Dynamic/Static Obstacle Avoidance - Test Specification Report

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

09-Jun-2021 18:44:46

Table of Contents

1. Compound_obstacle_avoidance

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 different simple scenarios at 40km/h and 100km/h.

1.1. Multiple_dynamic/static_obstacle 100km/h

1.1.1. 0° 100km/h

PostLoad Callback

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

%% Scenario Loading
map = [0 0; 5000 0];

% 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
          10000 0 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_ostacolo(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_ostacolo2(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	

1.1.2. 20° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 364]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.3. 45° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 1000]*5;

% 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_ostacolo
(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_ostacolo
(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_ostacolo2
(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, 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	

1.1.4. 70° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 2747]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.5. 90° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 0 1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.1.6. 110° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -364 1000]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.7. 135° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -1000 1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.1.8. 160° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -2747 1000]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.9. 180° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -1000 0]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.1.10. -20° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 -364]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.11. -45° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.1.12. -70° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 1000 -2747]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.13. -90° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; 0 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.1.14. -110° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -364 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.15. -135° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -1000 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.1.16. -160° 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
map = [0 0; -2747 -1000]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.1.17. 1000m curvature clockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.1.18. 500m curvature clockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.43*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_ostacolo(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_ostacolo2(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, 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	

1.1.19. 300m curvature clockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-300,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.45*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_ostacolo(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 = 0.35*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_ostacolo2(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, 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	

1.1.20. 300m curvature counterclockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(300,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.45*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_ostacolo(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 = 0.35*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_ostacolo2(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, 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	

1.1.21. 500m curvature counterclockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
```

```
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.43*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)];

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_ostacolo2(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, 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	

1.1.22. 1000m curvature counterclockwise 100km/h

PostLoad Callback

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

```
[X_rec, Y_rec, Theta_rec] = curve_generator(1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2. Multiple_dynamic/static_obstacle 40km/h

1.2.1. 0° 40km/h

PostLoad Callback

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

%% Scenario Loading
map = [0 0; 1000 0]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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	

1.2.2. 20° 40km/h

PostLoad Callback

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

```
map = [0 0; 1000 364]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, At ***any*** point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	

Enabled	Name	Definition	Requirements
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	

1.2.3. 45° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 1000 1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.2.4. 70° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 1000 2747];
```

```
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 -2000 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 = 0.25*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_ostacolo(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_ostacolo2(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, 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	

1.2.5. 90° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 0 1000]*5;

% 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_ostacolo(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_ostacolo
(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_ostacolo2(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_ostacolo2
(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, 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	

1.2.6. 110° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -364 1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.2.7. 135° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -1000 1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.2.8. 160° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -2747 1000];

% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
       obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 -2000 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 = 0.25*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_ostacolo(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_ostacolo2(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, 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	

1.2.9. 180° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -1000 0]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2.10. -20° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 1000 -364]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.2.11. -45° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 1000 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2.12. -70° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 1000 -2747];

% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);
```

```
idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 -2000 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 = 0.25*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_ostacolo(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_ostacolo2(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, 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	

1.2.13. -90° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; 0 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2.14. -110° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -364 -1000]*5;

% 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_ostacolo(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_ostacolo(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_ostacolo2(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, 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	

1.2.15. -135° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -1000 -1000]*5;

% 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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2.16. -160° 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
map = [0 0; -2747 -1000];

% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
    obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 -2000 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 = 0.25*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_ostacolo(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_ostacolo2(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, 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	

1.2.17. 1000m curvature clockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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_ostacolo(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_ostacolo(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_ostacolo2(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_ostacolo2(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, 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	

1.2.18. 500m curvature clockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.27*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_ostacolo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];

VObs2 = 15/3.6;               % [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 = 0.29*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_ostacolo2(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, 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	

Enabled	Name	Definition	Requirements
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	

1.2.19. 300m curvature clockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(300,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 200 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 = 0.38*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_ostacolo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];

VObs2 = 15/3.6;             % [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 = 0.25*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_ostacolo2(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, At ***any*** point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	

Enabled	Name	Definition	Requirements
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	

1.2.20. 300m curvature counterclockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(-300,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);
```

```
idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 -200 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 = 0.38*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_ostacolo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];

VObs2 = 15/3.6;               % [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 = 0.25*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_ostacolo2(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, 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	

1.2.21. 500m curvature counterclockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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)*0.15);
obst_2 = round(length(extended_map)*0.16);

idx = [obst_1
      obst_2];
% Set to 0 the static obstacles speed
V_obst = [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 = 0.27*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_ostacolo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];

VObs2 = 15/3.6;                 % [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 = 0.29*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_ostacolo2(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, 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	

1.2.22. 1000m curvature counterclockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% 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_ostacolo(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, 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	