Single Static Obstacle Avoidance - Test Specification Report

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

19-May-2021 14:13:40

Table of Contents

1. Static_obstacle_avoidance

Test Details

Releases	Current (2019b)
Description	This report describes the tests performed for the Obstacle Avoidance regarding a single static obstacle considering different sample scenarios at different reference speed.

1.1. Single_static_obstacle_50km/h

Test Details

Releases	Current (2019b)

1.1.1. 0° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.1.2. 20° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

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

1.1.3. 45° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.1.4. 70° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.1.5. 90° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 10000];
```

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

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

1.1.6. 110° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.1.7. 135° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.1.8. 160° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

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

1.1.9. 180° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 0];
```

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

1.1.10. -20° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.1.11. -45° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

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

1.1.12. -70° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.1.13. -90° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 -10000];
```

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

1.1.14. -110° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

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

1.1.15. -135° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.1.16. -160° 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.1.17. 1000m curvature clockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1200 -800];
```

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

1.1.18. 500m curvature clockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 -400];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.1.19. 300m curvature clockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 -240];
```

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

1.1.20. 300m curvature counterclockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 240];
```

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

1.1.21. 500m curvature counterclockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 400];
```

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

1.1.22. 1000m curvature counterclockwise 50km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 50/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2)
        -1200 800];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.2. Single_static_obstacle_10km/h

Test Details

Releases	Current (2019b)

1.2.1. 0° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.2.2. 20° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.2.3. 45° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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	

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

1.2.4. 70° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
```

```
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.2.5. 90° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 10000];
```

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

1.2.6. 110° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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	

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

1.2.7. 135° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
```

```
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.2.8. 160° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.2.9. 180° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 0];
```

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	

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

1.2.10. -20° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
```

```
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
            10000 -10000];
```

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

1.2.11. -45° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.2.12. -70° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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	

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

1.2.13. -90° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
```

```
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
            0 -10000];
```

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

1.2.14. -110° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.2.15. -135° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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	

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

1.2.16. -160° 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
```

```
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.2.17. 1000m curvature clockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1200 -800];
```

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

1.2.18. 500m curvature clockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 -400];
```

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

1.2.19. 300m curvature clockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 -240];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.2.20. 300m curvature counterclockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 240];
```

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

1.2.21. 500m curvature counterclockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 400];
```

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

1.2.22. 1000m curvature counterclockwise 10km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 10/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2)
        -1200 800];
```

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

1.3. Single_static_obstacle_100km/h

Test Details

Releases	Current (2019b)

1.3.1. 0° 100km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.3.2. 20° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.3.3. 45° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

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

1.3.4. 70° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.3.5. 90° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 10000];
```

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

1.3.6. 110° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

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

1.3.7. 135° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.3.8. 160° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.3.9. 180° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 0];
```

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

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

1.3.10. -20° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.3.11. -45° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.3.12. -70° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

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

1.3.13. -90° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 -10000];
```

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

1.3.14. -110° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.3.15. -135° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

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

1.3.16. -160° 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.3.17. 1000m curvature clockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1200 -800];
```

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

1.3.18. 500m curvature clockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 -400];
```

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

1.3.19. 300m curvature clockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 -240];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.3.20. 300m curvature counterclockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 240];
```

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

1.3.21. 500m curvature counterclockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 400];
```

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

1.3.22. 1000m curvature counterclockwise 100km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2)
        -1200 800];
```

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

1.4. Single_static_obstacle_40km/h

Test Details

Releases	Current (2019b)

1.4.1. 0° 40km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.4.2. 20° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.4.3. 45° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

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

1.4.4. 70° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.4.5. 90° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 10000];
```

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

1.4.6. 110° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

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

1.4.7. 135° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.4.8. 160° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.4.9. 180° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 0];
```

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

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

1.4.10. -20° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.4.11. -45° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.4.12. -70° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

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

1.4.13. -90° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 -10000];
```

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

1.4.14. -110° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.4.15. -135° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

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

1.4.16. -160° 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.4.17. 1000m curvature clockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1200 -800];
```

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

1.4.18. 500m curvature clockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 -400];
```

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

1.4.19. 300m curvature clockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 -240];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.4.20. 300m curvature counterclockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 240];
```

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

1.4.21. 500m curvature counterclockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 400];
```

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

1.4.22. 1000m curvature counterclockwise 40km/h

Test Details

Releases	Current (2019b)

PreLoad 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2)
        -1200 800];
```

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

1.5. Single_static_obstacle_30km/h

Test Details

Releases	Current (2019b)

1.5.1. 0° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.5.2. 20° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.5.3. 45° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

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

1.5.4. 70° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 10000];
```

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

1.5.5. 90° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 10000];
```

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

1.5.6. 110° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

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

1.5.7. 135° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.5.8. 160° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 10000];
```

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

1.5.9. 180° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 0];
```

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

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

1.5.10. -20° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.5.11. -45° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

1.5.12. -70° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        10000 -10000];
```

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

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

1.5.13. -90° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        0 -10000];
```

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

1.5.14. -110° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.5.15. -135° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
```

```
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

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

1.5.16. -160° 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -10000 -10000];
```

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

1.5.17. 1000m curvature clockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1200 -800];
```

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

1.5.18. 500m curvature clockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 -400];
```

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

1.5.19. 300m curvature clockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 -240];
```

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	
True	Left lane assessment 2	At any point of time, ***verify***(lateral_dev < 6) must be true	

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

1.5.20. 300m curvature counterclockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -600 240];
```

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

1.5.21. 500m curvature counterclockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2);
        -1000 400];
```

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

1.5.22. 1000m curvature counterclockwise 30km/h

Test Details

Releases	Current (2019b)

PreLoad Callback

```
%% Set Speed
V = 30/3.6;
%% 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
idx = round(length(extended_map)*0.5);
obstacle = [extended_map(idx,1) extended_map(idx,2)
        -1200 800];
```

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