Avoidance in Real Scenarios - Test Specification Report

Gianvincenzo Daddabbo, Gaetano Gallo, Alberto Ruggeri, Martina Tedesco, Alessandro Toschi

10-Jun-2021 19:16:51

Table of Contents

1. Avoidance_real_scenario 2
    1.1. Avoidance in real scenario (Hyundai Azera) 2
        1.1.1. A14 Highway 2
        1.1.2. Puglia 5
        1.1.3. Indianapolis Motor Speedway 8
    1.2. Avoidance in real scenario (BMW 325i) 11
        1.2.1. A14 Highway 11
        1.2.2. Puglia 13
        1.2.3. Indianapolis Motor Speedway 16
    1.3. Avoidance in real scenario (Ford E150) 19
        1.3.1. A14 Highway 19
        1.3.2. Puglia 22
        1.3.3. Indianapolis Motor Speedway 25
    1.4. Avoidance in real scenario (Suzuki Samurai) 28
        1.4.1. A14 Highway 28
        1.4.2. Puglia 31
        1.4.3. Indianapolis Motor Speedway 34
    1.5. Avoidance in real scenario (Volkswagen Beetle) 37
        1.5.1. A14 Highway 37
        1.5.2. Puglia 40
        1.5.3. Indianapolis Motor Speedway 43

1. Avoidance_real_scenario

Test Details

Description	This report describes the tests performed for the Obstacle Avoidance regarding two dynamic obstacles moving at a lower speed respect to the ego-vehicle and a set of static obstacles considering three real scenarios.

1.1. Avoidance in real scenario (Hyundai Azera)

1.1.1. A14 Highway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(1);

%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('A_14.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
```

```
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
          -8000 3500 0];


% DYNAMIC

VObs = 10/3.6;                % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
                repmat(VObs,length(T),1)];
```

```
VObs2 = 20/3.6;                % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(lateral_dev < 1) must be true must be true	

1.1.2. Puglia

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(1);

%% Set Speed
V = 40/3.6;

%% Scenario Loading
map = ScenarioLoading('puglia.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
```

```
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.91);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
          4000 6000 0];


% DYNAMIC

VObs = 10/3.6;                % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
              repmat(VObs,length(T),1)];

VObs2 = 20/3.6;               % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
```

```
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	

Enabled	Name	Definition	Requirements
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(lateral_dev < 1) must be true must be true	

1.1.3. Indianapolis Motor Speedway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(1);

%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('indianapolis.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
```

```
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
          400 -400 0];


% DYNAMIC

VObs = 10/3.6;                % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.5*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
              repmat(VObs,length(T),1)];

VObs2 = 20/3.6;               % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.28*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
```

```
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.2. Avoidance in real scenario (BMW 325i)

1.2.1. A14 Highway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(2);

%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('A_14.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
```

```
        0
        0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
         -8000 3500 0];


% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
             repmat(VObs,length(T),1)];

VObs2 = 20/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
```

```
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.2.2. Puglia

PostLoad Callback

%% Vehicle Parameters

```
param = loadParameters(2);
```

```
%% Set Speed
V = 40/3.6;

%% Scenario Loading
map = ScenarioLoading('puglia.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.91);
```

```
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
        4000 6000 0];


% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.2.3. Indianapolis Motor Speedway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(2);

%% Set Speed
```

```
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('indianapolis.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           400 -400 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.5*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.28*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.3. Avoidance in real scenario (Ford E150)

1.3.1. A14 Highway

PostLoad Callback

%% Vehicle Parameters

param = loadParameters(3);

```
%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('A_14.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
```

```
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
        -8000 3500 0];


% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.3.2. Puglia

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(3);

%% Set Speed
```

```
V = 40/3.6;

%% Scenario Loading
map = ScenarioLoading('puglia.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.93);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           4000 6000 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.20*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.3.3. Indianapolis Motor Speedway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(3);

%% Set Speed
```

```
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('indianapolis.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           400 -400 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.5*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.28*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.4. Avoidance in real scenario (Suzuki Samurai)

1.4.1. A14 Highway

PostLoad Callback

%% Vehicle Parameters

param = loadParameters(4);

```
%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('A_14.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
```

```
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
        -8000 3500 0];


% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.4.2. Puglia

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(4);

%% Set Speed
```

```
V = 40/3.6;

%% Scenario Loading
map = ScenarioLoading('puglia.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.91);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           4000 6000 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.4.3. Indianapolis Motor Speedway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(4);

%% Set Speed
```

```
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('indianapolis.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           400 -400 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.5*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.28*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.5. Avoidance in real scenario (Volkswagen Beetle)

1.5.1. A14 Highway

PostLoad Callback

%% Vehicle Parameters

param = loadParameters(5);

```
%% Set Speed
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('A_14.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
      obst_2
      obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
         0
         0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
```

```
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
        -8000 3500 0];


% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.5.2. Puglia

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(5);

%% Set Speed
```

```
V = 40/3.6;

%% Scenario Loading
map = ScenarioLoading('puglia.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.91);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           4000 6000 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	

1.5.3. Indianapolis Motor Speedway

PostLoad Callback

```
%% Vehicle Parameters

param = loadParameters(5);

%% Set Speed
```

```
V = 100/3.6;

%% Scenario Loading
map = ScenarioLoading('indianapolis.mat');

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
% STATIC
T = 0:Ts:distance/V;
% Define points where the static obstacles are
obst_1 = round(length(extended_map)*0.075);
obst_2 = round(length(extended_map)*0.23);
obst_3 = round(length(extended_map)*0.24);

idx = [obst_1
       obst_2
       obst_3];
% Set to 0 the static obstacles speed
V_obst = [0
          0
          0];

obstacle = zeros(length(idx),3);

for k = 1:length(idx)
   obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end

spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
           400 -400 0];
```

```
% DYNAMIC

VObs = 10/3.6;              % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.5*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs,length(T),1)];

VObs2 = 20/3.6;             % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.28*length(X_rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle  trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
            repmat(VObs2,length(T),1)];
```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, ***if*** an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane assessment 2	At any point of time, At ***any*** point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe overtake assessment	At any point of time, At ***any*** point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_ dev < 3,sec) < 1) must be true must be true	
True	Lateral acceleration assessment	At any point of time, At ***any*** point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	
True	Lateral deviation assessment	At any point of time, ***if*** there are no obstacles detected: verify(duration(lateral_dev > 0.75,sec)<1) must be true must be true	
True	Maximum lateral deviation assessment	At any point of time, ***if*** there are no obstacles det ected: verify(lateral_dev < 1) must be true must be true	