_exit_local_next = zeros(1,12);
42
43 temp_roundabout_pedestrian_bucket = pedestrian_bucket;
44
45 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46 %car in front of roundabout
47
48 for k = 1:4
49     if ( street_inwards(k,STREET_INTERSECTION+1) == CAR )
50         %entering roundabout with velocity 1 when possible
51         %roundabout position index
52         iR = mod(3*k+1,12);
```

```
53          % enter roundabout if car at position k*3 is about to exit and
54          % there is no car at position 3*k+1
55          if ( roundabout_exit(k*3) <= 1 && street_roundabout(iR) == EMPTY_STREET )
56              %enter roundabout
57              %decide which exit car is going to take
58              u = rand(1);
59              %if it takes 1. exit
60              if ( u <= (0.95/2*(1-pahead)))
61                  roundabout_exit_local_next(iR) = 1;
62                  %indicate
63                  street_roundabout_local_next(iR) = CAR_NEXT_EXIT;
64                  roundabout_speedlocal_next(iR) = 1;
65              %if it takes 2. exit
66              elseif ( u <= (0.95/2*(1+pahead)))
67                  roundabout_exit_local_next(iR) = 2;
68                  street_roundabout_local_next(iR) = CAR;
69                  roundabout_speedlocal_next(iR) = 1;
70              %if it takes 3. exit
71              elseif ( u <= 0.95 )
72                  roundabout_exit_local_next(iR) = 3;
73                  street_roundabout_local_next(iR) = CAR;
74                  roundabout_speedlocal_next(iR) = 1;
75              %if it takes 4. exit (turns around)
76              else
77                  roundabout_exit_local_next(iR) = 4;
78                  street_roundabout_local_next(iR) = CAR;
79                  roundabout_speedlocal_next(iR) = 1;
80              end
81
82          %car waiting in front of roundabout
83          else
84              street_inwards_next(k,STREET_INTERSECTION+1) = street_inwards(k,
                    STREET_INTERSECTION+1);
85              inwards_speed_next(k,STREET_INTERSECTION+1) = 0;
86          end
87      end
88  end
89
90  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91  %pedestrians
92
93
94  for k = 1:4
95      r = rand(1);
96      if (( street_inwards(k,STREET_INTERSECTION) == EMPTY_STREET || street_inwards(k,
            STREET_INTERSECTION) == PEDESTRIAN) && ...
97              (r <= pedestrian_density || pedestrian_bucket(1,k) > 0))
98          street_inwards_next(k,STREET_INTERSECTION) = PEDESTRIAN;
99          inwards_speed_next(k,STREET_INTERSECTION) = 0;
100         if(r <= pedestrian_density)
101             temp_roundabout_pedestrian_bucket(2,k) = 1;
102         end
103         if(pedestrian_bucket(1,k) > 0)
104             temp_roundabout_pedestrian_bucket(1,k) = 0;
105         end
106     end
107     r = rand(1);
108     if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN
```

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

```
160         e = 1;
161         while (e <= 5 && j + e <= STREET_INTERSECTION+1 && ((street_inwards(k,j+e)
                == EMPTY_STREET && street_inwards_next(k,j+e) == EMPTY_STREET) || ...
162                 ( street_inwards(k,j+e) == PEDESTRIAN && street_inwards_next(k,j
                        +e) == EMPTY_STREET) ))
163             e = e + 1;
164         end
165         gap = e - 1;
166         v = schreckenberg(inwards_speed(k,j), gap, dawdleProb);
167         if(j == 1)
168             inwards_gaps(1,k) = gap;
169         end
170         if(street_inwards(k,j) == CAR)
171             if ( ( street_inwards(k,j+v) == EMPTY_STREET && street_inwards_next(k,j+
                    v) == EMPTY_STREET) || ...
172                     ( street_inwards(k,j+v) == PEDESTRIAN && street_inwards_next(k,j
                            +v) == EMPTY_STREET) )
173                 street_inwards_next(k,j+v) = CAR;
174                 inwards_speed_next(k,j+v) = v;
175             else
176                 street_inwards_next(k,j) = CAR;
177                 inwards_speed_next(k,j) = 0;
178             end
179         end
180     end
181 end
182
183
184
185 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186 %car in roundabout
187
188 for j = 1:12
189     if ( street_roundabout(j) == CAR || street_roundabout(j) == CAR_NEXT_EXIT )
190
191         %cars in roundabout not at an exit
192         if (mod(j,3) ~= 0 )
193             %if space free, move one forward
194             if ( street_roundabout(j+1) == EMPTY_STREET &&
                    street_roundabout_local_next(j+1) == EMPTY_STREET)
195                 %take new position
196                 street_roundabout_local_next(j+1) = street_roundabout(j);
197                 roundabout_speedlocal_next(j+1) = 1;
198                 roundabout_exit_local_next(j+1) = roundabout_exit(j);
199             %if no space free, stay
200             else
201                 street_roundabout_local_next(j) = street_roundabout(j);
202                 roundabout_speedlocal_next(j) = 0;
203                 roundabout_exit_local_next(j) = roundabout_exit(j);
204             end
205
206         %car at an exit
207         else
208
209             %if car is at its exit
210             if ( roundabout_exit(j) == 1 )
211                 %if space free, leave roundabout
212                 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 r(12) with r(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, ...
```

```
14      street_crossroad, ...
15      crossroad_speed, ...
16      crossroad_exit, pedestrian_bucket, ...
17      inwards_gaps, dawdleProb, ...
18      pedestrian_density, ...
19      street_inwards_next, ...
20      inwards_speed_next, ...
21      street_outwards_next, ...
22      outwards_speed_next,EMPTY_STREET,CAR,CAR_NEXT_EXIT,PEDESTRIAN,
            STREET_INTERSECTION, ...
23      pahead, trace_left, trace_left_speed, trace_right_direction, ...
24      localphase, aheadphase, turnphase)
25  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26  %CROSSROAD Calculation of update for a certain crossroad, density and time
27  %step
28  %
29  %This program requires the following subprogams:
30  %PDESTINATION
31  %
32  %A project by Marcel Arikan, Nuhro Ego and Ralf Kohrt in the GeSS course "Modelling
33  %and Simulation of Social Systems with MATLAB" at ETH Zurich.
34  %Fall 2012
35  %Matlab code is based on code from Bastian Buecheler and Tony Wood in the GeSS
        course "Modelling
36  %and Simulation of Social Systems with MATLAB" at ETH Zurich.
37  %Spring 2010
38  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39
40  NO_EXIT_YET = 0;
41  EXIT_LEFT = 5;
42  EXIT_RIGHT = 6;
43  EXIT_STRAIGHT_TOP = 3;
44  EXIT_STRAIGHT_LEFT = 4;
45  EXIT_STRAIGHT_BOTTOM = 1;
46  EXIT_STRAIGHT_RIGHT = 2;
47
48  %clear local next variables
49  street_crossroad_next = ones(6,6)*EMPTY_STREET;
50  crossroad_speed_next = zeros(6,6);
51  crossroad_exit_next = zeros(6,6);
52  trace_left_next = ones(4,8)*EMPTY_STREET;
53  trace_left_speed_next = zeros(4,8);
54  trace_right_direction_next = ones(4,8)*NO_EXIT_YET;
55
56  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57  %set traffic light
58  %trafficlight = zeros(12,1) for car and pedestrians: red
59  trafficlight = settrafficlight(localphase, aheadphase, turnphase, pedestrian_density
        );
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61  %pedestrians
62  for k = 1:4
63      if (rand(1) <= pedestrian_density )
64          pedestrian_bucket(2,k) = 1;
65      end
66      if (( street_outwards(k,2) == EMPTY_STREET || street_outwards(k,2) == PEDESTRIAN
            ) && ...
67              pedestrian_bucket(2,k) > 0 && trafficlight(1+(k-1)*3,1)==1 )
```

```
68          street_outwards_next(k,2) = PEDESTRIAN;
69          outwards_speed_next(k,2) = 0;
70          pedestrian_bucket(2,k) = 0;
71      elseif ( street_outwards(k,2) == PEDESTRIAN)
72          street_outwards_next(k,2) = EMPTY_STREET;
73          outwards_speed_next(k,2) = 0;
74      end
75  end
76
77  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78  %car in front of crossroad and initializing direction
79
80  for k = 1:4
81      for l=1:STREET_INTERSECTION+1
82          %initializing randomly directions
83          if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==NO_EXIT_YET)
84              u=rand(1);
85              %if it goes left
86              if ( u < ((1-pahead)/2))
87                  trace_right_direction(k,l) = EXIT_LEFT;
88                  %if it goes ahead
89              elseif ( u <= ((1+pahead)/2))
90                  trace_right_direction(k,l) = k;
91
92                  %if it goes right
93              else
94                  trace_right_direction(k,l) = EXIT_RIGHT;
95
96              end
97          end
98
99          %take cars with EXIT_LEFT waiting into trace_left if space is free
100         if (street_inwards(k,l) == CAR && trace_right_direction(k,l)==EXIT_LEFT)
101             if(trace_left(k,1) == EMPTY_STREET)
102                 trace_left_next(k,1) = CAR;
103                 trace_left_speed_next(k,1) = inwards_speed(k,l);
104             else
105                 street_inwards_next(k,l) = CAR;
106                 inwards_speed_next(k,l) = 0;
107                 trace_right_direction_next(k,l)=EXIT_LEFT;
108             end
109         end
110
111         %for inwards
112         if (street_inwards(k,l) == CAR && trace_right_direction(k,l)~=EXIT_LEFT)
113             gap = crosslight_measure_gap(-k, l, trace_right_direction(k,l) ,
                    street_crossroad, ...
114                 street_outwards, street_outwards_next, 1, street_inwards,
                        street_inwards_next, trafficlight(3*k,1), ...
115                 EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                        EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
                        EMPTY_STREET);
116             v = schreckenberg(inwards_speed(k,l),gap,dawdleProb);
117             if(l == 1)
118                 inwards_gaps(1,k) = gap;
119             end
120             if (l+v<=STREET_INTERSECTION+1)
121                 street_inwards_next(k,l+v) = CAR;
```

```
122                 inwards_speed_next(k,l+v) = v;
123                 trace_right_direction_next(k,l+v) = trace_right_direction(k,l);
124             else
125                 ni = -k;
126                 nj = STREET_INTERSECTION+1;
127                 q = 1;
128                 while(q <= l+v-(STREET_INTERSECTION+1))
129                     if(ni > 0 || nj == STREET_INTERSECTION+1)
130                         [ni, nj] = crosslight_next_ij(ni, nj, trace_right_direction(
                                k,l) , ...
131                             EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,
                                    EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
                                    EXIT_STRAIGHT_RIGHT);
132                     else    %we are already in street_outwards
133                         %ni = ni;
134                         nj = nj+1;
135                     end
136                     q = q+1;
137                 end
138                 if (ni > 0)
139                     street_crossroad_next(ni,nj) = CAR;
140                     crossroad_speed_next(ni,nj) = v;
141                     crossroad_exit_next(ni,nj) = trace_right_direction(k,l);
142                 else
143                     street_outwards_next(-ni,nj) = CAR;
144                     outwards_speed_next(-ni,nj) = v;
145                 end
146             end
147         end
148
149         %for trace_left
150         if (trace_left(k,l) == CAR)
151             gap = crosslight_measure_gap(-k, l,EXIT_LEFT , street_crossroad , ...
152                 street_outwards, street_outwards_next, 1, trace_left,
                        trace_left_next, trafficlight(2+3*(k-1),1), ...
153                 EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                        EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
                        EMPTY_STREET);
154             v = schreckenberg(trace_left_speed(k,l),gap,dawdleProb);
155             if (l+v<=STREET_INTERSECTION+1)
156                 trace_left_next(k,l+v) = CAR;
157                 trace_left_speed_next(k,l+v) = v;
158             else
159                 ni = -k;
160                 nj = STREET_INTERSECTION+1;
161                 q = 1;
162                 while(q <= l+v-(STREET_INTERSECTION+1))
163                     if(ni > 0 || nj == STREET_INTERSECTION+1)
164                         [ni, nj] = crosslight_next_ij(ni, nj, EXIT_LEFT, ...
165                             EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,
                                    EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
                                    EXIT_STRAIGHT_RIGHT);
166                     else    %we are already in street_outwards
167                         %ni = ni;
168                         nj = nj+1;
169                     end
170                     q = q+1;
171                 end
```

```
172                 if (ni > 0)
173                     street_crossroad_next(ni,nj) = CAR;
174                     crossroad_speed_next(ni,nj) = v;
175                     crossroad_exit_next(ni,nj) = EXIT_LEFT;
176                 else
177                     street_outwards_next(-ni,nj) = CAR;
178                     outwards_speed_next(-ni,nj) = v;
179                 end
180             end
181         end
182     end
183 end
184
185 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186 %car in crossroad
187
188 for i = 1:6
189     for j = 1:6
190         if (street_crossroad(i,j) == CAR)
191             gap = crosslight_measure_gap(i, j,crossroad_exit(i,j), street_crossroad,
                      ...
192                 street_outwards, street_outwards_next, 0, street_inwards,
                        street_inwards_next, trafficlight(1+3*(k-1),1), ...
193                 EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                        EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION,
                        EMPTY_STREET);
194             v = schreckenberg(crossroad_speed(i,j),gap,dawdleProb);
195             ni = i;
196             nj = j;
197             q = 1;
198             while(q <= v)
199                 if(ni > 0)
200                     [ni, nj] = crosslight_next_ij(ni, nj, crossroad_exit(i,j), ...
201                         EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                                EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
202                 else    %we are already in street_outwards
203                     %ni = ni;
204                     nj = nj+1;
205                 end
206                 q = q+1;
207             end
208             if (ni > 0)
209                 street_crossroad_next(ni,nj) = CAR;
210                 crossroad_speed_next(ni,nj) = v;
211                 crossroad_exit_next(ni,nj) = crossroad_exit(i,j);
212             else
213                 street_outwards_next(-ni,nj) = CAR;
214                 outwards_speed_next(-ni,nj) = v;
215             end
216         end
217     end
218 end
219
220 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
221 %car outwards
222
223 for k = 1:4
224     for l = 1:STREET_INTERSECTION
```

```
225         %outwards street
226         e = 1;
227         while (e <= 5 && street_outwards(k,l+e) == EMPTY_STREET &&
                street_outwards_next(k,l+e) == EMPTY_STREET)
228             e = e + 1;
229         end
230         gap = e - 1;
231         v = schreckenberg(outwards_speed(k,l), gap, dawdleProb);
232         if(street_outwards(k,l) == CAR)
233             street_outwards_next(k,l+v) = CAR;
234             outwards_speed_next(k,l+v) = v;
235         end
236     end
237 end
238
239 end
```

Listing 10: crosslight-measure-gap.m

```
1  function [ gap ] = crosslight_measure_gap(i, j, direction, street_crossroad, ...
2      street_outwards, street_outwards_next, inwards, street_inwards,
           street_inwards_next, traffic_light, ...
3      EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,EXIT_STRAIGHT_BOTTOM,
           EXIT_STRAIGHT_RIGHT, STREET_INTERSECTION, EMPTY_STREET)
4  %crosslight_measure_gap this function will measure the gap to the next car
5  %in a crosslight
6
7  e = 1;
8  iterate = 1;
9  ni = i;
10 nj = j;
11 while (e <= 5 && iterate)
12     if((ni < 0 && nj == STREET_INTERSECTION+1 && inwards) || ni > 0)
13         [ni, nj] = crosslight_next_ij(ni, nj, direction, ...
14             EXIT_LEFT,EXIT_RIGHT,EXIT_STRAIGHT_TOP,EXIT_STRAIGHT_LEFT,
                   EXIT_STRAIGHT_BOTTOM,EXIT_STRAIGHT_RIGHT);
15     else
16         %ni = ni;
17         nj = nj+1;
18     end
19     if(ni > 0)
20         inwards = 0;
21         if(street_crossroad(ni,nj) == EMPTY_STREET)
22             e = e + 1;
23         else
24             iterate = 0;
25         end
26         if((direction == EXIT_LEFT || direction == EXIT_RIGHT) && e > 2)  %limit
               speed inside the crossection
27             e = 2;
28             iterate = 0;
29         end
30     else
31         if(inwards)
32             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
```

```
33                  if(traffic_light && street_inwards(-ni,nj) == EMPTY_STREET &&
                        street_inwards_next(-ni,nj) == EMPTY_STREET)  %% traffic_light
                        green and street empty
34                      e = e + 1;
35                  else
36                      iterate = 0;
37                  end
38              else
39                  if(street_inwards(-ni,nj) == EMPTY_STREET && street_inwards_next(-ni
                        ,nj) == EMPTY_STREET)
40                      e = e + 1;
41                  else
42                      iterate = 0;
43                  end
44              end
45          else
46              if(street_outwards(-ni,nj) == EMPTY_STREET && street_outwards_next(-ni,
                    nj) == EMPTY_STREET)
47                  e = e + 1;
48              else
49                  iterate = 0;
50              end
51          end
52      end
53  end
54  gap = e - 1;
55
56  end
```

Listing 11: crosslight-next-ij.m

```
1  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)
2  %crosslight_next_ij this function will return the next value for i and j
3  %which a car with a given direction and i j coordinates will have
4
5  switch(direction)
6      case EXIT_LEFT
7          if(i == 1 && j == 3)
8              ni = 2;
9              nj = 3;
10         elseif(i == 2 && j == 3)
11             ni = 3;
12             nj = 4;
13         elseif(i == 3 && j == 4)
14             ni = 4;
15             nj = 5;
16         elseif(i == 4 && j == 5)
17             ni = 5;
18             nj = 6;
19         elseif(i == 5 && j == 6)
20             ni = -4;
21             nj = 1;
22         elseif(i == 4 && j == 1)
23             ni = 4;
24             nj = 2;
25         elseif(i == 4 && j == 2)
26             ni = 3;
```

```
27              nj = 3;
28          elseif(i == 3 && j == 3)
29              ni = 2;
30              nj = 4;
31          elseif(i == 2 && j == 4)
32              ni = 1;
33              nj = 5;
34          elseif(i == 1 && j == 5)
35              ni = -1;
36              nj = 1;
37          elseif(i == 6 && j == 4)
38              ni = 5;
39              nj = 4;
40          elseif(i == 5 && j == 4)
41              ni = 4;
42              nj = 3;
43          elseif(i == 4 && j == 3)
44              ni = 3;
45              nj = 2;
46          elseif(i == 3 && j == 2)
47              ni = 2;
48              nj = 1;
49          elseif(i == 2 && j == 1)
50              ni = -2;
51              nj = 1;
52          elseif(i == 3 && j == 6)
53              ni = 3;
54              nj = 5;
55          elseif(i == 3 && j == 5)
56              ni = 4;
57              nj = 4;
58          elseif(i == 4 && j == 4)
59              ni = 5;
60              nj = 3;
61          elseif(i == 5 && j == 3)
62              ni = 6;
63              nj = 2;
64          elseif(i == 6 && j == 2)
65              ni = -3;
66              nj = 1;
67          elseif(i < 0)   %here I assume the car is in the last position of the
                inmwards street
68              if(i == -1)
69                  ni = 1;
70                  nj = 3;
71              elseif(i == -2)
72                  ni = 4;
73                  nj = 1;
74              elseif(i == -3)
75                  ni = 6;
76                  nj = 4;
77              elseif(i == -4)
78                  ni = 3;
79                  nj = 6;
80              end
81          end
82      case EXIT_RIGHT
83          if(i == 1)
```

```
84              if(j == 1)
85                  ni = -2;
86                  nj = 1;
87              else
88                  ni = -1;
89                  nj = 1;
90              end
91          elseif(i == 6)
92              if(j == 1)
93                  ni = -3;
94                  nj = 1;
95              else
96                  ni = -4;
97                  nj = 1;
98              end
99          elseif(i == -1)
100              ni = 1;
101              nj = 1;
102         elseif(i == -2)
103              ni = 6;
104              nj = 1;
105         elseif(i == -3)
106              ni = 6;
107              nj = 6;
108         elseif(i == -4)
109              ni = 1;
110              nj = 6;
111         end
112     case EXIT_STRAIGHT_TOP
113         if(i > 0)
114             nj = j;
115             ni = i-1;
116             if(ni < 1)
117                 ni = -EXIT_STRAIGHT_BOTTOM;
118                 nj = 1;
119             end
120         elseif(i == -EXIT_STRAIGHT_TOP)  %%check if it comes from BOTTOM
121             nj = 5;
122             ni = 6;
123         else
124             ni = i;
125             nj = j+1;
126         end
127     case EXIT_STRAIGHT_BOTTOM
128         if(i > 0)
129             nj = j;
130             ni = i+1;
131             if(ni > 6)
132                 ni = -EXIT_STRAIGHT_TOP;
133                 nj = 1;
134             end
135         elseif(i == -EXIT_STRAIGHT_BOTTOM)
136             nj = 2;
137             ni = 1;
138         else
139             ni = i;
140             nj = j+1;
141         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
%
%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 numeber 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');
%ylim([0 0.5]);


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');
%ylim([0 0.5]);


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');
%ylim([0 0.5]);

fig6 = figure(6);
% hold on;
% for di=1:length(d)
%     for pdi=1:length(pd)
%         plot3(pd(pdi), d(di) ,avSpeed(pdi,di), 'x');
%     end
```

```
% end

% imagesc(map);
% hold on;
% view(0,90);
surf(pd,d,avSpeed);

% plot3(pd, d ,avSpeed, 'x');
% set(gca,'FontSize',16);
title('Traffic Dynamics','FontWeight','bold');
xlabel('pedestrian density');
ylabel('traffic density');
zlabel('average speed');


fig7 = figure(7);
surf(pd,d,avFlow);
title('Traffic Dynamics','FontWeight','bold');
xlabel('pedestrian density');
ylabel('traffic 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
70                  map(mapI_m+street_length+4,mapI_n+i) = street_inwards(tI_m+2,tI_n+i)
                        ; % left, inwards
71                  map(mapI_m+2*street_length+7-i,mapI_n+street_length+4) =
                        street_inwards(tI_m+3,tI_n+i); % bottom, inwards
72                  map(mapI_m+street_length+3,mapI_n+2*street_length+7-i) =
                        street_inwards(tI_m+4,tI_n+i); % right, inwards
73              end
74              for i = 1:3
75                  map(mapI_m+street_length+1-i,mapI_n+street_length+4) =
                        street_outwards(tI_m+1,tI_n+i);  % top, outwards
76                  map(mapI_m+street_length+3,mapI_n+street_length+1-i) =
                        street_outwards(tI_m+2,tI_n+i);  % left, outwards
77                  map(mapI_m+street_length+6+i,mapI_n+street_length+3) =
                        street_outwards(tI_m+3,tI_n+i);  % bottom, outwards
78                  map(mapI_m+street_length+4,mapI_n+street_length+6+i) =
                        street_outwards(tI_m+4,tI_n+i);  % right, outwards
79              end
80              %filling fields for optics
81              map(mapI_m+street_length+1-4,mapI_n+street_length+3) = EMPTY_STREET;
                    % top, left
82              map(mapI_m+street_length+1-4,mapI_n+street_length+4) = EMPTY_STREET;
                    % top, right
83              map(mapI_m+street_length+3,mapI_n+street_length+1-4) = EMPTY_STREET;  %
                    left, top
84              map(mapI_m+street_length+4,mapI_n+street_length+1-4) = EMPTY_STREET;  %
                    left, bottom
85              map(mapI_m+street_length+6+4,mapI_n+street_length+3) = EMPTY_STREET;  %
                    bottom, left
86              map(mapI_m+street_length+6+4,mapI_n+street_length+4) = EMPTY_STREET;  %
                    bottom, right
87              map(mapI_m+street_length+3,mapI_n+street_length+6+4) = EMPTY_STREET;  %
                    right, top
88              map(mapI_m+street_length+4,mapI_n+street_length+6+4) = EMPTY_STREET;  %
                    right, bottom
89          end
90
91          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92          %write crossing into map
93
94          %check if intersection is a crossing with priority to the right
95          if ( config(a,b) == 1 )
96              %define index starting points for this crossroad
97              pI_m = (a - 1) * 6;
98              pI_n = (b - 1) * 6;
99              pIl_n = (b - 1) * 12;   % index for light
100             pIt_m = (a - 1) * 4;    % m-index for trace left
101             pIt_n = (b - 1) * 8;    % n-index for trace left
102             %write crossroad into map
103             map(mapI_m+street_length+1:mapI_m+street_length+6,...
104                 mapI_n+street_length+1:mapI_n+street_length+6) = ...
105                 street_crossroad(pI_m+1:pI_m+6,pI_n+1:pI_n+6);
106
107             %traffic lights
108             GREEN_LIGHT = 1.3;
109             RED_LIGHT = 1.6;
110             light(light==1) = GREEN_LIGHT;
111             light(light==0) = RED_LIGHT;
```

```
112
113             map(mapI_m+street_length-2, mapI_n+street_length+1) = light(a, pIl_n
                    +0*3+3); % top, inwards
114             map(mapI_m+street_length-2, mapI_n+street_length+4) = light(a, pIl_n
                    +0*3+2); % top, trace_left
115             map(mapI_m+street_length-1, mapI_n+street_length+6) = light(a, pIl_n
                    +0*3+1); % top, pedestrians
116
117             map(mapI_m+street_length+1, mapI_n+street_length-1) = light(a, pIl_n
                    +1*3+1); % left, pedestrians
118             map(mapI_m+street_length+3, mapI_n+street_length-2) = light(a, pIl_n
                    +1*3+2); % left, trace_left
119             map(mapI_m+street_length+6, mapI_n+street_length-2) = light(a, pIl_n
                    +1*3+3); % left, inwards
120
121             map(mapI_m+street_length+6+2, mapI_n+street_length+1) = light(a, pIl_n
                    +2*3+1); % bottom, pedestrians
122             map(mapI_m+street_length+6+3, mapI_n+street_length+3) = light(a, pIl_n
                    +2*3+2); % bottom, trace_left
123             map(mapI_m+street_length+6+3, mapI_n+street_length+6) = light(a, pIl_n
                    +2*3+3); % bottom, inwards
124
125             map(mapI_m+street_length+1, mapI_n+street_length+6+3) = light(a, pIl_n
                    +3*3+3); % right, inwards
126             map(mapI_m+street_length+4, mapI_n+street_length+6+3) = light(a, pIl_n
                    +3*3+2); % right, trace_left
127             map(mapI_m+street_length+6, mapI_n+street_length+6+2) = light(a, pIl_n
                    +3*3+1); % right, pedestrians
128
129             %trace left
130             trace_left_length = STREET_INTERSECTION+1;
131             for i = 1:trace_left_length
132                 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
133                 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
134                 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
135                 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
136             end
137
138             %write streets into map
139             for i = 1:street_length
140                 map(mapI_m+i,mapI_n+street_length+2) = street_inwards(tI_m+1,tI_n+i)
                        ; % top, inwards
141                 map(mapI_m+street_length+5,mapI_n+i) = street_inwards(tI_m+2,tI_n+i)
                        ; % left, inwards
142                 map(mapI_m+2*street_length+7-i,mapI_n+street_length+5) =
                        street_inwards(tI_m+3,tI_n+i); % bottom, inwards
143                 map(mapI_m+street_length+2,mapI_n+2*street_length+7-i) =
                        street_inwards(tI_m+4,tI_n+i); % right, inwards
144                 map(mapI_m+street_length+1-i,mapI_n+street_length+5) =
                        street_outwards(tI_m+1,tI_n+i);  % top, outwards
145                 map(mapI_m+street_length+2,mapI_n+street_length+1-i) =
                        street_outwards(tI_m+2,tI_n+i);  % left, outwards
146                 map(mapI_m+street_length+6+i,mapI_n+street_length+2) =
                        street_outwards(tI_m+3,tI_n+i);  % bottom, outwards
```

```
147                 map(mapI_m+street_length+5,mapI_n+street_length+6+i) =
                        street_outwards(tI_m+4,tI_n+i);  % right, outwards
148             end
149         end
150
151     end
152 end
153
154 %%illustrate trafic situation (now, not of next time step)
155 % fig1 = figure(1);
156 % imagesc(map);
157 % load('colormap2', 'mycmap')
158 % set(fig1, 'Colormap', mycmap)
159 % titlestring = sprintf('Density = %g',car_density);
160 % title(titlestring);
161 % drawnow;
162
163 end
```

References