# 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	3
1.1. Multiple_dynamic/static_obstacle 100km/h	3
<u>1.1.1. 0° 100km/h</u>	3
<u>1.1.2. 20° 100km/h</u>	5
<u>1.1.3. 45° 100km/h</u>	8
<u>1.1.4. 70° 100km/h</u>	10
<u>1.1.5. 90° 100km/h</u>	13
<u>1.1.6. 110° 100km/h</u>	
<u>1.1.7. 135° 100km/h</u>	
<u>1.1.8. 160° 100km/h</u>	
<u>1.1.9. 180° 100km/h</u>	
<u>1.1.1020° 100km/h</u>	
<u>1.1.1145° 100km/h</u>	
<u>1.1.1270° 100km/h</u>	
<u>1.1.1390° 100km/h</u>	
<u>1.1.14110° 100km/h</u>	
<u>1.1.15135° 100km/h</u>	
<u>1.1.16160° 100km/h</u>	
1.1.17. 1000m curvature clockwise 100km/h	
1.1.18. 500m curvature clockwise 100km/h	
1.1.19. 300m curvature clockwise 100km/h	
1.1.20. 300m curvature counterclockwise 100km/h	
1.1.21. 500m curvature counterclockwise 100km/h	
1.1.22. 1000m curvature counterclockwise 100km/h	
1.2. Multiple_dynamic/static_obstacle 40km/h	
<u>1.2.1. 0° 40km/h</u>	
<u>1.2.2. 20° 40km/h</u>	
<u>1.2.3. 45° 40km/h</u>	
<u>1.2.4. 70° 40km/h</u>	
<u>1.2.5. 90° 40km/h</u>	
<u>1.2.6. 110° 40km/h</u>	
<u>1.2.7. 135° 40km/h</u>	
<u>1.2.8. 160° 40km/h</u>	
<u>1.2.9. 180° 40km/h</u>	
<u>1.2.1020° 40km/h</u>	
<u>1.2.1145° 40km/h</u>	
<u>1.2.1270° 40km/h</u>	
<u>1.2.1390° 40km/h</u>	
<u>1.2.14110° 40km/h</u>	
<u>1.2.15135° 40km/h</u>	
<u>1.2.16160° 40km/h</u>	
1.2.17. 1000m curvature clockwise 40km/h	
1.2.18. 500m curvature clockwise 40km/h	98

1.2.19. 300m curvature clockwise 40km/h	100
1.2.20. 300m curvature counterclockwise 40km/h	102
1.2.21. 500m curvature counterclockwise 40km/h	
1.2.22. 1000m curvature counterclockwise 40km/h	

## 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 \text{ obst} = [0]
     0
     01;
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 ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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	Requi reme nts
True	Left lane as- sessment 1	At any point of time, <b>if</b> an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      0];
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
       -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                           % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
```

```
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.3.45° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 = \text{round}(\text{length}(\text{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 \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.5. 90° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map, VObs2, Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## $1.1.6.110^{\circ} 100 \text{km/h}$

### **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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 = \text{round}(\text{length}(\text{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 \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.7. 135° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 3];
```

```
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## $1.1.8.\,160^{\circ}\,100 km/h$

### **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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 = \text{round}(\text{length}(\text{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 \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.9. 180° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 3];
```

```
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
     0];
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
```

```
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.11. -45° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 3];
```

```
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

 $X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});$ 

```
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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 = \text{round}(\text{length}(\text{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 \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.1.13. -90° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = \text{round}(\text{length}(\text{extended map})*0.23);
obst 3 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
    obst 2
```

```
obst 31;
% Set to 0 the static obstacles speed
V \text{ 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 01;
% DYNAMIC
VObs = 10/3.6:
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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. V rec. That a rec] = reference, generator(map V Ts

[X\_rec, Y\_rec, Theta\_rec] = reference\_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

```
X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst_1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      0];
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
       -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                           % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
```

```
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.1.15. -135° 100km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});
Y \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = Y \operatorname{rec}(\operatorname{end});
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 = \text{round}(\text{length}(\text{extended map})*0.24);
idx = [obst_1]
     obst 2
     obst 3];
```

```
% Set to 0 the static obstacles speed
V obst = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                       % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

 $X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});$ 

```
Y \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = Y \operatorname{rec}(\operatorname{end});
Theta rec(end+1:end+p+20) = Theta rec(end);
% Define initial condition based on map
x0 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 = \text{round}(\text{length}(\text{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 \text{ obst} = [0]
      0
      01;
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 01;
% DYNAMIC
                           % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(duration(lateral_dev > 5 && lateral_dev < 3,sec) < 1) must be true	

Enabled	Name	Definition	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 = \text{round}(\text{length}(\text{extended map})*0.23);
obst 3 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
    obst 2
    obst 3];
% Set to 0 the static obstacles speed
V \text{ 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
                       % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 21;
% 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 01;
% DYNAMIC
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 21:
% Set to 0 the static obstacles speed
V_obst = [0]
     01;
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 01;
% DYNAMIC
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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 ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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_ostaco-lo2(spawn_idx+1:spawn_idx+length(T)));
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 21;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     01;
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_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 = 0.35\*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\_ostaco-lo2(spawn\_idx+1:spawn\_idx+length(T),1)];

### **Logical and Temporal Assessments**

#### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 21:
% Set to 0 the static obstacles speed
V_{obst} = [0]
      01;
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 01;
% DYNAMIC
                          % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% 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),1)];

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});
Y \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = Y \operatorname{rec}(\operatorname{end});
Theta rec(end+1:end+p+20) = Theta rec(end);
% Define initial condition based on map
x0 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
    obst 2
    obst 3];
% Set to 0 the static obstacles speed
V_{obst} = [0]
      0
      01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn fake = round(length(extended map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
       -8000 3500 0];
% DYNAMIC
VObs = 10/3.6:
                            % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
```

```
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec):
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map, VObs2, Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	

Enabled	Name	Definition	Requi reme nts
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+p+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn_fake = round(length(extended map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6:
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6:
% 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
```

#### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, <b>if</b> an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+p+20) = X \operatorname{rec}(\operatorname{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 = \text{round(length(extended map)*0.23)};
obst 3 = round(length(extended map)*0.24);
idx = [obst 1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      01;
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 01;
```

% DYNAMIC

```
VObs = 10/3.6:
                       % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6:
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.2.3.45° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 \text{ obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn_fake = round(length(extended map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6:
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6:
% 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
```

#### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 \text{ obst} = [0]
     01;
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
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% 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 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)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.2.5. 90° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+p+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      01;
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)];
```

spawn\_fake = round(length(extended\_map)\*0.9);

#### end

```
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
      -8000 3500 0];
% DYNAMIC
                       % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

 $X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});$ 

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 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V_{obst} = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn fake = round(length(extended map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
```

Y\_ostacolo(spawn\_idx+1:spawn\_idx+length(T)) theta\_ostaco-

```
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y_ostacolo2(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

### **Logical and Temporal Assessments**

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.2.7. 135° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+p+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      01;
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
                       % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn_idx+1:spawn_idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
                         % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

 $X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});$ 

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 21:
% Set to 0 the static obstacles speed
V_obst = [0]
     01;
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
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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 ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
                          % [m/s] Set the speed for the dynamic obstacle
VObs2 = 20/3.6;
% 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));
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 31:
% Set to 0 the static obstacles speed
V obst = [0]
     0
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn fake = round(length(extended map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6:
                          % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
```

```
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6:
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	

Enabled	Name	Definition	Requi reme nts
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.2.10. -20° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
   obst 2
   obst 31;
% Set to 0 the static obstacles speed
V \text{ 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 01;
% DYNAMIC
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      0
      0];
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended_map(idx(k),1) extended_map(idx(k),2) V_obst(k)];
end
spawn fake = round(length(extended map)*0.9);
fakeObs = [extended map(spawn fake,1) extended map(spawn fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                           % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
```

```
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	

Enabled	Name	Definition	Requi reme nts
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

# 1.2.12. -70° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+p+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 21;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     01;
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 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)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

 $X \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{end});$ 

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 = \text{round(length(extended map)*0.23)};
obst_3 = round(length(extended_map)*0.24);
idx = [obst 1]
    obst 2
    obst 31;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
VObs = 10/3.6:
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
```

dyn\_obstacle1 = [X\_ostacolo(spawn\_idx+1:spawn\_idx+length(T))
Y ostacolo(spawn\_idx+1:spawn\_idx+length(T)) theta ostaco-

```
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo2, Y_ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs2,length(T),1)];
```

### **Logical and Temporal Assessments**

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.2.14. -110° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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
     01;
obstacle = zeros(length(idx),3);
for k = 1:length(idx)
  obstacle(k,:) = [extended map(idx(k),1) extended map(idx(k),2) V obst(k)];
end
spawn_fake = round(length(extended_map)*0.9);
fakeObs = [extended_map(spawn_fake,1) extended_map(spawn_fake,2) 0
      -8000 3500 0];
% DYNAMIC
VObs = 10/3.6;
                       % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X_ostacolo, Y_ostacolo, theta_ostacolo] = reference_generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn_idx+1:spawn_idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

lo2(spawn\_idx+1:spawn\_idx+length(T))...
repmat(VObs2,length(T),1)];

### **Logical and Temporal Assessments**

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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';
eqoStates.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 \text{ obst} = [0]
     0
     01;
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 01;
% DYNAMIC
                         % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6:
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
```

```
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle1 = [X_ostacolo(spawn_idx+1:spawn_idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta_ostacolo2] = reference_generator(map,VObs2,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X rec);
spawn_idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X ostacolo2(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn_obstacle2 = [X_ostacolo2(spawn_idx+1:spawn_idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn idx+1:spawn idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

## 1.2.16. -160° 40km/h

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 = eve(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 21;
% Set to 0 the static obstacles speed
V \text{ 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 01;
% DYNAMIC
                       % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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 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)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 = \text{round(length(extended map)*0.23)};
obst_3 = round(length(extended_map)*0.24);
idx = [obst 1]
   obst 2
   obst 31;
% Set to 0 the static obstacles speed
V \text{ 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 01;
% DYNAMIC
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended_obs = [T' X_ostacolo(spawn_idx+1:spawn_idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y_ostacolo(spawn_idx+1:spawn_idx+length(T)) theta_ostaco-
lo(spawn idx+1:spawn idx+length(T))...
         repmat(VObs,length(T),1)];
```

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

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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 21;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
      01;
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

```
% [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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 ostaco-
lo(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
IX 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_ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
```

### **Logical and Temporal Assessments**

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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	Requi reme nts
True	Lateral accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ obst} = [0]
      01;
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 ostaco-
lo(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 ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 \operatorname{rec}(\operatorname{end}+1:\operatorname{end}+\operatorname{p}+20) = X \operatorname{rec}(\operatorname{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 \text{ kin} = [X \text{ rec}(1) \text{ Y rec}(1) \text{ Theta rec}(1) \text{ 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)];
eqoStates.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 21;
% Set to 0 the static obstacles speed
V \text{ obst} = [0]
     01;
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 ostaco-
lo(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_ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
```

### Assessments

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 21;
% 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 01;
% DYNAMIC
                        % [m/s] Set the speed for the dynamic obstacle
VObs = 10/3.6;
% 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 ostaco-
lo(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	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, verify(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> 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

```
%% 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 = eve(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 = \text{round(length(extended map)*0.24)};
idx = [obst 1]
    obst 2
    obst 31:
% Set to 0 the static obstacles speed
V \text{ 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 01;
% DYNAMIC
VObs = 10/3.6;
                        % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo, Y ostacolo, theta ostacolo] = reference generator(map,VObs,Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.3*length(X rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs = [T' X ostacolo(spawn idx+1:spawn idx+length(T)) Y ostaco-
lo(spawn idx+1:spawn idx+length(T)) repmat(VObs,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle1 = [X ostacolo(spawn idx+1:spawn idx+length(T))
Y ostacolo(spawn idx+1:spawn idx+length(T)) theta ostaco-
```

```
lo(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs,length(T),1)];
VObs2 = 20/3.6;
                         % [m/s] Set the speed for the dynamic obstacle
% Define trajectory of the dynamic obstacle
[X ostacolo2, Y ostacolo2, theta ostacolo2] = reference generator(map, VObs2, Ts);
% Define where the dynamico obstacle is at the start of the simulation
spawn = 0.18*length(X_rec);
spawn idx = round(spawn*V/VObs);
% Define dynamic obstacle object for simulation
extended obs2 = [T' X_ostacolo2(spawn_idx+1:spawn_idx+length(T)) Y_ostaco-
lo2(spawn idx+1:spawn idx+length(T)) repmat(VObs2,length(T),1)];
% Define dynamic obstacle trajectory for plotting
dyn obstacle2 = [X ostacolo2(spawn idx+1:spawn idx+length(T))
Y ostacolo2(spawn idx+1:spawn idx+length(T)) theta ostaco-
lo2(spawn_idx+1:spawn_idx+length(T))...
         repmat(VObs2,length(T),1)];
```

Enabled	Name	Definition	Requi reme nts
True	Left lane as- sessment 1	At any point of time, At <b>any</b> point of time, if an obstacle is detected: verify(lateral_dev >= 2 && lateral_dev <= 6) must be true must be true	
True	Left lane as- sessment 2	At any point of time, At <b>any</b> point of time, veri- fy(lateral_dev < 6) must be true must be true	
True	Safe over- take assess- ment	At any point of time, At <b>any</b> 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 accel- eration as- sessment	At any point of time, At <b>any</b> point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	