Software Design Document for 4 way stop Simulator

Interactive Planning
UBER ATG

Introduction:

4-way stop simulator should be able to emulate the 4-way stop sign scenario. There should be a number of configuration cars between 1- 4 on the road. They should follow the first come first go approach in case they arrive at the same time.

Goal:

System should be able to feed the gaussian generated parameters e.g mean, std. dev of the speed, acceleration of each car. It should generate the json file with frame, sdv and traffic information. It should also generate the images in a configurable folder IP_IMAGES with following

a) Reference AV mask image
b) Reference traffic mask image
c) Lane images with path, intersection and stop line.

Customer of simulator:

a) Flow based neural network
b) PnP based neural network

Mode of simulator:

a) Normal Mode: Simulator just prints the gaussian parameters
b) Debug Mode : Simulator prints lots of debugging information
c) Testing Mode: Simulator should able to localize the reference frame

Functional Specification:

Main :

Generate car objects with sim objects
Generate path objects

Each generated frame should contain the following:

(i) car objects
(ii) path objects
(iii) intersection objects
(iv) stop_line objects

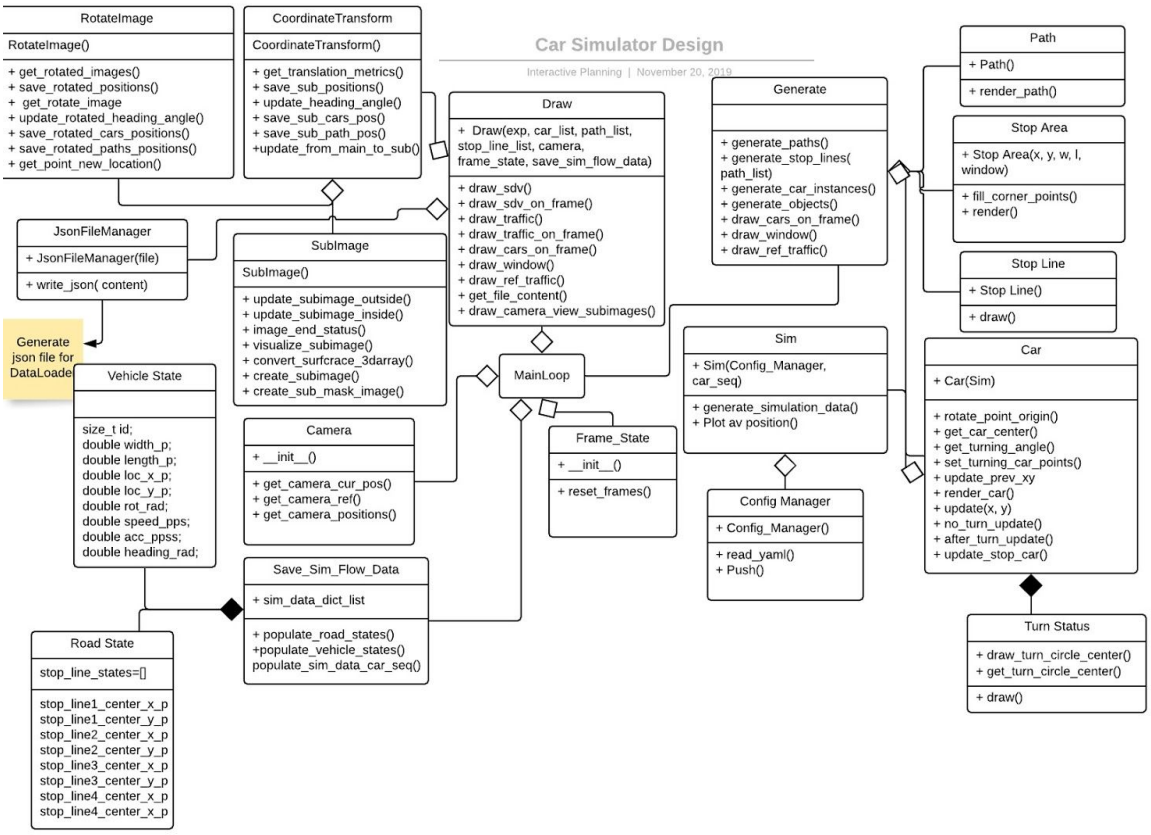
a) Car should be generated by the random number between 1 - 4
b) Car should not collide at the intersection

Technical Specification:

Technologies Used:

openCV, python, pygame, numpy, computation photography, coordinate transformations for turns

(a) High Level Design: (Class Diagram)



(b) Low Level Design:

Stop_and_go_sim

Generate sim data based on different phase of the car

a) self.update_cruise_before() : Configurable cruise distance when car starts
b) self.update_decel_before_stop() : Car's configurable deceleration before the stop line
c) self.update_stopped_time() : Car's time to stop at stop line
d) self.update_accel_after_stop() : Car's acceleration after the stop line

e) self.update_cruise_after() : Car's constant velocity after the acceleration
f) self.update_past_sim() : Car's constant velocity after the main window

Stop_and_go_main

Rule to pass through the intersection

```
For each car:
  If car has reached the end
    Increase the car's time_index
    Continue
  Car.collide = car's intersection_collision_detection()
  Get car's current time_index
 If ( car's current time_index in sim_motion_dict):
  Save car's speed and acceleration
 If ((car is at stop_line) & (car's mode 'decel'))
  Increase the car's time_index
 If ( car is at stop line & stop_timer < time_stopped)
  Increase the car's time_index
 else
  Check to clear the intersection
```

Assign each car's priority based on longest waiting time

```
Inside the Main Loop:
 If quit event :close the loop
 Fill the window with black color
 Check intersection collision
 Based on the lock status, car_list, path_list update_clear_stop_zone
 Check if any car is colliding with the intersection
 If all the car has reached the end:
   Reset the cars and other parameters
Validate the last time key for all the cars
 If ( has the number of valid frame ):
   Populate the data and get the draw status
   If (draw_status)
     Draw the objects on the frames
Else:
  Don't reset the frameno
```

stop_and_go_data

a) Open the json file in append mode
b) Dump the file content into json file

Populate_sim_data_car_seq

Create new sim data after cars have moved through the intersection. This makes data deterministic.

Only store the frames from start to end that is configured. Discard the frames before the start_frames and

Populate the vehicle states and road states (stop line)

If all the frames are less than the configured frames then discard the draw functions.

Stop_and_go_view

Camera:

If camera has preconfigured positions, that's the camera's position

FIXED_FRAME_CAMERA_VIEW

Get the camera's position based on the reference car's position at reference time.

Frame:

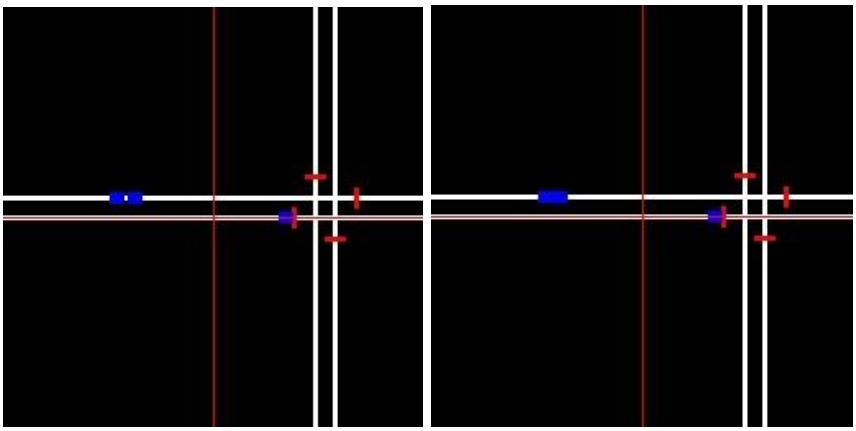
self.start_frame = start_frame // Start frame number
self.end_frame = start_frame + frame_span // End frame wrt frame span
self.ref_frame = start_frame + ref_frame // Reference Frame wrt start frame

(c) Scenario Design: True Vehicle Collision

Requirement :

a) There shall be no collisions between any vehicles in the simulation
b) Speed of the following vehicle is limited by the speed of the leading vehicle, in case of speed of trailing vehicle is more than the leading vehicle.
c) Distance gap between vehicles shall be configured. Trailing vehicles should always maintain the distance gap from the leading vehicle <= configurable distance.

From the frames below, we can see there is a possibility of collision between cars and they have collided over time.



Design Thoughts:

Trailing vehicle & LeadingVehicles : Vehicles are on the same path and have the same heading direction. Based on the vehicle's x,y position we decide the leading and trailing vehicles.

Check the distance between lead and trail vehicles. If it is > d (configured distance) it has no scope of collision yet.

If it is <= d (configured distance) it has scope of collision if the trailing vehicle's speed is greater than the leading vehicle's speed. In that particular case, we set the trailing vehicle's speed = leading vehicle's speed. It should maintain <= configured distance from the leading vehicle.

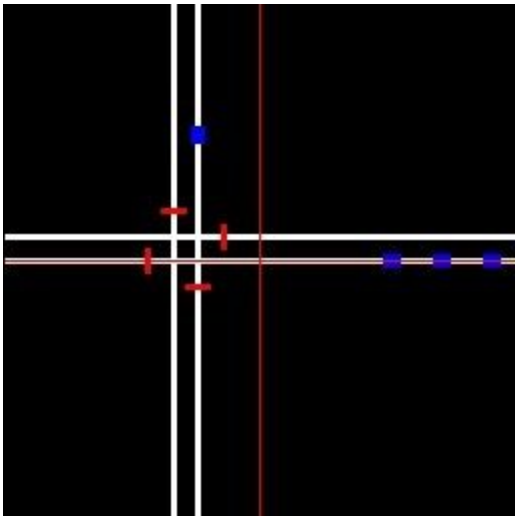
Pseudo Code:

```
for each_pair_car in car_list:
    Check if the cars have crossed the intersection
        Check the heading direction of the cars
            for the same heading direction check the distance between cars
                if (trailing_car.distance <= configured_distance)
                    If (trailing car's speed > leading car's speed)
                            set the trailing car's speed = leading car's speed
```

Verification/Testing:

Set TESTING = True in global parameters

In testing mode, We manually set the turns of the car in the same direction. So, After turn minimum 2 cars are heading in the same direction. Generating the 10 sim data points gets this scenario where trailing car's speed > heading car's speed. To check the log, sg.debug = DEBUG_EVEL_1.

E.g. 

From the above image:

If car_seq_3 : given turn = right
car_seq_1 : given turn = left
car_seq_4 : given_turn = no

All 3 cars will be heading to the north direction. Similarly we can check for all directions.

Steps to generate the testing datasets

a) Set the following parameters in stop_and_go_globals.py

TRUE_VEHICLE_COLLISION_CHECK = True
COMPLETE_TRAFFIC = True
TESTING = True

b) Generate the 10 data points by setting

TOTAL_DATA_POINTS = 10

c) Run the script

python stop_and_go_main.py

d) It will generate Image directory and create video by using

```
ffmpeg -r 30 -i stop_0000X_sub_%06d.jpg -s 256x256 -vcodec libx264 stop-and-go_sub_X.mp4
```

One of the videos out of 10 will have a clear example.

e) To check if the code is working unset the following parameter

TRUE_VEHICLE_COLLISION_CHECK = False

One out of 10 videos will have a clear collision.

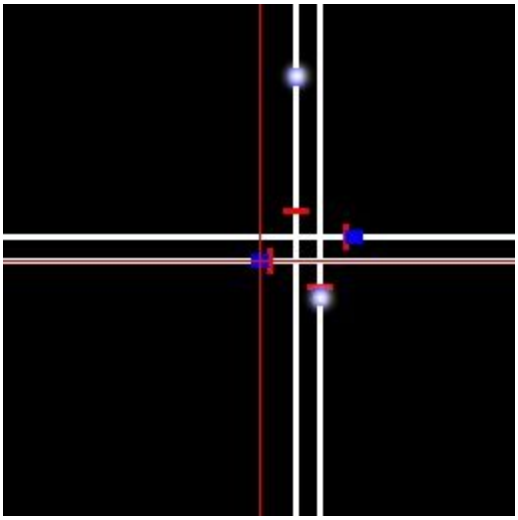
(d) Scenario Design : Natural Vehicle Simulation for 4-way stop interaction

Requirement :

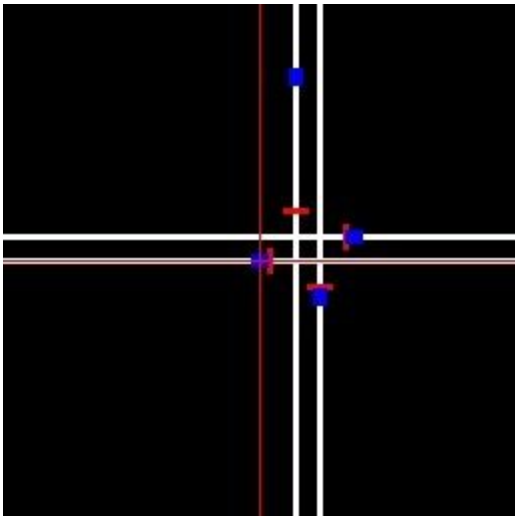
(a) Vehicles shall proceed through intersection obeying right-of-way for other actors and proceed as efficiently as possible while avoiding collision.

(i) Vehicle Intersection Case

(1) Two vehicles in opposite lanes across the intersection from one another and each wanting to proceed straight shall move through the intersection freely without waiting for one to clear the intersection.
E.g. In the below diagram, both the cars should proceed straight in the lanes without waiting for each other.



(2) Two vehicles in opposite or adjacent lanes at the intersection and each attempting to turn right shall both make right turns without waiting for one to clear the intersection.



E.g. 3 cars at the intersection should take right turns without waiting for each other.

(3) Vehicle shall never proceed when a collision is anticipated.
   a) Whenever the first vehicle is making a left turn, the other actor shall wait before proceeding.

(4) Other pertinent vehicle intersection cases.

Design Thoughts:

Intersection Rule for right of way:

To maintain the queue of cars at the intersection. Get the priority car (car is at the head of the queue) heading direction. If it is

a) Straight :
   i) Allow the opposite lane's car to take right or straight.
   ii) Allow clockwise adjacent lane's car to take right

b) Right :
   i) Allow the opposite lane's car to take a right turn.
   ii) Allow the opposite lane's car to go straight.
   iii) Allow clockwise/anti-clockwise adjacent lane's car to take a right turn.

c) Left :
   i) Allow clockwise/anti-clockwise adjacent lane's car to take a right turn.

We can have a map of each car's sequence number as an index and their opposite, clockwise adjacent, anti-clockwise adjacent lane's car's sequence number as values. At the intersection, we can check the map for the required lane's car sequence number. Based on the above rule we can allow the car to move together with the priority car.

There are some rules apply to cars at the intersection

a) Non right of way cars should keep on checking with priority cars when it is safe to start.
b) There shall not be any condition of any car stopping at the intersection.
c) The car has no right of way and should wait until the intersection is clear enough to move. When a leading car is about to clear the intersection, it is safe to start at the intersection.

Car Matrix at the intersection

Current priority car seq no	Car at the opposite lane	Car at the clockwise adjacent lane	Car at the anti-clockwise adjacent lane
1	3	2	4
2	4	3	1
3	1	4	2
4	2	1	3

There are 2 categories of the car at any given time.

Priority_car & Non Right of way cars:

Car which is at an intersection and waits longer than the stopped time is the priority car. Priority car and right of way cars can move together. Their trajectory should never overlap.

Priority_car has following properties:

a) Car first waited more than the intersection_stopped_time is the priority car.
b) There can be > 1 priority car in queue, but only the most priority car (front of the queue) can make right_of_way cars.
c) We remove priority car based on on FCFR (First come first remove)
d) Priority cars can only be safely removed from the queue. It's because we don't only populate the right_of_way queue for the most priority car.
e) Next element in the priority_queue becomes the most priority car.

Right_of_way cars of the priority car has following properties: { Not Implemented }

a) Car other than priority_car has waited more than the stopped time
b) Car based on its turn is into right_of_way_metrics of the priority_car

Non_Right_of_way cars have the following properties:

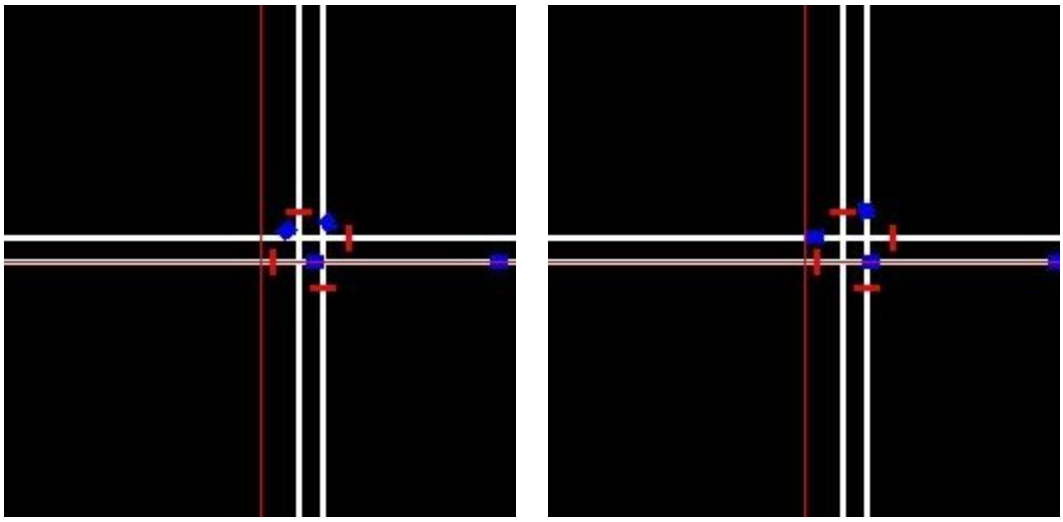
a) Car waiting at the intersection > stopped time or 2nd in priority sequence
b) It has to check with most priority cars (cars at the front of the priortiy_car queue) to make a safe movement through the intersection.

Right of way metrics at the intersection { Not Implemented }
E.g **Car Seq # 1**
3 = Opposite Lane Car
2 = Clockwise Lane Car
4 = Anti-clockwise Lane Car

Current Priority car seq no	Turn	Right of way car seq turn	No right of way car seq turn
1	straight	3: right, straight 2: right	3: left 2: left, straight 4: left, right, straight
	right	3: right 2: right 4: right, left	3: left, straight 2: left, straight 4: straight
	left	2: right 4: right	2: left, straight 4: left, straight 3: left, right, straight

Example of Priority car and right_of_way car travelling together without collision

Priority Car:
Car Seq# 3 (most right) is taking right turn
Right_of_way Cars:
Car seq#1 (most left) is taking no turn
Car seq#2 (clockwise of car seq#1) is taking right turn

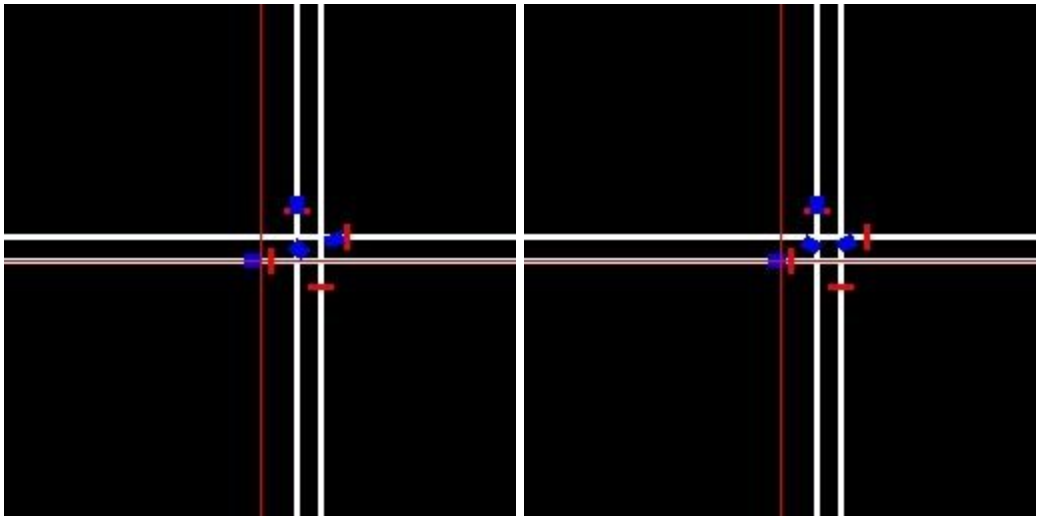


Non right of way metrics at the intersection

Not_right_of_way car seq#	**Priority_car_seq_lane_wrt_not_right_of_way_car**	**Priority_car_turns Left**	**Priority_car_turns Right**	**Priority_car_turns Straight**
Non right of way : Straight	Opposite Lane	Wait for it to cross the intersection area	X	X
	Anti-Clockwise Lane	Wait until it has crossed the center of intersection + buffer	Wait for it to cross the intersection area	Wait until it has crossed the center of intersection + buffer
	Clockwise Lane	Wait for it to cross the intersection area	X	Wait for it to cross the intersection area
Non right of way : Right	Opposite Lane	Wait for it to cross the intersection area	X	X
	Anti-Clockwise Lane	X	X	X
	Clockwise Lane	X	X	Wait for it to cross the intersection area
Non right of way : Left	Opposite Lane	Wait for it to cross the intersection area	Wait for it to cross the intersection area	Wait until it has crossed the center of intersection + buffer
	Anti-Clockwise Lane	Wait until it has crossed the center of intersection + buffer	X	Wait for it to cross the intersection area

	Clockwise Lane	Wait until it has crossed the center of intersection + buffer	X	Wait for it to cross the intersection area

Priority Car seq# 4: Taking Left turn
Not_right_of_way_car_seq# 3 : Taking left turn { turns after the car#4 has safely crossed the intersection }



Note: Most priority_car may create its right_of_way car at the end of crossing the intersection. This gives the priority to the right_of_way car instead of 2nd priority car.

Future Capabilities

a) 2 way front lanes. Rightmost lane is only turning right and the 2nd lane can turn right or front.
b) Adding pedestrian & to stop the vehicles for them
c) Adding configurable cars behind the lanes.
d) Angular right turns. Car's heading direction is straight while it takes angular right turn.

(e) Configurable Parameters:

Applicable to all the actors on the frame

(i) Global parameters:

(1) DEBUG : debugging level of displaying log statements
(2) TESTING : enable the testing to visualize the image
(3) IMAGE_BASE_DIR: Directory to get the generated the images
(4) WINDOW_WIDTH_PIXELS : Width of generated image size in pixels
(5) WINDOW_HEIGHT_PIXELS : Height of generated image size in pixels
(6) CAR_WIDTH_PIXELS : Width of the car size in pixels
(7) CAR_HEIGHT_PIXELS : Height of the car size in pixels
(8) SUB_IMAGE_WIDTH : camera view generated image width in pixels
(9) SUB_IMAGE_HEIGHT : Camera view generated image height in pixels
(10) SPRITE_AREA : Draw the mid way intersection
(11) REFERENCE_CAR_SEQ : car with camera.
(12) LANE_BUFFER_PIXELS : Path width
(13) PATH_LINE_WIDTH : Width of the path in pixels
(14) STOP_LINE_LEN_OFFSET : Offset from the middle of the image
(15) STOP_LINE_HOR_PIXELS : stop line horizontal pixels
(16) STOP_LINE_WIDTH : Width of the stop line in pixels
(17) SPRITE_MID_LEN_OFFSET : Offset of the mid area from the middle of the image
(18) DATASET_SPAN_FRAMES : Number of frames per simulation
(19) DATASET_REF_FRAMES : Referenced frame number
(20) DATASET_START_FRAMES : Starting frame per simulation
(21) COMPLETE_TRAFFIC : To generate the complete traffic
(22) GENERATE_SUBIMAGE : To generate the camera view subimages

(ii) Configurable Parameters:

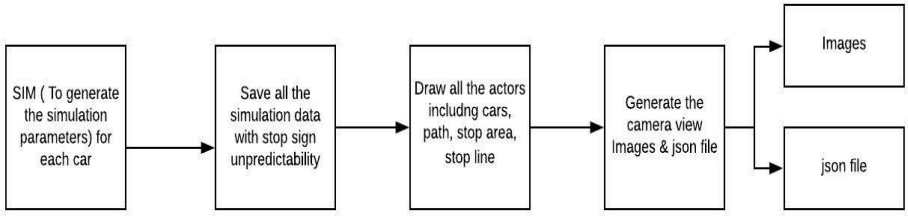
Applicable to the specific object on the frame

For each car's gaussian parameters:

(1) dist_before_stop: Number of pixels before the stop sign
(2) dist_after_stop: Number of pixels after the stop sign
(3) accl_after_stop_mean: mean acceleration of the car
(4) accl_after_stop_dev: std acceleration of the car
(5) decl_before_stop_mean: mean decel before the stop sign
(6) decl_before_stop_dev: mean std decel before the stop sign
(7) speed_before_stop_mean: mean speed before the stop sign
(8) speed_before_stop_dev: std speed before the stop sign
(9) speed_after_stop_mean: mean speed after the stop sign
(10) speed_after_stop_dev: std speed after the stop sign

(f) Data Generation:

Data Flow in the simulator:



In simulator data flows in 3 steps

a) Sim generates the parameters of each car
b) Save the sim data including stop sign unpredictability
c) Generate the images and json files for those parameters

Input Data :

Mean and std of Gaussian generated parameters from the reference document:
https://docs.google.com/spreadsheets/d/1A7xSu2Pn7Yu6H77T10eblTQKMHqrSmF4lwswlY3JsqU/edit#gid=0

Each car should have gaussian generated parameters based on stop_and_go_config.yml

Output Data :

It should generates following images

(a) Mask images for sdv
(b) Mask images for traffic
(c) Mask images for lanes and intersection

Json file should have following parameters

(a) Frame_no : frame number /simulation
(b) Num_actors : Number of cars other than the reference car
(c) Pix_per_m : Pixel / meter
(d) Ref_frame_no : Frame around which sdv should always point in x-direction
(e) Ref_state : Parameters of the reference car
    (i) acc_ppss : acceleration of the reference car in pixel/s^2
    (ii) Heading_rad : heading angle of the sdv wrt to the reference frame
    (iii) Length_p : Length of the sdv in pixel
    (iv) Loc_x_p : x coordinate of the sdv in pixel in inverse-y coordinate frame
    (v) Loc_y_p : y coordinate of the sdv in pixel in inverse-y coordinate frame
    (vi) Speed_pps : speed of the sdv in pixel/second
    (vii) Width_p : width of the sdv in pixels
(f) Seq_no : Sequence number of the frame
(g) Sim_name: Sequence number of the frame
(h) Stop_signs
    (i) Loc_x_p : x coordinate of the stop line in pixel in inverse-y coordinate frame
    (ii) Loc_y_p : y coordinate of the stop line in pixel in inverse-y coordinate frame
(i) Traffic: Based on the num_actors we can have following parameters / traffic car
    (i) Acc_ppss : acceleration of the sdv in pixel/second^2
    (ii) Heading_rad : heading angle of the traffic car wrt reference car at reference time
    (iii) Length_p : Length of the traffic car in pixels
    (iv) Loc_x_p : x coordinate of the traffic car in pixel in inverse-y coordinate frame
    (v) Loc_y_p : y coordinate of the sdv in pixel in inverse-y coordinate frame

(vi) Speed_pps : speed of the sdv in pixel/second
(vii) Width_p : width of the sdv in pixels

Software Organization :

(a) Stop_and_go_main.py : Mainloop to generate the images and start the pygame to start the 4 way traffic
(b) Stop_and_go_sim.py : Generate the simulation data for each car
(c) Stop_and_go_data.py : Provide file interface to write the json file
Populate the simulation data after each car crosses the intersection
(d) Stop_and_go_view.py : camera view and Frame State information
(e) Stop_and_go_actors.py : It should have actors
Stop_Area
Stop_Line
Rotation
Turn_Status
Car
Path
(f) Stop_and_go_config.yml : contains the configuration parameter of each car
(g) Stop_and_go_globals.py : Contains the global parameters for the Stop-n_go

User Interaction:

Commands to run under directory car_simulator

python stop_and_go_main.py

It should generate the following files. Images will be generated under IP_IMAGES directory.

a) stop_simno_SUB_expno.jpg : is the original 128 X 128 image
b) stop_simno_LABEL_expno.jpg : is the grey image of all AVs of size 128 X 128
c) stop_simno_LANES_expno.jpg : is the only path and stop sign image of size 128 X 128
d) test.json : Description about AV and SDV

To capture command output:
python stop_and_go_main.py > /tmp/file 2>&1

Run the command in the debugging mode:
Update the stop_and_go_global.py

Set following True

DEBUG = True

It should print lots of logging statements.

To change the parameters :

There are 2 files to be changed. All the global variables applied to all the actors in the simulation in stop_and_go_global.py

All the configuration related changes in the stop_and_go_config.yml
This file contains the gaussian parameters of the AV

Current Priority car seq no	Turn	Right of way car seq turn	No right of way car seq turn
1	straight	3: right, straight 2: right	3: left 2: left, straight 4: left, right, straight
	right	3: right 2: right 4: right, **left**	3: left, **straight** 2: left, straight 4: straight
	left	2: right 4: right	2: left, straight 4: left, straight 3: left, right, straight
2	straight	4: right, straight 3: right	4: left 3: left, straight 1: left, right, straight
	right	4: right, straight 3: right 1: right, straight	4: left 3: left, straight 1: left
	left	3: right 1: right	3: left, straight 1: left, straight 4: left, right, straight
3	straight	1: right, straight 4: right	1: left 4: left, straight 2: left, right, straight
	right	1: right, straight 4: right 2: right, straight	1: left 4: left, straight 2: left
	left	4: right 2: right	4: left, straight 2: left, straight 1: left, right, straight
4	straight	2: right, straight 1: right	2: left 1: left, straight 3: left, right, straight
	right	2: right, straight 1: right 3: right, straight	2: left 1: left, straight 3: left
	left	1: right	1: left, straight

		3: right	3: left, straight 2: left, right, straight