function [allData, scenario, sensors] = drivingScenario()

%drivingScenario - Returns sensor detections

% allData = drivingScenario returns sensor detections in a structure

% with time for an internally defined scenario and sensor suite.

%

% [allData, scenario, sensors] = drivingScenario optionally returns

% the drivingScenario and detection generator objects.

% Generated by MATLAB(R) 9.7 (R2019b) and Automated Driving Toolbox 3.0 (R2019b).

% Generated on: 09-Mar-2020 21:09:57

% Create the drivingScenario object and ego car

[scenario, egoVehicle] = createDrivingScenario;

% Create all the sensors

[sensors, numSensors] = createSensors(scenario);

allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, 'LaneDetections', {});

running = true;

while running

% Generate the target poses of all actors relative to the ego vehicle

poses = targetPoses(egoVehicle);

time = scenario.SimulationTime;

objectDetections = {};

laneDetections = [];

isValidTime = false(1, numSensors);

isValidLaneTime = false(1, numSensors);

% Generate detections for each sensor

for sensorIndex = 1:numSensors

sensor = sensors{sensorIndex};

% Generate the boundaries of all lanes relative to the ego car

lanes = laneBoundaries(egoVehicle, 'XDistance', linspace(0, sensor.MaxRange, 100));

type = getDetectorOutput(sensor);

if strcmp(type, 'Objects only')

[objectDets, numObjects, isValidTime(sensorIndex)] = sensor(poses, time);

objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>

elseif strcmp(type, 'Lanes only')

[laneDets, ~, isValidTime(sensorIndex)] = sensor(lanes, time);

laneDetections = [laneDetections laneDets]; %#ok<AGROW>

elseif strcmp(type, 'Lanes and objects')

[objectDets, numObjects, isValidTime(sensorIndex), laneDets, ~, isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);

objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>

laneDetections = [laneDetections laneDets]; %#ok<AGROW>

elseif strcmp(type, 'Lanes with occlusion')

[laneDets, ~, isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);

laneDetections = [laneDetections laneDets]; %#ok<AGROW>

end

end

% Aggregate all detections into a structure for later use

if any(isValidTime) || any(isValidLaneTime)

allData(end + 1) = struct( ...

'Time', scenario.SimulationTime, ...

'ActorPoses', actorPoses(scenario), ...

'ObjectDetections', {objectDetections}, ...

'LaneDetections', {laneDetections}); %#ok<AGROW>

end

% Advance the scenario one time step and exit the loop if the scenario is complete

running = advance(scenario);

end

% Restart the driving scenario to return the actors to their initial positions.

restart(scenario);

% Release all the sensor objects so they can be used again.

for sensorIndex = 1:numSensors

release(sensors{sensorIndex});

end

%%%%%%%%%%%%%%%%%%%%

% Helper functions %

%%%%%%%%%%%%%%%%%%%%

% Units used in createSensors and createDrivingScenario

% Distance/Position - meters

% Speed - meters/second

% Angles - degrees

% RCS Pattern - dBsm

function [sensors, numSensors] = createSensors(scenario)

% createSensors Returns all sensor objects to generate detections

% Assign into each sensor the physical and radar profiles for all actors

profiles = actorProfiles(scenario);

sensors{1} = radarDetectionGenerator('SensorIndex', 1, ...

'SensorLocation', [2.8 0.9], ...

'Yaw', 47.2457425658951, ...

'MaxRange', 50, ...

'FieldOfView', [90 5], ...

'ActorProfiles', profiles);

sensors{2} = radarDetectionGenerator('SensorIndex', 2, ...

'SensorLocation', [2.8 -0.9], ...

'Yaw', -44.1574757392596, ...

'MaxRange', 50, ...

'FieldOfView', [90 5], ...

'ActorProfiles', profiles);

sensors{3} = visionDetectionGenerator('SensorIndex', 3, ...

'SensorLocation', [1.9 0], ...

'DetectorOutput', 'Lanes and objects', ...

'ActorProfiles', profiles);

numSensors = 3;

function [scenario, egoVehicle] = createDrivingScenario

% createDrivingScenario Returns the drivingScenario defined in the Designer

% Construct a drivingScenario object.

scenario = drivingScenario;

% Add all road segments

roadCenters = [45.2 19 0;

36.5 -6.2 0;

8.7 -16.2 0];

laneSpecification = lanespec(3, 'Width', 1.95);

road(scenario, roadCenters, 'Lanes', laneSpecification);

roadCenters = [19.7 19.1 0;

31 2.2 0;

33.8 -3.6 0;

46.1 -17.4 0];

laneSpecification = lanespec(2, 'Width', 2.925);

road(scenario, roadCenters, 'Lanes', laneSpecification);

% Add the ego vehicle

egoVehicle = vehicle(scenario, ...

'ClassID', 1, ...

'Position', [47.3 17.4 0]);

waypoints = [47.3 17.4 0;

45.4 7.7 0;

43.6 1.9 0;

41.2 -2.9 0;

37.7 -7.7 0;

32.6 -11.1 0;

28.4 -13.8 0;

23.9 -15.3 0;

19.4 -16.7 0;

14.9 -17.4 0;

10.5 -17.8 0;

10.5 -17.8 0;

10.5 -17.8 0];

speed = 40;

trajectory(egoVehicle, waypoints, speed);

% Add the non-ego actors

car1 = vehicle(scenario, ...

'ClassID', 1, ...

'Position', [9.9 -14.4 0]);

waypoints = [9.9 -14.4 0;

19.1 -12.6 0;

25.6 -11 0;

30.7 -8.6 0;

36.3 -4.2 0;

39.4 1 0;

41.6 7.7 0;

42.9 14.6 0;

43.5 18.4 0];

speed = 40;

trajectory(car1, waypoints, speed);

truck = vehicle(scenario, ...

'ClassID', 2, ...

'Length', 8.2, ...

'Width', 2.5, ...

'Height', 3.5, ...

'Position', [22.2 18.9 0]);

waypoints = [22.2 18.9 0;

30.3 6.7 0;

35.3 -2.2 0;

41.4 -10.9 0;

46.1 -15.5 0;

46.1 -15.5 0;

46.1 -15.5 0];

speed = 40;

trajectory(truck, waypoints, speed);

bicycle = actor(scenario, ...

'ClassID', 3, ...

'Length', 1.7, ...

'Width', 0.45, ...

'Height', 1.7, ...

'Position', [19.4 17 0]);

waypoints = [19.4 17 0;

24.6 10.1 0;

29.4 1.9 0;

33.6 -5.9 0;

38.7 -12.5 0;

43.7 -16.7 0];

speed = 5;

trajectory(bicycle, waypoints, speed);

actor(scenario, ...

'ClassID', 4, ...

'Length', 0.24, ...

'Width', 0.45, ...

'Height', 1.7, ...

'Position', [27.2 -12.1 0], ...

'RCSPattern', [-8 -8;-8 -8]);

actor(scenario, ...

'ClassID', 5, ...

'Length', 2.4, ...

'Width', 0.76, ...

'Height', 0.8, ...

'Position', [35.4 -6.7 0]);

function output = getDetectorOutput(sensor)

if isa(sensor, 'visionDetectionGenerator')

output = sensor.DetectorOutput;

else

output = 'Objects only';

end