Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Project Report

Traffic Dynamics
Comparison of single-lane and two-lane roundabouts

Dominik Eugster & Roman Fuchs

Zurich
December 2010

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese Gruppenarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen-Hilsmittel verwenden habe, und alle Stellen, die wörtlich oder sinngemäss aus veröffentlichen Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Gruppenarbeit nicht, auch nicht auszugsweise, bereits für andere Prüfung ausgefertigt wurde.

Dominik Eugster

Roman Fuchs

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.

Dominik Eugster

Roman Fuchs

Contents

1 Individual contributions

Both have contributed to the project equally.

2 Introduction and Motivation

More and more roundabouts replace traditional crossroads. One reason is because of higher traffic flow, another advantage of roundabouts is that they reduce collisions and injuries, except for cyclists. Over the years different types of roundabouts have been constructed, for example, magic roundabouts, raindrop roundabouts and turbo roundabouts. [2]

In last semester course, Wood and Bücheler [3] considered traffic flow of crossroads compared to roundabouts. They used the Nagel-Schreckenberg model to determine the velocity of the agents. One of their result was that (single-lane) roundabouts have a much higher throughput at every traffic density. Wood and Bücheler concluded that their results justify the increasing popularity of roundabouts in the last years. Moreover, two-lane roundabouts gain more popularity as well. This leads to the questions how to model a two-lane roundabout, in which case it is superior to a single-lane roundabout and if the results are significantly different.

Wang and Ruskin [1] modeled different driver behaviour (moderate, conservative, urgent and radical) on how vehicles enter a roundabout. Furthermore, they used two different patterns how cars arrive to a two-lane roundabout. In pattern A, left turning vehicles used the left lane only whereas right turning and straight through vehicles share the right lane. In pattern B, straight through vehicles used both lanes.

The goal of this project was to implement a simple model to compare a two-lane against a single-lane roundabout. Additionally, the superiorness of two-lane roundabouts with respect to traffic distribution of the routes joined to are investigated. Finally the implications of the two patterns of Wang and Ruskin were considered.

3 Description of the Model

A cellular automaton is used to model a two-lane roundabout in a dense traffic situation. Subsequently the geometry, the cars and the rules are described.

3.1 Geometry

The roundabout consists of two lanes of 20 cells each and four legs, also of 20 cells each, are connected to it. Each leg has one lane to drive away and two lanes leading to the roundabout. Two different patterns of lane allocation are investigated. In pattern A, the left lane is reserved for left turning cars whereas cars turning right or driving straight through share the right lane. In pattern B, the cars driving straight through can use the left lane as well whereas the turning cars have to use the corresponding lane.

The cells are numbered such that the cars drive to descending cell numbers. Hence, the cell in front of a car is updated before the car is possibly driving.

A sketch of the roundabout is given in figure 1.

3.2 Cars

The cars drive to the next cell if it is free and they stop otherwise, i.e. they queue. Hence, the cars can jump to the next adjacent cell at maximum which corresponds to slow velocity or dense traffic situations.

The car's strategy is given at the start of the simulation. Accordingly, they approach the roundabout on the corresponding lane as explained above. Furthermore they start at a random time and the model is processed until all cars have arrived at their destinations. Hence, no periodic boundaries are applied for the cars.

The cars' sources and destinations can be specified. Hence, it is possible to study different types of junctions with different frequencies on the roads.

3.3 Rules

The rules at the roundabout are set heuristically based on usual traffic laws:

- A car enters the roundabout if there is enough space. To enter the outer lane, the cell in front and one cell on the left has to be free (see figure 2(a)). Entering to the inner lane requires that the same cells on the inner lane must be free as well (see figure 2(b)).
- A car leaves the roundabout if the road is free. A car on the inner lane has to change to the outer lane some time before it leaves the roundabout (see

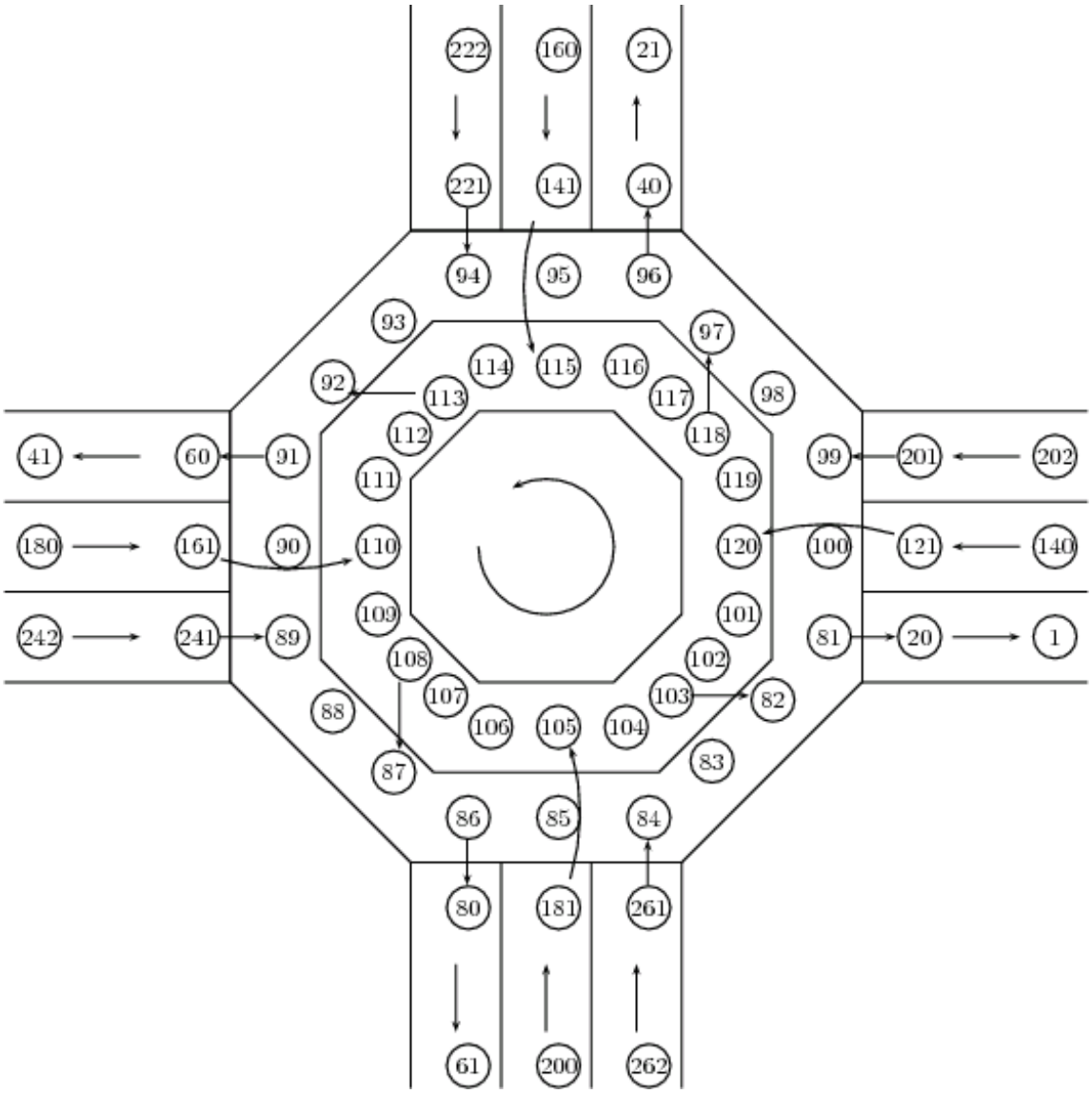
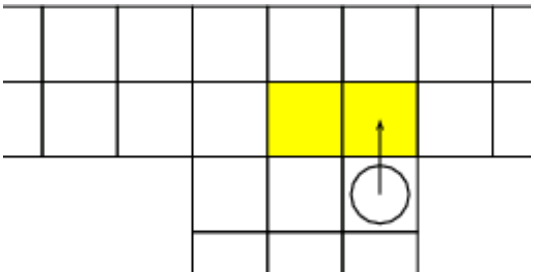
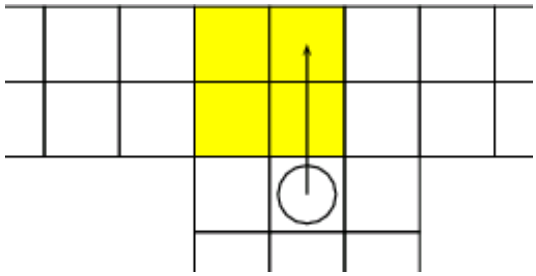


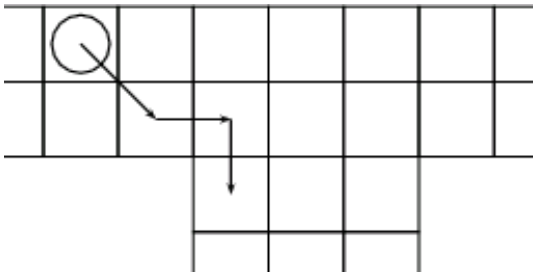
Figure 1: Sketch of the modeled roundabout and overview of cell numbering. The cars drive along descending numbers. Cars in the inner lane change to the outer lane at the arrows if they want to leave the roundabout by the next road.



(a) Entering the roundabout to the outer lane. Yellow cells must be free.



(b) Entering the roundabout to the inner lane. Yellow cells must be free.



(c) Leaving roundabout from inner lane.

Figure 2: Rules for entering and leaving the roundabout.

figure 2(c)). For simplicity, the cars have to change lane two cells before their exit road.

- A car on the inner lane has the right of way on the cars on the outer lane. This rule corresponds to the traffic law but may seem not very intuitive.

4 Implementation

The code was written in MATLAB. Its intuitive vector computation makes it easier to understand the code although it would be preferrable to write object-oriented code if the model is further developped.

4.1 Files

The program is structured in 10 functions which are briefly described:

main.m is the main file of the program. It contains the parameter definitons, calls the initialization and includes the time loop.

initCars.m initializes the data of all cars. Start times are distributed uniformly, origins and destinations are randomly generated according to the specified strategy. For generating the random numbers, standard MATLAB routine rand() is used.

initGeom.m sets the geometry of the roundabout. In this case, the size of the roundabout is static. One could imagine an implementation with dynamic roundabout size.

moveCarFromCross.m moves the cars that have already reached their destination route.

moveCarInnerRoundabout.m moves the cars driving in the inner roundabout lane

moveCarOuterRoundabout.m moves the cars driving in the outer roundabout lane.

moveCarFromCross.m moves the cars driving towards or entering the roundabout.

visualize.m draws the geometry with the occupied cases and generates a movie. The visualization can be suppressed to increase speed.

statistics.m analyzes the instantaneous situation and generates statistics on queue length and driving time.

showstats.m shows the final statistics of queuing in a plot and the running times in a histogram. The average values are also indicated.

4.2 Statistics

A car is defined as queuing if it is stopped two times consecutively. The running duration is computed and stored for each car. For calculating the average, the arithmetic mean is applied.

4.3 Traffic Rules

By calling the moving functions in the right order, the right of way is assured. Changing the ordering will result in changing the right of way.

5 Simulation Results and Discussion

5.1 Simulation Scenarios

Three scenarios are analyzed. Once a roundabout with equally important streets, then two cases with different road sizes.

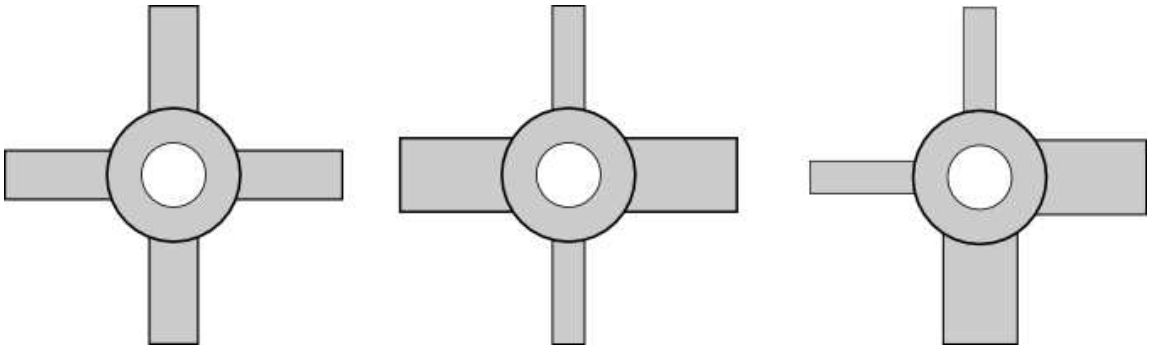


Figure 3: the 3 test scenarios

For all scenarios, three cases are taken into account:

(a) All cars take the outer lane

(b) Cars turning left take the inner lane

(c) Cars turning left take the inner lane, 50% of the cars going ahead take the inner lane

The first case may be considered as a single-lane roundabout.
The number of iterations assures that the system reaches a stable situation.

5.2 Simulation Results

5.2.1 Scenario 1

This scenario was simulated for a frequency of 8000 cars in 5000 timesteps.

In this scenario, we can see an improvement of the mean runtime if we choose a two-lane roundabout (see figure 4). Single-lane roundabout blocks at this frequence whereas there is still a flow for the double-lane roundabout.
We oberserve as well the fact, that people driving straight should preferrably all take the outer lane.

5.2.2 Scenario 2

This scenario was simulated for a frequency of 7500 cars in 5000 timesteps.

We observe results that are similar to the ones observed in scenario 1 (see figure 5).

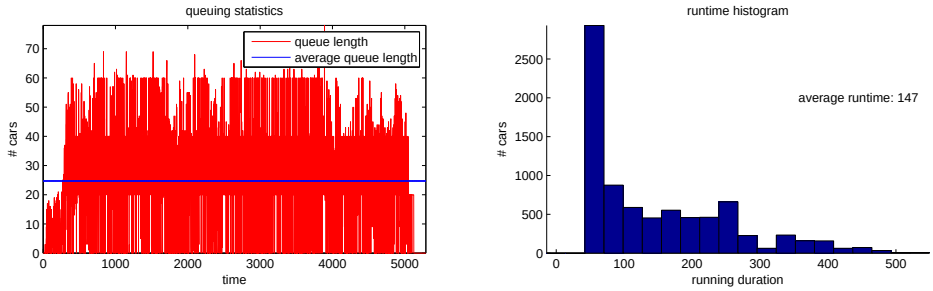
5.2.3 Scenario 3

This scenario was simulated for a frequency of 7000 cars in 5000 timesteps.

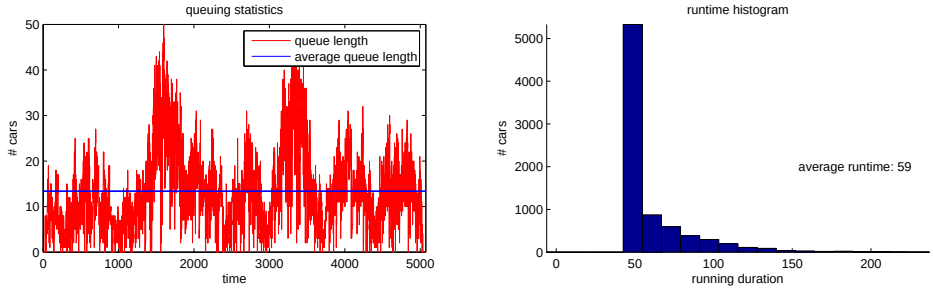
We oberserve that the flux of a single-lane roundabout is stable whereas the two versions of two-lane roundabouts 'crash' (see figure 6). This phenomena is probably due to the fact that cars entering the inner lane do not enter the roundabout even if the car blocking him is leaving the roundabout. This reflects a passive, non-confident behaviour or the case where nobody is using the turn indicator.
In this scenario it would probably be better to use a single-lane roundabout with a separate turning lane for cars turning right.

5.3 Conclusions

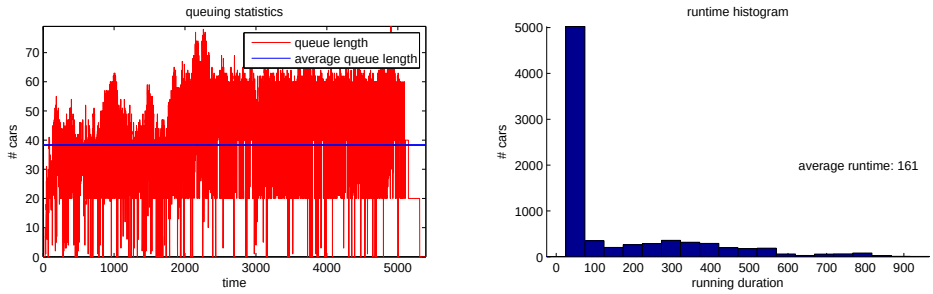
We conclude that two-lane roundabouts do not always augment the capacity of a crossroad. Although in most of the cases this is the case.
The attribution of the lanes practiced in Switzerland is in all our cases optimal. In practice, many drivers turning left take the outer lane which slows up the traffic. Another problem is the fact that cars driving in the inner lane have the priority over the ones driving on the outer lane. this is not enough known by Swiss drivers and hence people tend to drive on the outer lane by fear of an accident.



(a) Case 1.a

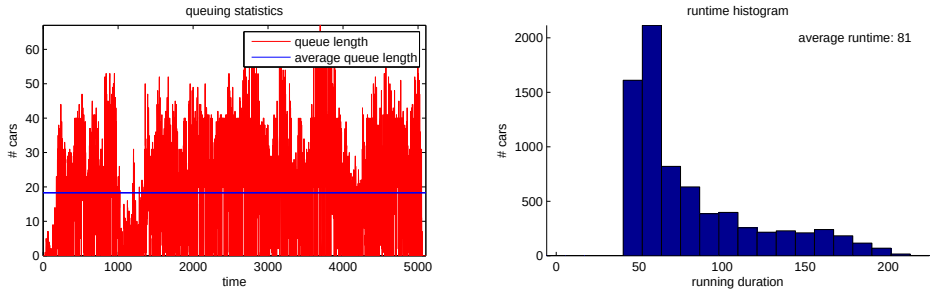


(b) Case 1.b

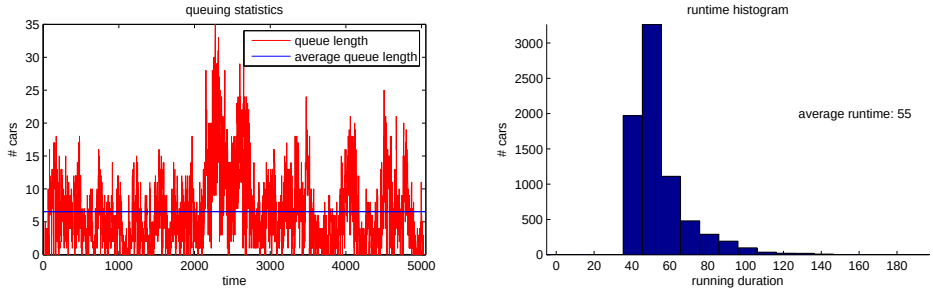


(c) Case 1.c

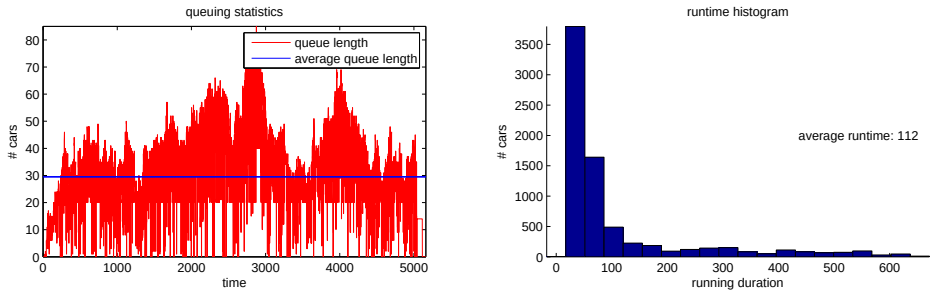
Figure 4: Roundabout with equally important streets.



(a) Case 2.a

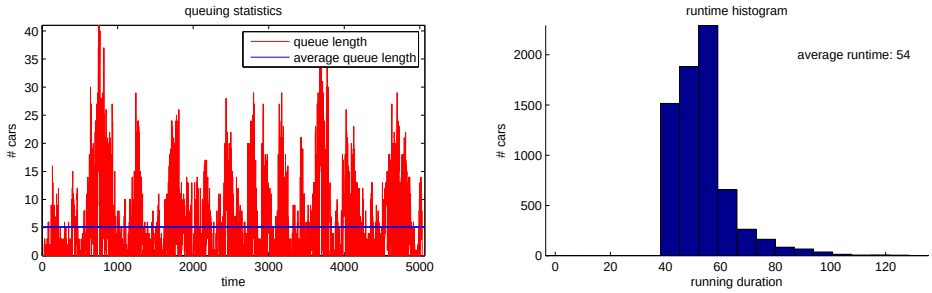


(b) Case 2.b

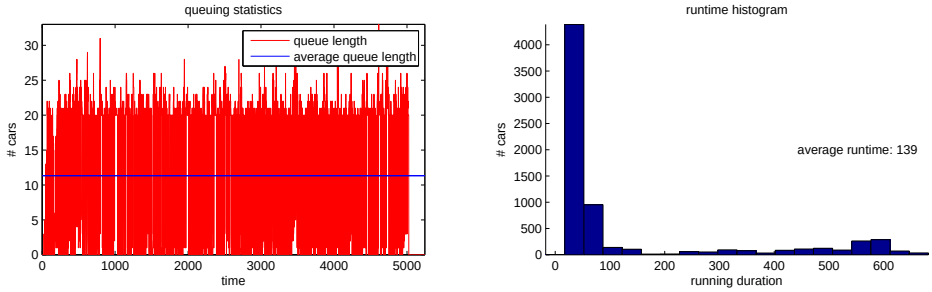


(c) Case 2.c

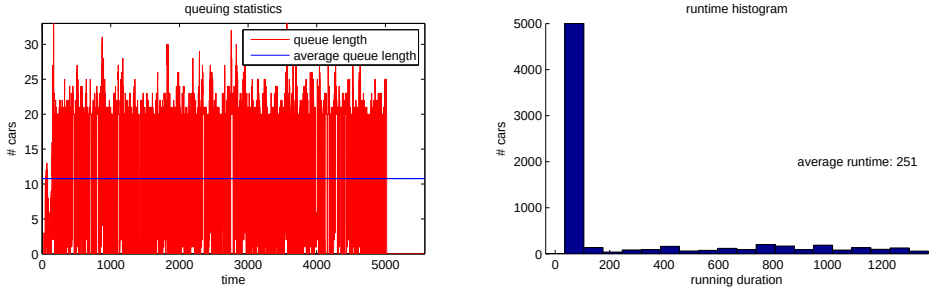
Figure 5: Roundabout with a main road straight through.



(a) Case 3.a



(b) Case 3.b



(c) Case 3.c

Figure 6: Roundabout with a main road around a corner.

6 Summary and Outlook

6.1 Summary

In this project a two-lane roundabout was modelled and the capacities of single- and two-lane roundabouts were compared. It was concluded that the frequencies of the different roads have to be taken into account to decide whether a two-lane roundabout augments the capacity. Laws and geometries of roundabouts in Switzerland do correspond to the optimal driver behavior.

6.2 Outlook

This program is able to simulate a very restricted number of scenarios. Here a few that could be implemented in a following project:

- user-defined geometries with more or less routes
- different lengths of the incoming roads
- dynamic cell size and speed considerations
- systems with more than one roundabout
- car frequencies varying over the time
- many different types of roundabout have been developed, i.e. magic roundabouts, turbo roundabouts, raindrop roundabouts. It could be interesting to investigate the different types.

7 Source Code

Listing 1: initCars.m

```
function cars = initCars( freq, totalCars, tmax, strategy)
%INITCARS Computes the initial parameters fo all cars.

cars=zeros(totalCars, 9);
% (x,1): actual position: cases.ID
% (x,2): last position: cases.ID
% (x,3): origin: [1:4]
% (x,4): destination [1:4]
% (x,5): queue: 0=not queuing, 1=queuing
% (x,6): Start time of agent
% (x,7): state: 1=not yet started, 0=driving, 2=finished
% (x,8): start: first case driven
% (x,9): arrival time

%%% Initialization of starting times:%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
randnbs=floor(tmax*rand(totalCars, 1));
randnbs=sort(randnbs);
cars(:,6)=randnbs;

%%% Initialization of states %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cars(:,7)=ones(totalCars,1);

%%% Initialization of origins and destinations%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

randnbs=100*rand(totalCars,3);
for i=1:totalCars
    if randnbs(i,1)<=freq(1,1) % Origin 1
        cars(i,3)=1;
        if randnbs(i,2)<= freq(1,2)
            cars(i,4)=2;
            cars(i,8)=220;
        elseif randnbs(i,2)<= freq(1,2)+freq(1,3)
            cars(i,4)=3;
            if(randnbs(i,3)<strategy(1))
                cars(i,8)=140; % initialization of first cases
            else cars(i,8)=220;
            end
        else
            cars(i,4)=4;
            if(randnbs(i,3)<strategy(2))
                cars(i,8)=140; % initialization of first cases
            else cars(i,8)=220;
```

```
43            end
44        end
45    elseif randnbs(i,1)<=freq(1,1)+freq(2,1)
46        cars(i,3)=2;
47        if randnbs(i,2)<= freq(2,2) %right turning
48            cars(i,4)=3;
49            cars(i,8)=240;
50        elseif randnbs(i,2)<= freq(2,2)+freq(2,3) %straight
51            cars(i,4)=4;
52            if(randnbs(i,3)<strategy(1)) cars(i,8)=160; % initialization of first
                  cases
53            else cars(i,8)=240;
54            end
55        else %left turn
56            cars(i,4)=1;
57            if(randnbs(i,3)<strategy(2)) cars(i,8)=160; % initialization of first
                  cases
58            else cars(i,8)=240;
59            end
60        end
61    elseif randnbs(i,1)<=freq(1,1)+freq(2,1)+freq(3,1)
62        cars(i,3)=3;
63        if randnbs(i,2)<= freq(1,2) %right turning
64            cars(i,4)=4;
65            cars(i,8)=260;
66        elseif randnbs(i,2)<= freq(3,2)+freq(3,3) %straight
67            cars(i,4)=1;
68            if(randnbs(i,3)<strategy(1)) cars(i,8)=180; % initialization of first
                  cases
69            else cars(i,8)=260;
70            end
71        else %left turn
72            cars(i,4)=2;
73            if(randnbs(i,3)<strategy(2)) cars(i,8)=180; % initialization of first
                  cases
74            else cars(i,8)=260;
75            end
76        end
77    else
78        cars(i,3)=4;
79        if randnbs(i,2)<= freq(4,2) %right turning
80            cars(i,4)=1;
81            cars(i,8)=280;
82        elseif randnbs(i,2)<= freq(4,2)+freq(4,3) %straight
83            cars(i,4)=2;
```

```
            if(randnbs(i,3)<strategy(1)) cars(i,8)=200; % initialization of first
                cases
            else cars(i,8)=280;
            end
        else %left turn
            cars(i,4)=3;
            if(randnbs(i,3)<strategy(2)) cars(i,8)=200; % initialization of first
                cases
            else cars(i,8)=280;
            end
        end
    end



end
    %cars
```

Listing 2: initGeom.m

```
function cases = initGeom(show)
%INITGEOM Creates the geography of the Roundabout
%   Detailed explanation goes here
% show: boolean, if the geometry should be shown.

cases=zeros(280,6); %contains information about the possible positions
% (x,1): occupied: car.ID if yes, 0 if empty
% (x,2): x-coordinate of center
% (x,3): y-coordinate of center

%% computation of coordinates
lane1=92:-4:16;
lane2=16:4:92;

% cases no. 1 to 80: routes from the roundabout away, counting from east counter-
    clockwise
% cells 20, 40, 60, 80 are the first cell after the roundabout, counting down
    when driving away.
cases(1:20,2)=lane1;
cases(1:20,3)=-4;
cases(21:40,2)=4;
cases(21:40,3)=lane1;
cases(41:60,2)=-lane1;
cases(41:60,3)=4;
cases(61:80,2)=-4;
cases(61:80,3)=-lane1;
```

```
25
26 % Outer lane of the roundabout, counting from west counter-clockwise (???)
27 v = [0 4 8 10.5 12]';
28 cases(81:85,2:3)=[sort(v,'descend'), [-4 -8 -10.5 -12 -12]'];
29
30 cases(86,2:3)=[-4, -12];
31 cases(87,2:3)=[-8, -10.5];
32 cases(88,2:3)=[-10.5, -8];
33 cases(89,2:3)=[-12,-4];
34 cases(90,2:3)=[-12, 0];
35
36 cases(91:100,2:3)=-cases(81:90,2:3);
37
38 % Inner lane of the roundabout, counting from west counter-clockwise
39 cases(101,2:3)=[9,-3.5];
40 cases(102,2:3)=[8,-6.3];
41 cases(103,2:3)=[6.3,-8];
42 cases(104,2:3)=[3.5,-9];
43 cases(105,2:3)=[0,-9];
44
45 cases(106,2:3)=[-3.5,-9];
46 cases(107,2:3)=[-6.3,-8];
47 cases(108,2:3)=[-8,-6.3];
48 cases(109,2:3)=[-9,-3.5];
49 cases(110,2:3)=[-9,0];
50
51 cases(111:120,2:3)=-cases(101:110,2:3);
52
53 % Left street to the roundabout
54 cases(121:140,2)=lane2;
55 cases(141:160,3)=lane2;
56 cases(161:180,2)=-lane2;
57 cases(181:200,3)=-lane2;
58
59 % Right street to the roundabout
60 cases(201:220,2)=lane2;
61 cases(201:220,3)=4;
62 cases(221:240,2)=-4;
63 cases(221:240,3)=lane2;
64 cases(241:260,2)=-lane2;
65 cases(241:260,3)=-4;
66 cases(261:280,2)=4;
67 cases(261:280,3)=-lane2;
68
69
70 %% Show the geometry
```

```
71 if(show)
72     plot(cases(:,2),cases(:,3),'o');
73     axis equal;
74 end
```

Listing 3: Main.m

```
1  % Main script
2  clear all; close all; clc;
3
4
5  %% Default parameters
6  %   freq: 4x4-matrix:
7  %     (x,1): percentage of total traffic coming from direction x
8  %     (x,2): percentage of traffic from direction x turning right
9  %     (x,3): percentage of traffic from direction x going straight
10 %     (x,4): percentage of traffic from direction x turning left
11
12 %%%%%%%%%% scenario 1
13 freq = ...
14     [25 33 34 33;...
15      25 33 34 33;...
16      25 33 34 33;...
17      25 33 34 33];
18 %%%%%%%%%% scenario 2
19 %  freq = ...
20 %     [35 15 70 15;...
21 %      15 40 20 40;...
22 %      35 15 70 15;...
23 %      15 40 20 40];
24 %%%%%%%%%% scenario 3
25 %  freq = ...
26 %     [35 70 15 15;...
27 %      35 15 15 70;...
28 %      15 20 40 40;...
29 %      15 40 40 20];
30
31 %
32 %   tmax: maximum number of iterations simulated
33 %
34 %   totalCars: number of totally simulated cars
35 %
36 %   strategy(1): percentage of cars driving straight taking the left lane
37 %   strategy(2): percentage of cars driving left taking the left lane
38 %
39 %   show: if 1, the occupied cases are drawn in each iteration and a movie
```

```
%   is generated

totalCars = 7500;
tmax = 5000;
strategy = [0;100];
show = 1;

%% Initialize geometry and cars
cases = initGeom(show);
cars = initCars( freq, totalCars, tmax, strategy);

%% initialize statistical variables
queue=[];
queuestats=[];
finalt=0; %duration of simulation

%% initialize movie
if(show)
    filmli = avifile('filmli.avi','compression','Cinepak','fps',6);
end
%% Time loop
 while(length(find(cars(:,7)==2))~=totalCars)%%% loop until all cars have reached
     their destination
    finalt=finalt+1;
% Assume that the cars' positions are in cars(:,2) (lastPosition).
% Move all cars in subroutines to a probably new position cars(:,1)
% (actualPosition).
% Then check, if these new positions are free. If so, then update.

    [cars, cases]=moveCarFromCross(cars, cases, finalt);
    [cars, cases] = moveCarOuterRoundabout(cars, cases);
    [cars, cases] = moveCarInnerRoundabout(cars, cases);
    [cars, cases]=moveCarToCross(cars, cases, finalt);
    queue = statistics(cars, cases, queue);
    if(show)
        filmli = visualize(cases, show, filmli, queue);
    end
 end

 %% analyze and show statistics
 queuestats=[queuestats;queue];
 runtimes=cars(:,9)-cars(:,6);
 if(show)
     filmli = close(filmli);
 end
showstats(finalt, queuestats, runtimes);
```

Listing 4: moveCarFromCross.m

```
function [cars, cases] = moveCarFromCross(cars, cases, time)

for caseID = 1 : 80
    if (caseID==1 ||caseID==21||caseID==41||caseID==61) %empty the exiting cars
        if cases(caseID,1)>0
            cars(cases(caseID,1),7)=2;
            cars(cases(caseID,1),9)=time;
            cases(caseID,1)=0;
        end
    elseif (cases(caseID,1) > 0) % if there is a car in this case number
        thisCar=cases(caseID,1);
        if(cases(caseID-1,1)==0) % if next case empty, drive forward
            cars(thisCar,2)=caseID; %last position
            cars(thisCar,1)=caseID-1; % actual position
            cars(thisCar,5)=0; %not queuing
            cases(caseID-1,1)=thisCar;
            cases(caseID,1)=0;
        elseif cars(thisCar,2)==cars(thisCar,1)
            cars(thisCar,5)=1; %queuing
        else
            cars(thisCar,2)=caseID; %not yet queuing
        end
    end

end
```

Listing 5: moveCarInnerRoundabout.m

```
function [cars, cases] = moveCarInnerRoundabout(cars, cases)
% MOVECARINNERROUNDABOUT Move all cars in the inner line of the roundabout
% cars: n x 7 - Matrix
%     (x,1): actual position: cases.ID
%     (x,2): last position: cases.ID
%     (x,3): origin: [1:4]
%     (x,4): destination [1:4]
%
% cases: n x 1 - Matrix
%     (x,1): occupied: car.ID if yes, 0 if empty

% Algorithm:
% ...
%
% Note: ???
% This algorithm implies that if the car cannot exit, it has to drive an
% additional turn.
```

```
18
19
20  for caseID = 101 : 120
21      if(cases(caseID) > 0) % if there is a car in this case number
22          thisCar = cases(caseID,1);
23          switch caseID
24              case 103
25                  % if car wants to drive to outer line here and there is free space
26                  if(cars(thisCar,4) == 1)
27                      if(cases(82) == 0)
28                          cars(thisCar,1) = 82; % new position
29                          cars(thisCar,2) = 103;% old position
30                          cases(82) = thisCar; % lock new cell
31                          cases(103) = 0; % free old cell
32                          cars(thisCar,5)=0;
33                      else
34                          if cars(thisCar,2)==cars(thisCar,1)
35                              cars(thisCar,5)=1;
36                          end
37                          cars(thisCar, 1)=cars(thisCar,2);
38                      end
39                  else % car wants to continue on inner line
40                      if(cases(102) == 0) % if cell is free, move forward
41                          cars(thisCar,1) = 102; % new position
42                          cars(thisCar,2) = 103;% old position
43                          cases(102) = thisCar; % lock new cell
44                          cases(103) = 0; % free old cell
45                          cars(thisCar,5)=0;
46                      else
47                          if cars(thisCar,2)==cars(thisCar,1)
48                              cars(thisCar,5)=1;
49                          end
50                          cars(thisCar, 1)=cars(thisCar,2);
51                      end
52                  end
53
54              case 118
55                  if(cars(thisCar,4) == 2)
56                      if(cases(97) == 0)
57                          cars(thisCar,1) = 97;
58                          cars(thisCar,2) = 118;
59                          cases(97) = thisCar;
60                          cases(118) = 0;
61                          cars(thisCar,5)=0;
62                      else
63                          if cars(thisCar,2)==cars(thisCar,1)
```

```
64                          cars(thisCar,5)=1;
65                      end
66                      cars(thisCar, 1)=cars(thisCar,2);
67                  end
68              else
69                  if(cases(117) == 0)
70                      cars(thisCar,1) = 117; % new position
71                      cars(thisCar,2) = 118;% old position
72                      cases(117) = thisCar; % lock new cell
73                      cases(118) = 0; % free old cell
74                      cars(thisCar,5)=0;
75                  else
76                      if cars(thisCar,2)==cars(thisCar,1)
77                          cars(thisCar,5)=1;
78                      end
79                      cars(thisCar, 1)=cars(thisCar,2);
80                  end
81              end
82
83          case 113
84              if(cars(thisCar,4) == 3)
85                  if(cases(92) == 0)
86                      cars(thisCar,1) = 92;
87                      cars(thisCar,2) = 113;
88                      cases(92) = thisCar;
89                      cases(113) = 0;
90                      cars(thisCar,5)=0;
91                  else
92                      if cars(thisCar,2)==cars(thisCar,1)
93                          cars(thisCar,5)=1;
94                      end
95                      cars(thisCar, 1)=cars(thisCar,2);
96                  end
97              else
98                  if(cases(112) == 0)
99                      cars(thisCar,1) = 112;
100                     cars(thisCar,2) = 113;
101                     cases(112) = thisCar;
102                     cases(113) = 0;
103                     cars(thisCar,5)=0;
104                 else
105                     if cars(thisCar,2)==cars(thisCar,1)
106                         cars(thisCar,5)=1;
107                     end
108                     cars(thisCar, 1)=cars(thisCar,2);
109                 end
```

```
110                 end
111
112             case 108
113                 if(cars(thisCar,4) == 4)
114                     if(cases(87) == 0)
115                         cars(thisCar,1) = 87;
116                         cars(thisCar,2) = 108;
117                         cases(87) = thisCar;
118                         cases(108)=0;
119                         cars(thisCar,5)=0;
120                     else
121                         if cars(thisCar,2)==cars(thisCar,1)
122                             cars(thisCar,5)=1;
123                         end
124                         cars(thisCar, 1)=cars(thisCar,2);
125                     end
126                 else
127                     if(cases(107) == 0)
128                         cars(thisCar,1) = 107;
129                         cars(thisCar,2) = 108;
130                         cases(107) = thisCar;
131                         cases(108) = 0;
132                         cars(thisCar,5)=0;
133                     else
134                         if cars(thisCar,2)==cars(thisCar,1)
135                             cars(thisCar,5)=1;
136                         end
137                         cars(thisCar, 1)=cars(thisCar,2);
138                     end
139                 end
140
141             case 101 % car has to drive forward on inner line
142                 if(cases(120) == 0) % if the next cell is free
143                     cars(thisCar,1) = 120;
144                     cars(thisCar,2) = 101;
145                     cases(120) = thisCar;
146                     cases(101) = 0;
147                     cars(thisCar,5)=0;
148                 else
149                     if cars(thisCar,2)==cars(thisCar,1)
150                         cars(thisCar,5)=1;
151                     end
152                     cars(thisCar, 1)=cars(thisCar,2);
153                 end
154
155
```

```
            otherwise
                if(cases(caseID - 1) == 0) % if next cell is free
                    cars(thisCar,1) = caseID-1; % move forward
                    cars(thisCar,2) = caseID; % save old position
                    cases(caseID-1) = thisCar; % lock new cell
                    cases(caseID) = 0; % free old cell
                    cars(thisCar,5)=0;
                else
                    if cars(thisCar,2)==cars(thisCar,1)
                        cars(thisCar,5)=1;
                    end
                    cars(thisCar, 1)=cars(thisCar,2);
                end
        end
    end
end

end
```

Listing 6: moveCarOuterRoundabout.m

```
function [cars,cases] = moveCarOuterRoundabout(cars, cases)
% MOVECAROUTERROUNDABOUT Move all cars in the outer line of the roundabout
% cars: n x 7 - Matrix
%     (x,1): actual position: cases.ID
%     (x,2): last position: cases.ID
%     (x,3): origin: [1:4]
%     (x,4): destination [1:4]
%
% cases: n x 1 - Matrix
%     (x,1): occupied: car.ID if yes, 0 if empty

% Algorithm:
% For each car, determine if it leaves at this exit. If yes,
% put it onto the road such that it can drive away from the roundabout.
% Else, push it forward in the roundabout.
%
%
% Note:
% This algorithm implies that if the car cannot exit, it stops.

for caseID = 81 : 100
    if(cases(caseID, 1) > 0) % if there is a car in this case number
        thisCar = cases(caseID, 1);
        switch caseID
            case 81
```

```
26              % if car wants to leave here (road 1) and there is free space
27              if(cars(thisCar,4) == 1)
28                  if(cases(20) == 0)
29                      cars(thisCar,1) = 20;           % new position
30                      cars(thisCar,2) = 81;           % old position
31                      cases(20) = cases(caseID, 1); % lock new cell
32                      cases(81) = 0;              % free old cell
33                      cars(thisCar,5)=0;
34                  else
35                      if cars(thisCar,2)==cars(thisCar,1)
36                          cars(thisCar,5)=1;
37                      end
38                      cars(thisCar, 2)=cars(thisCar,1);
39                  end
40              else % car doesn't indicate, hence drives forward inside the
                    roundabout.
41                  if(cases(100) == 0)
42                      cars(thisCar,1) = 100;           % new position
43                      cars(thisCar,2) = 81;           % old position
44                      cases(100) = cases(caseID, 1); % lock new cell
45                      cases(81) = 0;              % free old cell
46                      cars(thisCar,5)=0;            %no queuing
47                  else
48                      if cars(thisCar,2)==cars(thisCar,1)
49                          cars(thisCar,5)=1;        %queuing
50                      end
51                      cars(thisCar,2)=cars(thisCar,1);
52                  end
53              end
54
55          case 86
56              if(cars(thisCar,4) == 4)
57                  if(cases(80) == 0)
58                      cars(thisCar,1) = 80;
59                      cars(thisCar,2) = 86;
60                      cases(80) = cases(caseID, 1);
61                      cases(86) = 0;
62                      cars(thisCar,5)=0;            %no queuing
63                  else
64                      if cars(thisCar,2)==cars(thisCar,1)
65                          cars(thisCar,5)=1;        %queuing
66                      end
67                      cars(thisCar, 2)=cars(thisCar,1);
68                  end
69              else
70                  if(cases(85) == 0)
```

```
71                      cars(thisCar,1) = 85;
72                      cars(thisCar,2) = 86;
73                      cases(85) = cases(caseID, 1);
74                      cases(86) = 0;
75                      cars(thisCar,5)=0;          %no queuing
76                  else
77                      if cars(thisCar,2)==cars(thisCar,1)
78                          cars(thisCar,5)=1;      %queuing
79                      end
80                      cars(thisCar, 2)=cars(thisCar,1);
81                  end
82              end
83
84          case 91
85              if(cars(thisCar,4) == 3)
86                  if(cases(60) == 0)
87                      cars(thisCar,1) = 60;
88                      cars(thisCar,2) = 91;
89                      cases(60) = cases(caseID, 1);
90                      cases(91) = 0;
91                      cars(thisCar,5)=0;          %no queuing
92                  else
93                      if cars(thisCar,2)==cars(thisCar,1)
94                          cars(thisCar,5)=1;      %queuing
95                      end
96                      cars(thisCar, 2)=cars(thisCar,1);
97                  end
98              else
99                  if(cases(90) == 0)
100                     cars(thisCar,1) = 90;
101                     cars(thisCar,2) = 91;
102                     cases(90) = cases(caseID, 1);
103                     cases(91) = 0;
104                     cars(thisCar,5)=0;          %no queuing
105                 else
106                     if cars(thisCar,2)==cars(thisCar,1)
107                         cars(thisCar,5)=1;      %queuing
108                     end
109                     cars(thisCar, 2)=cars(thisCar,1);
110                 end
111             end
112
113         case 96
114             if(cars(thisCar,4) == 2)
115                 if(cases(40) == 0)
116                     cars(thisCar,1) = 40;
```

```
                    cars(thisCar,2) = 96;
                    cases(40) = thisCar;
                    cases(96) = 0;
                    cars(thisCar,5)=0;          %no queuing
                else
                    if cars(thisCar,2)==cars(thisCar,1)
                        cars(thisCar,5)=1;      %queuing
                    end
                    cars(thisCar, 2)=cars(thisCar,1);
                end
            else
                if(cases(95) == 0)
                    cars(thisCar,1) = 95;
                    cars(thisCar,2) = 96;
                    cases(95) = thisCar;
                    cases(96) = 0;
                    cars(thisCar,5)=0;          %no queuing
                else
                    if cars(thisCar,2)==cars(thisCar,1)
                        cars(thisCar,5)=1;      %queuing
                    end
                    cars(thisCar, 2)=cars(thisCar,1);
                end
            end

        otherwise % move forward inside the roundabout
            if(cases(caseID-1) == 0) % if next cell is free
                cars(thisCar,1) = cars(thisCar,1) - 1; % move forward
                cars(thisCar,2) = cars(thisCar,1); % save old position
                cases(caseID-1) = thisCar; % lock new cell
                cases(caseID) = 0; % free old cell
                cars(thisCar,5)=0;          %no queuing
            else
                if cars(thisCar,2)==cars(thisCar,1)
                    cars(thisCar,5)=1;      %queuing
                end
                cars(thisCar, 2)=cars(thisCar,1);
            end
    end
  end
end

end
```

Listing 7: moveCarToCross.m

```
function [cars, cases] = moveCarToCross(cars, cases, time)


for caseID = 121 : 280
    switch caseID
        %%%%%%%%%%%%%%%%%% CARS WANTING TO ENTER THE INNER LANE
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        case 121
            if cases(caseID,1)>0
                thisCar=cases(caseID,1);
                if (cases(100,1)==0 && cases(120,1)==0 && cases(81,1)==0 && cases
                    (101,1)==0);
                    cars(thisCar,2)=caseID; %last position
                    cars(thisCar,1)=120; % move into roundabout
                    cars(thisCar,5)=0; %not queuing
                    cases(120,1)=cases(caseID,1);
                    cases(caseID,1)=0;
                elseif cars(thisCar,2)==cars(thisCar,1)
                    cars(thisCar,5)=1; %queuing
                else
                    cars(thisCar,2)=caseID; %not yet queuing
                end
            end
        case 141
            if cases(caseID,1)>0
                thisCar=cases(caseID,1);
                if (cases(96,1)==0 && cases(115,1)==0 && cases(95,1)==0 && cases
                    (116,1)==0);
                    cars(thisCar,2)=caseID; %last position
                    cars(thisCar,1)=115; % move into roundabout
                    cars(thisCar,5)=0; %not queuing
                    cases(115,1)=cases(caseID,1);
                    cases(caseID,1)=0;
                elseif cars(thisCar,2)==cars(thisCar,1)
                    cars(thisCar,5)=1; %queuing
                else
                    cars(thisCar,2)=caseID; %not yet queuing
                end
            end
        case 161
            if cases(caseID,1)>0
                thisCar=cases(caseID,1);
                if (cases(91,1)==0 && cases(110,1)==0 && cases(90,1)==0 && cases
                    (111,1)==0);
                    cars(thisCar,2)=caseID; %last position
                    cars(thisCar,1)=110; % move into roundabout
```

```
43                  cars(thisCar,5)=0; %not queuing
44                  cases(110,1)=cases(caseID,1);
45                  cases(caseID,1)=0;
46              elseif cars(thisCar,2)==cars(thisCar,1)
47                  cars(thisCar,5)=1; %queuing
48              else
49                  cars(thisCar,2)=caseID; %not yet queuing
50              end
51          end
52      case 181
53          if cases(caseID,1)>0
54              thisCar=cases(caseID,1);
55              if (cases(86,1)==0 && cases(105,1)==0 && cases(85,1)==0 && cases
                    (106,1)==0);
56                  cars(thisCar,2)=caseID; %last position
57                  cars(thisCar,1)=105; % move into roundabout
58                  cars(thisCar,5)=0; %not queuing
59                  cases(105,1)=cases(caseID,1);
60                  cases(caseID,1)=0;
61              elseif cars(thisCar,2)==cars(thisCar,1)
62                  cars(thisCar,5)=1; %queuing
63              else
64                  cars(thisCar,2)=caseID; %not yet queuing
65              end
66          end
67      case 201
68          if cases(caseID,1)>0
69              thisCar=cases(caseID,1);
70              if (cases(99,1)==0 && cases(100,1)==0);
71                  cars(thisCar,2)=caseID; %last position
72                  cars(thisCar,1)=99; % move into roundabout
73                  cars(thisCar,5)=0; %not queuing
74                  cases(99,1)=cases(caseID,1);
75                  cases(caseID,1)=0;
76              elseif cars(thisCar,2)==cars(thisCar,1)
77                  cars(thisCar,5)=1; %queuing
78              else
79                  cars(thisCar,2)=caseID; %not yet queuing
80              end
81          end
82      case 221
83          if cases(caseID,1)>0
84              thisCar=cases(caseID,1);
85              if (cases(95,1)==0 && cases(94,1)==0);
86                  cars(thisCar,2)=caseID; %last position
87                  cars(thisCar,1)=94; % move into roundabout
```

```
88                  cars(thisCar,5)=0; %not queuing
89                  cases(94,1)=cases(caseID,1);
90                  cases(caseID,1)=0;
91              elseif cars(thisCar,2)==cars(thisCar,1)
92                  cars(thisCar,5)=1; %queuing
93              else
94                  cars(thisCar,2)=caseID; %not yet queuing
95              end
96          end
97      case 241
98          if cases(caseID,1)>0
99              thisCar=cases(caseID,1);
100             if (cases(90,1)==0 && cases(89,1)==0);
101                 cars(thisCar,2)=caseID; %last position
102                 cars(thisCar,1)=89; % move into roundabout
103                 cars(thisCar,5)=0; %not queuing
104                 cases(89,1)=cases(caseID,1);
105                 cases(caseID,1)=0;
106             elseif cars(thisCar,2)==cars(thisCar,1)
107                 cars(thisCar,5)=1; %queuing
108             else
109                 cars(thisCar,2)=caseID; %not yet queuing
110             end
111         end
112     case 261
113         if cases(caseID,1)>0
114             thisCar=cases(caseID,1);
115             if (cases(85,1)==0 && cases(84,1)==0);
116                 cars(thisCar,2)=caseID; %last position
117                 cars(thisCar,1)=84; % move into roundabout
118                 cars(thisCar,5)=0; %not queuing
119                 cases(84,1)=cases(caseID,1);
120                 cases(caseID,1)=0;
121             elseif cars(thisCar,2)==cars(thisCar,1)
122                 cars(thisCar,5)=1; %queuing
123             else
124                 cars(thisCar,2)=caseID; %not yet queuing
125             end
126         end
127     otherwise
128         if (cases(caseID,1) > 0) % if there is a car in this case number
129             thisCar=cases(caseID,1);
130             if(cases(caseID-1,1)==0) % if next case empty, drive forward
131                 cars(thisCar,2)=caseID; %last position
132                 cars(thisCar,1)=caseID-1; % actual position
133                 cars(thisCar,5)=0; %not queuing
```

```
                    cases(caseID-1,1)=cases(caseID,1);
                    cases(caseID,1)=0;
                elseif cars(thisCar,2)==cars(thisCar,1)
                    cars(thisCar,5)=1; %queuing
                else
                    cars(thisCar,2)=caseID; %not yet queuing
                end
            end
    end
end
%%%%%%%%%% Entering cars %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:size(cars)
    if (cars(i,6)>time)
        break;
    end
    if(cars(i,7)==1)
        switch(cars(i,8))
            case 140
                if(cases(140,1)==0)
                    cases(140,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=140;
                end
            case 160
                if(cases(160,1)==0)
                    cases(160,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=160;
                end
            case 180
                if(cases(180,1)==0)
                    cases(180,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=180;
                end
            case 200
                if(cases(200,1)==0)
                    cases(200,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=200;
                end
            case 220
                if(cases(220,1)==0)
                    cases(220,1)=i;
                    cars(i,7)=0; %state:driving
```

```
                    cars(i,1)=220;
                end
            case 240
                if(cases(240,1)==0)
                    cases(240,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=240;
                end
            case 260
                if(cases(260,1)==0)
                    cases(260,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=260;
                end
            case 280
                if(cases(280,1)==0)
                    cases(280,1)=i;
                    cars(i,7)=0; %state:driving
                    cars(i,1)=280;
                end
        end
    end
end
```

Listing 8: showstats.m

```
function showstats(finalt, queuestats, runtimes)
%SHOWSTATS Plots the final statistics of the simulation
%   Detailed explanation goes here
fig4=figure(4);
set(fig4, 'Position', [50 50 1200 300] );
hold on
subplot(1,2,1)
plot(1:length(queuestats(1,:)),queuestats(1,:),'-r',1:length(queuestats(1,:)),
    mean(queuestats(1,:)),'-b');

    title('queuing statistics');
    xlabel('time');
    ylabel('# cars');
    legend('queue length', 'average queue length');
    axis([0 finalt 0 max(queuestats(1,:))]);
    hold off

subplot(1,2,2)
hold on
    hist(runtimes,linspace(0,max(runtimes),20));
```

```
    %plot(1:max(runtimes),mean(runtimes));
    text(max(runtimes)*2/3,2000,['average runtime: ' int2str(mean(runtimes))])
    axis tight
    title('runtime histogram');
    xlabel('running duration');
    ylabel('# cars');

   hold off

end
```

Listing 9: statistics.m

```
function [ queue ] = statistics( cars, cases ,queue )
%STATISTICS Responsible for all kind of statistics
%   queue: number of cars actually in a queue

queue = [queue length(find(cars(:,5)))];

end
```

Listing 10: visualize.m

```
function [filmli]=visualize(cases, show, filmli, queue)
    screen_size = get(0, 'ScreenSize');
    figHandle=figure(1);
    set(figHandle, 'Position', [0 0 screen_size(3) screen_size(3)/2] );
    clf;
    figure(figHandle);
    subplot(2,2,[1 3])
    hold on;
    axis([-100 100 -100 100]);
    plot(cases(:,2),cases(:,3),'o','Color',[0.5 0.5 0.5]);

    occupied = find(cases(:,1) > 0);
    if( ~isempty(occupied) )
        plot(cases(occupied,2), cases(occupied,3), 'hr');

    end
    subplot(2,2,2)
    plot(1:length(queue),queue,'-r');
    title('queuing statistics');
    xlabel('time');
    ylabel('cars queuing');
    %axis([0 2*tmax 0 25]);

```

```
24      F = getframe(figHandle);
25      filmli = addframe(filmli,F);
```

References

[1] R. Wang and HJ Ruskin. Modelling Traffic Flow at Multi-Lane Urban Roundabouts. *International Journal of Modern Physics C*, 17(5):693–710, 2006.

[2] Wikipedia. Roundabout — wikipedia, the free encyclopedia, 2010. [Online; accessed 11-December-2010].

[3] Tony Wood and Bastian Bücheler. Traffic Dynamics — Traffic Flow Comparison of Roundabouts and Crossroads. *Lecture with Computer Exercises: Modelling and Simulating Social Systems with MATLAB — Project Report*, 2010.