

Multiple Static Obstacles Avoidance - Test Specification Report

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

23-May-2021 18:08:14

Table of Contents

1. Multiple static obstacle avoidance	3
1.1. Multiple static obstacle 100km/h	3
1.1.1. 0° 100km/h	3
1.1.2. 20° 100km/h	5
1.1.3. 45° 100km/h	7
1.1.4. 70° 100km/h	8
1.1.5. 90° 100km/h	10
1.1.6. 110° 100km/h	12
1.1.7. 135° 100km/h	14
1.1.8. 160° 100km/h	16
1.1.9. 180° 100km/h	17
1.1.10. -20° 100km/h	19
1.1.11. -45° 100km/h	21
1.1.12. -70° 100km/h	23
1.1.13. -90° 100km/h	25
1.1.14. -110° 100km/h	26
1.1.15. -135° 100km/h	28
1.1.16. -160° 100km/h	30
1.1.17. 1000m curvature clockwise 100km/h	32
1.1.18. 500m curvature clockwise 100km/h	33
1.1.19. 300m curvature clockwise 100km/h	35
1.1.20. 300m curvature counterclockwise 100km/h	37
1.1.21. 500m curvature counterclockwise 100km/h	39
1.1.22. 1000m curvature counterclockwise 100km/h	40
1.2. Multiple static obstacle 40km/h	42
1.2.1. 0° 40km/h	42
1.2.2. 20° 40km/h	44
1.2.3. 45° 40km/h	46
1.2.4. 70° 40km/h	47
1.2.5. 90° 40km/h	49
1.2.6. 110° 40km/h	51
1.2.7. 135° 40km/h	53
1.2.8. 160° 40km/h	55
1.2.9. 180° 40km/h	56
1.2.10. -20° 40km/h	58
1.2.11. -45° 40km/h	60
1.2.12. -70° 40km/h	62
1.2.13. -90° 40km/h	64
1.2.14. -110° 40km/h	65
1.2.15. -135° 40km/h	67
1.2.16. -160° 40km/h	69
1.2.17. 1000m curvature clockwise 40km/h	71
1.2.18. 500m curvature clockwise 40km/h	72

1.2.19. 300m curvature clockwise 40km/h	74
1.2.20. 300m curvature counterclockwise 40km/h	76
1.2.21. 500m curvature counterclockwise 40km/h	78
1.2.22. 1000m curvature counterclockwise 40km/h	79

1. Multiple_static_obstacle_avoidance

Test Details

Description	This report describes the tests performed for the Obstacle Avoidance regarding 5 static obstacles considering different simple scenarios at 40km/h and 100km/h.
-------------	---

1.1. Multiple_static_obstacle_100km/h

1.1.1. 0° 100km/h

PostLoad Callback

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
```

```

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

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

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.1.2. 20° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; 1000 364];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

1. Multiple_static_obstacle_avoidance

```
obst_5 = point_E;
```

```
idx_dyn = [obst_1;  
           obst_2;  
           obst_3;  
           obst_4;  
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.3. 45° 100km/h

PostLoad Callback

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);
```



```

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

```

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

obst_5 = point_E;

idx_dyn = [obst_1;

obst_2;

obst_3;

obst_4;

obst_5];

obstacle = zeros(length(idx_dyn),2);

for k = 1:length(idx_dyn)

obstacle(k,:) = [extended_map(idx_dyn(k),1) extended_map(idx_dyn(k),2)];

end

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.5. 90° 100km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 100/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; 0 1000];
```

```
% 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
```

1. Multiple_static_obstacle_avoidance

```
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
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: <code>verify(lateral_dev >= 2 && lateral_dev <= 6)</code> must be true must be true	

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

1.1.6. 110° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; -364 1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.1.7. 135° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; -1000 1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

1. Multiple_static_obstacle_avoidance

```
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.8. 160° 100km/h

PostLoad Callback

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);
```

```

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.9. 180° 100km/h

PostLoad Callback

```

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

```

```
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];
```

```
% 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
```

1. Multiple_static_obstacle_avoidance

```
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
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	

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

1.1.11. -45° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; 1000 -1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.1.12. -70° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; 1000 -2747];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

1. Multiple_static_obstacle_avoidance

```
obst_5 = point_E;
```

```
idx_dyn = [obst_1;  
           obst_2;  
           obst_3;  
           obst_4;  
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.13. -90° 100km/h

PostLoad Callback

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

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

% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];
```

```
% Evaluate total distance covered by the route on the map
```

```
distance = odometer(map);
```

```
%% Reference signal
```

```
% Upsample map based on speed and timestep
```

```
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.15. -135° 100km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 100/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; -1000 -1000];
```

```
% 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
```

1. Multiple_static_obstacle_avoidance

```
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
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: <code>verify(lateral_dev >= 2 && lateral_dev <= 6)</code> must be true must be true	

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

1.1.16. -160° 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

map = [0 0; -2747 -1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.1.17. 1000m curvature clockwise 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(-1000,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

obst_5 = point_E;

idx_dyn = [obst_1;

obst_2;

1. Multiple_static_obstacle_avoidance

```
obst_3;  
obst_4;  
obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.18. 500m curvature clockwise 100km/h

PostLoad Callback

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

```
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;

% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.19. 300m curvature clockwise 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(-300,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

```

egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Safe over-take assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.20. 300m curvature counterclockwise 100km/h

PostLoad Callback

%% Set Speed

V = 100/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(300,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

1. Multiple_static_obstacle_avoidance

```
obst_4 = point_D;
obst_5 = point_E;
```

```
idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.1.21. 500m curvature counterclockwise 100km/h

PostLoad Callback

```
%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

for k = 1:length(idx_dyn)
```



```

obstacle(k,:) = [extended_map(idx_dyn(k),1) extended_map(idx_dyn(k),2)];
end

```

Logical and Temporal Assessments

Assessments

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

1.1.22. 1000m curvature counterclockwise 100km/h

PostLoad Callback

```

%% Set Speed
V = 100/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

```

```

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	

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

1.2. Multiple_static_obstacle_40km/h

1.2.1. 0° 40km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 40/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; 1000 0];
```

```
% Evaluate total distance covered by the route on the map
```

```
distance = odometer(map);
```

```
%% Reference signal
```

```
% Upsample map based on speed and timestep
```

```
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
```

```
% Extend the reference signal to avoid index over limits
```

```
X_rec(end+1:end+p+20) = X_rec(end);
```

```
Y_rec(end+1:end+p+20) = Y_rec(end);
```

```
Theta_rec(end+1:end+p+20) = Theta_rec(end);
```

```
% Define initial condition based on map
```

```
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
```

```
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
```

```
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
```

```
egoStates.Plant = x0_kin';
```

```
egoStates.Covariance = eye(6)*1000;
```

```
% Obstacle definition
```

1. Multiple_static_obstacle_avoidance

```
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Safe over-take assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} >= 2, \text{sec}) <= 0.5)$ 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];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

```

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

```

```

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

```

```

obstacle = zeros(length(idx_dyn),2);

```

```

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.3. 45° 40km/h

PostLoad Callback

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);
```

```

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.4. 70° 40km/h

PostLoad Callback

```

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

```



```

% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.5. 90° 40km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 40/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; 0 1000];
```

```
% 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
```

1. Multiple_static_obstacle_avoidance

```
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
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: <code>verify(lateral_dev >= 2 && lateral_dev <= 6)</code> must be true must be true	

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

1.2.6. 110° 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

map = [0 0; -364 1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.2.7. 135° 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

map = [0 0; -1000 1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

1. Multiple_static_obstacle_avoidance

```
obst_5 = point_E;
```

```
idx_dyn = [obst_1;  
           obst_2;  
           obst_3;  
           obst_4;  
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.8. 160° 40km/h

PostLoad Callback

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

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);
```



```

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];

% Evaluate total distance covered by the route on the map
distance = odometer(map);
%% Reference signal
% Upsample map based on speed and timestep
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

```

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

obst_5 = point_E;

idx_dyn = [obst_1;

obst_2;

obst_3;

obst_4;

obst_5];

obstacle = zeros(length(idx_dyn),2);

for k = 1:length(idx_dyn)

obstacle(k,:) = [extended_map(idx_dyn(k),1) extended_map(idx_dyn(k),2)];

end

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.10. -20° 40km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 40/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; 1000 -364];
```

```
% 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
```

1. Multiple_static_obstacle_avoidance

```
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
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	

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

1.2.11. -45° 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

map = [0 0; 1000 -1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.2.12. -70° 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

map = [0 0; 1000 -2747];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

1. Multiple_static_obstacle_avoidance

```
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];

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

% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.14. -110° 40km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 40/3.6;
```

```
%% Scenario Loading
```

```
map = [0 0; -364 -1000];
```

```
% Evaluate total distance covered by the route on the map
```

```
distance = odometer(map);
```

```
%% Reference signal
```

```
% Upsample map based on speed and timestep
```

```
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ 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];
```

```
% Evaluate total distance covered by the route on the map
```

```
distance = odometer(map);
```

```
%% Reference signal
```

```
% Upsample map based on speed and timestep
```

```
[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);
```

```
% Extend the reference signal to avoid index over limits
```

```
X_rec(end+1:end+p+20) = X_rec(end);
```

```
Y_rec(end+1:end+p+20) = Y_rec(end);
```

```
Theta_rec(end+1:end+p+20) = Theta_rec(end);
```

```
% Define initial condition based on map
```

```

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: <code>verify(lateral_dev >= 2 && lateral_dev <= 6)</code> must be true must be true	

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

1.2.16. -160° 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

map = [0 0; -2747 -1000];

% Evaluate total distance covered by the route on the map

distance = odometer(map);

%% Reference signal

% Upsample map based on speed and timestep

[X_rec, Y_rec, Theta_rec] = reference_generator(map,V,Ts);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

```

point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Lateral acceleration assessment	At any point of time, At any point of time, verify(duration(Lateral_ acceleration >= 2,sec)<=0.5) must be true must be true	

1.2.17. 1000m curvature clockwise 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(-1000,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

obst_4 = point_D;

obst_5 = point_E;

idx_dyn = [obst_1;

obst_2;

1. Multiple_static_obstacle_avoidance

```
obst_3;  
obst_4;  
obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.18. 500m curvature clockwise 40km/h

PostLoad Callback

```
%% Set Speed
```

```
V = 40/3.6;
```

```
%% Scenario Loading
```

```
[X_rec, Y_rec, Theta_rec] = curve_generator(-500,V,Ts);
```

```
map = [X_rec Y_rec];
```

```
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;

% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.19. 300m curvature clockwise 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(-300,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

```

egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	

Enabled	Name	Definition	Requirements
True	Safe over-take assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.20. 300m curvature counterclockwise 40km/h

PostLoad Callback

%% Set Speed

V = 40/3.6;

%% Scenario Loading

[X_rec, Y_rec, Theta_rec] = curve_generator(300,V,Ts);

map = [X_rec Y_rec];

distance = odometer(map);

% Extend the reference signal to avoid index over limits

X_rec(end+1:end+p+20) = X_rec(end);

Y_rec(end+1:end+p+20) = Y_rec(end);

Theta_rec(end+1:end+p+20) = Theta_rec(end);

% Define initial condition based on map

x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';

extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];

egoStates.Plant = x0_kin';

egoStates.Covariance = eye(6)*1000;

% Obstacle definition

point_A = round(length(extended_map)*0.2);

point_B = round(length(extended_map)*0.4);

point_C = round(length(extended_map)*0.65);

point_D = round(length(extended_map)*0.68);

point_E = round(length(extended_map)*0.8);

obst_1 = point_A;

obst_2 = point_B;

obst_3 = point_C;

1. Multiple_static_obstacle_avoidance

```
obst_4 = point_D;  
obst_5 = point_E;
```

```
idx_dyn = [obst_1;  
           obst_2;  
           obst_3;  
           obst_4;  
           obst_5];
```

```
obstacle = zeros(length(idx_dyn),2);
```

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

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	
True	Left lane assessment 2	At any point of time, At any point of time, $\text{verify}(\text{lateral_dev} < 6)$ must be true must be true	
True	Safe overtake assessment	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{duration}(\text{lateral_dev} > 5 \ \&\& \ \text{lateral_dev} < 3, \text{sec}) < 1)$ must be true must be true	
True	Lateral acceleration assessment	At any point of time, At any point of time, $\text{verify}(\text{duration}(\text{Lateral_acceleration} \geq 2, \text{sec}) \leq 0.5)$ must be true must be true	

1.2.21. 500m curvature counterclockwise 40km/h

PostLoad Callback

```
%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(500,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';
x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

for k = 1:length(idx_dyn)
```

```

obstacle(k,:) = [extended_map(idx_dyn(k),1) extended_map(idx_dyn(k),2)];
end

```

Logical and Temporal Assessments

Assessments

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

1.2.22. 1000m curvature counterclockwise 40km/h

PostLoad Callback

```

%% Set Speed
V = 40/3.6;
%% Scenario Loading
[X_rec, Y_rec, Theta_rec] = curve_generator(1000,V,Ts);
map = [X_rec Y_rec];
distance = odometer(map);
% Extend the reference signal to avoid index over limits
X_rec(end+1:end+p+20) = X_rec(end);
Y_rec(end+1:end+p+20) = Y_rec(end);
Theta_rec(end+1:end+p+20) = Theta_rec(end);
% Define initial condition based on map
x0_kin = [X_rec(1) Y_rec(1) Theta_rec(1) V]';

```

```

x0_dyn = [X_rec(1) Y_rec(1) Theta_rec(1) V 0 0]';
extended_map = [X_rec Y_rec Theta_rec repmat(V,length(X_rec),1)];
egoStates.Plant = x0_kin';
egoStates.Covariance = eye(6)*1000;
% Obstacle definition
point_A = round(length(extended_map)*0.2);
point_B = round(length(extended_map)*0.4);
point_C = round(length(extended_map)*0.65);
point_D = round(length(extended_map)*0.68);
point_E = round(length(extended_map)*0.8);

obst_1 = point_A;
obst_2 = point_B;
obst_3 = point_C;
obst_4 = point_D;
obst_5 = point_E;

idx_dyn = [obst_1;
           obst_2;
           obst_3;
           obst_4;
           obst_5];

obstacle = zeros(length(idx_dyn),2);

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

```

Logical and Temporal Assessments

Assessments

Enabled	Name	Definition	Requirements
True	Left lane assessment 1	At any point of time, At any point of time, if an obstacle is detected: $\text{verify}(\text{lateral_dev} \geq 2 \ \&\& \ \text{lateral_dev} \leq 6)$ must be true must be true	

1. Multiple_static_obstacle_avoidance

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