Team Control Number

57566

Problem Chosen

B

2017
MCM/ICM
Summary Sheet

An optimized toll plaza design scheme

Summary

This paper proposes a design scheme for a better shaped and sized toll plaza based on the analysis of vehicle trajectories in the worst traffic condition: a large flow. The individual vehicles are navigated by a centralized algorithm control. The algorithm is inspired by partial analysis of the merging process. When the design is generated, a traffic model coded in $MATLAB$ is to evaluate and to ameliorate the performance of the toll station by means of simulation.

It is perceived that the boundary of the area where the vehicles actively move is the minimum area required by the normal functionality of toll station. A detailed navigated route is proposed for each vehicle in the area to establish an order among a heavy traffic, which is to successively and non-linearly queue the vehicles into lanes, enabling all those from the tollbooths to successfully merge without causing an accident. The trajectories vary accordingly with different time requirements. Their borders are used as a reference to construct the shape of the merging area. The shape is then input into the simulation to test for random traffic flows.

The barrier is simulated by a number of tollbooths of different types, each configurable for vehicle passage type, charging mode and lane direction. The vehicles are simulated by a set of rectangles in a two-dimensional plane, moving out of default acceleration and collision avoidance. The vehicles are generated by the tollbooth based on a probabilist model, given a fixed traffic flow and a predefined proportion of vehicle size. The vehicles submit to possible collisions with each other and the road boundary. Both autonomous cars and human-driven ones are simulated, distinguished by adapted moving strategies.

The size of toll station, run way length and lane numbers are configured directly by parameters in the simulation, the shape of the station is reflected by the set initial speed and direction of the vehicles passing through the specific tollbooth.

The performance of the design is evaluated by a weighted average of throughput, accident rate statistics and cost. The throughput is obtained by calculating the average merging completion numbers in reaction to multiple simulated traffic flows. The accident rate is calculated by the collision incidents divided by vehicle quantities. The cost is calculated in function of the construction area and tollbooth number. The experiments with different parameters of tollbooth and shape give an analysis of several design schemes, detailed in the report.

Keywords: ;

Contents

A letter to the New Jersey Turnpike Authority

Dear New Jersey Turnpike Authority,

It is a pleasure for us to learn that you are currently considering the employment new toll plaza designs and are actively seeking for inspiring ideas. We are a group of engineering students, proposing to you a new design idea, with a simulation program to evaluate the plaza design in random traffic conditions.

It is in our mind that the throughput of the toll is firstly limited by the toll barrier. Thus under certain conditions, the average time for a specific vehicle to pass through a tollbooth is determined by the speed limit, the charging method and the length of the vehicle. Thus, the number of tollbooth can, fix the maximum of throughput

Once you has determined the target traffic through put, the shape of the merging area could be drawn based on the vehicle trajectories in an extreme traffic flow: the case where each booth releases a vehicle at the same time and the vehicles are competing for the entrance to a limited number of lanes. A guideline for each vehicle's navigation in such circumstance are proposed in our report. The minimal area is determined by the active area of the vehicles, and out of security purposes, the area can be expanded to a certain degree.

The simulation model attached examines the performance of the minimal shape for more temperate flows. The simulated vehicles are equipped with movement decision strategies based on Car Following models, and are adjusted to avoid collisions with other vehicles on the road surface. Although we encourage an explicit guidance to be implemented for extreme circumstances mentioned as above, the shape is tested to be capable to accommodate moderate traffic flows with almost no accidents occurring in the simulation.

More specifically, the performance model is developed after taking all important elements related to this problem into consideration, like number of lanes and tollbooths, proportions of tollbooths, varieties of vehicles, change of flows, every decision made on directions and accelerations by drivers.

Sincerely yours,

Team # 57566

1 Introduction

1.1 Statement of the problem

The design of toll plaza is undoubtedly a state of art as it is hard to find a balance among safety, capacity and cost, facing different situations. It also acts as an essential part in the high-way traffic system. Considering a better toll station design is in demand, mathematical methods and simulation models are implemented to optimize the design schemes, striving to increase the throughput, decrease the cost and accident rate.

A shape design is therefore required to accommodate the traffic and to allow the vehicles go through the plaza smoothly without too much interference of the vehicles around.

1.2 Assumptions

- Assumptions for the toll station:

    - Toll station has a fixed configuration for each simulation: vehicle type and charging method.
    - Tollbooths allowing large vehicles are also available for smaller ones to pass.
    - It takes a certain time for every type of vehicle to leave the tollbooth.
    - An extra delay is caused by the charging process, the most fast for electronic payments, then for exact-changes, the slowest for conventional tollbooths.

- Assumptions for the vehicles.

    - The proportion of different types of vehicles passing through the barrier is fixed along the time.
    - All vehicles are enabled for all 3 types of charging.
    - All vehicles leave the tollbooths with a fixed speed.
    - All vehicles must move in respect of the direction of tollbooth lane.
    - Each vehicle is considered as a point located in the center of gravity, but the vehicles still have a volume.
    - The accelerations of all the vehicles is stimulated by the position and speed of the vehicles around.
    - The vehicles moves in a discrete fashion: the position and acceleration is updated every second.
    - The vehicles have a maximum acceleration for $a_{y,max}$, $a_{x,max}$, deceleration $a_{y,min}$, $a_{x,min}$ and speed $v_{y,max}$ and $v_{x,max}$.
    - The biggest wheel steering angle is $45°$, and the turning radius is neglected
    - When the vehicles moves in a greater speed than a critical speed, the vehicle is considered to head forward no matter it is moving towards left or right.

- Assumptions for the flow generating:

    - When a flow is fixed for a time duration, the total flow F_t is uniformly distributed into each second.

- Vehicles in the simulation must first pass the tollbooths, then join the merging flow.
- In circumstances where traffic is light, the flow passing through tollbooths respect the distribution mentioned above.
- In circumstances where traffic is busy, the vehicles begin to queue up reasonably before the tollbooths, therefore every tollbooth is allowing a maximum flow to pass following the proportion and time gap.
- Light and heavy traffic is distinguished by a critical flow F_c, which is determined by the tollbooth capacity in simulation.

Following is a list of notions used in the article:

Notations	Meanings
L	Number of lanes
B	Number of tollbooths
F_t	Total flow
F_c	Critical flow (the maximum of the total flow)
P_l, P_m, P_s	Probability of large-scale automobiles, medium-sized vehicles and compact cars
D_l, D_m, D_s	The lengths of large-scale automobiles, medium-sized vehicles and compact cars
$D_{general}$	The length of vehicles regardless of the type
v_{this}, a_{this}	Speed and acceleration of the vehicle we study
v_n, a_n	Speed and acceleration of the other vehicles which surround the vehicle we study
pos_x, pos_y	We build a rectangular coordinate system with the leftmost tollbooth constructed as the origin.
L_{length}	Merging area length
W_{lane}	lane width

2 Analysis of the Problem

To respect the notion of "barrier", and to avoid the complexity of gradient, this article discusses only the toll stations of one storey. It is considered that the approach ramps would complicate the traffic flow in both edge of the station, and the cost would be enormous. A separation of lanes implemented by the road rail would decrease the liberty of the traffic flow.
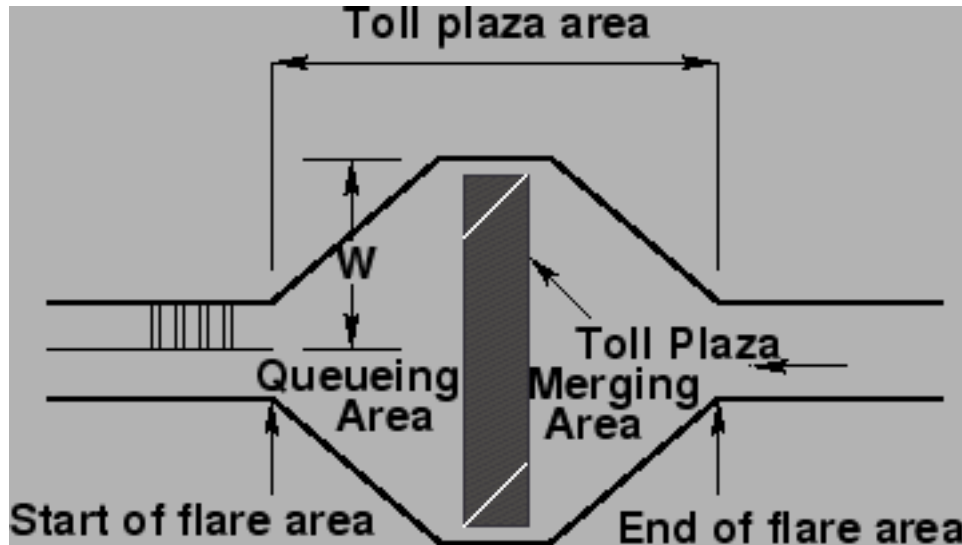
Figure 1: Toll station [Mathew, 2014]

In the merging area, all the vehicles merging from B tollbooths into L lanes in a short distance of L_{length} meters. With lane width predefined as 4 meters, the merging area is defined as the area surrounded by two curves on a finite two-dimension plane. The curves are represented by two functions, which would be a parameter to be optimized in the problem.

2.1   Obtaining an adopted shape design

A primitive idea of establishing an orderly traffic flow is to set up constraints and guide-lines for the individualist vehicles: if all participants follow a predefined, well-designed route, there wouldn't be unpredicted crosses at all. It is in addition more practical, to set up such rules in a limited area, where the traffic is limited by the entrance ports of the turnpike.

Thus a design scheme that is capable of coping with the worst traffic conditions, can be inspired by the dynamics of the worst traffic conditions: concurrent vehicles issued from every booth and a high frequency of passages. On finding routing solutions for those vehicle obeying to a centralized control, the convex area covered by the trajectories is the minimum area required by such a traffic in order to merge.

The possible solutions therefore, should guide the vehicles into a proper slot in a queue for each lane, and each vehicle ahead should not be chased by each other, nor the vehicles newly passed through the tollbooth.

Symmetry analyses give that the optimal shape is symmetrical about the central axis of the turnpike. The article concerns therefore only symmetrical designs.
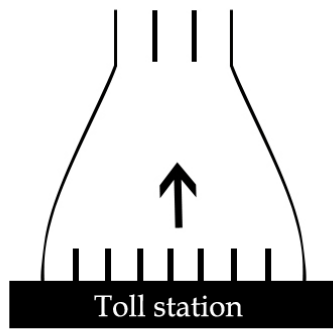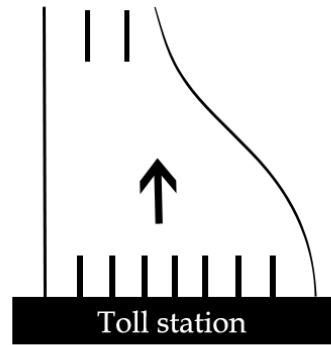
Figure 2: symmetrical   Figure 3: asymmetrical

Theorem 2.1. *The optimal shape is symmetrical about the central axis.*

Proof. Reductio ad absurdum: if the shape is not symmetrical, hence the traffic flow is not either, so there exist multiple vehicles that have no corresponding vehicles about the axis of symmetry of L lanes. If these vehicles are given slots of certain priority in the queue, and all vehicles successfully merged into L lanes, then the slots of the same priority can also be allocated to the non-existing vehicles in the symmetry. Thus this flow is capable of hold more vehicles, this is not the optimal shape. □

We can further show that the comportment of vehicles is also symmetrical about the axis. So only one side of the traffic is to be analyzed in the following section.

Theorem 2.2. *The quickest way for a vehicle originally side by another to outrun it is to run the most quickly the possible.*
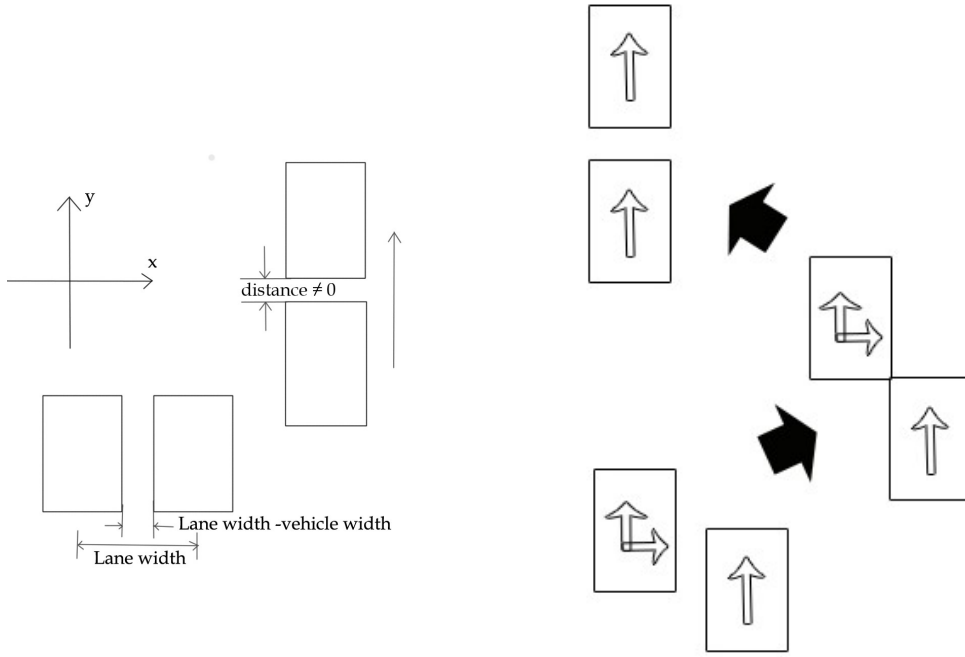
Figure 4: Two phases

Proof. Denote the direction where the road leads the vertical direction. Initially, the two vehicles are running side-by-side, and share the same speed v_0 and direction. The vertical distance between the two vehicles (calculated by the position of the center of the vehicle) is 0, the horizontal distance is:

$$x_2(t_0) - x_1(t_0) = lane - width \qquad [Equation \ A]$$

In the end, vehicle 2 is positioned right after vehicle 1.

$$y_1(t_e) - y_2(t_e) = L_d + vehicle_length \qquad [Equation \ B]$$

The integration of accelerations give the speed at a specific instant:

$$v_y(t) = v_y(t_0) + \int_{t_0}^{t} a_y(u)du \qquad [Equation \ C]$$

And the integration of speed give the displacement of a vehicle:

$$y(t) = y(t_0) + \int_{t_0}^{t} v_y(u)du = y(t_0) + v_y(t_0) \times (t - t_0) + \int_{t_0}^{t}\int_{t_0}^{w} a_y(u)dudw \qquad [Equatio \ D]$$

Equation B can therefore be transcribed as :

$$y_1(t_e) - y_2(t_e) = y_1(t_0) + v_{y,1}(t_0)(t_e - t_0) + \int_{t_0}^{t_e}\int_{t_0}^{w} a_{y,1}(u)dudw$$

$$-y_2(t_0) - v_{y,2}(t_e - t_0) + \int_{t_0}^{t_e}\int_{t_0}^{w} a_{y,2}(u)dudw$$

$$= \int_{t_0}^{t_e} \int_{t_0}^{w} (a_{y,1}(u) - a_{y,2}(u)) du dw = D_{general} \qquad [Equation\ E]$$

As the integration reaches for a constant, t_e can be minimized if the difference of the two accelerations can be maximized, from which conclude $a_{y,1}$ to be the maximum value permitted by the situation.

$$\int_{t_0}^{t} a_{y,i}(u) du \in [0, v_{y,max}]$$

$$\int_{t_0}^{t} a_{y,i}(u) du \in [0, v_{x,max}]$$

□

Theorem 2.3. *The quickest way for a moving vehicle to change a lane, is to accelerate to the side with maximum power (until a maximum sideway speed is reached) then decelerate, for all initial sideway speed.*

The 2 to 1 merging process can be split into two phases(Figure 4). In phase one, one vehicle moving faster in order to outrun the other one, the other is approaching its side. In phase two, the merging vehicle moves behind the other. The minimum time required by phase 1 meets the condition of the previous theorem, meanwhile the minimum time required by phase two is achieved by accelerating to the side then decelerate so that the sideway speed is reduced to 0 on arriving in place.

As a matter of fact, for all merging processes, it can be considered as a composition of the two phases presented above. Note that shown from the demonstration, that the duration of each phase is dependent upon the initial speed of the vehicles, because the acceleration and deceleration would reduce to zero if the maximum or minimum speed is reached. Thus the whole merging duration can be minimized if an optimized order of the phase repetitions can be found.

2.2 Simulation

In order to evaluate the performance such as throughput, accident rate, this group implements simulation algorithms, mocking the process of vehicles passing through the tollbooth and try to merge without a centralized control: as heavy traffic of everyday definition is not about vehicles of a large size passing the tollbooth continuously! The throughput is evaluated by the number of vehicles that successfully enter into the L-lane area, and accident rate is calculated by the simulated accident numbers divided by total vehicle numbers.

To begin with, the traffic flow scale is fixed for each simulation experience. F_t stands for the quantity of cars, of various size, passing through the tollbooths every fifteen minutes. The flows of large-scale automobiles, medium-sized vehicles and compact cars are $F_t P_l$, $F_t P_m$, $F_t P_s$.

Since only the traffic after the toll station is to be considered, details of incoming vehicles such as the queuing and proportion distribution are simplified in the model. As the assumptions stated, the proportion of vehicle size and the capability of autonomous driving are implemented by a probabilist model, of which the proportions are fixed by us.

The vehicles are instancialized with concrete parameters after the tollbooth. That is to say, the traffic to be simulated in the experiences has only one parameter to be adjusted in the performance evaluation: the flow. Corresponding to each flow level, the simulation algorithm adjusts its strategy to generate the traffic. It is remarked that different vehicles takes different times to leave the booth, and the time difference arisen by different charging method is also taken into consideration by adding a minimum time gap for each booth to allow the next passage.

The strategy is detailed as follows: for a fixed number of tollbooths in one performance evaluation experience, the maximum traffic is generated by allowing the passage of a maximum of vehicles through each tollbooth. Thus a critical flow F_c denotes the expectation of maximum traffic.

On cases where traffic flow configured for the simulation is greater than the critical flow, as assumed the traffic flow successfully finds an optimal queuing strategy, allowing each tollbooth to let pass vehicles continuously meanwhile respecting the time gap for each vehicle.

For other cases, the traffic is allocated into each second of the 15 minutes following a uniform fashion. With the assumption that all the vehicles arriving at the booth finds an appropriate booth to pass through, and if the vehicle number in this second surpasses the number of the tollbooth available, the extra vehicles are queued into the next second.

The vehicles are tagged with its size and nature whether it is human-driven on leaving the tollbooth. The two kinds of tags are independently proportionally distributed.

The merging flow are constituted by vehicles leaving the tollbooth. Since in the problem we are focusing on individual vehicle comportments in an area of $50m \times 200m$, the vehicles are not to be represented by moving points: the direction of the front of the vehicle, the direction of the speed and interactions from all orientations are to be taken into account.

It is suggested that at the moment where the vehicle leaves the booth, its speed and direction are restricted by the booth, since a speed limit is implemented and the car can only follow the shape of the lane passing through the booth. Thus, the shape of the booth are represented by the initial speed and direction of the vehicles. With each design of the toll station, an interpretation of initial speed and direction would be given.

One crucial difference of the model presented compared with the classic ones is that this model preserves the physical dimensions of the vehicle, and enables a relative free movement in a two-dimension plane, instead of a strict restraint of one lane. Another one is that the decisions of all vehicles active on the road are calculated for the same instant in the simulation, instead of updating firstly the followed vehicle then the following vehicle.

The human-driven vehicles are represented in the model by an approximation of their vertical projection, rectangles of different sizes. To implement the dynamics, Nagel-Schreckenberg (NS) model [K.Nagel and M.Schreckenberg, 1992], GM model and CF (car following) model are taken reference to simulate the driving strategies. The model presented by this paper considers the acceleration of each vehicle a compound decision of three factors: avoidance for collision with other vehicles.

In this article, an extended version of Car-following model is proposed, preserving the idea that driving consists of the process of perception, decision making and control.

Based on the active vehicles on the road and the road shape, the driver's decision includes acceleration or deceleration of the direction ahead and the one sideways.

$$[Response]_n = \alpha[Simulus]_n$$

interpreted as interactions between the vehicles, avoidance for collision with the road boundary, and its intrinsic willingness to achieve a maximum speed.

The acceleration takes three forms for different cases.

It is considered that on leaving a speed limited zone, the vehicles are supposed to reattain a speed of normal level. This process is simulated by a positive factor of the gap of current speed and the ideal speed. However, the acceleration ahead may cause collision with vehicles in front. For the avoidance, the acceleration decision is made based on avoidance of possible collisions: should the driver preserve its current speed, or do an acceleration, the position of the vehicle in the next moment of decision is to be calculated, and the vehicle would not be in collision with the vehicles in its general front. It is worth noting that if such a vehicle in front doesn't exist, the vehicle would accelerate to a maximum speed.

Thus If $\frac{v_y^2 - v_{y,forward}^2}{2a_{max}} < distance_{forward}$,

$$v_{y,forward} = \begin{cases} v_{y,forward} & (car) \\ 0 & (boundary) \end{cases}$$

Then for a_y:

$$\frac{(v_y + a_y t)^2 - v_{y,forward}^2}{2a_{max}} = distance_{forward}$$

if $a_y > a_{max}$, then $a_y = amax$, $a_x = 0$.

If a collision is predicted for current speed, the driver would take necessary measures to avoid the collision: firstly try to take another lane, or secondly if there's no other choice, slow down. In our model, taking another lane is generalized into taking a left or right angle.

To make the choice of taking another line possible, there should in the first place exist a lane to be taken. In order to keep the vehicles in the simulation from colliding with the edge of the road, a safe-distance psychological model, that the drivers drive away from the edge of the road, if the distance between the vehicle and the boundary is too small, is implemented.

Secondly, directing the vehicle to the left-front or right-front causes possible collision with other vehicles. The simulation implements a searching algorithm that the vehicle is to search the possibility to turn left or turn right.Firstly We assume that all the drivers know the speed of other vehicles around, which is correspond to the real case. Secondly, we assume that when turn left or turn right, all the vehicles are to preserve its current speed ahead. Thirdly, we neglect the time of the turning action, the Pos_x can change 4m in the next iteration (1 Second).

thus, $\forall j \ (j \neq this)$ if

- $Pos_{future\_of\_j}$ and $Pos_{future\_of\_i}$ are not collided, judging with a extra margin Δl to eliminating the deviation of speeds.

- $v_{y,this} < v_{y,j}$

- $Pos_{future\_of\_j} + \frac{v_y^2}{2a_{brake\_maximum}^2} \times \overrightarrow{e_y}$ is not out of boundary.

The third condition ensures that after the vehicle changing a lane, it can stop with the maximum braking acceleration before colliding with the boundary.

If the left/right lane satisfies these conditions, then the vehicle will move left/right 4 meters (the width of a lane).

In addition, we consider that $v_{y,this}$ will not change ($a_{y,this} = 0$), except for the case where $v_{y,this} = 0$(when it is waiting for the opportunity to turn). In this special case, we set $a_{y,this} = 1/2 \times a_{max\_acceleration}$.

Thus,

$$Pos_{future\_of\_this\_x} = Pos_{future\_of\_this\_x} \pm 4$$

$$\begin{cases} a_{y,this} = 0, & (v_{y,this} > 0) \\ a_{y,this} = \dfrac{1}{2} \times a_{max\_acceleration} & (v_{y,this} = 0) \end{cases}$$

$\delta y_{max} = v_y t$, as $a_y = 0$ when changing direction.

The autonomous vehicles are modeled in a similar way, whereas the extra margin Δl are smaller.

The simulation model is based on a combination of Nagel-Schreckenberg (NS) model [K.Nagel and M.Schreckenberg, 1992], GM model, CF (car following) model, and a strategy called Obstacle Avoidance of a Micro-bus[Fernandez et al.,2013], thus takes many some parameters that is merely experimental.

Thirdly, if the first two strategies can not be applied, the vehicle will brake by the acceleration determined by $v_{y-foward}$ and $distance_{foward}$, waiting for the opportunity to turn left or right.

$$a_y = \frac{v_{y-foward}^2 - v_y^2}{2 \times (distance_{foward} - distance_{limit})}$$

Where $distance_{limite}$ presents a safe distance. The determined a_y needs to be checked if $|a_y| > a_{brake_{maximum}}$
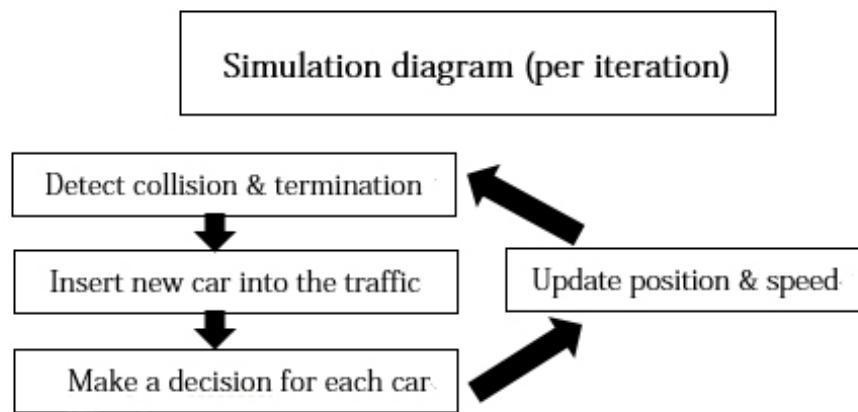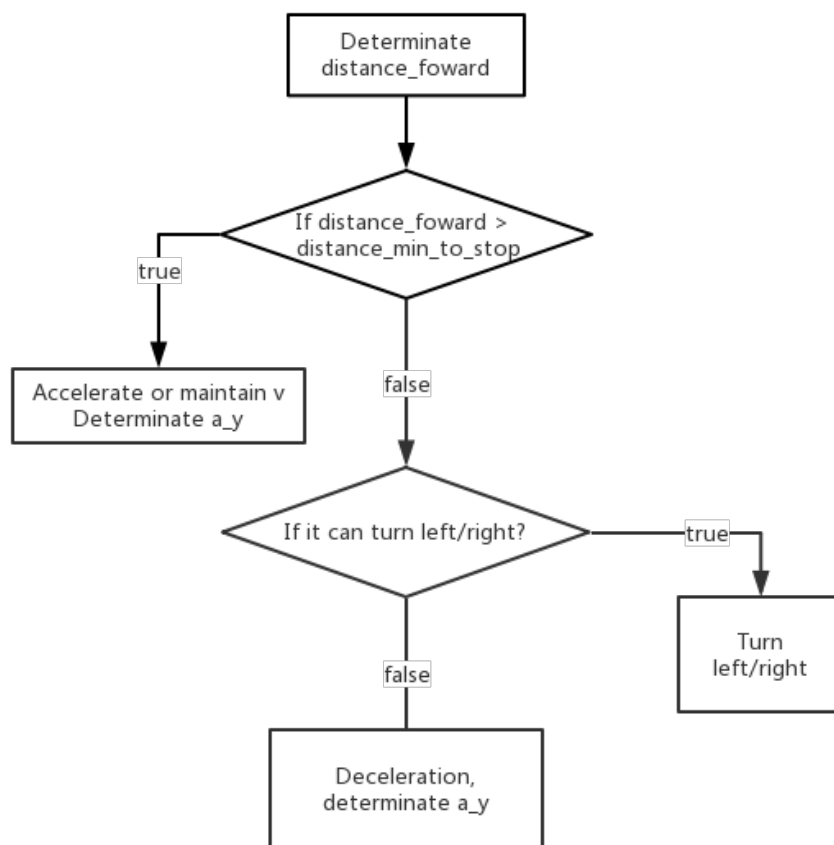
Figure 5: Our simulation

Figure 6: Make the decision for each car: decide acceleration

2.3   Evaluating the performance

In order to propose a better design, the performance of the design can be addressed as a weighted average index of three factors: throughput, accident rate and cost.

$$Thoughput(flow_{total,design}) = \mathbb{E}\left\{card(vehicle\ number\ who\ completes\ the\ merge)\right\}$$

$Accident\ rate(design) = \frac{\sum accident\ numbers}{\sum vehiclenumbers}$, sum for all simulation cases.

$$Cost(design) = \sum cost\_for\_booth(passage\ type, charge\ mode, lane\ length)$$
$$+cost(surface_{whole\ area} - L_{length} \times W_{lane} \times L)$$

3   Validating the Simulation Model

In order to adjust the parameters and evaluate the performance of the two-dimensional car-following model, motion data of real-life driving must be abundantly supplied to test the model. However, a degradation of dimension can be mocked on setting the tollbooth number B to 1 and the lane number L to 1: the parameters are to be adjusted that the simulated vehicles run similarly with those simulated by a classic car-following model.
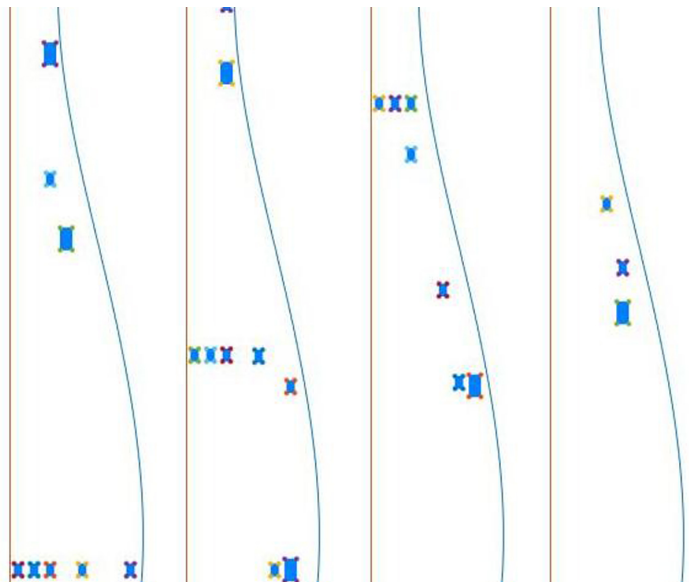
After running our program with the numeral coefficient adjusted, the vehicles pulling out of 1 tollbooth to 1 lane will not collide, which indicates our simulation model is correct to some extent.

4   The Model Results

After using the $MATLAB$ to run our program, the process of merging is simulated as follows:
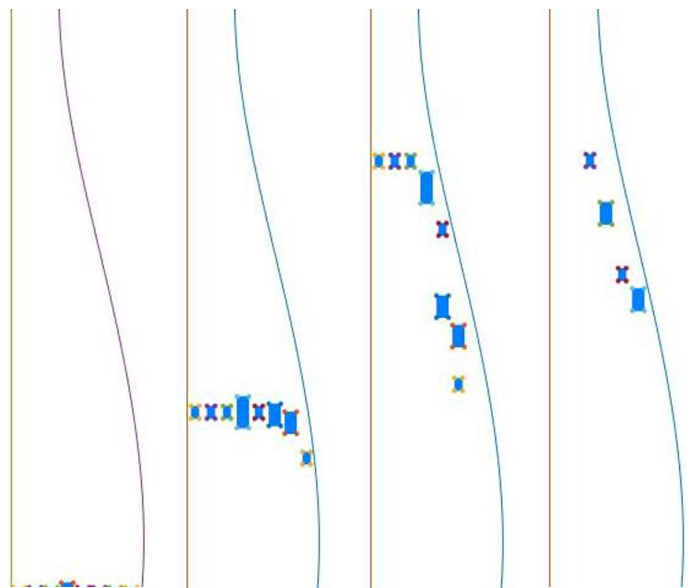
(a) process1

(b) process2
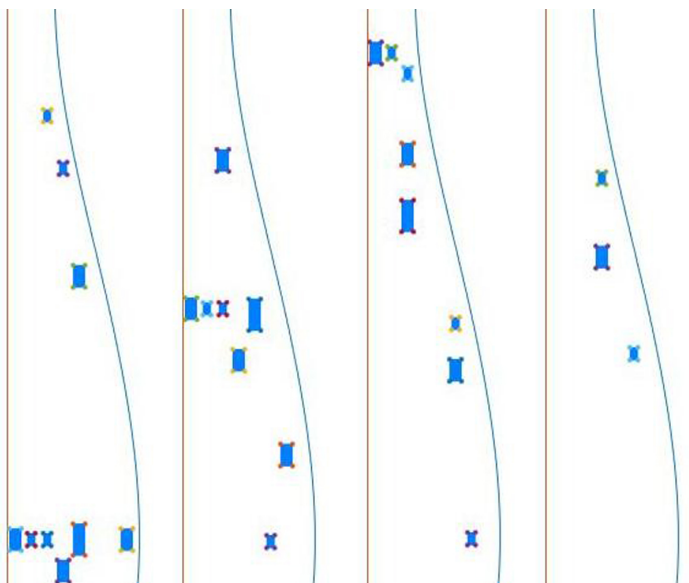
(c) process3

Figure 7: Process

5 Conclusions

6 Strengths and weaknesses

6.1 Strengths

- **Accommodation of extreme traffic flow**
  This model is designed apart from the consideration of extreme traffic condition, so the designed area is capable of handling all traffic.

- **Models proposed from two orientations**
  Two models are proposed in the article, one determining the shape, another for the simulation of traffic flow to evaluate the performance of such design.

6.2 Weaknesses

- The model doesn't include the limit of terrain, as the topographic condition is not always suitable for building a symmetrical area, in which case the shape and cost will change, and then our model is not optimal.

- Due to the

References

[Fernandez et al., 2013] Fernandez, C., Dominguez, R., Fernandez-Llorca, D., Alonso, J., and A.Sotelo, M. (2013). Autonomous navigation and obstacle avoidance of a micro-bus. *INTECH*.

[Gazis et al., 1961] Gazis, Herman, and Rothery (1961). Nonlinear follow the leader models of traffic flows[j]. *Operations Research*, (9(4)):545–567.

[K.Nagel and M.Schreckenberg, 1992] K.Nagel and M.Schreckenberg (1992). A cellular automaton model for freeway traffic. *J.PHYS.I(France)2,2221 2229*.

[Mathew, 2014] Mathew, T. V. (2014). Toll operation–lecture notes in traffic engineering and management.

[V.Mathew and Rao, 2007] V.Mathew, T. and Rao, K. V. K. (2007). Microscopic traffic flow modeling. *Introduction to transportation engeneering*.

Appendices

Here are simulation programmes we used in our model as follow.

Appendix A   Simulation

Input matlab source:

```matlab
filename = 'traffic_600.mat';

load(filename);
time_now = yyyymmdd(datetime('now'));
[hh,mm,~] = hms(datetime('now'));
filename = ['../../figure/',filename,num2str(time_now),'_',num2str(hh),'_',num2str(mm)];
figure(1);
filename = [filename, '.gif'];
% this code contains one simulation of the traffic
B = 8; % Tollbooth number
L = 3; % Regular lane number

global toll_barrier_config;
toll_barrier_config = [3,3,3,3,3,3,3,3; 10,10,10,10,10,10,10,10; 0 0 0 0 0 0 0 0];

%cell_size = 0.5; % cutting the road into small cells of 0.25 m^2

global merge_length; % Whole merge part length
merge_length = 200;
global width_veh % vehicle size config
global length_veh
width_veh = [2 3 3];
length_veh = [4 7 10];

global boundaryPoints
global vehicle_array
global vehicle_number
shapePoints = [32 0; 32 merge_length/4;25 merge_length/2;16 ...
    merge_length/4*3; 12 merge_length];
% (unit: m)the distance from the boundary of roads to
%the cell limit at y=50, 100, 150
boundaryPoints = zeros(merge_length,2);
% the second row presents the left boundary.
boundaryPoints(:,1) = interp1(shapePoints(:,2), shapePoints(:,1),...
    -0.5+(1:1:merge_length),'spline');

global has_collision
has_collision = 0;
global small_delay
global medium_delay
global large_delay
global initial_speed

global v_max;
v_max = 15;
global dt;
dt = 1;
```

```matlab
small_delay = 10; % delay caused by a small vehicle to pass a booth
medium_delay = 15;
large_delay = 30;
initial_speed = 5;
% line 1 for vehicle types: 1, small, 2, medium, 3, large
% line 2 for delay caused by charge mechanisms:
%10, conventional, 5, exact exchange, 2,
% electronic
flow_total = 600; % total flow

vehicle_array = zeros(flow_total,7);
% colomns 1, posx, 2, posy, 3, speed, 4, rad, 5 type, 6 collision, 7 is_AI
vehicle_number = 0; % the total vehicle number after the simulation start.

completion_count = 0;
global all_info_matrice
all_info_matrice = zeros(70, flow_total, 12);
global test_acc
test_acc = zeros(1,6);


for i=1:70 % one simulation per second;


    % detect position for collision and merge completion
    for j = 1:vehicle_number
        if vehicle_array(j,5) > 0 && vehicle_array(j,6) ~= 1
            % check if the merge is completed
            if vehicle_array(j,2) >= merge_length -1
                vehicle_array(j,5) = -1;
                completion_count = completion_count + 1;
            else
                % check if out of boundary
                isOut = isOutBoundary([vehicle_array(j,1),...
                    vehicle_array(j,2)],vehicle_array(j,5));
                if isOut == 1 % collision with road
                    has_collision = has_collision||isOut;
                    vehicle_array(j,6) = 1;
                    vehicle_array(j,3) = 0; % set speed to 0
                end
                % check if collision with other cars
                for a = 1:vehicle_number
                    if a ~= j && vehicle_array(a,5) ~= -1 &&...
                            vehicle_array(a,5) ~= 0

                        % showVehicule()

                        is_collide = isCollideSimple([vehicle_array(j,1:2) ...
                            vehicle_array(j,4)],[vehicle_array(a,1:2) ...
                            vehicle_array(a,4)],vehicle_array(j,5),...
                            vehicle_array(a,5),0);
                        if is_collide == 1

                            %showVehicule()
                            % collision with car
                            has_collision = has_collision||is_collide;
                            vehicle_array(j,3) = 0;
                            vehicle_array(a,3) = 0;
                            vehicle_array(j,6) = 1;
                            vehicle_array(a,6) = 1;
                        end
```

```matlab
                end
            end
        end
    end
end
% insert new cars into the traffic
if fix(i*dt) == i*dt
addNewVehicle(toll_barrier_state(71-floor(i*dt),:));
end

    % make and store decision for each driver
decision_array = zeros(vehicle_number,2); % colomn 1: acc_x, 2: acc_y
for j = 1:vehicle_number
    if vehicle_array(j,5) > 0 && vehicle_array(j,6) ~= 1
        decision_array(j,:) = decideAcc4(j);
        all_info_matrice(i, j, 7:12) = test_acc(1, 1:6);
    end
end
 for j = 1:vehicle_number
    if vehicle_array(j,5) > 0 && vehicle_array(j,6) ~= 1
        acc = decision_array(j,:)';
        speed_old = vehicle_array(j,3);
        speed = vehicle_array(j,3) + acc(2);

        if speed <= 0
            vehicle_array(j,3) = 0;
            continue
        end
        if speed > v_max
            speed = v_max;
        end
        vehicle_array(j,1) = acc(1) + vehicle_array(j,1);
        vehicle_array(j,2) = 1/2 * acc(2)*dt^2+ ...
            speed_old*dt + vehicle_array(j,2);
        vehicle_array(j,3) = speed;

    end
 end
    all_info_matrice(i, :, 1:6) = vehicle_array(:, 1:6);

 %    test part
plot(boundaryPoints(:,1),1:200,boundaryPoints(:,2),1:200);
axis([-100 100 0 200])
pic = imread('./blue.png');

points = zeros(2,4);
for t = 1:vehicle_number
    if vehicle_array(t,5) > 0
        hold on
        %pic1 = imrotate(pic, vehicle_array(t,4));

        imagesc([vehicle_array(t,1)-width_veh(vehicle_array(t,5))/2,...
            vehicle_array(t,1)+width_veh(vehicle_array(t,5))/2],...
            [vehicle_array(t,2)-length_veh(vehicle_array(t,5))/2 ,...
            vehicle_array(t,2)+length_veh(vehicle_array(t,5))/2],pic);

        points(:,1) = [ + width_veh(vehicle_array(t,5))/2 ; ...
            + length_veh(vehicle_array(t,5))/2]+vehicle_array(t,1:2)';
        points(:,2) = [ - width_veh(vehicle_array(t,5))/2 ; ...
            + length_veh(vehicle_array(t,5))/2]+vehicle_array(t,1:2)';
        points(:,3) = [ - width_veh(vehicle_array(t,5))/2 ; ...
```

```matlab
                    - length_veh(vehicle_array(t,5))/2]+vehicle_array(t,1:2)';
            points(:,4) = [ + width_veh(vehicle_array(t,5))/2 ; ...
                    - length_veh(vehicle_array(t,5))/2]+vehicle_array(t,1:2)';
            plot(points(1,:),points(2,:),'.');

            hold on
        end
    end

    drawnow
    frame = getframe(1);
    im = frame2im(frame);
    [imind,cm] = rgb2ind(im,256);
    if i == 1
        imwrite(imind,cm,filename,'gif', 'Loopcount',inf);
    else
        imwrite(imind,cm,filename,'gif','WriteMode','append');
    end
    clf('figure 1','reset');

end

visualisation
close all;
```

Appendix B   Decide the acceleration

Input matlab source:

```matlab
function[acc_v] = decideAcc4(i)
global vehicle_array % colomns 1, posx, 2, posy, 3, speed, 4, rad, 5 type
global vehicle_number
global boundaryPoints
global test_acc
%global v_max
global width_veh
global length_veh
global dt

DeBug_Pos_i = vehicle_array(i,2);
acc_max = 2;
acc_brake = 8;
Speed_v = [0;vehicle_array(i,3)];
distance_foward_max = 200;Speed_foward=0;
vehicle_other_v = zeros(40,4);
criticle_limite_intervehicle = 3;

acc_v = zeros(2,1);
foward_flag = 0; %can we go foward?

% vehicule restriction
for j = 1: vehicle_number
    if i ~= j && vehicle_array(j,5) > 0
        Speed_j = [0;vehicle_array(j,3)];
        Speed_j_v = (Speed_j - Speed_v);
        vehicle_other_v(j,3:4) = Speed_j_v;
        Pos_j_v = (vehicle_array(j,1:2)- vehicle_array(i,1:2))';
        Pos_j_future_v = Pos_j_v + Speed_j_v * dt;
```

```matlab
            vehicle_other_v(j,1:2) = Pos_j_future_v';


        if Pos_j_v(2) > 0
            if abs(Pos_j_future_v(1)) < 1 +...
                    1/2 *( width_veh(vehicle_array(i,5))+...
                    width_veh(vehicle_array(j,5)))
                if Pos_j_future_v(2) - 1/2 *( ...
                        length_veh(vehicle_array(i,5))+...
                        length_veh(vehicle_array(j,5))) - ...
                        criticle_limite_intervehicle < distance_foward_max
                    distance_foward_max = Pos_j_future_v(2) - ...
                        1/2 *( length_veh(vehicle_array(i,5))+...
                        length_veh(vehicle_array(j,5))) - ...
                        criticle_limite_intervehicle;
                    Speed_foward = Speed_j_v(2) + Speed_v(2);
                end
            end
        end
end

boundaryPoints_left = zeros(2,200);
boundaryPoints_left(2,:) = 1:200;
boundaryPoints_left(2,:) = boundaryPoints_left(2,:)- ...
    vehicle_array(i,2)*ones(1,200);
boundaryPoints_left(1,:) = boundaryPoints(:,2)- vehicle_array(i,1);

boundaryPoints_right = zeros(2,200);
boundaryPoints_right(2,:) = 1:200;
boundaryPoints_right(2,:) = boundaryPoints_right(2,:) - vehicle_array(i,2);
boundaryPoints_right(1,:) = boundaryPoints(:,1)- vehicle_array(i,1);

    [~,isPositiveLeft] = find(boundaryPoints_left(1,:) > 0 );
    [~,isPositiveRight] = find(boundaryPoints_right(1,:) > 0 );
    index_left = intersect(find(boundaryPoints_left(1,isPositiveLeft) > ...
        - width_veh(vehicle_array(i,5))/2), ...
        find(boundaryPoints_left(1,isPositiveLeft) <...
        width_veh(vehicle_array(i,5))/2 ));
    index_right = intersect(find(boundaryPoints_right(1,isPositiveRight) > ...
        - width_veh(vehicle_array(i,5))/2), ...
        find(boundaryPoints_right(1,isPositiveRight) < ...
        width_veh(vehicle_array(i,5))/2 ));
    if isempty(index_left) == 0
        distance_boundary = min( boundaryPoints_left(2,index_left));
    elseif isempty(index_right) == 0
        distance_boundary = min( boundaryPoints_right(2,index_right));
    else distance_boundary = 200;
    end

    if distance_foward_max > distance_boundary
        distance_foward_max = distance_boundary;
        Speed_foward = 0;
    end

    if distance_foward_max - criticle_limite_intervehicle - 25 < 0 &&...
            distance_foward_max - criticle_limite_intervehicle > 5
        foward_flag = -1;
        Speed_future_v(2) = sqrt(2 * acc_max * distance_foward_max + ...
            Speed_foward^2 - 10);
        acc_v(2) = Speed_future_v(2) - Speed_v(2);
```

```matlab
    elseif Speed_v(2)^2 -Speed_foward^2< ...
            (distance_foward_max -2*criticle_limite_intervehicle) * 2 * ...
            acc_brake/2
        Speed_future_v(2) = sqrt(2 * acc_max * distance_foward_max +...
            Speed_foward^2);
        acc_v(2) = Speed_future_v(2) - Speed_v(2);
        if acc_v(2) > acc_max
            acc_v(2) = acc_max;
        end
        acc_v(1) = 0;
        foward_flag = 1;

        test_acc = [acc_v(1),acc_v(2),0,0,0,0];
    end

turn_flag =0;
% second part
if(foward_flag == 0 || foward_flag == -1) && (Speed_foward == 0 || ...
        distance_foward_max < 10)
    Pos_future_v = [-4;0];
    count1 = 0; count2 =0;
    for j = 1: vehicle_number
        if i ~= j && vehicle_array(j,5) > 0
            if (~isCollideSimple(Pos_future_v,vehicle_other_v(j,1:2),...
                    vehicle_array(i,5),vehicle_array(j,5),5)) &&...
                    (~isOutBoundary([(Pos_future_v(1) + vehicle_array(i,1))...
                    ;(Pos_future_v(2)+ vehicle_array(i,2) + ...
                    vehicle_array(i,3) + vehicle_array(i,3)^2/2/acc_brake+1)],...
                    vehicle_array(i,5))) && (Speed_j_v(2) >= 0)
            else
                count1 = 1; break
            end
        end
    end
    if count1 == 1
        Pos_future_v = [4;0];
        for j = 1: vehicle_number
        if i ~= j && vehicle_array(j,5) > 0
            if (~isCollideSimple(Pos_future_v,vehicle_other_v(j,1:2),...
                    vehicle_array(i,5),vehicle_array(j,5),5)) &&...
                    (~isOutBoundary([Pos_future_v(1) + vehicle_array(i,1);...
                    Pos_future_v(2)+ vehicle_array(i,2) + vehicle_array(i,3)...
                    + vehicle_array(i,3)^2/2/acc_brake + 1],vehicle_array(i,5)))...
                    && (Speed_j_v(2) >= 0)
            else
                count2 = 1; break
            end
        end
        end
    end
    if count1 == 0
        turn_flag = 1;
        if Speed_v(2) == 0
            acc_v(1) =  -4;
            acc_v(2) = 1/2 * acc_max;
        else
            acc_v(1) = -4;
            acc_v(2) = 0;
        end
    elseif count2 == 0
        turn_flag = 1;
```

```matlab
            if Speed_v(2) == 0
                acc_v(1) =  4 ;
                acc_v(2) = 1/2 * acc_max;
            else
                acc_v(1) = 4;
                acc_v(2) = 0;
            end
        else foward_flag = 0;
        end

    end

    % thrid part
    if turn_flag == 0 && foward_flag == 0
        if distance_foward_max - criticle_limite_intervehicle < 5
            acc_v(2) = -abs((Speed_foward^2 - Speed_v(2)^2)/2/...
                (distance_foward_max - criticle_limite_intervehicle));
        else acc_v(2) = -abs((Speed_foward^2 - Speed_v(2)^2)/2/...
            (distance_foward_max - criticle_limite_intervehicle - 5));
        end
        if acc_v(2) < -acc_brake
            acc_v(2) = -acc_brake;
        end
        test_acc = [0,0,0,0,0,acc_v(2)];
    end


end
```

Appendix C   Add new vehicles

Input matlab source:

```matlab
function [  ] = addNewVehicle( toll_barrier_state_line )
% all vehicles freshly passed the toll join the flow
% generate lines of vehicle array
global vehicle_array
global vehicle_number
global initial_speed
global toll_barrier_config;
D = size(toll_barrier_state_line,2);
vehicle = zeros(D, 7);
count = 0;
for i = 1:D
    if toll_barrier_state_line(i) == 1
        count = count + 1;
        vehicle(count,:) = [i * 4 - 2, 0, initial_speed,...
            toll_barrier_config(3,i), 1,0,0];
        % initial position: center of the lane
        % initial speed: initial_speed m/s
        % initial speed position: 90"
    elseif toll_barrier_state_line(i) == 2
        count = count + 1;
        vehicle(count,:) = [i * 4 - 2, 0, initial_speed,...
            toll_barrier_config(3,i), 2, 0, 0];
    elseif toll_barrier_state_line(i) == 3
        count = count + 1;
        vehicle(count,:) = [i * 4 - 2, 0, initial_speed,...
```

```
            toll_barrier_config(3,i), 3, 0, 0];
    end
end
if count > 0
        output_args = vehicle(1:count,:);
        vehicle_array(vehicle_number+1:vehicle_number +...
            count,:) = output_args;
        vehicle_number = vehicle_number+count;
end

end
```

Appendix D   Generate vehicles

Input matlab source:

```matlab
% this code contains one simulation of the traffic
B = 8; % Tollbooth number
L = 3; % Regular lane number
global test_acc
test_acc = zeros(60,6);
global toll_barrier_config;
toll_barrier_config = [3,3,3,3,3,3,3,3; 10,10,10,10,10,10,10,10; ...
    0 0 0 0 0 0 0 0];

%cell_size = 0.5; % cutting the road into small cells of 0.25 m^2

global merge_length; % Whole merge part length
merge_length = 200;
global width_veh % vehicle size config
global length_veh
width_veh = [2 3 3];
length_veh = [4 7 10];

global boundaryPoints
global vehicle_array
global vehicle_number
shapePoints = [32 0; 32 merge_length/4;32 merge_length/2;...
    32 merge_length/4*3; 32 merge_length];
boundaryPoints = zeros(merge_length,2);
% the second row presents the left boundary.
boundaryPoints(:,1) = interp1(shapePoints(:,2), ...
    shapePoints(:,1),-0.5+(1:1:merge_length),'spline');

global has_collision
has_collision = 0;
toll_barrier_state = zeros(70,B);
% track vehicle departing from the tollbooth with historical info
global small_delay
global medium_delay
global large_delay
global initial_speed

global v_max;
v_max = 15;

small_delay = 10; % delay caused by a small vehicle to pass a booth
medium_delay = 15;
```

```
large_delay = 30;
initial_speed = 5;
% line 1 for vehicle types: 1, small, 2, medium, 3, large
% line 2 for delay caused by charge mechanisms:
%10, conventional, 5, exact exchange, 2,
% electronic
flow_total = 600; % total flow
flow_instant = zeros(901,1); % number of vehicles per 15 minutes
% distribute flow into each second
for i=1:flow_total
    ind = floor(rand()*900) + 1;
    flow_instant(ind) = flow_instant(ind) + 1;
end

vehicle_array = zeros(flow_total,7);
% colomns 1, posx, 2, posy, 3, speed, 4, rad, 5 type, 6 collision, 7 is_AI
vehicle_number = 0; % the total vehicle number after the simulation start.

completion_count = 0;

for i=1:70 % one simulation per second;
    [toll_barrier_state, flow_queue] = ...
        updateTollStation(flow_total, flow_instant(i), ...
        toll_barrier_state, toll_barrier_config);
    flow_instant(i+1) = flow_queue + flow_instant(i+1);
end
```

Appendix E   Collision

Input matlab source:

```
function[collide] = isCollide(pos1,pos2,type1,type2)
% pos including [x;y;alpha] alpha is between 0 to pi (pi/2 is vertical)
global width_veh
global length_veh
%width_veh = [1, 3*sqrt(2)/2]; %only use for test
%length_veh = [3,3*sqrt(2)];
%type1 = 1; type2 = 2;
%pos1 = [4.5;1.5;-pi/2]; pos2=[5.75;4.75;-pi/4];
pos1 = pos1';
pos2 = pos2';
Trans1 = [cos(pos1(3)) -sin(pos1(3)); sin(pos1(3)) cos(pos1(3))];
points1(:,1) = Trans1 * [+ width_veh(type1)/2 ; ...
    + length_veh(type1)/2]+[pos1(1) ;pos1(2)];
points1(:,2) = Trans1 * [ - width_veh(type1)/2 ; ...
    + length_veh(type1)/2]+[pos1(1) ;pos1(2)];
points1(:,3) = Trans1 * [ - width_veh(type1)/2 ; ...
    - length_veh(type1)/2]+[pos1(1) ;pos1(2)];
points1(:,4) = Trans1 * [ + width_veh(type1)/2 ; ...
    - length_veh(type1)/2]+[pos1(1) ;pos1(2)];

Trans2 = [cos(pos2(3)) -sin(pos2(3)); sin(pos2(3)) cos(pos2(3))];
points2(:,1) = Trans2 * [width_veh(type2)/2 ; +...
    length_veh(type2)/2]+[pos2(1);pos2(2) ];
points2(:,2) = Trans2 * [- width_veh(type2)/2 ;...
    length_veh(type2)/2]+[pos2(1);pos2(2) ];
points2(:,3) = Trans2 * [- width_veh(type2)/2 ; ...
    - length_veh(type2)/2]+[pos2(1);pos2(2) ];
```

```matlab
points2(:,4) = Trans2 * [width_veh(type2)/2 ;...
    - length_veh(type2)/2]+[pos2(1);pos2(2) ];

collide = 0;
for i = 1:4
    for j = 1:4
        if i==4
            i2 = 1;
        else i2 = i+1;
        end
        if j ==4
            j2 = 1;
        else j2 = j+1;
        end

        collide = isCollideHelp(points1(:,i),...
            points1(:,i2),points2(:,j),points2(:,j2));
        if collide == 1
            %warning('collode!');
            break;
        end
    end
end
end
```

Appendix F   Boundary

Input matlab source:

```matlab
function[isOut] = isOutBoundary(pos, type)
    global boundaryPoints
    global width_veh
    isOut = 1;
    if pos(2) >=0 && pos(2) <= 199
        if pos(1) < 200 && pos(1) > 0
            if (pos(1) < boundaryPoints(floor(pos(2))+1,1) - ...
                    width_veh(type)/2)  && (pos(1) >...
                    boundaryPoints(floor(pos(2))+1,2) + width_veh(type)/2)
                isOut = 0;
            end
        end
    end

end
```

Appendix G   Toll station

Input matlab source:

```matlab
function [toll_barrier_state, flow_queue] = ...
    updateTollStation(flow_total, flow_instant, ...
    toll_barrier_state, toll_barrier_config)
global small_delay
global medium_delay
global large_delay
```

```matlab
% B for lane number, D for time stamp
B = size(toll_barrier_state,1);
D = size(toll_barrier_state,2);
critical_flow = 300;
% aging the info
i = B;
while i > 1
    for j = 1:D
        toll_barrier_state(i,j) = toll_barrier_state(i-1, j);
    end
    i = i-1;
end
for j = 1:D
    toll_barrier_state(1,j) = 0;
end

% populating newest state
if flow_total > critical_flow
    % queue before toll station, continuous vehicle flow generated
    for i = 1:D
        % respect time interval of vehicles
        for j = 1:B
            if toll_barrier_state(j,i) == 1
                if j >= small_delay + toll_barrier_config(2,i)
                    % toll ready for next vehicle
                    tmp = rand(1);
                    if toll_barrier_config(1,i) == 3
                        if tmp > 0.8 % 20% large vehicle
                            toll_barrier_state(1,i) = 3;
                        elseif tmp > 0.5 % 30% medium vehicle
                            toll_barrier_state(1,i) = 2;
                        else
                            toll_barrier_state(1,i) = 1;
                        end
                    elseif toll_barrier_config(1,i) == 2
                        if tmp > 0.625
                            toll_barrier_state(1,i) = 2;
                        else
                            toll_barrier_state(1,i) = 1;
                        end
                    elseif toll_barrier_config(1,i) == 1
                        toll_barrier_state(1,i) = 1;
                    else
                        warning('Wrong toll_barrier_config');
                    end
                else
                    break
                end % j < 30, port not ready
            elseif toll_barrier_state(j,i) == 2
                if j >= medium_delay + toll_barrier_config(2,i)
                    % toll ready for next vehicle
                    tmp = rand(1);
                    if toll_barrier_config(1,i) == 3
                        if tmp > 0.8 % 20% large vehicle
                            toll_barrier_state(1,i) = 3;
                        elseif tmp > 0.5 % large_delay% medium vehicle
                            toll_barrier_state(1,i) = 2;
                        else
                            toll_barrier_state(1,i) = 1;
                        end
                    elseif toll_barrier_config(i,1) == 2
```

```matlab
                        if tmp > 0.625
                            toll_barrier_state(1,i) = 2;
                        else
                            toll_barrier_state(1,i) = 1;
                        end
                    else
                        warning('Wrong toll_barrier_config');
                    end
                else
                    break
                end % j < 40, port not ready
            elseif toll_barrier_state(j,i) == 3
                if j >= large_delay + toll_barrier_config(2,i)
                    % toll ready for next vehicle
                    tmp = rand(1);
                    if toll_barrier_config(1,i) == 3
                        if tmp > 0.8 % 20% large vehicle
                            toll_barrier_state(1,i) = 3;
                        elseif tmp > 0.5 % 30% medium vehicle
                            toll_barrier_state(1,i) = 2;
                        else
                            toll_barrier_state(1,i) = 1;
                        end
                    else
                        warning('Wrong toll_barrier_config');
                    end
                else
                    break
                end % j < 50, port not ready
            end
        end
        if j == B % port ready
            tmp = rand(1);
            if toll_barrier_config(1,i) == 3
                if tmp > 0.8 % 20% large vehicle
                    toll_barrier_state(1,i) = 3;
                elseif tmp > 0.5 % 30% medium vehicle
                    toll_barrier_state(1,i) = 2;
                else
                    toll_barrier_state(1,i) = 1;
                end
            else
                warning('Wrong toll_barrier_config');
            end
        end
    end
    flow_queue = 0;
else
    % toll station is idle
    count = 0;
    if flow_instant ~= 0
        % allocate tolls for vehicles
        tolls_available_ind = zeros(B);
        for i = 1:D
            % respect time interval of vehicles
            for j = 1:B
                if toll_barrier_state(j,i) == 1
                    if j >= small_delay + toll_barrier_config(2,i)
                        % toll ready for next vehicle
                        count = count + 1;
                        tolls_available_ind(count) = i;
```

```matlab
                else
                    break
                end % j < large_delay, port not ready
            elseif toll_barrier_state(j,i) == 2
                if j >= medium_delay + toll_barrier_config(2,i)
                    % toll ready for next vehicle
                    count = count + 1;
                    tolls_available_ind(count) = i;
                else
                    break
                end % j < 40, port not ready
            elseif toll_barrier_state(j,i) == 3
                if j >= large_delay + toll_barrier_config(2,i)
                    % toll ready for next vehicle
                    count = count + 1;
                    tolls_available_ind(count) = i;
                else
                    break
                end % j < 50, port not ready
            end
        end
        if j == B % port ready
            % toll ready for next vehicle
            count = count + 1;
            tolls_available_ind(count) = i;
        end
    end % traversing D and B
    for i=1:min(count,flow_instant)
        tmp = rand(1);
        if toll_barrier_config(1,tolls_available_ind(i)) == 3
            if tmp > 0.8 % 20% large vehicle
                toll_barrier_state(1, tolls_available_ind(i)) = 3;
            elseif tmp > 0.5 % 30% medium vehicle
                toll_barrier_state(1, tolls_available_ind(i)) = 2;
            else
                toll_barrier_state(1, tolls_available_ind(i)) = 1;
            end
        elseif toll_barrier_config(1,tolls_available_ind(i)) == 2
            if tmp > 0.625
                toll_barrier_state(1, tolls_available_ind(i)) = 2;
            else
                toll_barrier_state(1, tolls_available_ind(i)) = 1;
            end
        elseif toll_barrier_config(1,tolls_available_ind(i)) == 1
            toll_barrier_state(1, tolls_available_ind(i)) = 1;
        else
            warning('Wrong toll_barrier_config');
        end
    end
    flow_queue = max(flow_instant - count, 0);
else % flow_instant ~= 0
    flow_queue = 0;
end

end

end
```

Appendix H   Visualisation

```matlab
filename = 'traffic_400.mat';
time_now = yyyymmdd(datetime('now'));
[hh,mm,~] = hms(datetime('now'));
filename = ['../../figure/',filename,num2str(time_now),'_',...
    num2str(hh),'_',num2str(mm)];
if exist(filename,'dir')==0
    mkdir(filename);
end

for loop = 1:28
    set(gcf,'Visible','off');

    subplot(6,2,1);
    plot(all_info_matrice(:,loop,1));
    title('posx');

    subplot(6,2,2);
    plot(all_info_matrice(:,loop,2));
    title('posy');

    subplot(6,2,3);
    plot(all_info_matrice(:,loop,3));
    title('speed');

    subplot(6,2,4);
    plot(all_info_matrice(:,loop,4));
    title('angle');

    subplot(6,2,5);
    plot(all_info_matrice(:,loop,7));
    title('acc_(inter) x');

    subplot(6,2,6);
    plot(all_info_matrice(:,loop,8));
    title('acc_(inter) y');

    subplot(6,2,7);
    plot(all_info_matrice(:,loop,9));
    title('acc_(road) x');

    subplot(6,2,8);
    plot(all_info_matrice(:,loop,10));
    title('acc_(road) y');

    subplot(6,2,9);
    plot(all_info_matrice(:,loop,11));
    title('acc_(will) x');

    subplot(6,2,10);
    plot(all_info_matrice(:,loop,12));
    title('acc_(will) y');

    subplot(6,2,11);
    plot(all_info_matrice(:,loop,5));
    title('type');
```

```matlab
    subplot(6,2,12);
    plot(all_info_matrice(:,loop,6));
    title('AI');

    print ([filename, '/', num2str(loop), '.png'],'-dpng');
end
```